Ordenadore bidezko irudigintza

Joseba Makazaga [1]
Donostiako Informatika Fakultateko irakaslea
Konputazio Zientziak eta Adimen Artifiziala Saileko kidea

Asier Lasa [2]
Donostiako Informatika Fakultateko ikaslea

[1]Helbide elektronikoa: ccpmaodj@si.ehu.es
[2]Helbide elektronikoa: asier_eu@hotmail.com

Hitzaurrea

Liburu hau idaztearen ideia Informatika Fakultatean Ordenadore bidezko Irudigintza irakasgaia hasi zenean sortu zitzaidan, gaiaren inguruan euskaraz ezer ez aurkitzean hasi bainintzen liburu baten beharra ikusten.

Irakasgaiaren ardura nire esku zegoenez, han-hemenka liburu, artikulu eta material bila jardun nuen. Gehiena, dena ez esatearren, ingelesez zegoen; gerora, gazteleraz zenbait itzulpen azaldu dira, eta material horiek erabiliz klaseak euskaraz emateko itxaropenaz, apunte-sortatxoa egiteari ekin nion. Horretan nenbilela, Donostiako Euskararako Udal Patronatuak eskainitako dirulaguntza lortzeko aukera sortu zen eta heldu egin genion aukera horri. Dirulaguntzak, itzulpenez arduratu den Asier Lasari beka ordaintzeko balio izan zigun. Horrela, apunte-sorta osatu eta esku artean daukazun liburua idatzi dugu.

Euskaraz ikasten dutenek euskaraz idatzitako liburua edukitzea izan da helburua, horrez gain, gaiaren inguruan zerbait jakin nahi duenak, beste hizkuntzetara jo beharrik ez edukitzea ere buruan eduki dugu beti; horregatik, azalpenak irudi eta adibideen bidez laguntzen saiatu gara, azalpenetan errazenetik hasi eta zailagora joanez, gehiegi sakondu nahi ez duenak astunegia dela esan ez dezan.

Gaiaren inguruko terminologia gehiena ingelesetik datorkigu. Gainera, askotan asmatutako hitzekin eta akronimoekin egin dugu topo; horrelakoetan, euskaratzeko aukera ikusi badugu, euskaratu egin dugu, baina zenbaitetan ingelesezko akronimoa ere erabili dugu. Gaiaren inguruan euskaraz gauza gehiegirik ez dago; terminologia erabileraren bultzadaz hazi egingo da; baina, denean bezala, hazteko hasi egin behar!

Joseba Makazaga.

Eskerrak

Eskertu beharra daukat Donostiako Euskararako Udal Patronatuak eskainitako dirulaguntza, horri esker Asier Lasari beka bat ordaindu ahal izan baitiogu, eta klaseetarako prestatutako materialaz gain, Asierrek itzulitako hainbat eta hainbat atalekin osatu ahal izan baitugu liburua.

UZEI ere ezin utziko dut aipamenik gabe, makina bat zalantza argitu baitidate, eta horien antzera, zuzentzaile lanetan lagundutakoei ere eskerrak eman behar dizkiet, Patxi Angulo, Xabier Artola (fakultateko euskaltzain partikularra), X talde osoa, beren lana guk erabiltzeko moduan jartzeagatik, eta horien artean ez dut Ruben Urizar ahaztu nahi, hainbat hitzen itzulpena egitean eskainitako laguntzagatik, ez eta Iñaki Alegria ere, zuzentzaile ortografikoa erabil nezan hartu behar izan zuen lanagatik. Eta oro har lan hau egitera bultzatu eta animatu nauten guztiak eskertu behar ditut.

Azkenik, Joxerrra Etxeberriak egindako lan eskergaitza aipatu nahi nuke, azken zuzenketaz bera arduratu baita. Idazkeraren zuzentasuna, testuen argitasuna eta liburuak eduki lezakeen kalitatea berari zor diogu.

Gaien Aurkibidea

1 Sarrera **1**
- 1.1 Liburu honi buruz . . . 1
- 1.2 Lan-eremua . . . 3
- 1.3 Garapena . . . 3
- 1.4 Infografia-aplikazioen ezaugarriak . . . 5
  - 1.4.1 Eredua . . . 5
  - 1.4.2 Tratamendua . . . 6
  - 1.4.3 Ikustea . . . 7
  - 1.4.4 Elkarrekintza . . . 8
- 1.5 Grafikoen liburutegiak . . . 8

2 Oinarrizko marrazketa-funtzioak **11**
- 2.1 Sarrera . . . 11
- 2.2 Marra eta kurben marrazketa . . . 13
  - 2.2.1 Gehikuntzazko algoritmoak . . . 13
  - 2.2.2 Analizatzaile Diferentzial Digitala: DDA . . . 15
  - 2.2.3 Brasenhamen algoritmoa . . . 16
- 2.3 Poligonoen marrazketa: betetzea . . . 22
- 2.4 Aliasing delakoa ekiditeko bideak . . . 27
  - 2.4.1 Gain-laginketa . . . 29
  - 2.4.2 Azalera-laginketa . . . 31
  - 2.4.3 Laginketa estokastikoa . . . 32

3 Objektuen adierazpideak **33**
- 3.1 Sarrera . . . 33
- 3.2 Poligonozko adierazpidea . . . 34
  - 3.2.1 Poligonozko objektuak eraikitzen . . . 36
- 3.3 Txatalen sareak . . . 37

3.3.1 Objektuak txatal-sareen bidez eraikitzen 38
3.4 Solidoen geometria eraikitzailea edo CSG 40
3.5 Espazioaren partiketa-teknikak 42
3.5.1 Octreeak 42
3.5.2 BSP zuhaitzak 45
3.6 Irudia sortzeko estrategiak 48
3.6.1 Poligonozko objektuen irudia sortzen 48
3.6.2 Txatal parametrikoen sarearen irudia sortzen 55
3.6.3 CSG adierazpideko objektuen irudia sortzen 56

4 Gainazal eta kurben adierazpena 61
4.1 Sarrera 61
4.2 Bi dimentsioko kurben adierazpena 61
4.3 Interpolazioa, egokitzea eta itxuraren kontrola 62
4.3.1 Interpolazioa 63
4.3.2 Interpolazioaren arazoak 64
4.3.3 Egokitzea 65
4.3.4 Itxuraren kontrola 68
4.4 Polinomioen oinarriak 69
4.5 Bezier kurbak 71
4.5.1 Bezier kurbak lotzen 75
4.5.2 Bezier kurben propietateen laburpena 77
4.6 Spline funtzioak 77
4.6.1 Oinarrizko splineak 80
4.7 Ekuazio parametrikoetan oinarritutako inplementazioa 101
4.7.1 Interpolazioa 102
4.7.2 Egokitzea 102
4.7.3 Itxuraren kontrola 102
4.8 Gainazal erregularrak eraikitzen 104
4.8.1 Interpolazioa 104
4.8.2 Itxuraren kontrola 107
4.9 Gainazal biparametriko kubikoak 107
4.9.1 Bezier gainazalak elkartzen 112
4.9.2 B-splineen txatalak 115
4.10 Gainazal parametrikoen irudigintza 116
4.11 Poligonozko sareen bidezko gainazalen hurbilketa 119
4.12 NURBak eta β-splineak 124
4.12.1 NURBak 125

4.12.2 β-splineak 126
4.13 Objektuak sare parametriko bikubikoen bidez eraikitzen . . . 127
4.13.1 Gainazalen egokitzea 127
4.13.2 Zeharkako ebakiduraren bidezko diseinua 131
4.13.3 Kontrol-poligonoaren bidezko diseinua 133

5 Geometri aldaketak **139**
5.1 Sarrera 139
5.2 Bi dimentsiotako aldaketak 140
5.2.1 Leku-aldaketa 141
5.2.2 Neurri-aldaketa 141
5.2.3 Biraketa 142
5.2.4 Islada 146
5.2.5 Aldaketen kateaketa 148
5.2.6 Koordenatu homogeneoak 150
5.3 Hiru dimentsiotako aldaketak 151
5.3.1 Leku-aldaketa 151
5.3.2 Neurri-aldaketa 151
5.3.3 Islada 152
5.3.4 biraketa 154

6 Ikuste-Sistemak **161**
6.1 Sarrera 161
6.2 Bi dimentsiotako irudigintza 161
6.2.1 Mugaketa 162
6.2.2 laukiratzea 168
6.3 Hiru dimentsiotako irudigintza 170
6.3.1 ikuslearen erreferentzi sistema 172
6.3.2 1. irudigintza-sistema 174
6.3.3 2. irudigintza-sistema 180
6.3.4 3. irudigintza-sistema 182
6.3.5 4. irudigintza-sistema. PHIGS 182
6.3.6 Atze-aurpegien ezabapena 191

7 Errealitate-itxura **193**
7.1 Ezkutuko azalerak 193
7.1.1 Z-buffer. 195
7.1.2 Spanning Scanline 198

7.1.3 Sakonerarekiko ordenazio-algoritmoa 203
7.1.4 Ray-casting 207

8 Argiztatze-ereduak **211**
8.1 Sarrera 211
8.2 Oinarrizko ereduak 212
8.2.1 Ingurune-argia 212
8.2.2 Islada barreiatua 213
8.2.3 Ispilu-islada: Phong-en eredua 214
8.3 Islada argi-iturri anitzekin 218
8.4 Warn-en eredua 218
8.5 Intentsitatearen ahuldura 220
8.6 Koloreari dagozkion zuzenketak 221
8.7 Gardentasuna 222
8.7.1 Errefrakziorik gabeko gardentasuna 223
8.7.2 Errefrakziodun gardentasuna 224
8.8 Itzalak 227
8.9 Poligonoak argiztatzeko teknikak 227
8.9.1 Gouraud-en interpolazioa 228
8.9.2 Phong-en interpolazioa 231
8.9.3 Gouraud versus Phong 233

9 Izpi-hedaketa **237**
9.1 Sarrera 237
9.2 Oinarrizko algoritmoak 238
9.3 Ortzadarraren ezaugarri optikoak 239
9.4 Izpi-hedaketaren inplementazioa 241
9.4.1 Prozeduraren eraginkortasuna 243
9.5 Izpi-hedaketaren geometria 247
9.5.1 Ebaketak 247
9.6 Islada bidezko argiztatze-eredua 255
9.7 Itzalak 259
9.8 Izpiak desbideratuz 261
9.9 Izpi-hedaketa eta anti-aliasing 264
9.10 Izpi-hedaketa eraginkor bihurtzen 266
9.10.1 Moldaerazko sakonera-kontrola 266
9.10.2 Borne-bolumenak 267
9.10.3 Lehen talkaren azkartzea 271

9.10.4 Espazio-koherentzia 273

10 Erradiositatea **279**
10.1 Sarrera 279
10.2 Erradiositate-ekuazioa 280
10.3 Forma-faktoreak kalkulatzen 282
10.4 Txatalak azpitxataletan zatitzen 287
10.5 Urratsez urratseko hobekuntza 289
10.6 Forma-faktore zehatzagoen kalkulua 291

11 Kolore-espazioak **295**
11.1 Sarrera 295
11.2 Hiru osagairen bidezko adierazpena 296
11.3 Kolore-espazioak 298
11.3.1 RGB eredua 300
11.3.2 HSV eredua 301
11.3.3 HLS eredua 304
11.3.4 CIE XYZ espazioa 307
11.3.5 CIE xyY espazioa 311
11.3.6 CIE LUV espazioa 311
11.4 Monitore desberdinen azterketa 316
11.4.1 Monitore desberdinak eta kolore berdinak 316
11.4.2 Koloreen mugaketa 318

12 Animazio-teknikak **319**
12.1 Sarrera 319
12.2 Ordenadore bidezko animazio-teknikak 320
12.3 Behe-mailako kontrola 321
12.3.1 Gidoi-sistemak 321
12.3.2 Keyframing 321
12.3.3 Spline bidez zuzendutako animazioa 322
12.4 Adierazpenaren animazioa 322
12.4.1 Objektu artikulatuen animazioa 322
12.4.2 Objektu bigunen animazioa 323
12.5 Prozedura bidezko animazioa 327
12.5.1 Partikula-multzoen animazioa 327
12.5.2 Portaeraren animazioa prozedura bidez 328
12.5.3 Eredu analitikoen animazioa 329

A Spline marrazketarako programa **331**
A.1 Zenbait definizio: globala.def 331
A.2 Erazagupen globalak: globala.h 333
A.3 Poligonoa marrazteko funtzioa 333
A.4 B-splinearen hasieraketa 334
A.5 Ardatz- eta oinarri-marrazketa 336
A.6 B-splinearen marrazketarako funtzioa 337
A.7 Oinarriko splineen kalkulua 338
A.8 Programaren hasieraketa 340
A.9 Kontrol-puntuak mugitzeko funtzioak 342
A.10 Programa nagusia 343

Kapitulua 1

Sarrera

1.1 Liburu honi buruz

Esku artean daukazun liburu honen helburua ordenadore bidezko irudigintzaren inguruko algoritmo, metodo eta filosofia ezberdinak azaltzea da. Horretarako metodo eta filosofia ezberdinek irudiak sortzeko erabiltzen dituzten bideak zein diren eta lana nola egiten duten azalduko dugu.

Liburua kapitulutan banatuta dago; lehenengoa sarrera hau duzu, eta bertan irudiekin lan egiten duten aplikazioen ezaugarriak aztertzen dira. Ezaugarri horien artean gure arreta zeini eskainiko diegun azaltzen da, eta horrez gain ordenadore bidezko irudigintzaren historiaren gainbegiratutxoa ere aurkituko duzu.

Oinarrizko marrazketa-funtzioen kapituluan zenbait teknika eta algoritmo aurkituko dituzu; horiek pantaila batean marrak eta oinarrizko kurbak marrazteko erabil daitezke eta bakoitzaren azalpenak aurkitu ahal izango dituzu, jakina, eduki ditzaketen arazo eta konponbideak esplikatuz.

Objektuen adierazpideei buruzko kapituluan, eszenatoki bat adierazteko erabil daitezkeen metodo garrantzitsuenak zein diren eta bakoitzaren ezaugarriak azaltzen dira, era berean metodo bakoitzaren erabilerari buruzko azalpenak ere aurkitu ahal izango dituzu.

Gainazal eta kurben adierazpenei dagokien kapituluan, objektuak adierazteko metodo garrantzitsuenetako baten barrenak aztertzen dira. Gainazalak eta kurbak adierazteko metodo erabilienak sakonki aztertzen ditu kapituluak eta horretarako beharrezko diren oinarri matematikoak zein diren eta nola erabiltzen diren azaltzen du.

Geometri aldaketez hitz egiten duen kapituluan, objektu bati eragin diezazkiokegun oinarrizko aldaketak nola egiten diren azaltzen da. Oinarrizko aldaketatzat leku-aldaketa, biraketa, neurri-aldaketa eta isladapena hartzen ditugu, eta aldaketa horiek eragiteko bidea ulertzeko errazagoa izan dadin, lehenengo bi dimentsioetan nola egiten diren azaltzen dugu, horrela, ondoren, hiru dimentsioetan egiten dena errazago ulertzen baita.

Ikuste-sistemei buruzko kapituluan, kamera zer den azaltzen da. Ordenadore bidez irudiak sortu ahal izateko irudia nork eta nola ikusten duen zehaztu beharko da, zehaztasun horien bitartez irudia sortu beharko baita. Hori dela eta, zehaztasunak zein diren eta zehaztasun horiek erabiliz irudia nola lortzen den azaltzen du kapituluak.

Errealitate-itxura lortzeko bide erabilienak zein diren eta bakoitzak irudia nola lortzen duen irakurri ahal izango duzu hurrengo kapituluan. Objektuen adierazpidearen eta kameraren arabera irudian agertu beharko lukeena lortzeko bide ezberdinez hitz egiten dugu, eta bide bakoitzaren aldeko eta kontrakoak zein diren ere argitu ahal izango duzu.

Argiztatze-ereduei buruzko kapituluak argiek eszenatokian duten eragina aztertzen du, eta horiek lortzeko metodoak azaltzen ditu. Argi-isladak, gardentasuna, itzalak eta horrelakoak lortzeko algoritmoen azalpenak irakurri ahal izango dituzu kapitulu horretan.

Koloreei buruzko kapitulua ere aurki dezakezu liburuan, kapitulu horretan koloreak adierazteko metodo ezberdinen gainbegiratua aurki zenezake, eta era berean koloreak adierazteko metodo bakoitzak dauzkan mugak.

Irudiak sortzeko dauden beste bi filosofia ezberdinei dagozkien kapituluak ere badaude liburuan: bata izpi-hedaketari dagokion kapitulua da, bestea berriz erradiositatean oinarritutako filosofiaren metodoari buruzkoa. Bai bata eta bai bestea metodo garestiak dira, hau da, kalkulu-kopuru handia behar dute biek. Baina metodo horien bidez lortzen diren emaitzak oso onak dira eta ordenadoreen ahalmenak gora doazen neurrian, metodo horiek indar handiagoa hartzen ari dira. Bi kapitulu horietan oinarrizko metodoen funtzionamendua eta egin daitezkeen zenbait hobekuntza azaltzen dira.

Hurrengo kapituluak animazioak lortzeko teknika ezberdinak azaltzen ditu. Teknika bakoitzaren azalpentxoa irakurri ahal izango duzu baina sakonean sartu gabe.

Azkenik, eranskin gisa, splineak marrazten dituen programa ere azaltzen da. Programak C lengoaia erabiltzen du eta Unix inguruneterako idatzita dago, X sisteman oinarritzen baita. Hala ere, edozein inguruneterara eramateko egin beharreko bakarra, leiho-sistemari dagozkion funtzioak aldatzea izango

litzateke.

1.2 Lan-eremua

Infografia edo ordenadore bidezko irudigintzaz hitz egitean, informatikaren atala dela esan behar da. Atal hori grafikoak, irudiak edo marrazkiak erabiltzen edota sortzen dituzten aplikazioak sortzeaz arduratzen da.

Definizio hori oso zabala da; bertan irudiak erabiltzea sartzen da, eta era berean irudiak sortzea ere barnean darama. Hala ere, aplikazio horien artean bereizketa egitea komeni da:

1. Grafikoak eta irudiak manipulatzen dituzten aplikazioak. Horiek informazio grafikoa erabiliko eta manipulatuko dute, eta horren arabera irudi eta grafiko berriak ere sor ditzakete. Liburuan horrelako aplikazioez arduratuko gara.

2. Emaitzak adierazteko grafikoak erabiltzen dituztenak. Aplikazio horiek informazioaren erabilera grafikoa egiten dute, baina informazioak berak ez dauka grafikoa izan beharrik; zentzu grafikoa, emaitzak adierazteko erak edo bideak ematen dio.

1.3 Garapena

Ordenadore bidezko irudigintzak sorrera geldoa izan zuen, hardwarearen prezio altuengatik batez ere. Hasiera batean grafikoak karaktere alfabetikoekin egin izan ziren, eta ondoren pantaila grafikoak sortu ziren; baina garestiak zirenez, enpresa handietara mugatu zen erabilera. Hastapenetan erabiltzaileen eta ordenadoreen arteko informazioaren trukerako erabili izan ziren, baina benetako bultzada CAD[1] motako aplikazioek eman zieten, bai pantaila grafikoei eta bai infografiako aplikazioei ere. Gainera, memorien prezioen jaitsierak eta, oro har, hardwarearen prezioen jaitsierak erabilpen orokorra bultzatu zuten, eta, ondorioz, gaur egun bitxia gertatzen da kolorerik edo pantaila grafikorik gabeko ordenadorea.

Erabilera orokortzearen ondorioz, infografiako aplikazioak sortuz joan ziren; baina hasiera guztietan gertatzen den bezala estandarrik gabeko aplikazio, liburutegi eta arauak sortu ziren, eta software-etxe bakoitzak bere aplika-

[1] ordenadorez lagundutako diseinua

zioak sortu zituen eta merkatuan sartzen saiatu zen. Estandarizatze-saioak egon dira. Horien artean *c*ore, *g*ks, *p*higs eta *p*higs+ dira aipagarrienak: *c*ore eta *g*ks izenekoek porrot egin zuten; geroago, *p*higs ISO estandarra egin zuten eta hiru dimentsiotarako hedatu zuten, *p*higs+ izenez ezagutzen dena sortuz. Hala ere, Open GL liburutegia asko hedatu da merkatuan; hori Silicon Graphics-ek erabiltzen duen liburutegia da eta dagoenekoz ordenadore pertsonaletarako, LINUX sistema eragilerako eta abarretarako bertsioak lor daitezke. Horrek guztiak argi azaltzen du betiko arazoa, software-sortzaileek merkatua menperatzeko dauzkaten borrokak alegia. Eta horrek, beste kasu askotan bezala, estandarren porrota dakar. Dena den, hardware bitartez gauza asko egiten dira gaur egun eta horrek, nolabait, behe-mailako estandarrak sortzen ditu: horien artean Z-buffer izenekoa adibide modura har daiteke.

Gaur egun erabiltzen den hardwarearekin sortutako irudien kalitatea ona da. Bideogintzan erabiltzen diren irudien neurriak 720 x 512 pixelekoak dira, pixel bakoitzarentzat 24 bit erabiltzen direlarik. Hala ere, oso normala da neurri eta kalitate hobea daukaten irudiak erabiltzea, edozein ordenadorek 1024 x 1024 pixeleko irudiak (edo gehiagokoak) erabil baititzake eta gainera pixel bakoitza adierazteko 96 bit edukitzea ere ez baita beste munduko ezer. Baina oraindik arazoak ere badaude. Esate baterako, bideogintzan begiari mugimendua dagoela sinestarazteko, segundoko 25 irudi erakutsi behar zaizkio, eta horrenbeste irudi sortzeko behar den kalkulu-abiadura gaur egun ere ez da lortzen. Ondorioz, azkartze-algoritmo eta metodoak garatu dira, eta, jakina, algoritmo horiek azken finean irudia ez kalkulatzeko egiten diren tranpatxoak dira.

Esandakotik irudigintzarako hardwarearen ezaugarriak atera ditzakegu. Honako hauek izango lirateke:

- Memoria handia behar dugu. Irudi bakar bat adierazteko, bideogintzan adibidez, 1.000.000 byte baino gehiago behar ditugu, eta kontuan eduki segundoko 25 irudi behar ditugula!

- Kalkulu-abiadura handia. Nahiko memoria edukitzearekin ez dugu ezer lortzen, memoria horretan gorde beharreko irudia kalkulatzeko, denbora luzean zain egon behar badugu.

- Erakuste-abiadura handia. Irudiak kalkulatzeaz gain pantailan erakusteko abiadura ere behar dugu; kontuan eduki segundoko 25 irudi nahi

ditugula, eta horiek jartzea eta kentzea abiadura handian egin behar dela.

1.4 Infografia-aplikazioen ezaugarriak

Ordenadore bidezko irudigintzaren barnean mota askotako aplikazioak sar genitzake; bakoitzak bere bete beharrak izango ditu, baina gehienetan ezaugarri jakin batzuk ikus daitezke:

- Ereduren bat sortu edo erabiltzen dute.
- Ereduaren aldaketak, edo bere tratamendu eta prozesatzea egiten da.
- Emaitzak erakutsi behar dira; horretarako, ikuste-prozesua egongo da.
- Erabiltzailearekiko elkarrekintza egon ohi da.

1.4.1 Eredua

Ordenadore bidezko irudigintzaren barnean sar daitezkeen aplikazioek gehienetan objektuekin egiten dute lan, eta azken finean objektu horiei dagozkien irudiak lortu nahi izango ditugu. Hori ez da beti egia, kasu askotan lortu nahi ditugun irudiak ez baitira objektuenak izango, baizik eta, beste modu bateko informazioa erabiliz informazio hori adierazteko irudiak ulergarriago egiten direnez, grafikoki azaltzea izango baitute helburutzat. Dena den, gehienetan objektuekin egiten da lan eta objektu horiek nola edo hala adierazi behar dira. Benetako objektuak gehienetan oso konplexuak dira eta horiek adieraztea zaila izan daiteke; baina gaur egun irudigintza aplikatzen den alor gehienetako objektuak gizakiek sortutako objektuen alorrekoak dira eta gehienetan oso geometrikoak izaten dira (gizakiak makinen bidez sortzeko modukoak) eta, ondorioz, objektuen eredua sortzea ez da hain zaila izango. Bestalde, konplexutasun handiko objektuentzat hurbilketak erabil daitezke: poligono bidezko txatalak, azalera interpolatzaileak ...

Horrelako objektuez gain adieraz erraza ez den objektu-mota asko dago. Horientzat eredu edo modelo bereziagoak erabili beharko ditugu: adibide bezala sua, kea edo lainoa izan daitezke. Beste zenbait objektu adierazteko gutxi gora-beherazko ereduak erabil daitezke, prozedura ezagunen bidez sortutakoak: esaterako, fraktalak horien artean sar genitzake.

Azken finean irudiko osagaien barne-eredua beharko dute eta hori lortzeko bide asko dago. Zenbaitetan eredua eskaner bidez, digitalizatzaile bidez, X izpi bidez edo horrelako metodoen bidez hartuko dugu zuzenean; baina beste askotan sortu egin beharko dugu eredua eta, horretarako, geometri informazioa edo topologi informazioa beharko dugu. Sortzeko bideak ere asko dira, aipagarrienak prozeduretan oinarritutakoak eta geometrian oinarritutakoak dira, fraktalak prozeduran oinarritutakoak dira, baina nagusitasuna geometri informazioan oinarritutakoen esku dago.

Geometrian oinarritutako ereduek betebehar gehienetarako balio dute, eta erabiltzailearentzat ulerterraz eta erabilterrazak dira. Horren aurrean eredu horiek sortzeko programaren zailtasuna dago, erabiltzaileari eman beharreko aukerak programatzea ez baita erraza. Horien oinarriak ondokoak dira:

- Aldaketa afinak: neurriaren, kokapenaren eta antzeko aldaketak.
- Eragiketa boolearrak: oinarrizko objektuen batuketa, diferentzia, ebaketa eta antzekoak.
- Eragiketa eulertarrak. Erpin berrien gehiketak, aurpegi-aldaketak eta antzekoak.

Kasu askotan poligono bidezko hurbilketa ez da nahikoa izango. Esate baterako, espazio-untzi, abioi edo itsasuntzien eraikuntzan lan egiten dutenek, azalera aerodinamikoak behar dituzte eta horiek leunak izan behar dute; gainera, azalera horien arteko loturek ere leunak izan beharko dute. Hori matematikoki ikusita, azalera auzokideek ukitze-puntuan deribatu jarraiak eduki beharko dituztela esatea da; eta ez lehenengoak bakarrik, gehienetan bigarrenak ere bai.

Horrelako ereduak eraikitzea ez da erraza izango. Erabiltzaileak zertan diharduen jakin beharra izango du, baina hala ere gauzak asko erraztu beharko zaizkio eta horrek era intuitiboan lan egiteko aukera eskatuko du. Hori dela eta, horrelako objektuen itxura kontrolatzeko bideak jorratu beharko ditu aplikazioak; eta horrek bere zailtasunak izango ditu, bai programatzean, bai eta, lehen aipatu dugun bezala, erabiltzean ere.

1.4.2 Tratamendua

Tratamenduak ereduarekin zerikusi handia dauka: datu-egitura jakin bat lortu ondoren, egitura horri aldaketak egin nahiko dizkiogu, irudi ezberdinak

lortu ahal izateko. Aldaketa horiek mota askotakoak izan daitezke: geometri aldaketak, ezaugarri aldaketak (bolumena, pisua, indarrak . . .). Beraz, tratamenduaren atalean kalkulu aldetik pisu handiko prozedurak egon ohi dira eta denbora askokoak izan daitezke.

Beste aplikazio-mota batzuetan, aurretik hartutako irudiekin dihardutenak adibidez, dagoeneko eredua sortuta eduki dezakete, eta eredu hori nola edo hala aldatzea izango da helburua. Digituzko imajinekin lan egiten duen aplikazio batek irudiari iragazkiren bat pasa diezaioke, anti-aliasing algoritmoekin jardun daiteke edo enfokatzen saia daiteke. Horrelakoetan tratamendua irudiari berari egingo zaio eta aplikazio horien eredua irudia bera adierazteko eredua izango da.

1.4.3 Ikustea

Eredua lortu eta horri aldaketak eragitearekin, ez dugu ezer handirik lortzen; garrantzitsuena eredua eta bere gaineko aldaketei dagozkien irudiak lortzea izango da, izan ere atal horrek ematen baitio izena irudigintza edo infografiari. Horren guztiaren barnean, azpiatal asko dago:

- Marra eta kurbak marraztea.
- Ikuste-leihoaren barnealdekoa mugatzea.
- Proiekzioak kalkulatzea.
- Azalerak betetzea.
- Ikustezinak diren zatiak kentzea.
- Errealitate-itxura lortzea. Argiztatuz, testura edo txantiloiak erabiliz . . .
- Animatzea. Hori azken finean ereduaren tratamenduaren barnean sar genezake, tratamenduaren ondoren berriz irudi berri bat sortzea izango da-eta animatzea.

Azpi atal horiek guztiak hasieran software bidez egin izan dira, baina gero eta VLSI zirkuitu integratu gehiago dago eta ikuste-prozesua azkartzeko laguntza ikaragarria ematen dutela esan behar da. Hori dela eta, algoritmo asko integratu ahal izateko moduan sortu izan da.

1.4.4 Elkarrekintza

Aplikazio gehienek erabiltzailearekin elkarrekintza-lanean jardun behar izaten dute, eta jakina, grafikoak sortzeko aplikazioak izanik, erabiltzailearekin egin behar dituen informazio-aldaketak ere grafikoki egiten dituzte. Hau da, interfaze grafikoak dira nagusi. Interfaze horiek erabilterrazak izan behar dute eta horrez gain erabiltzaileak egin ditzakeen hutsegiteak konpontzeko bidea eman behar dute. Ezaugarri nagusiak ondokoak izan daitezke:

- Erabiltzen errazak.
- Azkarrak.
- Hutsegiteak konpontzeko bidea emango dutenak.
- Adimentsuak?

1.5 Grafikoen liburutegiak

Grafikoen liburutegiak irudiak sortzen lagunduko diguten funtzioen bildumak dira; abstrakzio maila handiagoa eskaintzen digute eta bertako funtzioek oinarrizko irudi konplexuagoak erabiltzeko aukera ematen digute. Horrez gain iturburu-programak ingurunez aldatu ahal izango ditugu; hori egin ahal izateko erabili dugun liburutegia, ingurune berrian ere eduki beharko dugu, baina horretaz liburutegi egileak arduratzen dira. Ingurune guztietan liburutegi hori egon dadin aukera onena, estandarrak erabiltzea da, bai onartutakoak edo bai erabilera dela eta estandartzat har daitezkeenak ere.

Guk sortu nahi dugun aplikazioak liburutegirik erabiliko ez badu, aukera onena geuk sortzea izango da; hau da, oinarrizko funtzioen liburutegi minimoa egitea izango litzateke zuzenena, eta bertan gutxienez hiru motatako funtzioak eduki beharko genituzke:

1. Marrazteko eta poligonoak betetzeko oinarrizko funtzioak, hau da, ikusteko oinarrizko funtzioak.
2. Sarrera eta irteerako eragiketak egiteko funtzioak.
3. Egoeraren berri emateko funtzioak.

Funtzio horietan guztietan parametroak erabili beharko dira; horrela, posibilitate guztiak erabiltzeko aukera izango dugu eta ez dugu oso antzeko prozedura piloa sortu beharko. Hiru mota horietan sartzen diren prozedurak asko izan daitezke, batez ere sarrera, irteera eta egoeraren berri ematen dutenak. Irteerako funtzioen artean zer eta nola ikusi nahi dugun adierazi beharko dugu; horren barnean ereduaren erabilerarako funtzio guztiak ere sar daitezke eta horri buruzko informazioa lortu eta aldatzekoak. Beraz liburutegietan funtzio kopuru handia egongo da; gainera arazo horietan guztietan ados jartzea zein zaila den ere imajina dezakegu, eta ondorioz estandarrak lortzeko beste arazo bat ere uler dezakegu.

Kapitulua 2

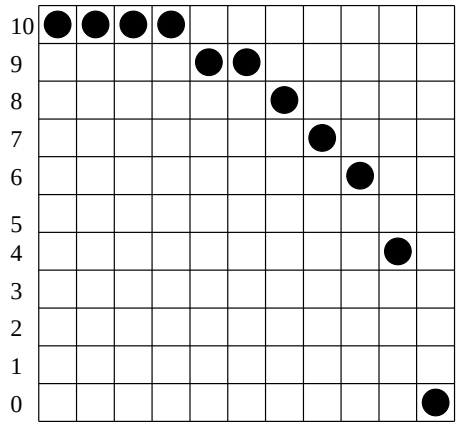
Oinarrizko marrazketa-funtzioak

2.1 Sarrera

Gaur egun gehien erabiltzen den marrazte- edo erakuste-gailua ordenadorearen pantaila edo CRT delakoa da. Pantaila hori, azken finean, pixelez osatutako matrizea besterik ez da, eta beraz, zerbait marrazterakoan bertako pixelak piztu edo itzaltzea besterik ez dugu egin beharko. Hala ere piztea era askotan kontsidera daiteke:

- Bi egoeratara mugatuz, hau da, piztuta ala itzalita egon daitekeela kontsidera dezakegu.
- Aztertu nahi dugun arazoaren arabera ez da nahikoa izango piztuta ala itzalita egon daitekeela pentsatzearekin; beraz, askotan zuri eta beltzaren artean gris mota asko daudela edo lor ditzakegula kontsidera genezake.
- Errealitatea irudikatu nahi dugunean, ez dugu aurrekoarekin nahikoa izango, eta pixel bakoitzaren koloreaz hitz egin beharko dugu.

Dena den, pixelaren izaeraz gain, pantailan marrazterakoan argi eduki behar da matrizearen kontzeptua eta, azken finean, marrazkia puntu zehatz batzuetara mugatzen dela. Pixelen artean ez dago ezer; jakina, tartea oso txikia izango da, baina bi pixelen artean tartea egongo da. Horrek zenbait arazo sor ditzake, guk irudiz agertu nahi dugunaren izaera jarraia izango baita,



x	0	1	2	3	4	5	6	7	8	9	10
y	10	9.9	9.7	9.5	9.1	8.6	8.0	7.1	6.0	4.3	0
biribilduz	10	10	10	10	9	9	8	7	6	4	0

Irudia 2.1: zirkunferentzi laurdena marrazteko arazoak.

baina horretarako erabiliko dugun dispositiboa, berriz, diskretua. Bataren eta bestearen arteko ezberdintasunak gainditzen baditugu, sortutako irudiaren itxura, guk nahi duguna bezalakoa izango da. Baina beti argi eduki behar dugu ondoko arazoen gainditu beharra:

1. Informazio-galera. Hasieran jarraia zena jarraia ez den zerbait bihurtuko da, eta jakina, aldaketa horretan informazio-galera egongo da.

2. Sortutako irudiak hutsuneak edo jarraitasun ezak eduki ditzake; adibide modura, 2.1 irudia har daiteke.

3. Aliasing. Gure garunak irudi diskretua ikustean ezaguna duen zerbaitekin erlazionatzeko joera du, baina ikusten duen horretara ez dago ohituta; beraz, antzekotasunak bilatzen ditu eta askotan adierazi nahi zenaren ezberdina den zerbaitekin parekatzen du, hau da, beste zerbait balitz bezala ikusten dugu. Irudikatu nahi duguna eta beste zerbait horren marrazki diskretuak berdinak direlako sortzen da arazoa.

4. Egin beharreko kalkulu-kopuru handia. Pixel bakoitzaren informazioa jaso eta aldatu behar dugu; gainera, animazioaren kasuan segundoko 20 irudi sortu behar dira, eta eszena bakoitzak, bideoaren kasuan 720 x

512 x 24 bit behar ditu. Argi dago informazio-kopuru handia sortu eta maneiatu behar dela. Horren ondorio zuzena, oinarrizko algoritmoen garrantzia da; horiek ahalik eta eraginkorrenak izan behar dute eta ahal den neurrian zirkuitu integratu berezietan inplementatuko dira. Batez ere poligonoak marraztu eta betetzeko algoritmoak integratzen dira; baina horretarako algoritmo eraginkorrak lortu behar dira.

2.2 Marra eta kurben marrazketa

Lan horietarako algoritmo asko sortu da. Hemen ez ditugu denak azalduko, baina algoritmo horien ideia nagusiak azaltzen saiatuko gara.

2.2.1 Gehikuntzazko algoritmoak

Batez era marra zuzenak egiteko erabiltzen dira eta ekuazio esplizitua erabili beharrean marraztu behar den pixelaren aurrekoaren ezagutzan oinarritzen dira.

Ekuazio esplizitua erabiliz eta malda 0 eta 1 artekoa dela suposatuz, x bakoitzarentzat y lortu behar da eta gure kasuan:

$$x_i = x_{i-1} + 1$$

$$y(x_i) = mx_i + k$$

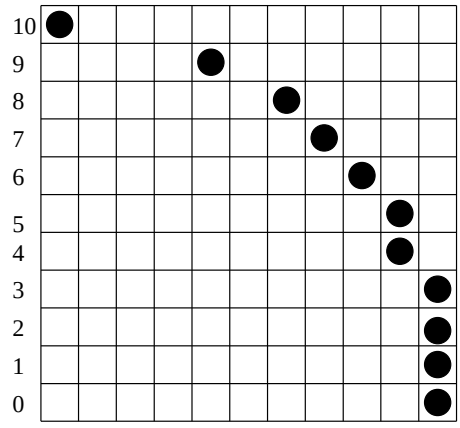
y kalkulatzeko, biderkaketa, batuketa eta biribilketa egin behar da. Algoritmo hori ikaragarri erabiliko dela kontuan hartzen bada, gehiegitxo dela esan behar da, eta eraginkorragoa egitea komeni da.

Aurreko bi ekuazioetatik ondokoa esan dezakegu:

$$y(x_{i+1}) = mx_{i+1} + k = m(x_i + 1) + k = mx_i + k + m = y(x_i) + m$$

Ikus dezakegunez, marraren aurreko puntuan oinarrituz hurrengoa lor dezakegu eta gainera, batuketa eta biribilketa eginez, biderkaketarik gabe. Dena den, horrek guztiak malda 1 baino txikiagoa denean besterik ez du balio, malda handiagoa balitz, marraren jarraitasuna galduko genuke, eta hori gerta ez dadin x eta y-ren paperak aldatu beharko genituzke, eta honako ekuazio hauek erabili:

$$y_i = y_{i-1} + 1$$



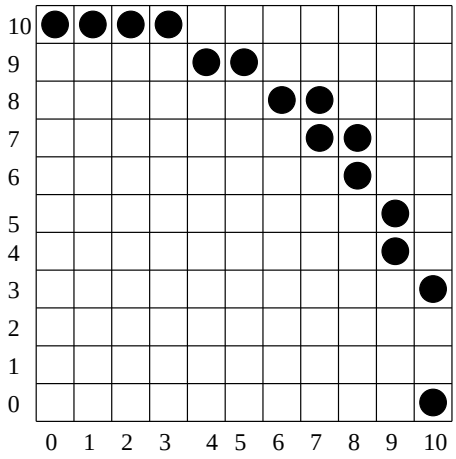
y	0	1	2	3	4	5	6	7	8	9	10
x	10	9.9	9.7	9.5	9.1	8.6	8.0	7.1	6.0	4.3	0
biribilduz	10	10	10	10	9	9	8	7	6	4	0

Irudia 2.2: X aldagaia Y-ren funtzio modura marraztuz.

$$x(y_i) = \frac{1}{m} y_i + h$$

Zirkunferentzien kasurako horrelako algoritmoak ez digu balioko. Adibide modura 2.1 eta 2.2 irudiak erabil ditzakegu. Bertako zirkunferentziaren ekuazioa $x^2 + y^2 = 100$ da; lehenengoan aldagai aske modura x erabili dugu eta bestean y, baina hala ere hutsuneak gelditu zaizkigu. Zirkunferentzian bi aldagaien aldaketa ez da finkoa eta momentu batzuetan x azkarrago handitzen da, baina beste batzuetan y aldagaiaren aldatze-abiadura handiagoa da; beraz, ezin dugu bataren eta bestearen arteko paper-trukaketarekin hutsuneen arazoa konpondu.

Pentsa genezakeen irtenbidea, tarteko pixel gehiago kalkulatzea izan daiteke. Beharbada, horrela pixelen arteko hutsuneak bete ahal izango genituzke; baina hori ez da nahikoa. Diametroaren arabera tarteko puntu gehiago edo gutxiago kalkulatu beharko genituzke eta, gainera, alferrikako lan gehiegi egin beharko genuke. Adibide modura, gure zirkunferentzi laurdenari dagozkion pixel-kopuruaren bikoitza kalkulatuz lortuko genukeena, 2.3 irudiak adierazten du. Pixel-kopurua bikoiztu arren, bertan hutsuneak agertzen zaizkigu oraindik. Ondorioz, beste algoritmo-mota batzuk erabili beharko ditugu.



Taula horretako balioak 2.1 irudikoarekin osotu behar dira.

x	0.5	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
y	9.9	9.8	9.6	9.3	8.9	8.3	7.6	6.6	5.2	4.3
biribilduz	10	10	10	9	9	8	8	7	5	4

Irudia 2.3: puntu-kopuru bikoitza erabiliz.

2.2.2 Analizatzaile Diferentzial Digitala: DDA

Metodo horiek 1. mailako ekuazio diferentzialetan oinarritzen dira. Kurba, ekuazioaren soluzioa izango da, eta ekuazioaren ebazpena zenbakizko metodo bidez egiten dute. Ekuazio diferentziala $y' = F(x, y)$ modukoa izango da eta hasierako balioa ezaguna da, hau da, $y(x_0) = y_0$.

Gure helburua kurbako (x_i, y_i) bikoteak lortzea da; horretarako, zenbakizko metodoren bat erabili beharko dugu. Eulerrena izan daiteke posibilitateetako bat. Gogoratu Eulerren metodoak deribatua maldaren bitartez hurbiltzen duela, hau da, (x_i, y_i) eta (x_{i+1}, y_{i+1}) direlakoak kurbako puntuak izanik, deribatua lehenengo puntuan bi puntuak lotzen dituen zuzenaren malda dela suposatzen du:

$$y_i' = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

Demagun $x_0, x_1, \ldots x_n$ balioei dagozkien $y_0, y_1 \ldots y_n$ lortu nahi ditugula. Euskarriko balioen arteko aldea $x_{i+1} - x_i = h_i$ eran adieraziko dugu, balio horiek zenbat eta txikiago izan, orduan eta hurbilketa hobea lortuko dugu.

Eulerren metodoarekin, hau da, deribatua lortzeko malda erabiliz honako hau daukagu:

$$y'_i = F(x_i, y_i) = \frac{y_{i+1} - y_i}{h_i}$$

Ondorioz:

$$\begin{aligned} y_{i+1} &= y_i + h_i \; F(x_i, y_i) \\ x_{i+1} &= x_i + h_i \end{aligned} \tag{2.1}$$

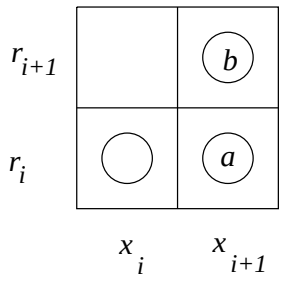
Hasierako balioak ezagunak direnez, ekuazioan $F(x_i, y_i)$ balioa besterik ez zaigu falta, eta hori kurba bakoitzaren arabera egongo da. Marra zuzenen kasuan deribatua balio konstantea da, hau da, $\Delta y/\Delta x$; beraz, marrako puntuak lortzea berehalakoa da. Puntuen artean hutsunerik egoterik nahi ez badugu, x-ren kasuan h_i balioa begiratu behar da. Ezin da 1 baino handiagoa izan; bestela x ardatzean hutsuneak sortuko lirateke. y ardatzaren kasuan, puntu batetik bestera pasatzean $h_i F(x_i, y_i)$ gehitzen zaionez, balio hori da zaindu beharrekoa; ezin izango du horrek bat baino handiagoa izan.

Zirkunferentziaren kasuan deribatuari dagokion ekuazioa $y' = -x/y$ da; beraz, 2.1 ekuazioan $F(x_i, y_i)$ dagoen lekuan $-x_i/y_i$ jarri beharko dugu. Kontuan eduki behar da, horrek errorea sortzen duela, eta ondorioz ez ditugula benetako kurbaren puntuak lortzen, egindakoa hurbilketa baita.

2.2.3 Brasenhamen algoritmoa

Hori DDA motako algoritmoen artean sar genezake, eta marra zuzenak marrazteko sortutako metodoa da. 2.2.2 atalean azaldu dugun bezala, marrako puntuak aurkitu behar ditugu. Marraren malda 1 edo txikiagoa bada, hutsunerik ez sortzeko, $x_{i+1} = x_i + 1$ egitearekin nahikoa da; eta 2.1 ekuazioan deribatuaren balioa jartzean, hau da, malda jartzean $y_{i+1} = y_i + m$ lortzen dugu. Hasierako balioak jakinak direnez, marrako balioak lor ditzakegu; dena den, y_{i+1} balioa biribildu egin beharko da, eta biribilketan errorea sortzen da.

Puntu batetik bestera pasatzerakoan, azken finean, bi aukeren artean bat hartzea besterik ez dugu egiten; 2.4 irudian ikus daiteke argiago horren esanahia. x_i balioari r_i egokitu badiogu, marraren malda 1 edo txikiagoa denez, x_{i+1} balioari r_i edo r_{i+1} egokitu behar diogu. Egokitze horretan sortutako errorea $e_{i+1} = y_{i+1} - r_{i+1}$ izango da. Beraz, bai a kasuan eta bai b kasuan bete behar diren baldintzak begiratuz gero:



Irudia 2.4: marra zuzena marraztean puntu baten auzokide bi izan daitezke.

a kasurako:

$$r_{i+1} = r_i \Longleftrightarrow \text{int}(y_{i+1} + 0.5) = r_i$$

Hau da:

$$r_i \leq y_{i+1} + 0.5 < r_i + 1$$

Baina $r_i = y_i - e_i$ eta $y_{i+1} = y_i + m$ direnez:

$$y_i - e_i \leq y_i + m + 0.5 < y_i - e_i + 1$$

$$-e_i \leq m + 0.5 < -e_i + 1$$

$$-0.5 \leq m + e_i < 0.5 \tag{2.2}$$

Bi baldintza horiek betetzen direnean, marra zuzenaren hurrengo puntua aurrekoaren eskuinekoa izango da. Hala ere biribiltze-errorea balio absolutuan 0.5 baino txikiagoa izango da, hau da, $-0.5 \leq e_i < 0.5$, eta ondorioz, $-0.5 \leq e_i + m$ beti beteko da, malda 0 eta 1 artekoa baita. Beraz, 2.2 ekuazioko baldintza honako honetara mugatzen da:

$$m + e_i < 0.5 \tag{2.3}$$

b kasurako:

$$r_{i+1} = r_i + 1 \Longleftrightarrow \text{int}(y_{i+1} + 0.5) = r_i + 1$$

Hau da:

$$r_i + 1 \leq y_{i+1} + 0.5 < r_i + 2$$

Baina $r_i = y_i - e_i$ eta $y_{i+1} = y_i + m$ direnez:

$$y_i - e_i + 1 \leq y_i + m + 0.5 < y_i - e_i + 2$$

$$-e_i + 1 \leq m + 0.5 < -e_i + 2$$

$$0.5 \leq m + e_i < 1.5 \tag{2.4}$$

2.4 inekuazioaren bigarren zatia beti egiazkoa dela konturatzea komeni da; hau da, m bat baino txikiagoa edo berdina da eta e_i 0.5 baino txikiagoa da. Ondorioz, hauxe geratzen da:

$$m + e_i \geq 0.5 \tag{2.5}$$

Laburbilduz, aukera egiterakoan kontuan eduki beharrekoa ondokoa besterik ez da:

1. $e_i + m < 0.5$. Orduan eskuineko pixela hartu behar da. Gainera, aukera horrekin egindako errorea $e_{i+1} = y_{i+1} - r_{i+1} = y_i + m - r_i = e_i + m$ da. Hori kalkulatzeko, aurreko errorea erabil dezakegu eta malda gehitze hutsarekin errore berriaren balioa lortuko genuke; horrela, hurrengo puntua kalkulatzeko erabili ahal izango dugu, erroreari malda gehitzean zer gertatzen den ikusi behar baita pixel berria aukeratzeko.

2. $e_i + m \geq 0.5$. Orduan eskuineko pixelaren gainekoa hartu behar da. Oraingoan errorea $e_{i+1} = y_{i+1} - r_{i+1} = y_i + m - r_i - 1 = e_i + m - 1$ da. Berriz ere, aurreko errorean oinarrituz, batuketa eta kenketa eginez lortzen dugu. Eta berriro, hurrengo puntua kalkulatzeko prest geundeke.

Hori guztia algoritmo erraz batean jaso dezakegu. 2.5 irudian agertzen den algoritmoak, aipamen horiek jarraituz marra bati dagozkion puntuak marraztuko lituzke.

Algoritmo horretan ikus daitekeenez, eragiketa oso gutxi eginez, puntu guztiak lor ditzakegu; gainera, bi aldagai izan ezik beste guztiak integer motakoak dira. Bi aldagai horiek koma higikorreko aritmetika erabiltzera behartzen gaituzte, baina zenbait aldaketa eginez, algoritmoak zenbaki osoekin

```
zuzena-marraztu(int x0,int y0,int x1,int y1)
{
int xi,ri;
double ei,m;
m = (y1 − y0) / (x1 − x0);
ei = 0;
ri = y0;
marraztu(x0, ri);
for(xi = x0 + 1;xi < x1;xi++)
        {
        ei+ = m;
        if (ei ≥ 0.5);
                {
                ri + +;
                ei − −;
                }
        marraztu(xi, ri;)
        }
}
```

Irudia 2.5: zuzen bati dagozkion pixelen kalkulurako algoritmoa.

lan egitea lor daiteke. Algoritmoak koma higikorreko aldagaiak if aginduak egiten duen konparaziorako behar ditu. Bertan, e_i balioa 0.5 balioarekin konparatzen du, baina horren baliokide dira ondoko konparazioak:

$$e_i \geq 0.5 \Leftrightarrow 2e_i \geq 1 \Leftrightarrow 2e_i - 1 \geq 0$$

Hau da, $2e_i - 1$ balioaren ikurra begiratzearen baliokidea da. Bestalde, balio horri zenbaki positiboren bat biderkatuz gero, ikurra ez da aldatzen eta ondorioz Δx balioa aurreko guztiari biderkatuz jartzean aurreko algoritmoaren if aginduan ondoko konparazioa egin behar da:

$$\text{if}(\Delta x(2e_i - 1) \geq 0)$$

Horren guztiaren balioa E_i aldagaian jaso genezake. Horrela, hasieran $e_i = 0$ betetzen denez, $E_0 = -\Delta x$ beteko da, eta balio horrekin hasiera dezakegu. Algoritmoak for aginduan bira bat ematean, bi kasu gerta daitezke:

1. $e_i < 0.5$ kasua: Kasu horretan e_{i+1} balioa $e_i + m$ da, eta ondorioz, $E_{i+1} = \Delta x(2e_{i+1} - 1)$ denez:

$$E_{i+1} = \Delta x(2(e_i + m) - 1) = \Delta x(2e_i - 1) + 2m\Delta x = E_i + 2\frac{\Delta y}{\Delta x}\Delta x =$$
$$= E_i + 2\Delta y$$

2. $e_i \geq 0.5$ kasua: Kasu horretan e_{i+1} balioa $e_i + m - 1$ da, eta ondorioz, $E_{i+1} = \Delta x(2e_{i+1} - 1)$ denez:

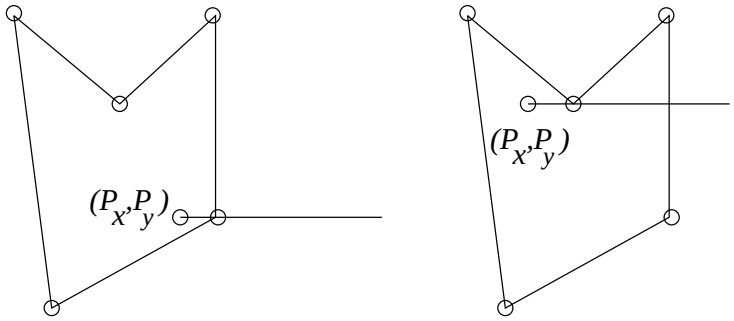
$$E_{i+1} = \Delta x(2(e_i + m - 1) - 1) = \Delta x(2e_i - 1) + 2m\Delta x - \Delta x =$$
$$= E_i + 2\Delta y - \Delta x$$

Beraz, for aginduaren barneko bi kasuetan, hurrengo birarako E_{i+1} nola kalkulatu behar den badakigu. Gainera, E_0 integer motakoa izan daiteke, eta horren aldaketa zenbaki osoak gehituz eta kenduz egiten denez, E_i aldagaia integer motakoa izan daiteke; gainera m aldagaia ez dugu behar, Δx eta Δy erabiltzen baititugu. Hori dela eta, 2.5 irudiko algoritmoa alda dezakegu, eta zenbaki osoekin lan egiten duen 2.6 irudiko algoritmoa erabili ahal izango dugu marraren puntuak kalkulatzeko. Algoritmo hori Brasenhamen algoritmo izenaz ezagutzen da.

Brasenhamen algoritmoa hardware bitartez eragin daiteke; horrela marra zuzenak marraztea ikaragarri azkarra izango da.

```
zuzena-marraztu(int x0,int y0,int x1,int y1)
{
int xi,ri,Δx,Δy,Ei;
Δx = (x1 − x0);
Δy = (y1 − y0);
Ei = −Δx;
ri = y0;
marraztu(x0, ri;)
for(xi = x0 + 1;xi < x1;xi++)
        {
        Ei+ = 2Δy;
        if (Ei ≥ 0);
                {
                ri + +;
                Ei− = Δx;
                }
        marraztu(xi, ri;)
        }
}
```

Irudia 2.6: Zenbaki osozko aritmetika erabiliz zuzen baten pixelak kalkulatzeko algoritmoa.



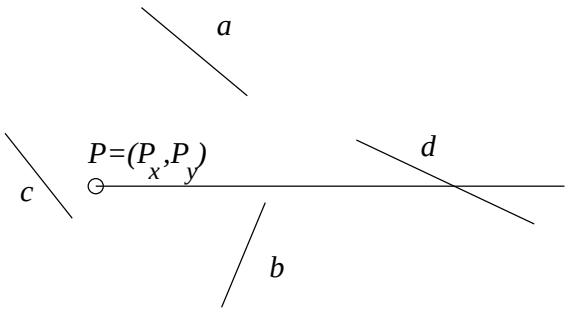
Irudia 2.7: puntua barnean dagoen ala ez erabakitzean sor daitezkeen arazoak.

2.3 Poligonoen marrazketa: betetzea

Poligonoak marraztea bi eratara uler daiteke. Alde batetik, poligonoaren mugak marraztea izan daiteke; horretarako, Brasenhamen algoritmoa erabiltzearekin nahikoa izango genuke, poligonoaren muga bakoitza marra zuzena izango baita. Baina poligonoak marraztean, gehienetan, bere barnealdean koloreren bat edo ehunduraren bat jartzea nahi izango dugu, eta horretarako barneko puntuak zehaztu beharko ditugu.

Era batera zein bestera poligonoa adierazteko, erpin-multzoa beharko dugu eta erpinetik erpinera doan marra, poligonoaren muga izango da. Erpinak zehazterakoan, ordenatuta egon beharko dute; eta gainera, beti norantza berean ordenatzea komeni da, erabakia hartu eta beti berdin egiteko. Horrela, poligonoak zuloak baldin badauzka, zuloari dagozkion erpinak alderantzizko norantzarekin ordenatu beharko lirateke, zuloa dela adierazteko. Puntu bat poligonoaren barnean dagoen ala ez erabakitzeko, algoritmoren bat beharko dugu, eta horretan Jordanen teoremak lagun diezaguke: teoremak dioenez, puntu horretatik infinitura doan zuzenak poligonoaren mugak zenbat aldiz zeharkatzen dituen begiratu behar da; kopuru hori bakoitia bada, puntua barnealdean dago; bestela, poligonotik kanpoko puntua izango da.

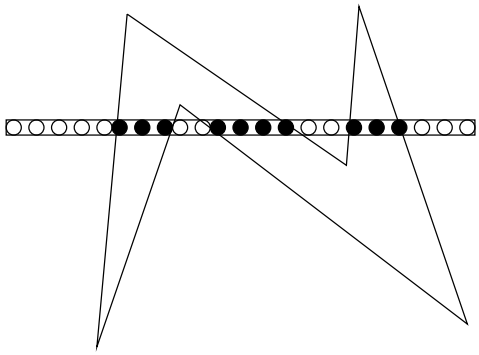
Jordanen teoremaren arabera barneko puntuak zehazteak erraza dirudi, baina praktikan jartzerakoan zenbait arazo aurki dezakegu. Marrak poligonoa zenbat aldiz mozten duen jakiteko, poligonoaren albo bakoitza mozten



Irudia 2.8: marra horizontalaren eta poligonoaren ertzen arteko ebaketaren kalkuluan ager daitezkeen egoera nagusiak.

ote duen begiratu behar da; baina kontu handiz jokatu beharko dugu. 2.7 irudian bi adibide agertzen dira. Kasu batean marra erpin baten gainetik pasatzen da, eta ondorioz poligonoaren bi albo mozten dituela erabaki dezakegu, erpina bi poligonotakoa baita. Arazo hori konpontzeko, bi alboetako bakarra kontatzea erabaki dezakegu. Baina bigarren kasuan ikus daitekeen bezala, horrela ere arazoak sortzen dira; poligonoarekiko ukitzailea bada, irtenbide horrek ez baitu arazoa konpontzen. Arazo horiei irtenbidea emateko bi marra erabil daitezke; baina horrek kalkulu-kopurua bikoiztuko luke eta ez da hori komeni. Beraz, benetako irtenbidea marraren eta poligonoaren arteko ebaketak zehazki aztertzea izango da; horretarako, marra horizontala erabiltzea da errazena.

Kasu gehienak 2.8 irudian agertzen direnak izango dira, Irudian $P = (p_x, p_y)$ puntutik hasita eskuinerantz doan marrak irudiko lau alboak mozten dituenentz begiratu behar dugu: a kasuan, marra behetik pasatzen da, $p_y < y_0, y_1$, eta ez du mozten poligonoaren alboa izango litzatekeen zatia; b kasuan, gainetik pasatzen da, $p_y > y_0, y_1$, eta, beraz, ez dago ebaketarik. c kasua konplexuagoa da: p_y balioa y_0 eta y_1 balioen artean dago, baina ebaketa-puntuari dagokion x balioa p_x baino ezkerrerago dago; beraz, ez dago ebaketarik. Azken kasuan, ebaketa dagoela esan dezakegu: p_y y_0 eta y_1 balioen artean dago eta ebaketa puntua p_x baino eskuinerago. Lau kasu horiez gain, ebaketa-puntua alboaren erpina izatea gerta liteke, eta kasu hori era berezian aztertu beharko dugu, marra poligonoarekiko ukitzailea den ala benetan ebaketa dagoen erabaki beharko baitugu. Ukitzailea bada, ukitzen duen erpinera heltzen diren bi ertzak, marra baino gorago (ala biak behera-



Irudia 2.9: lerro bakoitzean dauden pixel-ilarek adierazten dute poligonoa.

go) egongo dira; eta ebaketa balego bat gainean eta bestea azpian egongo lirateke. Beste era errazagorik ere erabiltzen da: pixela ertzaren maximoa bada, ez da kontuan hartzen, eta minimoa bada, kontuan hartuko litzateke. Erabaki hori erabiliko bagenu, poligono baten goreneko muturra kasualitatez lerro batek ukituko balu, ez genuke marraztuko; eta, era berean, poligono horrek gainean marra horizontala baleuka eta lerroan kokatuta balego, ez genuke marraztuko. Baina marra hori kasu gehienetan poligono auzokide baten beheko ertza izango da eta poligono horren barneko pixel gisa har daitezke. Azken finean, gure aukera izango da erabaki bat ala bestea hartzea.

Aipatu ditugun testen bidez zehazki esan daiteke puntua poligonoaren barnean ala kanpoan dagoen; baina denbora alferrik galtzea izango litzateke, marrazguneko pixel bakoitzeko azterketa horiek guztiak egitea. Hori egin beharrik ez izateko, objektuen koherentzia-ezaugarriak erabili ahal izango dira. Poligonoak goren daukan erpina baino gorago ez da poligono barneko punturik egongo, eta alderantziz, erpinen $Y_{\min}$ baino beheragoko lerroetan ere ez da poligonoaren barnealdeko pixelik egongo. Ezaugarri horrek lerroz lerroko azterketarako lerro askotan ezer egin behar ez izatea dakar, eta kasu askotan, poligonoa lerro gutxi batzuetara mugatzen denean, azterketa askoz ere azkarragoa izango da.

Objektuen koherentziaz gain, lerro bakoitzaren koherentzia-ezaugarriak kontuan hartzea komeni da; hau da, lerro bakoitza poligono barneko pixel-ilaraz eta poligonoz kanpoko ilaraz osatzen da, 2.9 irudian agertzen den bezala. Hori horrela izanik, lerro bakoitzean barne eta kanpo arteko mugak zein diren jakingo bagenu, mugatik mugara dauden pixelen kasuan ez

genuke azterketarik egin beharko; muga igarotzean poligono barnera sartzen gara (ala poligonotik irteten gara) eta hurrengo mugara arteko pixelak barnekoak (edo kanpokoak) direla badakigu, ondorioz azterketarik egin beharrik ez dugu izango. Beraz, gure arazoa mugak kokatzea da, baina egia esan, arazo hori konpontzeko bide erraza ezagutzen dugu. Hau da, mugak poligonoaren ertzak marraztean jaso genitzake, horrela, lerro bakoitzeko, lerroa mozten duten ertzen zerrenda jaso ahal izango dugu; zerrenda hori ordenatuz gero lerroko lehenengo pixeletik hasi eta azkenekoraino joan gaitezke, mugatik mugarainoko pixel-ilarak marraztuz. Azaldutako hori ongi ulertu behar da: ertz bakoitzak lerroa non mozten duen jaso behar dugu, ez ertz bakoitza adierazteko lerro horretan marraztutako pixelak; ertz bati lerro batean pixel bat baino gehiago egoki diezaiokegu, baina ertzak lerroa puntu bakarrean mozten duela kontuan eduki behar da, bestela, ertza marrazteko erabili diren pixel denak hartuko bagenitu, poligonoak lerro horretan behar baino ebaketa-puntu gehiago eduki ahal izango lituzke eta aipatutako metodoa gaizki erabiliko genuke. Hori egiten duen algoritmoa ulerterraza da; hiru betebehar nagusi izango ditu:

1. Aztertzen ari garen lerroak poligonoaren ertzekin dituen ebaketa-puntuak lortu.

2. Ebaketa-puntuak x koordenatuaren arabera ordenatu.

3. Ebaketa-puntuen bikoteen arteko pixelak, poligono barnekoak bezala marraztu, lehen esandakoaren arabera.

Orain arte azaldutakoaren arabera, objektuen koherentzia eta lerroaren koherentzia kontuan hartzeak abiadura azkartzen duela ikus daiteke; baina bada beste koherentziarik ere, lerro auzokideen arteko koherentziarena hain zuzen. Hau da, lerro batetik hurrengora dauden ezberdintasunak oso txikiak izan ohi dira, eta nolabait aurreko lerroaren informazioa erabiltzea lortuko bagenu, ziur aski abiadura azkartzea lortuko genuke. Horrek ez dauka horrela izan beharrik, baina memoria-arazoa daukaten makinekin lerro guztiei dagozkien ebaketa-puntu guztiak jaso beharrean, lerroz lerro joaten diren algoritmoak erabili izan dira; horientzat lerro baten informazioa hurrengorako berriz erabilgarria dela ikus daiteke, lerro bat mozten duen ertzak hurrengoa ere moztu egingo baitu, aurrekoan bukatzen ez bada behintzat, eta gainera ebaketa-puntua aurrekotik oso gertu egongo da, eta maldaren arabera kalkulatu ahal izango dugu. i lerroan ertzak ebaketa x_i puntuan baldin bazeukan,

```
void ebaketa_berria( int *x,int *zbkitzailea,int Δx int Δy)
{
*zbkitzailea += Δx;
while (*zbkitzailea >= Δy) /* Δx > Δy gerta daiteke */
      {*x ++;
      *zbkitzailea− = Δy;
      }
}
```

Irudia 2.10: ertz baten ebaketa-puntua eta maldaren datuekin ebaketa-puntu berria kalkulatzen duen algoritmoa.

orduan $i+1$ lerroan $x_{i+1} = x_i + 1/m$ puntuan edukiko du ebaketa berria. Beste era batera esateko, x_i-ri $\Delta x/\Delta y$ gehitu behar zaio. Jakina, hori marra bat kalkulatzearen parekoa da eta, nahi izanez gero, zenbaki osoak erabiltzen dituen algoritmo baten bidez adieraz daiteke. Horretarako, x_i balioari gehitu behar zaion balioa zein den begiratzea komeni da. Hor zenbakitzailea eta izendatzailea ageri dira, biak zenbaki osoak. Lerro bakoitzean ebaketa--puntua, x_i, eta zenbakitzailea jaso behar dira, eta lerro batetik bestera pasatzean aurreko lerroko zenbakitzaileari Δx gehitu behar zaio; horrela, lortutako zenbakitzailea izendatzailea baino handiagoa bada, ebaketa-puntuari 1 gehitu eta zenbakitzaileari izendatzailea kendu behar zaio. Horrek ebaketa--puntuaren zati osoa emango digu. Zati osoa beharrean biribilketa lortu nahi badugu, bi bider zenbakitzailea izendatzailea baino handiago denean gehituko dugu ebaketa-pixelaren balioa; eta, jakina, zenbakitzaileari izendatzailea kentzean zenbaki negatiboa lor daiteke, baina beti ere izendatzailearen erdia baino txikiagoa. Era horretan lan egitean, malda 1 baino txikiagoa izateak ez du arazorik sortzen; zenbakitzailea handiagoa bada, agian, behin baino gehiagotan kendu beharko diogu izendatzailea, eta ondorioz, 1 baino gehiago gehitu beharko zaio ebaketa-pixelari. Hori guztia 2.10 algoritmoan ikus daiteke.

Lerro auzokideen arteko koherentzia erabiliz, lerro-azterketaren algoritmoa berehalakoa da. Ezagutu behar den informazioa, lerro horretan parte hartzen duten ertzei dagokiena da. Bakoitzarentzat, 2.10 algoritmoak behar dituen datuak jaso beharko dira; horrela, hurrengo lerrora pasatzean, ertza bukatu ez bada behintzat, ebaketa-puntu berria kalkulatuko dugu. Algoritmoak bi taula erabiltzen ditu; lehenengoan irudiko ertz guztiak jasoko dira, bakoitza zein lerrotan hasten den adieraziz; bigarrenean, berriz, momentuko

lerroan parte hartzen duten ertzen informazioa edukiko dugu. Hasieran lerro guztiak irudiko ertzen taulan egongo dira, eta lerroko ertzen taula hutsik egongo da. Lerro bakoitza aztertzeko lerroari dagokion taulan, lerro horretan hasten diren ertzak sartuko ditugu eta ertzen taulatik kendu; gainera, aurreko lerroko ertzak ere kontuan hartuko ditugu, baina ertz horietakoren bat bukatu egin bada, lerroaren taulatik kendu egin behar da. Horrela, lerroaren taulan aurreko lerrotik datozen ertzak eta lerroan hasten direnak egongo dira, lerroa zeharkatzen duten guztiak alegia. Ertz berriei dagokien ebaketa-puntua berehalako da, lerro horretan baitaukate hasiera, eta aurrekotik datozenen ebaketa 2.10 algoritmoa erabiliz kalkula baitaiteke. Ebaketa horiek guztiak kalkulatu ahala, lista ordenatu atean sar genitzake eta, horrela, binaka hartuz, poligonoaren barneko eta kanpoko pixel-ilarak berehala marraztu ahal izango genituzke.

2.4 Aliasing delakoa ekiditeko bideak

Aliasing izenez ezagutzen dena, gure garunean gertatzen den zerbait da: objektu bati dagokion marrazki diskretuan oinarrituz, beste objekturen bat ikusten dugunean gertatzen da. Hau da, marrazki diskretu horren *alias* bat ikusten dugu. Horren adibide garbia, marra zuzena bereizmen txikiko pantailan adierazi nahi dugunean gertatzen da, marra zuzenaren ordez eskailera moduko zerbait ikusten baitugu.

Aliasing efektuaren arrazoi nagusia, espazioko eremuaren laginketa bidezko irudien adierazpidea da; hau da, irudia pixel-matrize baten bidez adierazi behar dugunez, espazioaren laginketa egin behar dugu, eta izatez jarraia den irudia, jarraia ez den eta azalera jakin bateko pixel-matrizera itzuli behar dugu. Laginketa modura ulertzea garrantzitsua da, pixelaren azaleraren zentroari dagokion intentsitatea baitagokio pixel osoari.

Beste era bateko aliasing berezirik ere ager daiteke; hori animazioan gertatzen da, eta hori ere jarraia den zerbait diskretu bihurtzearen ondorioa da, denboran zehar (denbora jarraia da) pixel baten intentsitatearen gaineko laginketa egiten baitugu. Segundoan 20 irudi erabiliko bagenitu, irudi bakoitzak segundoaren hogeiren bat bete behar luke, eta irudi hori sortzeko denbora-zati horretako momentu jakin bat erabiltzen dugunez, momentu horretan mugitzen ari den objektu txiki bat pixel batean egokituko balitz, denbora-zati batean objektu txiki hori ikusiko genuke; gainera, posiblea da objektu horrek hurrengo irudietako pixelen zentrorik ez ukitzea eta pixka ba-

tera, lauzpabost irudi beranduago, berriz ere pixelen baten zentroan agertzea. Hori gertatuko balitz, pixel horiek dir-dir eginez ikusiko genituzke eta hori ere aliasing modura har daiteke. Efektu horiek simulazio-kasuetarako oso kaltegarriak izan daitezke, zeren horrelakoetan, batez ere aldaketen aurreko jokaeren trebatzea baita helburua eta dir-dir egitea aldaketa baita. Aliasing efektua zenbait kasutan oso erraz gertatzen da:

- Argitasun-aldaketa dagoenean. Batez ere objektuen borneetan, horiek argitasun-aldaketen mugak izaten baitira eta gure begiek argitasunaren aldaketak jasotzeko gaitasun handia baitaukate. Kolore-aldaketa baino errazago bereizten dugu argitasun-ezberdintasuna; horrexegatik gertatzen da batez ere borneetan. Kasu horren barnean sar genitzake eskailera moduan ikusten diren marra zuzenak.

- Kolore-aldaketa askoko azaleretan ere gertatzen da, nahiz eta ez izan aurrekoa bezain nabarmena.

- Objektu txikien eraginez animazioan dir-dir eginez ager daitezkeen pixelak.

Irudien eraikuntza laginketa modura hartzeak, seinale-teoria erabili ahal izatea dakar, eta horrela Fourierren teoria erabil dezakegu anti-aliasing algoritmoak garatzeko. Irudia $f(x)$ funtzioak adierazten badu, bertako goi-frekuentziak kentzea komeniko litzateke; hori, $k(x)$ iragazkiarekin konboluzioa egitea izango litzateke. Konboluzioa egitea, Fourierren transformatuari iragazkiari dagokion funtzioa biderkatzea eta ondoren alderantzizko transformatua lortzea izango litzateke. Horrek irudia lausotzen du eta aldaketa handiak (frekuentzia jakin batetik gorakoak) ezabatzen ditu. Ez gara era horretako azterketa teorikoetan sartuko, baina ordenadore bidezko irudigintzan aliasing efektua desagertarazteko edo gutxitzeko erabiltzen diren metodo nagusiak azalduko ditugu:

- Gain-laginketa edo supersampling.

- Bi dimentsiotako anti-aliasing motako iragazkien hurbilketa egin eta goi-frekuentzien ezabaketa egitea.

- Laginketa estokastikoa.

Aliasing efektuaren jatorria laginketa dela esan dugu; beraz, hori gerta ez dadin onena laginketa hori ahalik eta jarraiena izatea komeni da. Horretarako, ikuste-gailuaren menpe gaude eta teknologia muga da. Bereizmena handituz doa baina hala ere bide horrek ez du arazoa konpontzen; irudiak hobetu egingo ditugu, baina aliasing efektuarekin ezingo dugu bukatu.

2.4.1 Gain-laginketa

Hiru urratsetan egiten den prozesua da, urrats horiek ondokoak dira:

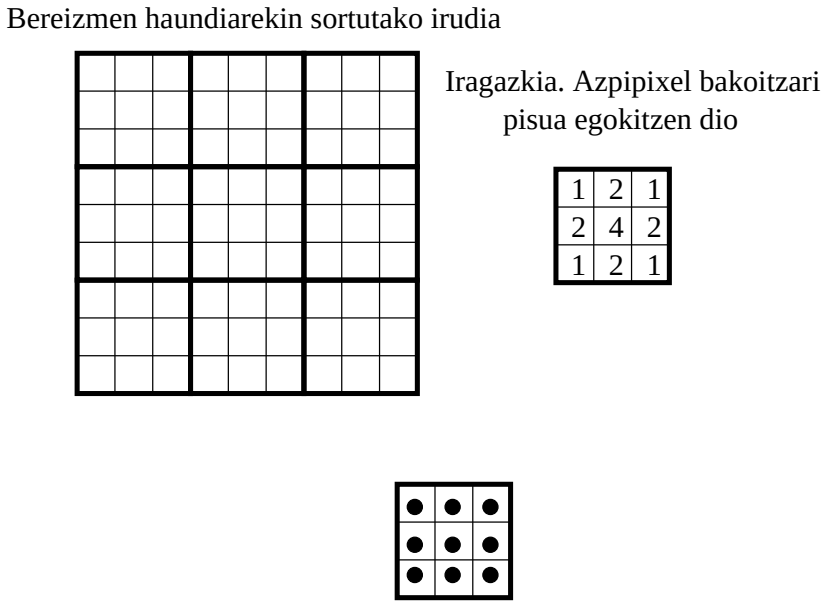
1. Irudi jarraiari ikuste-gailuaren bereizmena baino handiagoko laginketa eragiten zaio; hau da, dauzkana baino pixel gehiago dauzkala suposatuz irudi diskretua kalkulatzea izango litzateke, berezko pixel bakoitzeko $n \times n$ kalkulatuz.

2. Laginketa horri iragazkiren bat eragiten diogu.

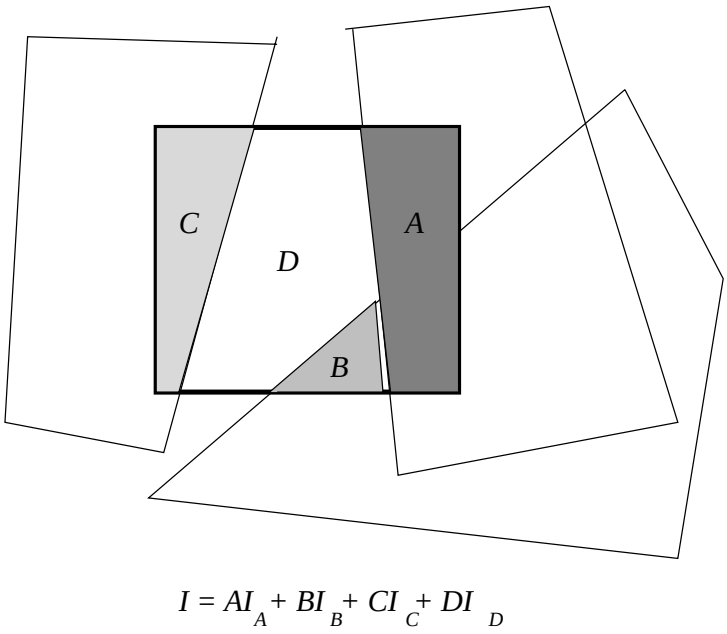
3. Horrela lortutako irudi diskretuarekin, ikuste-gailuari dagokion bereizmena duen irudi diskretua lortzen da. Azken bi urratsak batera egiten dira.

Ondorioz, azken irudiko pixel bakoitzaren balioa, bereizmen handiagoko irudiaren zenbait pixel erabiliz lortu da. Ohikoa da ikuste-gailua baino hiru aldiz handiagoko bereizmena erabiltzea, horrela pixel bakoitzerako 3×3 azpipixel erabil daitezke. Jakina, zenbat eta azpipixel gehiago kalkulatu, orduan eta lan handiagoa izango dugu gain-laginketari dagokion irudi diskretua kalkulatzean, kalkuluen denbora n^2 ordenan handitzen baita. Pixela kalkulatzeko, azpipixel bakoitzari pisu bat egoki diezaiokegu; hau da, pisudun iragazkia erabil daiteke.

Gain-laginketa metodoak n-ren balioan eta erabilitako iragazkiaren arabera aldatzen dira. Batzuek batezbestekoa kalkulatzen dute; hala ere, emaitza hobeak lortzen dira pixelaren zentroaren inguruko azpipixelei pisu handiagoa ematen dieten iragazkiekin. Adibide gisa, 2.11 irudian dagoena ikus daiteke. Pixel baten balioa kalkulatzeko, azpipixel bakoitzaren balioa iragazkiak egokitzen dion pisuaz biderkatu behar da, eta ondoren guztiak batu. Beraz, hori ere kontuan eduki beharreko eragiketa-multzo bihurtzen da. Adibidez 512×512 pixeleko irudia kalkulatzeko eta pixel bakoitzarentzat 5×5 azpipixel erabili badira, $512 \times 512 \times 25$ biderkaketa eta beste horrenbeste batuketa egin behar dira. Horrelako iragazkiak erabiltzean, irudia lausotzen dugu,



Irudia 2.11: Gain-laginketaren urratsak.



Irudia 2.12: Pixelean ager daitezkeen azpiazalerak.

gehiegizko aldaketak ekidinez. Zenbat eta iragazki zabalagoa erabili, orduan eta gehiago lausotuko dugu; baina, era berean, orduan eta aliasing gutxiago edukiko dugu. Bestalde, metodo horrek eskatzen duen memoria-tamaina asko handitu daiteke Z-buffer bidezko irudi kalkulurako, eta horrek ez du metodo horren alde jokatzen.

2.4.2 Azalera-laginketa

Algoritmo horren oinarrian pixelaren azalera baino txikiagoko objektuen geometria dago. Pixel bakoitzean irudi jarraiko zein azalera agertzen diren kalkulatzen da eta bakoitzak pixelaren azaleraren zein zati betetzen duen. Horrela pixel bati dagokion intentsitatea eta kolorea, bertan agertzen diren azalera-zatien kolore eta intentsitateen araberakoak izango dira; azalera gehien betetzen duenak eragin gehien izango du pixelean. Hori 2.12 irudian ikusten da.

Horrela azalduta, algoritmoak oso erraza dirudi, baina egin behar dituen kalkulu guztietan pentsatzen hasiko bagina, garestiegia dela konturatuko

ginateke. Kontuan izan behar dugu, pixel bati dagokion azaleraren arabera, hor barnean zein objektu ikusten diren erabaki behar dugula; hau da, pixelaren azaleraren mugen barnean zein objektu ikus daitezkeen erabaki behar da, ondoren, objektu horiek sakoneraren arabera ordenatu eta zeinek zein estaltzen duen erabaki. Metodo hori azkartzeko, algoritmo ezberdinak sortu izan dira. Horietako batek pixel baten azaleraren zatiak zein itxura izan dezakeen mugatuta dauka (pixel osoa hartzen duena, hiruki-itxurakoa...) eta bakoitzari pisu jakin bat egokitzen dio; horrela eginez, ez dauka azalera zatiak kalkulatzen hasi beharrik.

2.4.3 Laginketa estokastikoa

Metodo hori gure begien izaeraren ikerketen ondorioa dela esan daiteke. Gure begiek fotorrezeptoreak dauzkate, baina horiek ez daude uniformeki banatuta; hori dela eta, aliasing efektua gutxitu egiten da eta begi bistarekin askoz ere gutxiago gertatzen da. Erdialdean askoz gehiago daude eta hortik aldendu ahala gutxituz doaz. Metodo horren oinarria lagineko puntuen kokapena ausaz aldatzea da, eta bi urratsetan egiten du lan:

1. Irudiaren lagina lortu, baina lagineko puntu bakoitzaren kokapenari ausazko aldaketa egokituz.

2. Aurreko urratsean lortutako laginarekin iragazkiren bat erabiliz aldaketarik gabeko lagineko pixel bakoitzaren intentsitatea kalkulatu.

Ausazko aldaketaren ondorioz, berezko irudiari zarata gehitzen diogu; baina, era berean, aliasing efektua ekiditen dugu. Metodo horren bi urratsak direla eta, irudia kalkulatzeko era horretara egoki daitezkeen algoritmoetan erabili behar da. Kasu horretakoa da izpi-hedaketa izenaz ezagutzen den algoritmoa, irudi jarraian lan eginez lagina lor baitaiteke. Z-buffer edo lerroz lerroko algoritmoekin ausazko aldaketa eragitea, zaila gertatzen da, algoritmo horiek pixeletik pixelerako aldaketa uniformean oinarrituta baitaude. Hala ere, Z-bufferrean oinarritutako algoritmoak garatu izan dira, baina objektuak mikroazaleratan zatitzen dituzte, bakoitzak intentsitate konstantea duela suposatuz. Ondoren, mikroazalera bakoitzaren laginketa estokastikoa egiten da, eta pixel bakoitzarentzat laginketa horren bidez lortutako intentsitateak iragazi egiten dira.

Kapitulua 3

Objektuen adierazpideak

3.1 Sarrera

Objektuen itxura zein den jakiteko, adierazpide ugari garatu da, adierazpide bakoitzak bere abantailak eta desabantailak dituelarik. Beraz, kasu bakoitzean komenigarria den adierazpidea erabiltzeko, ondoko puntuak aztertuz egin daiteke aukera:

- erabil dezakegun hardwarea, algoritmo ezberdinen kodea eta datu-egitura bera.
- objektuak azken irudia lortu arte zeharkatzen duen prozesuaren kostua.
- objektuak azken irudian izango duen itxura (adierazpide batzuk besteak baino zehatzagoak dira).
- objektu baten itxura edo forma editatzeko erraztasuna edo sailtasuna.

Lau adierazpide-mota azalduko ditugu, erabiltzen den maiztasunaren eta garrantziaren arabera sailkatuta:

1. **Poligonozko adierazpidea.** Objektua aurpegiz osatzen da, horiek poligono planoak dira eta benetako objektuaren hurbilketa besterik ez dute adierazten.
2. **Polinomio kubikoen bidezko adierazpide parametrikoa.** Objektua, txatalez osatutako sare modura adierazten da.

3. **Solidoen geometria eraikitzailearen adierazpidea: CSG.** Objektua, oinarrizko objektuen arteko eragiketaz adierazten da.

4. **Espazio-partiketaren adierazpidea.** Objektua, okupatzen duen espazioaren bidez adierazten da. Espazioko puntu bakoitza objektuek okupatzen duten espazioaren arabera etiketatzen da.

Lehenengo eta laugarren adierazpideak erabiliz objektuaren itxuraren hurbilketa lortzen da. Ez da objektua modu zehatzean adierazten eta objektuaren gainazala ezagutzen da. Bigarren eta hirugarren adierazpideek berriz, objektuaren forma era zehatzean adierazten dute eta objektuek okupatzen duten bolumena ezaguna da.

Zenbaitetan funtzio matematiko inplizituak ager daitezke eta hori beste adierazpide bat bailitzan har dezakegu. Horren adibidea esferaren ekuazioa izan daiteke:

$$x^2 + y^2 + z^2 = r^2$$

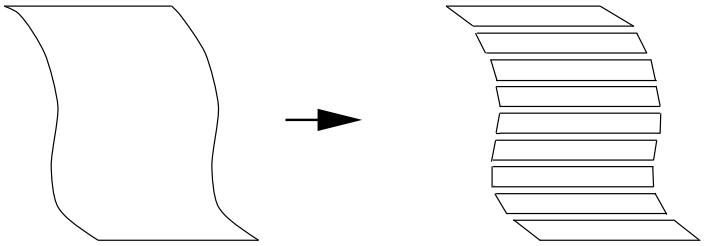
Baina horien erabilgarritasuna mugatua da, funtzio matematiko inplizituen bidez objektu gutxi adieraz daitezkeelako.

3.2 Poligonozko adierazpidea

Poligono-sarearen adierazpideari gehienetan **B-rep** (boundary representation) deitzen zaio, batez ere, objektuen gainazalaren deskribapen geometriko eta topologikoa delako.

Oro har objektuek forma kurbatuak edukiko dituzte eta adierazpide hori erabiliz poligono-sare batez hurbiltzen dira. Objektu kurbatu horien hurbilketen itxura kalitate txikikoa izan ez dadin, poligono asko erabili beharko dugu.

Interpolazioa erabiltzen duten itzaleztatze-algoritmoek, adierazpide horren erabilera bultzatu dute. Algoritmo horien bitartez, objektuak "biribiltzea"lortzen da, eta objektuaren irudi diskretua lortu ondoren ez da horrenbeste nabaritzen non hasten eta non bukatzen den aurpegi edo poligono bakoitza; hau da, emaitza onak ematen dituzte. Bestalde, poligonozko adierazpidea sinplea da eta bertan oinarritutako hardware-mailako inplementazio asko dago. Hori dela eta, askotan, nahiz eta erabiltzaileak objektuak eraikitzeko CSG edo adierazpide parametrikoa erabili, irudi diskretua sortzean objektu guztiak poligonozko adierazpidera itzultzen dira.



Irudia 3.1: Objektu kurbatu baten poligonozko hurbilketa.

Adierazpide horretan poligono bakoitza bere aldetik trata daiteke. Ezaugarri hori oso baliagarria gertatzen da, adibidez, itzaleztatze interpolatua eta Z-buffer algoritmoa aplikatzean.

Kasu errazenean, poligonozko sare bat poligonoz osatzen da, horiek erpin bidez adierazten dira eta erpinak (x, y, z) koordenatuak erabiliz. Beraz, objektua deskribatzeko, bera osatzen duten poligono eta erpinen zerrendan oinarritzen gara. Poligonozko sarearen egituran, geroago erabiliko den informazio gehiago ere gorde daiteke: poligonoen bektore normalak, erpinetako bektore normalak ... Horrela, asko erabiltzen diren bektore normalak, behin bakarrik kalkulatu beharko ditugu.

Objektua osatzen duten poligonoak modu hierarkiko batean antolatzea komenigarria da. Horrela, poligonoak gainazaletan eta gainazalak objektuetan taldekatuko genituzke. Taldekatze horretaz baliatuz, hurbilketaren eraginez sortutako ertzak eta errealitatean objektuaren parte diren ertzak bereiz ditzakegu. Eta eszenaren irudi diskretua sortzerakoan, ertz desberdinak era desberdinean tratatzeko posibilitatea izango dugu.

Poligonozko adierazpidearen beste hurbilketa bat, ertzen deskribapenean oinarritzen da. Hurbilketa horretan, objektuaren ertz bakoitzeko bi erpin eta ertza konpartitzen duten bi poligonoak gordetzen dira. Objektua modu horretan adieraziz, irudia sortzeko prozesua eraginkorragoa izango da, ertz bakoitza behin bakarrik tratatzen delako. Poligono bakoitza banaka tratatuko bagenu, bi poligonoren arteko ertza bi aldiz prozesatuko genuke, kostuaren ikuspuntutik horrek dakarren gain-kargarekin.

3.2.1 Poligonozko objektuak eraikitzen

Poligonozko objektuak eraikitzeko erabiltzen diren lau metodo azalduko ditugu. Lehenengo bi metodoek errealitateko objektuak poligonozko objektuetara itzuliko dituzte, ordenadorea ulertzeko gai izan dadin. Azken biek, berriz, objektuen definizioetan oinarrituz sortzen dituzte poligonozko objektuak.

Poligonozko objektuak eskuz eraikitzen

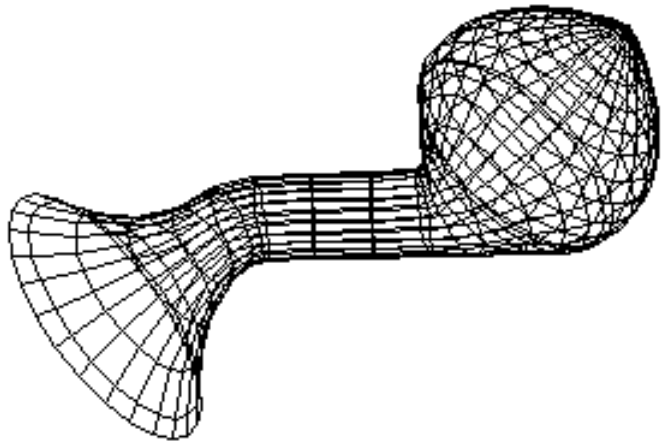
Errealitateko objektu baten poligonozko adierazpidea lortzeko modu errazena, hiru dimentsioko digitalizatzailea erabiltzea da. Digitalizatzailea maneiatzen duen pertsonak, poligonozko objektuaren erpinak izango diren puntuak margotu behar ditu errealitateko objektuan. Orduan puntuen hiru dimentsioko koordenatuak bidaltzen dira ordenadorera. Digitalizatzailea erabiliz lortuko den objektua nolakoa izango den jakiteko, objektu errealeko puntuz osatutako sare bat marraztea komeni da.

Poligonozko objektuak automatikoki eraikitzen

Errealitateko objektuen poligonozko hurbilketa zehatzak lortzeko baliagarria den periferiko bat, laser-aztertzailea da. Objektua gorantz eta beherantz mugitzen den eta bere buruaren inguruan biratzen dabilen mahai baten gainean jartzen da. Laser-izpiak, objektua biratuz doan bitartean horren gainazalarekin talka egiten du, eta laser-aztertzaileak izpiaren sortzeedo jatorri-puntutik talka-puntura dagoen distantzia neurtzen du. Biraketa baten ondoren, lortutako distantzia guztiak erabiliz, poligonozko objektua eraiki daiteke.

Poligonozko objektuak matematikoki eraikitzen

Deskribapen matematiko batean oinarrituz objektuak sortzeko bide errazena, objektuaren ebakidura bat bide batean zehar mugituz sortzea izango litzateke. Erabilitako algoritmoan erraz kontrola daiteke, poligonozko objektua sortzeko erabiliko den bereizmena. Hala ere, forma edo itxurarekiko menpekotasuna duten bereizmen-arazoak gerta daitezke. Toruaren kasuan, ebakidura zirkular bat bide zirkular batean mugituz sortuko litzateke, eta horrela sortutako toruaren kanpoaldeko aurpegiak, barnealdekoak baino handiagoak izango lirateke.



Irudia 3.2: Zirkunferentzia mugituz eta era berean neurria aldatuz eraikitako poligonozko objektu bat.

Poligonozko objektuak ebakidurak mugituz eraikitzen

Orokor dezagun aurreko puntuan azaldutako ideia. Esandako horri bi ezaugarri gehituz, itxura askotako objektuak lor daitezke:

1. Ebakidura edozein kurbatan zehar mugitzeko ahalmena izango dugu. Normalean, erabiltzaileak modu elkarreragilean, B-spline edo beste teknika batez baliatuz, definitu ahal izango ditu kurbak.

2. Bidean mugituz doan heinean, ebakiduraren itxura aldatu ahal izango dugu. Horren adibide bat 3.2 irudian ikus daiteke.

Metodo hori aplikatuz, poligonozko objektu asko sortzeko gai izango gara.

3.3 Txatalen sareak

Txatal bat puntu guztiak definitzen dituen gainazal kurbatua da. Txatala bi parametro dituen $Q(u, v)$ funtzio bat da, non $0 \leq u, v \leq 1$ eta Q polinomio kubikoa den. $Q(u, v)$ definitzeko, bere koefizienteak ezagutu behar ditugu, eta koefiziente horiek kalkulatzeko, kontrol-puntu izenaz ezagutzen diren 16

puntu erabiltzen dira. Kontrol-puntuek definitzen dute txatalak izango duen itxura.

Aplikazio gehienen kasuan, zailagoa da objektuak txatal bikubikoak erabiliz sortzea, hiru dimentsioko digitalizatzaileekin eraikitzea baino. Txatal bikubikoen kasuan, 16 kontrol-puntu definitu behar dira txatal bakoitzeko, eta adierazpidearen osotasuna mantentzeko, txatal bakoitza ezin da txatal auzokideekiko independenteki definitu.

Baina adierazpide horrek baditu zenbait abantaila, batez ere CAD aplikazioetan erabilgarria egiten dituztenak. Objektu bat sortzeko era, kontrol--puntuak mugituz eta txatal berria azalduz, intuitiboa da. Hala ere, objektu bat eraikitzea ez da dirudien bezain erraza, txatalen arteko jarraitasun--baldintzak mantendu behar dira eta.

Bestalde, txatal bikubikoen adierazpideari esker, objektu baten ezaugarriak kalkula ditzakegu: masa, bolumena, azalera ... Poligonozko adierazpidearen kasuan, objektuko erpin guztien koordenatuak gordetzen dira, eta txatal bikubikoen adierazpidean, berriz, nahikoa da txatal bakoitzeko 16 kontrol-puntuak gordetzea, memoria handia aurreztuz.

Poligonozko adierazpidean oinarritutako objektuen eszena batean, irudi diskretua lortzerakoan, urrun dagoen objektu oso konplexu bat pixel bakar batean proiektatzea gerta daiteke, kalkulu konplexu asko eginez irudian hain eragin gutxi izateko. Txatal bikubikoen bidez adierazitako objektuen kasuan, ordea, txatala azpitxataletan zatitzeko ahalmena daukagu, horretarako algoritmoak badaude-eta. Horrela, nahi adinako bereizmena lortzeko aukera daukagu. Objektu bat pixel bakar batean proiektatzen den azpitxataletan zatitzeko aukera ematen digu, eta beharrezkoak ez diren kalkuluak ekidingo genituzke; hau da, objektuaren bereizmena gure beharretara egokitzeko aukera dugu.

3.3.1 Objektuak txatal-sareen bidez eraikitzen

Txatal bikubikoen adierazpidean oinarritutako objektuak eraikitzeko, bi teknika dira nagusi:

- **Gainazalen egokitzea:** gainazalen interpolazioa burutzeko metodo partikular bat da. Erabiltzaileak eraiki nahi duen gainazalean egongo diren puntuekin, txatal bikubikoetan oinarritutako gainazal bat kalkulatuko dugu. Plano bateko puntu batzuk ezagutuz lerro bat kalkulatzearen parekoa da; hiru dimentsioko puntu batzuen koordenatuak

jakinik, gainazal bat kalkulatzen dugu. Teknika hori oso baliagarria da, objektu baten zenbait puntu digitalizatu ondoren gainazala lortzeko. Hala ere, ez da errealitateko gainazalaren guztiz berdina den gainazala lortuko. Zenbat eta puntu gehiago eman, orduan eta antz handiagoko gainazala lortuko dugu. Digitalizatzailetik lortutako puntuak erabiliz poligonozko adierazpidea erabiliko bagenu, lortutako gainazalak ez luke jarraitasun-maila handia izango; txatal bikubikoak erabiliz, ordea, jarraitasun-baldintzak betearazteko aukera daukagu.

- **Ebakiduren mugimendua:** Kurba kubiko bakar bat edo kurba kubikoen segmentu-multzo bat espazioan zehar mugituz, gainazal bat lortuko dugu. Espazioko zein bidetan mugitu behar dugun jakiteko, bidea deskribatuko duen kurba kubiko bat erabiliko dugu. Ebakidura adierazten duen kurba, mugimendua adierazten duen kurban zehar mugituko dugu, eta horretarako urrats-kopuru jakin bat eman beharko dugu. Urrats bat ematen dugun bakoitzean, sortzen den gainazalari dagokion txatal bikubikoa kalkulatu beharko dugu.

Txatalaren adierazpidearen desabantailak:

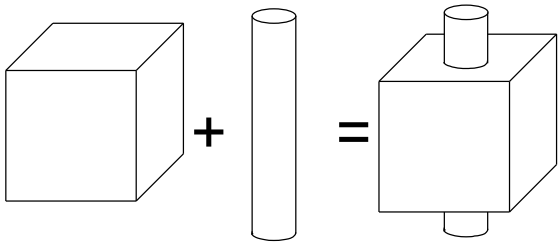
- Aplikazio gehienetan, hiru dimentsioko objektu baten itxura deskribatzen duen datu-egitura eraikitzea, zailagoa da txatal bikubikoak erabiltzean.

- Txatal bakoitzeko 16 kontrol-puntu finkatu behar dira. Eta horrez gain, adierazpidearen osotasuna mantentzeko, jarraitasun-baldintza batzuk mantendu behar dira txatalen muga guztietan. Txatal bat definitzerakoan, ezin gara horren auzokideez ahaztu.

CAD sistemetarako aproposak diren abantailak:

- Dagoeneko definituta dagoen objektua, modu elkarreragilean alda dezakegu, edozein kontrol-punturen kokapena aldatuz. Hala ere, aldaketak burutzea ez da dirudien bezain erraza, jarraitasun-baldintzak bete behar dira-eta.

- Adierazpide horrek, objektuaren itxura zehatza deskribatzen du. Horri esker, objektuaren zenbait propietate kalkula daitezke: masa, bolumena, azalera, inertzi momentua, ...



Irudia 3.3: Batuketaren eragiketa boolearra.

- Objektu baten itxura modu zehatz batean eta, adibidez, poligonozko adierazpidea erabiliz gorde nahiko bagenu, izugarrizko memoria-kantitatea beharko genuke, irudia sortzean memoria-atzipen asko eginez eta, ondorioz, irudia sortzeko prozesuaren abiadura txikiagotuz. Objektu bera txatalen bidez adieraziz, arazo horiek (memoriaren beharra eta hori atzitzeak dakarren abiadura galera) ekidingo genituzke.

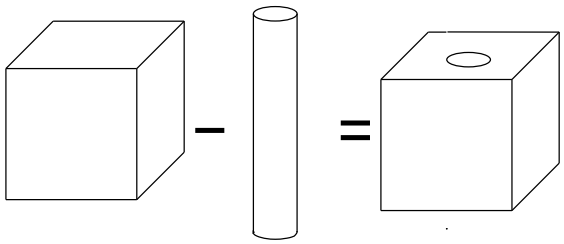
3.4 Solidoen geometria eraikitzailea edo CSG

Adierazpide horretan objektu bat zuhaitz batez adieraziko dugu. Zuhaitzaren hostoek oinarrizko objektu geometrikoak adierazten dituzte; zuhaitzaren barne-erpinek, berriz, oinarrizko objektuen artean egin behar diren eragiketak. Oinarrizko objektu geometrikoak esferak, konoak, zilindroak, kutxak etab. izan daitezke, eta horiei aplikatuko dizkiegun eragiketak, aldaketa linealak eta eragiketa boolearrak izango dira. Zuhaitzean objektu bat sortzeko eman diren urrats guztiak gordetzen dira.

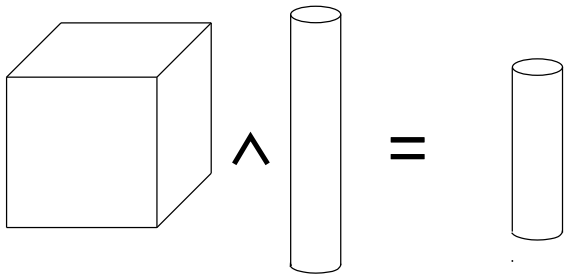
3.3, 3.4 eta 3.5 irudietan bi objekturekin egin daitezkeen eragiketa boolearrak azaltzen dira. Bi objektuen arteko batuketak, bi objektuek dituzten puntu guztiak izango dituen objektu berri bat sortuko du. Bi objektuen arteko kenketa egitean, kenkizuna den objektuari kentzailea den objektuaren puntu guztiak kenduko dizkiogu. Azkenik, $\wedge$ eragiketa boolearra aplikatzean, bi objektuetan dauden puntuek osatzen duten objektua lortuko dugu.

Bi kutxa eta zilindro batez sortutako objektu baten zuhaitza ikus daiteke 3.6 irudian.

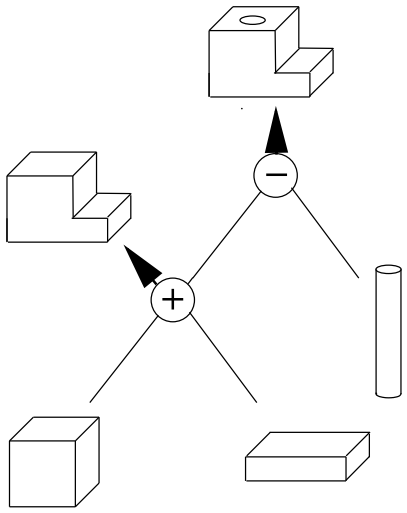
Adierazpide horrek ere badu zenbait desabantaila. Horrela adierazitako objektu baten irudi diskretua sortzeko, denbora luzea behar da. Bestalde, eragiketa boolearrak globalak dira, hots, objektu osoan dute eragina. Eragi-



Irudia 3.4: Kenketaren eragiketa boolearra.



Irudia 3.5: Ebaketaren eragiketa boolearra.



Irudia 3.6: Hiru objekturen artean eragiketa boolearrekin lortuko genukeen objektuaren CSG adierazpena.

keta boolear lokalak (bi objekturen zatien artean aplikatzen direnak) inplementatzen oso zailak dira. Arrazoi horregatik, gehienbat B-rep adierazpidea erabiltzen da, eta horretan eragiketa boolearrak egiteko aukera ematen da.

3.5 Espazioaren partiketa-teknikak

Teknika horietan espazioko puntu guztiak etiketatzen dira, puntu bakoitza zein objektuk okupatzen duen adieraziz. Espazio osoa *voxel* izenez ezagutzen diren kutxatxotan (horiek okupatuko duten bolumena erabili nahi dugun bereizmenaren menpe egongo da) zati dezakegu, puntu bakoitza etiketatu beharrean voxelak etiketatuz. Argi dago, eskema horri jarraituz memoria asko beharko dugula. Eta horrenbeste memoria ez erabiltzeko, voxeletan oinarritzen den zenbait egitura garatu da.

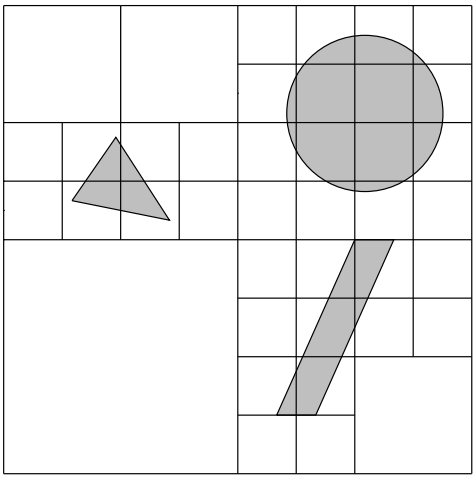
Egitura horiek, datu-egitura laguntzaile modura erabiltzen dira. Zenbait kasutan errazagoa da datu-egitura hori erabiliz irudia sortzea; adibide gisa CSG kasua aipa daiteke. Horregatik, askotan, eszenari dagokion irudia sortu aurretik, itzulketa-prozesu bat ematen da, objektuak CSG adierazpidetik datu-egitura laguntzailera pasatuz.

Geroago ikusiko dugun bezala, datu-egitura hori oso baliagarria gertatzen da izpi-hedaketan.

3.5.1 Octreeak

Octree bat objektuek okupatzen dituzten eskualde kubikoak deskribatzen dituen datu-egitura hierarkikoa da. Aipatu dugun bezala, eskualde kubiko horiei voxel deitzen zaie. Octree bat nolakoa den erakusteko, quadtree baten adibide bat azalduko dugu. Azken hori octreearen berdina da baina bi dimentsiotan; ondorioz, azalpenetarako egokiagoa.

Objektu sinple batzuk dituen bi dimentsioko eskualde bat eta horri dagokion quadtree adierazpenak, 3.7 eta 3.8 irudietan ikus daitezke. Zuhaitza sortzerakoan, eskualde guztia barneratzen duen karratu batekin (hiru dimentsioen kasuan kutxa bat izango litzateke) hasiko gara. Eskualde hori zuhaitzaren erroarekin adierazten da. Eskualde horren barnean objektuak daudenez, eskualdea lau azpieskualdetan zatitzen da; eta hori erroaren seme diren lau erpinek adieraziko dituzte. Hiru dimentsiotan egongo bagina, kutxa zortzi azpikutxatan zatituko genuke, zuhaitzean zortzi seme sortuz; hortik dator octree izenaren zergatia. Horrela, objekturen bat daukan edo-



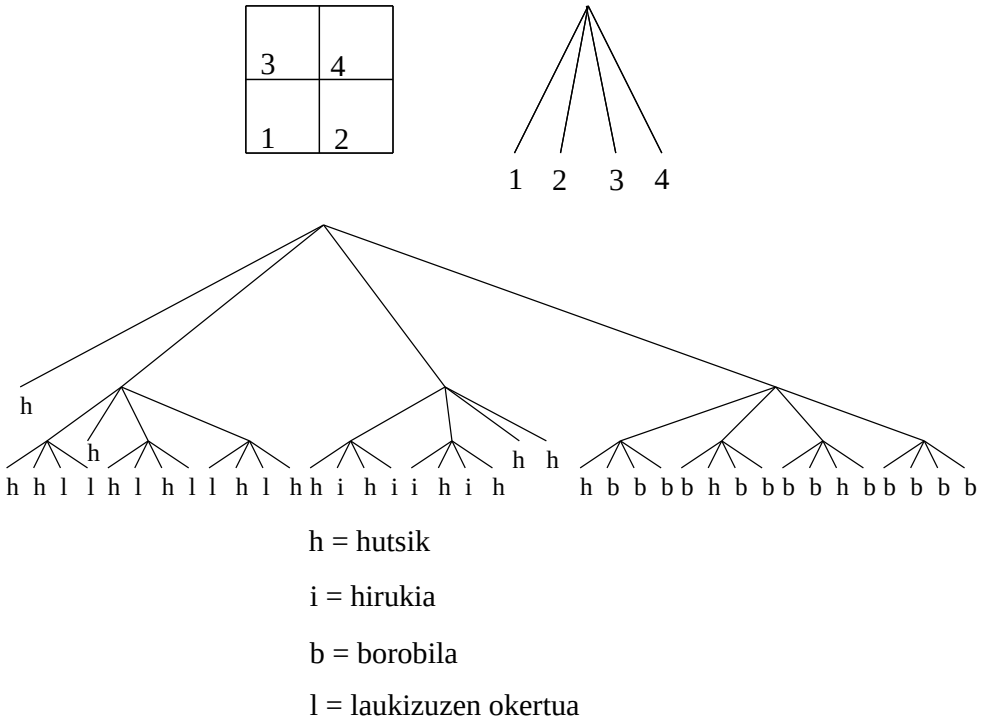
Irudia 3.7: Hiru objektuk bi dimentsiotan eragindako zatiketa.

zein azpieskualde zatitzen segituko genuke, azpieskualdearen neurria minimo bat izatera heldu arte, hau da, finkatutako bereizmen-mailara iritsi arte. 3.7 irudiko adibidean erabilitako bereizmena, hirukoa izan da. Lauko bereizmena erabili izan bagenu, zirkunferentziak zeharo estaltzen duen eskualdea (zuhaitzeko 37. hostoa) ez genuke zatitu beharko; hutsik ez dauden beste eskualde guztiak, ordea, bai.

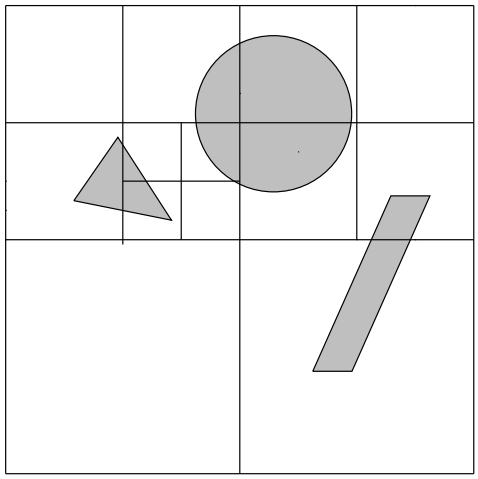
Beraz, bi hosto-mota ager daitezke zuhaitzean. Alde batetik objekturik gabeko azpieskualdeei dagozkienak eta bestetik neurri minimora heldu diren azpieskualde okupatuak adierazten dituztenak. Bi dimentsioko kasuan, objektu baten barnean dagoen eskualdea ez da okupatu modura adierazten. Eskualde bat zatitu behar ote den jakiteko, objektuaren gainazaletan oinarritzen gara, eta quadtree bidez objektuaren gainazalak adierazten ditugu. Hiru dimentsiotan lan egingo bagenu, objektuak beren gainazalen bidez adieraziko genituzke eta gainazalaren zatiak leuzkaketen eskualdeak bakarrik zatituko genituzke.

Eszena octree bidez adierazitakoan, espazioaren zatiketa bi modutan egin daiteke:

1. Goian azaldutako zatiketa-eskemari jarrai diezaiokegu. Baina bereizmen handia erabiltzean, hau da, eskualde oso txikiak erabiltzean, eszena konplexu batean adibidez, zuhaitzak memoria handia beharko luke.



Irudia 3.8: Bi dimentsioko eszenaren quadtree adierazpena. Eszena hiru dimentsiokoa balitz, octree bat erabiliko genuke.



Irudia 3.9: Eszenako eskualde baten barnean objektu bakar bat dagoenean, zatiketa-prozesua gelditu egiten da.

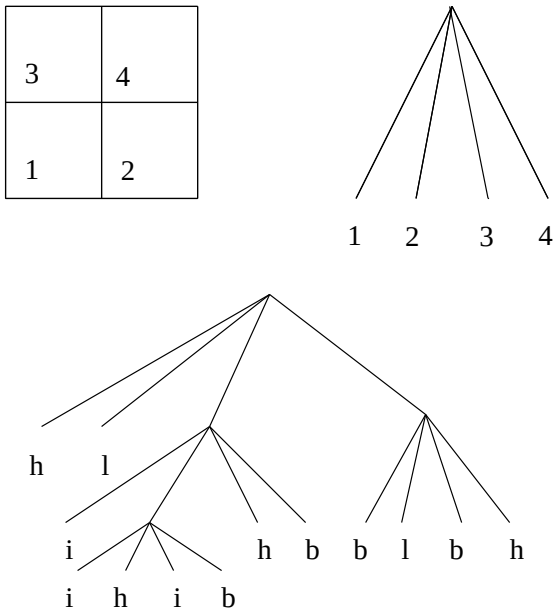
2. Orain arte erabilitako adierazpidean, guztiz okupatuta edo libre dagoen eskualdera iritsi arte egiten da eskualde-zatiketa. Baina, objektu bakar batek guztiz edo partzialki okupatzen duen eskualde bat aurkitzean geldi dezakegu eskualde-zatiketa, memoria asko aurreztuz. Aurreko 3.7 irudian zatiketa-estrategia berria aplikatuko bagenu, lortuko genituzkeen emaitzak 3.9 eta 3.10 irudietakoak izango lirateke.

Ideia orokortuz, hosto bakoitzak poligono batek edo gehiagok betetako eskualdea adieraziko luke, eta hosto bakoitzean poligonoen erakusle-zerrenda izango genuke.

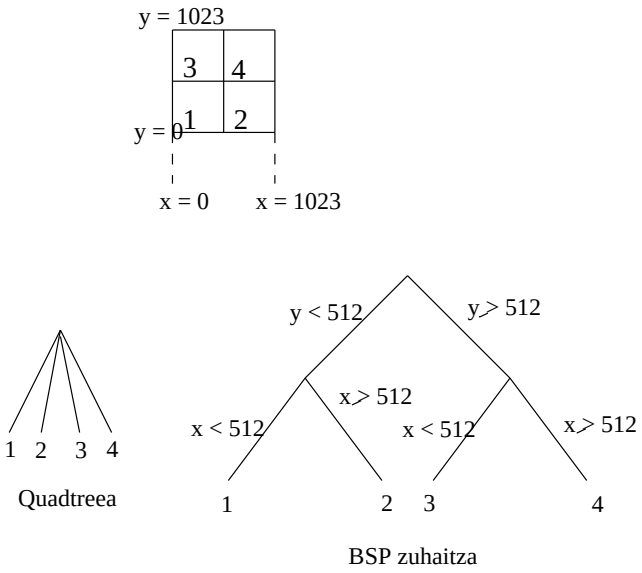
3.5.2 BSP zuhaitzak

BSP egitura octreearen ideian oinarritzen da. 3.11 irudian maila bateko zatiketa adierazten duten quadtreea eta BSP (binary space partition) zuhaitzak ikus daitezke. BSP zuhaitzeko hostoa ez den erpin bakoitzak, espazioa bi zatitan zatitzen duen plano bat adierazten du. Eta adierazten duen eskualdea ebakitzen duten poligonoen erakusleen zerrenda bat jasoko luke hosto bakoitzak.

Izpi-hedaketan oso eraginkorra da BSP egitura erabiltzea. Izpiari dagokion ibilbidea zein den jakinik, erraza izango litzateke zuhaitzean zehar



Irudia 3.10: Zatiketa-estrategia berria aplikatuz lortzen den zuhaitzean, hostoek objektura doazen erakusleak gordetzen dituzte.



Irudia 3.11: Bi dimentsioko eszena baten maila bateko zatiketaren quadtree eta BSP adierazpenak.

mugitzea, izpiak zeharkatzen duen eskualdea aurkitu arte; eta eskualdea aurkitu ondoren, bertan dauden objektuekin ebaketa-testak egin beharko lirateke.

BSP zuhaitzaren moldaerazko zatiketa

Aipatu dugun bezala, BSP zuhaitzeko erpin bakoitzak espazioa bi zatitan banatzen duen plano bat adierazten du. Beraz, zatiketa ez da laukitan egin beharrik. Espazioa zatitzeko erabiltzen diren planoak, edozein izan daitezke. BSP zuhaitzak Fuchs-ek (1980) erabili zituen lehen aldiz, eta bere helburua ikuspuntu bat emanik eszenako poligono guztiak atzetik aurrera ordenatzea izan zen. Zatiketa egiteko erabili zituen planoak, eszena osatzen zuten poligonoek definitutakoak izan ziren, eta beren kokapena edozein izan zitekeen.

Planoen kokapena edozein izan daitekeela jakinik, moldaerazko zatiketaren teknika azalduko dugu. Bi dimentsioko bi kasu aztertuz, teknika horren erabilera eta abantailak hobeto ulertuko dira. Bi kasuak azaltzen dituzten 3.12 eta 3.13 irudietan, a-tik p-ra etiketatutako 16 objektu dituen eskualde

baten espazio-zatiketa egiteko erabili diren bi hurbilketa ikus daitezke.

3.12 irudian BSP adierazpidea erabiliz adierazten den quadtree-zatiketa azaltzen da. Zuhaitz horren sakonera-maila maximoa zortzikoa da, eta zortzi izango litzateke, puntu bat zein eskualdean dagoen identifikatzeko beharko litzatekeen bilaketaren luzera maximoa.

3.13 irudiko zuhaitza eraikitzeko, moldaerazko zatiketa erabili da. Partiketa-lerroa aukeratzerakoan, bi aldeetara objektu-kopuru berdina dituena erabili dugu. Estrategia horren ondorioz, orekatuago dagoen zuhaitza lortzen da, eta zuhaitzaren sakonera txikiagoa izaten da. Kasu horretan, bilaketaren luzera maximoa laukoa da.

3.6 Irudia sortzeko estrategiak

Ondoren, objektuen adierazpide bakoitzerako irudiak sortzeko erabiltzen diren algoritmo eta estrategiak azalduko ditugu.

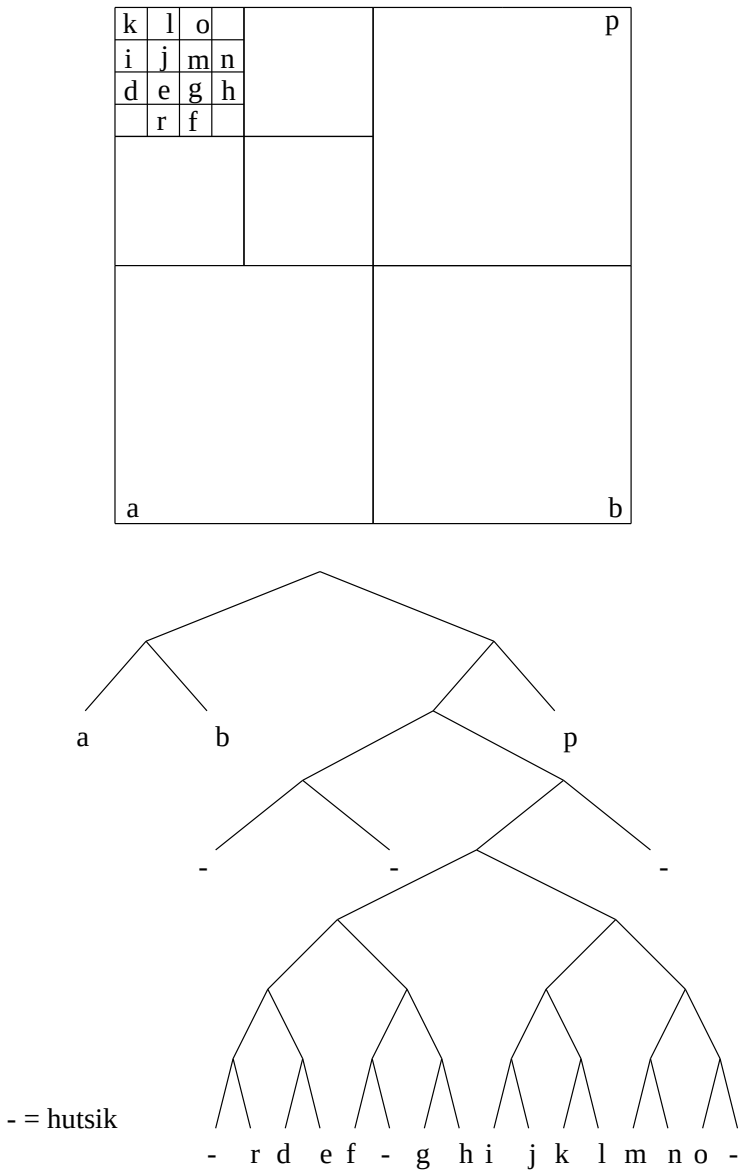
3.6.1 Poligonozko objektuen irudia sortzen

Poligonozko objektuen irudia sortzean, poligono-zerrenda bat jasotzen dugu sarrera gisa eta kalkulatu beharreko irteera, irudiko pixel bakoitzaren kolorea izango da. Irteera hori lortu aurretik, zenbait urrats eman beharko ditugu, eta objektuak erreferentzi sistema desberdinetatik pasako ditugu. Igarotako erreferentzi sistema bakoitzean, zenbait eragiketa egingo dugu objektuetan.

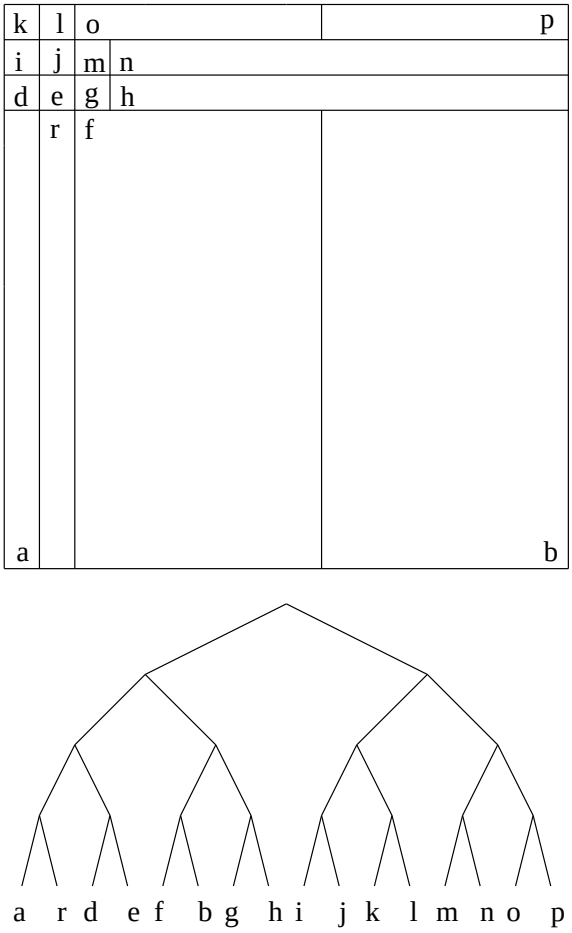
Ondoren, irudia sortzeko prozesuaren adibide bat azalduko dugu. Azalpena jarraitzeko, begira itzazu 3.15, 3.16 eta 3.17 irudiak.

Objektuen datu-basean, objektu bakoitzaren erpinen koordenatuak eta beste zenbait informazio gordetzen da; erpin horiek erreferentzi sistema lokal batean adierazi ohi dira. Objektu bakoitzak bere erreferentzi sistema dauka, eta komenigarria izango litzateke bertako jatorria objektuaren puntu bat izatea, gure adibidearen kasuan kuboaren erpin bat adibidez.

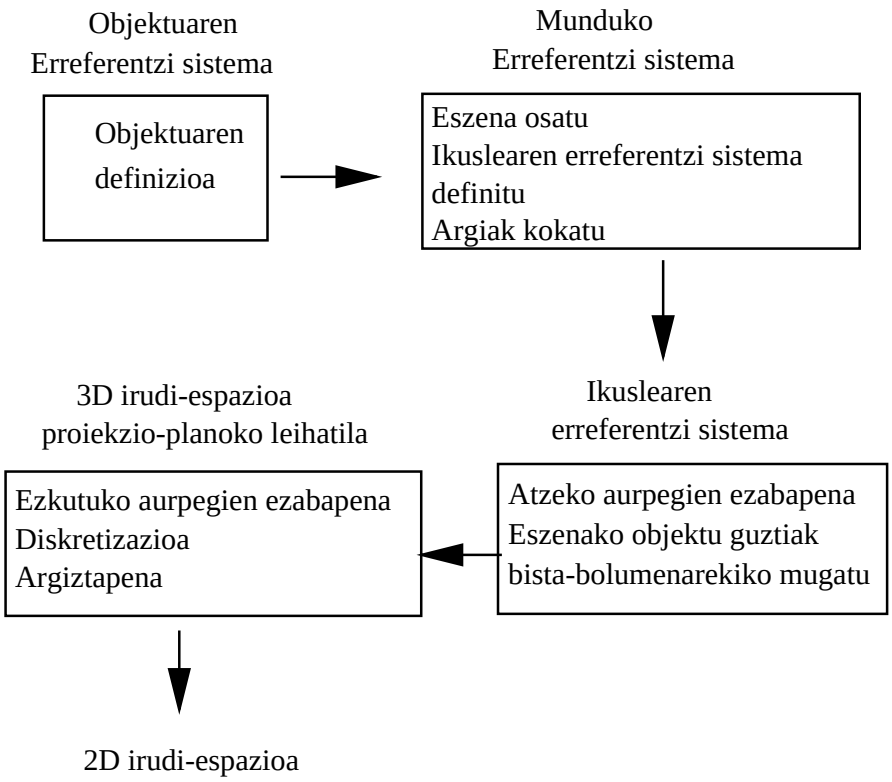
Eszena eraikitzeko, objektu bakoitzaren erreferentzi sisteman dauden erpinak munduko erreferentzi sistemara pasa behar dira; horrela, objektu guztiek erreferentzi sistema berdina izango dute. Erreferentzi sistema horretan argi-iturrien kokapenak finkatzen dira. Bestalde, ikuslearen kokapena eta nora begiratzen duen jakinik, ikuslearen erreferentzi sistema kalkulatzen da.



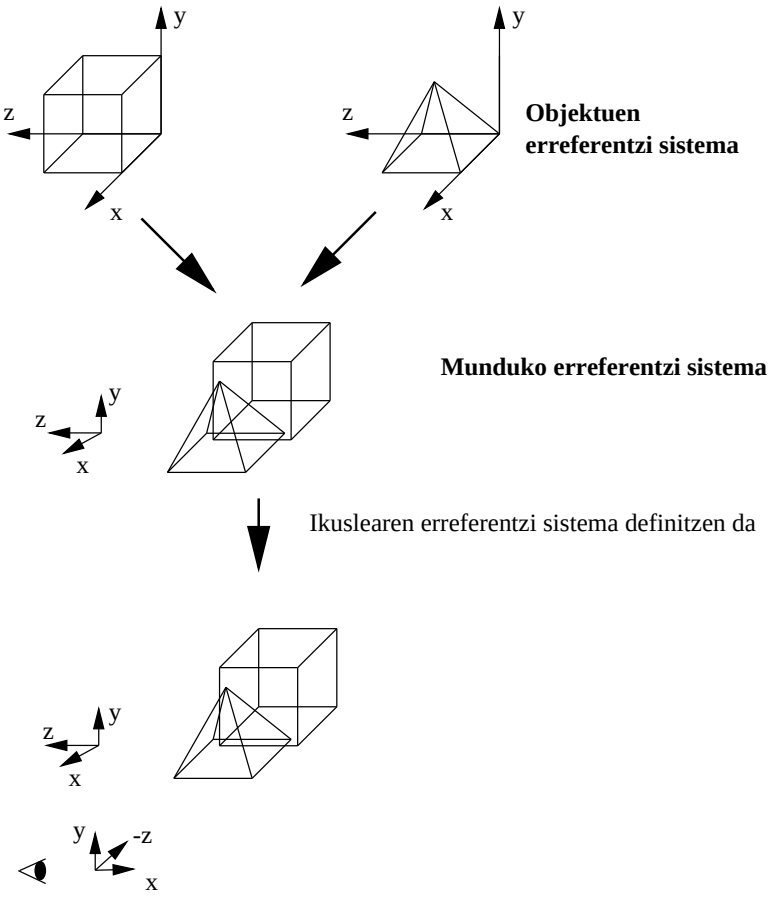
Irudia 3.12: Eszenaren zatiketa arrunta. Zuhaitzaren sakonera maximoa zortzikoa da.



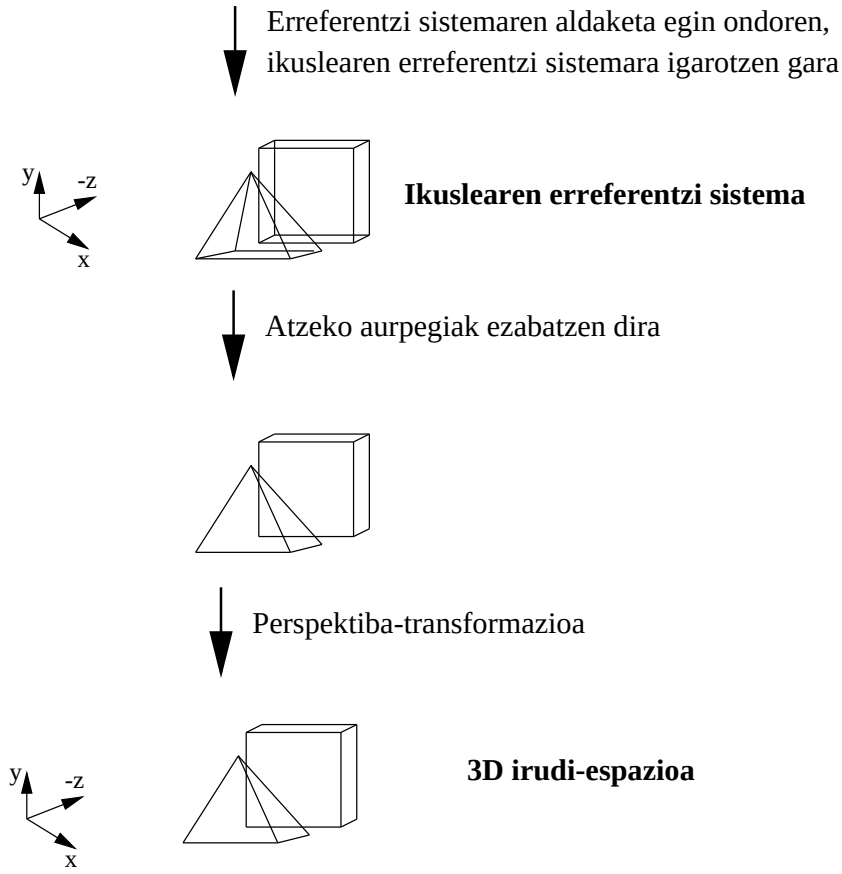
Irudia 3.13: Eszenaren moldaerazko zatiketa. Zuhaitzaren sakonera maximoa laukoa da.



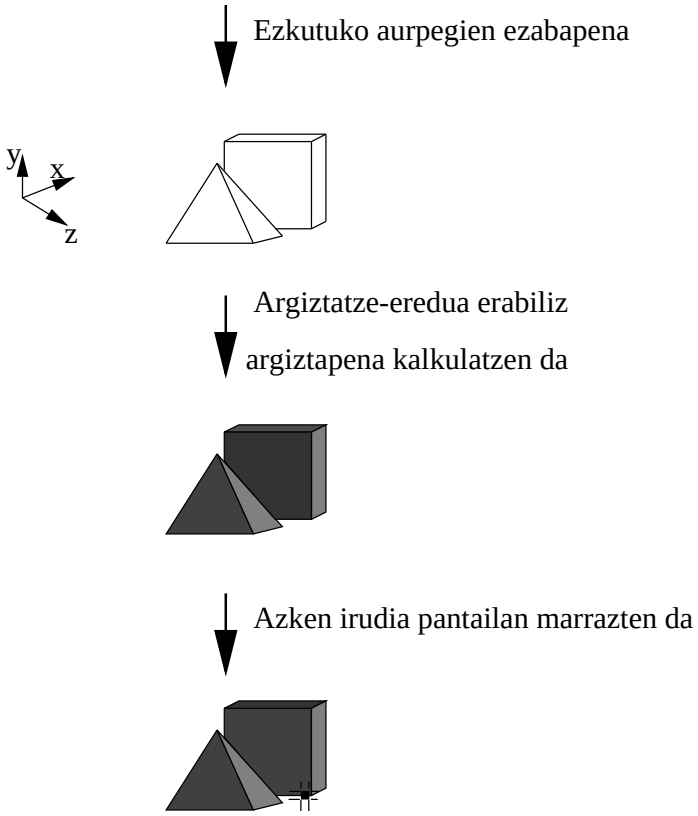
Irudia 3.14: Irudia sortzeko prozesua.



Irudia 3.15: Irudia sortzeko prozesuaren adibidea.



Irudia 3.16: Irudia sortzeko prozesuaren adibidearen jarraipena.



Irudia 3.17: Irudia sortzeko prozesuaren adibidearen jarraipena.

Ondoren, munduko erreferentzi sisteman dauden objektu guztiak ikuslearen erreferentzi sistemara pasa behar dira. Erreferentzi sistema horretan atze-aurpegien ezabapena egiten da, ikusleak ikusi ezin dituen poligonoak ezabatzean datza. Atze-aurpegien ezabapenaren eta ezkutuko aurpegien ezabapenaren arteko desberdintasuna 3.16 eta 3.17 irudietako bigarren eta lehenengo urratsetan ikus daiteke hurrenez hurren. Atzealdeko aurpegien ezabapena aplikatuz, objektuak banaka tratatzen dira, eta ikusleak ikusten ez dituen poligonoak ezabatzen dira. Hala ere, ikuskorra den objektu baten A poligono bat, ikuskorra den beste objektu baten B poligonoaren aurrean egon daiteke. Beraz B poligonoak A poligonoa guztiz edo partzialki ezkutatuko luke, eta B ikusi beharko litzateke. B poligonoa ikusi behar dela jakiteko erabiltzen dira ezkutuko aurpegien ezabapen-algoritmoak.

Egingo diren azken bi eragiketak itzaleztatzea eta irudi diskretua sortzea izango dira. Itzaleztatze-prozesuan, poligono bakoitzak argiarekiko duen orientazioa aztertuz, poligonoko gainazalari edo puntuei kolorea esleitzen zaie. Irudi diskretua sortzeko prozesuan, proiekzioko pixel bakoitzari dagokion kolorea erabakitzen da.

Deskribatutako lehenengo lau urratsak irudia sortzeko prozesu guztietan egiten dira, poligonozko adierazpidea erabiltzen badute behintzat. Azken eragiketak era desberdinetan inplementa daitezke, ezkutuko aurpegien ezabapena, itzaleztatzea eta irudi diskretua sortzeko betebeharrak burutzeko algoritmo asko baitago.

Azaldu ez dugun eta egiten den prozesu bat, hiru dimentsioko mugaketa da. Ikusleak ikusten duena, piramide moduko bolumen batean barnera daiteke, piramide horretatik kanpo dagoen guztia ikustezina izanik. Piramide horrek adierazten duen bolumenari, ikuste-bolumena deritzo. Beraz ikuste-bolumenetik kanpo dauden objektu guztiak ezaba daitezke. Prozesu hori, eszenako objektu guztiak ikuslearen erreferentzi sisteman daudenean egiten da.

3.6.2 Txatal parametrikoen sarearen irudia sortzen

Txatal bikubiko parametrikoei dagokien irudia sortzeko biderik onena, edo errazena behintzat, txatal-sareak poligono adierazpenera itzultzean datza.

Dagoeneko bereizmen handiko adierazpen bat erabiltzen ari bagara, zergatik egin itzulpen hori? Bi arrazoi daude. Baliteke sare parametrikoen adierazpidea erabiltzearen arrazoia modelatzailea erabiltzailearen parte hartzeaz

baliatzen delako izatea, elkarreragilea delako. Kasu horretan poligonozko adierazpidea edo adierazpide parametrikoa, erabiltzailearentzat berdina da; aplikazioa bera da, zailtasunak ekiditeko txatal bikubikoetara jotzen duena. Eta bestalde, bereizmenaren arazoa ebazteko erraza da. Gainazal bat deskribatzen duen sare parametriko zehatz bat edukita, horren itzulpena guk erabakitako neurriko poligono-sarera egingo litzateke, hurbilketa oso zehatzak lortzeko aukera izanik. Hurbilketaren bereizmena guk finkatuko genuke. Eta sare parametrikoa poligonozko adierazpidean izanik, horri dagokion irudia sortzeko, nahikoa izango litzateke poligonoetan oinarritutako algoritmoak aplikatzea.

Orokorrean, hurbilketaren poligono-kopurua sareko txatal-kopurua baino handiagoa izango da.

Txatal-sare bat poligonoetara itzultzeko egin behar den prozesua, bi urratsetan bana daiteke. Lehenengo urratsean, txatala Bezier motakoa ez bada, Bezier motara itzuli behar dugu. Itzulpen hori, matrize-biderkaketen bidez egiten da.

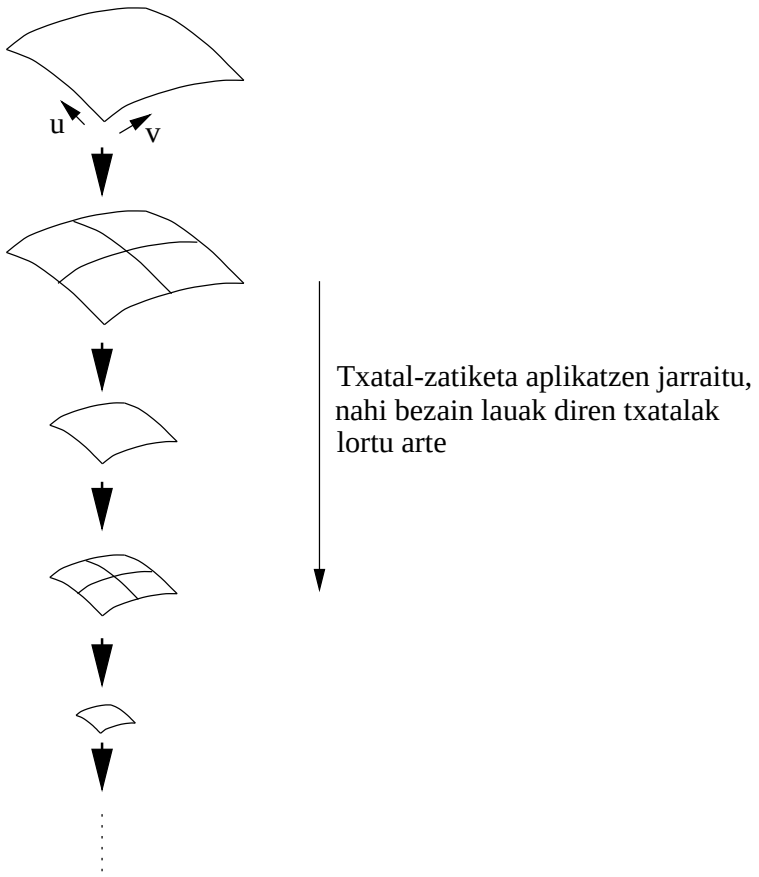
Bigarren urratsa txatal-sarea zatitzea da. Txatal bakoitzaren zatiketa, lautasun-maila jakin bateraino iritsi arte aplikatzen da. Txatalaren zatiketa erabakitako lautasun-maila batera iristean gelditzen da, 3.18 irudian dakusagun bezala. Zatiketa egiteko, txatal bakoitza lau azpitxataletan zatitzen da. Lau txatal horiek kurba isoparametrikoak "marraztuz"$(u = 0.5, v = 0.5)$ lor daitezke. Txatal baten lautasuna kalkulatzeko, nahikoa da txatalaren ertzetako erpinetatik igarotzen den planoaren eta txatalaren arteko distantzia kalkulatzea.

3.6.3 CSG adierazpideko objektuen irudia sortzen

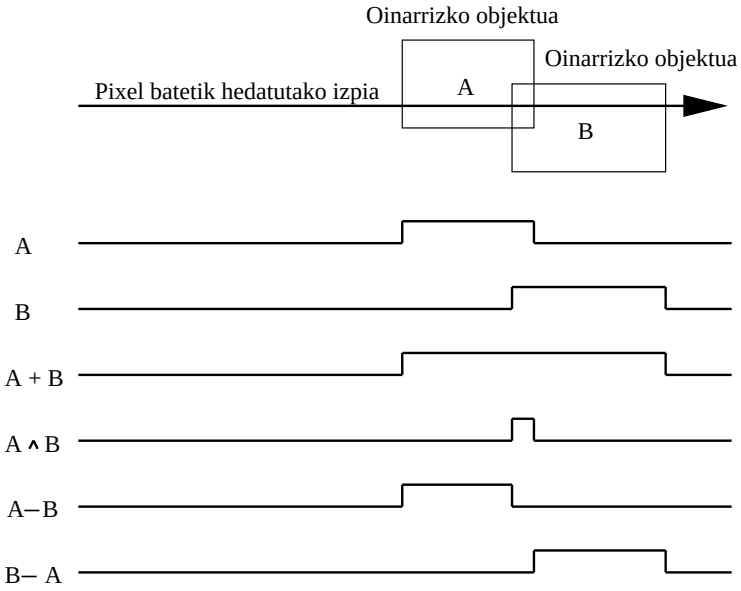
CSG adierazpidea erabiliz elkarreragilea eta intuitiboa den objektu eraikitzailea programa daiteke. Abantaila horiengatik ordaindu beharreko prezioa, irudia sortzeko estrategia konplexu eta garestia da.

Arazoa CSG objektu baten gainazal edo mugak lortzean datza. Horretarako, hiru teknika garatu dira:

1. CSG izpi-hedaketa.
2. Objektuak CSG adierazpidetik, voxel adierazpidera itzuli eta horri dagokion irudia sortzea.
3. Z-buffer algoritmoa erabiltzea.



Irudia 3.18: Txatal-zatiketa delako prozesua.



Irudia 3.19: Eragiketa boolearren aplikazioa, objektu batzuk ebakitzen dituen izpi baten gain.

Lehenengo hurbilketaren azalpen labur bat emango dugu, eta azalpena egiteko dimentsio bakar batean oinarrituko gara; gainera, proiekzio paraleloa erabiltzen dugula suposatuko dugu, azalpenaren argitasuna mantentzeko, batez ere. Proiekzio paraleloa erabiliko dugunez, hedatuko ditugun izpiak beren artean paraleloak izango dira. Izpi-hedaketa CSG adierazpidean nola aplikatzen den ulertzeko, izpi bat hedatuko dugu. Objektu baten oinarrizko objektuen eta izpiaren arteko testak burutu beharko dira. Ebaketa guztiak z sakontasunaren arabera ordenatuko ditugu. Horrela, oinarrizko objektuen sailkapena izango dugu izpi bakoitzeko. Orain, 3.19 irudian ikus daitekeen bezala, oinarrizko objektuen artean egiten diren eragiketa boolear berberak, izpien eta oinarrizko objektuen ebaketen artean egin behar dira.

Kalkulatzen ari garen pixela objektu bati badagokio, argiztatze-eredua aplikatuko da.

Baina, izpi-hedaketa CSG objektuen gainean horrela aplikatzea, garestiegia gertatzen da. Bukatzeko, ohar zaitez metodo horrek eredu bakar batean integratzen dituela ondoko kontzeptuak:

1. CSG adierazpidean oinarritutako objektu baten mugen ebaluazioa.

2. Ezkutuko aurpegien ezabapena. Ikuslearengandik gertuen dauden bi oinarrizko objektuen ebaketak kontsideratzen dira soilik.

3. Itzaleztatzea.

Proiekzio paraleloa aplikatu dugunez, CSG deskribapenaren espaziotik abiatuz, zuzenean pantailako pixela marrazteko gai gara, ikuste-espazioaren konplexutasun guztiak alde batera utziz.

Kapitulua 4

Gainazal eta kurben adierazpena

4.1 Sarrera

Ikusi dugunez, poligono bidez adierazitako zenbait objekturen kasuan, datuak (erpin guztien koordenatuak, aurpegien egitura,...) gordetzeko memoria handia behar da, eta objektuan aldaketaren bat eragitean, aldaketa hori gainazaleko puntu guztietan eragiten da, eraginkortasuna galduz. Arazo hori gainditzeko, txatal bikubiko parametrikoak azalduko ditugu.

4.2 Bi dimentsioko kurben adierazpena

Objektuak edo kurbak, ardatzen eta objektuko puntuen koordenatuen arteko erlazioa finkatzen duten ekuazioen bitartez adierazten dira. Ekuazioak hiru motatakoak izan daitezke:

1. Ekuazio esplizitua: $y = f(x)$
2. Ekuazio inplizitua: $f(x, y) = 0$
3. Ekuazio parametrikoak: $x = f(t)$ eta $y = f(t)$

Ekuazio esplizitua: $y = f(x)$ motako ekuazioa da, $f(x)$ jarraia eta deribagarria. Objektuko puntu bati dagokion (x, y) bikotea lortzeko, x aldagaiari balioak emango dizkiogu.

Desabantailak:

- Kurba itxiak edo x bakarrerako y-ren bi balio edo gehiago dituzten kurbak ez ditu onartzen.
- Aldaketa afinak eragin ondoren ez dute azpiespazio berdina mantentzen: $y = x^2$ ekuazioak adierazten duen kurba $90°$ biratu ondoren, lortutako kurba adierazten duen ekuazioa $y = \pm\sqrt{x}$ da.

Ekuazio inplizitua: $f(x, y) = 0$ motako ekuazioen bidez kurba itxiak adieraz daitezke eta erraza da puntu bat kurbarena den edo ez jakitea.

Desabantailak:

- Kurbaren puntuak lortzea zaila da.
- Aldaketa afinak eragitean $f(x, y)$ funtzioari dagokion azpiespazioa alda daiteke.

Ekuazio parametrikoak: Ekuazio horiek erabiliz, kurbaren puntuak kalkulatzea erraza da eta aldaketa afinek, $x(t)$ eta $y(t)$ polinomikoak direnean, ez dute horien azpiespazioa aldatzen.

Kurba bera ekuazio desberdinekin adieraz daiteke:

$$x = 2\cos t \qquad\qquad x = 2\frac{1-t^2}{1+t^2}$$

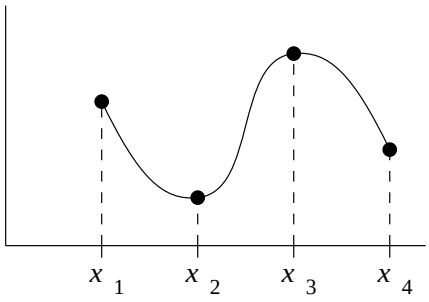
$$y = 2\sin t \qquad\qquad y = 2\frac{2t}{1+t^2}$$

Desabantailak:

- x balioa jakinda, kurbako (x, y) bikotea ezagutzea zaila izan daiteke. Ardatz bati dagozkion puntuen koordenatuak lortzea oso zaila da.

4.3 Interpolazioa, egokitzea eta itxuraren kontrola

Kurba bat definitzerakoan, kurbaren ekuazioak ez dira beti ezagunak izango, eta puntu batzuen koordenatuetan oinarrituz kalkulatu beharko ditugu ekuazioak. Kurbaren ekuazioa kalkulatzeko erabiltzen ditugun puntuei, kontrol-puntu deitzen zaie. Kurbaren ekuazioak lortzeko hiru metodo azalduko ditugu, bakoitzak bere abantaila eta desabantailak dituelarik: interpolazioa, egokitzea eta itxuraren kontrola.



Irudia 4.1: Lau puntuz osatutako euskarri baten interpolazioa.

4.3.1 Interpolazioa

Metodo horren bidez kontrol-puntuetatik pasako den kurbaren ekuazioak lortuko ditugu, 4.1 irudian ikus daitekeen modura.

Kontsidera dezagun interpolazioaren euskarri izango den OX ardatzeko $n+1$ puntuz osatutako multzoa kontsidera dezagun:

$$x_0, \ldots, x_n \ / \ x_i < x_{i+1}$$

x_i eta x_{i+1} balioen arteko distantzia edozein izan daiteke, nahiz eta normalean distantzia hori uniformea izan. x_i bakoitzari dagokion y_i balioa kalkulatuko duen funtzioa lortu nahi dugu. Horretarako, $n+1$ dimentsioko funtzioen bektore-espazioaren oinarri bat erabiliko dugu. Beraz, espazioko edozein funtzio gure oinarriko funtzioen konbinazio lineal modura idaztea posible izango da.

$$B = \{g_0(x), g_1(x), \ldots, g_n(x)\}$$

$$f(x) = a_0 g_0(x) + a_1 g_1(x) + \ldots + a_n g_n(x)$$

Horrela, espazioko funtzio guztiak definitzeko gai izango gara, eta gure helburua kontrol-puntuek definitzen duten funtzioa kalkulatzea izango da. Ezezagun bakarrak funtzioaren $a_0, \ldots, a_n$ koefizienteak dira. Kalkulatuko dugun $f(x)$ funtzioak, kontrol-puntuetatik pasatzen den kurba izan dadin, ondoko baldintzak bete beharko ditu:

$$f(x_i) = y_i$$

$n+1$ ezezagun eta $n+1$ ekuazio izango dituen ekuazio-sistema lortuko dugu.

$$G\,a = y$$

$$\begin{pmatrix} g_0(x_0) & g_1(x_0) & \dots & g_n(x_0) \\ g_0(x_1) & g_1(x_1) & \dots & g_n(x_1) \\ \vdots & \vdots & & \vdots \\ g_0(x_n) & g_1(x_n) & \dots & g_n(x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Ekuazio-sistemak soluzio bakarra izan dezan, matrizearen determinanteak zeroren desberdina izan beharko luke. Eta oinarri egoki bat erabiliz, matrizearen determinantea beti (euskarria edozein dela ere) zeroren desberdina izatea lortuko bagenu, oinarriak Haar-en baldintza beteko luke. Adibidez, hona hemen Haar-en baldintza betetzen duten zenbait oinarri:

$$1, x, x^2, \dots, x^n$$

$$1, \sin x, \cos x, \sin 2x, \cos 2x, \dots, \sin nx, \cos nx$$

$$1, e^x, e^{2x}, \dots, e^{nx}$$

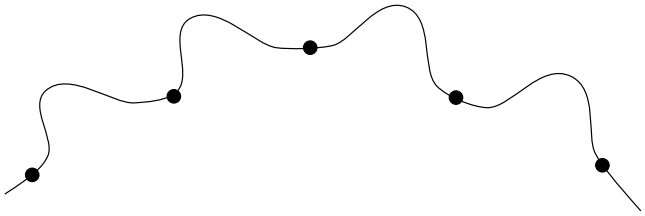
Ekuazio-sistema horren zenbait ezaugarri kontuan edukitzekoak dira:

- Euskarria da $\det(G) \neq 0$ baldintzan eragina duen bakarra,
- OX ardatzeko puntuak mantenduz, horiei dagozkien altuerak aldatzen baditugu, G ez da aldatzen: $a = G^{-1}y$. Ondorioz, G behin kalkulatzea nahikoa izango litzateke. Beraz, $y = (y_0, \dots, y_n)$ berri bati dagokion $a = (a_0, \dots, a_n)$ berria, ekuazio-sistema askatu gabe kalkula daiteke.

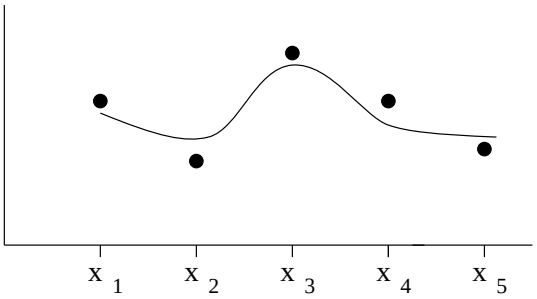
4.3.2 Interpolazioaren arazoak

Interpolazioa erabiltzeko orduan, sor daitezkeen zenbait arazo kontuan edukitzea komeni da:

- Erabiliko den funtzioen bektore-espazioa erabaki, eta bertako oinarria aukeratu behar da. Aukera horrek bere ondorio eta arazoak sor ditzake; adibidez, $\{1, x, x^2, \dots, x^n\}$ oinarria erabiltzeak gehiegizko konputazio-kostua dakar.



Irudia 4.2: Interpolazioan ematen den Runge deituriko efektua.



Irudia 4.3: Lau puntuko euskarri bati egokitze-metodoa erabiliz dagokion kurba.

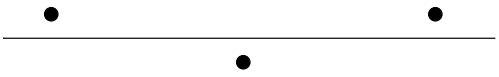
- Interpolazioaren bidez lortutako kurbak, 4.2 irudiko kurbaren motakoak izan daitezke, oszilakorrak.
- Runge efektua: interpolazioa ez da beti konbergentea izango, hau da, nahiz eta kontrol-puntu gehiago erabili, kalkulatu nahi dugun kurbaren hurbilpen okerragoa lor dezakegu kasu batzuetan.

Azken bi arazo horiengatik, komenigarriagoa da egokitzearen metodoa erabiltzea interpolazioarena baino.

4.3.3 Egokitzea

Metodo horren bidez lortutako funtzioak ez dira kontrol-puntuetatik pasatzen, 4.3 irudiko kasuan adibidez. Kurbaren funtzioa kalkulatzerakoan, hurbiltasun irizpide bat minimo egiten duen funtzioa aukeratzen da; hau da, kontrol-puntuetara ondoen egokitzen den funtzioa da komeni zaiguna.

Hurbiltasun-irizpide desberdinak erabil daitezke:



Irudia 4.4: 1. mailako polinomio bat erabiliz, ezinezkoa da, lerrokaturik ez dauden hiru kontrol-puntutatik pasatzen den kurba definitzea.

a) norma:

$$\sum |y_i - f(x_i)|$$

b) norma euklidearra:

$$\Big(\sum (y_i - f(x_i))^2 \Big)^{\frac{1}{2}}$$

c) norma uniformea:

$$max(y_i - f(x_i))$$

Irizpide sinpleena norma euklidearrarena da, minimo karratuen egokitzea.

$n-1$ baino maila handiagoko polinomioen espazioa erabiltzen badugu, kontrol-puntuak interpolatuko dituen funtzioa izango dugu (norma euklidearra 0 izango da kontrol-puntu guztietan) eta efektu oszilakorrak agertuko dira.

Bestalde, $n-1$ baino maila txikiagoko polinomioen espazio bat erabiltzea erabakitzen badugu, oro har, interpolazio-arazoak ez du soluziorik izango, ezinezkoa izango delako kontrol-puntuetatik igarotzen den eta $n-1$ baino maila txikiagoa izango duen funtzio bat aurkitzea. Oraingoan, hurbiltasun--irizpidea minimo egingo duen funtzioa aurkitu beharko dugu.

Adibidez, hiru kontrol-puntu ($n = 2$) emanik, $n-1$ mailako polinomioen espaziora mugatzen bagara, hiru puntuetatik igaroko den funtziorik ez dugu aurkituko, hiru puntuak lerrokaturik ez badaude behintzat, 4.4 irudian argi ikusten den bezala.

$$n - 1 = 1 \Rightarrow y = mx + b \qquad \text{(zuzenaren ekuazioa)}$$

$m+1$ dimentsioko espazio bat eta $g_0(x)$, ..., $g_m(x)$ oinarri bat aukeratzean, $g(x)$ funtzio bat ondoko moduan adieraziko genuke:

$$g(x) = a_0 g_0(x) + \ldots + a_m g_m(x)$$

Eta norma euklidearra erabiliz:

$$F^2(x) = \sum_{i=0}^{n} \Big(\; y_i - g(x_i) \; \Big)^2 = \sum_{i=0}^{n} \Big(\; y_i - \textstyle\sum_{j=0}^{m} a_j g_j(x_i) \; \Big)^2$$

Aurreko espresioa minimo egiten duen $g(x)$ funtzioa lortzea interesatzen zaigu eta aurreko espresioa minimo egiten badu, bere deribatua 0 izango da.

$g(x) = \sum_{j=0}^{m} a_j g_j(x)$ delakoa, a_0,...,a_n aldagaien funtzioa kontsidera daiteke eta $g(x)$ funtzioaren minimoa, bere deribatu partzialak zerorekin berdinduz kalkula daiteke:

$$\frac{\partial F(a_0, \ldots, a_m)}{\partial a_j} = 0$$

$$2 \textstyle\sum_{i=0}^{n} \Big(\; y_i - \sum_{k=0}^{m} a_k g_k(x_i) \; \Big) (-g_j(x_i)) = 0 \qquad j = 0, \ldots, m$$

$$-2 \textstyle\sum_{i=0}^{n} y_i g_j(x_i) + 2 \sum_{i=0}^{n} \sum_{k=0}^{m} a_k g_k(x_i) g_j(x_i) = 0 \qquad j = 0, \ldots, m$$

$$\textstyle\sum_{i=0}^{n} \sum_{k=0}^{m} a_k g_k(x_i) g_j(x_i) = \sum_{i=0}^{n} y_i g_j(x_i) \qquad j = 0, \ldots, m$$

$$\textstyle\sum_{k=0}^{m} \sum_{i=0}^{n} a_k g_k(x_i) g_j(x_i) = \sum_{i=0}^{n} y_i g_j(x_i) \qquad j = 0, \ldots, m$$

$$\textstyle\sum_{k=0}^{m} a_k \sum_{i=0}^{n} g_k(x_i) g_j(x_i) = \sum_{i=0}^{n} y_i g_j(x_i) \qquad j = 0, \ldots, m$$

$j = 0, \ldots, m$ kasuetan:

$$\begin{array}{ll} j = 0 & \textstyle\sum_{k=0}^{m} a_k \sum_{i=0}^{n} g_k(x_i) g_0(x_i) = \sum_{i=0}^{n} y_i g_0(x_i) \\ j = 1 & \textstyle\sum_{k=0}^{m} a_k \sum_{i=0}^{n} g_k(x_i) g_1(x_i) = \sum_{i=0}^{n} y_i g_1(x_i) \\ \vdots & \vdots \\ j = m & \textstyle\sum_{k=0}^{m} a_k \sum_{i=0}^{n} g_k(x_i) g_m(x_i) = \sum_{i=0}^{n} y_i g_m(x_i) \end{array}$$

Bestalde,

$$g_{kj} = \sum_{i=0}^{n} g_k(x_i) g_j(x_i)$$

$$b_j = \sum_{i=0}^{n} y_i g_j(x_i)$$

izanik, ondoko ekuazio-sistema izango dugu:

$$\begin{pmatrix} g_{00} & g_{01} & \dots & g_{0m} \\ g_{10} & g_{11} & \dots & g_{1m} \\ \vdots & \vdots & & \vdots \\ g_{m0} & g_{m1} & \dots & g_{mm} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{pmatrix}$$

g_{ij}=g_{ji} berdintza mantentzen denez, matrizea simetrikoa da eta, ondorioz, soluzio bat du gutxienez.

$1, x, \dots, x^m$ oinarria eta gehienez m ordenako polinomioen kasuan, sistema ondokoa izango litzateke:

$$\begin{pmatrix} n+1 & \sum_{i=0}^{n} x_i & \dots & \sum_{i=0}^{n} x_i^m \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \dots & \sum_{i=0}^{n} x_i^m \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^{n} x_i^m & \sum_{i=0}^{n} x_i^{m+1} & \dots & \sum_{i=0}^{n} x_i^{2m} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} y_i x_i \\ \vdots \\ \sum_{i=0}^{n} y_i x_m \end{pmatrix}$$

4.3.4 Itxuraren kontrola

Kasu honetan, kontrol-puntuak ez dira funtzioa interpolatzeko edo gerturatzeko erabiltzen, lortuko den kurbaren itxuraren erreferentzia modura baizik.

Dimentsio finituko funtzioen bektore-espazio bat eta oinarri zehatz bat aukeratzen dira. Aukeratutako oinarriak eragin handia du kurba definitzen duen funtzioa balioztatzean, kontrol-parametroak oinarriko funtzioen koefiziente modura jartzen baitira eta azken finean oinarriko funtzioak balioztatu behar baitira.

$$g(x) = a_0 g_0(x) + a_1 g_1(x) + \ldots + a_n g_n(x)$$

$a_0, a_1, \dots, a_n$ kontrol-parametroak dira.

a_i kontrol-parametro edo koefiziente bat aldatzean $g_i(x)$ funtzio osoa aldatzen ari gara.

$$g(x) = a_0 g_0(x) + \ldots + a_n g_n(x)$$
$$\overline{g}(x) = \overline{a_0} g_0(x) + \ldots + a_n g_n(x)$$

Eta bi funtzioen arteko diferentzia hauxe izango da:

$$g(x) - \overline{g}(x) = (a_0 - \overline{a_0})g_0(x)$$

$g_0(x)$ funtzioak kurbaren eremu osoan balio oso handiak itzultzen baditu, bi funtzioen arteko diferentzia handia izango da. Baina $g_0(x)$ funtzioak balio txikiak itzultzen baditu eta kurbaren eremu txiki batean bakarrik eragina badu, $g(x)$ eta $\overline{g}(x)$ oso antzekoak izango dira. Bi funtzioen arteko diferentzia bakarra $g_0(x)$ funtzioaren definizio-eremuan emango da.

Laburbilduz, egokiena balio txikiak itzultzen dituzten eta eremu mugatu batean definituta dauden funtzioak erabiltzea da. Ikusiko dugunez, ez da erraza propietate horiek dituzten funtzio polinomikoak lortzea, eta horregatik, splineak erabiliko ditugu.

4.4 Polinomioen oinarriak

- **berrekizun-oinarria:** $1, (x-a), (x-a)^2, \ldots, (x-a)^n$ oinarri hori erabiliz, itxuraren kontrola ezinezkoa da; interpolazioaren ebazpenerako, $\frac{2n^3}{3}$ eragiketa egin behar dira; eta egokitzearen kasuko minimo karratuen kalkulurako bakarrik da erabilgarria.

- **Lagrange-ren oinarria:** OX ardatzeko puntuen funtzio modura adieraz daitekeen oinarria da:

$$L_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

$$P(x) = \sum_{i=0}^{n} a_i L_i(x)$$

Interpolazioa egin ahal izateko, $L_i(x_k) = 0$ denez, $x = x_k$ denean ondokoa suposatu beharko dugu:

$$P(x_k) = a_k$$

Interpolazioan soluzioa berehalakoa da, oinarria kalkulatzea da zaila dena, baina euskarria erregularra balitz, asko sinplifikatuko litzateke. Oinarri-mota horrek ez du egokitze edo itxuraren kontrolerako balio.

- **Newton-en oinarria:** Newton-en oinarria erabiliz, interpolazioaren a_i koefizienteak kalkulatzeko, metodo errepikari batez balia gaitezke:

$$N_i = (x - x_0)(x - x_1)\dots(x - x_i)$$

$$N_0 = 1$$

eta

$$y(x_i) = y_i$$

$$y(x_i, x_j) = \frac{y(x_j) - y(x_i)}{x_j - x_i}$$

$$y(x_i, x_j, x_k) = \frac{y(x_j, x_k) - y(x_i, x_j)}{x_k - x_i}$$

$$a_i = y(x_0, x_1, \dots, x_i)$$

izanik, puntuak interpolatzeko:

$$P(x) = \sum_{i=0}^{n} a_i N_i(x)$$

- **Bernstein-en oinarria:**

$$B_k(x) = \begin{pmatrix} n \\ k \end{pmatrix} x^k (1 - x)^{n-k}$$

$(1 - x)$ kendurarengatik [0,1] tartean definituta dago. Eta

$$P(x) = \sum_{i=0}^{n} a_i B_i(x)$$

(x_i, a_i) puntuek kontrol-poligono bat osatzen dute, ondoko baldintzak betetzen dituena:

1. $P(0) = a_0$ eta $P(1) = a_n$

2. $P(x)$ polinomioaren 0 eta 1 puntuetako ukitzaileak, poligonoaren lehenengo eta azkeneko aldeei esker kalkula daitezke.
3. $P(x)$ polinomioaren puntuak, poligonoaren inguratzaile konexuaren barnean daude.
4. $P(x)$ funtzioa kontrol-poligonoak baino gehiagotan mozten duen zuzenik ez dago.

Oinarri-mota hori itxuraren kontrolerako egokiagoa dela esan daiteke.

Poligonoen azterketaren laburpena:

- Polinomioak ez dira interpolaziorako egokiak: kurba oszilakorrak sortzen dituzte eta gehiegizko konputazio-kostua eragiten dute, kontrol-puntuen kopuruak polinomioaren maila zehazten baitu.
- Itxuraren kontrolaren kasuan, maila-arazoak eman arren, Bernstein-en oinarria erabil daiteke.

4.5 Bezier kurbak

Bezier kurbak ondoko polinomioaren bitartez adierazten dira:

$$P(x) = a_0(1-x)^3 + a_1 3x(1-x)^2 + a_2 3x^2(1-x) + a_3 x^3$$

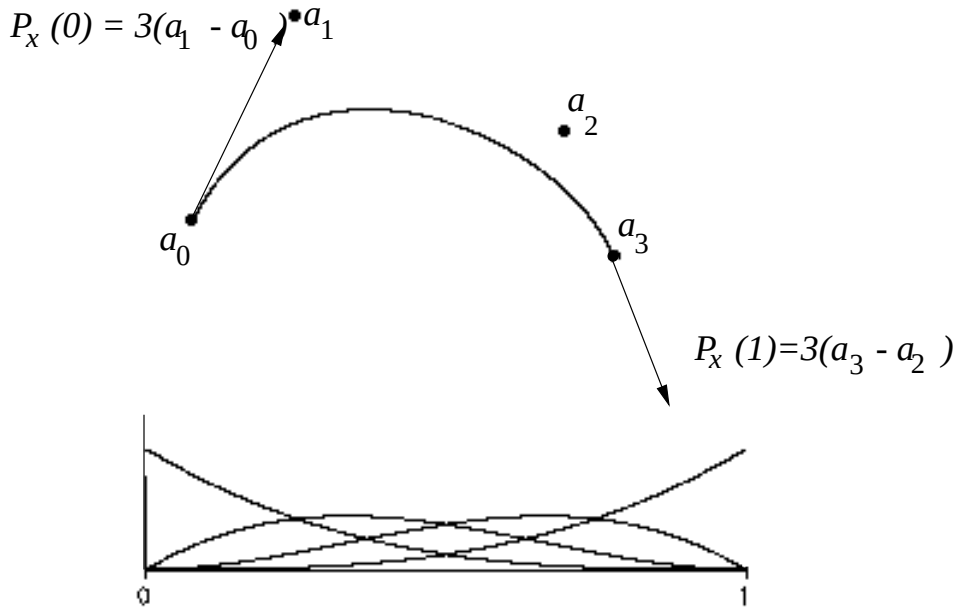
a_0, a_1, a_2 eta a_3 kontrol-puntuak eta $(1-x)^3, 3x(1-x)^2, 3x^2(1-x), x^3$ Bernstein-en oinarriaren funtzioak dira. Kurbaren itxura kontrol-puntuek definitzen dute. $P(x)$ polinomio kubikoa x aldagaiarekiko deribatuz:

$$P_x(0) = 3(a_1 - a_0)$$

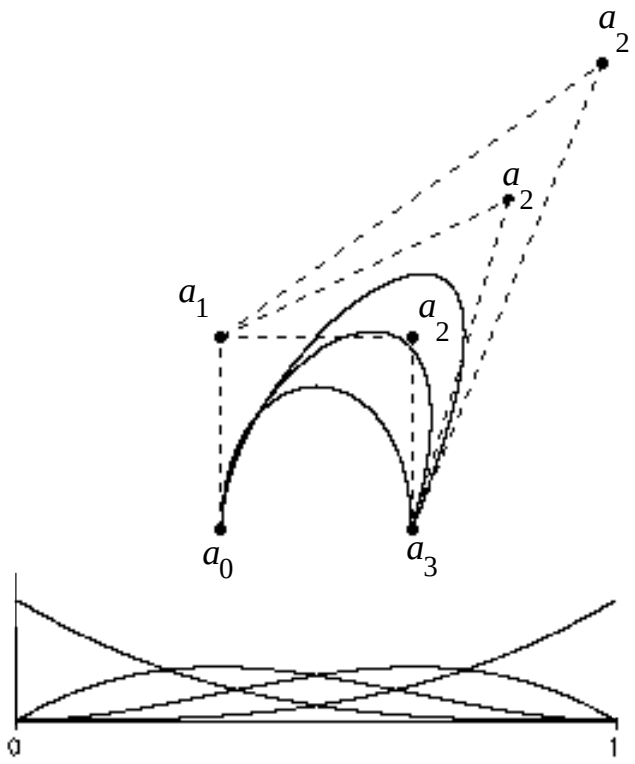
$$P_x(1) = 3(a_2 - a_3)$$

Horiek, kurbak mutur-puntuetan dituen ukitzaileen norabideak dira, eta 4.5 irudian ikus daitekeenez a_1 eta a_2 puntuek bektore ukitzailearen definizioan parte hartzen dute.

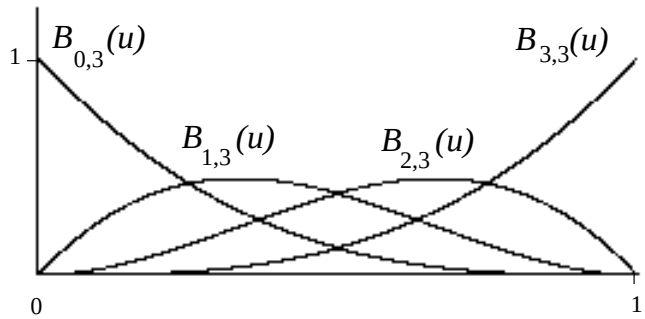
Beraz, lau kontrol-puntuak aldatuz, kurbaren itxura eta kokapena aldatzeko gai izango gara. 4.6 irudian kontrol-puntu baten aldaketak kurbaren gain duen eragina ikus daiteke. Bezier kurbekin lan egitean, erdiko bi kontrol-puntuen kokapen-aldaketak kurba erakartzen dutela konturatuko



Irudia 4.5: Bezier kurbako kontrol-puntuen eta bektore ukitzaileen arteko erlazioa.



Irudia 4.6: Kontrol-puntu baten kokapen-aldaketak kurbaren gain duen eragina.



Irudia 4.7: Bezier-en oinarria osatzen duten funtzioen grafikoak.

gara. Erdiko kontrol-puntuetako edozein mugituz, kurba herrestan daramagula dirudi.

Azter dezagun kontrol-puntuak oinarriko funtzioetan eta lortuko dugun azken kurban duten eragina. Azaldu dugunez, kurbaren polinomioa

$$P(x) = \sum_{i=0}^{n} a_i B_{i,3}(x)$$

da, non batuketako batugai bakoitza kontrol-puntu eta oinarriko funtzio baten arteko biderkadura den. Oinarria osatzen duten funtzio kubikoen grafikoak 4.7 irudian ageri dira. Kontrol-puntu baten balioak, dagokion oinarriko funtzioak kurban izango duen garrantzia adieraziko du. Adibidez, a_0-ren balioa zenbat eta handiagoa izan, orduan eta eragin handiagoa izango du $B_{0,3}$ funtzioak lortuko dugun kurban. a_0-ren eragin handiena $x = 0$ izatean ematen da, beste hiru funtzioek 0 balioa itzultzen dutelako. x a_1 kontrol-punturantz mugitzen goazen heinean, kurbaren itxuran B_0 eta B_1 polinomioek eragin handiagoa dute, nahiz eta B_2 eta B_3 polinomioek ere eragina izan. $x = 1/3$ izatean, a_1 kontrol-puntuak izango du eragin handiena, eta $x = 2/3$ izatean, berriz, a_2 puntuak.

Edozein kontrol-puntu mugitzeak, kurba osoan izango du eragina. Hori Bezier kurben desabantaila da, kontrola globala baita. Polinomio kubikoetarako erabiliko dugun oinarria, ondokoa da:

$$\begin{aligned} B_{0,3}(x) &= (1-x)^3 \\ B_{1,3}(x) &= 3x(1-x)^2 \\ B_{2,3}(x) &= 3x^2(1-x) \\ B_{3,3}(x) &= x^3 \end{aligned}$$

Oinarri hori, n. mailako Bernstein-en oinarria erabiliz lortuko genukeen polinomioaren kasu partikular bat da:

$$B(x) = \sum_{i=0}^{n} a_i B_{i,n}(x),$$

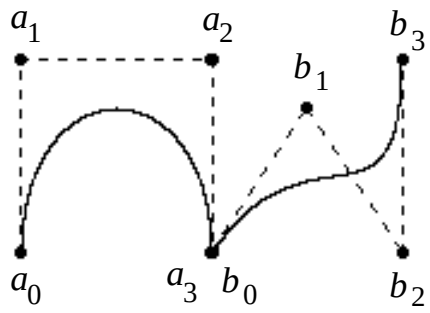
non $B_{i,n}(x) = \begin{pmatrix} n \\ i \end{pmatrix} x^i (1-x)^{n-i}$ den, $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{(i!(n-i)!)}$ delakoa koefiziente binomikoa izanik. Horren ondorioz, n kontrol-puntu izanik $n+1$ mailako Bezier kurba erabiltzeko aukera izango genuke. Baina $n > 3$ izatean x^n kalkulatzeak konputazio-kostu handia dakar. Horregatik, kontrol-puntu asko izatean, B-splineak erabili ohi dira.

Edozein kurba-motaren gain aldaketa afin bat (biraketa, leku-aldaketa, neurri-aldaketa ...) eragiteko, kurbak kontrol-puntuen konbinazio linealen bidez definiturik daudenez, nahikoa da aldaketa kontrol-puntuen gain eragitea. Beraz, aldaketa kontrol-puntuen gain eragiten dugu, eta kontrol-puntu berrien bidez kurba aldatua kalkulatu ahal izango dugu. Perspektibaren aldaketa ez da afina, eta, horren ondorioz, ezin ditugu kontrol-puntuak irudi-espazioan kokatu, gero bertan kurba aldatua kalkulatzeko.

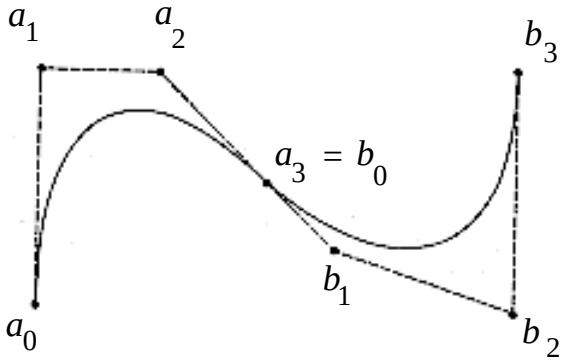
4.5.1 Bezier kurbak lotzen

Kurba konplexuagoak marrazteko, lau kontrol-punturen bidez definituriko Bezier kurbak lotu daitezke. Bezier kurbak lotuz lortutako kurbak zatikatuak dirudite, hots, ez dute jarraitasun itxurarik azaltzen. Kurba konplexuak marrazteko beste metodo bat polinomioaren maila handiagotzea izango litzateke, metodo horrek dakartzan arazoak (konputazio-kostua eta desabantaila matematikoak) pairatuz. Errazagoa da kurba osoa azpikurba kubikoetan zatitzea. Ondoren, horiek lotu egin behar dira. Lotura-puntuetan G^0 edo G^1 jarraitasun-baldintzak bete daitezela erabaki dezakegu. G^0 bete dadin, lehenengo azpikurbako azken kontrol-puntuak bigarren azpikurbako lehenengo kontrol-puntua izan behar du; horrela, azpikurba bat bukatzen den toki berean hasiko da bestea. Kokapen-jarraitasuna eta ukitzaileen jarraitasunaren adibideak 4.8 eta 4.9 irudietan ikus daitezke.

Bestalde, azpikurba baten bukaerako bektore ukitzailea hurrengo kurbaren hasierako bektore ukitzailearen proportzionala bada, G^1 beteko da eta ukitzaileen jarraitasuna ematen dela esan ahal izango dugu. Bi kurben kontrol-puntuak a_i eta b_i badira, G^1 jarraitasun-baldintza bete dadin lehenen-



Irudia 4.8: Bi Bezier kurben arteko kokapen-jarraitasuna.



Irudia 4.9: Bi Bezier kurben arteko ukitzaileen jarraitasuna.

goaren azken kontrol-puntuak eta bigarrenaren lehenengoak berdinak izan behar dute, eta gainera:

$$(a_3 - a_2) = k(b_1 - b_0)$$

Baldintza hori betearaziz n kurba lotu ditzakegu G^1 jarraitasun-baldintza mantenduz. Hala ere, baldintza hori betearaztean eta kontrol-puntuen koordenatuak aldatzean, hasierako kurben itxura gehiegi alda dezakegu. Arazo hori ekiditeko soluzioa, polinomioen maila handiagotzea edo kurba bakoitza azpikurba txikiagotan zatitzea da.

4.5.2 Bezier kurben propietateen laburpena

- n kontrol-puntu baditugu, polinomioaren maila $n-1$ izango da. Ordenadore bidezko irudigintzan 3. mailako polinomioak erabiltzen dira.
- Kurbak, kontrol-puntuek adierazitako poligonoaren itxura hartzen du, eta horiek osatzen duten poligono inguratzailearen barnean kokatzen da.
- Edozein kontrol-punturen kokapen-aldaketak kurba osoan du eragina. Kontrola globala da.
- Kurba bat definitzen duten lehenengo eta azken kontrol-puntuak, beti interpolatzen dira.
- Kurbaren muturretako kontrol-puntuen (a_0 eta a_3, $n = 4$ bada) bektore ukitzaileek, kontrol-puntuek adierazten duten poligonoaren lehenengo eta azken ertzen norabide berdina dute.
- Kurbaren gain edozein aldaketa afin (hau da, aldaketa linealen edozein konbinazio) egiteko, kontrol-puntuei aldaketa eragin eta horrela lortutako puntu berriek definitzen duten kurba kalkulatzea nahikoa da.

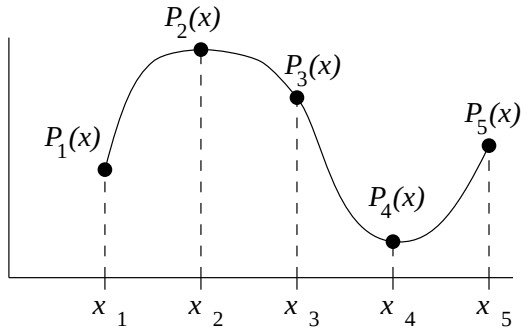
4.6 Spline funtzioak

Polinomio desberdinak elkartzean datza, horien lotura-puntuetan ahalik eta ordena handieneko jarraitasuna lortu beharko dugularik, 4.10 irudian dakusagun bezala.

Suposa dezagun OX ardatzeko n puntu ditugula, $x_1 < x_2 < \ldots < x_n$ izanik eta $[x_i, x_{i+1}]$ tarte bakoitzean polinomio bat kalkulatu nahi dugula. Polinomioek tarte bakoitzeko lotura-puntuetan ahalik eta ordena altueneko deribatu jarraia izan beharko dute (gure kasuan gutxienez 2. ordenakoa).

Kurba osatzen duten polinomioen maila k edo txikiagoa izatean, jarraitasuna gehienez $(k-1)$. ordenakoa izango da, eta kurba k. mailako spline bat dela esango dugu.

Spline kubikoak (3. mailakoak), oso erabilgarriak dira eta hagaxka malgu batek puntu batzuetatik pasatzean deskribatzen duen kurba simulatzen dutela esan daiteke. Hagaxkak, kurbaren bi kontrol-puntuk mugatutako tartean, 3. mailako polinomio baten itxura hartzen du. Spline kubikoa kurba



Irudia 4.10: x_1,...,x_5 euskarriak definitutako splinea.

osoan zehar batezbesteko kurbadura eta oszilazioa minimo egiten duen funtzio interpolatzailea da.

Spline bat eraikitzeko, har dezagun $x_1, x_2, \ldots, x_n$ euskarri erregularra ($x_{i+1} - x_i = h$) eta izan bitez $y_1, y_2, \ldots, y_n$ euskarriko puntu bakoitzari dagozkion altuera. Bilatzen ari garen spline kubikoa, tarte bakoitzeko (x_i, y_i) eta (x_{i+1}, y_{i+1}) puntuak interpolatzen dituen 3. mailako (edo gutxiagoko) polinomioez osatua egongo da. $P_i(x)[x_i, x_{i+1}]$ tarteko polinomioa bada:

$$P_i(x) = c_{0i} + c_{1i}\left(\frac{x-x_i}{h}\right) + c_{2i}\left(\frac{x-x_i}{h}\right)^2 + c_{3i}\left(\frac{x-x_i}{h}\right)^3$$

Eta splinea jarraia izan dadin, muturretako bi puntuak interpolatu beharko ditu:

$$P_i(x_i) = y_i = c_{0i}$$

$$P_i(x_{i+1}) = y_{i+1} = c_{0i} + c_{1i} + c_{2i} + c_{3i}$$

Bi tarte auzokideri dagozkien polinomioek puntu komuna edukitzeaz gain, puntu horretan lotura leuna izan dadin, bi polinomioek deribatu berdina eduki behar dute:

$$P_i'(x_i) = P_{i-1}'(x_i)$$

$$P_i'(x_{i+1}) = P_{i+1}'(x_{i+1})$$

Hau da, x_i puntuko deribatuek [x_i,x_{i+1}] eta [x_{i-1},x_i] tarteetako polinomioetan berdinak izan beharko dute.

$$P_i^{'}(x) = \frac{c_{1i}}{h} + 2\frac{c_{2i}}{h}\left(\frac{x-x_i}{h} \right) + 3\frac{c_{3i}}{h}\left(\frac{x-x_i}{h} \right)^2$$

$$P_i^{'}(x_i) = \frac{c_{1i}}{h} = \sigma_i$$

$$P_i^{'}(x_{i+1}) = \frac{c_{1i}}{h} + 2\frac{c_{2i}}{h} + 3\frac{c_{3i}}{h} = \sigma_{i+1}$$

Ekuazio horiek erabiliz, koefizienteak kalkula daitezke:

$$c_{0i} = y_i$$

$$c_{1i} = h\sigma_i$$

$$c_{2i} = 3(y_{i+1} - y_i) - h(2\sigma_i + \sigma_{i+1})$$

$$c_{3i} = h(\sigma_i + \sigma_{i+1}) - 2(y_{i+1} - y_i)$$

Koefiziente horiei esleituko zaien balioa splinearen euskarriko puntuetako deribatuaren araberakoa izango da, eta deribatu horren balioa ezezaguna da; baina oraindik beste baldintza bat betetzea falta da. Bigarren deribatuaren jarraitasunaren baldintza:

$$P_{i-1}^{''}(x_i) = P_i^{''}(x_i)$$

$$P_i^{''}(x) = \frac{2c_{2i}}{h^2} + \frac{6c_{3i}}{h^2}\left(\frac{x-x_i}{h} \right)$$

Ondorioz:

$$\frac{2c_{2i-1}}{h^2} + \frac{6c_{3i-1}}{h^2}\left(\frac{x_i-x_{i-1}}{h} \right) = \frac{2c_{2i}}{h^2} + \frac{6c_{3i}}{h^2}\left(\frac{x_i-x_i}{h} \right)$$

$x_i - x_i = 0$ eta $x_i - x_{i-1} = h$ direnez:

$$\frac{2c_{2i-1}}{h^2} + \frac{6c_{3i-1}}{h^2} = \frac{2c_{2i}}{h^2}$$

c_{2i-1}, c_{3i-1} eta c_{2i} aldagaiak ordezkatuz:

$$\sigma_{i-1} + 4\sigma_i + \sigma_{i+1} = \frac{3(y_{i+1} - y_{i-1})}{h} = b_i$$

n ezezagun eta $n-2$ ekuazio izango ditugu, baina σ_1 eta σ_n ezagunak suposatzen baditugu, $n-2$ ekuazio eta $n-2$ ezezaguneko ekuazio-sistema izango dugu:

$$\begin{pmatrix} 4 & 1 & 0 & \dots & & & 0 \\ 1 & 4 & 1 & \dots & & & 0 \\ 0 & 1 & 4 & \dots & & & 0 \\ \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & \dots & & 1 & 4 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \vdots \\ \sigma_{n-2} \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} b_2 - \sigma_1 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-2} \\ b_{n-1} - \sigma_n \end{pmatrix}$$

Hadaward-en matrize izenaz ezagutzen da hori, eta bere determinantea zeroren desberdina da. Beraz, soluzio bakarra du eta σ_i balioak lortzeko aukera emango digu. σ_i balioak ordezkatuz c_{ki} polinomioen koefizienteak kalkulatuko ditugu, puntuak interpolatzen dituen splinea lortuz.

Lortutako soluzioa ez dago euskarriaren menpe. OX ardatzeko n puntuei dagozkien Y koordenatuak, σ_1 eta σ_n balioak emanik spline bakarra kalkula daiteke. Spline kubikoen multzoak $n+2$ dimentsioko espazioa osatzen du.

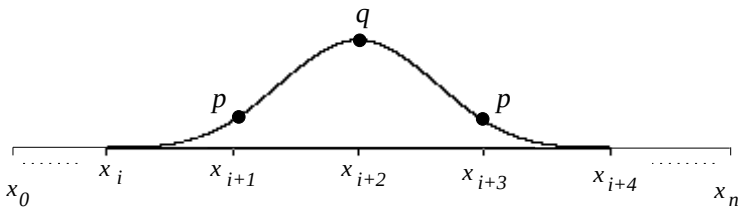
4.6.1 Oinarrizko splineak

Itxuraren kontrolaren atalean, kurbaren funtzioa definitzeko erabilitako oinarria oso garrantzitsua dela aipatu dugu. a_k koefiziente bat aldatzean, funtzioan gertatzen den aldaketa finkatu dugu:

$$g(x) - \overline{g}(x) = (a_0 - \overline{a_0})b_k(x)$$

Aldaketa hori globala izan beharrean, lokala izatea komeni da, funtzioaren aldaketa a_k puntuaren inguruan bakarrik eman dadin. Horretarako, b_k funtzioak a_k-ren inguruan balio txikiak (aldaketa handiak ez izateko) eta ez-nuluak izan beharko lituzke. $b_k(x)$-ren itxura 4.11 irudikoa izan beharko litzateke.

Horrelako funtzio bat, splineak erabiliz, erraz lor daiteke. Splineak bete beharreko baldintzak hauexek dira:



Irudia 4.11: Oinarrizko splinea.

$$
\begin{aligned}
&B(x_1) = B(x_2) = \ldots = B(x_{k-2}) = 0\\
&B(x_{k-1}) = p\\
&B(x_k) = q\\
&B(x_{k+1}) = p\\
&B(x_{k+2}) = B(x_{k+3}) = \ldots = B(x_n) = 0\\
&B^{'}(x_0) = 0 = \sigma_1\\
&B^{'}(x_n) = 0 = \sigma_n
\end{aligned}
$$

Balio horiekin eta $q = 4p$ hartuta, $\sigma_i (i = 2 \ldots n-1)$ kalkulatzeko ekuazio--sistematik ondoko emaitzak lortuko genituzke:

$$
\begin{aligned}
&\sigma_2 = \sigma_3 = \ldots = \sigma_{k-2} = 0\\
&\sigma_{k-1} = \tfrac{3q}{4h}\\
&\sigma_k = 0\\
&\sigma_{k+1} = \tfrac{-3q}{4h}\\
&\sigma_{k+2} = \sigma_{k+3} = \ldots = \sigma_{n-1} = 0
\end{aligned}
$$

Balio horiek polinomioen koefizienteak ematen dizkiguten ekuazioetan ordezkatuz:

$$
\begin{aligned}
&c_{0i} = y_i\\
&c_{1i} = h\sigma_i\\
&c_{2i} = 3(y_{i+1} - y_i) - h(2\sigma_i + \sigma_{i+1})\\
&c_{3i} = h(\sigma_i + \sigma_{i+1}) - 2(y_{i+1} - y_i)
\end{aligned}
$$

Horrela, splinea osatzen duten polinomioen ekuazioak q-ren arabera lor ditzakegu, tarte bakoitzeko polinomio bat. Spline bakoitza nolakoa den badakigu, baina $n-4$ spline besterik ez dauzkagu: $B_3, B_4, \ldots, B_{n-2}$. Spline

kubikoen oinarri bat osatzeko falta diren 6 funtzioak lortzeko (espazioa $n+2$ dimentsiokoa da), 6 puntu txertatuko ditugu, 3 puntu eskuinean eta beste hiru ezkerrean.

Oinarrizko spline bakoitzaren $P_i(x)$ funtzioak:

$$
\begin{array}{lrl}
i = 1 \ldots k-3: & P_i(x) = & 0 \\
\\
i = k-2: & P_i(x) = & 0 + h0 \left(\frac{x-x_i}{h} \right) + \\
 & & + \left(3(p-0) - h(0+\frac{3q}{4h}) \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(h(0+\frac{3q}{4h}) - 2(p-0) \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & 0 + 0 + \left(3p - \frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(\frac{3q}{4} - \frac{2q}{4} \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & \left(\frac{3q}{4} - \frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(\frac{3q}{4} - \frac{2q}{4} \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & \frac{q}{4} \left(\frac{x-x_i}{h} \right)^3 \\
\\
i = k-1: & P_i(x) = & p + h\frac{3q}{4h} \left(\frac{x-x_i}{h} \right) + \\
 & & + \left(3(q-p) - h(2\frac{3q}{4h}+0) \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(h(\frac{3q}{4h}+0) - 2(q-p) \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & \frac{q}{4} + \frac{3q}{4} \left(\frac{x-x_i}{h} \right) + \left(3\frac{3q}{4} - 2\frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(\frac{3q}{4} - 2\frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^3 \\
 & = & \frac{q}{4} + \frac{3q}{4} \left(\frac{x-x_i}{h} \right) + \frac{3q}{4} \left(\frac{x-x_i}{h} \right)^2 - \frac{3q}{4} \left(\frac{x-x_i}{h} \right)^3 \\
\\
i = k: & P_i(x) = & q + h0 \left(\frac{x-x_i}{h} \right) + \\
 & & + \left(3(p-q) - h(2\ \ 0 + \frac{-3q}{4h}) \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(h(0-\frac{3q}{4h}) - 2(p-q) \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & q + \left(\frac{-9q}{4} + \frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^2 + \\
 & & + \left(\frac{-3q}{4} + 2\frac{3q}{4} \right) \left(\frac{x-x_i}{h} \right)^3 = \\
 & = & q + \frac{-6q}{4} \left(\frac{x-x_i}{h} \right)^2 + \frac{3q}{4} \left(\frac{x-x_i}{h} \right)^3
\end{array}
$$

$$\begin{aligned} i=k+1: \quad P_i(x) = \; & p + h\tfrac{-3q}{4h}\left(\ \tfrac{x-x_i}{h}\ \right) + \\ & + \left(\ 3(0-p) - h(2\tfrac{-3q}{4h}+0)\ \right)\left(\ \tfrac{x-x_i}{h}\ \right)^2 + \\ & + \left(\ h(\tfrac{-3q}{4h}+0) - 2(0-p)\ \right)\left(\ \tfrac{x-x_i}{h}\ \right)^3 = \\ = \; & \tfrac{q}{4} - \tfrac{3q}{4}\left(\ \tfrac{x-x_i}{h}\ \right) + \left(\ \tfrac{-3q}{4} + 2\tfrac{3q}{4}\ \right)\left(\ \tfrac{x-x_i}{h}\ \right)^2 + \\ & + \left(\ \tfrac{-3q}{4} + 2\tfrac{q}{4}\ \right)\left(\ \tfrac{x-x_i}{h}\ \right)^3 = \\ = \; & \tfrac{q}{4} - \tfrac{3q}{4}\left(\ \tfrac{x-x_i}{h}\ \right) + \tfrac{3q}{4}\left(\ \tfrac{x-x_i}{h}\ \right)^2 - \tfrac{q}{4}\left(\ \tfrac{x-x_i}{h}\ \right)^3 \end{aligned}$$

Spline kubikoen oinarri bat osatzen duten oinarrizko $n+2$ splineak lortu ondoren, B-spline horien bidez, $S(x)$ edozein spline adieraz dezakegu:

$$S(x) = \sum_{k=0}^{n+1} a_k B_k(x)$$

$x \in [x_k, x_{k+1}]$ izanik, $S(x)$ kalkulatzeko nahikoa izango da 4 batuketa kalkulatzea, tarte horretan 4 B-splinek baitaukate zeroren ezberdina den balioren bat. $q = 2/3$ [1] balioa hartzen badugu:

$$B_{k-1}(x) = \tfrac{1}{6} - \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right) + \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right)^2 - \tfrac{1}{6}\left(\ \tfrac{x-x_k}{h}\ \right)^3$$

$$B_k(x) = \tfrac{2}{3} - \left(\ \tfrac{x-x_k}{h}\ \right)^2 + \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right)^3$$

$$B_{k+1}(x) = \tfrac{1}{6} + \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right) + \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right)^2 - \tfrac{1}{2}\left(\ \tfrac{x-x_k}{h}\ \right)^3$$

$$B_{k+2}(x) = \tfrac{1}{6}\left(\ \tfrac{x-x_k}{h}\ \right)^3$$

Beraz:

$$S(t) = a_{k-1}(\tfrac{1}{6} - \tfrac{1}{2}t + \tfrac{1}{2}t^2 - \tfrac{1}{6}t^3) + a_k(\tfrac{2}{3} - t^2 + \tfrac{t^3}{2}) +$$

$$+a_{k+1}(\tfrac{1}{6} + \tfrac{t}{2} + \tfrac{t^2}{2} - \tfrac{t^3}{2}) + a_{k+2}(\tfrac{t^3}{6}) \qquad \text{non} \qquad t = \tfrac{x-x_k}{h}$$

x puntu bati zein tarte dagokion jakin eta gero, t=$\frac{x-x_k}{h}$ kalkulatzen da splineak itzulitako balioa lortzeko. Aurreko espresioa sinplifikatuz:

[1] q=2/3 eta p=1/6 izatean, B-splineak zenbait ezaugarri berezi ditu; adibidez, bere azpiko azalera 1 da.

$$S(x) = \frac{a_{k-1}+4a_k+a_{k+1}}{6} + \frac{a_{k-1}+a_{k+1}}{2}t + \frac{a_{k-1}+2a_k+2a_{k+1}}{2}t^2 +$$

$$+\frac{a_{k-1}+3a_k+3a_{k+1}+a_{k+2}}{6}t^3 = s_0 + s_1t + s_2t^2 + s_3t^3$$

Kalkulu osoa, puntu bakoitzeko, 3 biderkaketa eta 3 batuketetara mugatzen da.

B-splineen **garrantzi handiko ezaugarriak**:

- Oinarrizko 4 B-splineek puntu batean duten balioen batura beti 1 da; ondorioz, 4 kontrol-puntu izanik, beti betetzen da honako hau:

$$S(x) = a_{k-1}B_{k-1}(x) + a_kB_k(x) + a_{k+1}B_{k+1}(x) + a_{k+2}B_{k+2}(x) \leq$$

$$\leq \max(a_{k-1}, a_k, a_{k+1}, a_{k+2})$$

  Horri esker, kurba, kurba bera definitzen duten kontrol-puntuek osatutako poligono inguratzailearen barnean egongo da.

- a_i kontrol-puntu bat aldatuz, kurba berriaren eta aurreko kurbaren arteko diferentzia

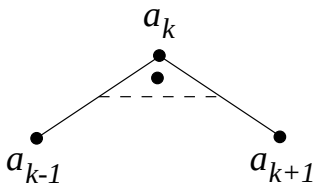
$$(a_i - \overline{a_i})B_i(x)$$

  izango da. Eta $B_i(x)$ polinomioak kurban duen eragina lau tartetara mugatzen denez, kurbaren aldaketa lau tarte horietan bakarrik emango da. Kontrola lokala izango da.

- B-splineetan, $x = x_i$ euskarriko puntu bati dagokion balioa

$$S(x_i) = (a_{k-1} + 4a_k + a_{k+1})/6$$

  da eta balio hori a_k puntutik oso gertu dago. B-splineak itzuliko digun balioa, a_i, $\frac{a_{i-1}+a_i}{2}$ eta $\frac{a_i+a_{i+1}}{2}$ puntuek osatzen duten triangeluaren barizentroa izango da, 4.12 irudian ikus daitekeen bezala. Ezaugarri horri esker, kontrol-puntuak osatzen duten poligonoaren itxura hartuko du kurbak.



Irudia 4.12: Hiru puntuk osatutako triangeluaren zentroidea.

Oinarrizko splineen bidezko kurben interpolazioa

Kontrol puntuetan splineak duen balioa ezaguna denez, hau da, $S(x_i)$ balioa a_{i-1}, a_i eta a_{i+1} kontrol puntuen menpe dagoenez, arazoa alderantziz plantea dezakegu. Hau da, kurbako puntuen balioak finka ditzakegu, ondoren a_i koefizienteak kalkulatzeko, n ekuazio eta $n+2$ ezezagun izanik:

$$a_{i-1} + 4a_i + a_{i+1} = 6y_i \qquad i = 1, 2, \ldots, n$$

$$\begin{pmatrix} 4 & 1 & 0 & \ldots & & & 0 \\ 1 & 4 & 1 & \ldots & & & 0 \\ 0 & 1 & 4 & \ldots & & & 0 \\ \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 4 & 1 \\ 0 & 0 & 0 & \ldots & & 1 & 4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} 6y_1 - a_0 \\ 6y_2 \\ 6y_3 \\ \vdots \\ 6y_{n-1} \\ 6y_n - a_{n+1} \end{pmatrix}$$

B-splineen bidezko kurben edo gainazalen interpolazioari, gutxienez bi erabilera ematen zaizkio ordenadore bidezko irudigintzan. Lehenengoa objektuen eraikuntzan eta bigarrena ordenadore bidezko animazioan. Objektu bat eraikitzerakoan, baliteke objektua osatzen duten puntuen lagin bat besterik ez izatea, laser aztertzaile baten bidez lortua adibidez. Kasu horretan, lortutako puntuak interpolatuko lituzkeen gainazala lortzea nahiko genuke, eta ondoren gainazalen egokitze-teknikak eragin beharko genituzke. Ordenadore bidezko animazioan, berriz, kurba bat eszenako objektu batek denboran zehar egingo duen bidea adierazteko erabil daiteke. Objektua zenbait puntutatik igarotzea ezinbestekoa da, eta puntu horien arteko bidea kurbak deskribatuko du. Helburu hori lortzeko, B-splineak erabili ohi dira, beren C^2 jarraitasuna edo bigarren deribatuaren jarraitasuna oso baliagarria delako. Animazioetan objektuen mugimendu jarraiak lortzea interesatzen zaigu, eta

B-spline kurben bidez posible izango da hori lortzea. B-spline kurbak objektuaren kokapena denboraren funtzioan itzuliko digu. Horrez gain, kameraren mugimendurako ere erabilgarriak dira. Jarraitasun-ezaugarriak direla eta, denboraren funtzioan mugimendu leunak lortzen ditugu kamerarentzat; horrela, irudi batetik besterako diferentziak denborarekiko jarraitasuna izango du.

Nahiz eta normalean kurben teoria gainazaletarako erabilgarria izan (gainazalak kurben kasu orokorra direlako), oraingoan ez da hori gertatzen. Gainazalen egokitzean arazo berri asko sortzen da; hori dela eta, kurben egokitzea eta gainazalena guztiz desberdintzen dira.

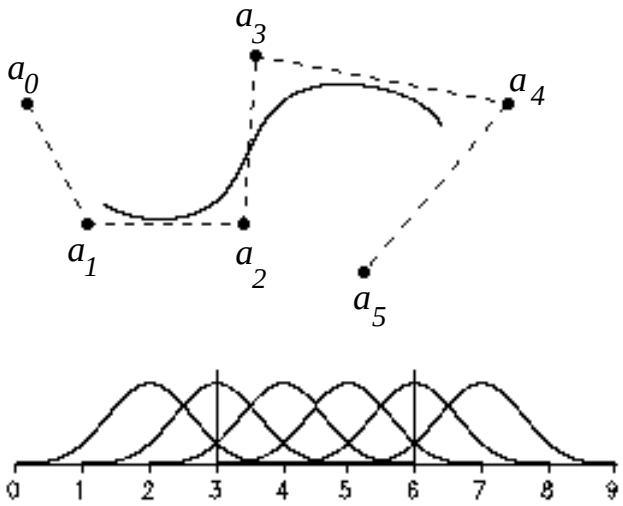
Gerta daiteke, puntu multzo bat adierazi ondoren puntu guztiak interpolatu nahi ez izatea, komenigarriagoa izan baitaiteke zenbait puntu interpolatuko duen eta besteetatik gertu pasako den kurba bat kalkulatzea. Bestetan, puntu guztiak interpolatzea nahi izango dugu.

B-spline uniformeak

B-spline kurba baten azpikurba bat definitzeko, oinarrizko lau funtzio eta lau kontrol-puntu behar dira. Beraz azpikurba-kopurua baino hiru kontrol-puntu gehiago eta oinarrizko hiru funtzio gehiago behar ditugu. Azpikurbak elkartzen dituzten puntuen x balioari, **korapilo-balio** deritzo. Eta B-spline bat uniformea dela esatea, korapilo-balioak elkarren artean distantzia berdinera daudela esatearen baliokidea da. 4.13 irudian sei kontrol-puntuz definitutako spline bat ikus daiteke. Gogoratu lau kontrol-puntu behar direla azpikurba bat definitzeko; gainera, azpikurba horren hasiera lehenengo hiru kontrol-puntuen barizentroan dagoela, eta bukaera, berriz, azkeneko hiru kontrol puntuen barizentroan. Kontrol-puntu bakoitzak oinarrizko Spline bat edo B-spline bat kontrolatzen du; beraz, sei kontrol puntuk sei B-spline behar dituzte. Lehenengo lau kontrol-puntuek azpikurba bat definitzen dute, hurrengo azpikurba definitzeko, lau horietako lehenengoa kendu eta bosgarren kontrol-puntua hartu behar dugu; eta hirugarren azpikurbarako, seigarrena sartu eta azken lau horien lehenengoa kendu beharko dugu.

A_3 azpikurba, $B_0 \dots B_3$ B-splineek eta $a_0 \dots a_3$ puntuek zehazten dute.
A_4 azpikurba, $B_0 \dots B_3$ B-splineek eta $a_1 \dots a_4$ puntuek zehazten dute.
A_5 azpikurba, $B_0 \dots B_3$ B-splineek eta $a_2 \dots a_5$ puntuek zehazten dute.

Azpikurbek kontrol-puntuak konpartitzen dituztenez, beren arteko C^2 jarraitasuna mantentzea lortzen da. Bestalde, 4.14 irudian B-splineak erabiliz



Irudia 4.13: Sei kontrol-puntuz eta hiru azpikurbaz definituriko B-spline kurba.

lortzen den kontrol lokala ikus daiteke.

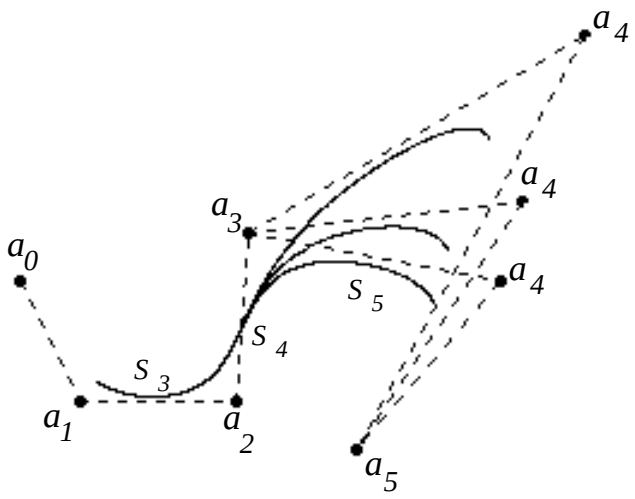
Ondoren, kurba definitzen duten eta oinarria osatzen duten funtzioak aztertuko ditugu. Oinarrizko funtzio bakoitza, jarraian dauden lau tartetan $(x_i, x_{i+1}, \ldots, x_{i+4})$ balio ez-nulua hartzen duten lau azpikurbaz osatutako funtzio kubikoa da; bere itxura 4.11 irudian ikus daiteke. x_{i+2} puntua, B-splinearen erdian kokatzen da beti. Oinarriko funtzioak bata bestearen kopiak dira eta korapilo-balioen arteko distantzien eraginez, kopia guztiak lekuz aldatuta aurkitzen dira. Azaldutako guztia 4.15 eta 4.16 irudiak begiratuz hobeto uler daiteke.

4.16 irudiko $3 \leq x \leq 6$ tarteko edozein puntutan, oinarriko funtzioen balioen batura 1 da. Horri esker, B-spline kurba osoa kontrol-puntuek osatzen duten inguratzaile konbexoaren barnean egongo da.

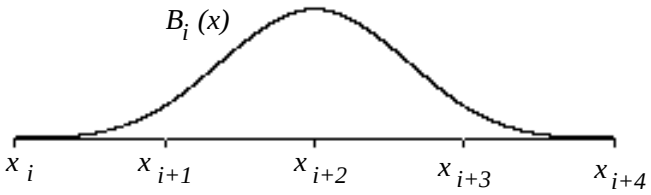
Azpikurba bakoitza marraztean erabiliko ditugun funtzioak 4.17 irudian ikus daitezke. Orokorrean, korapilo-balioak ez diren x balioen kasuetan oinarriko lau funtzio erabiliko ditugu. Korapilo-balioen kasuetan oinarriko funtzio baten balioa zero izango denez, oinarriko funtzio erabilgarriak hiru izango dira.

$m-2$ azpikurbaz definitutako B-splinea eraikitzeko, oinarrian $m+1$ funtzio eta $m+5$ korapilo-balio beharko dira.

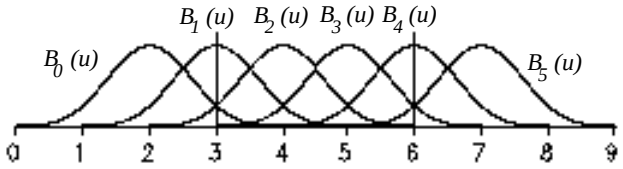
Itzul gaitezen 4.17 irudira. $x_i \leq x \leq x_{i+1}$ tartean B_i, B_{i-1}, B_{i-2} eta B_{i-3}



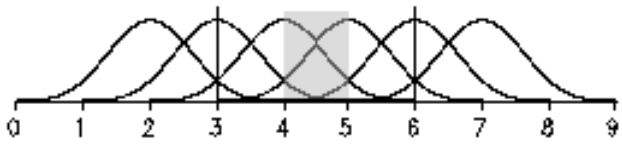
Irudia 4.14: a_4 kontrol-puntua mugitu arren S_3 azpikurba ez da aldatzen.



Irudia 4.15: B-splinea.



Irudia 4.16: Kontrol-puntu bakoitzak horietako B-spline bat kontrolatuko du.



Irudia 4.17: Azpikurba bakoitza definitzen duten lau funtzioak.

balioztatu beharko ditugu, $0 \leq x \leq 1$ tarteaz ordezkatuz:

$$B_i = 1/6x^3$$
$$B_{i-1} = 1/6(-3x^3 + 3x^2 + 3x + 1)$$
$$B_{i-2} = 1/6(3x^3 - 6x^2 + 4)$$
$$B_{i-3} = 1/6(1 - x)^3$$

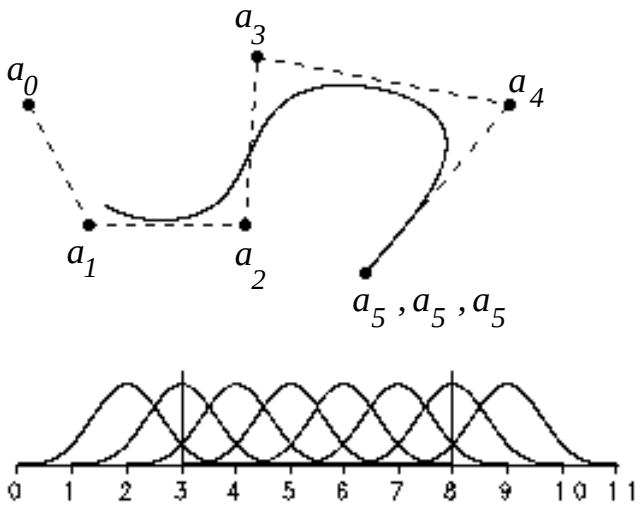
Horrela azpikurba bat lortuko dugu.

Dakigunez, B-splineak erabiliz ez ditugu kurbaren muturretako puntuak interpolatuko. Orokorrean B-splineek ez dute kontrol-punturik interpolatuko. Hala ere, kontrol-puntu berbera behin baino gehiagotan erabiliz, hau da, kontrol-puntu bat errepikatuz, kontrol-puntu errepikatua interpolatuko duen splinea lor dezakegu, nahiz eta teknika horren erabilerak, kurbaren jarraitasunaren galera ekarri. Kontrol-puntua errepikatuz puntuak kurbaren gain duen eragina handiagotzen dugu. Kontrol-puntua errepikatzearen ondorioz, azpikurba bat definitzerakoan kontrol-puntu berbera behin eta berriz erabiliko da. Konpara ditzagun 4.13 eta 4.18 irudietako kurbak. Bigarrenean azken kontrol-puntua hiru aldiz errepikatu da. Horren ondorioz, kurbak bost azpikurba ditu eta kontrol-puntu errepikatua (a_5) behin ebaluatzen da S_5 azpikurban, bi aldiz S_6 azpikurban eta hiru aldiz S_7 azpikurban. Kurba $3 \leq x \leq 8$ tartean definituta dago. $x = 8$ denean, kurbak a_5 interpolatzen du.

4.19 irudietan tarteko kontrol-puntu bat errepikatzeak kurban duen eragina ikus daiteke. a_3 kontrol-puntua behin errepikatzean, jarraitasuna C^2, G^2 izatetik C^2, G^1 izatera pasatzen da. Kontrol-puntu berdina bigarren aldiz errepikatzean, a_3 kurban hiru aldiz azalduz, kurbaren jarraitasuna C^2, G^0 izatera heltzen da.

Zenbait kontzeptu argi edukitzea komeni da:

- Oinarria, berdinak diren baina lekuz aldatuta dauden funtzioek osatzen dute. Oinarria osatzen duten funtzioak, 4 azpikurbaz osatutako



Irudia 4.18: a_5 hiru aldiz errepikatu ondoren, kurbak a_5 kontrol-puntua interpolatzen du.

oinarrizko spline kubikoak dira (B-splineak).

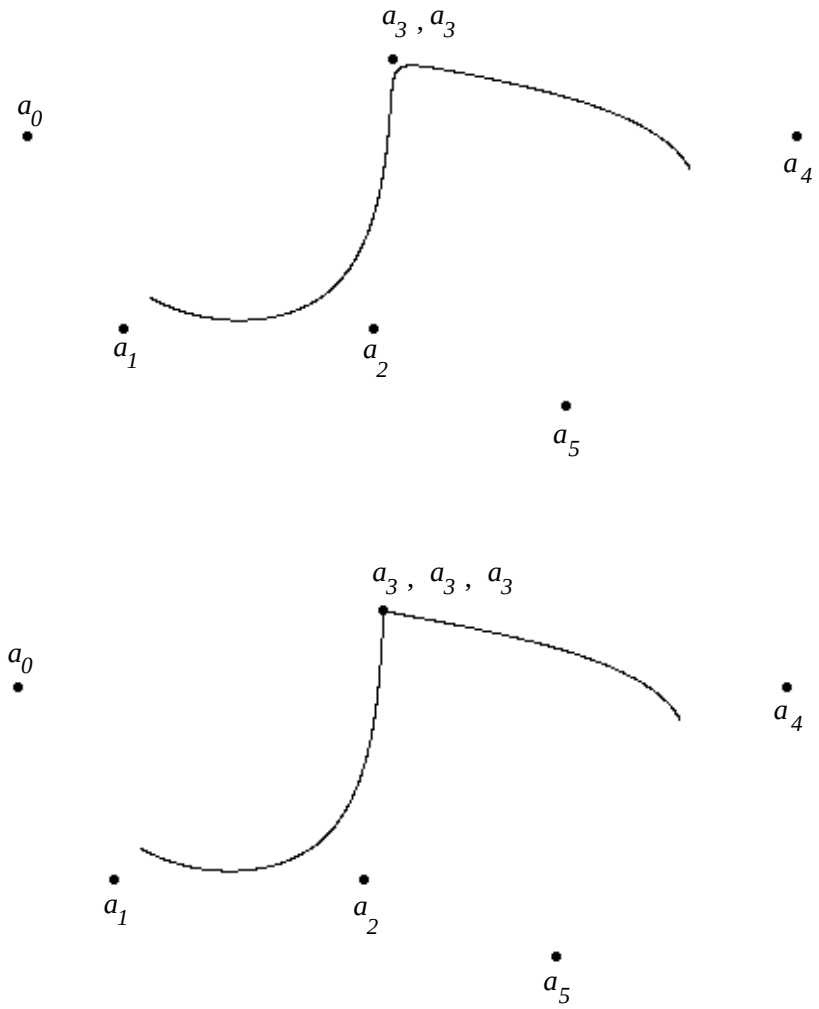
- Horrela lortutako kurba baten azpikurba bakoitza definitzerakoan, bata bestearen aldamenean dauden azpikurbek kontrol-puntuak konpartitzen dituzte. Horri esker, C^2 jarraitasuna mantentzen da.

- Kontrol-puntuak interpolatzeko, beren errepikapenaz baliatuko gara. Kontrol-puntu bat errepikatzea, kurban duen eragina handiagotzearen baliokidea da, eta oinarria osatzen duten funtzioek ez dute aldaketarik jasango.

- Interpolazioa egiteko kontrol-puntuen errepikapenaz baliatzean, jarraitasuna galduko dugu.

B-spline ez-uniformeak

B-spline ez uniformeetan korapilo-balioen arteko distantziek ez dute berdinak izan beharrik. B-spline ez-uniformeak erabiliz, bi korapilo-balioen arteko distantzia 0 izan daiteke eta oinarriko funtzioak, B-spline uniformeetan ez bezala, desberdinak izan daitezke beren artean. Beraz, puntu batean



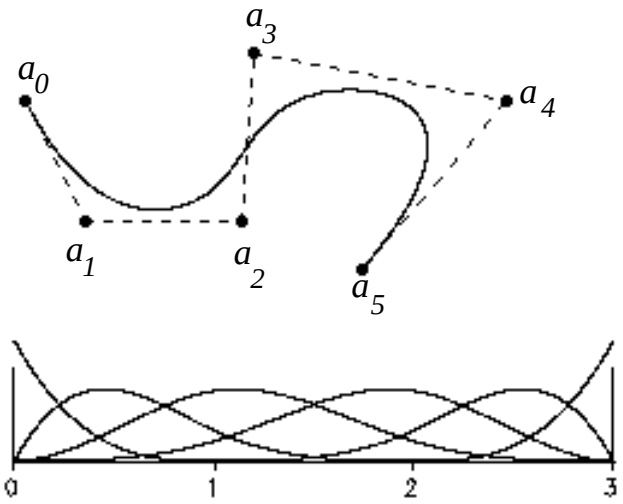
Irudia 4.19: Kontrol-puntuen errepikapenaren ondorioak.

korapilo-balio bat baino gehiago eduki daiteke. Ezaugarri hori interpolaziorako baliagarria da. Interpolazioa egiteko, metodo horren erabilerak zenbait abantaila du kontrol-puntuak errepikatzearekin konparatuz. Kontrol-puntuak errepikatuz, kontrol-puntu errepikatuaren bi aldeetako azpikurbak lerro zuzenak dira. B-spline ez uniformeekin kontrol-puntuak interpolatzean ez da hori gertatuko.

4.13 irudiko kurbari dagozkion korapilo-balioak $x = 3, 4, 5, 6$ dira. Korapilo-balioen arteko tartea unitate batekoa da. Korapilo-balio ez-uniformeak erabiliko bagenitu, oinarriko funtzioak ez lirateke berdinak izango. 4.20 irudiko kurba kalkulatzeko 4.13 irudiko kontrol-puntu berdinak erabili dira. Hala ere, kurba berriak muturretako puntuak interpolatzen ditu, korapiloen bektorearen muturretan korapilo-balioak errepikatu baitira. Erabilitako korapiloen bektorea [0,0,0,0,1,2,3,3,3,3] izan da. Eta bektore hori erabiltzearen ondorioz hartu beharko ditugun oinarriko funtzioak ez dira berdinak izango. 4.20 irudian oinarria osatzen duten B-spline ez uniformeak ere marraztuta ikus ditzakezu.

Oinarriko Spline bakoitzeko bost korapilo-balio behar ditugu; hau da, oinarriko spline bakoitzak lau tarte behar ditu. Spline ez-uniformeetan, ordea, tartearen luzera zero izan daiteke, eta adibideko korapilo-bektorearekin horixe gertatzen da. Lehenengo splinea [0,0,0,0,1] korapiloek zehazten dute; Spline horrek korapilo-balioen artean azpikurba bana edukiko du, baina lehenengo hiru tarteen luzera zero denez, tarte horietako azpikurbek puntu bakar bana daukate. Azken tartearen luzera ez da zero, beraz, tarte horri dagokion kurba, S_3 alegia, oinarriko lehenengo splineari itxura ematen dion azpikurba da. Bigarren splinearen itxura jakiteko, [0,0,0,1,2] korapilo-balioak aztertu behar ditugu. Lehenengo bi tarteen luzera zero denez, puntu bakar batez osatutako azpikurbak izango dira splinearen lehenengo biak. Beraz, beste bi tarteei dagozkien azpikurbak izango dira oinarriko bigarren splineari itxura emango dietenak; hau da, spline hori bi tarteetan zehar luzatuko da. 4.20 irudian ikus daiteke azaldutakoa. Irudiko azpialdean oinarriko splineak agertzen dira: lehenengoa zero eta bat puntuen arteko kurba da; bigarrena, berriz, zero eta 2 puntuen artean marraztutakoa ...

Korapilo-bektore horrek, 4.20 irudikoak, bederatzi tarte dauzka, S_0-tik S_8-ra zenbakituta daudenak. S_0, S_1, S_2 eta S_6, S_7, S_8 tarteen luzera zero denez, horiei dagozkien azpikurbak puntu bakar batez adierazten dira. Puntu bakar batera mugatzea, korapilo-balioen errepikapenaren ondorioa da. S_3, S_4 eta S_5 tarteei dagozkien azpikurbak ez dira puntu bakarrera mugatzen, horiei dagozkien azpikurbak 3. mailako polinomio bidez adieraziko dira.



Irudia 4.20: Lehenengo eta azken kontrol-puntuak interpolatzen dituen B-spline ez-uniformea.

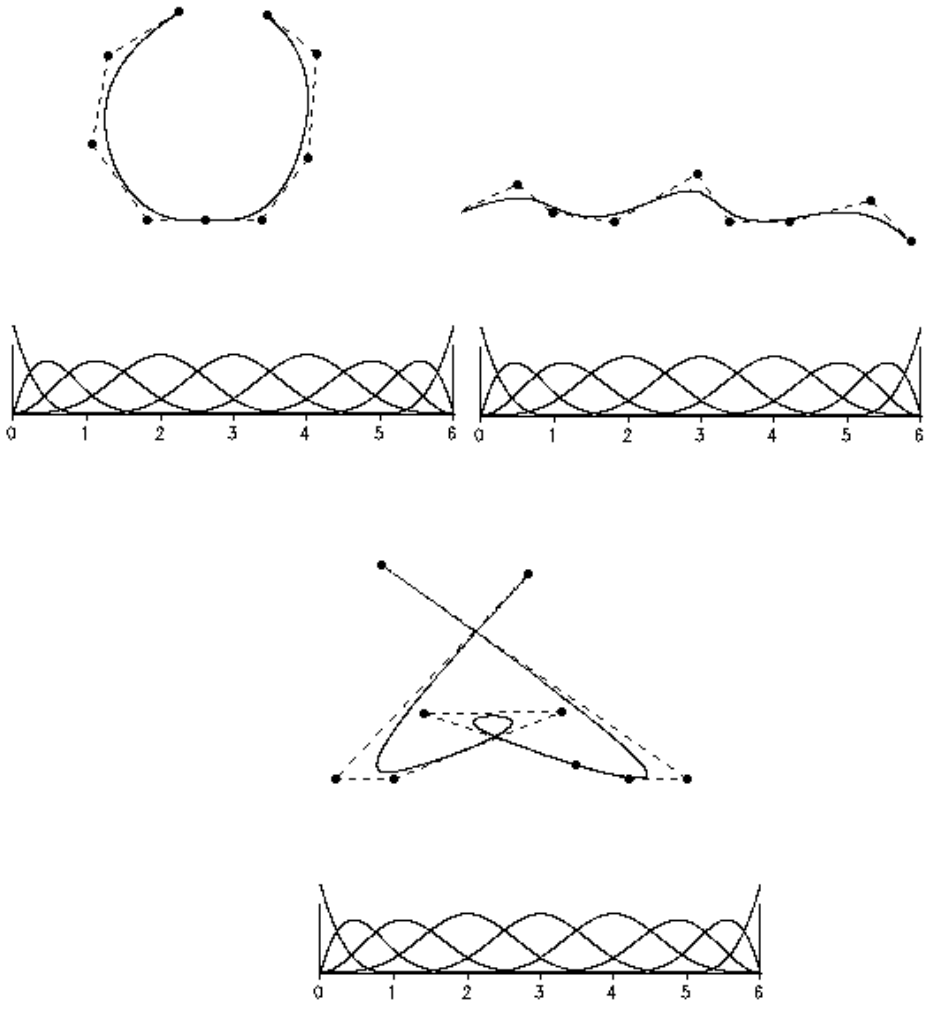
Zortzi tarte horiek oinarriko 6 spline ez-uniforme osatzen dituzte:

$S_0 - S_3 \rightarrow$ 1. splinea [0,1] tartean
$S_1 - S_4 \rightarrow$ 2. splinea [0,2] tartean
$S_2 - S_5 \rightarrow$ 3. splinea [0,3] tartean
$S_3 - S_6 \rightarrow$ 4. splinea [0,3] tartean
$S_4 - S_7 \rightarrow$ 5. splinea [1,3] tartean
$S_5 - S_8 \rightarrow$ 6. splinea [2,3] tartean

B-spline ez-uniformeen malgutasuna azaltzen duten adibide batzuk 4.21 irudian ikus daitezke. Irudiko kurbak marrazteko erabili den korapiloen bektorea, [0,0,0,0,1,2,3,4,5,6,6,6,6] izan da.

Korapiloen bektore bat, x_0-tik x_{m+4}-ra doan segida ez-beherakorra da. Ikusi dugun bezala, ondoz ondoan dauden korapilo-balioak berdinak izan daitezke eta bektorean korapilo-balio berdinak dituzten korapilo-kopuruari, **anizkoiztasuna** deritzo.

[0,0,0,0,1,1,1,1] korapiloen bektorea erabiltzean, a_0 eta a_3 puntuak interpolatuko dituen azpikurba bat lortuko dugu. Korapiloen bektore horrek definitutako oinarriko funtzioak, Bezier-en oinarriko funtzioak izango dira.



Irudia 4.21: B-spline kurben malgutasunaren adibideak. Kurba guztietan [0,0,0,0,1,2,3,4,5,6,6,6,6] korapilo-bektorea erabili da.

Azter dezagun korapiloen errepikapenak oinarriko funtzioetan duen eragina. 4.22 irudiko lehenengo marrazkian, 0,1,2,3,4 korapilo-balioek definitzen duten B-spline uniformea ikus daiteke. Azaldu dugunez, B-spline uniformea lau polinomio kubikoren bidez definitzen da, eta lau polinomio kubiko horietako bakoitza zein tartetan definituta dagoen finkatzeko, 0,1,2,3 eta 4 korapilo-balioak erabiltzen dira.

$$B_0(x) = \begin{cases} b_{0-0}(x) = 1/6x^3 & 0 \leq x \leq 1 \\ b_{0-1}(x) = -1/6(3x^3 - 12x^2 + 12x - 4) & 1 \leq x \leq 2 \\ b_{0-2}(x) = -1/6(3x^3 - 24x^2 + 60x - 44) & 2 \leq x \leq 3 \\ b_{0-3}(x) = -1/6(x^3 - 12x^2 + 48x - 64) & 3 \leq x \leq 4 \end{cases}$$

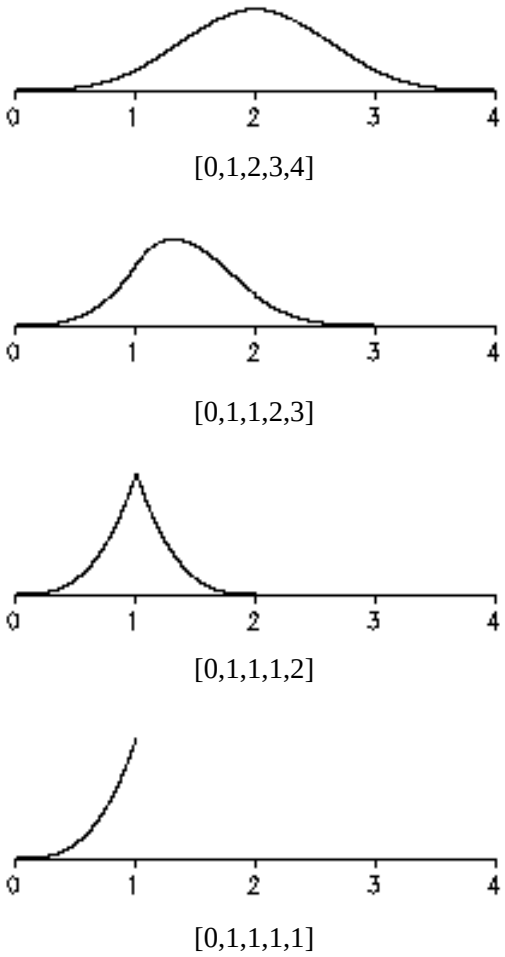
Bigarren korapiloa bikoizten badugu [0,1,1,2,3], b_{0-1} funtzioa [1,1] tartean egongo da definituta, kurban izango duen eragina nulua izanik. [0,1,1,2,3] bektorearen eraginez izango genukeen oinarriko B_0 funtzioa, 4.22 irudian ikus daiteke. Korapilo bat bikoiztean, bigarren deribatuaren jarraitasuna galtzen dugu, lehenengo deribatuaren jarraitasuna mantenduz. Bigarren korapiloa hirukoiztean, lehenengo deribatuaren jarraitasuna galtzen dugu, kokapen--jarraitasuna edo G^0 mantenduz. Bigarren korapiloa laukoiztean, [0,1,1,1,1], kokapen-jarraitasuna ere galdu egingo dugu. Zenbat eta errepikapen gehiago egin, orduan eta jarraitasun gutxiago izango dugu.

4.20 irudiko adibidean, oinarriko lehenengo funtzioa ([0,0,0,0,1] korapiloen bektorea duena) kokapen-jarraitasunik gabeko funtzio asimetrikoa da. Bigarren funtzioa korapilo bat hirukoiztua duen bektore baten bitartez definitzen da, [0,0,0,1,2]. Hirugarrena ere asimetrikoa da, [0,0,1,2,3]. Adibide horretako funtzio guztiak asimetrikoak dira:

Korapiloen bektorea	Oinarriko funtzioa
00001	B_0
00012	B_1
00123	B_2
01233	B_3
12333	B_4
23333	B_5

Orain, korapilo-bektoreko muturretan ez dauden korapiloak errepikatzean zer gertatzen den aztertuko dugu. Azalpenak 4.23 eta 4.24 irudietako adibideetan oinarrituta daude. Bertan lau kurba ikus daitezke.

4.23 irudiko lehenengo kurbaren korapilo-bektorea [0,1,2,3,4,5,6,7,8,9,10] da; korapilo-bektore hori uniformea da eta ondorioz oinarria osatzen duten



Irudia 4.22: Irudi honetan, oinarriko funtzioen gain korapiloaren errepikapenak duen eragina ikus daiteke.

spline guztiak berdinak dira. Korapilo-bektore horrekin, 11 korapilo dituenez, oinarriko zazpi spline osa ditzakegu, eta horiekin lau tartetan marraz dezakegu zazpi kontrol-puntuk kontrolatutako kurba. Lehenengo azpikurba [3,4] tarteari dagokiona da; bertan lau spline dauzkagu, a_0, a_1, a_2 eta a_3 kontrol-puntuek kontrolatzen dituztenak. Hurrengo azpikurba [4,5] tarteari dagokiona da, hirugarrena [5,6] tartekoa izango litzateke eta azkena [6,7] tarteari dagokio. Azken horretan a_3, a_4, a_5 eta a_6 kontrol-puntuek kontrolatzen dute azpikurbaren itxura.

Irudiko bigarren kurban, goialdeko eskuinekoan, korapilo-bektoreko erdialdeko korapilo bat bikoizteak kurban duen eragina ikus daiteke. Tarte baten luzera zero da. Ondorioz, oinarria osatzen duten splineen itxura aldatu egin da, ez guztiena, baina bai tarte hori ukitzen dutenena. Horren ondorio zuzena, kurba osatzen duten azpikurben kopurua hirura pasatzea izan da; lehenengo azpikurba [3,4] tarteari dagokiona da, eta hori da a_0, a_1, a_2 eta a_3 kontrol-puntuek kontrolatzen dutena. Bigarren azpikurba [4,5] tartean definitzen da eta a_2, a_3, a_4 eta a_5 puntuek kontrolatzen dute. Hirugarrena, [5,6] tartekoa, a_3, a_4, a_5 eta a_6 puntuek kontrolatzen dute. Lau azpikurba behar ziren lekuan hiru marraztu ditugu, baina ongi aztertuz gero, lau daudela esan genezake; laugarren hori [4,4] tartean definitzen da, hau da, puntu bakarrak osatzen du, bere balioa [3,4] tarteko azpikurbako bukaerako balioa da, eta era berean [4,5] tartekoaren hasierako balioa. Gainera, puntu horretako balioa a_1, a_2, a_3 eta a_4 puntuek kontrolatzen dutela ere esan genezake, baina a_1 eta a_4 puntuek kontrolatutako splineek puntu horretan zero balioa dute; ondorioz, a_2 eta a_3 kontrol-puntuen arteko balioa lortuko dugu. Hori dela eta, lehenengo azpikurbaren bukaera bi kontrol-puntu horiek lotzen dituen ertzean dago; eta, jakina, hurrengoaren hasiera ere horixe bera da.

4.24 irudiko lehenengo kurban, korapilo bat hirugarren aldiz errepikatzen da. Oinarriko splineak berriz aldatu dira eta oraingoan bi dira zeroko luzera duten tarteak. Lehenengo azpikurba a_0, a_1, a_2 eta a_3 puntuek kontrolatzen dute, ondoren puntu bakarrez osatutako bi azpikurba daude, eta azkenik, a_3, a_4, a_5 eta a_6 puntuek kontrolatutako azpikurba. Hala ere, kokapen-jarraitasuna mantendu egin da; baina korapiloa berriz errepikatzean, azken kurban egiten den bezala, eta eskuinaldean beste kontrol-puntu bat jartzen badugu, jarraitasuna galtzen da. Horren arrazoia ulertzeko, oinarriko splineen itxura begiratu behar da. Laugarren splinea [3,4] tartean dago definituta; hau da, [3,4,4,4,4] korapilo-balioek finkatzen dituzte bere mugak. Tartearen bukaeran aurreko hiru splineen balioa zero da eta spline horrena 1; beraz, $t = 4$ puntuan spline hori kontrolatzen duen a_3 kontrol-puntuaren

balioa hartuko du kurbak. Hurrengo splinea [4,5] tartean dago definituta, [4,4,4,4,5] korapilo-balioek mugatzen duten tartea alegia. Tarte horren hasieran, $t = 4$ puntuan, bosgarren splinearen balioa 1 da eta tarte horretan agertzen diren beste hiru splineek zero balio dute; ondorioz, azpikurbak hartzen duen balioa, spline hori kontrolatzen duen kontrol-puntuarena izango da, irudiko kasuan a_4. Horra hor etena, a_3 puntutik a_4 puntura pasa gara tartean ezer marraztu gabe.

B-spline ez-uniformeen oinarriko funtzioak kalkulatzeko erabilgarria den algoritmo bat, Cox-De Boor algoritmoa da (De Boor, 1972; Cox, 1972). Horrez gain, algoritmo errekurtsibo horri esker edozein mailatako B-spline uniforme edo ez-uniformeak kalkulatzeko gai izango gara.

$$B_{i,1}(x) = \begin{cases} 1 & x_i \leq x \leq x_{i+1} \\ 0 & \text{bestela} \end{cases}$$

$$B_{i,2}(x) = \frac{x-x_i}{x_{i+1}-x_i} B_{i,1}(x) + \frac{x_{i+2}-x}{x_{i+2}-x_{i+1}} B_{i+1,1}(x)$$

$$B_{i,3}(x) = \frac{x-x_i}{x_{i+2}-x_i} B_{i,2}(x) + \frac{x_{i+3}-x}{x_{i+3}-x_{i+1}} B_{i+1,2}(x)$$

$$B_{i,4}(x) = \frac{x-x_i}{x_{i+3}-x_i} B_{i,3}(x) + \frac{x_{i+4}-x}{x_{i+4}-x_{i+1}} B_{i+1,3}(x)$$

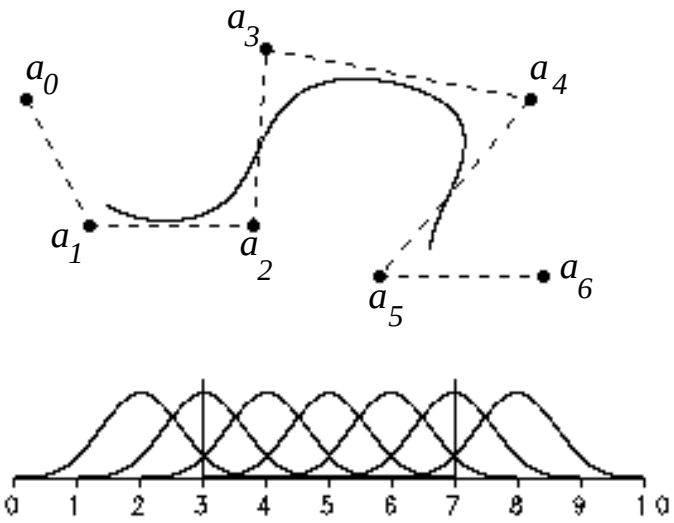
Korapiloak errepikatzerakoan, $0/0$ eragiketarekin aurkituko gara; $0/0 = 0$ berdintza egia dela suposatzen da. Algoritmoaren inplementazioan, zenbakitzailea 0 denentz aztertu beharko litzateke, eta horrela balitz, zatiketaren emaitza modura 0 erabili beharko genuke. CAD sistema komertzialetan, B-splineek erabiltzen duten korapilo-bektorea aurredefinituta egon daiteke.

B-splineen propietateen laburpena

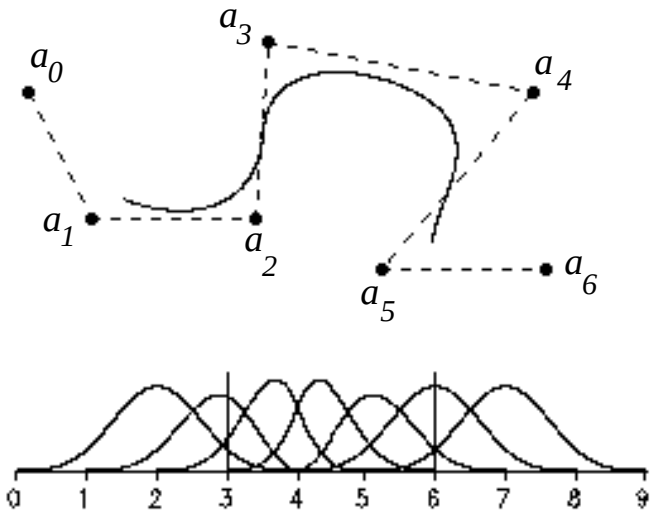
- kurbak kontrol-puntuek adierazten duten poligonoaren itxura hartzen du eta horiek mugatzen duten inguratzaile konbexoaren barnean kokatuko da.

- Kurbaren gain edozein aldaketa afin (hau da, aldaketa linealen edozein konbinazio) egiteko, aldaketa kontrol-puntuen gain eragin behar da.

- B-splineak itxura-kontrolerako oso egokiak dira, kontrol lokala eskaintzen digutelako; hau da, a_i koefiziente bat aldatzen badugu, kurba

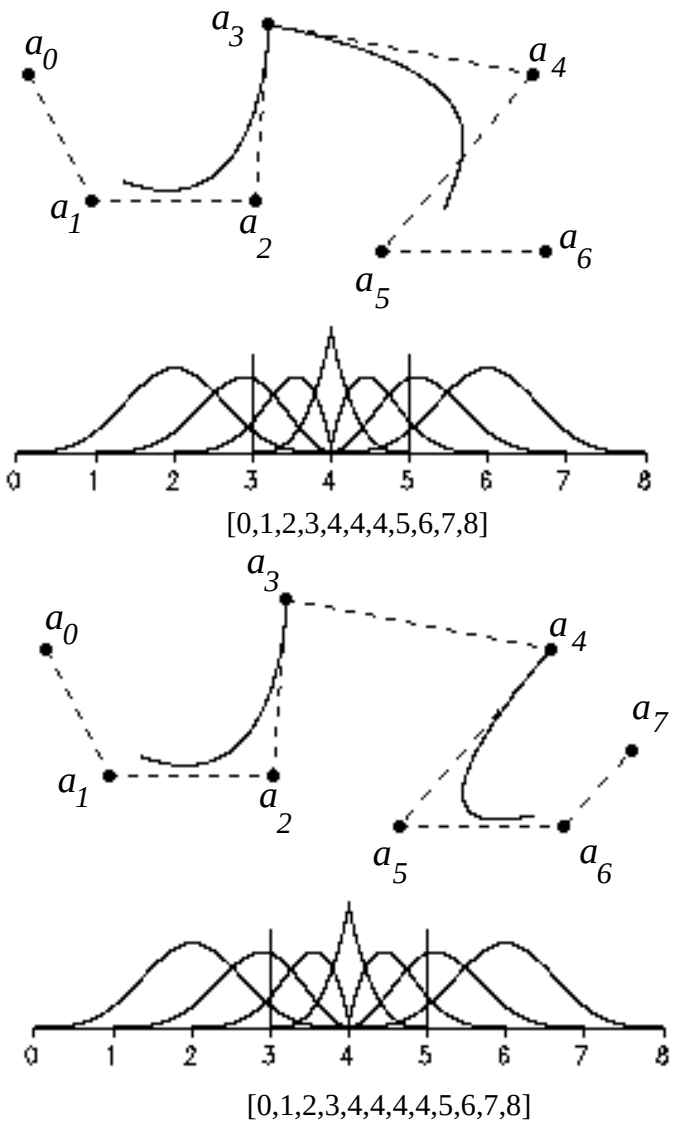


[0,1,2,3,4,5,6,7,8,9,10]



[0,1,2,3,4,4,5,6,7,8,9]

Irudia 4.23: Korapiloak errepikatzearen ondorioak: S_4 segmentua puntu bihurtu da.



Irudia 4.24: Korapiloak errepikatzearen ondorioak (jarraipena): goiko irudian korapilo bat hiru aldiz errepikatu da, ondorioz, S_4 eta S_5 puntu bihurtu dira; S_3 eta S_4 ukitu egiten dira. Beheko irudian, korapiloa lau aldiz errepikatzean, hiru segmentu puntu bihurtu dira, eta kurban etena eragin dute.

berriaren eta zaharraren arteko diferentzia

$$(a_i - \overline{a_i})B_i(x)$$

izango da. B_i bakoitzak 4 azpitartetan du eragina; kurbaren aldaketa 4 azpitarte horietan emango da. Gainontzeko azpitarteek ez dute aldaketarik jasango.

- x=x_k puntu baten kasuan, $S(x_k) = \frac{a_{k-1}+4a_k+a_{k+1}}{6} = s_0$. Balio hori a_k-tik oso gertu dago. Kurba kontrol-puntuek osatutako poligonoari egokitzera behartuta dago eta egokitze hori ingurune txiki batean gertatzen da.

- Aurreko ezaugarria dela eta B-splineak interpolatzeko ere erabil daitezke. $S(x_i)$ balioak finka ditzakegu eta horien arabera a_i aldagaien edo kontrol-puntuen balioak kalkula ditzakegu; n ekuazio eta $n+2$ ezezaguneko ekuazio-sistema lortuko dugu:

$$\begin{pmatrix} 4 & 1 & 0 & \dots & & & 0 \\ 1 & 4 & 1 & \dots & & & 0 \\ 0 & 1 & 4 & \dots & & & 0 \\ \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & \dots & & 1 & 4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} 6y_1 - a_0 \\ 6y_2 \\ 6y_3 \\ \vdots \\ 6y_{n-1} \\ 6y_n - a_{n+1} \end{pmatrix}$$

a_0 eta a_{n+1} aldagaiei balio bana emanda, erraza da ekuazio-sistema hori askatzea.

4.7 Ekuazio parametrikoetan oinarritutako inplementazioa

Azaldutako guztiaren inplementazioa ekuazio parametrikoak erabiliz egin behar da, aldaketa afinekiko aldaezinak direlako eta marrazteko erraztasuna ematen dutelako.

4.7.1 Interpolazioa

$(x_0, y_0), \ldots, (x_n, y_n)$ bi dimentsioko puntu-zerrenda bat izanik, horien interpolazioa ekuazio parametrikoetan oinarrituz egiteko $x = f(t)$ eta $y = g(t)$ funtzioak kalkulatu beharko ditugu. Bi funtzio horiek independenteki kalkula daitezke, M matrizeak, funtzio interpolatzailea aurkitzeko erabiltzen dugun matrizeak, OX ardatzeko puntuak erabiltzen baititu (kasu horretan OT):

$$\begin{pmatrix} g_0(t_0) & g_1(t_0) & \ldots & g_n(t_0) \\ \vdots & \vdots & & \vdots \\ g_0(t_n) & g_1(t_n) & \ldots & g_n(t_n) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} x_0 \\ \vdots \\ x_n \end{pmatrix} \text{edo} \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

4.7.2 Egokitzea

Gauza bera gertatzen da:

$$\begin{pmatrix} g_{00} & g_{01} & \cdots & g_{0m} \\ g_{10} & g_{11} & \cdots & g_{1m} \\ \vdots & \vdots & & \vdots \\ g_{m0} & g_{m1} & \cdots & g_{mm} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{pmatrix}$$

4.7.3 Itxuraren kontrola

Oraingoan, kontrol-poligonoaren manipulazioa bi aldagaietan egin daiteke. Bernstein-en oinarria erabiliz:

$$x = \sum x_i b_i(t)$$

$$y = \sum y_i b_i(t)$$

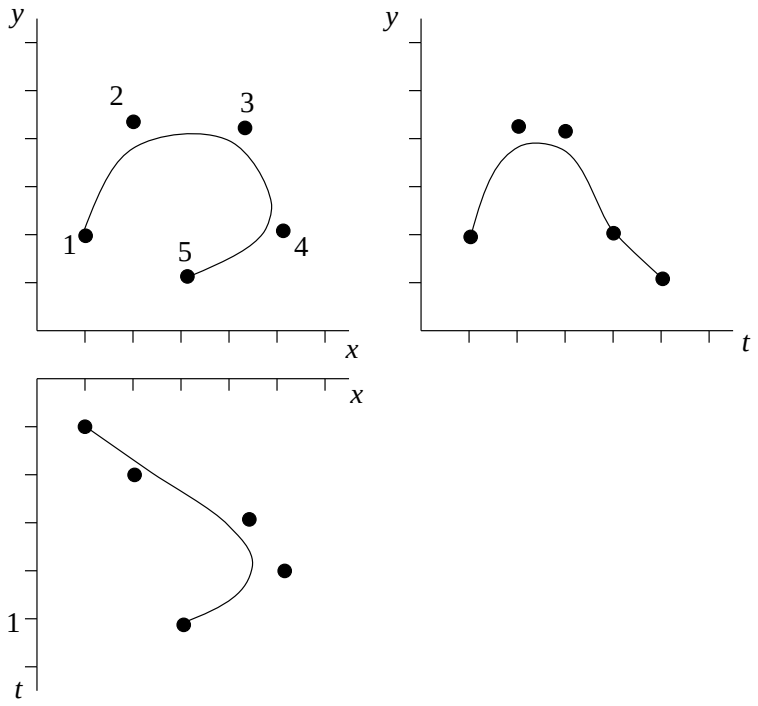
t aldagaiak ez du [0,1] tartean egon beharrik; beraz, ez dugu aldagai aldaketarik egingo.

Bernstein-en oinarria erabiliz, (x_0,y_0) eta (x_n,y_n) puntuak interpolatzen dituzten ondoko bi funtzioak lortuko genituzke:

$$x = \sum_{i=0}^{n} x_i \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i}$$

$$y = \sum_{i=0}^{n} y_i \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i}$$

Funtzio horiek marrazten dituzten grafikoak 4.25 irudikoak dira.



Irudia 4.25: Itxuraren kontrolerako x eta y koordenatuak bakoitza bere aldetik hartuz lortzen diren bi funtzioen grafikoak, eta horrela lortutako $f(x, y)$ kurba.

4.8 Gainazal erregularrak eraikitzen

Hiru dimentsioko gainazalak sortzea, bi dimentsioko kurbak eraikitzea baino konplexuagoa da. Eta eraikitako gainazalak ikusteko errealismoa lortu nahi duten algoritmoek, gainazaletan egin beharko dute lan (ezkutuko azalerak ezabatuz, argiztatuz,...). Gainazalen adierazpenak:

Esplizitua: $z = f(x, y)$, adierazpen horrekin ezkutuko zatien ezabapenerako algoritmo onak garatu dira, baina, hala ere, ez da asko erabiltzen.

Inplizitua: $f(x, y, z) = 0$ oso gutxi erabiltzen da. Izpi-hedaketaren algoritmoa erabil daiteke gainazalak irudikatzeko.

Parametrikoak: $x = f(u, v), y = g(u, v)$ eta $z = h(u, v)$. Gehien erabiltzen den adierazpena da.

Ikuspuntu praktiko batetik oso zaila da horrelako adierazpenekin lan egitea, eta ondokoa erabiltzen da:

$$z = f(x, y) = f_1(x)f_2(y)$$

Hirugarren aldagaia, beste bi aldagaien funtzioen arteko biderkaketa modura definitzen da; $f(x, y)$ delakoa f_1 eta f_2-ren biderkaketa tentsoriala dela esaten da.

4.8.1 Interpolazioa

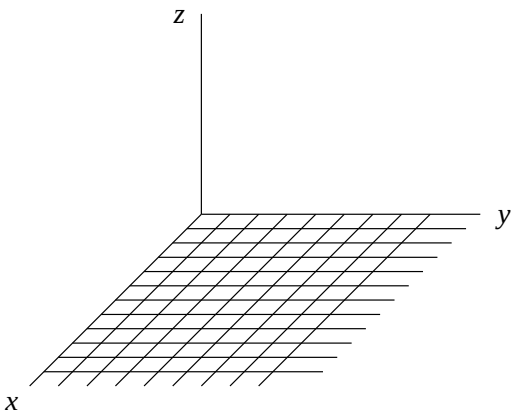
Suposa dezagun gainazala bi funtzioen arteko biderkaketa tentsorialaren bidez definituta dagoela, eta euskarri gisa $x_0, \ldots, x_n$ eta $y_0, \ldots, y_m$ puntuak dituela. Euskarriko puntu horiek bi dimentsioko sare bat, 4.26 irudikoa alegia, adieraziko dute hiru dimentsioko espazioan.

Edozein $z_{ij} = f(x_i, y_j)$ gainazal interpolatzeko, $n+1$ funtzio interpolatzaile kalkulatuko ditugu, y_j bakoitzerako funtzio bat, 4.27 irudian dakusagun bezala.

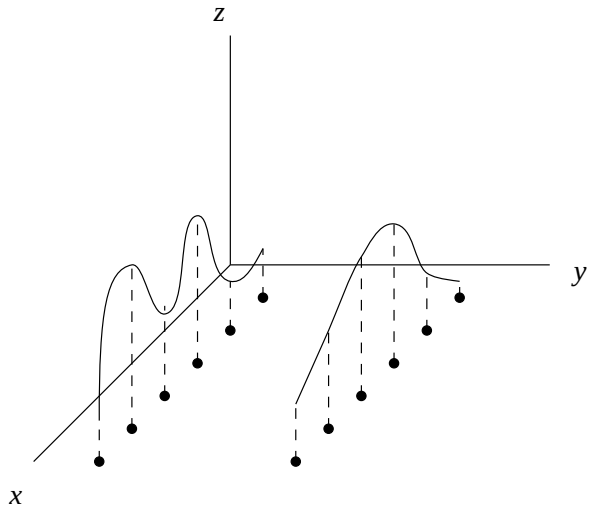
Beraz, g_0,...,g_n funtzioen oinarria erabiliz, ondokoa bilatzen ari gara:

$$\begin{array}{lllll}
g(x,0) = \sum_{i=0}^{n} a_{i0} g_i(x) & \text{funtzioak} & z_{00} \ldots z_{n0} & \text{interpolatzen ditu} \\
g(x,1) = \sum_{i=0}^{n} a_{i1} g_i(x) & \text{funtzioak} & z_{01} \ldots z_{n1} & \text{interpolatzen ditu} \\
\vdots & & & \vdots \\
g(x,m) = \sum_{i=0}^{n} a_{im} g_i(x) & \text{funtzioak} & z_{0m} \ldots z_{nm} & \text{interpolatzen ditu}
\end{array}$$

Ekuazioak berridatziz:



Irudia 4.26: Euskarriak adierazten duen sarea.



Irudia 4.27: y_j bakoitzerako funtzio interpolatzaile bat kalkulatuko dugu.

$$g(x,0) = a_{00}g_0(x) + a_{10}g_1(x) + \ldots + a_{n0}g_n(x)$$

$$g(x,1) = a_{01}g_0(x) + a_{11}g_1(x) + \ldots + a_{n1}g_n(x)$$

$$\vdots$$

$$g(x,m) = a_{0m}g_0(x) + a_{1m}g_1(x) + \ldots + a_{nm}g_n(x)$$

Bestalde, $m+1$ funtzioko oinarri baten bidez $a_{00}, a_{01}, \ldots a_{0m}$ interpolatzen dituen funtzioa lortzeko gai gara, eta era berean ondoko funtzioak lor ditzakegu:

$$\begin{array}{ccc} h(0,y) = \sum_{j=0}^{m} b_{0j}h_j(y) & \text{funtzioak} & a_{00} \ldots a_{0m} \text{ interpolatzen ditu} \\ \vdots & & \vdots \\ h(n,y) = \sum_{j=0}^{m} b_{nj}h_j(y) & \text{funtzioak} & a_{n0} \ldots a_{nm} \text{ interpolatzen ditu} \end{array}$$

Ikusitako guztiaren ondorioz:

$$f(x,y) = \sum_{i=0}^{n} a_{ij}g_i(x) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}h_j(y)g_i(x)$$

(x, y_m) kasuan:

$$\sum_{i=0}^{n} \Big(\sum_{j=0}^{m} b_{ij}h_j(y_m) \Big) g_i(x) = \sum_{i=0}^{n} h(i,y)$$

(x_2, y_3) kasuan:

$$\sum_{i=0}^{n} \Big(\sum_{j=0}^{m} \Big(b_{ij}h_j(y_3) \Big) g_i(x_2) \Big) = \sum_{i=0}^{n} a_{i3}g_i(x_2) = z_{23}$$

$f(x,y)$ beste era batean idatziz:

$$\Big(\sum_{i=0}^{n} r_i g_i(x) \Big) \Big(\sum_{j=0}^{m} s_j h_j(y) \Big) = f_1(x)\, f_2(y)$$

$r_i s_j = b_{ij}$ izanik.

4.8.2 Itxuraren kontrola

Interpolazioaren kasuan egin dugun bide berbera egin dezakegu gainazalaren adierazpena lortzeko. Interpolazioan bezala, gainazala definitzeko kontrol-sare bat erabiliko dugu, eta bilatzen ari garen funtzioa bi funtzioren bidez adieraziko dugu:

$$f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij} g_i(x) h_j(y)$$

Lortuko dugun $f(x,y)$ funtzioa, erabilitako oinarriaren araberakoa izango da, eta z_{ij} balio bat aldatzean gure funtzioak aldaketa lokal bat jasatea izango litzateke egokiena. Oinarriko funtzio gisa Bernstein-en polinomioak edo B-spline kubikoak erabil daitezke.

Bezier gainazalek Bernstein-en oinarria erabiltzen dute:

$$B_i(x) = \left(\begin{array}{c} n \\ i \end{array} \right) x^i (1-x)^{n-i}$$

$$P(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} \left(\begin{array}{c} n \\ i \end{array} \right) \left(\begin{array}{c} m \\ j \end{array} \right) y^j (1-y)^{m-j} x^i (1-x)^{n-i} z_{ij}$$

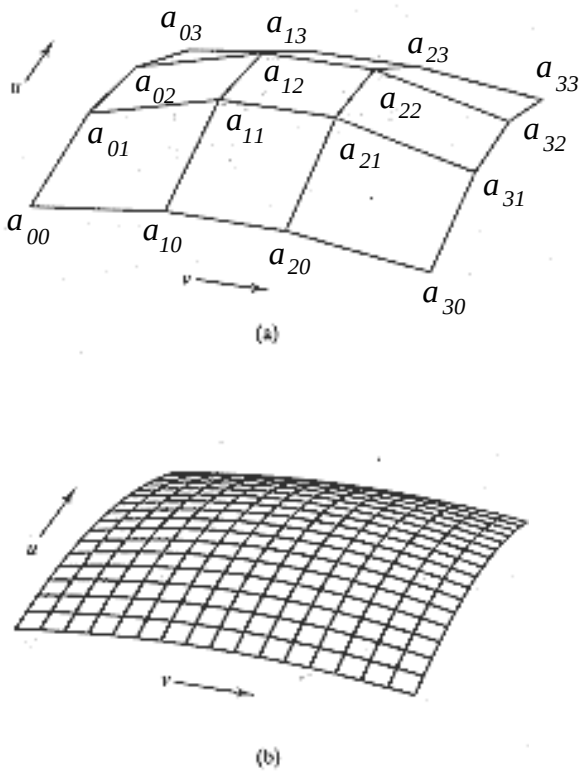
Poligono horrek Bezier kurben propietate berdinak izango ditu.

4.9 Gainazal biparametriko kubikoak

Txatal biparametriko kubikoak, kurba parametriko kubikoen orokorpenak dira. Gainazaleko txatal baten puntu bat funtzio biparametriko baten bidez kalkula daiteke. Funtzio biparametrikoko parametro bakoitzari funtzioen oinarri bat dagokio. Beraz, oinarrizko funtzioen bi multzo izango ditugu. Bezier-en txatal kubiko baten definizioa:

$$P(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} B_i(u) B_j(v)$$

Bezier txatal bat eta bere kontrol-puntuak 4.28 irudian ikus daitezke; bertan, txatala marrazteko lerro isoparametrikoak erabili dira. Irudia begiratuz, kontrol-puntuek osatzen duten sarearen kanpoaldeko 12 puntuek txatalaren



Irudia 4.28: (a) Kontrol-puntuz osaturiko poliedroa (b) Aurreko kontrol-puntuek definitzen duten gainazala.

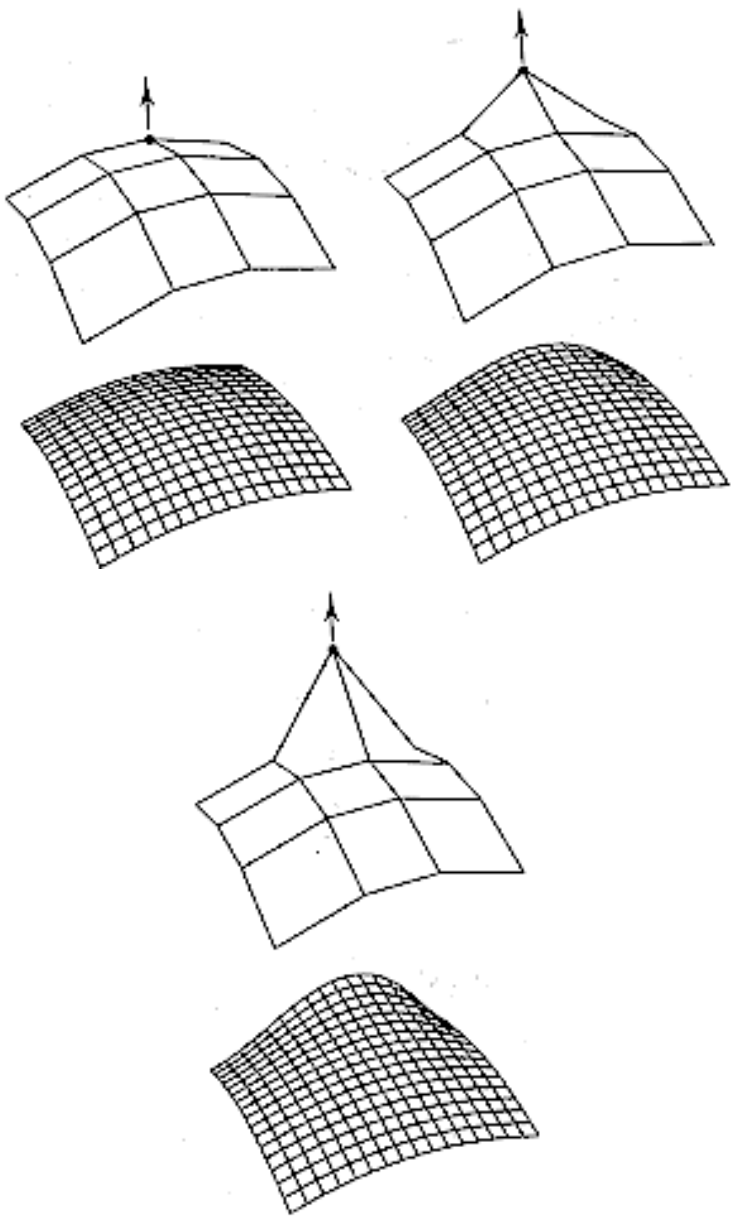
lau muturrak definitzen dituztela ikus daiteke, eta barnealdeko lau kontrol--puntuek, gainazalaren barnealdearen itxura kontrolatzen dute.

4.29 irudian, Bezier txatalen erabilera zein intuitiboa den ikusten da argi. Gainazalak lehenengo mailako jarraitasuna mantentzen du. Eta horren gain aldaketa bat eragin nahi badugu, nahikoa izango da gainazala definitzen duten kontrol-puntuen gain egitea aldaketa.

Kontrol-puntuek gainazalaren gain duten eragina, kurben gain duten eraginaren parekoa da. Bestalde, Bezier formulazioari esker, erabiltzaileak ez du bektore ukitzaileekin eta horrelako kontzeptuekin lan egin beharrik.

$P(u, v)$[2] Bezier gainazala matrize moduan idatziz:

[2]Gainazalen definizio-espazio gisa, (u, v) espazioa izendatuko dugu.



Irudia 4.29: Kontrol-puntu bat mugitzeak txatalaren gain duen eragina.

$$P(u,v) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} B & A & B^T \end{pmatrix} \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix}$$

Eta:

$$B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

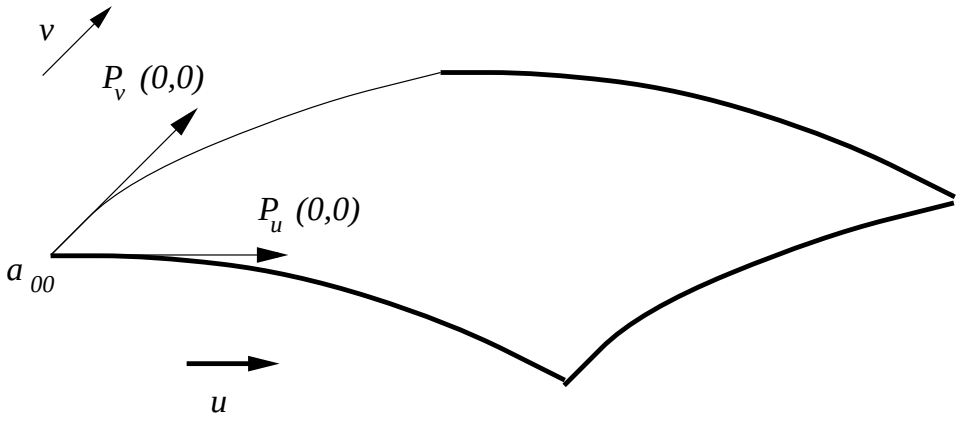
$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Bezier gainazalen interpretazio geometrikoa Bezier kurbena baino konplexuagoa da. Hala ere, txatal bateko izkinetako bektore deribagarrien eta kontrol-puntuen arteko erlazioa aztertzea komeni da. Adibidez, har dezagun $u = v = 0$ izkina. a_{00} erpinarekin erlazionatuta dauden bektoreen eta gainontzeko kontrol-puntuen arteko erlazioa ondokoa da:

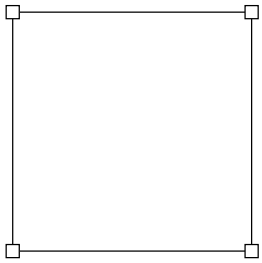
$$\begin{aligned} P_u(0,0) &= 3(a_{10} - a_{00}) \\ P_v(0,0) &= 3(a_{01} - a_{00}) \\ P_{uv}(0,0) &= 9(a_{00} - a_{01} - a_{10} - a_{11}) \end{aligned} \tag{4.1}$$

4.30 irudian bektore horiek azaltzen dira. $P_u(0,0), P(0,0)$ puntuko v norabideko bektore ukitzailearen eta konstante baten arteko biderkadura da. $P_v(0,0)$ ere, konstante baten eta $P(0,0)$ puntuko (v norabideko) bektore ukitzailearen arteko biderkadura izanik. Izkinetako puntuen P_{uv} bektoreak, batzuetan bihurdura-bektore izenarekin ezagutzen denak, u eta v-rekiko bektore ukitzaileen aldaketaren arrazoia ematen digu. Bihurdura-bektorea, bektore ukitzaileek definitutako planoarekiko elkarzuta da.

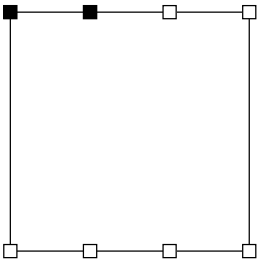
Gainazal baten gain argiztapen kalkuluak egiteko, gainazalaren normalak behar ditugu. Eta horiek kalkulatzeko bi aukera ditugu:



Irudia 4.30: a_{00} kontrol-puntuko bektore ukitzaileak.

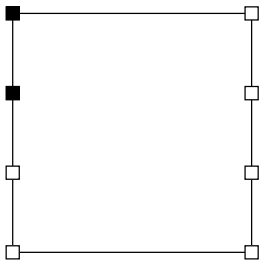


Izkinetako puntuak



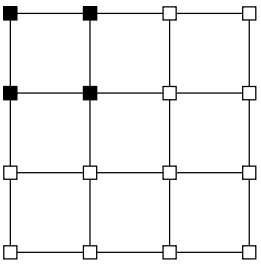
v norabideko bektore tangentea

$P_v(0,0) = 3\,(a_{01} - a_{00})$



u norabidekoa:

$P_u(0,0) = 3\,(a_{10} - a_{00})$



(0,0)-ko bihurdura-bektorea:

$P_{uv}(0,0) = 9\,(a_{00} - a_{01} - a_{10} + a_{11})$

Irudia 4.31: Bihurdura-bektoreak.

- Txatala azpitxataletan zatitzea, ia plano batean egongo diren txatalak lortu arte (aukera hori aurrerago azalduko dugu); horrela, plano batean dagoela kontsideratuz Gouraud, Phong edo beste argiztatze ereduren bat eragin ahal izango diegu. Erpin bakoitzeko bektore normala, bi bektore ukitzaileen arteko bektore-biderkaketa kalkulatuz lor daiteke. Adibidez:

$$a = a_{01} - a_{00}$$
$$b = a_{10} - a_{00}$$
$$N = a \times b$$

- Bi deribatu partzialen (∂P/∂u eta ∂P/∂v) arteko bektore-biderkaketa erabiltzea. Horrela gainazalaren puntu bakoitzeko normala kalkulatuko genuke. Baina bigarren aukera hori, konputazio-kostua dela-eta, ez da bideragarria. Gehiegizkoa izango litzateke gainazala osatzen duten puntu guztien normala kalkulatzea, ondoren, horiengan argiztatze ereduren bat eragiteko.

Txatalaren zehaztasunari esker, txatal eta poligonozko objektuen artean itzulpen desberdinak egin daitezke, ikuslearen distantziaren arabera. Horrela, ikuslea objektura gerturatzean, itzulpen zehatzagoa eginez, objektuaren itxura egokia kalkulatu ahal izango dugu, ikusleak objektua osatzen duten poligonoak nabarmendu ez ditzan. Hala ere, objektuak eraikitzeko eskaintzen duten erraztasuna da txatalen abantaila nagusia.

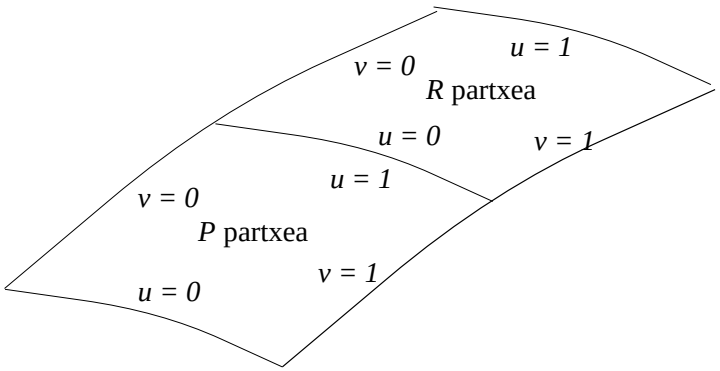
4.9.1 Bezier gainazalak elkartzen

Bi txatalen arteko lehen mailako jarraitasuna mantentzea, kurben arteko elkarketaren arazoaren hedadura da. 4.32 irudian ertz komuna duten bi txatal agertzen dira, P eta R. Kokapen-jarraitasuna eman dadin:

$$P(1, v) = R(0, v) \quad \text{non} 0 \leq v \leq 1$$

Aurreko baldintzak, ertz komuna edukitzera behartzen ditu bi txatalak eta:

$$P_{30} = R_{00}$$
$$P_{31} = R_{10}$$
$$P_{32} = R_{20}$$
$$P_{33} = R_{30}$$



Irudia 4.32: P eta R txatalak.

P_{ij} eta R_{ij} Bezier txatalen kontrol-puntuak izanik.

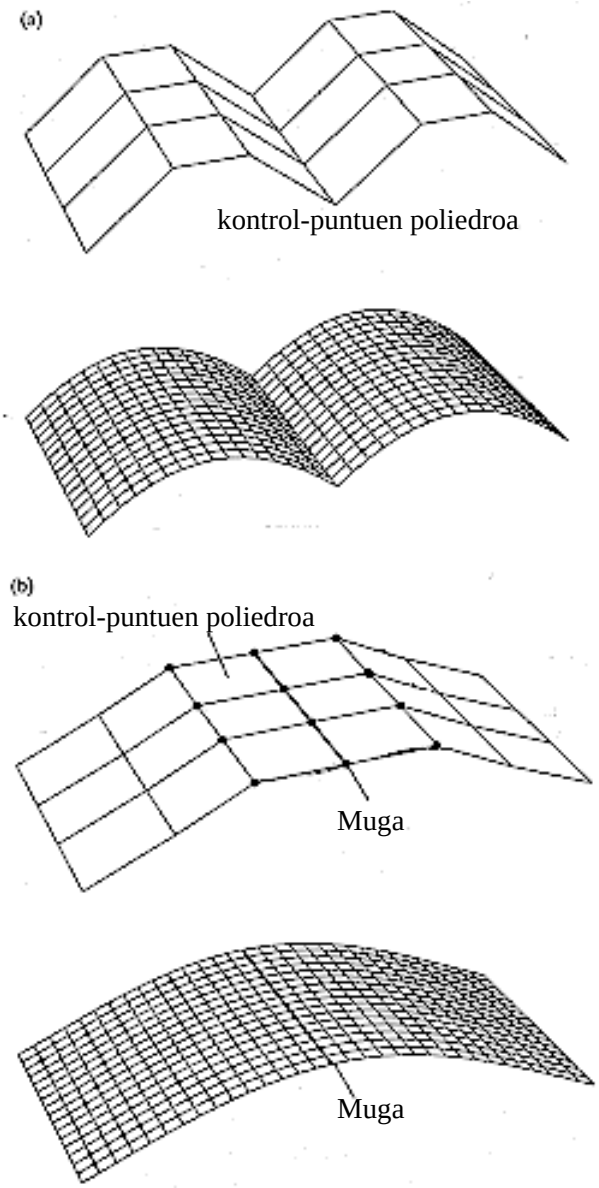
Lehen mailako jarraitasuna eman dadin, lehenengo txatalean $u = 1$ denean eta bigarrenean $u = 0$ denean bektore ukitzaileek baldintza bat bete behar dute edozein v-rentzat. Hau da:

$$(P_{3i} - P_{2i}) = k(R_{1i} - R_{0i}) \quad i = 0, \ldots, 3$$

Faux eta Pratt-ek (1979), CAD inguruneetan, gainazal bat Bezier txatalen elkarketa bidez eraikitzerakoan, baldintza horiek gogorregiak direla adierazi zuten. Adibidez, bi txatal elkartzerakoan, bigarren txatalaren 8 kontrol-puntu finkatuta egongo dira, eta elkartutako bi txatalei beste bat elkartuz, ukitzaileen jarraitasunaren ekuazioa bete dadin, balio zehatz bat esleituko zaie 12 kontrol-puntuei.

Bezierrek 1972. urtean, hain gogorrak ez diren baldintzak garatu zituen. Bere baldintzak erabiliz, elkarguneek kokapen-jarraitasuna izango lukete; ez, ordea, ukitzaileen jarraitasuna. Hala ere, elkarguneetako bektore ukitzaileek planokide izan behar dute. Nahiz eta baldintza horiek erabiliz malgutasun handiagoa lortu, oraindik ere arazoak agertzen dira txatalak elkartzerakoan. Soluzio bat, maila kubikoa baino altuagoko txatalak erabiltzea izango litzateke.

Orain arteko azterketa txatal errektangeluarretan oinarritu da, baina txatal-mota hori erabiliz, ezin dira forma guztiak adierazi. Esfera-itxurako objektu bat eraikitzea ezinezkoa izango litzateke. Esfera bat adierazteko, esferaren poloetako txatal errektangeluarrak triangelu bihurtu beharko lirateke. Horregatik, itxura konplexuetako objektuak eraikitzeko, egokiagoak



Irudia 4.33: (a)Bi Bezier txatalen arteko kokapen-jarraitasuna eta (b) ukitzaileen jarraitasuna.

dira txatal triangeluarrak.

Frain-en iritziz (1988), CAD sistemetan ematen den txatal errektangeluarren nagusitasunaren zergatia, txatalei hasieran eman zitzaien aplikazioan datza. Hasiera batean txatalak kotxeen diseinurako erabili ziren. Eta kotxeen kanpoaldeak berez itxura errektangeluarra dutenez, horiek eraikitzeko txatal egokienak errektangeluarrak zirela erabaki zen.

4.9.2 B-splineen txatalak

B-splineen gainazal bikubikoek duten ekuazioa ondokoa da:

$$P(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_i(u) B_j(v)$$

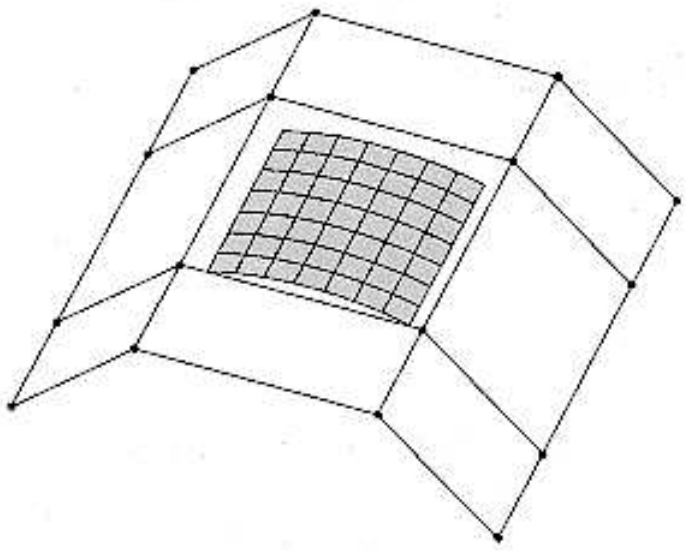
non P_{ij} kontrol-puntuen matrizea den.

B-splineen txatala B-splineen kurben hedadura da. B-spline kurba baten segmentu bat kalkulatzeko lau kontrol-puntu behar genituen. Ondorioz, txatal-zati bat kalkulatzeko 4×4 kontrol-puntutako sarea beharko dugu. Gogora ezazu, 4.16 eta 4.17 irudiak begiratuz, B-spline kurba baten segmentu bakar batek 8 korapiloko bektorea behar zuela. Beraz, txatal-zati bat lortzeko 8×8 korapilo-balioko matrizea beharko dugu.

4.34 irudian 16 kontrol-puntuz definitutako B-spline baten txatal-zati bat ikus daiteke. Txatala erdi aldeko 4 kontrol-puntuen inguruan kokatzen da. B-spline kurbek kontrol-puntuak interpolatzen ez dituzten modu berean, B-spline txatalak ez ditu kontrol-puntuak interpolatuko.

Erpin anitzak erabiliz, kontrol-puntuek osatzen duten poliedroaren ertzetan txatalaren itxura kontrola dezakegu. Begira ezazu 4.35 irudia; bertan albo bateko erpinen multzoa hirukoiztu dugu, 24 kontrol-puntutako matrizea osatuz. Erpin errepikatuen eraginez, hiru segmentuko txatal bat sortu dugu. Marrazkian ikusten denez, txatala erpin errepikatuen norabidean zabaldu dugu, erpin errepikatuek bere erakargarritasun-indarra hirukoiztu baitute. Hala ere, txatalak ez du erpinik interpolatzen.

4.36 irudiko adibidean bi albotako erpinen multzoak hirukoiztu ditugu, bederatzi segmentuko txatala lortuz, erpin lerrokideak interpolatu dituena. Era berean, barnealdeko kontrol-puntuak errepikatuz (bikoiztuz, hirukoiztuz, etab.) objektuak eraikitzerakoan oso ahaltsuak diren efektuak lor daitezke.



Irudia 4.34: B-spline txatal bat.

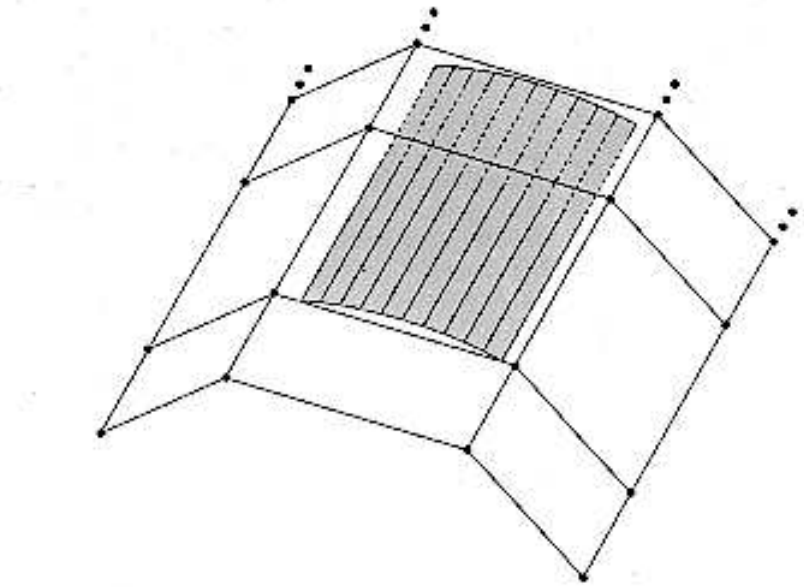
4.10 Gainazal parametrikoen irudigintza

Gaur egun, gainazal parametrikoak erabiliz irudi errealistak sortzeko bi hurbilketa erabiltzen dira:

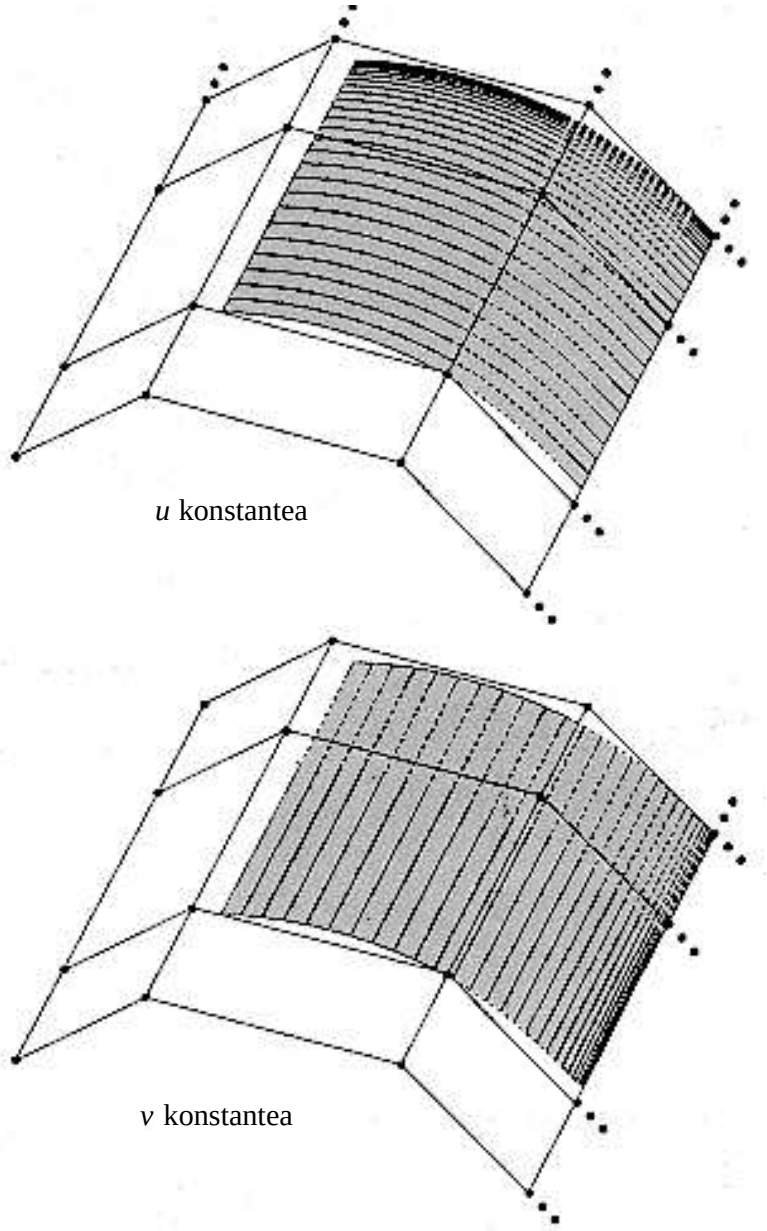
- Gainazalaren adierazpide parametrikoa erabiliz zuzenean irudia sortzea.
- Gainazal parametrikoak, lautzat har ditzakegun poligonoz osatutako sarera itzultzea, ondoren poligonoetan oinarritutako algoritmoren bat erabiltzeko.

Bigarren hurbilketa gehiago erabiltzen da. Batez ere, inplementatzeko errazagoa eta konputazio-kostuaren ikuspuntutik lehenengo hurbilketa baino merkeagoa delako.

Era parametrikoan definitutako gainazalek, poligonozko adierazpideak eskaintzen dizkigun zenbait datu ez dizkigute ematen. Objektu baten poligonozko adierazpideak gordetzen dituen zenbait datu oso lagungarri gertatzen dira objektuari dagokion irudi errealista sortzerakoan:



Irudia 4.35: Kontrol-puntuen lerro baten hirukoizketa.



Irudia 4.36: Kontrol-puntuen lerro eta zutabe bana hirukoiztuz lortutako bederatzi segmentuko B-spline txatal bat.

- Erpinen zerrenda erabiliz, oso erraza da Y maximo eta minimoa kalkulatzea.

- Gehikuntzazko ekuazioak erabiliz, poligonoen ertzak Y-ren funtzio modura lor ditzakegu.

- Gehikuntzazko ekuazioak erabiliz, puntu bakoitzaren Z sakonera X-ren funtzio modura kalkula daiteke.

Gainazal parametrikoek ez digute informazio hori ematen.

4.11 Poligonozko sareen bidezko gainazalen hurbilketa

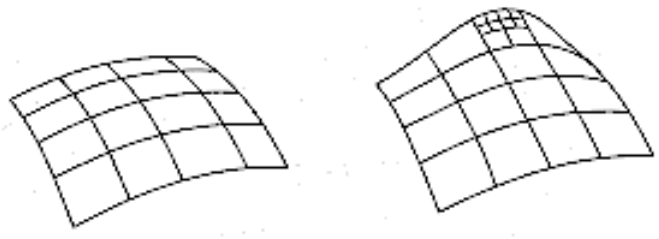
Txatal parametriko bat poligono bidezko adierazpidera itzultzea erraza da. Aukera bat, kurba isoparametrikoak erabiltzea izango litzateke. Horien bidez, gainazala deskribatzen duten puntuak kalkula daitezke, eta horiek lortu ondoren poligono bat deskriba dezakegu. Hurbilketa horren bidez gainazala deskribatzen duten n puntu lor daitezke. Baina itxura irregularra duten gainazalen deskribapena lortzeko, puntu-kopuru hori txikiegia bada, ez dugu hurbilpen zehatza lortuko; eta, alderantziz, n handiegia aukeratuz, konputazio-kostua gehiegizkoa izango da. Posibilitate bat, txatalaren proiekzioak irudian mugatzen duen azaleraren arabera, txatal bat itzultzeko erabiliko dugun n puntu-kopurua finkatzea izango litzateke.

Baina bereizmena komeni zaigun moduan aldatzea, egokiagoa izango da. Horrela ia lauak diren gainazalek ez dute ia zatiketarik jasango; gora-behera asko azaltzen dutenek, berriz, zatiketa asko pairatuko dituzte. Irizpide horri jarraituko liokeen algoritmo batek Bezier txatal batean izango lukeen eragina, 4.37 irudian ikus daiteke.

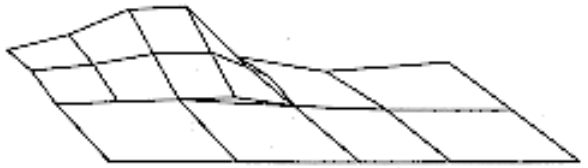
Txatalak nahi bezain lauak izan arte zatituko ditugu, ondoren poligonozko adierazpidera itzuliz. Txatalak zatitzeak zenbait abantaila du:

- azkarra da.

- txatal bakoitzean egingo dugun gehienezko zatiketa-kopurua finka daiteke. Ezaugarri hori elkarreraginezko sistemetarako garrantzi handikoa da.



Irudia 4.37: Bezier txatal baten zatiketa ez-uniforme eta uniformea.

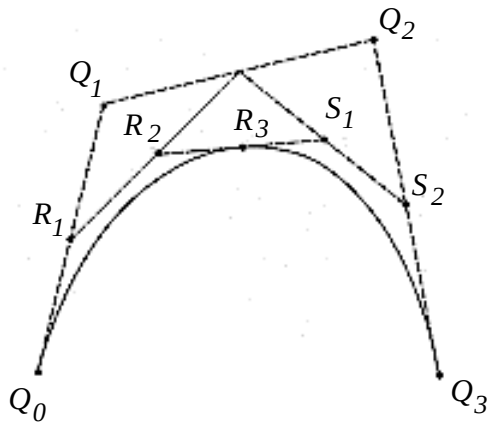


Irudia 4.38: Bezier txatal baten gain zatiketa ez-uniformea eraginez lortutako hutsune bat.

Zatiketa ez-uniformeak ere badu desabantaila bat. A eta B txatalak auzokideak izanik, A txatala azpitxataletan zatitzen badugu (B txatala dagoen bezala utziz), A eta B artean hutsune bat ager daiteke. Arazo horren adibide bat 4.38 irudian azaltzen da.

Zatiketa-algoritmoak azaltzeko, kurbetan oinarrituko gara, gero gainazaletara orokortzea erraza baita. Kurbaren hurbilketa lortzeko, kurbaren puntuak kalkulatu beharrean, gero eta zehatzagoa izango den hurbilketa era errekurtsiboan garatuko dugu. Dei errekurtsiboak, lortutako hurbilketa nahi bezain zehatza izatean bukatuko dira. Bezier kurba bat bi kurbatan zatituko dugu; horretarako, kontrol-puntuak zatitu beharko ditugu. Beraz, kontrol--puntuen bi multzo berri sortuko ditugu, R_i eta S_i. $R_3 = S_0$ lehenengo kurbaren azkeneko puntua eta bigarrenaren lehenengo puntua izango dira. Zatiketa hori behar bezala egiteko, ondoko ekuazioak erabil genitzake:

$$\begin{array}{ll} R_0 = Q_0 & S_0 = R_3 \\ R_1 = (Q_0 + Q_1)/2 & S_1 = (Q_1 + Q_2)/4 + S_2/2 \\ R_2 = R_1/2 + (Q_1 + Q_2)/4 & S_2 = (Q_2 + Q_3)/2 \\ R_3 = (R_2 + S_1)/2 & S_3 = Q_3 \end{array}$$



Irudia 4.39: Bezier kurba bat zatitzen.

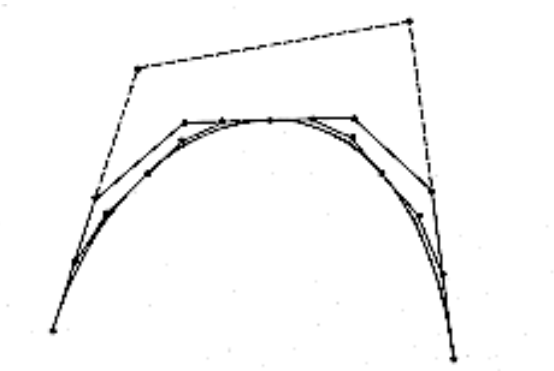
Zatiketa bakar baten ondorioz lortuko dugun hurbilketa, aurreko hurbilketa baino hobea izango da beti, 4.39 irudiko adibidean azaltzen den bezala. Hiru zatiketa egin ondoren lortuko genukeen hurbilpena, 4.40 irudian ageri da.

4.41 irudian, zatiketa-algoritmo bera txatalen gain eragiteko prozesua azaltzen da. Txatala, u konstantea duten lau kurbek eta v konstantea duten beste lau kurbek osatzen dute, zortzi kurba guztira. Kurba horien kontrol-puntuak, kontrol-puntuen matrizearen lerroak eta zutabeak dira. Kurbetarako zatiketaren algoritmoa u konstanteko lau kurben gain eraginez, hasierako txataletik bi azpitxatal lortuko ditugu. Eta prozesua v konstanteko kurbentzat errepikatuz, beste bi txatal lortuko ditugu. Beraz, hasierako txatala lau azpitxataletan zatitu dugu.

Zatiketaren sakonera, hau da, txatal bati eragiten zaion zatiketa-kopurua, modu errazean kontrola daiteke:

Kurba beti kontrol-puntuek osatzen duten poligono inguratzaile konbexoaren barnean kokatuko da. Zatiketa-prozesua, Bezier kurba baten erdiko kontrol-puntuetatik izkinetako kontrol-puntuak lotzen dituen lerro zuzenera dagoen distantzia kalkulatuz kontrola daiteke. Distantzia hori, zatiketa-prozesua exekutatuz goazen heinean txikiagotuz joango da, sortutako Bezier kurbak gero eta lauagoak izango baitira. Distantzia horren bidez Bezier kurba bakoitza zenbateraino laua den jakingo dugu.

Test hori, modu errazean egoki daiteke txatalen kasura. Lerrokideak ez diren hiru kontrol-puntu erabiliz, plano bat defini dezakegu, ondoren plano



Irudia 4.40: Kurba bat zatituz lortuko genituzkeen kontrol-puntuak.

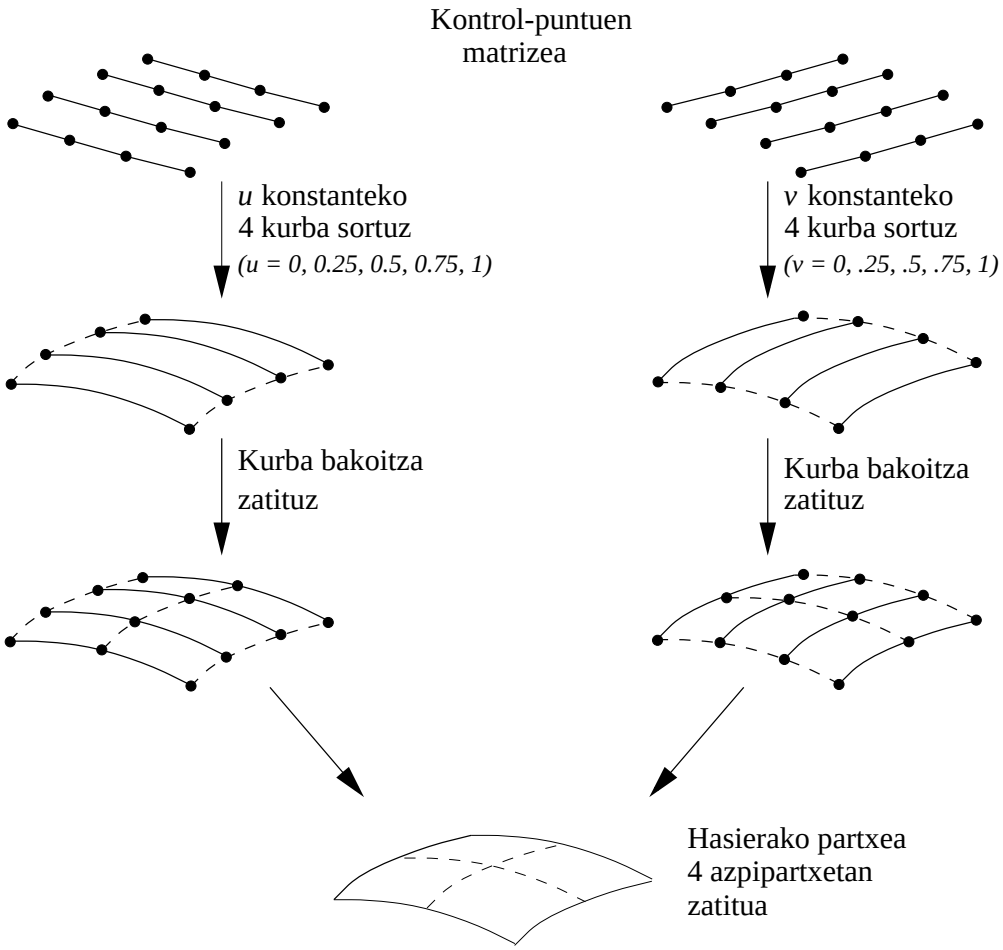
eta gainontzeko 13 kontrol-puntuen arteko distantzia kalkulatu, eta horietako bat perdoi-balioa baino handiagoa balitz, txatala azpitxataletan zatitu beharko genuke. Perdoi-balioak txatala laua noiz kontsidera daitekeen adierazten digu.

Txatalen zatiketa ez-uniformearen arazo bat, txatalen lautasun-testaren kostua da. Txatal bat zatitzen jarraitzea beharrezkoa den jakiteko bere lautasun-maila jakin beharra dugu, kostuaren aldetik arazo bihurtuz. Hala ere, txatalen zatiketa uniformea sinpleagoa eta hobea denentz, eztabaidagarria da.

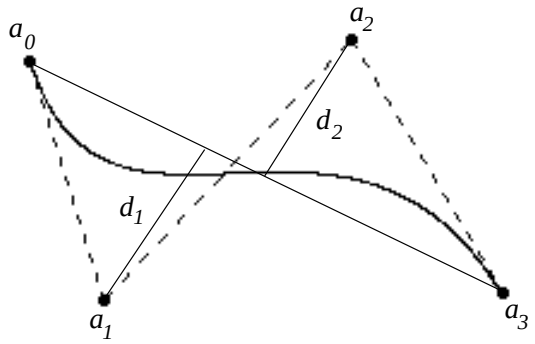
Txatalen zatiketa eragiten duen *spanning scanline* motako algoritmo bat, Lane-Carpenter-ena da:

1. Txatal guztiak ordenatu, txatal bakoitzaren Y balio maximoaren arabera. Balio hori estimatzeko modu bakarra, poligono konbexo inguratzailea osatzen duten kontrol-puntuen Y maximoa lortzea da.

2. Pixel-lerro bakoitzeko, txatal aktibo eta ez-aktiboen bi zerrenda eguneratu behar dira. Txatal aktibo bakoitzeko, zatiketa lautasun-maila batera iritsi arte egiten da. Zatiketaren ondorioz lortutako poligono "lauak", poligono aktiboen zerrendara gehitu behar dira, beste edozein poligono bezala tratatuak izan daitezen. Zatiketaren ondorioz sortutako txatalak, egoera ez-aktibora igarotzean, txatal ez-aktiboen zerrendara bidal daitezke. Beraz, poligono lau aktiboen zerrenda bat eta txatal aktibo eta ez-aktiboen beste bi zerrenda erabiliko ditugu.



Irudia 4.41: Txatalen zatiketa-prozesua.



Irudia 4.42: Erdiko kontrol-puntuetatik muturretako kontrol-puntuak lotzen dituen zuzenera dagoen distantzia ezagutuz, kurba zenbateraino laua den jakin dezakegu.

Txatalen zatiketa egin ondoren, *urratze-arazoa* ikusi ahal izango dugu. Arazo hori zatiketa ez-uniformearen ondorioa da. A eta B txatalak auzokideak izanik, A txatala azpitxataletan zatitzen badugu (B txatala dagoen bezala utziz), bi zatien artean hutsune bat sor daiteke. Arazo horren adibidea 4.38 irudian azaltzen da. Lautasun-irizpidea gogortuz, hutsuneak txikiagoak izango dira, baina konputazio-kostua handiagotu egingo da.

Zatiketa ez-uniformea eraginez, gora-behera handiko gainazal-zonak azpitxatal askotan zatituko ditugu. Alde lau zabalek, berriz, ez dute zatiketarik jasango. Lautasun-mailaren irizpidea gogortzean, askoz ere azpitxatal gehiago sortuko dugu eta, beraz, gainazalari dagokion irudia sortzeko, denbora luzeagoa beharko dugu.

4.12 NURBak eta β-splineak

Gainazalen eraikuntzarako erabilgarriak diren beste bi adierazpide azalduko ditugu orain, NURBak (B-spline razional ez-uniformeak) eta β-splineak. NURB adierazpideak kontrol-puntu bakoitzari datu bat gehitzen dio; datuak kontrol-puntuak kurbaren gain izango duen eragina adierazten du: pisua. β-spline adierazpidean, berriz, kontrol-puntuen egitura berdin uzten da; hau da, hiru dimentsioko koordenatu izaten jarraitzen dute, baina gainazal osoa-

ren gain eragina duten, globalak diren, bi aldagai erabiltzen dira. Aldagai berri horiei *alborapena* eta *tentsioa* deritze.

4.12.1 NURBak

Funtzio razionala bi funtzio polinomikoen arteko arrazoia da. Beraz, spline razionala bi splineren funtzioen arteko arrazoia da:

$$P(u) = \frac{\sum_{k=0}^{n} w_k a_k B_{k,d}(u)}{\sum_{k=0}^{n} w_k B_{k,d}(u)} \tag{4.2}$$

a_k k. kontrol-puntuaren kokapena da eta w_k k. kontrol-puntuaren pisua. Kontrol-puntu baten pisua zenbat eta handiagoa izan, kurba orduan eta kontrol-puntutik gertuago pasako da. Pisu guztien balioa 1 izatean, B-spline arrunta izango dugu, 4.2 ekuazioko izendatzailea (B-splineen batura) 1 izango baita.

Spline razionalek garrantzizko diren bi abantaila dituzte spline arruntekin konparatuz:

1. Elipse- eta zirkunferentzia-motako kurba konikoen adierazpen zehatza eskaintzen dute. Spline arruntek, berriz, polinomioez definitutakoek, kurba horien hurbilketa besterik ez dute lortzen.

2. Kontrol-puntuen gain perspektibaren aldaketa eragin ondoren, horiek adierazten duten gainazala ez da distortsionatuko. Spline arruntetan perspektibaren aldaketak distortsionatu egiten du gainazala.

B-spline arruntak inplementatzeko korapilo-bektore ez-uniformeen adierazpidea, NURBen (nonuniform rational B-spline) adierazpidea, erabili ohi da.

Spline arruntak adierazteko, koordenatu homogeneoen adierazpidea erabiltzen da, 4.2 ekuazioko izendatzailea koordenatu homogeneotan adierazitako puntu baten laugarren osagaia izanik.

$$\begin{pmatrix} x_h(u) \\ y_h(u) \\ z_h(u) \\ h \end{pmatrix}$$

B-spline razional bat kalkulatzeko, kontrol-puntuen multzo bat, erabiliko den polinomioaren maila, pisuen balioak eta korapilo-bektorea behar dira.

NURBen bidez kono-itxurako ebakidura bat marrazteko, spline funtzio koadratiko bat ($d = 3$) eta hiru kontrol-puntu behar dira. Spline funtzioak lortzeko ondoko korapilo-bektorea erabil dezakegu:

$$\left[\ 0, 0, 0, 1, 1, 1\ \right]$$

Ondoren, pisuen balioak finkatuko ditugu:

$$\begin{array}{l} w_0 = w_2 = 1 \\ w_1 = \frac{r}{1-r} \quad 0 \leq r < 1 \end{array}$$

Eta B-spline razionalaren adierazpidea:

$$P(u) = \frac{a_0 B_{0,3} + \left(\ r/(1-r)\ \right) a_1 B_{1,3} + a_2 B_{2,3}}{B_{0,3} + \left(\ r/(1-r)\ \right) B_{1,3} + B_{2,3}}$$

w_1 eta r parametroez baliaturik, ebakidura koniko desberdinak kalkula daitezke:

$r > 1$	$w_1 > 1$	(hiperbola-ebakidura)
$r = 1/2$	$w_1 = 1$	(parabola-ebakidura)
$r < 1/2$	$w_1 < 1$	(elipse-ebakidura)
$r = 0$	$w_1 = 0$	(lerro zuzena)

4.12.2 β-splineak

β-splineak (Barsky eta Beatty, 1983) B-spline kurbetako segmentuen formulazio bat dira. Adierazpide horretan B-splineei beste bi parametro gehitzen zaizkie: alborapena eta tentsioa. Bi parametro horiek, kurba osoaren gain dute eragina, eragina uniformeki edo ez uniformeki aplika dezakegularik. Alborapen- eta tentsio-balio batzuen kasuan, β-splineen formulazioa B-spline kubiko uniformeen berdina da.

β-spline bat $m - 2$ segmentuz definitzen da, i. segmentua:

$$S_i(u) = \sum_{r=-2}^{1} a_{i+r} b_r(\beta_1, \beta_2, u)$$

izanik, non

$$\begin{array}{l} a_i \ \text{ kontrol-puntua den} \\ 0 \leq x \leq 1 \\ i = 2, \ldots, m-1 \end{array}$$

$b_r(\beta_1,\beta_2,\mathrm{u})$ polinomio kubiko bat da, oinarriko r. β-spline funtzioa deiturikoa. Barsky-k, lau funtzioz osatutako oinarri berri bat garatu zuen. Oinarri hori erabiliz lortuko ditugun kurbak C^2 jarraitasunik ez dute azalduko, baina bai ordea G^2 jarraitasuna:

$$\begin{aligned}
b_{-2}(\beta_1,\beta_2,u) &= (2\beta_1^3/\delta)(1-u)^3 \\
b_{-1}(\beta_1,\beta_2,u) &= (1/\delta)[(2\beta_1^3 u(u^2-3u+3)+2\beta_1^2(u^3-3u^2+2) \\
&\quad +2\beta_1(u^3-3u+2)+\beta_2(2u^3-3u^2+1)] \\
b_0(\beta_1,\beta_2,u) &= (1/\delta)[2\beta_1^2u^2(-u+3)+2\beta_1 u(-u^2+3) \\
&\quad +\beta_2 u^2(-2u+3)+2(-u^3+1)] \\
b_1(\beta_1,\beta_2,u) &= (2u^3)/\delta
\end{aligned}$$

non

$$\delta = 2\beta_1^3+4\beta_1^3+4\beta_1+\beta_2+2.$$

β_1=1 denean, kurba ez-alboratua dela esaten da. β_1-en balioa handiagotuz eta txikiagotuz, kurba alde batera edo bestera albora dezakegu. β_2 (tentsioa) handiagotzean, kurba kontrol-puntuek osatzen duten poligonorantz gerturatuz joango da. β_1=1 eta $\beta_2 = 0$ direnean, β-splinea, B-spline kubiko uniforme bilakatzen da.

β-splineak B-spline orokortuak direnez, kontrol-puntuak edo korapiloak errepikatzean, B-splineek duten portaera berbera azaltzen dute.

4.13 Objektuak sare parametriko bikubikoen bidez eraikitzen

Txatal parametrikoen sareak objektuak eraikitzeko erabil daitezke, dagoeneko azaldu dugunez. Ondoren, txatalen erabilera sakonkiago aztertuko dugu.

Objektuak eraikitzerakoan, helburua objektu berri bat sortzea edo dagoeneko sortutako objektu bat aldatzea izan daiteke. Adierazpide hori erabiliz, objektu baten itxura era intuitiboan aldatzeko aukera izango dugu. Bestalde, txatal bakar bat aldatzea erraza den bitartean, arazo asko aurkituko dugu txatalen sare batekin lan egitean.

4.13.1 Gainazalen egokitzea

Teknika horren bidez, hiru dimentsiotan kokatutako puntu-multzo bat gainazal parametriko baten bidez interpolatzen saiatuko gara. Helburua lor-

tzeko, lehenik, kontrol-puntuetatik igarotzen den kurba-sare bat kalkulatuko dugu, puntu bakoitza u eta v konstanteko kurba batez interpolatuz. Kurba horiek, B-spline kurben interpolazioaren teknika erabiliz kalkulatutako B-splineak izango dira. Bigarren urratsean, spline horiek Bezier kurbetara itzuliko dira, spline bakoitza segmentu anitzeko Bezierren kurba modura adieraziko dugu. Ondoren, Bezierren kurba-segmentuak launaka hartuz, Bezierren txatal-saretzat har dezakegu guztia. Lau kurba-segmentuak Bezierren txatal baten ertzen mugak izango dira eta kurba-segmentu horiek erabiliz, Bezierren txatala definituko duten kontrol-puntuak kalkulatu ahal izango ditugu. Beraz, hiru dimentsioko puntu-multzotik abiatuta, Bezierren txatalez osatutako sare batera heldu gara, eta horretarako B-spline kurbetan oinarritu gara.

Azter dezagun itzulpen-prozesuaren lehenengo urratsa, hots, puntuak interpolatzen dituzten kurben kalkulua. Pauso horretan eman daitekeen arazo nagusia, puntuen topologia ez ezagutzean datza (kurben interpolazioan puntuak bata bestearen atzetik ematen zizkigutela bagenekien). Arazo horren soluzioa kontestuaren menpekoa izango da; soluzio posibleen hurbilketak Watt eta Wattek (1992) azaldu zituzten.

Suposa dezagun objektua deskribatzen duten puntuak eskuzko digitalizatzaile baten bidez lortu direla, eta kurbak erabiliz interpolatu behar diren puntuen sekuentzia ezaguna dela. Beraz, lehenengo urratsa egiteko, ez dugu arazorik aurkituko. Lan-kontestu hori askotan ematen da.

Jarrai dezagun bigarren urratsarekin. Kalkulatutako B-spline kurbak segmentu anitzeko Bezier kurba bihurtu behar ditugu. B-spline bakar bat kontsideratuz, Bezier kurba batera itzultzea sinplea da:

$$\begin{pmatrix} P_0 & P_1 & P_2 & P_3 \end{pmatrix} = B^{-1} B_s \begin{pmatrix} Q_0 & Q_1 & Q_2 & Q_3 \end{pmatrix} =$$

$$\frac{1}{6} \begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{pmatrix}$$

non

P Bezier kontrol-puntuak,
Q B-splineen kontrol-puntuak,
B Bezier matrizea eta
B_s B-spline matrizea baitira.

Oinarrizko matrizearen aldaketa **B-spline uniformeekin bakarrik** egin daiteke. Kurben sarea B-spline ez-uniformez osatuta badago, itzulpena egiteko, korapiloen txertaketa egin behar da. B-spline ez-uniformeen bigarren hurbilketa orokor hori, Watt eta Watten (1992) lanean azaltzen da.

Segmentu anitzeko B-spline kurben kasuan, formula hori behin eta berriz eragin beharko genuke kontrol-puntu egokien gain. Adibidez, kontsidera dezagun Q_0, Q_1, Q_2, Q_3 eta Q_4 kontrol-puntuek definitutako bi segmentutako B-spline kurba. Itzulpen-formula bi aldiz eragin beharko genuke, segmentu bakoitzeko behin, hau da, behin $\left\{ \begin{array}{cccc} Q_0 & Q_1 & Q_2 & Q_3 \end{array} \right\}$ kontrol-puntuen multzoak definitutako segmenturako eta beste behin $\left\{ \begin{array}{cccc} Q_1 & Q_2 & Q_3 & Q_4 \end{array} \right\}$ multzoko kontrol-puntuenerako.

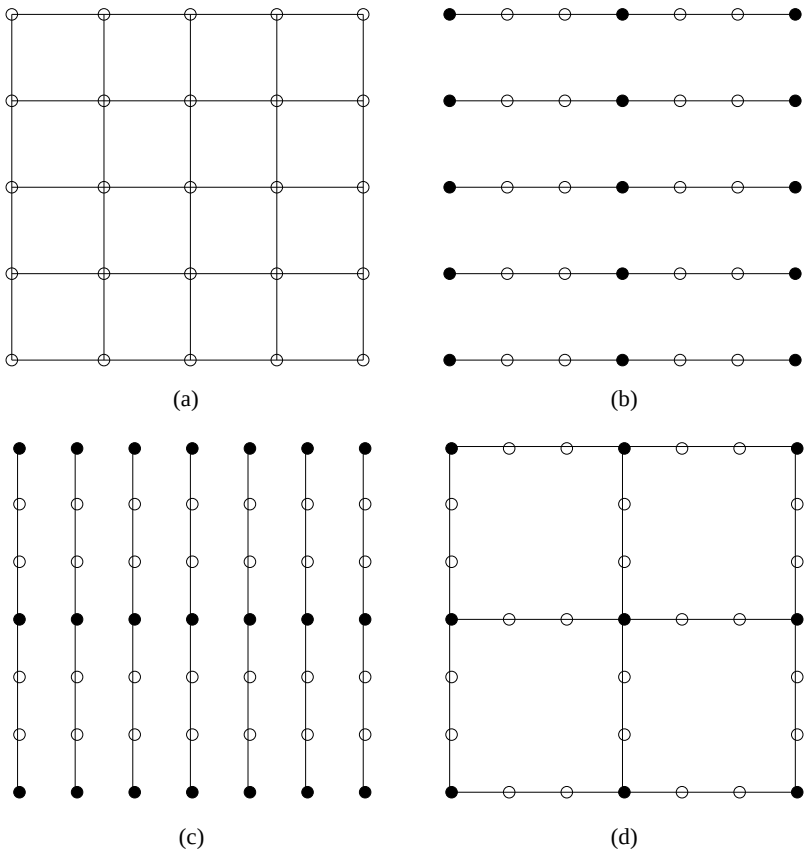
Orain, har dezagun 4.43 irudiko 5×5 kontrol-puntuko sarea. Lerro bakoitzak, bost kontrol-puntu dauzkanez, bi segmentuko splinea definitzen du, eta aipatu berri dugun prozesua lerroz lerro eraginez, bi segmentuko bost Bezier kurba lortuko ditugu, 4.43 irudiko b atalean ikus daitekeen bezala. Har ditzagun orain, bost Bezier kurba horien kontrol puntuak, eta zutabez zutabe mugi gaitezen. Zutabe bakoitzeko B-spline kurba baten bost kontrol-puntu ditugula pentsa dezakegu, eta bakoitza bi segmentuko Bezier kurbara itzuliko dugu, horrela, bi segmentuko Bezier kurben 7×7 neurriko sarea lortuz, 4.43 irudiko c atalean dakusagunez. Bezier txatalen mugak Bezierren kurbak direla badakigu; beraz, 4 txatalen mugei dagozkien Bezier kurbak ezagutzen ditugu, hau da, 4.43 irudiko d atalean agertzen diren kurbak izango dira txatalen mugak.

5×5 kontrol-puntuko sarearekin Bezierren lau txatal (2×2) lortu ditugu. Txatal bakoitzaren mugak ezagutzen ditugu, baina txatalen barnealdeko kontrol-puntuak ez; ondorioz, txatal bakoitzaren 16 kontrol-puntuetako 12 kalkulatu ditugu.

Beraz, nola kalkulatuko ditugu barnealdeko 4 kontrol-puntuak? 4.1 ekuazioa gogoratuz, kontrol-puntu horiek gainazalean duten eragina jakingo dugu:

$$P_{uv}(0,0) = 9(a_{00} - a_{01} - a_{10} + a_{11})$$

Formula horrek gainazal baten bihurdura deskribatzen du, eta $(0,0)$ puntuko normalaren baliokidea da. Normala 0 dela suposatzen badugu, a_{00}, a_{01} eta a_{10} kontrol-puntuak erabiliz eta ekuazioa askatuz, a_{11} lor dezakegu. P_{uv} 0 dela suposatzean, kalkulatuko dugun a_{11} puntua beste hiru kontrol-puntuek osatzen duten planoan egongo da.



Irudia 4.43: B-spline kurben sare bat Bezier txatalen adierazpenera itzultzen. (a) Bi segmentuko B-spline kurben 5×5 puntuko sarea. (b) Bi segmentuko 5 Bezier kurba. (c) Bost Kurben kontrol puntuekin zutabez zutabe lortutako kurbak, 7×7 kontrol-puntuko sarea. (d) Bezier txatalen mugak osatzen dituzten Bezier kurbak.

Horrela, barnealdeko puntuak kanpoaldekoak erabiliz kalkula daitezke. Suposizio hori egiteak, txatalaren mugetako kurbak, gutxi gora-behera, bata bestearen translazio izatea inplikatzen du. Praktikan, bi faktoreren menpe egongo da hori: objektuaren forma eta erabilitako kurba sarearen uv bereizmena. Gainazal-txatal baten bihurdura kalkulatzeko metodo gehiago ere badago, baina guk ez ditugu hemen azalduko.

4.13.2 Zeharkako ebakiduraren bidezko diseinua

Zeharkako ebakiduraren bidezko diseinua erabiliz, hodi-itxurako objektuak sor daitezke.

Mugimendu lineala

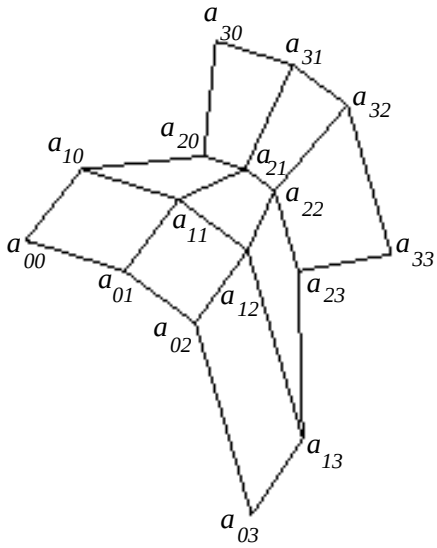
Zeharkako ebakidura segmentu anitzeko Bezier kurba baten bidez egon daiteke definituta. Ebakidura hori espazioan zehar mugituz goazen eran, bere neurria aldatzeko aukera izango dugu. Adibidez, 4 kontrol-puntuz osatutako kurba itxi bat izanik, $S(v)$ Bezier segmentu bat, espazioko hirugarren ardatzean zehar mugituko dugu, eta hori mugituz goazen heinean, 4 kontrol-puntuz osatutako beste Bezier kurba baten bidez, $r(v)$, neurriz aldatuko dugu. Emaitza gisa, gainazal bat definituko duten kontrol-puntuak lortuko ditugu:

$$P(u,v) = r(v)P(u,0)$$

Zeharkako ebakidura OZ ardatzean zehar 4 urratsetan mugituz, gainazala osatuko duten kontrol-puntuak kalkulatzeko:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \begin{pmatrix} r_0 & r_1 & r_2 & r_3 \end{pmatrix} + \begin{pmatrix} k \\ k \\ k \\ k \end{pmatrix} \begin{pmatrix} z_0 & z_1 & z_2 & z_3 \end{pmatrix}$$

Formula bera, poligonozko objektuak lortzeko erabil daiteke. Aurreko adibidean gainazaleko kontrol-puntu zirenak, poligonozko objektuko erpin izatera pasa dira.



Irudia 4.44: Kurba bat definitzen duten lau kontrol-puntu $(a_{00}, a_{01}, a_{02}, a_{03})$ linealki mugituz lor ditzakegun gainazalaren kontrol-puntuak. Kurbaren neurria, mugituz goazen heinean alda dezakegu.

4.13.3 Kontrol-poligonoaren bidezko diseinua

Oinarrizko teknika

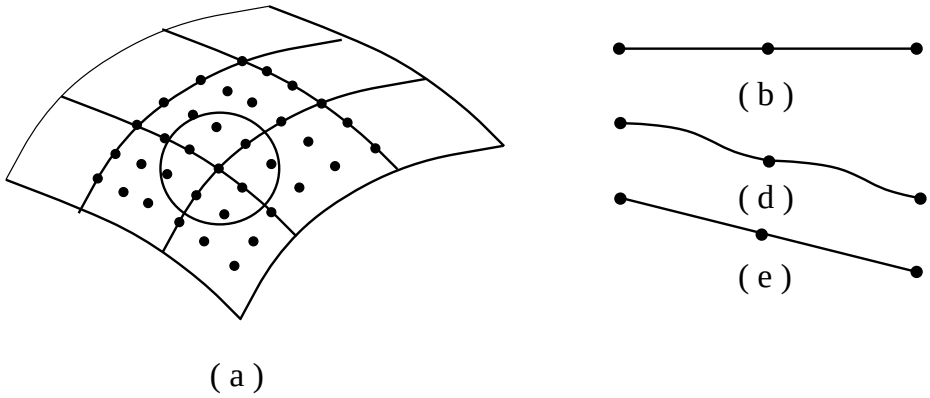
Hurbilketa hori liburu gehienetan azaltzen da. Teknika hori erabiltzerakoan, erabiltzaileari txatalen bidez definitutako oinarrizko objektu batekin elkarreragiteko aukera ematen zaio. Beraz, erabiltzailearekin elkarreragiteko definitu beharreko interfazea ahalik eta intuitiboena izatea komeni da. Oinarrizko objektua definitzen duten kontrol-puntuen kokapenen aldaketaren bidez, erabiltzaileak bere itxura alda dezake. Edozein kontrol-punturen kokapena aldatuz, kontrol-puntua mugituz, objektuak pairatutako aldaketak berehala ikusi beharko lirateke.

Hurbilketa hori erabiltzerakoan, arazo garrantzitsu batekin egingo dugu topo. Txatal-sare batekin lan egitean, ezin ditugu kontrol-puntuak edozein lekutara mugitu. Txatal bakoitzak, bere inguruko txatalekiko jarraitasun--baldintza batzuk bete behar ditu. Adibidez, oinarrizko objektua Bezier txatalen bidez definitzen badugu, kontrol-puntu bat mugitzeko eta jarraitasun--baldintzak mantentzeko aukera bat, kontrol-puntu mugituaren auzokide diren 8 kontrol-puntuak ere mugitzea izango litzateke, 4.45 irudian ikus daitekeen modura. Metodo hori jarraituz, jarraitasun-baldintzak beteko ditugu, baina gainazala deforma dezakegu. Kurben kasuan eragingo dugun deformazioa argi ikus daiteke: 4.45 irudiko b atalean, bi segmentuko Bezier kurba dakusagu. Kontrol-puntuak hirunaka mugitzen baditugu, hau da, kurbaren jarraitasun-baldintzak mantentzen baditugu, kurban deformazioak eragingo ditugu, 4.45 irudian azaltzen direnak bezalakoak eta, beraz, ezinezkoa izango da 4.45 irudiko d ataleko aldaketa lortzea.

Kontrol zehatza

Oinarrizko teknikaren beste arazoa, kontrol lokalean datza. Kontrol--puntu bat mugitzean aldaketak kontrol-puntua konpartitzen duten txatalen gain eman arren, objektu-espazioan ematen den deformazioaren garrantzia, txatalen azalerarekin erlazionatuta dago. Beraz, aldaketaren garrantzia edo bere eragina, aldaketa jasaten duten txatalak azpitxataletan zatituz kontrola dezakegu. Azaldutako ideia, **B-splineen deformazio hierarkikoa** (Forsey eta Bartels, 1988) izeneko teknikaren oinarria da.

Har dezagun B-spline baten definizioa:



Irudia 4.45: (a) Bezier txatalean kontrol-puntu bat mugitzeko, bere ingurukoak ere mugitu egin behar dira. (b) bi Bezier kurba elkarri lotuta. Bi kurben arteko kontrol-puntua mugituko bagenu, kontrol-puntu auzokideak ere mugitu egin beharko lirateke, eta emaitza (d) irudikoa izango litzateke; ezingo genuke (e) irudia lortu.

$$P(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} B_i(u) B_j(v)$$

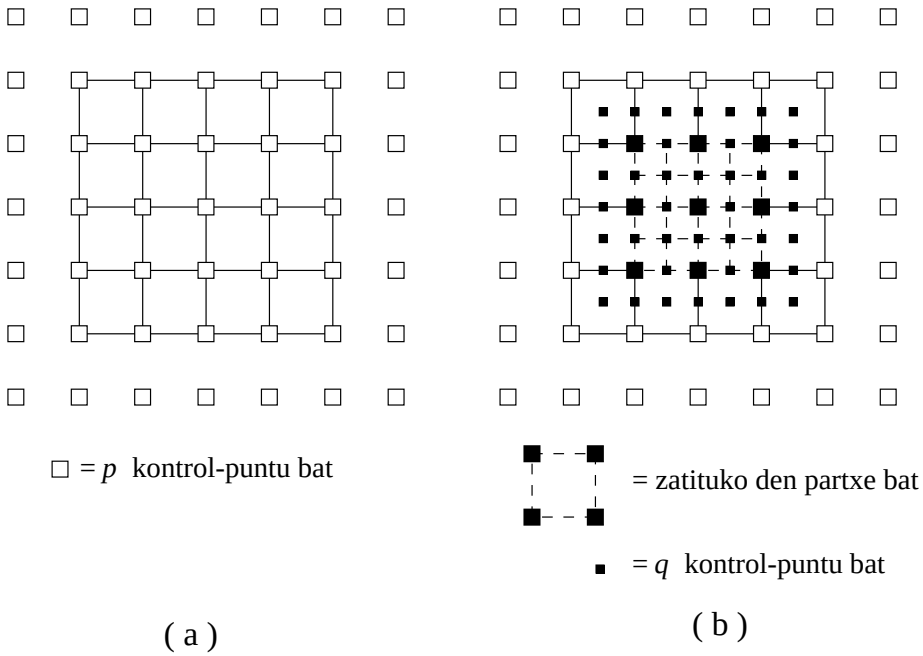
Hori bera, korapiloak txertatu eta gero, ondoko moduan berridatz dezakegu:

$$P(u,v) = \sum_{i=0}^{N} \sum_{j=0}^{M} R_{ij} B_i(u) B_j(v)$$

non $N > n$ eta $M > m$ diren.

Kontrol-puntu berriak Forsey eta Bartels-ek (1988) deskribatutako metodoaz lor daitezke. Baina nola eragingo dugu azaldutako teknika, interesatzen zaigun gainazaleko eskualdearen gain? Forsey eta Bartels-ek gainazal minimoaren kontzeptua definitu zuten. Gainazal minimoak bi baldintza betetzen ditu:

- Kontrol-puntu berrien mugimenduak eragiten dituzten deformazioak gainazal minimoaren gain ematen dira.



Irudia 4.46: (a) 16 txatal eta 49 kontrol-puntuz osaturiko gainazal minimoa. (b) Erdialdeko 4 txatal 16 txataletan zatituta.

- Gainazal minimoaren mugetako deribatuak ez dira aldatzen.

Beraz, gainazal berriaren (zatituaren) gain ematen diren deformazioek ez dute gainazal berriaren jatorria den gainazalean aldaketarik eragiten. Eta jarraitasun-baldintzak gainazal guztian zehar mantentzen dira. Zatitze-prozesua nahi bezain zehatza den kontrola lortu arte eragin daiteke.

Gainazal minimo bat 7×7 kontrol-punturen matrize batez definitutako 16 txatalek osatzen dute (4.46 (a) irudia). Irudiko erdialdean dauden lau txatal 16 txataletan zatitzeko kalkulatu behar diren kontrol-puntuak, 4.46 (b) irudian ikus daitezke. Txatal berriek hasierako 3×3 kontrol-puntu konpartitzen dituzte. Kontrol-puntuak ordenadorearen memorian gordetzeko datu-egitura egokia, zuhaitza da. Bertan, gainazalen zatiketak sortutako kontrol-puntu berriak isladatu beharko lirateke.

Gutxi gora-beherako kontrola

Aurreko kasuaren aurkakoa aztertuko dugu orain, alegia, gutxi gora-beherako kontrola. Gutxi gora-beherako kontrola erabiltzean, deformazio globala lortuko dugu. Gure eragiketek kontrol-puntu guztien gain izango dute eragina. Lehenik, kurben kasu partikularra aztertuko dugu.

Lau a_i kontrol-puntuz osatutako $P(t)$ kurba bat izanik, 4.47 irudikoa adibidez, lehenengo urratsa, kurba, unitate bateko karratu batean barneratzea izango litzateke. Karratua $R_{ij}(i = 0, \ldots, 3; j = 0, \ldots, 3)$ puntuen sare erregular batez osatuta dago. Karratua uv espazio modura hartzen badugu:

$$(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} R_{ij} B_i(u) B_j(v)$$

R_{ij} puntuen sarea $R'_{i,j}$ sarera aldatzen badugu, (u, v) puntua (u', v') puntua bihurtuko da:

$$(u', v') = \sum_{i=0}^{3} \sum_{j=0}^{3} R'_{ij} B_i(u) B_j(v)$$

Eta hasierako kurbaren kontrol-puntuek aldaketa berdina jasango dutenez, $P'(t)$ kurba berri bat izango dugu.

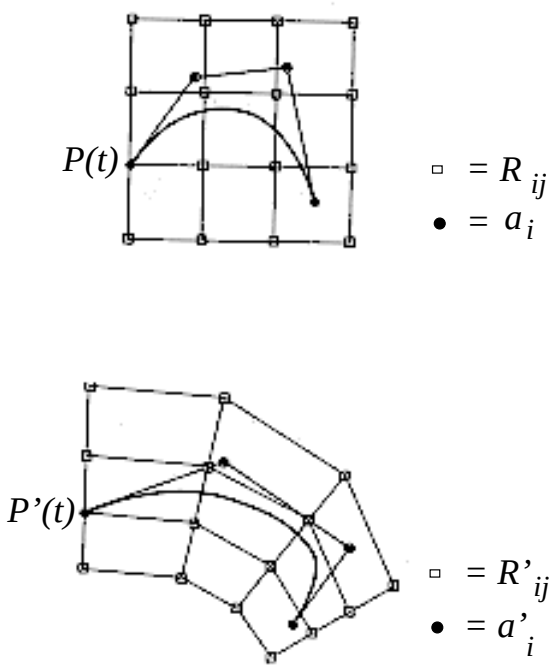
Metodo horren bidez kurba oso baten itxura alda dezakegu. Txatal lau batean barneratzen badugu kurba, txatalaren kontrol-puntuak mugituz txatala deformatzen dugu eta, era berean, kurbaren itxura ere aldatzen dugu.

Ideia bera txataletara hedatuz, txatalaren kontrol-puntuak, $a_{i,j}$, oinarriko hiru funtzioz osaturiko Bezier hipertxatal batean barneratu beharko ditugu; hori, hiru dimentsioko kontrol-puntuen sare batez definituta egongo da, eta kontrol-puntuen sareak kubo unitario bat osatuko du:

$$(u', v', w') = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} R'_{ijk} B_i(u) B_j(v) B_k(w) \tag{4.3}$$

4.3 ekuazioan kubo unitarioa distortsionatuz txatalaren kontrol-puntu berriak, a'_{ij}, kalkulatuko ditugu.

Teknika hori adierazpide parametrikoa erabiliz definitutako edozein objekturen gain aplika daiteke. Ondorioz, B-splineen txatalak ere Bezier hipertxataletan barnera ditzakegu. Poligonozko adierazpidearen bidez definitutako objektuak ere deformatzeko gai izango gara, objektua definitzen duten



Irudia 4.47: Kurba baten distortsio globala.

erpinak mugituz objektua deformatzea lortuko dugularik. Hipertxatalaren barneko edozein x punturi, hipertxatala definitzen duten kontrol-puntuak mugitu ondoren, x' puntu bat egokitu beharko genioke. 4.3 ekuazioa erabiltzeko, x puntuari dagokion (u, v, w) aurkitu, gero (u, v, w) puntuari dagokion (u', v', w') kalkulatu eta emaitza espazio cartesiarrera itzuli beharko genuke. Espazio-aldaketa horiek egiteko ekuazioak:

$$x = x_0 + u\mathbf{u} + v\mathbf{v} + w\mathbf{w}$$

eta

$$\begin{aligned} u &= \mathbf{u}(x - x_0) \\ v &= \mathbf{v}(x - x_0) \\ w &= \mathbf{w}(x - x_0) \end{aligned}$$

dira.

x_0 balioak (u, v, w) espazioaren jatorria definitzen du eta $\mathbf{u}$, $\mathbf{v}$ eta $\mathbf{w}$ aldagaiek espazioa definitzen dute. Balio horiek erabiltzaileak finkatu beharko ditu, berak kuboa distortsionatua izango den objektuarekiko kokatzen duelako.

Teknika hori, lehen aldiz Bezierrek garatu zuen, baina ordenadore bidezko irudigintzara zuzendutako lan gehienek Sederberg eta Parry-ren (1986) lana aipatzen dute, eta teknika horri itxura askeko deformazioa edo FFD (free form deformation) izena ematen diote.

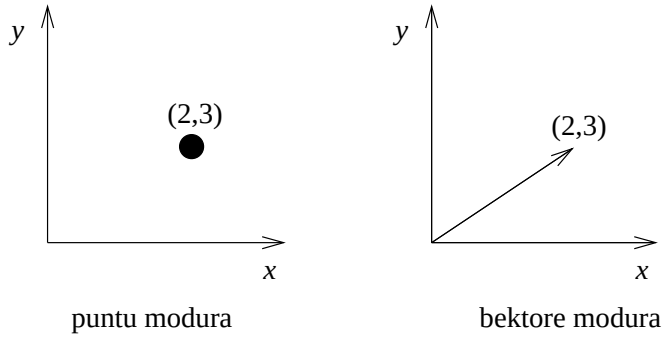
Kapitulua 5

Geometri aldaketak

5.1 Sarrera

Irudigintzan lan egitean eszenatokiaren ikuspegia lortzea dugu helburu. Horretarako, bertako osagaien ereduren bat eduki behar dugu, eta eredu edo modelo horiek nola edo hala sortu eta aldatu egin behar dira. Eszenatokiaren arabera eredu-mota ezberdinak erabil daitezke. Batzuk programa bidez sor daitezkeen itxura bitxiak izan daitezke; beste batzuetan Naturako fenomeno fisikoen itxura har dezaketen algoritmo errekurtsiboekin lan egin beharko dugu (fraktalak); esfera moduko unitateetan oinarritzen diren ereduak ere aurki ditzakegu, metaball delakoak; edo azalera erregular eta leunak lortu nahi dituztenak ere hor daude. Hala ere, gehienetan espazioko puntu eta marraz osatutako poliedro bidez lortutako hurbilketak erabiltzen dira. Horren arrazoia erraz uler daiteke, ordenadorez lagundutako diseinu lanetan sortzen diren objektu gehienak poliedrikoak baitira; eta poliedrikoak ez direnean, azalera erregularrak baitira, hau da, konikoak, esfera-itxurakoak edo spline edo Bezierren azalerak edo antzekoak, eta horiek ere espazioko puntu bitartez maneiatzen baitira. Ondorioz, objektuak aldatzerakoan, puntu horiek aldatu beharko ditugu.

Gai honetan erpin eta ertzez osatutako objektu poliedrikoak aldatzeko egin behar diren eragiketak azalduko ditugu. Hasteko, bi dimentsiotan jardungo dugu eta ondoren hiru dimentsiotara hedatuko ditugu emaitzak. Objektuak mugitzeko, neurriz aldatzeko eta biratzeko egin beharrekoak azalduko ditugu; era berean, objektu baten islada kalkulatzeko bidea ere adieraziko dugu; eta horiek guztiek matrize bidezko formulazioa onartzen dutela ikusiko



Irudia 5.1: Puntuen eta bektoreen arteko erlazioa.

dugu. Gainera, matrize bidez egin daitezkeen aldaketa guztien ezaugarriak ere azalduko ditugu.

5.2 Bi dimentsiotako aldaketak

Bi dimentsiotan lan egitean puntuen bi koordenatuak maneiatu beharko ditugu eta puntu hori bektore modura ere har dezakegu. Ikus 5.1 irudia.

Puntua edo bektorea $v = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ eran adieraziko dugu, eta puntu horri aldaketak eragiteko, gehienetan, matrizeak erabiliko ditugu, eta ondorioz, aldaketa $v^{'} = Av$ modura adieraziko dugu. Adierazpide horretan A matrizea 2×2 dimentsiotakoa izango da eta $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ itxura izango du.

Ondorioz, horrelako aldaketen emaitza modu honetakoa izango da:

$$x^{'} = ax_1 + by_1$$

$$y^{'} = cx_1 + dy_1$$

Eragiketa horren ezaugarri garrantzitsua linealtasuna da:

$$A(v_1 + v_2) = Av_1 + Av_2$$

$$A(rv) = r(Av)$$

non A aldaketa-matrizea den, v puntua eta r zenbaki erreala izanik.

Propietate horren ondorio modura, marra zuzen baten puntuen aldaketak emaitza gisa beste marra zuzen baten puntuak ematen dituela aipa dezakegu; ondorioz, poligono baten aldaketak beste poligono bat emango digu. Horrez gain, erreferentzi sistemaren jatorria era horretara ez da inoiz aldatzen, eta hori arazoa izan daiteke objektuak mugitzerakoan, era horretan ezingo baitugu aldaketa adierazi. Hala ere, arazo hori koordenatu homogeneoen sarrerarekin konpondu ahal izango dugu.

5.2.1 Leku-aldaketa

Objektuaren erpin guztiak leku batetik bestera aldatzeko, nahikoa dugu puntu bakar baten aldaketa jakitearekin; hau da, puntu guztiek aldaketa bera jasango dute eta guztiak norabide eta luzera berean mugitu behar dira. Mugimendu-bektorea $\begin{pmatrix} m \\ n \end{pmatrix}$ baldin bada, puntu bakoitzari egin behar zaion aldaketa honako hau izango da:

$$v' = v + \begin{pmatrix} m \\ n \end{pmatrix}$$

Leku-aldaketak aldatu egingo luke jatorrian legokeen puntua; beraz, ez du matrize bidezko adierazpidea onartzen, eta ezingo dugu $v' = Av$ eran adierazi. Hala ere, bi punturen arteko distantzia ez du aldatzen, ez eta objektu baten marren arteko angelurik ere; hau da, objektuaren itxura-aldaketarik egin gabe, kokagune batetik besterako aldaketa besterik ez da.

5.2.2 Neurri-aldaketa

Objektuen neurria aldatzeko, nahikoa dugu bere erpinen koordenatuak zenbaki batez biderkatzearekin; adibidez, bikoiztu egin nahi badugu, $x' = 2x$ eta $y' = 2y$ egitearekin nahikoa izango da. Aldaketa hori matrize bidez adieraz dezakegu:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Matrize horrekin neurri-aldaketa bi dimentsioetan proportzionala egiten da; alegia, zabalera hirukoiztu egin badu, luzera ere hirukoiztu egingo du. Hala ere, askotan proportzioa aldatu egin nahiko dugu, eta horretarako matrize

aldatzaileak $\begin{pmatrix} p & 0 \\ 0 & q \end{pmatrix}$ erakoa izan beharko du. Horrela, dimentsio bakoitza bere erara alda genezake, eta bata handitu eta bestea txikitzea ere zilegi izango genuke. Kontuan eduki behar da, $p, q \geq 1$ betetzen bada, neurria handitzen ari garela; baina era berean, $0 < p, q \leq 1$ izatean, neurria txikituko diogu objektuari. Dena den, beti, neurri-aldaketa nahi badugu behintzat, bai p eta bai q zenbaki positiboak izango dira; bestela, neurri-aldaketaz gain islada edo biraketa nahastuko dizkio aldaketari, hau da, aldaketa konposatua izango da.

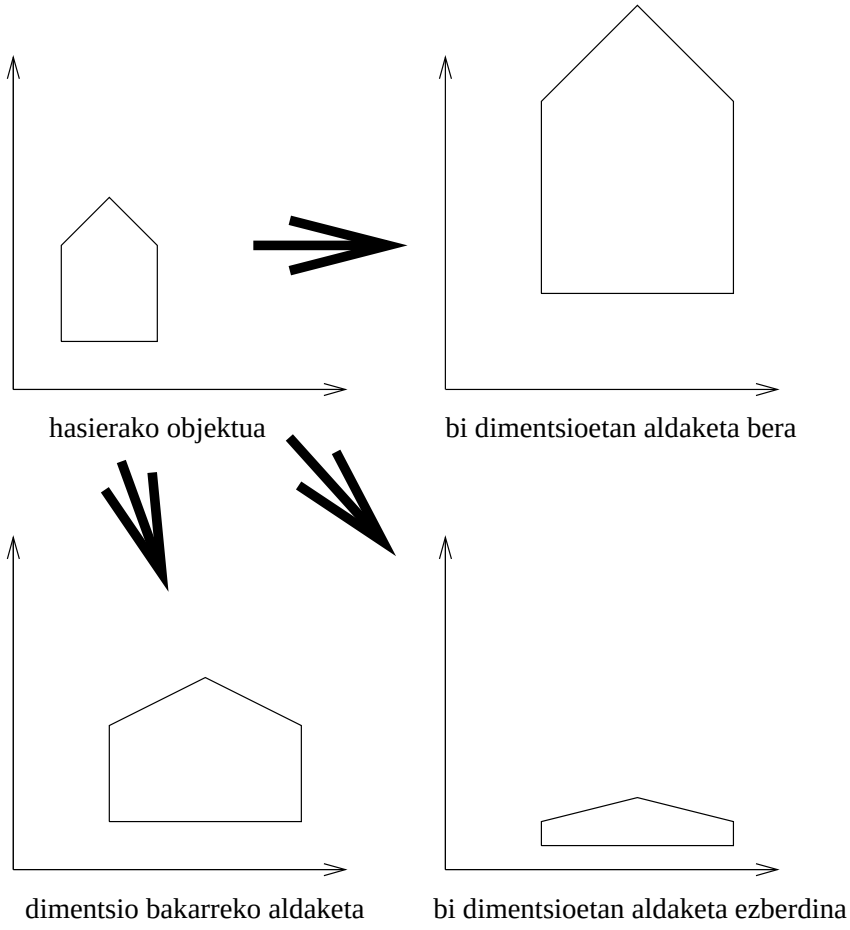
Neurri-aldaketak objektuaren kokagune-aldaketa sor lezake (ikus 5.2 irudia). Objektua erreferentzi sistemaren jatorritik kanpo baldin badago, neurri-aldaketaren ondorioz, handitze edo txikitze adina jatorritik urrundu edo hurbilduko da. Jatorriarekiko mugitze hori gertatzerik nahi izango ez bagenu, jatorriaren inguruan kokatu ondoren egin beharko genuke neurri-aldaketa, eta ondoren berriz lehengo lekura eraman beharko litzateke objektua. Hau da, toki berean mantendu nahi dugun puntua lehenengo jatorrira eraman behar da; gehienetan, objektuaren zentroa izango da puntu hori. Horren ondoren neurri-aldaketa egin behar da, eta azkenik jatorrian dagoen puntu hori lehengo kokagunera eramango duen leku-aldaketa eragin behar zaio. Guztira hiru aldaketa bata bestearen ondoren; 5.3 irudian ikus daitezke horiek.

5.2.3 Biraketa

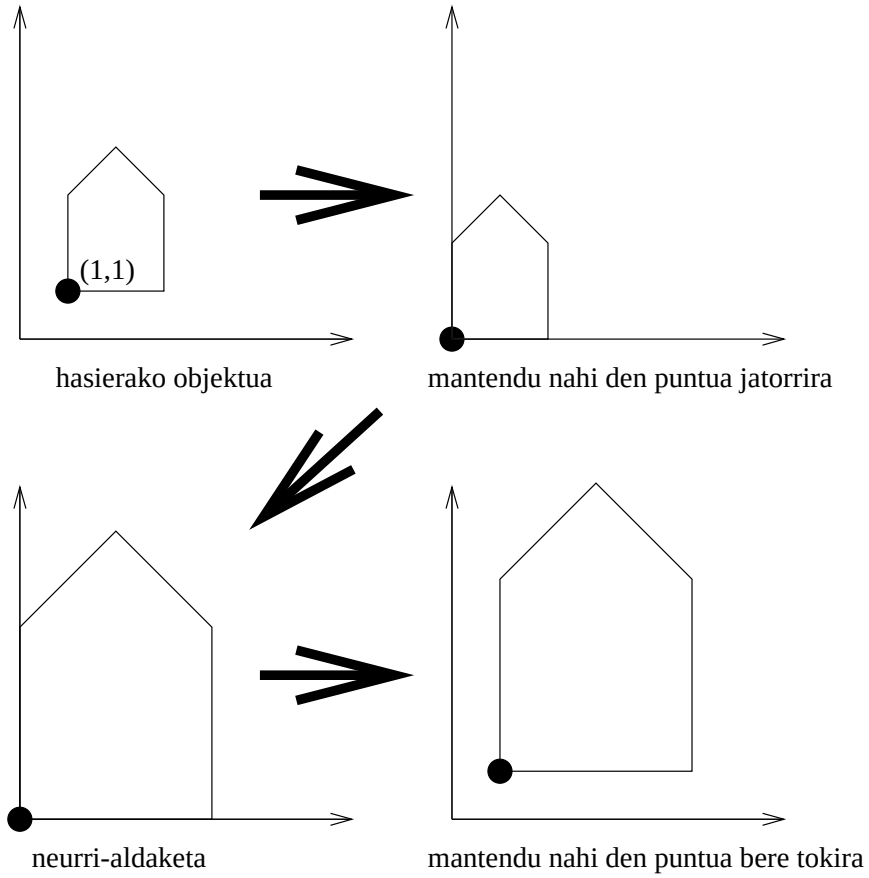
Bi dimentsiotan biraketa egitean , beti, puntu jakin baten inguruan egiten da eta guretzat puntu hori jatorria izango da; dena den, biraketa beste puntu baten inguruan egin nahi bada, neurri-aldaketarekin egin dugun bezala, puntu hori jatorrira eraman, biraketa egin eta berriz puntua hasierako kokagunera eramatearekin konponduko genuke arazoa.

Biraketa egiteko orduan, ohiko erreferentzi sistema erabili beharrean, koordenatu polarrak erabiliko ditugu, horiekin errazago ulertzen baita biraketa sortzen duen matrizearen jatorria. Oroimena laguntzeko, koordenatu polarrak puntu baten eta jatorriaren arteko distantziaz eta bi puntu horiek lotzen dituen eta ardatz horizontalaren arteko angeluaz osatzen direla esango dugu. Hori 5.5 irudian ikus daiteke.

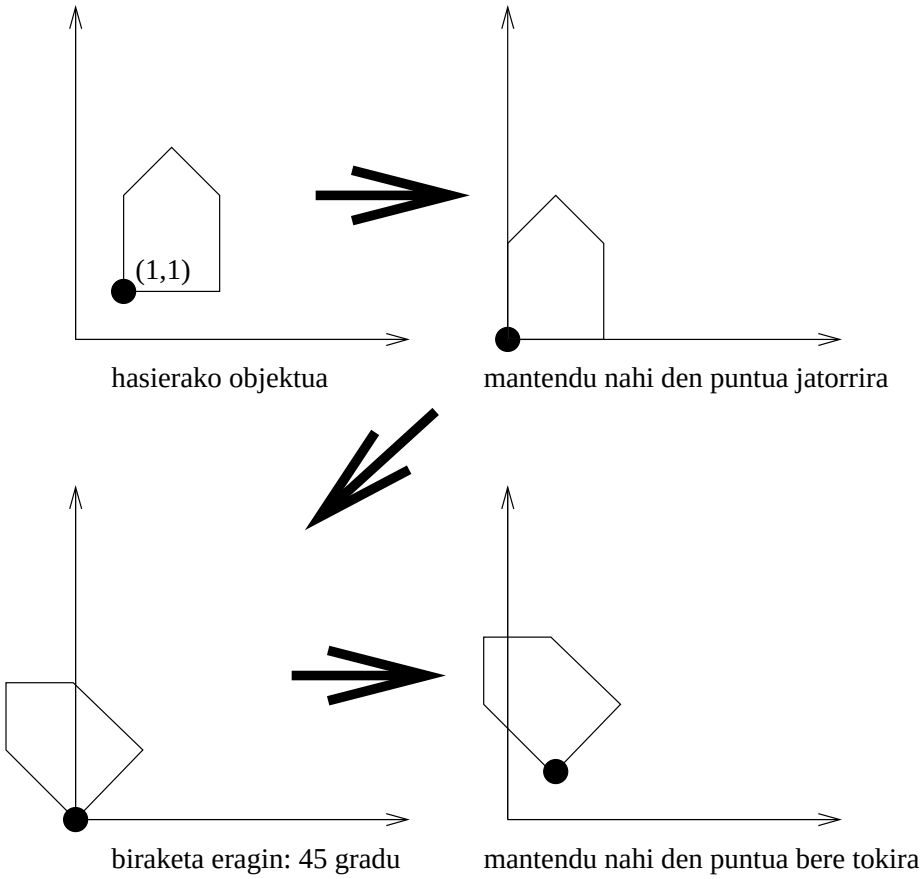
Koordenatu polarretatik ohikoetara pasatzeko, eta P puntuaren adierazpen polarra $P = \begin{pmatrix} L \\ \alpha \end{pmatrix}$ izanik, ohiko koordenatuetan $P = \begin{pmatrix} L\cos\alpha \\ L\sin\alpha \end{pmatrix}$ izango litzateke.



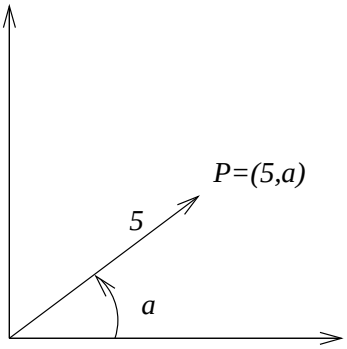
Irudia 5.2: Neurri-aldaketak eragindako leku-aldaketa.



Irudia 5.3: Puntu jakin bat lekuz alda ez dadin lotu beharreko aldaketak.



Irudia 5.4: Objektu-biraketa.



Irudia 5.5: koordenatu polarrak.

Puntu bat, bi dimentsiotan, jatorriarekiko β graduz biratu behar badugu, puntua $\begin{pmatrix} L\cos\alpha \\ L\sin\alpha \end{pmatrix}$ izatetik $\begin{pmatrix} L\cos(\alpha+\beta) \\ L\sin(\alpha+\beta) \end{pmatrix}$ izatera pasako litzateke. Ondorioz, aldaketa adierazteko $x' = L(\cos\alpha\cos\beta - \sin\alpha\sin\beta)$ [1] eta $y' = L(\cos\alpha\sin\beta + \sin\alpha\cos\beta)$ [2] erabil genitzake. Horrela, eta $x = L\cos\alpha$ eta $y = L\sin\alpha$ kontuan hartuta:

$$x' = x\cos\beta - y\sin\beta$$

$$y' = x\sin\beta + y\cos\beta$$

Eta matrize moduan adieraziz:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Matrize horren ezaugarri garrantzitsua ortonormaltasuna da. Bere lerro eta zutabeak ortogonalak dira eta gainera 1 luzeradunak. Matrize horietan iraulia eta alderantzizkoa matrize bera dira; beraz, $A^{-1} = A^i$. Horrez gain, matrize horiek bektoreen luzera errespetatzen dute; hori dela eta, biraketak puntuen arteko distantziak mantendu egiten dituela esan dezakegu.

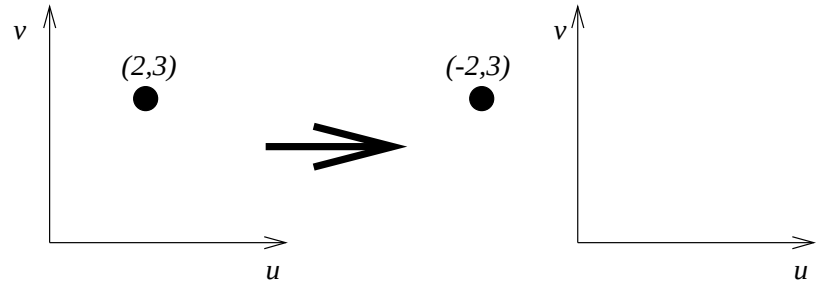
5.2.4 Islada

Marra zuzen batekiko simetrikoa den irudia lortzea da helburua. Izatez, simetria ardatza edozein izan liteke, baina guk jatorritik pasatzen dela suposatuko dugu; hala ez balitz ere, leku-aldaketak eginez, neurri- eta biratze-aldaketekin bezala, ardatza jatorritik pasaraz genezake, islada kalkulatu eta ondoren alderantzizko mugimendua egin.

Islada kalkulatzeko, zenbakizko algebra linealeko tresna teoriko batzuk erabil ditzakegu: Housenholder-en matrize isladatzaileak. Matrize horiek luzeratzat unitatea duten bektoreetan definitzen dira eta $A = I - 2uu^i$ erakoak dira. I delakoa identitate matrizea eta u delakoa luzeratzat unitatea duen bektorea izanik. Horrela definitutako matrizeek, bi propietate berezi dituzte:

1. $Au = -u$, hau da, u bektorea matrizeaz biderkatuz zeinu-aldaketa eragiten dio.

[1] $\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$
[2] $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$



Irudia 5.6: v ardatzarekiko puntu baten simetrikoa.

2. u bektorearen elkarzuta den edozein v bektoreri A matrizea biderkatuz gero, aldaketarik ez dio eragiten, alegia, $Av = v$

Bi propietate horiek direla eta, erreferentzi sistema u eta v bektoreek osatuko balute, edozein puntu A matrizeaz biderkatuko bagenu, puntuaren v bektorearekiko islada den puntua izango litzateke emaitza.

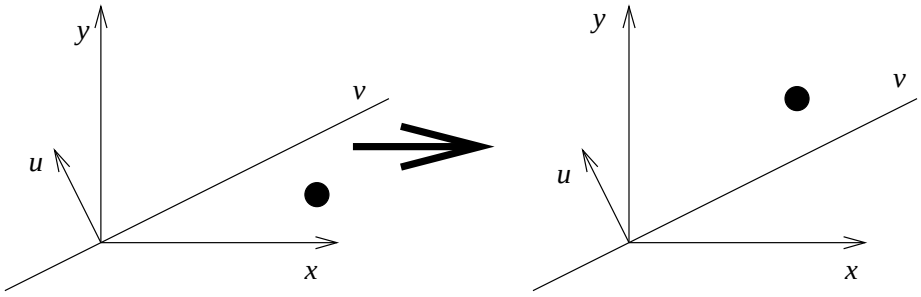
Horretan oinarrituz, edozein ardatzekiko islada kalkulatu behar duguneean, ardatz horrekiko elkarzuta den bektorea, u, lortu beharko genuke, eta, ondoren, matrize isladatzailea lortu ahal izango dugu zuzenean. Ardatz horrek x bektorearekiko α gradu baditu, bere koordenatuak $\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$ izango dira eta horrekiko elkarzuta den u bektorearenak, $\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}$. Ondorioz, guk behar dugun matrizea hauxe izango da:

$$A = I - 2uu^i = \begin{pmatrix} 1 - 2\sin^2\alpha & 2\sin\alpha\cos\alpha \\ 2\sin\alpha\cos\alpha & 1 - 2\cos^2\alpha \end{pmatrix}$$

Matrize horretan $\sin 2\alpha$ eta $\cos 2\alpha$ ordezkatuz, ondoko matrizera hel gaitezke:

$$A = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}$$

Argi dago, $\alpha = 0$ eta $\alpha = \frac{\pi}{2}$ kasuetan x eta y ardatzekiko islada kalkulatuko lukeen matrizea lortuko genukeela.



Irudia 5.7: Simetria ardatza edozein dela ere, kasu horretan v, 2D objektuen simetrikoa.

5.2.5 Aldaketen kateaketa

Aldaketa ezberdinak bata bestearen atzetik egitean, hasierako puntu edo objektuari aldaketa jakin zehatza eragingo diogu; baina aldaketa berdinak ordena ezberdinean eginez, emaitza ezberdina emango digute gehienetan.

Azken finean, aldaketa ezberdin horiek $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix}$ erako aldaketara muga ditzakegu, eta edozein matrize $A = UDV$ eran jar daiteke, non $D = \begin{pmatrix} p & 0 \\ 0 & q \end{pmatrix}$ eta U eta V matrize ortonormalak diren, hau da, edozein aldaketa-katea, aldaketa ortonormala, neurri-aldaketa, beste aldaketa ortonormala eta leku-aldaketen katea izatera mugatzen da. Aldaketa ortonormala, biraketa- edo islada-aldaketa izan daiteke.

Hori dela eta, aldaketa horiek guztiek marra zuzena beste marra zuzen batera aldatzen dutela ziurta dezakegu. Marra bat bertako bi puntuk defini dezakete; adibidez, P_0 eta P_1 puntuetatik pasatzen dela suposa dezakegu. Marra horretako puntu guztiak beste marra bateko puntuetara aldatzen dituela ikusteko, marrako edozein punturi eragiten dion aldaketa zein den ikusi behar da. Marrako puntu bat $P = tP_0 + (1-t)P_1$ eran adierazi ahal izango dugu, eta hori aldatzean P_0' eta P_1' puntuek definitutako marrako puntua lortu behar da. Hori egia dela ikusteko, nahikoa dugu matrizeen eta bektoreen arteko biderkaketaren propietateak erabiltzea:

$$P' = AP + T = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} =$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \left(t \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + (1-t) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \right) + \begin{pmatrix} m \\ n \end{pmatrix}$$

Baina biderkaketaren propietateen ondorioz:

$$P' = \begin{pmatrix} a & b \\ c & d \end{pmatrix} t \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} a & b \\ c & d \end{pmatrix} (1-t) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} =$$

$$t \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + (1-t) \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + (t+1-t) \begin{pmatrix} m \\ n \end{pmatrix}$$

Zatiak behar bezala bilduz gero:

$$P' = t \left(\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} \right) + (1-t) \left(\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix} \right)$$

Eta, jakina, horrela frogatutzat jo dezakegu:

$$P' = tP_0' + (1-t)P_1'$$

Hori frogatzeaz gain, moztu egiten diren marrak aldaketaren ondoren ere moztu egingo direla esan dezakegu, mozte-puntuak bi zuzenen ekuazioa betetzen baitu eta puntuaren aldaketak bi zuzen berrien ekuazioak ere beteko baititu; beraz, bi marrak moztu egingo dira. Hori beste era batera ere ikus daiteke. Bi marra paralelo aldatuz gero, beste bi marra paralelo bihurtzen direla ikus dezakegu; horretarako, marra bat aldatzean lortutako zuzen berriaren malda begiratu behar dugu. Hasierako zuzenaren malda $\frac{\Delta y}{\Delta x}$ izango da; aldaketa eragin ondoren, beste malda izango du: $\frac{\Delta y'}{\Delta x'}$. Malda berri hori zeren menpe dagoen ikusi behar da. Hasierako maldaren eta aldaketaren menpe besterik ez badago, bi zuzen paralelok malda bera dutenez, aldatu ondoren ere malda bera izango dute.

$$\frac{\Delta y'}{\Delta x'} = \frac{(cx_1 + dy_1 + n) - (cx_0 + dy_0 + n)}{(ax_1 + by_1 + m) - (ax_0 + by_0 + m)} = \frac{c(x_1 - x_0) + d(y_1 - y_0)}{a(x_1 - x_0) + b(y_1 - y_0)} =$$

Aldaketatxo bat eginez[3]

$$\frac{c(x_1 - x_0) + d\text{malda}(x_1 - x_0)}{a(x_1 - x_0) + b\text{malda}(x_1 - x_0)} = \frac{c + d\text{malda}}{a + b\text{malda}}$$

Ikus daitekeen moduan, aurreko maldaren eta aldaketa-matrizearen funtzioa da malda berria.

[3] zuzenaren malda=$\frac{\Delta y}{\Delta x}$; ondorioz $y_1 - y_0 = \text{malda}(x_1 - x_0)$

5.2.6 Koordenatu homogeneoak

Orain arte espazio afin batean gabiltza. Espazio afin horretan hainbat puntu eta bektore maneiatzen dugu, baina bataren eta bestearen arteko ezberdintasuna ez da argi geratzen, bai batzuk eta bai besteak bi koordenaturen bidez adierazten baititugu, eta, adibidez, puntu bat lekuz aldatzeko puntu horri bektore bat gehitu behar diogu, mugimendu-bektorea hain zuzen ere, eta hori adierazteko $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} m \\ n \end{pmatrix}$ eran adierazten dugu. Baina adierazpen horretan puntua zer den eta bektorea zer den ez da oso argi geratzen. Arazotxo horri aurre egiteko koordenatu homogeneoak erabil ditzakegu. Koordenatu homogeneoetan puntua adierazteko orduan, $\begin{pmatrix} hx \\ hy \\ h \end{pmatrix}$ erabiliko dugu; hor $h = 1$ izango da gehienetan, hau da, $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ izango da. Bektorea, berriz, $\begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$ izango da. Hori ulertzeko, zuzen bateko edozein punturen koordenatuak begiratzea komeni da.

$$\begin{pmatrix} h(x_0 + t(x_1 - x_0)) \\ h(y_0 + t(y_1 - y_0)) \\ h \end{pmatrix}$$

Hor h-ren balio gisa $\frac{1}{t}$ jartzen badugu, eta puntua infinituan badago, hau da, $t = \infty$, orduan puntu hori marra zuzenaren norabidea adierazten duen bektore modura har daiteke, eta bere koordenatu homogeneoek horixe adierazten dute:

$$\begin{pmatrix} \frac{1}{t}(x_0 + t(x_1 - x_0) \\ \frac{1}{t}(y_0 + t(y_1 - y_0) \\ \frac{1}{t} \end{pmatrix}_{t \to \infty} = \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \\ 0 \end{pmatrix}$$

Puntua eta bektorea bereizteaz gain, leku-aldaketa egitean gertatzen zen tratamendu-bereizketa gainditzen laguntzen digute koordenatu homogeneoek; hau da, lehen leku-aldaketa egiteko, puntuari bektorea gehitzen genion, $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix}$, biraketa, neurri-aldaketa eta beste aldaketa guztiak,

matrize bat biderkatuz lortzen genituen bitartean. Orain, ordea, matrize bat biderkatuz leku-aldaketa egin dezakegu:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & m \\ 0 & 1 & n \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

5.3 Hiru dimentsiotako aldaketak

Orain bi dimentsiotan egin zenaren hedapen modura azalduko ditugu hiru dimentsiotako aldaketak; hala ere, ezberdintasun batzuk egongo dira, nahiz eta gehienetan hedapen hutsa izan. Puntua $\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ izango da; bektorea bada, azken osagaia 0 izango du. Eta, bi dimentsiotan gertatzen zen antzera, puntuari matrizea biderkatzea eragiketa lineala izango da. Lehen bezala, aldaketa guztiak matrize bidezko biderkatze soil gisa adierazi ahal izango ditugu, baita leku-aldaketa ere, koordenatu homogeneoei esker, eta eragiketa horiek guztiak afinitatetzat har ditzakegu; hau da, lehen marra zuzena beste marra zuzen batean aldatzen bazen, orain planoa beste plano bihurtuko dugu. Gainera, plano paraleloak plano paralelo bihurtuko dira, eta moztu egiten ziren planoak, aldaketa eragin ondoren ere moztuko dira. Hori guztia bi dimentsiotan adierazitakoaren baliokidea da, eta modu berean ikus daiteke.

5.3.1 Leku-aldaketa

Zuzenean eta esplikazio beharrik gabe, puntuari leku-aldaketaren bektorea gehitzea dela esan daiteke, ondorioz:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & m \\ 0 & 1 & 0 & n \\ 0 & 0 & 1 & o \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

5.3.2 Neurri-aldaketa

Aldaketa hori ere bi dimentsiotan egindakoaren antzekoa da. Aldatzen den bakarra, dimentsio-kopurua da, eta ondorioz, hiru ardatzetan eragin deza-

kegu aldaketa. Bakoitzean ezberdina izan daiteke aldaketa, bi dimentsiotan egiten genuen bezala, eta aldaketaren ekuazioa ondokoa izango da:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} p & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Jakina da, horrelako aldaketek erreferentzi sistemaren jatorria mantendu egiten dutela; baina beste puntu guztiak aldatu egin ditzake, eta hori dela eta, objektuen neurria aldatzean, objektua mugitu egin daiteke. Askotan hori ez dugu nahiko, eta hori gerta ez dadin, bere lekuan mantendu nahi den puntu bat adierazi behar da, gehienetan zentroidea,[4] eta puntu hori mugitu ez dadin, jatorrira eraman behar da, edo beste era batera, jatorria puntu horretan jarri behar da; ondoren, neurri-aldaketa egin, eta alderantzizko leku-aldaketa egitearekin bukatuko litzateke. Hau da, jatorria lehengo tokira eraman behar da, objektuari alderantzizko leku-aldaketa eraginez.

5.3.3 Islada

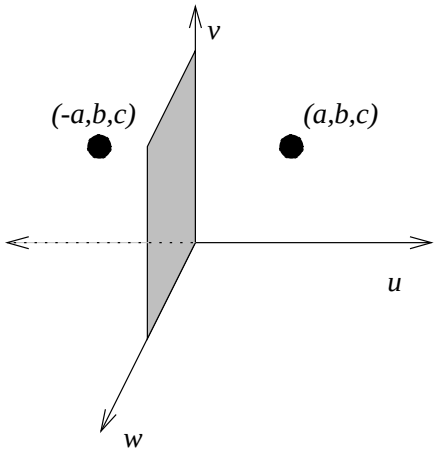
Objektuen islada kalkulatzeko orduan, plano jakin batekiko islada kalkulatu behar da; bi dimentsiotan simetria ardatz batekiko kalkulatzen genuen era berean, orain planoekiko islada lortu beharko dugu. Hiru dimentsiotan ere matrize-mota bera erabil dezakegu, hau da, Housenholder-en matrize isladatzaileak, matrize horiek dimentsio guztietarako balio baitute; beraz, orain ere:

$$A = I - 2uu^i$$

I delakoa identitate matrizea, u delakoa bat moduludun bektorea eta u^i delakoa u bektorearen iraulia izanik. Horrela lortutako matrize horrek, aurreko bi propietateak betetzen ditu:

1. $Au = -u$
2. $v \perp u$ izanik, $Av = v$

[4] objektuaren zentroidea kalkulatzeko, bere erpinen koordenatuen batuketa egiten da, eta emaitza erpin kopuruaz zatitu, hau da, $x_c = \frac{\sum_{i=1}^{n} x_i}{n}$, $y_c = \frac{\sum_{i=1}^{n} y_i}{n}$, eta azkenik $z_c = \frac{\sum_{i=1}^{n} z_i}{n}$

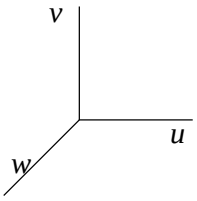


Irudia 5.8: 3D objektuen simetrikoa.

Hori jakinda, edozein planorekiko islada kalkulatzea erraza da: u bektore egokia aukeratzea besterik ez zaigu falta. Aukeratzeko, betebeharra zein den ongi aztertzea komeni da. Suposa dezagun islada erreferentzi sistemako bi bektorek definitzen duten plano batekiko kalkulatu behar dela, adibidez ZY planoarekiko islada. Kasu horretan, $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ puntuaren islada $\begin{pmatrix} -x \\ y \\ z \end{pmatrix}$ puntua izango da; hau da, x bektoreak u bektorearen lana egiten du, eta bektore hori simetri planoaren elkarzuta den bektorea da. Hori orokortuz, plano batekiko islada kalkulatuko duen matrizea lortzeko behar den u bektorea, plano horrekiko elkarzuta den bektorea izango da.

Interesatzen zaigun bektorea $u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$ balitz, islada kalkulatuko lukeen matrizea ondokoa izango litzateke:

$$A = \begin{pmatrix} 1-2u_x^2 & -2u_xu_y & -2u_xu_z \\ -2u_xu_y & 1-2u_y^2 & -2u_yu_z \\ -2u_xu_z & -2u_yu_z & 1-2u_z^2 \end{pmatrix}$$



Irudia 5.9: hiru dimentsiotako erreferentzi sistema.

5.3.4 biraketa

Hiru dimentsiotan biraketa eragiteko, lehenengo eta behin zeren inguruan biratu behar den jakin behar da. Bi dimentsiotan puntu baten inguruan biratzen genuen. Hirutan ordea ardatz baten inguruan biratu beharko da. Ardatz hori edozein izan daiteke, eta edonon egon daiteke; baina, guretzat, jatorritik igarotzen den ardatza izango da. Jatorritik igaroko ez balitz, leku-aldaketa erabiliaz bertatik igaroarazi, biraketa eragin eta berriz lehengo lekura eramatearekin helburua lor dezakegu-eta.

Biraketa erreferentzi sistemaren ardatz baten inguruan egin behar badugu, bi dimentsiotan ikusitakoaren orokortzea dela esan dezakegu, kasu horretan ardatz horri dagokion koordenatua ez baita aldatzen, eta aldatzen direnak beste biak baitira, bi dimentsiotan gertatzen zen bezalaxe. Bi dimentsiotako aldaketa ondokoa zen:

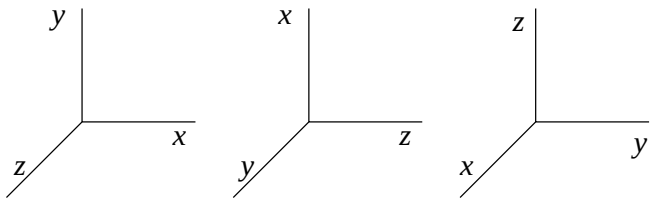
$$\begin{pmatrix} x^{'} \\ y^{'} \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Edo beste era batera ikusita:

$$x^{'} = x\cos\alpha - y\sin\alpha$$

$$y^{'} = x\sin\alpha + y\cos\alpha$$

Baina orain hiru dimentsio ditugu. Bertako erreferentzi sistema hiru ardatzek osatzen dute; 5.9 irudian agertzen diren u, v eta w ardatzak har genitzake adibidez. Garrantzitsua da hiru ardatzen arteko erlazioa agertzea; hau da, erreferentzi sistemaren ezaugarriak argi edukitzea komeni da. Eta gure kasuan, u eta v bektoreen arteko bektore-biderkaketaren emaitza hirugarren bektorea, w, dela argi eduki behar dugu. Berezitasun hori beti beteko du gure sistemak; ondorioz, gure eszenatokia marrazteko erabiliko dugun ikuspuntua edozein izanda ere, beti ezaugarri hori bete beharko du.



Irudia 5.10: erreferentzi sistema beraren hiru ikuspegiak.

Sistema horretan ardatz batekiko, w-rekiko adibidez, biraketa egin behar badugu, biraketaren ekuazioak zehazterakoan, ardatz batekiko aldaketarik ez dagoela eta beste bi ardatzekiko aldaketa bi dimentsiotan gertatzen zen bezalakoa izango dela izan behar ditugu kontuan. Horren arabera, aldaketaren ekuazioak ondokoak izango dira:

$$u^{'} = u\cos\alpha - v\sin\alpha$$

$$v^{'} = u\sin\alpha + v\cos\alpha$$

$$w^{'} = w$$

Hala ere, gure erreferentzi sistemaren ardatzek izena dute, x, y eta z, eta hiru horiekin gure sistema hiru ikuspuntu ezberdinetatik begiratuta, 5.10 irudian agertzen diren hiru bistak lor ditzakegu. Kontuan eduki behar dugu, hirurak sistema bera adierazten dutela, eta hiruretan x eta y ardatzen arteko bektore-biderkaketak z ardatza duela emaitzatzat. Ardatz bakoitzarekiko biraketa egitean u, v eta w gure x, y, eta z ardatzekin parekatu behar dira, eta horren arabera, aldaketa ematen duen matrize egokia lortu.

z ardatzarekiko biraketa

Bi dimentsiotako biraketan oinarrituta, z ardatzarekiko biraketaren ekuazioak, honako hauek izango dira:

$$x^{'} = x\cos\alpha - y\sin\alpha$$

$$y^{'} = x\sin\alpha + y\cos\alpha$$

$$z^{'} = z$$

Hori matrize bidez adierazi nahi izanez gero, honela adieraz genezake:

$$\begin{pmatrix} x^{'} \\ y^{'} \\ z^{'} \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5.1}$$

x ardatzarekiko biraketa

Arrazoi berean oinarrituz, x ardatzarekiko biraketa honako ekuazio hauek lortzen dute:

$$y^{'} = y\cos\alpha - z\sin\alpha$$

$$z^{'} = y\sin\alpha + z\cos\alpha$$

$$x^{'} = x$$

Orain kontuan eduki behar da, bi dimentsiotatik hirura zabaltzeko erabili ditugun u, v eta w ardatzak zein diren. Marrazkian ikusten denez, u ardatza kasu horretan y ardatza izango litzateke, v ardatzaren tokian z jarri dugu eta w-ren lanean x; alegia, x ardatzari dagokion koordenatua ez dugu aldatzerik nahi. Hori matrize bidez adierazteko:

$$\begin{pmatrix} x^{'} \\ y^{'} \\ z^{'} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5.2}$$

y ardatzarekiko biraketa

Azkenik, y ardatzarekiko biraketa daukagu. Horretarako, aurreko bi kasuetan bezalaxe, irudian ardatz bakoitzaren betebeharra zein den zehaztea besterik ez zaigu falta, eta kasu horretan ondokoa izango da:

$$z^{'} = z\cos\alpha - x\sin\alpha$$

$$x^{'} = z\sin\alpha + x\cos\alpha$$

$$y^{'} = y$$

Ekuazioa horiek matrize bidezko adierazpenera errazago pasa ahal izateko, pittin bat ordenatzea komeni da, nahasi samar daude-eta:

$$x^{'} = x\cos\alpha + z\sin\alpha$$

$$y^{'} = y$$
$$z^{'} = -x\sin\alpha + z\cos\alpha$$

Ikus daitekeenez ekuazioak, ez dira aldatu, baina matrize erara pasatzeko itxura aproposagoa daukate, hori honelakoa izanik:

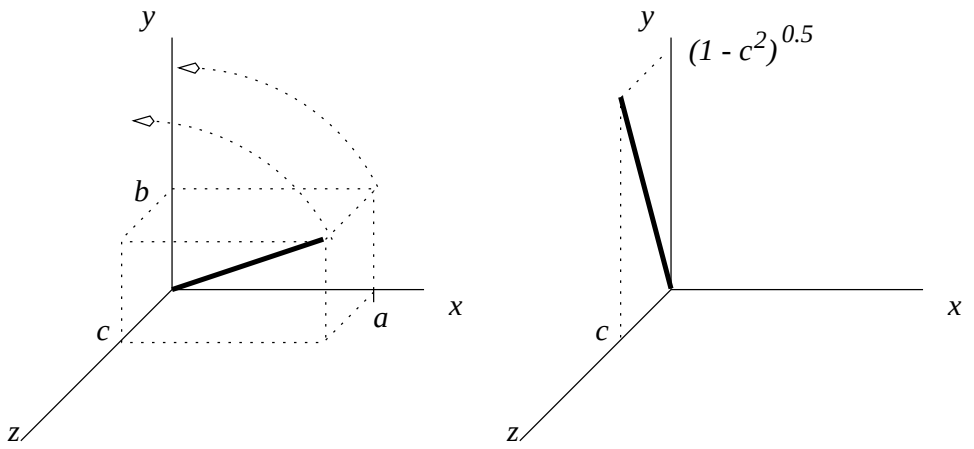
$$\begin{pmatrix} x^{'} \\ y^{'} \\ z^{'} \end{pmatrix} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Edozein ardatzekiko biraketa

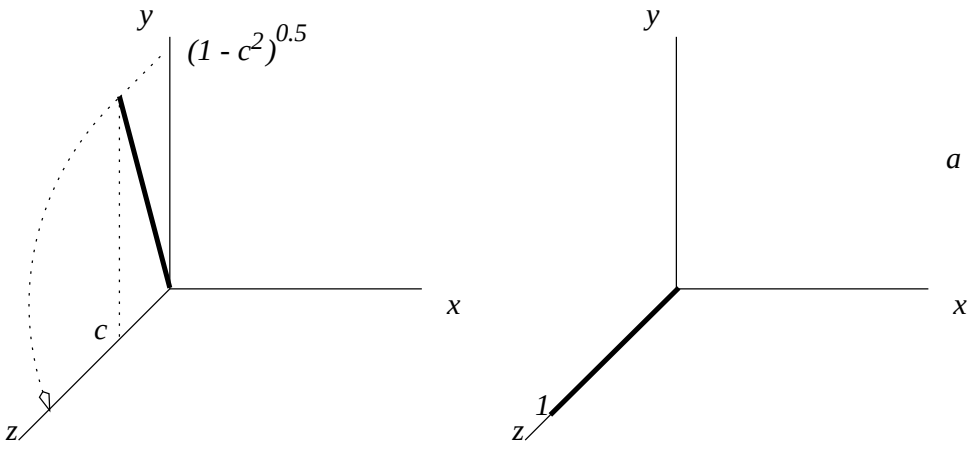
Objekturen bat edozein ardatzen inguruan birarazi nahi badugu, orain arte ikusi ditugun biraketetan oinarritu gaitezke. Horretarako, biraketa-ardatza erreferentzi sistemaren ardatz bat izatea lortu behar da, eta hori lortuz gero, aurretik agertu dugun biraketaren bat egitera mugatzen da gure biraketa. Ondorioz, hiru aldaketa ezberdin egin beharko ditugu: lehenengoak biraketa ardatza erreferentzi sistemaren ardatz bihurtuko du, bigarrenak ardatzarekiko bira eragingo du, eta hirugarrenak lehenengo aldaketaren alderantzizkoa eragin beharko du, hau da, biraketa ardatza hasieran zegoen egoeran utzi behar du. Oraingo arazoa biraketa-ardatza erreferentzi sistemako ardatz bihurtzea da, baina hori erreferentzi sistemaren aldaketa soil bat besterik ez da, eta bi biraketa modura uler daiteke:

1. Lehenengo biraketa 5.11 irudian ikus daiteke, bertan bektore batez adierazitako ardatza marraztu dugu, bere koordenatuak (a, b, c) dira eta bektorearen luzera 1; hau da, $a^2 + b^2 + c^2 = 1$ betetzen da. Egoera horretan, bektorea YZ planora eramateko, irudian agertzen den biraketa egin behar da. Biraketa horrek XY planoko $(a, b, 0)$ bektorea y ardatzean kokatzen du; jakina, bere luzera $(a^2+b^2)^{0.5} = (1-c^2)^{0.5}$ izango da. Biraketa horretan z ardatza mantendu egiten dugu, eta ezezaguna den bakarra, angelua da; baina hori ere ez dugu behar, $\sin\alpha = \frac{a}{(1-c^2)^{0.5}}$ baita eta era berean, $\cos\alpha = \frac{b}{(1-c^2)^{0.5}}$. Beraz, aldaketa hori lortzen duen matrizea ezaguna da: 5.1 ekuazioan agertzen den matrizean balio horiek ordezkatzean lortzen dugun matrize bera izango da.

2. Bigarren aldaketa 5.12 irudian ikus daiteke. Biraketa hori x ardatza mantenduz egiten den biraketa da, eta YZ planoan utzi dugun bektorea z ardatzera eramaten du. Horrela, hasierako ardatza erreferentzi



Irudia 5.11: bektore normal baten bidez adierazitako ardatza ZY planora eramateko biraketa.



Irudia 5.12: ZY planoan dagoen ardatza z ardatz bihurtzeko biraketa.

sistemako ardatz bihurtu dugu eta ondoren ardatz horrekiko biraketa egin beharko genuke. Dena den, bigarren biraketa horretako datuak zehaztu behar ditugu, $\sin\alpha = (1 - c^2)^{0.5}$ izango da eta $\cos\alpha = c$. Beraz, 5.2 ekuazioan agertzen den matrizean behar ditugun datuak lortu ditugu.

Hemen azaldutako bi aldaketek, erreferentzi sistema aldaketa egiten dute. Hori egin ondoren, ardatz horrekiko biraketa egin behar dugu. Biraketa horrek ardatz horri dagozkion koordenatuak (z koordenatuak) mantendu egingo ditu; hau da, x eta y ardatzei dagozkienak bakarrik aldatuko ditu. Gogoratu, hori 5.1 ekuazioan azaltzen den matrizeak egiten duela.

Ardatzarekiko biraketa burutu ondoren, hasierako erreferentzi sistemara itzultzea besterik ez zaigu falta; eta horretarako, lehenengo 5.12 irudiko aldaketaren alderantzizkoa egin behar da, eta ondoren 5.11 irudikoaren alderantzizkoa. Kontuan eduki behar da, aldaketa horiei dagozkien matrizeak ortonormalak direla, eta, ondorioz, matrizearen alderantzizkoa eta iraulia berdinak direla. Beraz, bat ezagutuz bestea berehala lor daiteke.

Azalpen horren arabera, P puntua biraketaren ondoren non geldituko den jakiteko, ondoko eragiketak egin beharko ditugu:

$$P' = A^i B^i C B A P$$

Hemen A matrizeak 5.11 irudiko biraketa eragiten duen aldaketa adierazten du; B matrizeak, berriz, 5.12 irudikoa; C matrizeak, z ardatzarekiko erabiltzaileak eskatu adina graduko biraketa; B^i matrizeak, B-ren alderantzizkoa; eta A^i-k A-ren alderantzizkoa. Pausoz pauso egin beharrean, $M = A^i B^i C B A$ matrizea aurrez kalkula daiteke, eta ondoren puntu bakoitzari matrize hori biderkatu. M matrizea biraketa-ardatzaren a, b, eta c koordenatuen eta erabiltzaileak eskatutako α gradu-kopuruaren araberakoa da, eta ez du beste ezerk aldatzen matrizea. Beraz, M matrizearen elementu bakoitza erraz lor daiteke.

Kapitulua 6

Ikuste-Sistemak

6.1 Sarrera

Erpinez osatutako objektuak mugitzen, eskalatzen eta biratzen ikasi dugu; baina oraindik ez dakigu erpin horiek, hiru dimentsiotan dauden puntu horiek, bi dimentsio dituen dispositibotan irudikatzen. Kapitulu honetan horretaz arduratuko gara, eta hiru dimentsiotako espaziotik bi dimentsiotakora pasatzeko egin behar diren kalkulu eta aldaketak zein eta nola egin behar diren azalduko dugu. Eman behar diren urrats guztiak asko dira, eta errazago ulertzeko, poliki-poliki azalduko ditugu. Hasieran, bi dimentsiotako irudiak pantailan agertzeko eman beharreko urratsak azalduko ditugu; hori egin ondoren, hiru dimentsiotako eszena bi dimentsiotan irudikatzeko jarraitu behar diren urratsak azalduko ditugu. Horretarako, balizko kamera definitu eta zehaztuko dugu.

6.2 Bi dimentsiotako irudigintza

Irudikatu nahi dugun mundua bi dimentsiotakoa bada, egin behar diren aldaketak nahiko sinpleak izango dira. Lehenengo, munduaren zein zati agertuko den erabaki behar da, eta ondorioz zati horretan dagoenaren irudia besterik ez dugu agertuko pantailan. Esandakoaren arabera, errazena, bi urratsetan egin beharko litzatekeela pentsatzea da; hau da, hasieran, irudiz agertu nahi dugun zatian gure munduko zer sartzen den eta zer sartzen ez den erabaki beharko da, eta ondoren, sartzen den guztia pantailaren erreferentzi sistemara pasatu. Lehenengo urratsari **mugaketa** deituko dio-

gu; izan ere, irudiz agertuko dugunaren mugak ezartzea izango baita. Urrats hori bai bi dimentsioetan bai hirutan egin daiteke, hots, sistemaren arabera dago. Bigarren urratsari, hau da, lanean darabilgun bi dimentsiotako koordenatu-sistematik irudia sortzeko erabiliko dugun dispositiboaren koordenatu-sistemara egin behar den aldaketari, **laukiratzea** deituko diogu.

6.2.1 Mugaketa

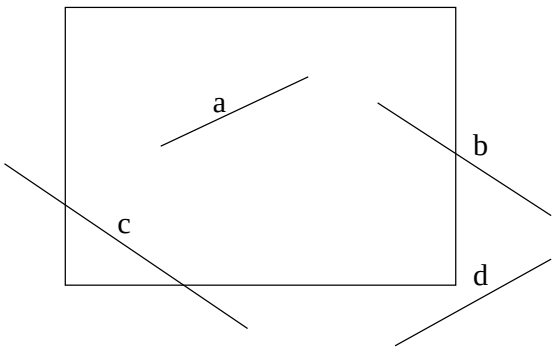
Azken irudian agertuko den mundu-zatia nondik norainokoa den adierazi behar da, hau da, munduko zein zatiren irudia lortu nahi den esan behar zaio sistemari. Adibidez, hiru dimentsiotan perspektibaz agertuko den irudi batean, zati hori muturrik gabeko piramidearen modukoa izango da, eta ondorioz hor barruan dagoenaren irudia sortuko dugu. Hori guztia proiekzio--planoan proiektatu behar da eta irudiak hori besterik ez du erakutsi behar. Beraz, zati hori gure dispositiboaren erreferentzi sistemara egokitu beharko dugu.

Aurrekoaren arabera piramidearen barnean zer dagoen eta zer ez dagoen erabaki behar da, kanpoan gelditzen denaren irudirik ez baitugu sortuko. Hori egiteko bi bide dauzkagu:

1. Hiru dimentsiotan mugatzea, hau da, piramidearen bolumenaren barnean zer dagoen erabakitzea.

2. Bi dimentsiotan mugatzea; horretarako, lehendabizi eszena osoaren proiekzioa lortu, eta ondoren, pantailari dagokion leihatilan legokeena besterik ez aukeratzea izango litzateke egin beharrekoa. Oraingoz, hori da gu arduratzen gaituen bidea.

Aukeratutako bideak bere ondorioak ditu. Lehenengoan eszenako zatitxo horretan dauden erpinak besterik ez dira proiektatu behar, baina erabaki horrek zenbaki errealekin lan egitera behartzen gaitu. Beste aukerak eszena osoa proiektatzera behartzen gaitu, baina mugaketa egiteko orduan zenbaki osoekin egin genezake lan. Gainera zenbaki osoekin egiten diren prozesu asko eta asko oso erraz egin daitezke hardware bitartez. Mugaketa egiteko, algoritmo-kopuru handia dago; guk bi ikusiko ditugu, bat marra bidezko irudientzako eta bestea poligono koloreztatu bitartez sortutako irudiak dituzten sistementzat.



Irudia 6.1: marrak leihoarekiko eduki ditzakeen lau egoerak.

Marren mugaketa: Cohen-Sutherland algoritmoa

Bigarren aukera hartuz gero, eszena osoaren proiekzioa egin behar da, eta ondoren, komeni zaigun leihatilan dagoena aukeratu behar dugu. Irudia hari-egiturakoa denean, marraz osatutako irudiak lortzen dira, eta horrelakoetarako egokia izan daiteke algoritmoa. Azken finean, laukitxo batean marrak nola dauden erabaki behar dugu, eta azterketa arin baten ondorioz gure arazoa lau kasutara mugatzen dela ikus dezakegu (ikus 6.1 irudia).

Hor ikus daitekeenez, 'a' kasuan marra guztiz barnean dago, 'd' marra berriz zeharo kanpoan eta beste bi kasuetan zatitxo bat laukiaren barruan daukate. Hasiera batean pentsa daiteke, marraren mugak non dauden jakitearekin nahikoa dugula marraren egoera zehazteko; baina 'c' eta 'd' kasuetan erpinak kanpoan egon arren, lehenengoaren zatitxo bat barruan dago, bestea berriz oso osorik kanpoan. Argi dagoena 'a' kasua da: biak barruan daudenean marra osoa barruan dagoela lasai esan daiteke. 'b' kasua ere argia da: bat barruan eta bestea kanpoan daudenean, badakigu zati bat barruan eta bestea kanpoan izango dituela. Aztertuko dugun algoritmoak bereziki mugekin ebaketarik ez daukaten marrak topatzen ditu. Horretarako proiekzio-planoa bederatzi zatitan banatzen du, 6.2 irudian ikus daitekeen bezala.

Bertan ikusten denez, zati bakoitzari 4 bit egokitzen dizkio: gure laukiaren ezkerretan badago, eskuineko bita 1 izango da; eskuinean badago, ezkerrekoak 1 balioa izango du; goian dagoenean, eskuineko bigarrenak izango du 1 balioa; eta abar.

Erpin bakoitzari lau bitak egokitu ondoren, marren kasu ezberdinak on-

0011	0010	1010
0001	0000	1000
0101	0100	1100

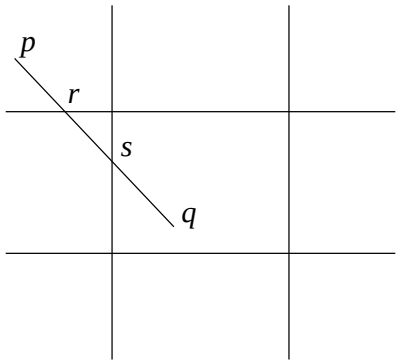
Irudia 6.2: proiekzio-planoaren 9 eskualdeak.

dokora mugatzen dira:

- bi erpinen kodea 0 balioa izatea. Kasu horretan marra guztiz barnekoa da
- bi erpinen balioen *and* eragiketa logikoaren emaitza 0 **ez** izatea. Beste kasu horretan bi erpinak gure laukitxoaren alde berean daude (goian edo behean edo eskuinean edo ezkerrean). Beraz, guztiz kanpoan geratzen da marra.
- Aurreko kasuetakoren bat ez bada, azterketa sakonagoa egin behar da, kasu horretan aurreko marrazkiko 'c', 'd', eta 'b' marrak sartzen dira.

Hala ere, irudikatze prozesuetan ematen diren kasu gehienak lehenengo bi kasuetan sartzen direla esan beharra dago; eta gutxi batzuekin egin behar izaten da azterketa sakonagoa. Azterketa berezi horren muina ondokoa da; filosofia behintzat ondokoa da, nahiz eta ezberdintasun txikiko algoritmo ezberdin asko egon.

Laukitxotik kanpo dagoen erpina aukeratu (beti bi erpinetako bat gutxienez laukitxotik kanpo egongo dela, gauza jakina da), erpin horretatik bestera joateko planoko zein muga mozten diren begiratu. Mozte-puntuak marra bi zatitan banatzen du; horrela, zati bakoitzari aurreko algoritmoa aplikatu, zein zati legokeen barnean eta zein kanpoan zehatz mehatz erabaki arte.



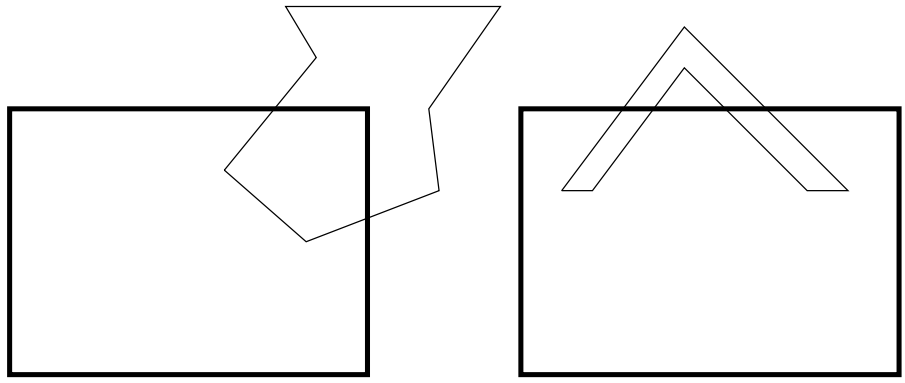
Irudia 6.3: marren mugaketan aukerak garrantzia izan dezake.

6.3 irudian ikus daitekeenez p-tik q-ra doan marra zatitu egin behar da. Zatitze horretan lehenengo r puntua hartzen badugu, p-tik r-ra doana, laukitik kanpo geratzen da; baina r-tik q-rakoarekin ez dakigu zer gertatzen den, eta berriz ebaketa puntua topatu beharko genuke, hau da, s puntua. Horrela, r-tik s-rakoa kanpoan geratzen da, eta s-tik q-rakoa barnean. Hasieratik s puntua hartu izan bagenu, prozesua azkartuko genuke; beraz, algoritmo horren optimizazioak egin daitezke, eta badaude.

Poligono-mugaketa: Sutherland-Hodgman algoritmoa

Poligonoen irudiak sortzeko erpinen arteko marrak besterik erabiliko ez baditugu, aurreko algoritmoarekin nahikoa dugu. Baina poligonoa kolore edo ehunduraz bete behar badugu, aurreko algoritmoak konpondu ezin dituen arazoak sortzen zaizkigu, 6.4 irudian ikus daitezkeenak adibidez.

Ezkerreko kasuan, mugaketaren ondorioz lortutako poligonoak lehen agertzen ez zen erpina dauka, pantailaren eskuineko goialdean dagoena alegia, eta erpin hori aurreko algoritmoak ez du lortzen. Eskuineko irudian beste arazo konponezin bat agertzen da: poligono bakarretik bi poligono sortzen dira mugaketaren ondorioz, eta arazo horri ere soluzioa eman behar zaio. Behar dugun algoritmoak, edozein poligono hartu eta zuzenean poligonoak betetzeko algoritmo batek erabili ahal izango duen poligonoa sortu behar du. Horrelakoa da Sutherland-Hodgman algoritmoa, eta bere filosofia poligono berriak sortzean oinarritzen da. Gainera, poligono berri horiek endekatuak izan daitezke, hau da, alde bat beste baten zati izan daiteke. Poligono berri



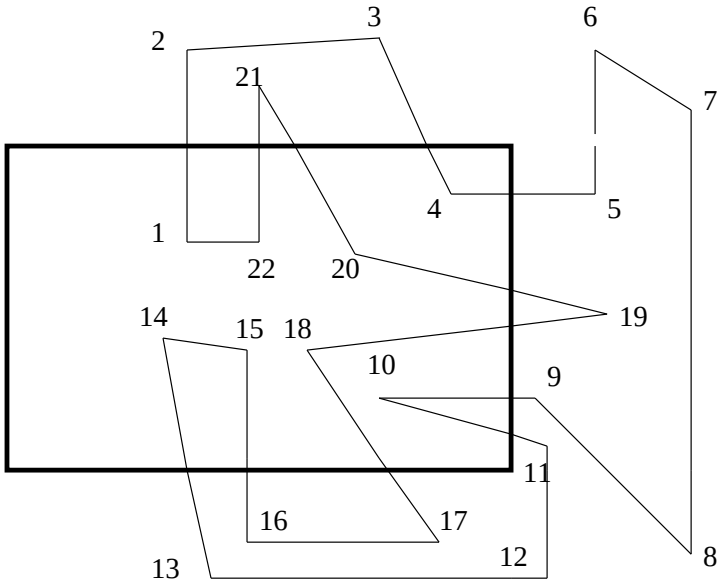
Irudia 6.4: poligonoen mugaketan sor daitezkeen arazoak.

horiek sortzeko, hasierako poligonoa pantailaren alderen batek adierazten duen marra infinituarekin mozten du, horrela erpin eta alde berriak sortuz. 6.5, 6.6, 6.7 eta 6.8 irudietan ikusten da algoritmoak nola egiten duen lan.

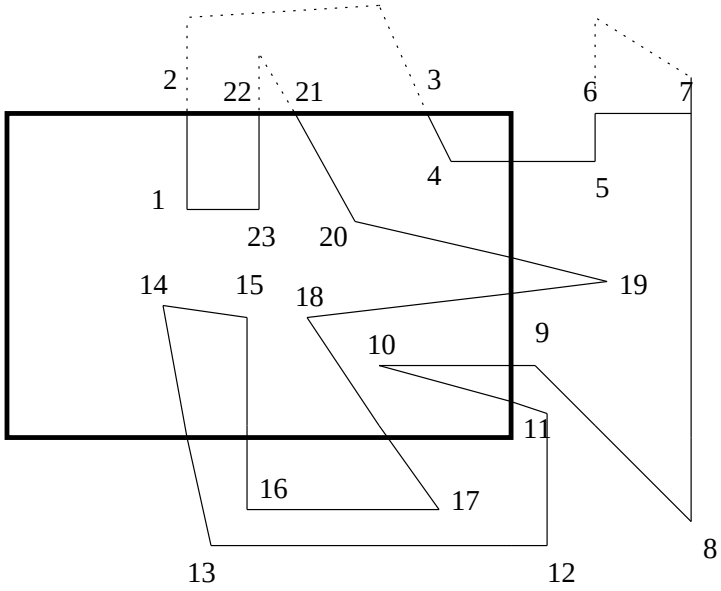
6.5 irudian pantaila marra lodiz adierazitako laukia izango litzateke, eta bertara mugatu behar da 22 erpin dituen poligonoa.

Hasierako egoeratik 6.6 irudikora pasatzeko, poligonoa pantailaren goiko marrari dagokion zuzen infinituarekin moztu dugu. 2, 3, 6, 7 eta 21 erpinak marraren goian daudenez, desagertu egingo dira. Gainera, beheko eta goiko erpinak lotzen dituzten aldeak ere aldatu egingo dira; adibidez, 1 eta 2 erpinak lotzen dituen aldea, muga igarotzen duenez, mugaraino bakarrik joango da orain, eta mugaren puntu horretan erpina jarriko dugu. Erpin berri horren hurrengoa, hau da, poligonoaren bigarren aldea adieraziko duen erpina, muga igarotzen den bigarren puntua izango da, eta kasu horretan, poligonoaren aldeak jarraituz bigarren muga igarotzea 3 erpinetik 4 erpinera doan aldeak egiten du, beraz hurrengo erpina mugaren puntu horretan egongo da eta bi erpin berri horiek poligono berriaren aldea osatuko dute. Adibide horretan 6 eta 7 erpinekin gauza bera gertatzen da baina 21 erpinarekin kasu berezia ikus dezakegu, hau da, erpin bakarra dagoen tokian bi agertuko dira, eta gainera, sortuko dugun alde berria lehen geneukan alde handiagoaren zatia da, ondorioz, sortuko dugun poligono berriak 23 erpin izango ditu eta gainera endekatua da.

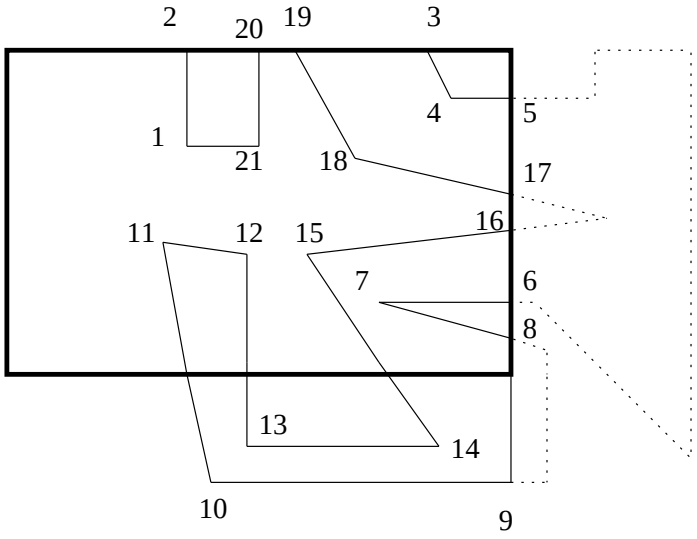
6.6 irudiko poligonotik 6.7 irudikoa lortzeko eskuinaldeko marra infini-



Irudia 6.5: mugaketa behar duen poligonoa.



Irudia 6.6: leihoko goiko marrak eragindako mugaketa-urratsa.



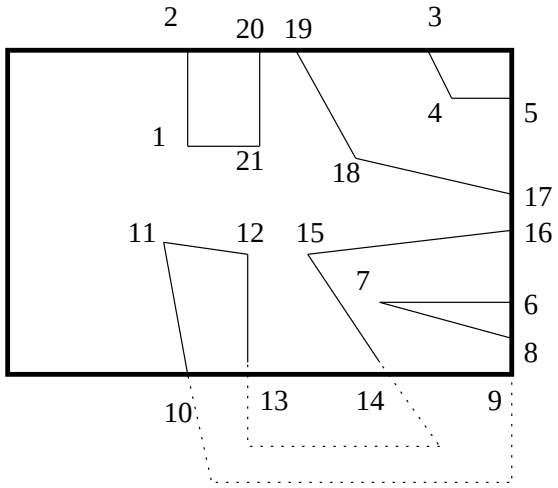
Irudia 6.7: bi lerrori dagokien mugaketa egin ondorengo egoera.

tuarekin moztuko dugu poligonoa, aurreko lan metodologia bera erabiliaz. Egoera horretan 5, 6, 7, 8, 9, 11, 12 eta 19 erpinak kendu eta sei berri sartuko ditugu. Muga lehenengoz 4tik 5era joaterakoan igarotzen da, puntu horretan erpin berria sartuko dugu eta horren hurrengoa mugaren bigarren igaro-puntuan, hau da, 9tik 10era garamatzan aldeak igarotzen duen puntuan. Ikus daitekeenez, bost erpin kendu eta bi besterik ez ditugu sortu; bestalde, urrats horretan, aurrekoan bezala, alde endekatuak sortu dira.

Azken urratsean (berez ezker aldekoarekin ere moztu beharko litzateke, baina adibide horretan ez da beharrezkoa), 6.8 irudiko 9 zenbakidun erpina sortuko dugu. Erpin hori aurreko algoritmoak inondik inora lortu ezingo zukeen erpina da, eta aurreko adibidean jarritako arazoaren soluzioa lortzen duela ikus dezakegu. Era berean, azken poligonoak eduki behar dituen "lau poligonoak"bakarraren bidez adierazten ditu; horrela, kolorez betetzeko algoritmo bati pasatu ahal izango diogu gure poligono berria. Poligono bitxi hori 6.8 irudian ikus daiteke.

6.2.2 laukiratzea

Prozesu hori erreferentzi sistemaren aldaketa soila besterik ez da, eta proiekzio-planoko erreferentzi sistematik dispositiboaren sistemara pasatze



Irudia 6.8: leiho mugen barruan gelditzen den poligono-zatia.

huts modura defini daiteke. Azken finean, proiekzio-planoko puntuei zein pixel dagokien esatea izango litzateke.

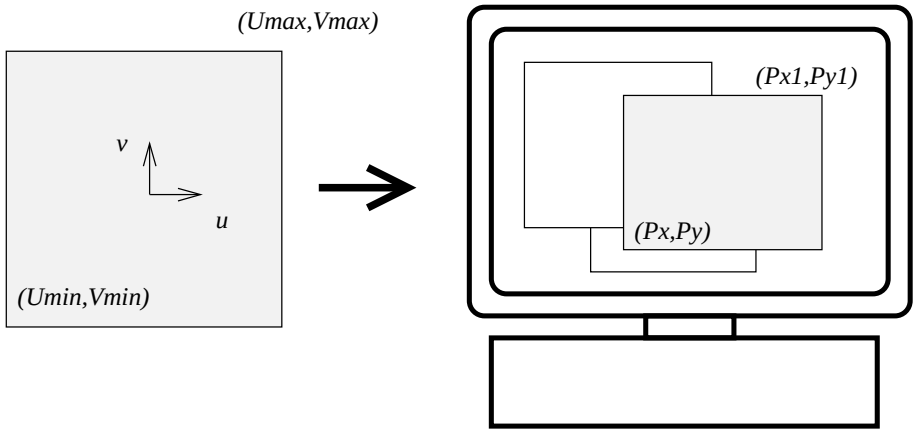
6.9 irudian agertzen den ezkerreko laukia, eskuineko pantailako laukitxo bihurtu behar dugu. Argi dago, afinitatea izan behar duela; hau da, lerro paraleloak direnak paralelo izaten jarraituko dute. Horrelako aldaketak $\overline{u} = Au + T$ motatakoak dira, eta bi dimentsiotarako hiru punturen aldaketak ezagutzea nahikoa da, A matrizea eta T leku-aldaketako bektorea ezagutzeko. Kasu horretan hiru punturen aldaketak ezagunak ditugu:

1. (U_{min}, V_{min}) puntua (P_x, P_y) puntua bihurtzen da.
2. (U_{min}, V_{max}) puntua (P_x, P_{y1}) puntua bihurtzen da.
3. (U_{max}, V_{max}) puntua (P_{x1}, P_{y1}) puntua bihurtzen da.

Hori jakinez gure sei ezezagunentzat sei ekuazio lortzen ditugu eta horien ebazpenak ondoko emaitzak ematen ditu:

$$A = \begin{pmatrix} \frac{P_{x1}-P_x}{U_{max}-U_{min}} & 0 \\ 0 & \frac{P_{y1}-P_y}{V_{max}-V_{min}} \end{pmatrix} \quad T = \begin{pmatrix} \frac{P_x.U_{max}-P_{x1}.U_{min}}{U_{max}-U_{min}} \\ \frac{P_y.V_{max}-P_{y1}.V_{min}}{V_{max}-V_{min}} \end{pmatrix}$$

Ikus daitekeenez, aldaketa hori eskalatze eta mugitze soilez osatzen da eta, nahiz eta mugitzean distortsiorik ez sortu, eskalatzerakoan irudia distortsiona



Irudia 6.9: leihotik marrazgunerako aldaketa.

daiteke. Hori gertatzerik nahiko ez bagenu, proiekzio-planoko laukiak eta pantailan erabiliko dugun laukiak proportzio berekoak izan behar dute; hau da, ondoko berdintasuna bete behar da:

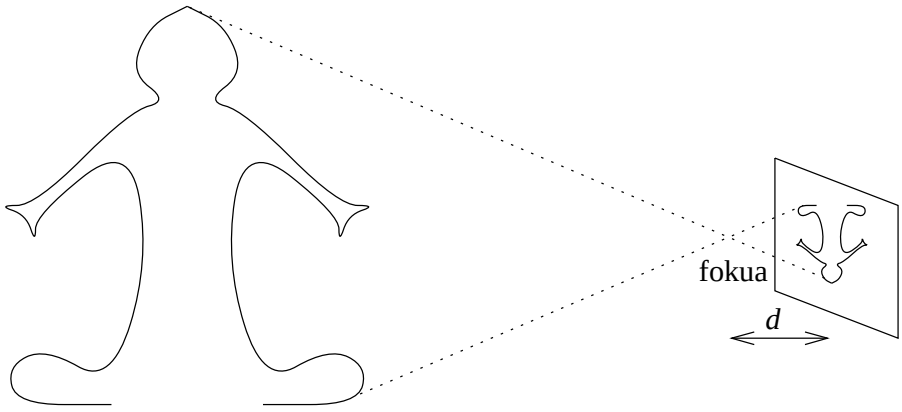
$$\frac{V_{max} - V_{min}}{U_{max} - U_{min}} = \frac{P_{y1} - P_y}{P_{x1} - P_x}$$

Hori lortzeko errazena, biak lauki zuzenak edo karratuak izatea da.

Aldaketa horrekin dispositiboetatik aske gelditzen gara; hau da, aldaketa hori egiteak laneko unitateak aukeratzeko askatasuna uzten digu, baina jakina, horrek ordaina dauka, eta ordain hori eragiketa horiek egin beharra da, denbora eskatzen duten eragiketak beti ere. Dena den, beste aukera batzuk ere ematen dizkigu, adibidez **zoom** eta **pam** direlakoak egiteko aukerak.

6.3 Hiru dimentsiotako irudigintza

Hiru dimentsiotan adierazitako munduaren zatiren baten irudia sortzeko, urrats asko eman behar da, eta horiek ulertzeko, lehenengo kameren funtzionamendua edo gure begien funtzionamendua gogoratzea komeni da. Guk begiaren barnean argiarekiko sentikorrak diren zelulak dauzkagu. Zelula horiek argi izpi bakarra jasotzen dute (erraz ulertzeko sinplifikatzea), eta horrela, zelula bakoitzak leku konkretu bateko izpia jasotzen du. Hori hala izan

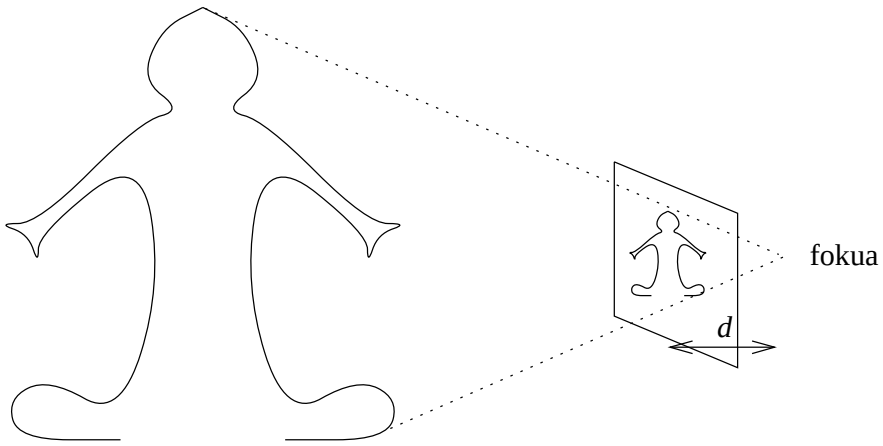


Irudia 6.10: proiekzioa begi eta kameretan.

dadin, kanpotik datozen izpi guztiak puntu batetik, fokutik, pasa arazten ditu begiak, 6.10 irudian ikus daitekeen bezala.

Irudi horretan, benetan gertatzen denaren sinplifikazioa izan arren, gure begietan irudia lortzeko gertatzen dena azaltzen da. Begi-barnean irudia alderantziz lortzen dugu eta irudi hori zelula sentikorrak dauden zatian lortzen dugu. Argazki-kameren funtzionamendua ere antzekoa da, eta irudia pelikula fotosentikorrean lortzen da. Bai begiaren eta bai kameraren kasuan irudia alderantziz dago; baina pelikula fotosentikorra fokuaren eta objektuaren artean jarriko bagenu, lortutako irudia benetakoaren proiekzioa izango litzateke. Hori 6.11 irudian ikus daiteke.

Bi dimentsiotako irudia lortzeko hiru dimentsiotako objektuko puntu bakoitzetik izpia bota behar da fokurantz, eta izpi horrek proiekzio-planoa (kameraren kasuan pelikula fotosentikorra, begiarenean zelula berezi horiek dauden lekua) igarotzen duen lekuan, puntua jarri behar da; horrela, objektuko puntu guztiekin berdina egin ondoren, proiekzio-planoan guk nahi genuen irudia lortuko dugu (bi dimentsiotan). Proiekzio-planoa, izatez, mugagabea da, baina begien edo kameren funtzionamendua ordenadore bitartez eragin nahi badugu, plano horretan kamerako pelikula-zatiari dagokion zatia mugatu beharko dugu. Irudian agertzen den moduan, planoan mugak jartzean, sortutako irudia munduaren zein zatitan dagoen adierazten dugu; hau da, irudiko piramidearen barnean dagoenaren irudia besterik ez da sortuko, beste guztia proiekzio-planoko laukitxotik kanpo proiektatzen baita. Gainera, oso urruti dauden gauzak ez direla ikusten pentsatu behar dugu,



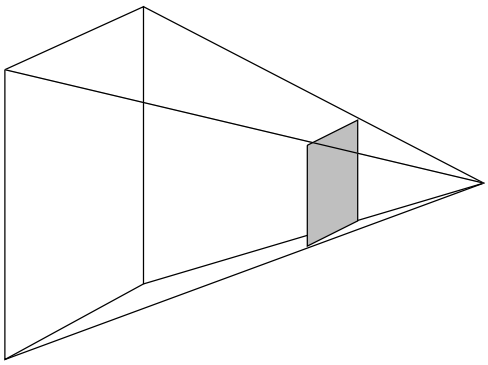
Irudia 6.11: proiekzioa irudigintzan.

eta ondorioz, distantzia jakin batetik aurrera ezer ez dagoela suposa daiteke; horregatik, piramideak bukaera dauka. Hori dena 6.12 irudian agertzen da.

6.3.1 ikuslearen erreferentzi sistema

Esan dugunez, kamerak edo begiak ikusten duena, piramide antzeko horrekin adieraz daiteke; beraz, piramidea zehatz-mehatz adierazi behar da. Gainera, era errazean aldatu ahal izateko adierazpena parametro bidez egin behar da. Kontuan eduki beharreko kontzeptu asko dago, batez ere, guk erabiliko dugun kamerak edozein motatako proiekzioak kalkulatu ahal izatea nahi badugu. Azken finean, pantailan jarri beharrekoa kamera edo ikusle horrek ikusten duena da, eta, kamerarekin egin daitezkeen gauza guztiak egiteko aukera izan nahi badugu, kameraren parametro guztiak ulertu behar dira; eta ez ulertu bakarrik, baizik eta erabiltzen jakitea ere beharrezkoa izango dugu.

Egokiena estandarretara jotzea da, baina jakina, aukera guztiak eman ahal izateko, horiek parametro asko dauzkate, eta ondorioz guztiak azaldu behar dira. Guk **PHIGS** irudigintza-sistema adieraziko dugu, baina urratsez urrats azalduko ditugu kontzeptuak, kamera sinpleenetik hasi eta konplexuagoetara poliki eta dena ulertu ondoren joanez. Irudigintza-sistema sinpleenak ere parametro-multzoa beharko du eta horiekin gure munduko objektuen informazioa aldatuz, pantailan erakutsiko duten egoerara iristeko ahalmena



Irudia 6.12: kameraren ikuste-bolumena.

eduki behar dugu. Beraz, parametro horiek emango digute mundu-koordenatuetatik dispositiboarenera pasatzeko bidea.

Parametroek nolakoak izan behar duten ulertzeko, lehenengo pantailan eszenari dagokion irudia sortzeko eman behar diren urratsak zein diren ulertzea komeni da. Hasteko, ulergarriak dira eman behar diren bi urrats:

- Lehenengo urratsa bista-aldaketa edo erreferentzi sistemaren aldaketa da. Alegia, kameraren erreferentzi sistemara pasatu beharko litzateke informazio guztia; horrekin, kamerarentzat goian dagoena goian edukiko dugu, edo urrun egon behar duena urrun egongo da.

- Bigarren eginbeharra, informazioa behin kameraren erreferentzi sisteman edukita, kameraren aurrean dagoen proiekzio-planoan proiektatzea da.

Bi urrats horiek beharrezkotzat har daitezke; horien ondoren egin behar gehiago ere aipa daitezke, baina oraingoz alde batera utziko ditugu. Aurreko bi urrats horiek mundu-koordenatuetako edozein plano proiekzio-plano modura erabiltzeko aukera ematen digute.

Irudigintza-sistema txikienak ere parametroak edukiko ditu eta parametro horiek munduko erreferentzi sistematik (3D) irudia sortzeko azaleraren (dispositiboaren) koordenatu-sistemara (2D) pasatzeko bidea eman behar digute. Horretarako munduko zein zatiren irudia lortu nahi dugun ere adierazi behar da. Hori, proiekzio-planoan leihatila definituz egiten da; eta azkenean, leihatila horretatik dispositiborako aldaketa egin behar da.

Laburtuz, kamera adierazteko behar dugun informazioak, ondoko eginbeharrak egiteko bidea eman behar digu:

1. Kamerak nola ikusten duen adierazi behar du, hau da, kameraren erreferentzi sistema.

2. Kamerak ikus lezakeen guztiaren artean zer ikusten duen, hots, zeren irudia lortu behar den ere agertu behar da.

3. Azkenik, pantailaren zein zatitan azalduko den irudia.

6.3.2 1. irudigintza-sistema

Lehenengo sistema hori kamera oso sinple bati dagokion sistema modura hartu behar da. Kamera horrentzat bi parametro besterik ez dira behar, baina kamera horrek ezin du ia ezer egin eta irudiak sortzerakoan muga handiak dauzka.

- Parametroak:

  1. Hiru dimentsiotako puntua, P.
  2. Proiekzio-planorako distantzia, d.

- Kamerak beti toki berdinera begiratzen du, hau da, beti erreferentzi sistemaren jatorrira begiratzen du.

- Proiekzio-planoaren orientazioaz ere ez da ezer esaten, baina gauzak ondo irudika daitezen, begiratze-norabidearen perpendikularra izan behar du.

- Era berean proiekzio-planoko leihatilarik ez da agertzen.

- Ikuste-bolumena ez da ematen.

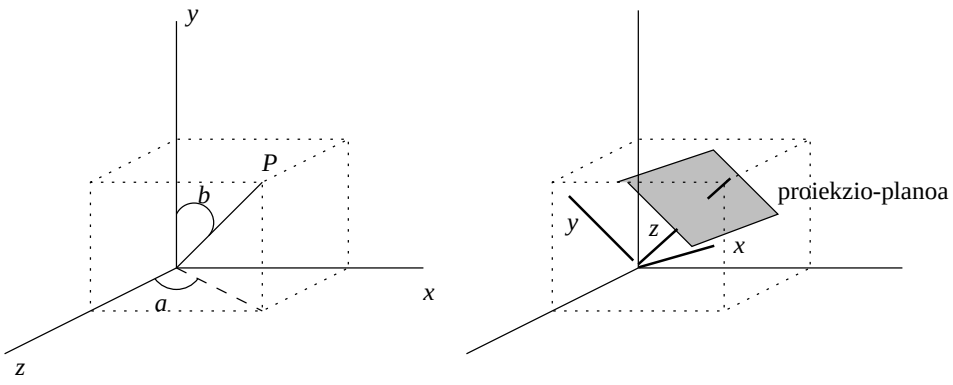
- Proiekzioaren fokua edo erreferentzi gunea, emandako puntua da, eta proiekzio-planoa puntu horretatik d distantziara kokatzen da, beti ere erreferentzi sistemaren jatorrirantz.

Sistema horretan kamera beti jatorrirantz begira dago. Beraz, lanerako ez da batere egokia, baina kontzeptuak ulertzeko lagungarria da.

Datu horietatik hasita, erreferentzi sistemaren aldaketa egin behar da, hots, kamerak nola ikusten duen adierazi behar da eta (x, y, z) sistematik (x_k, y_k, z_k) sistemara pasatu behar dugu informazio guztia; baina hori egin ahal izateko, erreferentzi sistema berria zehaztu behar dugu. Argi dago, kamerarentzat urruntasuna bere kokapenetik jatorrirantz doan bektoreak adierazten duela; beraz, bektore hori z_k izango da. Kamera P puntuan badago, eta bere koordenatuak (P_x, P_y, P_z) badira, jatorritik P puntura doan eta luzeratzat unitatea daukan bektorea lortzeko, P puntuaren koordenatuak jatorrirainoko distantziaz zatitu beharko ditugu; hau da, koordenatu bakoitza $\sqrt{P_x^2 + P_y^2 + P_z^2}$ balioaz zatitzean, kamerarentzat urruntasuna adierazten duen norabidea daukan 1 luzeradun bektorea lortzen da. Bektore hori (z_{kx}, z_{ky}, z_{kz}) dela pentsa dezagun. Oraingoz, munduko erreferentzi sisteman adierazita azaltzen da, baina badakigu kamerarentzat bektore hori $(0, 0, 1)$ bektorea izango dela.

Goranzko norabidea zehazteko, guk egiten dugun bezala egiten duela suposa daiteke; hau da, grabitate indarraren laguntzaz, guk badakigu kamera noiz dagoen zuzen jarrita; horretarako, z_k bektoreak, y bektoreak eta y_k bektoreak plano berean egon behar dute, baina, jakina, z_k eta y_k elkarzutak dira. z_k eta y ezagunak ditugunez, bi horiek osatzen duten planoa ere ezaguna dela esan daiteke. Gainera, badakigu x_k plano horrekiko elkarzuta dela. Ondorioz, y_k ardatza zein planotan dagoen badakigu, nahiz eta oraindik ez ezagutu bere koordenatuak. Bestalde, x_k plano horren beraren elkarzuta dela ere badakigu. Errazena, x_k bektorearen koordenatuak lortzea da, bektore-biderkaketa soil batez lor baitaitezke.

Azaldu dugun bezala, x_k bektorea z_k eta y_k bektoreek osatutako planoarekiko elkarzuta da. y_k oraindik ez dugu ezagutzen, baina plano horretako beste bektore bat, y, ezagutzen dugu. Bestalde, plano bateko bi bektoreren arteko biderkaketaren emaitza planoarekiko elkarzuta den bektorea denez, y eta z_k biderkatuz, x_k norabidea daukan bektorea lortuko dugu; hala ere, bektore horren luzerari buruz ez dakigu ezer. Kontuz ibili behar da bektorearen norantzarekin; $z_k \times y$ biderkaketaren emaitza $-x_k$ norabideko bektorea baita. Lanerako erabiliko dugun erreferentzi sistemaren ezaugarrietako bat $x \times y = z$ izango bada, $y \times z = x$ bete behar du. Horregatik diogu kontuz ibili behar dela. Norabidea ezaguna dela edo lor daitekeela argi dago, baina x_k bektorearen luzera unitatea izatea nahi dugu; horretarako, z_k bektorea



Irudia 6.13: Jatorrirantz begira dagoen kameraren erreferentzi sistema.

lortzeko egin dugun bezala, koordenatu bakoitza bektorearen luzeraz zatitu beharko dugu, eta horrela bai, x_k bektorea zein den jakin ahal izango dugu. Jakina, koordenatuak munduko erreferentzi sisteman adierazita daude, baina kameraren erreferentzi sisteman $(1, 0, 0)$ bektorea izango dela badakigu.

Azkenik, x_k eta z_k ezagutzen ditugunez, y_k lortzea besterik ez da falta, eta biekiko elkarzuta denez, bien arteko bektore-biderkaketa egitearekin nahikoa izango da; baina, berriz ere, biderkatzaileen ordena kontuan izan beharko dugu; $z_k \times x_k$ biderkaketaren emaitza y_k da. Gainera, oraingo honetan, zuzenean y_k lortzen dela badakigu, z_k eta x_k bat luzeradunak baitira eta elkarrekiko perpendikularrak. Biderkaketaren emaitzak y_k bektoreak munduko erreferentzi sisteman duen adierazpena azaltzen du; (y_{kx}, y_{ky}, y_{kz}). Hala ere, badakigu bektore hori kameraren erreferentzi sisteman $(0, 1, 0)$ bektorea izango dela.

Horrela lortutako erreferentzi sistema errazago uler daiteke 6.13 irudia begiratuz.

Erreferentzi sistema berria zehaztu ondoren, batetik besterako aldaketa egin behar da; hau da, kamerak ikusten duen bezala adierazi behar dira puntu guztiak. Aldaketa hori egiteko, aldaketa adierazten duen matrizea biderkatu behar zaio puntu bakoitzari; baina oraindik ez dugu matrize hori ezagutzen. Zerbait badakigu ordea; matrize horrek x_k bektorea (1, 0, 0) bektore bihurtzen du; y_k bektorea (0, 1, 0); eta z_k bektoreari matrizea biderkatuz gero

(0, 0, 1) bektorea lortuko dugu emaitza gisa. Beste era batera esanda:

$$M \times \begin{pmatrix} x_{kx} \\ x_{ky} \\ x_{kz} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$M \times \begin{pmatrix} y_{kx} \\ y_{ky} \\ y_{kz} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$M \times \begin{pmatrix} z_{kx} \\ z_{ky} \\ z_{kz} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Eta guztia batera jarriz gero:

$$M \times \begin{pmatrix} x_{kx} & y_{kx} & z_{kx} \\ x_{ky} & y_{ky} & z_{ky} \\ x_{kz} & y_{kz} & z_{kz} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Alegia, M matrizea eta beste matrize horren biderkaketaren emaitza I matrizea da; ondorioz, M matrizea beste horren alderantzizko matrizea da; matrize hori ortonormala denez, bere alderantzizkoa iraulia bera da:

$$M = \begin{pmatrix} x_{kx} & x_{ky} & x_{kz} \\ y_{kx} & y_{ky} & y_{kz} \\ z_{kx} & z_{ky} & z_{kz} \end{pmatrix}$$

Eszenaren bi dimentsiotako adierazpena lortzeko, proiekzioa kalkulatzearekin nahikoa genuke. Horretarako proiekzio-planoaren eta kameraren kokapena eduki beharko ditugu kontuan. Kamera OZ ardatzean kokatuta dago, eta jatorritik $d_1 = \sqrt{P_x^2 + P_y^2 + P_z^2}$ distantziara aurkitzen da (kameraren erreferentzi sisteman ari gara). Proiekzio-planoa, berriz, OZ ardatzarekiko elkarzuta da eta kameraren kokapenetik erreferentzi sistemako jatorrirantz d distantziara ebakitzen du ardatza; d distantzia erabiltzaileak emandako datua da. Beraz, proiekzio-planoa non dagoen badakigu, eta, kalkuluak errazteArren, erreferentzi sistemako jatorria proiekzio-planora eramango dugu, OZ ardatza ebakitzen duen puntura, hain zuzen ere. Puntu hori $(0, 0, d_1 - d)$ puntua da, beti ere kameraren erreferentzi sisteman oinarrituz, eta jatorri--aldaketa egiteko, eszenako puntu guztiei $(0, 0, d_1 - d)$ kendu behar zaie. Aldaketa horren ondoren, proiekzio-planoa OX eta OY ardatzek osatutako

planoa dela esan daiteke; eta bertan proiektatu behar dira eszenako puntuak. Demagun (a, b, c) koordenatuak dituen puntua proiektatu nahi dugula. proiekzio-fokua $(0, 0, d)$ puntua da, jatorri-aldaketa egiterakoan $(0, 0, d_1)$ puntuari $(0, 0, d_1 - d)$ kendu baitiogu. Puntua proiektatzean $(P_x, P_y, 0)$ lortuko dugu; 6.14 irudian ikus daitekeen bezala, puntua eta fokua lotzen dituen marrak proiekzio-planoa non ebakitzen duen kalkulatu behar da, eta horretarako, hirukien propietate bat erabil dezakegu:

$$\frac{P_x}{d} = \frac{a}{d-c}$$

eta

$$\frac{P_y}{d} = \frac{b}{d-c}$$

Horrela, bi koordenatuen balioa:

$$P_x = \frac{a}{1-\frac{c}{d}}$$
$$P_y = \frac{b}{1-\frac{c}{d}}$$

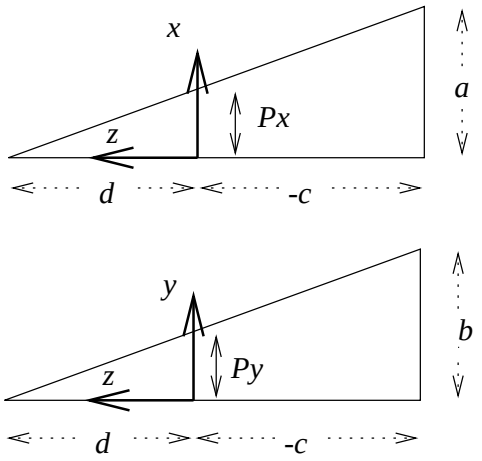
Edo beste era batera ikusita:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \text{ puntutik } \begin{pmatrix} \frac{x}{1-\frac{z}{d}} \\ \frac{y}{1-\frac{z}{d}} \\ \frac{z}{1-\frac{z}{d}} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} x \\ y \\ z \\ 1-\frac{z}{d} \end{pmatrix} \text{ puntua lortu behar da.}$$

Azken ikuspegiaren arabera, bila gabiltzan proiekzioa lortzeko, ondoko eragiketa erabil daiteke:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{d} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1-\frac{z}{d} \end{pmatrix}$$

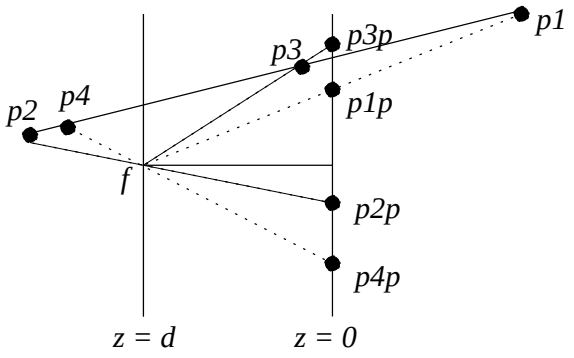
Eta hori ere, matrizeen eta bektoreen arteko eragiketa da; beraz, puntuei eragindako beste matrizeekin bateragarria da. Dena den, matrize hori, besteekin konparatuz, berezia dela ohartzea komeni da; azken lerroa beste guztietatik ezberdina dauka, eta horri esker proiekzioa lortzeko gai da. Ez ditugu matrize horren bereizitasun guztiak aztertuko, baina interesgarria deritzodan



Irudia 6.14: (a, b, c) puntua XY planoan proiektatzean lortzen diren x eta y koordenatuak.

bat aipatu nahi nuke: $z = d$ planoko edozein puntu hartuz gero, eta matrizea biderkatuz gero, emaitza puntu bat izan beharrean bektore bat da, hau da, laugarren koordenatu bezala 0 dauka. Kontuan eduki behar da kamera plano horretan kokatu dugula. Horren interpretazioa infinituko puntuena izan daiteke; puntu horiek infinitura eramaten dituela alegia, eta espazioa deformatu egiten duela. Deformazio horrek perspektibako proiekzioa proiekzio paralelo bihurtzen du; horregatik, aldaketa eragin ondoren, puntuen lehenengo bi koordenatuek puntuaren proiekzioa adierazten dute. Aldaketaren eragina, eszena infinitutik begira bageunde bezala ikustea da, baina, jakina, gure kamerak ikusten duen bezala ikusiz.

Interpretazio horretan sakonduko bagenu, arazo batera helduko ginateke; hau da, kameraren aurretik atzera doan edozein marren arazora: $z = d$ planoa zeharkatzen badu, planora hulbildu ahala infiniturantz hurbiltzen da, planoan bertan dagoenean infinituan dago, eta pasatu ondoren, berriz infinitutik datorrela esan daiteke. Horrela esanda hitz jokoa dirudi, baina, 6.15 irudian ikus daitekeen bezala, bere eragina dauka: Planoaren alde banatan dauden bi punturen, $p1$ eta $p2$, arteko bidea hartzen badugu, bi puntuen arteko puntuetatik pasako gara, tartean $p3$. Proiekzio-matrizea biderkatu ondoren, ibilbide bera egiteko, bi puntuei dagozkien proiekzioen arteko bidea egin beharko dugu; hau da, $p1p$ eta $p2p$ puntuen arteko bidea. Baina, eta hona hemen arazoa, batetik besterako bidean $p3$ puntuari dagokion proiekzio



Irudia 6.15: Bi punturen arteko puntuak, proiektatu ondoren, bi puntuei dagozkien proiekzioen artean egon beharrik ez du.

puntua igaro behar da, $p3p$, eta puntu hori ez dago $p1p$ eta $p2p$ artean. Biak elkartzen dituen marran dago, baina ez bien artean. Horra hor, infinitura joatea eta bertatik etortzearen adibide polita, $p1p$ puntutik $p2p$ puntura joateko, lehendabizi infinitura jo, $p2p$ puntutik urrunduz, eta gero handik bueltan, beste aldetik, $p4p$ puntutik pasatuta, $p2p$ puntura iristea.

Gertakizun hori dela eta, kameraren atzean dagoena, proiekzioa kalkulatu baino lehen mugatzea komeni dela esan behar da.

6.3.3 2. irudigintza-sistema

Argazkilariek aurreko kamerarekin ezer gutxi lortu ahal izango lukete. Izan ere, argi dago muga handiegiak dituela kamera horrek; horietako bi, gogorrenak, ondoko biak izan daitezke:

1. Kamerak beti erreferentzi sistemaren jatorrirantz begiratzen du; edozein tokitan koka daiteke, baina jatorrirantz begira.

2. Kamerari ez diogu lagatzen bertikaltasuna apurtzen. Alegia, argazkiak **tente** agertuko lizkiguke beti; ezin ahal izango genuke Eiffel dorrearen argazki okerra sortu.

Arazo horiek konpontzeko, ondoko sistema sor dezakegu:

- Parametroak:

    1. Hiru dimentsiotako bi puntu, P eta Q.

2. Proiekzio-planorako distantzia, d.
3. Okertze-angelua, bertikaltasuna zer neurritan hautsiko den adieraziko du.

- Berriz ere, proiekzio-planoaren orientazioaz ez da ezer esaten, baina aurrekoan bezala, begiratze-norabidearen elkarzuta da.
- Lehen bezala, proiekzio-planoko leihatilarik ez da agertzen.
- Eta ikuste-bolumenik ez da ematen.
- Proiekzioaren fokua edo erreferentzi gunea, emandako puntua da, eta proiekzio-planoa puntu horretatik d distantziara kokatzen da, baina oraingoan, bigarren punturantz.

Berriz ere, kameraren erreferentzi sistema zehaztu behar da. Horretarako, lehengo moduan egin daiteke, baina orain z_k bektorea bi puntuen arteko bektorea izango da. Gainera, beste bi bektoreak z_k-rekiko okertze-angeluak adierazten duen adina biratuta egongo dira; horrela, bertikaltasuna guk nahi bezalakoa izango da.

Bigarren sistema horretan agertzen diren parametroak, beste era batera agertzea egokiagoa gerta daiteke. P eta Q eman beharrean, nahikoa da P eta P-tik Q punturantz doan edozein bektore ematea. Bestalde, okertze-angelua adierazi beharrean, bektore bitartez adieraz daiteke goranzko norabidea zein den; horrela, gainera, eragiketak erraztuko genituzke. Beraz, kamera horrek behar dituen parametroak honako hauek izango lirateke.

1. Hiru dimentsiotako puntua, P.
2. Proiekzio-planorako distantzia, d.
3. Bi bektore, bata begiratze-norabidea, bestea goranzko norabidea.

Parametro horiekin kameraren erreferentzi sistema, (x_k, y_k, z_k) alegia, lehen esandakoari jarraituz zehaztuko genuke. Hau da, x_k bektorea goranzko norabidearen eta begiratze-norabidearen arteko bektore-biderkaketa, eta y_k begiratze-bektorearen eta x_k-ren artekoa (z_k begiratze-bektorea bera da, jakina!). Horrela, erreferentzi sistemaren aldaketa egin eta proiekzioa kalkulatzearekin nahikoa dugu.

6.3.4 3. irudigintza-sistema

Badirudi aurreko kamerak aukera gehiago ematen dituela, baina hala ere, kamera horrekin **zoom** edo **pam** deritzena ezin dira egin. Gainera, ez du mugatzen kamerak ikusten duen bolumena, eta kalitatezko kamerak aukera horiek behar ditu (kalitatezkoa estandar modura hartu behar da); beraz, parametro gehiago beharko ditugu.

- Parametroak:

  1. Hiru dimentsiotako puntua, P.
  2. Begiratze-norabidea adieraziko duen bektorea.
  3. Goia zein norabidetan dagoen adieraziko duen bektorea.
  4. Proiekzio-planorako distantzia, d.
  5. Plano urrunerako distantzia, f. Distantzia horretaz harantzago objektuak ikusezin bihurtzen dira.
  6. Proiekzio-planoko leihatilaren aldearen luzera. Horrekin, ikuste--bolumena adierazten duen piramidearen neurriak ezartzen ditugu.

- Kamerak begiratzen duen norabidea, aurrekoa bezala, edozein izan daiteke.

- Proiekzio-planoak edozein bektore dauka goranzko bektore gisa, eta jakina, begiratze-norabidearen elkarzuta da.

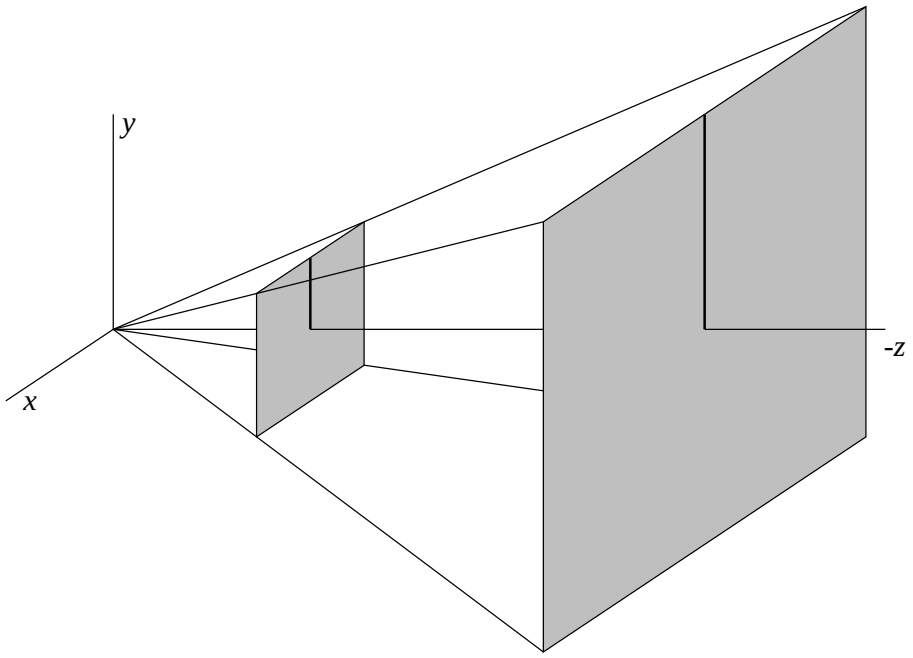
- Ikuste-bolumena ematen da, horretarako proiekzio-planoko leihatilaren aldearen neurria eta urruneko planoa ematen dizkigute.

Parametro horiekin ikuste-bolumena adierazten digute, 6.16 irudian agertzen den piramidea hain zuzen ere.

6.3.5 4. irudigintza-sistema. PHIGS

- Parametroak:

  1. Hiru dimentsiotako puntua, VRP. Puntu hori erreferentzi sistema berriaren jatorria izango da.

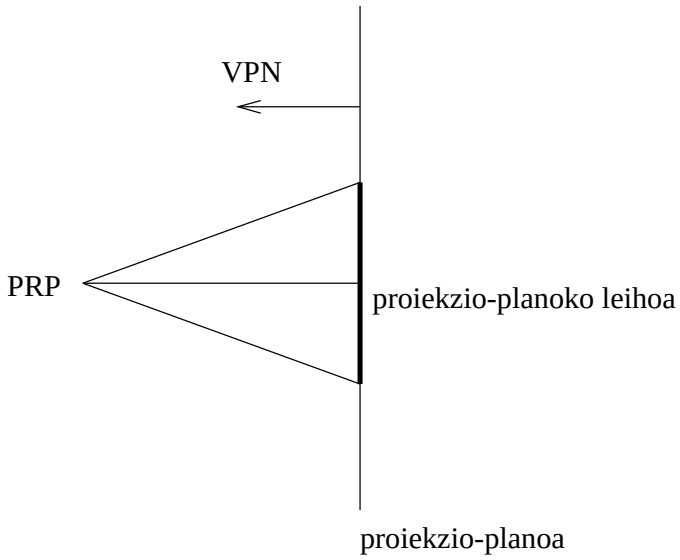


Irudia 6.16: Edonora begira dagoen kameraren ikuste-bolumena.

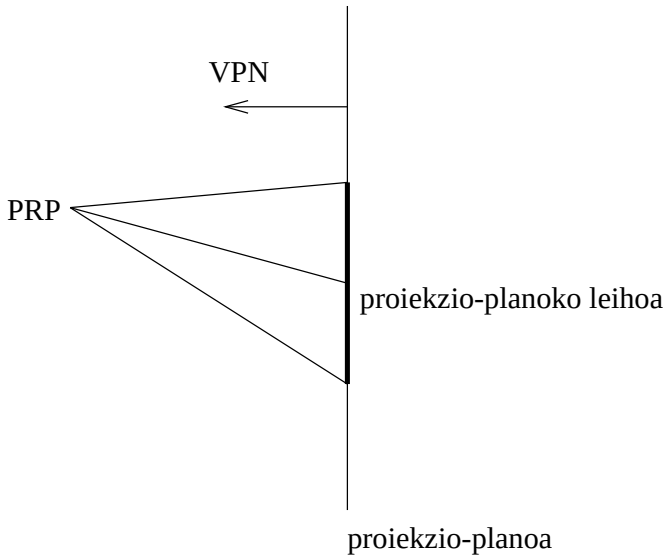
2. Begiratze-norabidea adieraziko duen bektorea, VPN.
3. Goia zein norabidetan dagoen adieraziko duen bektorea, VUP.
4. Proiekzio-mota, perspektiba ala paraleloa.
5. Proiekzioaren fokua edo nora proiektatuko den adieraziko duen puntua, PRP. Bere koordenatuak kameraren erreferentzi sisteman adierazten dira.
6. Plano urrunerako distantzia, b. Distantzia horretaz harantzagoko objektuak ikusezin bihurtzen dira.
7. Plano hurbilerako distantzia, f. Horren eta aurrekoaren arteko objektuak baino ez dira ikusiko.
8. Proiekzio-planorako distantzia, d. Hiru plano horiek erreferentzi sistema berriaren x_k, y_k bektoreek osatzen duten planoaren paraleloak dira, eta b, f eta d zenbakiek plano horien kokapena adierazten dute. Kokapen hori erreferentzi sistema berriaren jatorriarekiko distantzia adieraziz agertzen da; jakina, z_k bektorea non mozten duten agertzen dute.
9. Proiekzio-planoko leihatilaren neurria. Hori, $x_{\min}$, $y_{\min}$, $x_{\max}$ eta $y_{\max}$ parametroekin adierazten da, eta lau datu horiek proiekzio-planoko leihoa adierazten dute. Horrela, ikuste-bolumena mugatuta gelditzen da, PRP-tik leiho horren lau ertzetara doazen marrek mugatutako piramidea izango baita bolumen hori. Hala ere, bolumen horretan hurbileko eta urruneko planoen artean sartzen dena izango da benetako ikuste-bolumena. Kontuan eduki beharrekoa da, leihoaren erditik PRP-ra doan bektoreak, ikuste-norabidearen paraleloa izan beharrik ez duela; beraz, proiekzio zeiharrak lor daitezke (6.17 eta 6.18 irudiak ikusi).

- Kamerak begiratzen duen norabidea, aurrekoa bezala, edozein izan daiteke.
- Proiekzio-planoak edozein bektore dauka goranzko bektore gisa, eta begiratze-norabidearen elkarzuta da.
- Ikuste-bolumena ematen da; horretarako, proiekzio-planoko leihatila, urruneko planoa eta hurbileko planoa ematen dizkigute.

Sistema horretan irudia sortzeko, ondoko urratsak eman behar dira:



Irudia 6.17: Proiekzioa.



Irudia 6.18: proiekzio zeiharra.

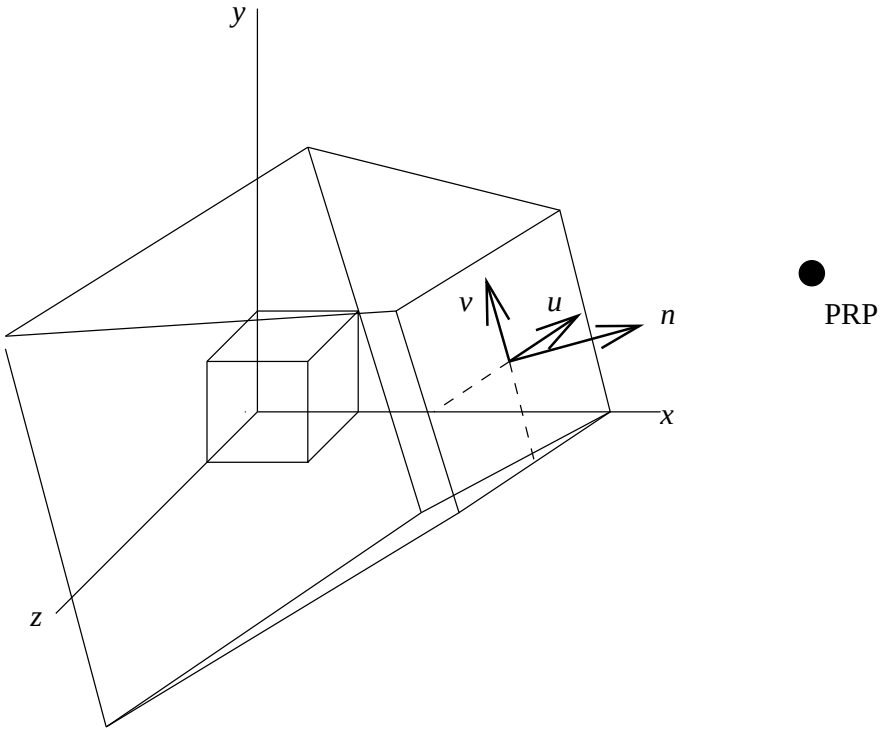
1. Munduaren erreferentzi sistematik kamerarenera pasatu behar da. Horretarako, aurreko sisteman bezalaxe, sistema zein den zehaztuko da eta aldaketa egiten duen matrizea lortu behar da. Hori egiteko, VRP, VPN eta VUP parametroak erabili behar dira. Beheko irudietan agertzen den bezala, lehendabizi erreferentzi sistema berriaren oinarria zehaztu behar da. Sistema berrian hurbiltasuna edo urruntasuna adierazten duen bektorea VPN izango da; hori sistema berriko z_k da. Horrez gain, goranzko norabidea adierazten duen bektorea daukagu; baina gure y_k berriak z_k bektorearekiko elkarzuta izan behar du, eta VUP bektoreak ez dauka baldintza hori bete beharrik. Hala ere, badakigu VUP eta VPN bektoreek mugatzen duten planoan egongo dela y_k berria; gainera x_k plano horrekiko elkarzuta da. Beraz, x_k lortzeko VUP eta VPN bektoreen arteko bektore-biderkaketa egitearekin nahikoa dugu (eta moduluagatik zatitu). z_k eta x_k ezagututa, y_k lortzea erraza da, biekiko elkarzuta baita; beraz, bien arteko bektore-biderkaketak adieraziko du y_k.

   6.19 irudian u, v eta n bektoreek osatzen dute erreferentzi sistema berria. Hiru bektoreak ezagutuz, erreferentzi sistemaren aldaketa egitea besterik ez zaigu falta; hau da, eszenako puntu guztiak oinarri berrian adierazi behar ditugu. Horretarako, lehenengo jatorria kokapen berrira eraman behar da. Hori 6.20 irudira pasatzeko urratsa izango litzateke.

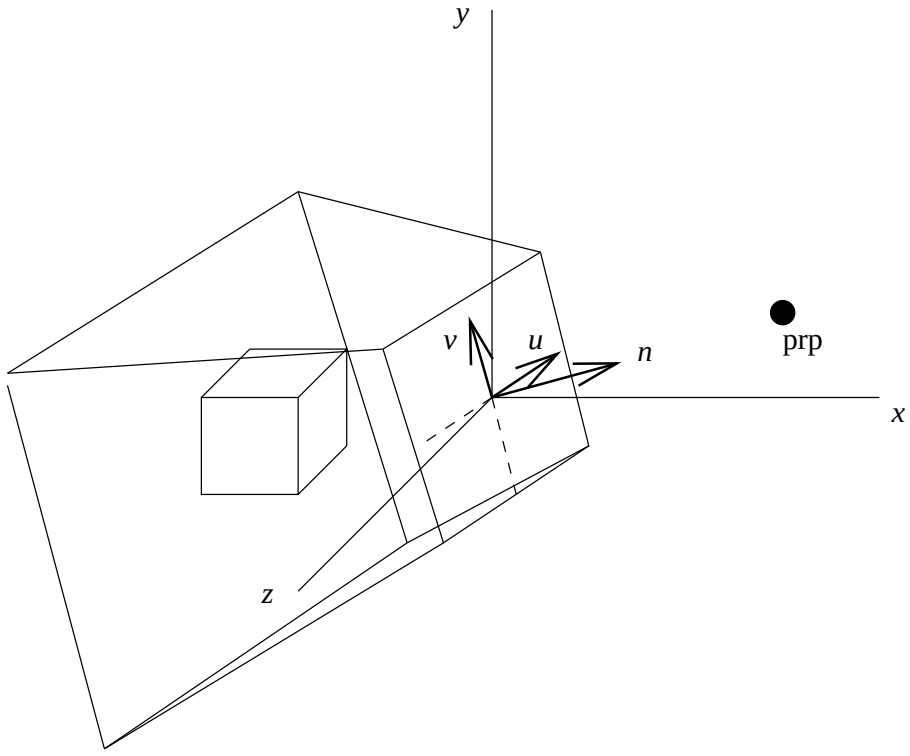
   Ondoren, puntu guztien koordenatuak aldatu behar dira, (u, v, n) sisteman adierazi behar baitira. Azken finean, bizpahiru biraketa egitea besterik ez da. Urrats hori 6.21 irudiaren egoerara eramango gintuzkeena da.

   Horrela, eszena kamerak ikusten duen bezala adieraztea lortu dugu, eta gu kameraren lekuan egongo bagina, berak bezala ikusiko genuke, hau da, 6.22 irudiak agertzen duena ikusiko genuke.

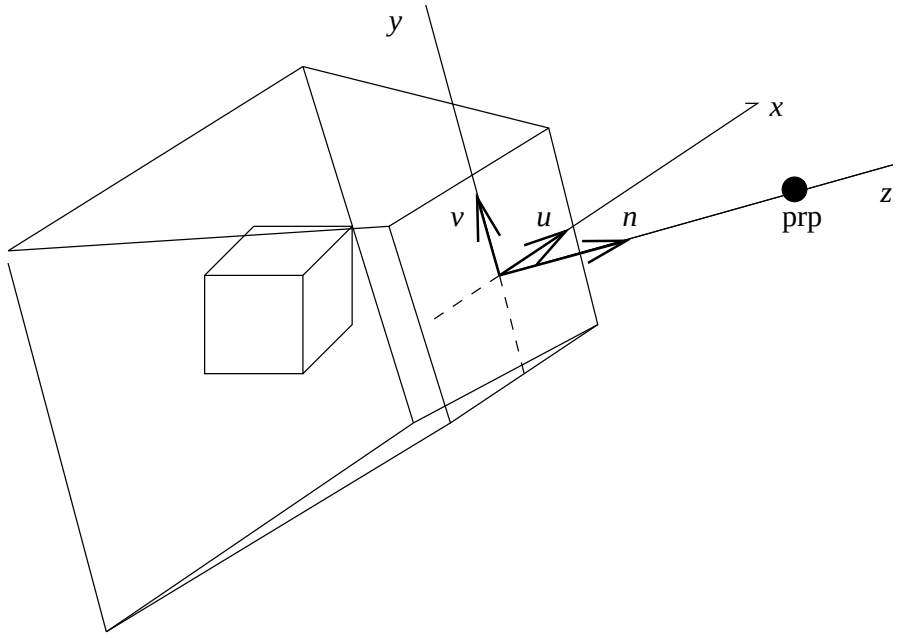
2. Erreferentzi sistemaren aldaketa egin ondoren, proiekzioa lortuko dugu, baina proiekzioa kalkulatzean leihatilaren erdigunetik PRP-ra doan bektorea eta VPN bektorea paraleloak ez badira, hau da, proiekzioa zeiharra bada, proiektatu baino lehen bi bektore horiek bat egin behar dute, **shear** izeneko aldaketaren bitartez lor daiteke hori, eta ondoren ikuste-bolumenaren barnean dagoen guztia proiektatuko da.



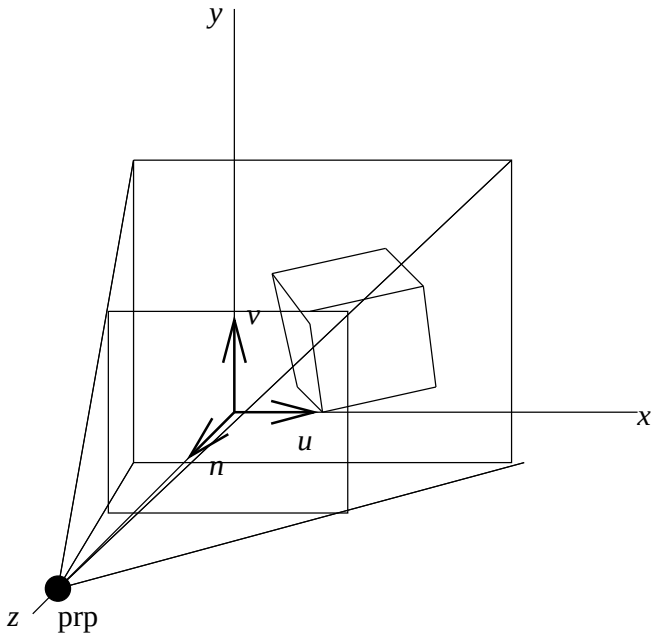
Irudia 6.19: eszena baten ikuspegia, munduko erreferentzi sisteman.



Irudia 6.20: munduaren jatorriaren aldaketa.



Irudia 6.21: Kameraren erreferentzi sistema munduaren ikuspuntutik.



Irudia 6.22: kameraren ikuspegia.

3. Azkenik, proiekzio-planoan dagoena, pantailan edo paperean jarri beharko dugu. Horretarako, neurri-aldaketa egin beharko da, proiekzio-planoko leihatilatik dispositiboko leihatilara neurriak aldatu egin baitaitezke. Gainera, gerta liteke proiekzio-planoko leihatila eta dispositibokoa proportzio berekoak ez izatea; kasu horretan eskalatua ez da bi dimentsioetan berdina izango.

Sistema horrekin edozein ikuspegi azal daiteke, **zoom** eta **pam** lor daitezke eta horrez gain proiekzio zeiharren bidez irudi aldatuak ere sor ditzakegu.

6.3.6 Atze-aurpegien ezabapena

Objektu baten atze-aurpegiak zein diren jakiteko, planoaren ekuazioa erabil dezakegu. (x, y, z) puntu bat A, B, C eta D parametroak dituen poligonozko gainazal (plano) baten barnean dago, ondokoa betetzen badu:

$$Ax + By + Cz + D < 0$$

A, B eta C parametroek, poligonoaren bektore normala definitzen dute. Bektore horren eta V bektorearen arteko angelua -90 eta 90 gradutako tartearen barnean badago, poligonoa atze-aurpegia izango da:

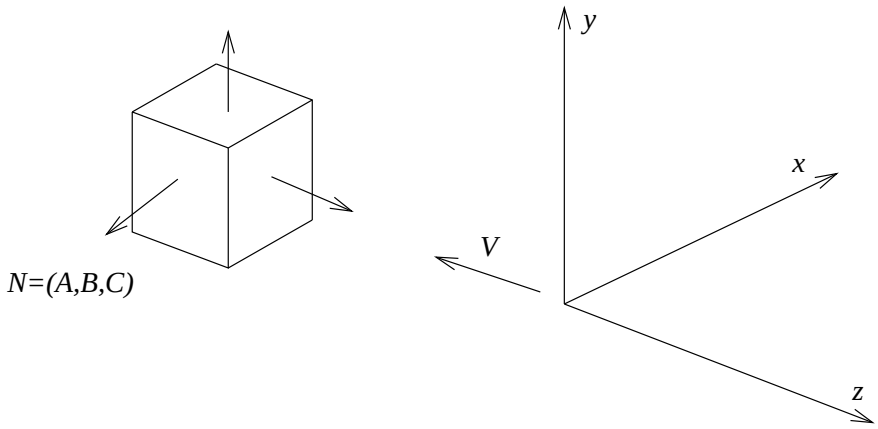
$$VN = |V||N| \cos(\alpha) > 0$$

Atze-aurpegiaren testa ikuslearen erreferentzi sisteman eta proiektatu ondoren kalkulatzen bada, kalkuluak asko errazten dira. Proiektatu ondoren ikuslea $(0, 0, \infty)$ kokagunera doa eta ondorioz proiekzioa paraleloa da, hau da, ikuslearengandik gainazaleranzko V bektorea $(0, 0, -1)$ izango da eta biderkaketaren emaitza, berriz:

$$VN = 0A + 0B - 1C = -C$$

Beraz, ikuslearen erreferentzi sistema proiektatuan atze-aurpegiak zein diren jakiteko, poligono bakoitzaren normalaren hirugarren osagaiaren zeinua aztertu behar da, hots, N bektorearen hirugarren osagaia. C=0 izatean, poligonoa ikuslearekiko elkarzuta izango da, eta hori ikustea ezinezkoa izango da. Beraz, edozein poligono atze-aurpegia dela esango dugu, proiektatu ondoren, bere bektore normalaren hirugarren osagaia $C \leq 0$ izatean.

Orokorrean, ikuslearen erreferentzi sisteman, $C \leq 0$ izatean poligonoa atze-aurpegia denik ezin daiteke esan. Zeinua, erreferentzi sistema motaren,



Irudia 6.23: Atze-aurpegi ez direnen bektore normala ikuleari begira dago.

ikuslearen kokapenaren ($V_z < 0$ edo $V_z > 0$) eta poligonoa definitzen duten erpinen ordenaren araberakoa (erlojuaren zentzuan ala aurkakoan) izango baita.

Kapitulua 7

Errealitate-itxura

7.1 Ezkutuko azalerak

Ezkutuan geratzen diren azalerak kentzen dituzten algoritmoen sailkapena, Shutherland-ek 1974. urtean idatzitako paper batean agertzen da, eta gaur egun oraindik ere esanguratsua dela onartu behar da. Sailkapenean objektu-espazioan ala irudi-espazioan mugitzearen arabera banatzen dira algoritmoak, eta horrez gain, *koherentzia*ren erabilera ere neurtzen zaie. Elkarren artean gertu dauden puntuetako ezaugarriak antzekoak izaten direnez, puntu bateko ezaugarriak kalkulatzerakoan, aurretik kalkulatutako inguruko puntuetako ezaugarrien informazioa erabil dezakegu. Puntu batetik besterako aldaketa ezaguna bada, eta aldaketa hori batuketa soil gisa adieraz badaiteke, algoritmoen abiadura handitu egin daiteke. Objektuen edo eta irudien koherentzi maila askotan aprobetxa daiteke:

- *Objektuen arteko koherentzia.* Objektuak elkarren artean guztiz banatuta baldin badaude, ezagutza lagungarria eskain diezagukete kasu askotan. Adibidez, objektu bat beste baten atzean baldin badago, atzekoak ez du sekula aurrekoa estaliko eta atzeko objektuaren azalerek aurrekoarenak estali ote ditzakeen ez dugu konprobatu beharko.

- *Azaleren arteko koherentzia.* Azalera bateko ezaugarriak gehienetan oso gutxi aldatzen dira azaleran zehar, eta ondorioz, aurretik kalkulatutako puntuaren ezaugarriei batuketa soilak egitera muga dezakegu puntu baten ezaugarrien kalkulua.

- *Ertzen arteko koherentzia.* Ertz bat ikuskorra baldin bada, ikuskorra izango da puntu batean beste ertz edo azalera batekin topo egin bitartean.

- *Ertz inplizituen koherentzia.* Azalera bat beste batean sartzen bada, bien arteko ebaketa marra, ertz inplizitua, mozte-puntuen bidez kalkula daiteke.

- *Lerroen arteko koherentzia.* Irudi batean lerro batetik hurrengora dagoen aldea oso txikia izan ohi da; bata eta bestearen artean aldaketa gutxi agertzen dira, eta batean dauden objektuak bestean ere agertu ohi dira.

- *Pixelen arteko koherentzia.* Pixel auzokideak azalera ikuskor berekoak izan ohi dira.

- *Span edo tarteen arteko koherentzia.* Aurrekoaren espezializazioa da. Horren arabera, lerro batean dauden tarte bakoitzeko pixelak azalera bakarrari dagozkio; beraz, tarte bakoitzeko kalkulu bakarrarekin tarte osoko pixelak zein azalerari dagozkion jakin dezakegu.

- *Sakoneraren koherentzia.* Objektu bateko puntuen sakonera gertu dauden puntuetan gutxi aldatzen da; gainera, azalera bereko puntuetako sakonera aldaketa erregularra izan ohi da eta batuketa bezala adieraz daiteke. Ondorioz, puntu bateko sakonera ezaguna bada, bere albokoarena kalkulatzeko, batuketa bakar baten bidez kalkula dezakegu.

- *Irudien arteko koherentzia.* Animazio-sekuentzietan irudi baten eta hurrengoaren artean aldaketa oso gutxi egon ohi da. Kontuan eduki, segundoko hogei irudi inguru behar direla, begiak jarraitasuna soma dezan; ondorioz, bi irudiak ia momentu berekoak dira eta oso antzekoak. Bata lortzeko egindako kalkuluek, bigarrenarentzat balio dezakete.

Zati ikuskorrak zein diren erabakitzeko, algoritmo-mota asko dago, mota bakoitzak koherentzi baldintza ezberdinak erabili ahal izango ditu, algoritmo batzuk baldintza gehiagoz baliatuko dira, eta beste batzuk ezingo dute horrenbestez baliatu. Dena den, ahalik eta errendimendu handiena lortzen saiatu beharko dute, ikusgaitasuna erabakitzea ez baita nola-halako lana, eta gehienetan kalkulu konplexuak egin beharko baititugu; edo kalkulu horiek ez egitearren, algoritmo edo egitura konplexu eta handiak erabili beharko baititugu.

Algoritmo horien artean bi mota oso erabiliak direla esan behar da; bata *Z-buffer* metodoan oinarritutako algoritmoek osatzen dute, eta bestea *scan-line* edo lerroz lerroko azterketan oinarritutakoek. Horiez gain, mota gehiago ere bada, baina ez dute hain harrera ona izan, edo betebehar jakinetarako garatutako algoritmoak dira. Ikusgaitasuna aztertzeaz gain, ikusten denaren kolorea eta intentsitatea ere kalkulatu behar da, eta askotan elkarrekiko lotura eduki dezakete; hala ere, ikusgaitasunaren eta kolorearen kalkulua batera egiten duen algoritmorik ere bada. Horien artean harrera oso ona izan dutenak, *Ray Tracing* edo izpi-hedaketa motako algoritmoak dira.

7.1.1 Z-buffer.

Algoritmo horrek, pixel bakoitzeko bertan ager daitezkeen poligono guztien artetik gertuen dagoena aukeratzen du. Esandakoaren arabera, irudiaren espazioan lan egiten duen algoritmoa litzateke Shutherland-en sailkapenean. Dena den, algoritmoak lan egiteko era berezia dauka, eta pixel bakoitzari dagozkion poligono guztiak kalkulatu beharrean, poligono bakoitzari dagozkion pixelak kalkulatuz dihardu. Horregatik, aurreko poligonoen informazioa gordetzera behartzen gaitu, eta pixel bakoitzeko momentu horretara arteko poligonoek zein sakonera eman diguten jaso behar dugu. Hurrengo poligonoaren pixelak kalkulatu ondoren, horiei dagokien sakonera aurrekoenarekin konparatu behar da; berria gertuago balego, berri horri dagokion kolorea jarri beharko genioke pixelari, eta dagokion sakonera ere eguneratu egin beharko genioke. Bestalde, poligonoari dagokion pixel baten urruntasuna aurreko poligonoena baino handiagoa bada, pixel horretan ez dugu aldaketarik egingo, ez kolore ez eta sakoneran ere. Nabaria da, lan egiteko era horrek pixel adina sakonera gordetzeko matrizea behar duela: horri *Z-buffer* deritzo.

Z-buffer algoritmoak hainbat abantaila ditu (ikusi 7.1 sasikodea). Horietako bat objektuaren adierazpidea edonolakoa izan daitekeela da. Puntu bateko sakonera ezagutzeko gaitasuna baldin badauka adierazpenak, Z-bufferraren bidez ageriko pixel guztiak lor daitezke eta ezkutukoak automatikoki ezabatuko genituzke. Horrela, nahiz CSG adierazpidea, nahiz azalera parametrikoa, nahiz poligonozkoa, nahiz horien antzera sakonera ezagutzeko posibilitatea daukan edozein adierazpiderekin lan egin ahal izango dugu.

Hala ere, arazorik ere sor dezake algoritmoak, nagusiena lanerako behar duen memoria-tamaina da. Irudia osatzeko behar den pixel adinako matrizea edo bufferra behar du eta pixel bakoitzeko sakonera jasotzeko gaitasunarekin. Sakonera zenbaki bidez adierazten denez, matrizea zenbakiz osa-

```
Z-buffer(objektuak,(x, y) pixelak)
    {
    for (x, y) guztiak
        {
        Z-buffer[x, y] = sakonera_maximoa;
        Pixel[x, y] = sakoneko_intentsitatea;
        }
    for (objektu guztiak)
        for (objektuaren aurpegi guztiak)
            for (aurpegiaren (x_a, y_a) pixel guztiak)
                if ( Z-buffer[x_a, y_a] ≥sakonera(x_a, y_a))
                    {
                    Z-buffer[x_a, y_a] = sakonera(x_a, y_a);
                    Pixel[x_a, y_a] = Intentsitatea(x_a, y_a);
                    }
    for ((x, y) guztiak)
        marraztu Pixel[x, y];
    }
```

Irudia 7.1: Z-buffer algoritmoa.

tuko da, baina zenbakiak adierazteko heina aukeratu behar da. Darabilgun eszenatokiaren arabera, objektu-kopuruaren arabera eta horien arteko distantzien arabera zenbaki horiek era askotakoak edo zehaztasun ezberdinekoak izango dira, baina gutxienez 20 bit beharko dira, 32 balira hobe. Baina, jakina, matrizearen neurria zenbaterainokoa izan daitekeen begiratu behar da; adibidez, imajinak 700×700 pixel balitu, eta sakonera adierazteko 24 bit (3 byte) hartuko bagenitu, 1470000 byte beharko genituzke matrizea adierazteko, eta zenbaki hori kontuan hartzeko modukoa da. Irudi handiagoek matrize handiagoa beharko dute, eta 3 byte beharrean 4 beharko bagenitu, matrizearen neurria gehiegitxo izatera ere hel liteke. Dena den, gaur egun terminal grafiko askok Z-buffer dedikatua daukate; horrela, memoria nagusitik hori guztia hartu beharrik ez dugu izango. Horrelakorik ez dagoen kasuetan ere, irtenbideak lor daitezke; adibidez, irudia zati edo azalera ezberdinetan bana daiteke, eta Z-buffer txikiagoarekin zati bakoitzerako kalkuluak egin ditzakegu.

Memoriaren behar horri aurre egiteko, lerroz lerroko lana ere egin daiteke; hau da, irudiko pixel lerro bakoitzeko lerro hori ukitzen duten objektuen pixelak hartuz, irudia bere osotasunean kalkula daiteke. Jakina, lan gehiago emango digu, lerro bakoitzean parte hartzen duten objektuak zein diren erabaki behar baita; baina, horrela nahiko memoria ez daukaten makinetan ere erabili ahal izango genuke algoritmo hori. Horrelako algoritmoak *Scanline Z-buffer* izenaz ezagutzen dira.

Z-bufferraren beste aukeratako bat, bi eszenatoki bakar batean jartzearena da, edo alde batetik zenbait objekturen irudia kalkulatu, beste batetik (prozesadore paralelo baten bidez, adibidez) beste zenbaiten irudia, eta bukaeran pixel bakoitzeko bietan gertuen dagoena aukeratzea besterik ez da egin behar. Horrela animazioetarako ere erabilgarritasunik aurki geniezaioke. Eszenatoki jakin batean objektu gutxi batzuk mugitzen badira, eszenatokiaren Z-bufferra lortu eta jaso ondoren, azken irudiari mugikorrak diren objektuak gehitzea besterik ez zaio falta; gainera, hurrengo irudirako eszenatokiaren Z-bufferra kalkulatuta geneukake eta horrek lana asko azkartuko liguke.

Beste mugetako bat objektu gardenena litzateke, era horretan ezin baititugu bi objektu puntu berean ikusi. Algoritmo horri zenbait aldaketa eginez, *A-buffer*raren algoritmoa lor daiteke. Horretan, gertuen dagoen objektua gardena bada, pixelari sakonera eta kolorea jarri beharrean, erakusleren baten bidez egitura berezi bat adieraz geniezaioke. Erakuslearen bidez egiturako informazioa lor daiteke; hori sakonera eta atzean leukakeen objektua zein

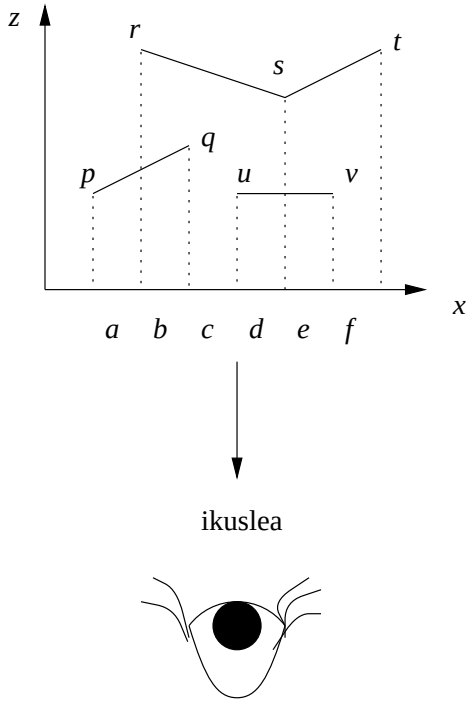
eta nolakoa den adierazteko erabil daiteke, bere kolore eta sakonerarekin. Horrela, objektu gardena eta atzekoak nahastu ahal izango genituzke, pixelari dagokion kolorea emateko. Hori irtenbide bat izan daiteke, baina izpien jokaera ez du kontuan hartzen; hau da, izpiak objektuak igarotzean norabide-aldaketa jaso dezake, eta fenomeno hori ezingo genuke lortu.

Bestalde, algoritmoaren abiadura mantsotzen duten kalkulu asko egiten da, batez ere pixel bakoitzari dagokion kolore edo intentsitatea kalkulatzean. Egia da objektu guztien puntu guztiei kolorea ez zaiela kalkulatzen, baina momentuko gertuena denean, kalkulatu egin behar zaio, eta ondoren datorren objektu bat bere aurrean baldin badago, berriz kalkulatu behar da kolorea, eta aurreko kalkulu guztiak alferrikakoak izan dira. Ikus daitekeenez, algoritmo horrek gauza asko du alde, baina aurkako batzuk ere bai. Hala ere, memoriaren muga gero eta txikiagoa da, batez ere merkatuaren garapenaren ondorioz, eta hain sinplea izateak, hardware bidez egiteko aukera ematen du; horregatik, oso hedatuta dago.

7.1.2 Spanning Scanline

Z-buffer algoritmoarekin puntu askotako kolorea alferrik kalkulatzen genuen eta memoriaren arazoa ere hor zegoen. Memoriarena konpontzeko, Z-buffer scanline metodoa erabil daiteke, hau da, pixel-lerro batean parte hartzen duten objektu guztiak kontuan hartuz, horiei dagozkien pixelak kalkulatu eta lerro-bufferrean jasotzearekin nahikoa da. Dena den, objektu bakoitzari dagozkion pixelen kolorearen kalkulua, askotan alferrik egingo da, beranduago gainjar daitekeen objektu bat ager baitaiteke. Kalkulu horiek egin nahi ez baditugu, lerroan ager daitezkeen objektuak eta horiei dagokien azken irudia nolakoa den begiratzea komeni da; bertan, lerroa aurpegi jakin batzuei dagozkien pixel-ilara txikiz osatzen dela ikusten da. Ilara horien hasiera eta bukaera, beti aurpegiren baten hasiera edo bukaeran kokatzen da; eta, jakina, aurpegi berririk hasten edo bukatzen ez den bitartean, pixelak objektu berari dagozkiela ziurta genezake.

Hitzez esandakoa 7.2 irudian ikus daiteke. Bertan, lerro bati dagozkion ilara ezberdinak karaktere batez adierazten dira, a-tik f-ra. Tarte bakoitzean aurpegi bakar bati dagozkion pixelak dira ikuslearengandik gertuen daudenak. Beraz, aurpegi horren kolorea izango dute pixelok. Lehenengo tartean, a tartean, objektu bakarra ikus daiteke; ondorioz, berari dagokion kolorea jarri behar da. Bigarrenean ordea, b tartean, bi objektu agertzen dira. Tarte horren hasiera r-s aurpegiaren hasierak kokatzen du; bukaera, berriz, p-q aur-

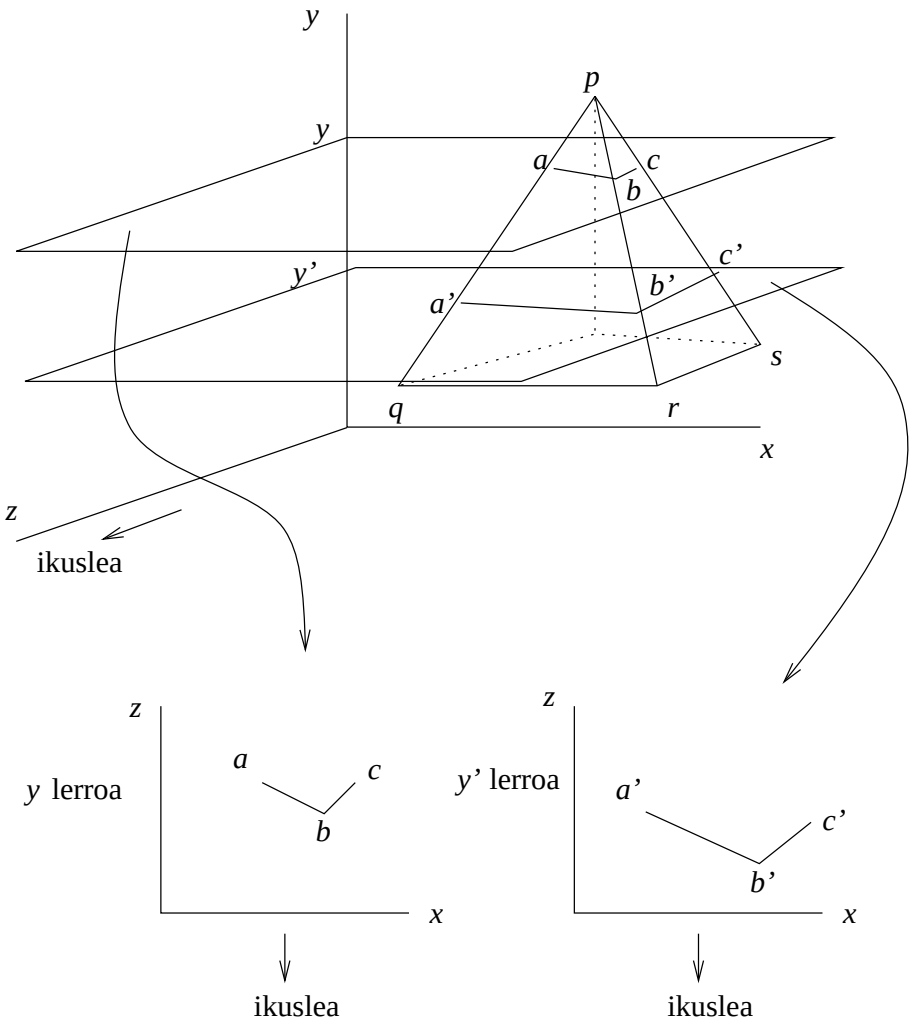


Irudia 7.2: objektuak bata bestearen barnean sartu gabe ikusleak lerro batean ikus ditzakeen tarteak.

pegiaren bukaerak. Bi objektu daudenez, bietan gertuen dagoenaren kolorez argituko dira pixelak, gure kasuan p-q aurpegiaren kolorez. Hirugarren tartean, c tartean, aurpegi bakarra dago. Tartearen hasiera p-q aurpegiaren bukaerak eta bukaera u-v aurpegiaren hasierak kokatzen dute, eta tartean r-s aurpegia besterik ez da agertzen; beraz, berari dagokion kolorez piztuko dira pixelak. Hirugarren tarte horretan q eta u puntuek zehazten dituzte mugak, baina kontuan eduki beharreko aurpegia ez da ez q-ri dagokiona ez eta u-ri dagokiona ere. Beraz, argi eduki beharko dugu aurpegi bakoitzaren hasiera eta bukaera non kokatzen den, eta, era berean, lerroko tarteak zein diren. Tarte bakoitzean aurpegi berri bat hasten edo bukatzen dela ere kontuan edukitzekoa da. Tarteak mugatzeko, aurpegien puntu guztiak ordenatu behar dira, x txikienetik handienera; beraz, lerro bakoitzeko aurpegiak non hasten eta non bukatzen diren jaso beharko dugu.

Lerro batean aurpegi berri bat, orain arteko lerroetan agertu ez den aurpegi bat, agertzen bada, aurpegi horren goialdeko erpinera heldu garela esan nahi du. Momentu horretan ertzak lerroa non mozten duen jakitea erraza da, erpinaren x koordenatuak adierazten du; baina hurrengo lerroa non mozten duen jakiteko, ertzaren malda jasotzea komeniko litzateke, y-ren aldaketekiko x-ren aldaketa nolakoa den esaten baitigu. Era berean, sakonera ere ezagutu behar da, beraz, y-ren aldaketekiko z nola aldatzen den ere jaso behar dugu ertz bakoitzean. 7.3 irudian ikus daitekeenez, ertz bakoitzak x eta z ezberdinetan zeharkatzen du y bakoitzari dagokion lerroa ($y = k$ planoa), eta zeharkatze-puntuak kalkulatzeko, ertzen aldaketa nondik norakoa den jasoz gero, lerroz aldatzean, zeharkatze-puntuari aldaketa horiek gehitzearekin nahikoa dugu. 7.3 irudian p-q ertzak y-ri dagokion lerroa a puntuan zeharkatzen du; irudi berean y lerroari dagozkion tarteetan ikus daitekeen bezala, puntu horrek x jakina eta z jakina izango ditu; era berean, ertz berak y'-ri dagokion lerroa a' puntuan zeharkatzen du, eta puntu horri dagozkion x eta z ere ezagutu behar ditugu, lerro horretan zein aurpegi ikus daitezkeen jakin ahal izateko.

Lerroa mozten duen ertz bakoitzak, aurpegi baten hasiera edo eta bukaera adierazten digu (ikus 7.3 irudia), y lerroak piramidearen mutur inguruan p-q, p-r eta p-s ertzak mozten ditu, horiek a, b eta c puntuak definitzen dituzte. a puntuak p-q-r aurpegiaren hasiera adierazten du, b puntuak, berriz, aurpegi horren bukaera eta p-r-s aurpegiaren hasiera, eta azkenik, c puntuak p-r-s aurpegiaren bukaera. Ikus daitekeenez, aurpegi bakoitzak beti hasiera eta bukaera edukiko ditu eta ertzek lerroko tarteak non hasten diren eta non bukatzen esaten digute; baina tarte batean aurpegi bat baino gehiago ager-



Irudia 7.3: Eszenatoki bateko ertzek lerroetan eragiten dituzten ebaketak eta lerro bakoitzari dagozkion aurpegiak.

```
Spanscanline(ertzak, aurpegiak)
{
Aurpegien taula sortu eta eragin bita faltsua jarri
Ertz eraginkorren taula hutsa sortu
for (lerro bakoitzeko)
        {
        if (lerroan ertz berririk hasten da)
                Ertz eraginkorren taulan sartu
        Ertz eraginkorren x kalkulatu
           /* ertzak lerroa zein x-etan mozten duen */
        x horiek ordenatu eta lerroa tartetan zatitu
        for (tarte bakoitzeko)
                {
                ertza = tartearen hasiera markatzen duena
                ertzari dagozkion aurpegiei eragin-bita aldatu
                   /* ertzak aurpegi-hasiera edo eta bukaera adieraz dezake*/
                Eragin-bita egiazkoa duten aurpegietatik gertuenekoa hartu
                Aurpegi horri dagokion kolorea eman tarteko pixel guztiei
                }
        if (lerroan ertzik bukatzen da)
                Ertz eraginkorren taulatik kendu
        }
}
```

Irudia 7.4: Lerroz lerro tarteak marrazten dituen algoritmoa: Spanning Scanline.

tzen bada, adibidez 7.2 irudian gertatzen den bezala, aurpegi horiek tarte horretako puntu batean zein sakonera daukaten jakitea komeni da. Horretarako, planoaren ekuazioa erabil genezake, x eta y jakinak direnez, ezezagun bakarra z baita. Kontuan jaso beharko da, tarte bakoitzean zein aurpegi ager daitekeen; horretarako, tarte bakoitzaren mugetan, aurpegi berri baten hasiera edo bukaera adierazten badu, hala jaso beharko da. Ondorioz, ertz horri dagozkion aurpegiak zein diren non edo non jasota eduki behar da.

Esandako guztiarekin argi dago, ertzen informazioarekin egituraren bat behar dugula, eta era berean aurpegiei dagokien informazioa ere adierazi behar dugula; gainera, lerro bakoitzean ertz guztien arteko batzuk besterik

ez ditugu kontuan eduki beharko; lerroan eragina duten ertzen taula edo lista beharko dugu ondorioz. Horri lotuta, ertz horiek aurpegi jakin batzuei dagozkie, eta ertz bat igarotzen dugun bakoitzean, aurpegi berri bat kontuan eduki behar dugula edo aurpegiren bati kasu ez diogula egin behar esan nahi du; hau da, tarte bakoitzean zein aurpegik eduki lezakeen eragina jaso beharko dugu beste taula edo lista batean.

Ertz bakoitzak gutxienez ondoko informazioa edukiko du:

- Lehenengo erpinaren x koordenatua.
- Lehenengo erpinaren y koordenatua.
- Beste erpinaren y koordenatua.
- $\frac{\triangle x}{\triangle y}$, lerroz aldatzean x berria kalkulatu ahal izateko.
- Zein aurpegitakoa den adierazteko identifikatzailea.

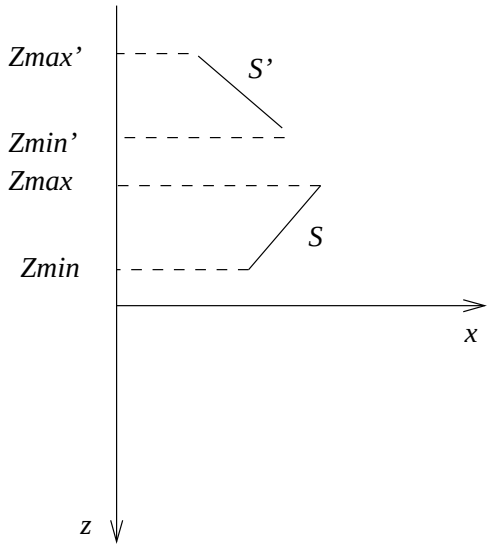
Aurpegi bakoitzak, berriz, honako hau beharko du:

- Aurpegiaren identifikatzailea.
- Aurpegia dagoen planoaren ekuazioa (sakonera ezagutzeko moduren bat).
- Aurpegiari dagokion koloreari buruzko informazioa, edo kalkulatzeko bidea emango duen informazioa.

Horrekin guztiarekin Spanning Scanline algoritmoa 7.4 algoritmoa izan daiteke; hori erpinei eta aurpegiei dagokien informazioa erabiliz osa daiteke. Bestalde, algoritmoa era askotan gara daitekeela argi eduki behar da. Posibilitate asko dago: memoria arazorik ez badago, ertz guztiek lerro bakoitza non zeharkatzen duten aurretik kalkula daiteke, horrela, lerro bakoitzean zein ertz dauden aurretik jakingo dugu; gainera, zeharkatze-puntu guztiak ordenatuta sar ditzakegu lerro bakoitzean, horrela tarteak automatikoki kalkulatuta geneuzkake. Egitura hori *y-bucket* egitura izenaz ezagutzen da.

7.1.3 Sakonerarekiko ordenazio-algoritmoa

Algoritmo horrek, poligono guztiak sakoneraren arabera sailkatzen ditu, ondoren poligonoak atzetik aurrera marraztuz. Modu horretan, marrazkilariek egiten duten bezala, lehenik urrunen dauden poligonoak marraztuko



Irudia 7.5: Sakoneran gainjartzen ez diren bi poligono.

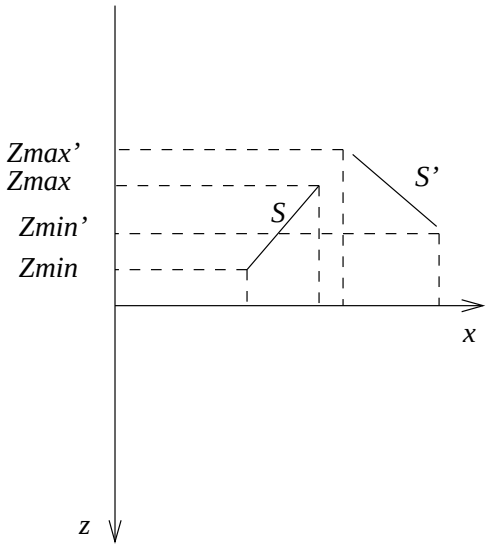
ditugu gero gertuen daudenak horien gainean marrazteko. Algoritmoa aplikatzeko jarraitu behar diren urratsak hauexek dira:

1. Ikuslearen erreferentzi sisteman kokatutako poligonoak beren z koordenatuaren arabera sailkatzen dira lista batean.

2. Listatik S poligono urrunena atzitzen da eta listan geratzen diren poligonoak aztertzen dira. S eta listako poligonoren bat OZ ardatzean gainjarri egiten direnentz kalkulatzen da. Gainjartzen den poligonorik ez badago, S marraztu egiten da eta hurrengo poligonoari ekingo diogu. S poligonoa, OZ ardatzean, beste batekin gainjartzen bada, bi poligonoen arteko konparaketak egin beharko ditugu, lista ordenatuan aldaketarik egin behar den jakiteko. Ondoren, lista hustu arte, lehenengo urratsera itzuliko ginateke.

Bi urratsak errepikatuz, lista hustean, irudia marraztuta izango dugu. 7.5 irudian, z norabidean ez baina xy planoan gainjarrita dauden bi poligono ikus daitezke.

Bi poligono OZ ardatzean gainjartzen direnean, zenbait konparaketa egin beharko ditugu, lista ordenatua berriz ordenatu behar den jakiteko. Ondoko baldintzaren bat betetzen bada, ez dugu horrelakorik egin beharko:



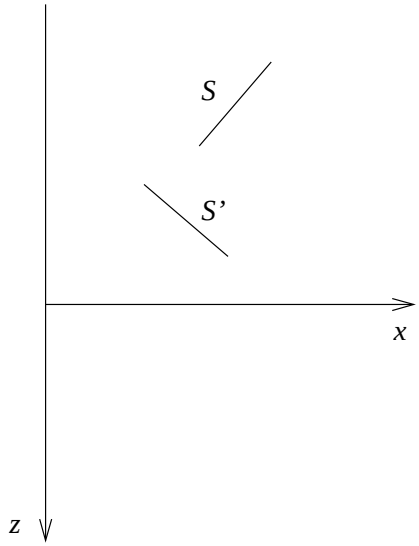
Irudia 7.6: Sakoneran gainjartzen diren bi poligono.

- xy planoko bi poligonoen borne-errektangeluak ez dira gainjartzen.
- S poligonoa, ikuslearen kokapenarekiko, S' poligonoaren atzean dago.
- S' poligonoa, ikuslearen kokapenarekiko, S poligonoaren aurrean dago.
- Bi poligonoen proiekzioak, proiekzio-planokoak, ez dira gainjartzen.

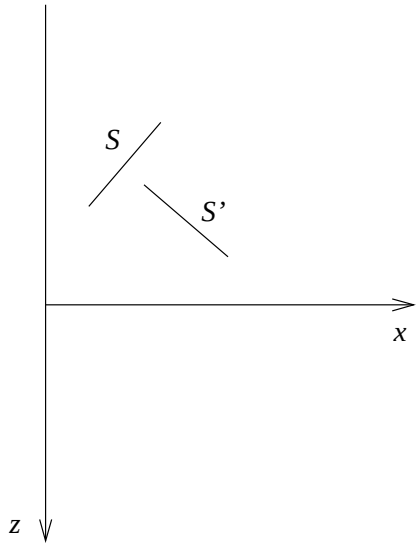
Baldintza horiek bata bestearen ondoren aztertu beharko lirateke, eta, baldintzaren bat betetzean, testa bukatutzat emango genuke; hau da, sakonerarekiko ordenatuta dagoen poligono-lista, behar bezala egongo litzateke.

Lehenengo testa egiteko, bai OX ardatzaren norabidean eta bai OY ardatzarenean, poligonoen borne-errektangeluarrak gainjarri egiten direnentz aztertu behar da; borneak gainjartzen ez badira, poligonoak ez dira elkarren artean estaltzen. OZ norabidean bai baina OX norabidean gainjartzen ez diren bi poligonoren adibidea, 7.6 irudian ikus daiteke.

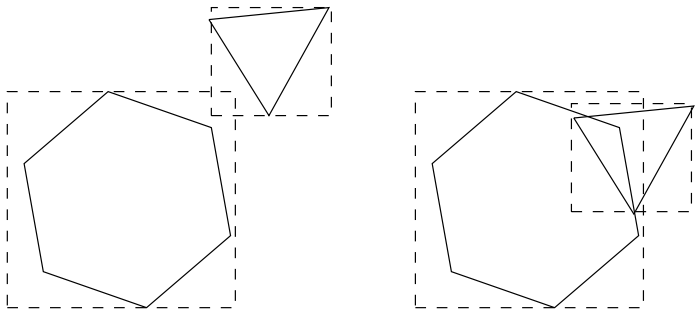
Bigarren eta hirugarren testak egiteko, poligono baten erpin guztiak beste poligonoaren planoaren ekuazioan ordezkatu eta emaitzaren zeinua aztertu beharko dugu, erpin bakoitza planoaren zein aldetan dagoen jakiteko. Adibide batzuk 7.7 eta 7.8 irudietan ageri dira.



Irudia 7.7: S poligonoko erpinak $S^{'}$ poligonoaren atzean daude, baina $S^{'}$-ko erpinez ezin dugu esan ez S poligonoaren aurrean ez atzean daudenik.



Irudia 7.8: $S^{'}$ poligonoa S-ren aurrean dago.



Irudia 7.9: Borne-bolumenak gainjarri arren, objektuek elkar estali beharrik ez dute.

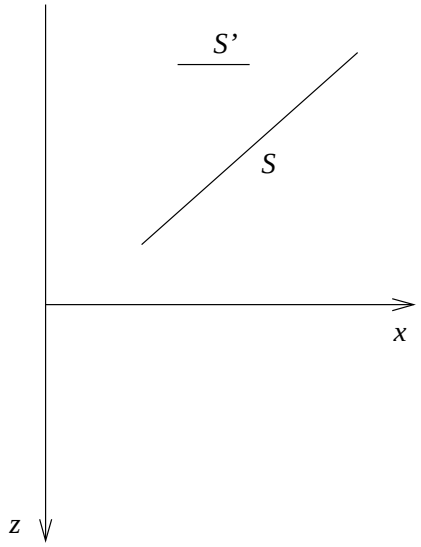
Lehenengo hiru baldintzetako bat ere betetzen ez bada, laugarren testa aztertu beharko dugu. Laugarren testa betetzen den ala ez jakiteko, poligonoak xy planoan (proiekzio-planoan) ebakitzen direnentz aztertu behar da, poligonoetako ertzak beren artean ebakitzen direnentz begiratuz. 7.9 irudian, nahiz eta bi poligonoen borne-errektangeluarrak gainjarri, bi poligonoek elkar estaltzen ez dutela ikus daiteke. Kasu horretan, laugarren testa beteko litzateke.

Lau baldintzetako bat ere ez betetzean, S eta S' gainazalak lista ordenatuan trukatu beharko lirateke, S gainazala S' gainazalaren atzean dagoenik ezin daitekeelako esan. Listan trukatu beharko liratekeen bi gainazal, 7.10 irudian ikus daitezke. 7.11 irudiko S eta S" elkarrekin trukatuko genituzke, ondoren S" eta S' elkarrekin trukatzeko eta listaren ordenazio zuzena lortzeko. Poligono baten kokapena trukatzean, poligono horrentzat test-prozesua errepikatu behar da.

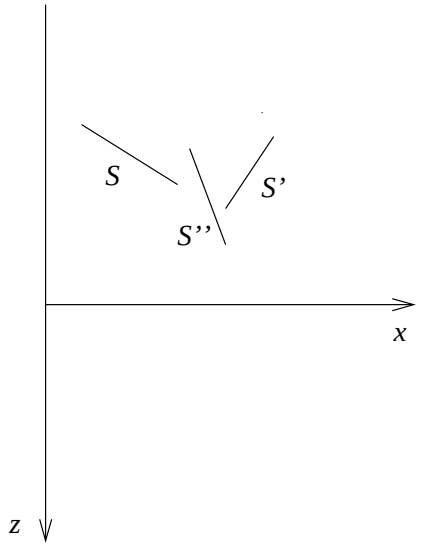
7.12 irudian bezala, bi poligonok edo gehiagok elkar estaltzen dutenean, deskribatutako algoritmoa amaierarik gabeko zikloan sar daiteke, hau da, behin eta berriz ordenatzen saiatuko litzateke. Horrelako bigiztak ekiditeko, ordenatutako poligonoa markatu egin beharko genuke; horrela, dagoeneko markatuta dagoen poligonoa berriz ordenatu behar bada, bi zatitan banatuko genuke eta poligono markatua bere bi azpipoligonoek ordezkatuko lukete.

7.1.4 Ray-casting

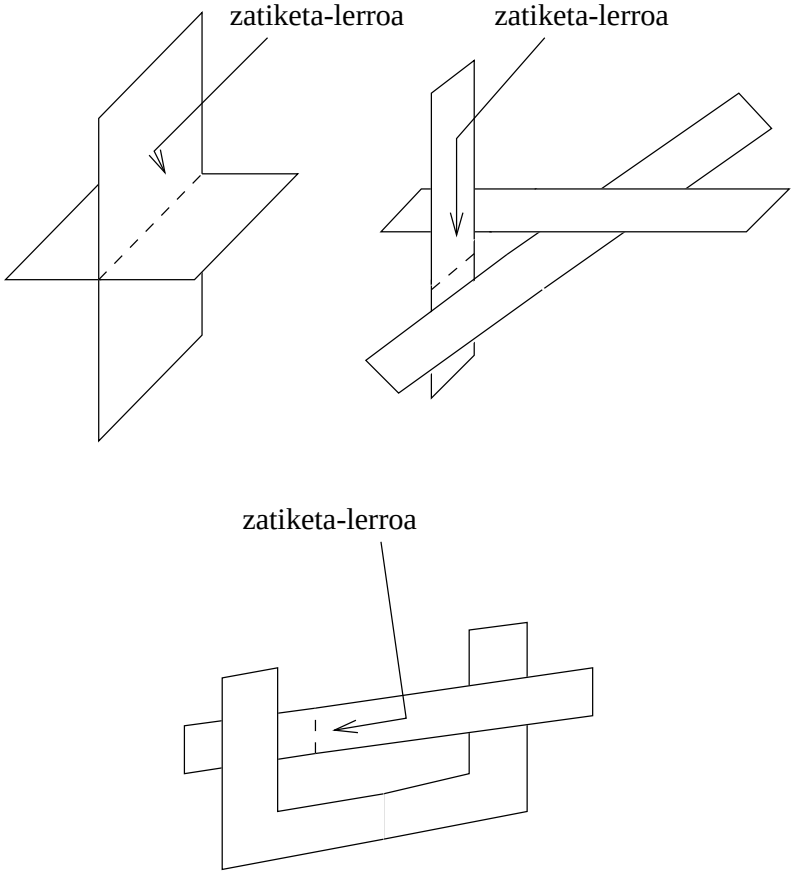
Ray-casting izenez ezagutzen den algoritmoan, ikuslearengandik pixel bakoitzera doan izpiak, zein objekturekin egiten duen talka aztertuz, ikuslea-



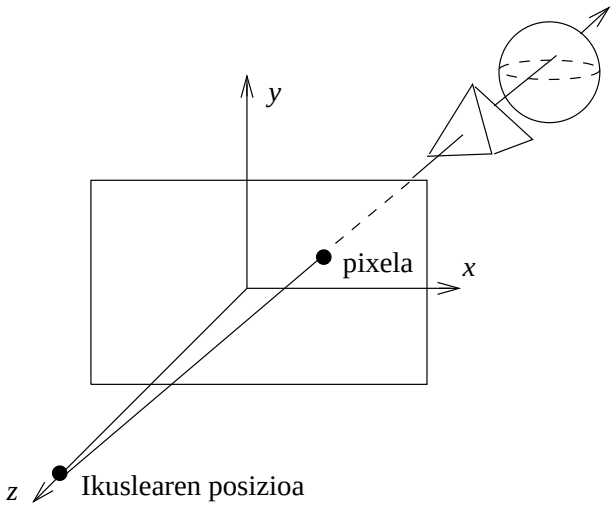
Irudia 7.10: Ordena trukatu beharko litzaieke.



Irudia 7.11: Hiru poligonoen ordena bi aldiz aldatu beharko litzateke.



Irudia 7.12: Kasu horietan ordena-aldaketek ez dute arazoa konpontzen; zatitu egin beharko lirateke.



Irudia 7.13: Ikuslearen tokitik proiekzio-planoko puntuetarako izpiek zein objektu ebakitzen duten zehaztu behar da Ray-casting metodoan.

rentzat objektu ikuskorrak zein diren kalkulatzen da. Pixel bakoitza zeharkatzen duen izpiak talka zein objekturekin egiten duen kalkulatu ondoren, puntu ikuskorra ikuslearengandik gertuen egongo den talka-puntua izango da. Talka-puntuan argiztapen-eredua aplikatu ondoren, kalkulatutako intentsitatea egokituko zaio pixelari. Ondoren, hurrengo pixelaren gain prozesu bera egingo dugu. Ray-casting metodoaren exekuzioa, deskribatutako prozesua irudiko pixel guztien gain aplikatu ondoren amaituko da.

Ray-casting metodoa izpi-hedaketaren kasu partikularra da. Algoritmo hori eragitean pixel batetik gertueneko objektura doan izpia erabiltzen dugu, objektu horretatik errefraktatzen eta isladatzen diren izpiak aztertu gabe. Izpi errefraktatua eta isladatua kontsideratzen ez ditugunez, Ray-casting algoritmoak ez du izpi-hedaketaren algoritmoaren izaera errekurtsiboa izango.

Kapitulua 8

Argiztatze-ereduak

8.1 Sarrera

Errealitate-itxurako irudiak lortzeko, objektuen perspektiba-proiekzioa kalkulatu, eta ikuskorrak diren gainazaletan argiztatze naturalaren efektuak aplikatzen dira.

Iluminazio- edo argiztatze-eredua, objektu baten gainazaleko puntu batean ikusi beharko genukeen argi-intentsitatea kalkulatzeko erabiltzen da. Eta errealitate-itxura emateko algoritmoek, argiztatze-ereduaren arabera kalkulatutako argi-intentsitateak erabiltzen dituzte, irudiko pixel bakoitzaren argi-intentsitate egokia zein den erabakitzeko. Batzuetan itzaleztatze-eredua aipatzen da argiztatze-eredua esan nahi denean, eta liburu honetan biek esanahi bera izango dute.

Zenbait liburutan, gainazalei errealitate-itxura emateko prozesua esan beharrean, gainazal-itzaleztatzea aipatzen da, eta alderantziz. Nahasteak ekiditeko, gainazal bateko puntu baten argi-intentsitatea kalkulatzen duen ereduari argiztatze-eredua deituko diogu, eta eszena bateko gainazal proiektatuen pixel bakoitzeko argiztatze-eredua aplikatzen duen prozesutik bereizi egingo dugu.

Ordenadore bidezko irudigintzan fotorrealismoak bi elementu eskatzen ditu:

- Objektuen adierazpen zehatza.
- Argiaren efektu fisikoen deskribapen ona.

Argiztatze-ereduaren barnean, argi-isladak, gardentasuna, itzalaren efektuak eta gainazalen ehundura modelatu behar dira.

8.2 Oinarrizko ereduak

Ikus ditzagun argi-intentsitatea kalkulatzeko erabiltzen diren metodo sinplifikatuak; azalduko ditugun eredu enpirikoek, gainazal bateko puntu baten argi-intentsitatea era sinple eta azkarrean kalkulatzeko aukera emango digute. Argiztapen-kalkuluak, gainazalen propietate optikoen, argiaren baldintzen eta argi-iturrien espezifikazioaren arabera egiten dira. Parametro optikoak gainazalaren zenbait propietate finkatzeko erabiltzen dira, hala nola opakotasuna, gardentasuna edo gainazalaren distira ezaren modukoak. Horiek, gainazalera heltzen den eta bertan isladatu eta zurgatzen den argi--kantitatea kontrolatzen dute. Argi-iturri guztiak, kokapen eta intentsitate (kolore) bidez zehazten dira, eta kontrakorik esaten ez dugun bitartean, puntuzko argi-iturritzat hartuko ditugu; hau da, norabide guztietan argiztatzen dutela suposatuko dugu, gela baten erdian jarritako bonbila baten antzera. Hala ere, beste zenbait argi-iturri mota erabiltzeko bideak ere azalduko ditugu, garrantzitsuenak ondokoak izango dira:

- Puntuzko argi-iturria. Horrelakoek kokapena eta intentsitatea besterik ez dute edukiko.

- Kono-motako argi-iturria. Kokapena eta intentsitateaz gain norabidea ere zehaztu beharko zaie argi horiei; gainera, irekiera-angelua ere beharko dute, hau da, kono horren zabalera nola edo hala zehaztu beharko da.

- Norabide hutsezko argi-iturria. Argi horiek kokapenik ez dute behar, norabidea ezagutzearekin nahikoa izango da. Eguzkia, adibidez, horrelakoa da, eta objektu guztietara norabide berarekin heltzen da argia.

8.2.1 Ingurune-argia

Gure oinarrizko argiztatze-ereduan, distira-maila bat ezar dezakegu eszena osorako. Gainazal ezberdinen artean gertatzen diren isladak modelatzeko modu sinple bat da hori. Ingurune-argiak, ez dauka ez kokapen-

ez norabide-ezaugarririk. Objektu bakoitzera iristen den ingurune-argiaren kantitatea konstantea da puntu guztientzat, edozein dela ere beren kokapena.

Eszenako ingurune-argiaren balioa finkatzeko I_a parametroa erabil dezakegu, eta gainazal bakoitzak, gutxienez, argi-kantitate konstante hori jasoko du.

8.2.2 Islada barreiatua

Islada barreiatua konstantea da gainazal osoan: ikuste-norabidearekiko independentea da. Gainazal bakoitzak, heltzen zaion argi-kantitatearen zati bat norabide guztietan barreiatzen du eta hori K_d konstantearekin adierazten da, hau da, islada-barreiapenaren koefizientearekin. K_d 0 eta 1 artean dagoen balio konstantea da. Gainazalak argi asko islada dezan, batetik gertuko balioa beharko du. Horrek, distira handiko gainazal bat emango luke, eta gainazalak barreiatutako isladaren intentsitatea, heltzen den argiaren intentsitatetik gertu egongo litzateke. Argi asko zurgatzen duen gainazalaren K_d koefizientea zerotik gertu dago.

Gainazalak ingurune-argia bakarrik jasoko balu, edozein puntuk barreiatuko lukeen argiaren intentsitatea honakoa izango litzateke:

$$I_{\text{ingbarr}} = K_d I_a$$

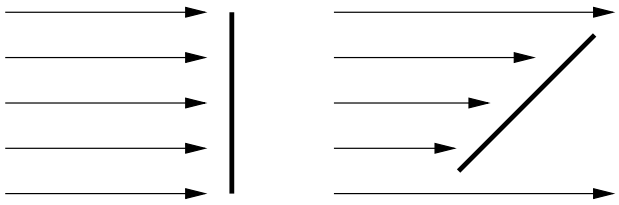
Baina ingurune-argia besterik ez daukan eszenarekin lortzen diren irudiak ez dira oso onak; horregatik, argi-iturri bat bada ere beti jartzen da.

Islada barreiatua modelatzerakoan, ezin dugu ahaztu, gainazal baten distira gainazalak argiarekiko duen orientazioaren menpe dagoela. Argiarekiko elkarzuta den gainazalak, argiarekiko elkarzuta ez den beste edozein gainazalek baino distira handiagoa izango du, 8.1 irudian ikus daitekeen bezala.

Erasotze-angelua α bada, gainazalak jasotako argi-kantitatea $\cos(\alpha)$ balioarekiko proportzionala izango da. Argi-izpi bat gainazaleko puntu batekiko elkarzuta bada, puntu horrek argi-kantitate maximoa jasoko du. Argiztatze-norabidea gainazaleko normalaren norabidetik urrunduz doan heinean, gainazalera heltzen den argi-kantitatea jaitsiz doa. Argi-iturriaren intentsitatea I_l izanik, puntu bateko islada barreiatua ondoko ekuazioaz kalkula daiteke:

$$I_{l,\text{barr}} = K_d I_l \cos(\alpha)$$

Gainazal batek, erasotze-angelua 0 gradu eta 90 graduren tartean badago bakarrik jasoko du argia. Argi-iturria gainazalaren atzean dagoenean, $\cos(\alpha)$ balioa negatiboa izango da.



Irudia 8.1: Argi-izpi erasotzaileekiko elkarzuta den gainazalak etzandakoak baino argi gehiago jasoko du.

N delakoa gainazalaren bektore normal unitarioa eta L delakoa puntuzko argi-iturritik gainazalera doan bektore unitarioa badira, $\cos(\alpha) = NL$ da. Aurreko ekuazioa berridatziz:

$$I_{l,\text{barr}} = K_d I_l \ (N \ L)$$

Beraz, orain artekoa ikusita, islada barreiatu osoa ematen digun ekuazioa idatz dezakegu:

$$I_{\text{barr}} = K_a I_a + K_d I_l \ (N \ L)$$

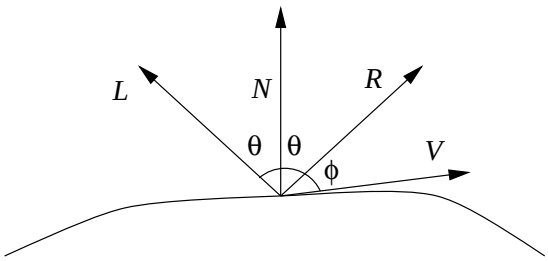
K_a delakoa ingurune-argiaren islada koefizientea da. Parametro berri horren bidez gainazal bakoitzak ingurune-argi gehiago edo gutxiago islada dezake, Dena den, gehienetan K_d koefizientearen berdina izaten da.

8.2.3 Ispilu-islada: Phong-en eredua

Argiztatuta dagoen gainazal distiratsu bati begiratzen badiogu (sagar bati, bilarreko bola bati ...), norabide batzuetatik puntu distiratsu edo puntu-ingurune distiratsu bat ikus dezakegu. Fenomeno horri ispilu-islada deitzen zaio.

Azter dezagun 8.2 irudia.

1. R, ispilu-islada idealaren norabidea duen bektore unitarioa da.
2. L, gainazal puntutik puntuzko argi-iturrirantz doan bektore unitarioa da.
3. V, gainazal puntutik ikuslerantz doan bektore unitarioa.



Irudia 8.2: Erasotze-norabidearen eta normalaren arteko angelua, normalaren eta ispilu-isladaren norabidearen artekoaren berdina da.

4. N, gainazalarekiko elkarzuta den bektore unitarioa.

5. ϕ angelua R eta V bektoreek osatzen duten angelua da.

6. θ angelua N eta R bektoreek osatutako angelua da.

Isladatzaile ideal batek (ispilu batek), argi erasotzailea R norabidean besterik ez du isladatzen. Egoera horretan, gainazaleko puntuan isladatutako argia ikusi ahal izateko, V eta R norabideek berdinak izan behar dute, hau da, bi bektoreen arteko angeluak 0 izan behar du.

Objektua isladatzaile ideala ez bada, ispilu-islada R bektorearen inguruko kono-itxurako bolumenean kokatutako edozein puntutatik ikus daiteke; zenbat eta isladatzaile kaskarragoa izan kono hainbat eta zabalagoa izango da.

Ispilu-islada Phong-en ereduaren bitartez modela daiteke. Eredu horretan ispilu-isladaren intentsitatea $\cos^{ns}(\phi)$-rekiko proportzionala da. ϕ angeluak V eta R bektoreen arteko angelua adierazten du eta 0 eta 90 graduen arteko balioak eduki behar ditu, bestela ezingo baita ispilu-isladarik ikusi. Bestalde, ns ispilu-isladaren parametroaren arabera, gainazal ezberdinen izaera kontrola dezakegu. Oso distiratsua den gainazal bat lortzeko, ns-ri balio altua egokitu beharko zaio (100 edo gehiago). Balio txikiekin ispilu-islada ikus daitekeen ingurua zabaldu egiten da, baina ispilu-islada teorikoaren norabidetik urrundu ahala, isladatutako intentsitatea txikituz doa; azken finean, ns parametroak txikitze-abiadura kontrolatzen du, edo beste era batera esateko, ispilu-isladaren barreiapena kontrolatzen du. Ondorioz, ispilu baten kasuan ns-k infinitua izan beharko luke, norabide teorikotik ateraz gero, ez bailitzateke isladarik ikusiko. Esandakoaren arabera, orokorrean, ispilu-islada

kalkulatzeko ekuazioa

$$I_{\text{isp}} = K_s I_l \cos^{ns}(\phi)$$

izango litzateke; baina zenbait materialen izaera berezia simulatzeko, K_s parametro konstantearen ordez, W(ω) funtzioa erabil daiteke. Funtzio horrek ω angeluaren arabera, hau da, V eta N bektoreen arteko angeluaren arabera, 0 eta 1 arteko balioak itzuliko ditu eta funtzio hori erabiliz Phong-en ispilu-isladaren eredua

$$I_{\text{isp}} = W(\omega) I_l \cos^{ns}(\phi)$$

ekuazioak adieraziko luke, bertan I_l aldagaiak argi-iturriaren intentsitatea adierazten duelarik.

W(ω) funtzioaren erabileratako bat, berez isladatzaile ez diren gainazalentzat izan daiteke. Kasu askotan, ω angelua handia denean (90 gradu inguru), gainazalak islada sortzen du, nahiz eta berez isladatzailea ez izan. Horren adibide garbia errepidea izan daiteke. Berez ez da isladatzailea, baina oso luzea eta zuzena bada, urrutian dauden gauzak isladatu egiten ditu. Horrelako zerbait lortzeko, errepideari ns parametro handia eman behar diogu, islada egon dadin; baina W(ω) funtzioak 0 itzuli behar du, baldin eta $\omega \ll 90$ bada. Alderantziz, $\omega \approx 90$ bada, 1 balioa itzuliko du, eta horrela, islada berezi hori lortu ahal izango genuke.

V eta R bektoreak unitarioak direnez, cos(ϕ) bi bektoreen arteko biderkaketa eskalarraren bidez kalkula dezakegu. Ispilu-isladaren koefizientea konstantea dela suposatuz, ispilu-islada gainazal bateko puntu batean ondoko formularen bidez kalkula dezakegu:

$$I_{\text{isp}} = K_s I_l \ (V \ R)^{ns}$$

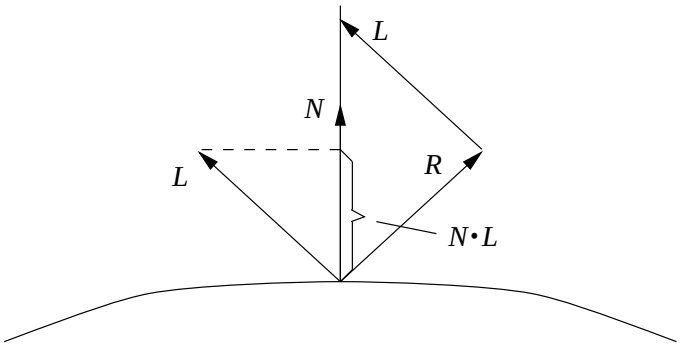
Baina nola kalkulatu R bektorearen norabidea? Ondoko eragiketak errazago ulertzeko, begira ezazu 8.3 irudia. L eta N bektore ezagunak erabiliz lortuko dugu R. Bestalde, L bektoreak N bektorean duen proiekzioa, $N \ L$ biderkaketa eskalarraren bidez lor daiteke. Beraz:

$$R + L = (2N \ L)N$$

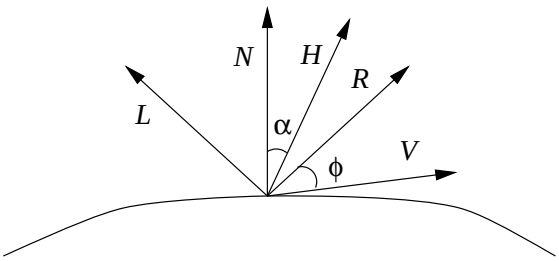
R askatuz:

$$R = (2N \ L)N - L$$

Erdibideko bektorea: L eta V bektoreen artean dagoen erdibideko bektorea erabiliz, H erabiliz, Phong-en eredu sinpleago bat lor daiteke. V



Irudia 8.3: R bektorearen kalkulua.



Irudia 8.4: H bektorea.

eta R bektoreen arteko angeluaren ordez H eta N bektoreen artekoa hartzen badugu, 8.4 irudian agertzen den $\cos(\phi)$ kalkulatu beharrean, irudi bereko $\cos(\alpha)$ balioa kalkulatuko dugu. Bektore berri hori kalkulatzeko, L eta V bektoreak batu behar dira:

$$H = \frac{(L+V)}{|L+V|}$$

Ikuslea eta argi-iturria gainazaletik nahiko urrun badaude, bai V eta bai L gainazalean konstanteak dira, eta beraz H ere konstantea da gainazaleko puntu guztietan. Planoak ez diren gainazaletan, N H eragiketa kalkulatzeak V R kalkulatzeak baino konputazio-kostu txikiagoa dauka, R kalkulatzean gainazaleko edozein puntutan aldakorra izango den N bektore normala erabili beharko baikenuke.

H bektoreak, ikuste-norabidean ispilu-islada handiena izango duen gainazalaren norabidea adierazten du. Bestalde, V bektorea L eta R bektoreek

definitzen duten plano berean badago,(eta beraz N ere bai), α angeluak $\frac{\phi}{2}$ balioa izango du. V, L eta N plano berean ez daudenean, $\alpha > \frac{\phi}{2}$ izango da eta α-ren balioa hiru bektoreen norabideen menpe egongo da.

8.3 Islada argi-iturri anitzekin

Puntuzko argi-iturri bakar bat badaukagu, gainazal bateko puntu bateko argi-intentsitatea ingurune-argia, islada barreiatua eta ispilu-islada konbinatuz lor dezakegu:

$$I = I_{\text{barr}} + I_{\text{isp}} = K_a I_a + K_d I_l \ (N \ L) + K_s I_l \ (N \ H)^{ns}$$

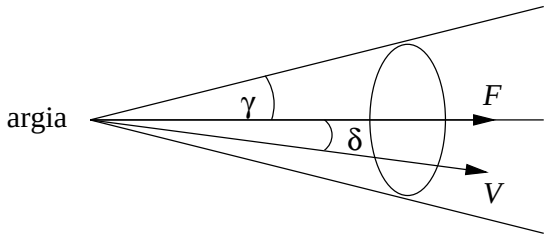
Baina eszenan puntuzko argi-iturri bat baino gehiago badago, puntu bateko argi-intentsitatea kalkulatzean, argi guztien eragina eduki beharko dugu kontuan:

$$I = K_a I_a + \sum_{i=1}^{n} I_{li}[K_d(N \ L_i) + K_s(N \ H_i)^{ns}]$$

Pixel bakoitzeko intentsitatea adieraz dezakegun intentsitate maximoa baino handiagoa ez dela ziurtatzeko, nolabaiteko normalizazio-prozedura bat aplika dezakegu. Prozedura sinple batek, argi-intentsitatearen osagairen batek (gorriak berdeak edo urdinak) balio maximoa gainditukо balu, osagai horren balioa kendu eta balio maximoa jarriko lioke. Beste normalizazio posible batek, intentsitate-osagaien balio handiena hartu eta osagai guztiak balio horregatik zatituko lituzke. Normalizazio-prozedura konplexuago batean eszenako pixel guztien kolorea kalkulatuko litzateke, gero lortutako intentsitateak onargarria den kolore-intentsitateen eremuarekin eskalatzeko.

8.4 Warn-en eredua

Errealitateko argi-iturriek ez dute norabide guztietan berdin argiztatzen. Warn-ek argien norabidea kontrolatzeko asmatu eta argazkilariek izaten dituzten tresnak, edozein argiztatze ekuaziori modu errazean gehitzeko bidea garatu zuen. Phong-en ereduan, edozein argi-iturrik kokapena eta intentsitatea du eta norabide guztietan argiztatzen du. Warn-en ereduaren bidez norabide bakar batean argiztatzen duten argiak lor ditzakegu. Argi horien



Irudia 8.5: Kono-motako argi-iturria.

eragina eszenako eremu batera mugatzeko, Warn-ek hegalak eta konoak inplementatu zituen.

Hegalak erabiltzean, eszenako laukizuzen bat argiztatzeko posibilitatea dugu. Argi bakoitzak sei flap edo hegal izango ditu, ardatz bakoitzeko balio minimo eta maximoen bidez definituak. Flap bakoitzari, boolearra izango den balioa ere egokituko zaio, aktibatuta dagoen ala ez jakiteko. Beraz, hegalak dituen argi baterako, puntu batek jasotzen duen argi-intentsitatea kalkulatzerakoan, argiztatze-eredua aktibaturik dauden hegalek deskribatzen duten eremuaren barnean dagoenean bakarrik aplikatzen da.

Argi batek eszenako zein eremu argiztatzen duen definitzerakoan, konoak erabiltzeko era ere aurkitu zuen Warn-ek. Konoaren muturra argi-iturria kokatzen den tokian egongo litzateke, eta bere ardatzak, konoaren puntatik oinarriko zirkunferentziaren erdiko puntura doanak, F norabidea jarraituko luke. 8.5 irudian ikus daitekeen bezala, argi iturrian hasi eta F ardatzaren inguruan γ angeluak mugatzen duen bolumena izango da konoa. Horrela definitutako konoa erabiltzean, eszenako puntu bat konoaren bolumenaren barnekoa den ala ez jakin behar dugu, barnean badago argia jasoko baitu baina kanpokoa bada ez. Hori jakiteko, argi-iturritik puntura doan bektorearen eta F bektorearen arteko angelua ezagutu behar dugu. Angelu hori, δ angelua, γ baino txikiagoa bada, $\cos(\delta) > \cos(\gamma)$ izango da, eta kasu horretan argi-iturriak puntu horri bidaltzen dion argia hartu beharko dugu kontuan.

Kono-motako argi-iturriek F ardatzaren inguruan argi gehiago eman dezakete. Hau da, nahiz eta F ardatzaren inguruan γ angeluak mugatutako eremu osoa argitu, ardatzetik gertuko tokietan urrunxeagokoetan baino argi gehiago eman dezakete, eta fenomeno hori kontuan hartu nahi izango bagenu, ezin ahal izango genuke I_l besterik gabe erabili; ardatzetik urrundu ahala, argi-iturritik jasotako argi-kantitatea gutxitu beharko genuke, eta hori F eta

V bektoreen arteko angeluaren arabera egin beharko genuke:

$$I_{\text{konoa}} = I_l(-V\ F)^p$$

p parametroak argia F ardatzaren inguruan banatzeko balio du. Parametro horrekin jokatuz, eszenako eremu batek jasotzen duen argi-intentsitatea alda daiteke. p-ren balioa zenbat eta handiagoa izan, argia orduan eta gehiago kontzentratzen da F bektorearen inguruan. Ondorioz, p-ri balio handiak emanez, F bektorearen inguruan dagoen eremu mugatu batean argi kontzentrazio handia lortuko genuke, eta balio txikiak egokituz, F inguruko eremu zabalagoan banatuko luke argia. p parametroa 0 balitz, argi-iturri arrunta izango genuke, hau da, leku guztietan berdin argiztatuko luke.

Kono-motako argi-iturriek, beraz, p parametroa, γ angelua eta F bektorea behar dituzte. Dena den, γ angelurik gabe ere erabil daitezke, baina horrela, F bektorearekiko 90 gradu edo gutxiagora dagoen edozein puntuk argia jasoko luke, nahiz eta p parametroaren bidez jasotako argi-kantitatea mugatzeko aukera izan.

8.5 Intentsitatearen ahuldura

Orain arte garatutako argiztatze-ereduan, gainazal bateko puntu bat argiztatzean puntua argi-iturritik zein distantziatara dagoen ez dugu kontuan hartu. Argiztatze-eredu hori ez da oso errealista. Errealitatean ez da berdina gainazal bat metro batera edo 200 metrotara dagoen argi-iturri batekin argiztatzea. Beraz, argiztatze-eredu sinesgarriak sortzeko, intentsitateak distantzian zehar jasaten duen ahuldura kontuan eduki beharko genuke.

Pentsa dezagun zer gertatuko litzatekeen horrela egingo ez bagenu. Har ditzagun bi gainazal, biak propietate optiko berdinekin, baina bat bestea baino urrunago eta eskuinerago. Kasu horretan bi gainazaletarako intentsitate berdina kalkulatuko genuke eta ondorioz ezingo genuke bien arteko muga bereizi.

Intentsitatea ahultzeko, $\frac{1}{d^2}$ funtzioa erabil dezakegu argiztatze-ereduan. Hala ere funtzio hori ez da oso egokia helburua lortzeko. Distantzia txikietarako, $\frac{1}{d^2}$ funtzioak aldaketa handiak sortuko lituzke, eta distantzia handietarako ia ez genuke intentsitate-aldaketarik nabariko.

Arazo horiek ekiditeko ondoko ahuldura-funtzioa erabil dezakegu:

$$f(d) = \frac{1}{(a_0 + a_1 d + a_2 d^2)}$$

Funtzio horretan, a_0 balio egokia aukeratuz, d oso txikia denean $f(d)$ oso handia izatea ekidin daiteke. Beste koefizienteen balio egokia eta gainazalen propietate optiko egokiak ere aukeratu beharko dira, batez ere, isladatutako intentsitateak onargarria den maximoa gainditu ez dezan.

Ondoko ahuldura-funtzioa erabiliz, $f(d)$-ren balioa 1 baino handiagoa izango ez dela ziurta daiteke:

$$f(d) = \min(1, \frac{1}{(a_0 + a_1 d + a_2 d^2)})$$

Funtzio hori oinarrizko argiztatze-ereduan aplikatuz:

$$I = K_a I_a + \sum_{i=1}^{n} f(d_i) I_{li}[K_d(NL_i) + K_s(NH_i)^{ns}] \tag{8.1}$$

non, d_i argi-iturritik gainazaleko puntura dagoen distantzia den.

8.6 Koloreari dagozkion zuzenketak

Gure argiztatze-ereduko formulak aplikatuz, kolore bakarreko irudiak lortuko ditugu. Lortutako irudietan koloreak ager daitezen argi-iturri eta gainazal bakoitzari kolorea esleitu beharko diogu. Eta esleitutako kolorea adierazteko erabil dezakegun adierazpidetako bat, RGB adierazpidea da. Adierazpide horretan kolore bat deskribatzeko kolorea osatzen duten gorri-, berde- eta urdin-kantitatea adierazten da. Kolore zuriaren kasuan hirurak balio maximoa izango lukete; kolore beltzarenean berriz, minimoa.

Argi-iturri bati kolorea esleitzeko, nahikoa da RGB (Red, Green, Blue) bektorea izango den atributua gehitzea. Gainazal bateko puntu batek isladatzen duen kolorea kalkulatzeko, RGB bektoreko osagai bakoitzari dagokion intentsitatea lortu behar da. Horretarako, argiztatze-ereduko koefiziente bakoitza hiru osagaiko bektore bihurtu behar dugu, eta osagai bakoitzeko intentsitatearen kalkuluak egin beharko ditugu. Argiztatze-ereduko formuletara itzuliz, gainazal batek izango lukeen kolore urdinaren osagaia kalkulatzeko:

$$I_B = K_{aB} I_{aB} + \sum_{i=1}^{n} f(d_i) I_{lBi}[K_{dB}(NL_i) + K_{sB}(NH_i)^{ns}]$$

Kalkulu bera egin beharko litzateke kolorearen beste bi osagaientzat. Bestalde, K_s bektore moduan jarri beharrean konstante bezala uzten badugu,

objektuaren ispilu-isladaren kolorea iristen zaion argiarenaren berdina izango da, plastikozkoen moduko objektuak lortuz. Argi-iturria zuria balitz, objektuaren ispilu-islada zuria izango litzateke.

Beste hurbilketa bat K_a,K_d eta K_s konstanteak kendu gabe (S_{dr},S_{dg},S_{db}) eta (S_{sr},S_{sg},S_{sb}) kolore-bektoreak txertatzean datza. Bigarren hurbilketa hori erabiliz:

$$I_B = K_a S_{dB} I_{aB} + \sum_{i=1}^{n} f(d_i) I_{lBi} [K_d S_{dB}(NL_i) + K_s S_{sB}(NH_i)^{ns}]$$

Bigarren hurbilketaren abantaila malgutasunean datza. Hurbilketa horretan, gainazalaren kolorea bere islada-koefizienteekiko independenteki finka baitaiteke.

8.7 Gardentasuna

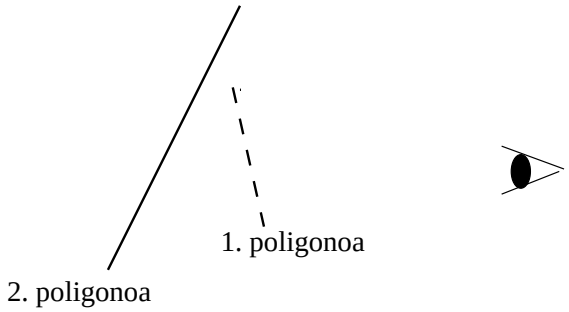
Objektuetan zehar igarotzen den argiak eszenan izango duen eragina, gainazalaren gardentasun-maila eta gainazal gardenaren atzean egon daitezkeen argi-iturri edo argiztatutako gainazalen menpe dago.

Gainazal garden bat modelatzean, intentsitate-ekuazioek gainazala zeharkatzen duen argia kontuan eduki beharko dute. Gehienetan objektua igarotzen duen argia gainazalaren atzean dauden objektuek isladatutakoa izaten da eta objektu gardenaren gainazaleko intentsitatean eragina du.

Gainazal gehienek izan dezakete ispilu-islada eta islada barreiatua. Eta argia transmititzen dutenak, gardenak edo argi eroaleak izan daitezke.

Gardenak diren materialen atzekoa ikusteko gai izan behar dugu, baina materiala zeharkatzen duten izpiak errefraktatuak izan daitezkeela ere hartu behar da kontuan.

Islada barreiatuaren transmisioa, argia igarotzen uzten duten materialetan ematen da. Material horiek zeharkatzen dituzten izpien norabidea okertu edo aldatu egiten da, eta ondorioz material horietan zehar ikusten dena, irudi zirriborrotsu modura agertzen da. Errefrakzio barreiatua lortzeko, errefraktatutako argiaren intentsitatea gutxitu eta errefrakzioa sortzen duen gainazalaren azalera finitu bateko puntu bakoitzaren intentsitatea zabalduz lortzen da. Manipulazio horiek konputazio-denbora gehiegi eska dezakete.



Irudia 8.6: Bi poligonoren irudia, 1. poligonoa gardena izanik.

8.7.1 Errefrakziorik gabeko gardentasuna

Gardentasuna modelatzeko eredu sinpleenak alde batera uzten du errefrakzioa, bere eragina kontuan izan gabe. Beraz, argi-izpi batek gainazal bat zeharkatzean, norabide-aldaketarik jasaten ez duela suposatzen da. Nahiz eta errefrakziorik gabeko gardentasuna errealista ez izan, askotan errefrakzioa baino erabilgarriagoa da, gainazal baten atzean dagoena distortsiorik gabe ikusi nahi dugunean batez ere.

Objektu bat beste batean zehar ikus daitekeenean, bien koloreak konbinatzeko bi bide jarraitu ohi dira. Bata gardentasun interpolatua da eta bestea gardentasun iragazia.

Gardentasun interpolatua. Pentsa dezagun zer gertatzen den gardena den 1. poligonoa, ikuslearen eta gardena ez den 2. poligonoaren artean dagoenean, 8.6 irudian bezala.

Gardentasun interpolatuaren bidez, bi poligonoen proiekzioen ebaketa-puntu den pixel bateko intentsitatea kalkulatzeko, bi poligonoen intentsitatea (poligono bakoitzak bere intentsitatea izango du) interpolatzen da:

$$I = (1 - K_t)I_1 + K_t I_2$$

$(1 - K_t)$ balioak poligonoaren opakotasuna adierazten du. K_t transmisio koefizienteak 1 poligonoaren gardentasuna neurtzen du, eta bere balioa 0 eta 1 artean egongo da. K_t koefizientea 0 denean, poligonoa guztiz opakoa da eta bigarren poligonoak ez du inongo eraginik I-ren balioan.

Gardentasun interpolatua beste era batera ere uler daiteke: opakoa den sare mehe batekin lan egitea bezalakoa da, poligonoa sare opako eta mehe

bat izanik, eta bertan zehar beste objektu batzuk ikus daitezke. Horrela, K_t delakoa, gainazalean zehar ikus daitekeen frakzioa izango litzateke.

Guztiz gardena den eta era horretan modelatzen den poligono batek ez luke ispilu-isladarik izango. Efektu sinesgarriagoa lortzearren, 1. poligonoaren islada barreiatu eta ingurune-argiaren isladaren osagaiak interpola ditzakegu 2. poligonoari dagokion intentsitate osoarekin, eta ondoren 1. poligonoaren ispilu-islada gehitu lortutako emaitzari.

Errefrakziorik gabeko gardentasunaren beste hurbilketa bat ere badago, marrazgune gardentasuna deritzona. Lehen aipatutako sarearen ideia jarraitzen du. Gardena den poligono bat izanik, pixel bakoitzeko intentsitatea erabakitzean, maskara bat erabiliko da, poligonoaren puntu bakoitzeko 1 edo 0 balioa izango du. 1 balioa badu, poligono gardenaren intentsitatea esleitzen zaio pixelari, eta 0 bada, atzean dagoenaren intentsitatea esleitzen zaio. Beraz, maskararen bidez, 1ak eta 0ak toki egokietan jarriz, poligonoaren gardentasun-sarea lor dezakegu. Hurbilketa horren arazo nagusia, gardenak diren zenbait poligono bata bestearen atzean jartzean azaltzen da. Kasu horretan, poligono gardenen atzean dagoenaren ezer gutxi ikusiko genuke. Arazo hori objektu desberdinen maskaren elkarreraginagatik ematen da.

Gardentasun iragazia. Objektu batek argi guztiekin berdin jokatzen ez duenean, aurreko hurbilketak nekez lor dezake emaitza errealista. Objektuak, berari dagokion intentsitateaz gain, atzekoaren osagairen bat edo beste ikusten laga diezaguke, baina ez guztiak. Hau da, kolore batzuek poligonoa zeharka dezakete baina besteek ez:

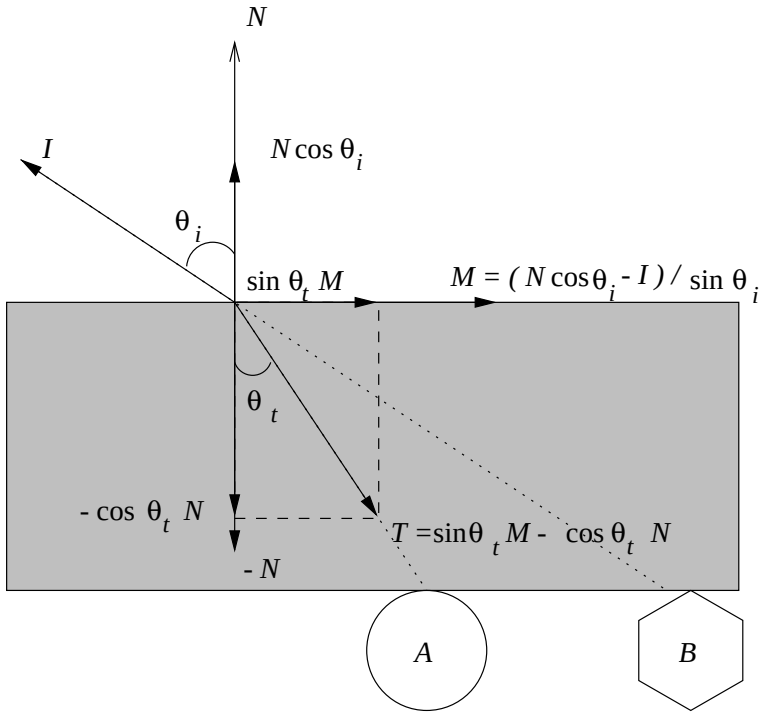
$$I = I_1 + K_t O_{t\lambda} I_2$$

non, $O_{t\lambda}$ 1 poligonoaren gardentasun kolorea den. Iragazki horrek kolore bakarra transmitituko luke. Kolore asko transmititzeko, λ bakoitzeko $O_{t\lambda}$-ri balioa emango genioke.

Bai gardentasun interpolatuan eta bai gardentasun iragazian, gardenak diren zenbait poligono bata bestearen aurrean edo atzean badaude, kalkuluak atzetik aurrera egingo lirateke, errekurtsiboki, pauso batean lortutako I balioa hurrengo pausoko I_2 izatera pasatuz.

8.7.2 Errefrakziodun gardentasuna

Errefrakziodun gardentasuna modelatzea, errefrakzio gabekoa baino konplexuagoa da, pixel batera heldutako izpiak ez baitu marra zuzen bati dagokion bidea egin. 8.7 irudian, errefrakziorik ez balego objektu gardenean zehar



Irudia 8.7: Errefrakzioak argi-izpien norabidean duen eragina.

B objektua ikusiko genuke; baina berez, errefrakzioa dela eta, A objektua ikusi beharko litzateke. Izpi erasotzailearen angeluaren eta errefraktatuaren angeluaren arteko erlazioa, hau da, θ_i eta θ_t angeluen arteko erlazioa, Snell-en legeak adierazten du:

$$\frac{\sin(\theta_i)}{\sin(\theta_t)} = \frac{\eta_t}{\eta_i}$$

non, η_i eta η_t direlakoak argiak zeharkatzen dituen materialen errefrakzio--indizeak diren. Material bati dagokion errefrakzio-indizea, argiak materialean duen abiaduraren eta hutsean duen abiaduraren arteko zatidura da. Balio horiek, errefrakzio-indizeek, aldaketak jasan ditzakete argiaren uhin-luzera ezberdinekin eta tenperatura ezberdinetan. Hutsaren errefrakzio-indizea 1.0 da eta material guztiek balio handiagoak dituzte.

Errefrakzio-bektorea kalkulatzen. Errefrakzio-norabidea duen bektore unitarioa kalkulatzeko:

$$T = \Big(\sin(\theta_t)M, -\cos(\theta_t)N \Big)$$

M delakoa, I izpi erasotzaileak eta N normalak osatzen duten planoan dagoen eta N-rekiko elkarzuta den bektore unitarioa da. R kalkulatzeko erabili genuen S bektorea gogoratuz, $M = (N\cos(\theta_i) - I)/\sin(\theta_i)$. Ordezkatuz:

$$T = \frac{\sin(\theta_t)}{\sin(\theta_i)}(N\cos(\theta_i) - I) - \cos(\theta_t)N$$

$\eta_r = \frac{\eta_i}{\eta_t} = \frac{\sin(\theta_t)}{\sin(\theta_i)}$ izanik

$$T = (\eta_r\cos(\theta_i) - \cos(\theta_t))N - \eta_r I$$

$\cos(\theta_i) = NI$ da eta $\cos(\theta_t)$ ondoko moduan kalkula daiteke:

$$cos(\theta_t) = \sqrt{1 - \sin^2(\theta_t)} - \sqrt{1 - \eta_r^2\sin^2(\theta_i)} = \sqrt{1 - \eta_r^2(1 - (N\ I)^2)}$$

Ondorioz:

$$T = \Big(\eta_r(N\ I) - \sqrt{1 - \eta_t^2(1 - (N\ I)^2)} \Big) N - \eta_t I \tag{8.2}$$

Erabateko barne-islada. Argia errefrakzio indize txikiagoa duen ingurune edo giro batera pasatzerakoan, errefrakzioa dela eta, izpiak jarraitzen duen norabide berriari dagokion θ_t, aurreko ingurunean zeraman norabidearen θ_i baino handiagoa da. θ_i behar adina handituz, θ_t angelua 90 gradu baino handiago izatea lor daiteke eta ondorioz, argi-izpia objektuaren barnean sartu beharrean haserako ingurunean zehar isladatuko da. Fenomeno horri erabateko barne-islada deitzen zaio, eta fenomeno hori gerta dadin, beharrezkoa den θ_i angelu minimoari, berriz, angelu kritikoa.

Barne-isladapen fenomenoa ikusteko, eskua urez beteriko arrain-ontzi baten atzean jar dezakegu. Behaketa angelua angelu kritikoa baino handiagoa denean, eskuaren zati ikuskor bakarrak urontzia ukitzen ari direnak dira, hau da, urontzi eta eskuaren artean airerik ez daukaten zatiak (airearen errefrakzio indizea beirarena edo urarena baino txikiagoa da). Angelu kritikoa θ_i-ren balioa bat da, $\sin(\theta_t)$ 1 izatera behartzen duena hain zuzen ere. Ekuazioan θ_t-ri 1 balioa emanda, angelu kritikoaren balioa $\sin^{-1}(\eta_t/\eta_i)$ dela ikus daiteke. 8.2 ekuazioaren erro karratua irudikaria izatean gertatzen da erabateko barne-islada.

8.8 Itzalak

Ezkutuko gainazalak lortzeko erabiltzen diren metodoak, argiak sortutako itzalak aurkitzeko erabil daitezke. Argi-iturri baten kokapena ikuslearen kokapen gisa hartuz, eta ezkutuko aurpegien ezabapen metodoren bat aplikatuz, argiak ikusi ezin dituen gainazalak identifikatuko genituzke. Horiek itzal-azalerak izango lirateke. Argi-iturri guztiek sortzen dituzten itzal-azalerak identifikatu ondoren, itzalak gainazal txantiloi modura trata daitezke eta txantiloi tauletan gorde.

Behin itzal-txantiloiak kalkulatuz gero, horiek ikuslearen edozein kokapenetarako erabil daitezke. Argi-iturrien kokapena aldatzen ez den bitartean behintzat.

Ikuskorrak diren gainazaletan argiztatze-eredua aplikatuko genuke eta gainazalaren ehundurarekin konbinatu. Itzaletako azaleretan ingurune argiaren eragina bakarrik eduki beharko genuke kontuan.

8.9 Poligonoak argiztatzeko teknikak

Puntu bateko argi-intentsitatea kalkulatzean, Phong-en ereduaren erabilera oso hedatuta dago ordenadore bidezko irudigintzan, eta batez ere poligono-sareen bidezko adierazpidea erabiltzen duten sistemetan. Poligono-sareen bidezko adierazpena duen objektuan, geometri informazioa erpinetan bakarrik agertzen da; hori dela eta, erpinei dagokien argi-intentsitatea baino ezin dugu kalkulatu. Poligonoaren barneko puntuei dagokien intentsitatea kalkulatzeko, erpinetako informazioa erabili behar dugu, horiei dagokien intentsitateak interpolatuz edo antzeko zerbait eginez. Barneko puntuak lerroz lerro kalkulatzen dira, eta horrela pixeletik pixelera informazioa pasa dezakegu. Horrela eginez, oso erraza da Z-buffer edo lerroko tartekako algoritmoetan interpolazioz poligonoen barneko pixelen intentsitatea kalkulatzea.

Batez ere bi teknika erabiltzen dira: Gouraud-en interpolazioa eta Phong-en interpolazioa. Bigarrenak ispilu-isladarekiko emaitza hobeak lortzen ditu eta horregatik erabiliagoa da, hala ere Gouraud-ena azkarragoa denez, elkarrekintzazko sistemetan bitarteko emaitzak lortzeko, aurreikusketak egiteko edo ispilu-islada gabeko emaitzak lortzeko erabil daiteke.

Bi teknikek poligonoen barne-puntuetako argi-intentsitatea ahalik eta modu eraginkorrenean kalkulatzen dute eta poligonoen arteko mugak ez antzemateko moduko emaitzak lortzen dituzte. Horrela, objektuari itxura

leuna ematea lortzen da, baina hala ere, objektuaren ingerada poligonoz osatuta dagoela ikus daiteke.

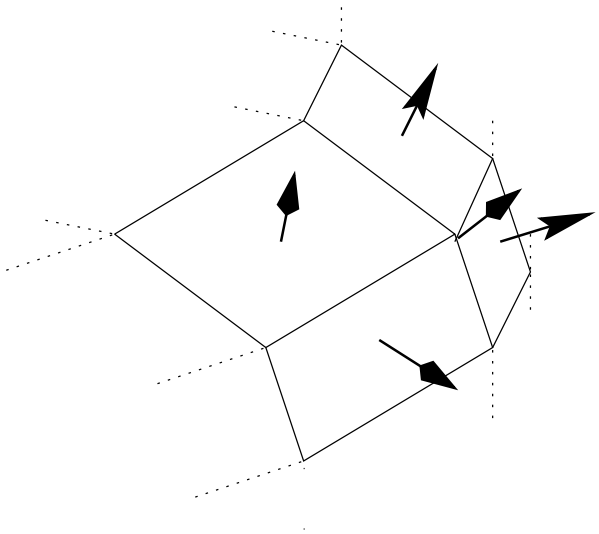
8.9.1 Gouraud-en interpolazioa

Gouraud izenez ezagutzen den teknikak, era azkar eta merkean, objektua osatzen duten poligonoak irudikatzerakoan objektu leuna osatuko balute bezala irudikatzea lortzen du. Horretarako, poligonoen ertzetako puntuen intentsitatea linealki aldatzen du; hau da, ertzaren bi erpinetako intentsitatea kalkulatu ondoren, tarteko puntuen intentsitatea bi erpinetako intentsitatea interpolatuz lortzen du. Ondoren poligonoaren barneko puntuetako intentsitatea kalkulatzeko, lerroz lerro egiten ditu kalkuluak, eta lerro bakoitzean, poligonoaren bi ertzen arteko pixelen intentsitatea ertz bakoitzean lortutako intentsitateak interpolatuz kalkulatzen du. Horrela, aldaketak leunak izatea lortzen da, eta gure begiek intentsitatearen lehenengo deribatuarekiko sentikorrak direnez, aldaketa leuna izatean, objektua leuna dela kontsideratzen dute. Hala ere objektuaren poligono-izaera ez du zeharo ezkutatzen, objektuaren ingeradan poligonoen ertzak ikusi egin ahal izango baititugu.

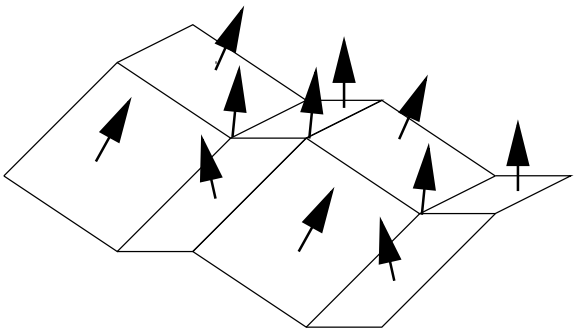
Azaldutakoaren arabera erpin bakoitzeko intentsitatea kalkulatu behar dugu, eta horretarako 8.1 ekuazioa erabili beharko dugu. Ekuazio horretan N bektorea azaltzen da, hau da, erpinak daukan bektore normala. Bektore hori kalkulatzeko, erpinaren inguruko poligonoak har ditzakegu eta poligono bakoitzaren planoaren bektore normalen arteko batezbestekoa erabili, 8.8 irudian ikus daitekeen bezala. Kontuan eduki behar da, bektore hori ez dela erpinaren benetako bektore normala, hurbilketa dela; baina objektua hartu bezain laster kalkula daiteke bektore hori, eta behin egindako kalkuluak, betiko balioko digu.

Hala ere arazotxo bat ager daiteke. Mugaketa dela-eta, zenbait erpin proiekzio planoko leihotik kanpo gera daitezke, eta ondorioz, leihoaren ertzetan erpin berriak kalkulatu beharra gerta dakiguke. Erpin berri horien bektore normala ere kalkulatu beharko dugu, eta horretarako zuzenena, erpin berria kokatu behar den ertzeko bi erpinen bektore normalak interpolatzea da, horrela, mugaketa dela-eta kanpoan gelditzen den zatia desagertuko ez balitz bezalako emaitza lortuko baita.

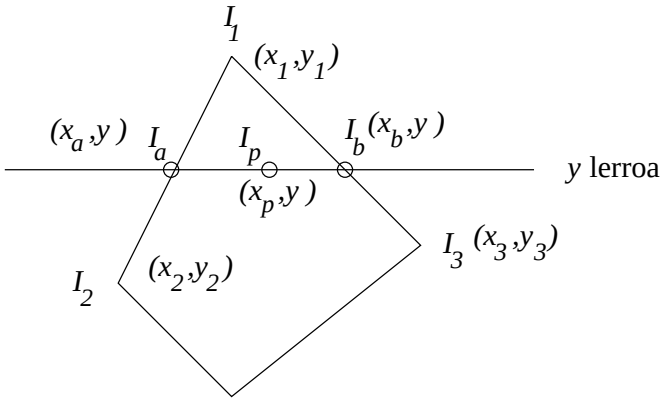
Erpinetako bektore normala inguruko poligonoen batezbesteko modura kalkulatzeak, beste arazorik ere sor dezake. Bektore hori era horretan kalkulatzean, gero objektuen zimurrei dagozkien bektoreak kalkulatzean emaitza txarrak lor daitezke, 8.9 irudian ikus daitekeen bezala. Kasu horretan objek-



Irudia 8.8: erpin baten bektore normala, aurpegien bektore normalaren arabera



Irudia 8.9: Erpinetako bektore normala kalkulatzean lor daitekeen emaitza txarra



Irudia 8.10: Erpinen intentsitatea ezagutuz poligonoen barneko puntuena interpola daiteke

tua laua balitz bezala argiztatuko genuke, irudiko poligonoen elkarguneetan lortzen diren bektoreak berdinak baitira eta ondorioz, poligonoen barnera intentsitate bera hedatuko baita.

Erpin bakoitzeko N bektorea ezagututa, beraien argi-intentsitatea kalkula dezakegu eta interpolazio prozesua lerroz lerroko algoritmoetan erraz barnera daiteke. Horretarako, 8.10 irudiko intentsitateak kalkulatzeko gai izan behar dugu eta ondoko interpolazioak egin beharko dira: Lerro bakoitzak ebakitzen duen ertz bakoitzeko ebaketa-puntuaren intentsitatea, ertz horren bi erpinen intentsitatearen arabera kalkulatu beharko dugu; hau da, 8.10 irudiko I_a eta I_b kalkulatzeko:

$$I_a = \frac{1}{y_1 - y_2}(I_1(y - y_2) + I_2(y_1 - y))$$

$$I_b = \frac{1}{y_1 - y_3}(I_1(y - y_3) + I_3(y_1 - y))$$

Era horretan y delakoa y_1-ren baliotik zenbat eta gehiago urrundu, hainbat eta eragin gutxiago edukiko du I_1 intentsitateak, $y - y_2$ eta $y - y_3$ balioak txikiagoak izango baitira; eta, alderantziz, I_2 eta I_3 intentsitateen eragina hainbat eta nabariagoa egingo da. Bi balio horiek kalkulatu ondoren, poligonoaren barneko puntuen intentsitatea kalkulatu beharko dugu, I_a eta I_b balioak interpolatuz:

$$I_p = \frac{1}{x_b - x_a}(I_a(x_b - x) + I_b(x - x_a))$$

Horrela I_a eta I_b intentsitateen arteko interpolazioa egiten dugu, eta x delakoa x_a-ren baliotik urrundu ahala, I_a intentsitatearen eragina txikituz doa I_b intentsitatea handituz doan heinean, $x_b - x$ txikitu egiten baita eta $x - x_a$ handitu.

Eraginkortasuna handitzeko pixel bakoitzeko kalkulu horiek guztiak egin beharrean, intentsitatearen kalkulua gehikuntzaz egitea komeni da. Lerro batean intentsitate-aldaketa ezaguna denez, $\Delta I = I_b - I_a$, x-ren aldaketaren arabera intentsitatea zenbat aldatzen den jakin dezakegu; hau da, intentsitate-aldaketa x unitate bakarrean aldatzen bada, $\Delta I_u = \frac{\Delta I}{\Delta x} = \frac{I_b - I_a}{x_b - x_a}$ izango da, eta x jakin bati dagokion intentsitatea ezaguna bada, $x+1$ puntuan intentsitatea honako hau izango da:

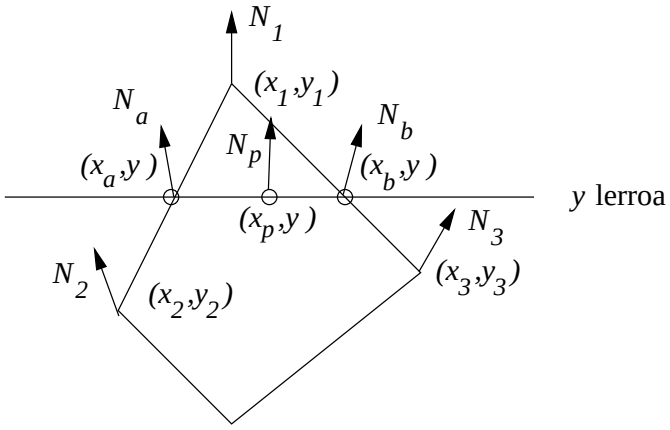
$$I_{x+1} = I_x + \Delta I_u$$

Batuketa bakar bat eginez, puntu bateko intentsitatea kalkulatzeko gai gara, eta hori, jakina, oso azkarra eta eraginkorra da. Teknika bera aplika daiteke ertzeko pixelen intentsitateak kalkulatzerakoan; hau da, I_a eta I_b ere batuketa bakarra eginez lor daitezke. Baina teknika hori Z-buffer algoritmoan edo tartekako lerroz lerroko algoritmoan barneratu beharko da.

Gouraud-en interpolazioaren arazo nagusienetakoa ispilu-isladarena da; ez ditu emaitza onak lortzen, eta horren arrazoia ulertzeko erraza da. Ispilu-islada poligono baten barnealdean ikusi beharko balitz, poligonoaren erpinetan ez litzateke ispilu-isladarik ikusiko, eta poligonoaren barneko puntuetako intentsitateak erpinetako intentsitateen interpolazio bidez lortzen direnez, ispilu-islada galdu egingo da. Eta alderantziz, ispilu-islada poligonoaren erpinen batean ikusiko balitz, erpinaren intentsitatea poligonoaren barnealdeko puntuetara hedatuko litzateke, eta ondorioz, ispilu-isladaren efektua poligonoan zehar barreiatuko litzateke.

8.9.2 Phong-en interpolazioa

Phong-ek, interpolazioa erabiliz poligonoaren barneko puntuen intentsitatea kalkulatzeko beste era bat aurkeztu zuen. Gainera, metodo horrek Gouraud-en interpolazioan ispilu-isladarekin gertatzen zen arazoa gainditzen du eta emaitza hobeak lortzen ditu. Horretarako, Gouraud-en interpolazioan bezala, poligonoen erpinetako informazioa interpolatzen du, baina bertako intentsitatea interpolatu beharrean, bektore normalaren interpolazioa egiten du, eta bektore hori erabiliz, pixel bakoitzari dagokion intentsitatea kalkulatzea proposatzen du. Interpolazioa egiteko 8.11 irudiko bektoreen kalkulua



Irudia 8.11: Erpinen bektore normala erabiliz, poligonoen barneko puntuena interpola daiteke

egin behar da, eta horretarako ekuazioak, Gouraud-en kasuan egiten genuenaren antzera, ondokoak izango dira:

$$N_a = \frac{1}{y_1 - y_2}(N_1(y - y_2) + N_2(y_1 - y))$$

$$N_b = \frac{1}{y_1 - y_3}(N_1(y - y_3) + N_3(y_1 - y))$$

$$N_p = \frac{1}{x_b - x_a}(N_a(x_b - x) + N_b(x - x_a))$$

Kontuan izan behar dugu, N delakoa bektorea dela eta hiru osagai dauzkala. Beraz, ekuazio horietako bakoitza hirukoiztu egin beharko da; gainera, eraginkorragoa izan dadin, gehikuntzaz kalkulatzea komeni da, eta Gouraud-ekin egiten genuena gogoratuz, azterketa-lerroan x kokapeneko N_x ezaguna dela suposatuz eta x-ren aldaketa bakoitzeko ΔN_u ere lor daitekeenez:

$$N_{px,x+1} = N_{px,x} + \Delta N_{ux}$$

$$N_{py,x+1} = N_{py,x} + \Delta N_{uy}$$

$$N_{pz,x+1} = N_{pz,x} + \Delta N_{uz}$$

Pixel bakoitzari dagokion bektorea kalkulatu ondoren, 8.1 ekuazioa erabiliz pixel horri dagokion intentsitatea kalkulatu ahal izango dugu, eta kalkulu horretan ispilu-isladari dagozkion argiak ager daitezke.

8.9.3 Gouraud versus Phong

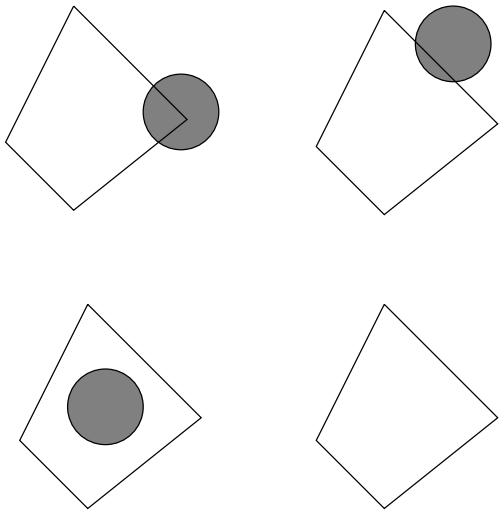
Phong-en interpolazioa Gouraud-enarekin konparatuz, kalkulu-kopurua asko aldatzen dela esan behar da. Gouraud-enean, poligonoko erpin bakoitzeko I kalkulatu behar da 8.1 ekuazioa erabiliz, eta ondoren poligonoko pixel bakoitzekoa batuketa bakarra eginez lor daiteke. Phong-en interpolazioan berriz, erpin bakoitzeko I kalkulatu egin behar da; horrez gain, poligonoen barneko pixel bakoitzeko N interpolatu behar da; azken finean, pixel bakoitzeko hiru batuketa, eta ondoren pixel bakoitzeko 8.1 ekuazioa balioztatu behar da. Aldea handia da bataren eta bestearen artean, baina emaitzetan ere bada alderik. Ispilu-isladaren aldetik, Gouraud-enak ez du emaitza ona lortzen; Phong-enak bai, ordea. Horren ordaina kalkulu-kopurua da.

Diferentzia hori dela-eta, ispilu-isladak garrantzirik ez badu, Gouraud-en metodoa erabiltzea komeniko da; eta alderantziz, ispilu-islada nahi badugu, Phong-ek proposatutako bidea hartzea hobea izango da. Hirugarren aukera ere proposa daiteke; hau da, biak erabiltzea, ispilu-isladarik gabeko irudia lortu eta beharrezkoa den eremuetan Phong-en interpolazioa erabiltzea. Baina beharrezkoa noiz den erabaki behar da, eta horretarako poligono bakoitzean azterketaren bat egin beharko dugu. Azterketa horren bidez, ispilu-isladarik ager daitekeen ala ez zehaztu behar da, eta horretarako, ager daitezkeen lau posibilitateen artean zein ematen den begiratu beharko dugu; lau posibilitateak 8.12 irudian ikus daitezkeenak dira.

Lehenengo kasua zehaztea erraza da. Poligonoko erpinen batean ispilu-isladari dagokion batugaia gutxieneko jakin bat baino handiagoa bada, erpin horrek ispilu-islada dauka, eta, ondorioz, Phong-en interpolazioa erabiltzea komeni da poligono horretan.

Bigarren kasurako, poligonoko alde bakoitzeko pixeletan ispilu-isladari dagokion terminoa gutxieneko balioa baino handiagoa den ala ez begiratu behar da; izan ere, hala bada, poligonoa argiztatzeko Phong-en interpolazioa erabili beharko baita.

Hirugarren kasua gertatzen den ala ez zehaztea konplexuagoa da; hala ere, poligonoa kobexoa bada, ez da zaila. Poligonoko alde bakoitzean zer gertatzen den begiratu behar da. Poligonoko alde guztietan, aldea mugatzen duten bi pixelei dagokien ispilu-islada baino handiagoko pixelik balego, poligonoaren barnealdean ispilu-islada ikusi beharko litzateke; hau da, poligonoa mugatzen duten ertz guztietan ispilu-islada maximoa ertzaren barnean ematen bada, eta ez ertza mugatzen duten erpinetan, ziurtasun osoz esan daiteke, poligonoaren barnean ispilu-islada maximoa egongo dela; beraz, poligono



Irudia 8.12: Ispilu-isladarekin gerta daitezkeen lau kasuak.

horrentzat Phong-en interpolazioa erabili behar da.

Bigarren eta hirugarren kasuetarako azterketak poligonoko ertz bakoitza aztertzea eskatzen du; beraz, biak batera egin daitezke. Ertz bateko pixel batean gutxieneko ispilu-islada gaindituko balitz, azterketa bukatu eta bigarren kasua gertatzen dela esan beharko da; baina bigarren kasua ematen ez den bitartean, aldagairen batean jaso beharko da ea hirugarren kasua gerta daitekeen. Kontuan izan behar da, nahikoa dela ertz batean maximoa ez egotea, bi erpinen artean hirugarren kasuan ez gaudela esateko. Hala ere, nahiz eta hirugarren kasua gertatzen ez dela jakin, ertz guztiak aztertzen jarraitu behar da, bigarrena gerta baitaiteke. Azterketa horiek amaitu ondoren emaitza garbirik ez lortzea, laugarren kasuaren adierazle izango da.

Azken finean azterketa horiek poligonoko erpinek erdibideko H bektorearekiko daukaten angelua begiratuz egin behar dira. Angelu hori txikia bada, ispilu-islada ikusi beharko da, eta handia bada, ez. *H-test* izenaz ezagutzen den algoritmoak aurreko lau kasuak aztertzen ditu, eta bere emaitzak, 8.13 algoritmoan ikus daitekeen bezala, poligonoak Phong-en interpolazioa ala Gouraud-ena behar duen adierazten du.

```
#define PHONG 1
#define GOURAUD 0
boolean H-test(double gutxienekoa, bektorea h)
{
ertz_guztietan_maximoa = TRUE;
for (i=0; i<erpin_kop; i++)
        {
        Bektorea[i] = bektore_normala_lortu(erpinak[i]);
        if (bek_biderkatu(bektorea[i],h) > gutxienekoa)
           return(PHONG); /* erpinak gutxieneko balioa gainditzen du */
        }
for (i=0; i<erpin_kop; i++)
   {
   a = bek_biderkatu(bektorea[i],h);
   b = bek_biderkatu(bektorea[div(i+1,erpin_kop-1).rem],h);
   c = bek_biderkatu(bektorea[i], bektorea[div(i+1,erpin_kop-1).rem]);
   d = b * c - a;
   e = a * c - b;
   f = d * e;
   if ( f > 0 ) /* ertzean maximoa dago */
      {
      g = a * d + b * e;
      if (d * g > 0) /* maximoa positiboa da */
         if (g * g >= gutxienekoa*gutxienekoa * (d*d +2*c*f + e*e)
                  /* ertzean ispilu-islada ikus daiteke */
            return(PHONG);
      }
   else
      ertz_guztietan_maximoa = FALSE;
   }
if (ertz_guztietan_maximoa) return(PHONG);
        else return(GOURAUD);
}
```

Irudia 8.13: H-test algoritmoa

Kapitulua 9

Izpi-hedaketa

9.1 Sarrera

Izpi-hedaketa, gaur egun ordenadore bidezko irudigintzan erabilitako argiztatze-eredu onenetakoa da. Aurretik azaldutako islada lokalaren ereduak, izpi-hedaketaren sinplifikazio modura har daitezke. Bai islada lokalaren ereduetan eta bai izpi-hedaketan, argia infinituki mehea den eta eszenan zehar bide zuzena egiten duen izpia gisa hartzen da.

Izpi-hedaketaren ideia nagusia ondokoa da: ikusle batek gainazal batean ikusten duen puntu baten kolorea, gainazalak puntu horretan eszenako edonondik datozen izpiekin duen elkarrekintzaren emaitza da. Eredu sinpleetan, gainazaleko puntuei argiek besterik ez dietela eragiten suposatzen da, eta horri zuzeneko argiztapena deritzo. Hala ere, orokorrean, izpi bat gainazal batera beste gainazal batean isladatu ondoren edo objektu gardenetan zehar edo bien konbinaketaren ondorioz hel daiteke. Argi-iturriak sortutako izpietara mugatu beharrean, inguruko objektuetatik datorren argia kontuan hartzeari argiztapen globala deritzo.

Metodoaren desabantaila konputazio-kostua da. Zenbait irudi lortzeko, minutuak, orduak edo egunak itxaroten egon beharko dugu. Abantaila nagusiak, argiztapen globalaren soluzio partzial bat izatea eta eredu bakar batean ondoko puntuen konbinazioa lortzea dira:

- Ezkutuko aurpegien ezabapena.
- Zuzeneko argiztapenaren eraginez ematen den itzaleztapena.
- Argiztapen globalaren eraginez ematen den itzaleztapena.

- Itzalen konputazioa.

Ondoren izpi-hedaketa aplikatzen duen programa bat garatzeko beharrezkoak diren zenbait xehetasun teoriko eta praktiko ikusiko ditugu, gero izpi-hedaketaren zenbait hedadura azaltzeko.

9.2 Oinarrizko algoritmoak

Izpi-hedaketaren algoritmoak objektu-espazioan egiten du lan. Irudi--planoko puntu bakoitzeko izpi bat edo gehiago hedatzen da. Izpi hori ikuspuntutik irudi-planoko puntura doa eta objekturen bat ebakitzen badu, kalkulu lokalen bidez puntuari dagokion kolorea lortuko dugu; hori, zuzeneko argiztapenaren emaitza izango da. Zuzeneko argiztapena, argi-iturri batetik heltzen den eta gainazalean isladatzen den argiaz arduratzen da. Objektuak partzialki argia isladatzen badu, edo partzialki gardena bada, edota partzialki gardena izanik partzialki argia isladatzen badu, irudi-planoan dagoen puntuaren koloreak gardentasunaren edo isladaren ondorioz sortutako izpi berrien eragina jasango du. Izpi berri horiek atzerantz hedatu beharko lirateke, puntuaren kolorean duten eragina kalkulatzeko. Izpi berri horien kolorea erabakitzeak, izpiak zein objekturekin egiten duen talka eta sor daitezkeen izpi berrien kalkulua egitea ekar dezake. Beraz, planoko puntuaren kolorea kalkulatzeko, sor daitezkeen izpi guztien hedaketa kalkulatu behar da. Izpi baten hedaketa, objekturik ebakitzen ez duenean (izpiari dagokion kolorea inguruarena da) edo izpi berrien hedaketek puntuaren kolorean eragin mesprezagarria duenean bukatzen da.

Oinarrizko programa batean, izpi bat irudiko puntu bakoitzaren erditik bidaliko litzateke. Puntu bakoitzeko izpi bat erabiltzen da. Prozesuak espazio-koherentziairik ez duela erabiltzen esaten da (izpi guztiak independenteki hedatzen dira, auzokide diren izpietan lortutako emaitzak ez dira erabiltzen) eta laginketa burutzeak aliasing efektua sortzen du. Bestalde, izpi bakoitzaren kalkulua beste guztietatik independente izateak, guztiz erraza egiten du ordenadore paraleloetan egin daitekeen izpi-hedaketaren algoritmoaren inplementazioa.

Hasiera batean izpi-hedaketa Apple etxeak (1968) erabili zuen. Ezkutuko Aurpegien ezabapen-arazoaren soluzio modura aplikatu zuen. Helburu horretarako erabil daitekeen algoritmo sinple batek, izpi bakoitzak zein gainazal ebakitzen duen kalkulatuko luke, eta ikuslearengandik gertuen dagoen gainazala izango litzateke puntu ikuskorra edukiko lukeena. Horrela, eszenako

puntu ikuskorrak kalkulatzen ziren. Hasierako izpi horiei, ikuslearen izpi, ezkutuko aurpegien izpiak edo lehen mailako izpiak deritze.

Eszenan isladatzaileak diren objektuak badaude, ez du zentzurik ezkutuko aurpegien ezabapena aplikatzeak, atzeko aurpegiak kentzean, bai isladaren eta bai gardentasunaren kalkulurako erabilgarriak diren aurpegien informazioa galduko bikenuke.

9.3 Ortzadarraren ezaugarri optikoak

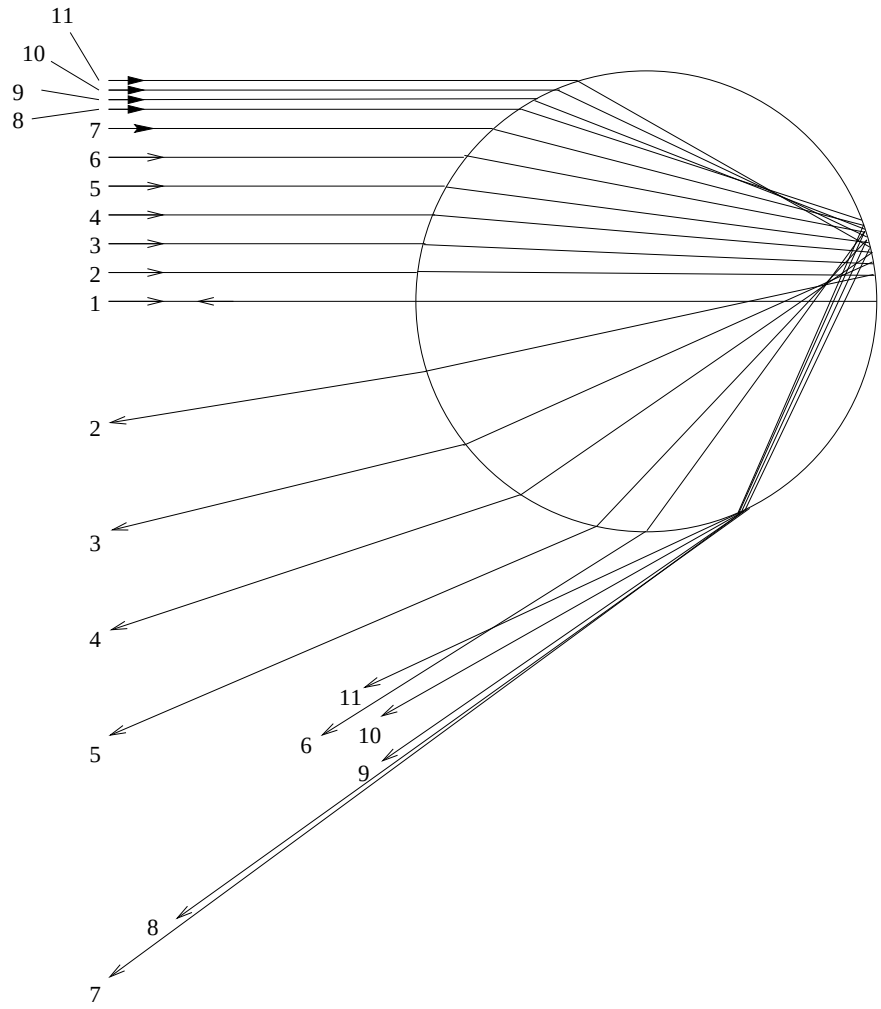
Pertsona askok izpi-hedaketaren teknika berria dela pentsatzen du, baina betidanik izan da optikaren zati bat. Rene Descartesek izpi-hedaketaren kontzeptua erabili zuen ortzadarraren itxura edo forma azaltzeko. Descartesek, ordurako ezagunak ziren errefrakzio- eta isladapenaren legeak erabili zituen esfera-itxurako ur-tanta batean zehar izpiak hedatzeko.

Izpiak esfera-itxurako ur-tanta batean sartzean, aire/ura elkarreragina dela-eta, errefraktatu egiten dira, ur/airea elkarreraginaren ondorioz barnean isladatu eta azkenik ur-tantatik ateratzean errefraktatu.

9.1 irudian izpi ezberdinek jarraitzen duten bidea ikus daiteke. Maximo bateraino, izpiaren irteera-angelua izpi erasotzailearen altueraren (diametro horizontalarekiko altueraren) funtzioa da. Portaera hori izpi konkretu batera heldu arte ematen da. Izpi horretara heltzean, portaera lehengoaren alderantzizkoa da, izpi erasotzailearen eta irteerako izpiaren arteko angelua txikituz doa. Izpi konkretu horri Descartesen izpia deritzo eta bere sarreraren eta irteeraren arteko angelua 42 gradukoa da. Descartesen izpitik gertu dauden izpi erasotzaileei dagozkien irteerako izpiak, Descartesen izpiaren irteerarekiko oso gertu egongo dira. Irudian Descartesen izpiaren inguruan ematen den izpi-kontzentrazioa ikus daiteke. Kontzentrazio horren eraginez, ortzadarra ikuskorra da.

Eguzkirantz begira ez dagoen ikusleak, 42 gradutako izpiak sortuko luketen ortzadarra ikusiko luke. Izpi horiek konoerdia sortzen dute eta bere muturra ikuslearen tokian kokatzen da.

Nahiz eta ortzadarraren itxuraren zergatia esplikatu, oraindik ez dugu horren kolorearen arrazoia azaldu. Kolorearen ezaugarria Newtoni esker ulertu zen. Newton argi zuria uhin-luzera guztietako argiz konposatuta zegoela ohartu zen. Material bakoitzaren errefrakzio-indizea uhin-luzera ezberdinentzat aldatu egiten dela jakinik, Descartesen hasierako eredua modu errazean heda daiteke. Eredua ikuslearen begian zentratutako konoerdien multzo



Irudia 9.1: Izpien eta ur-tanta baten arteko elkarrekintza.

baten modura har daiteke.

9.4 Izpi-hedaketaren inplementazioa

Izpi-hedaketaren algoritmoaren inplementazio orokorra zaila da; batez ere, kalkulu geometrikoak konplexuak direlako eta exekuzio-denbora handia delako. Exekuzio denbora handiegia izan ez dadin, eszenako irudia gutxi gora-behera nola irudikatuko den ikusteko, bereizmen espazial gutxiago izango duen irudi bat kalkula dezakegu; baina errore txikiak ikusi ahal izateko, bereizmen egokia erabili behar da.

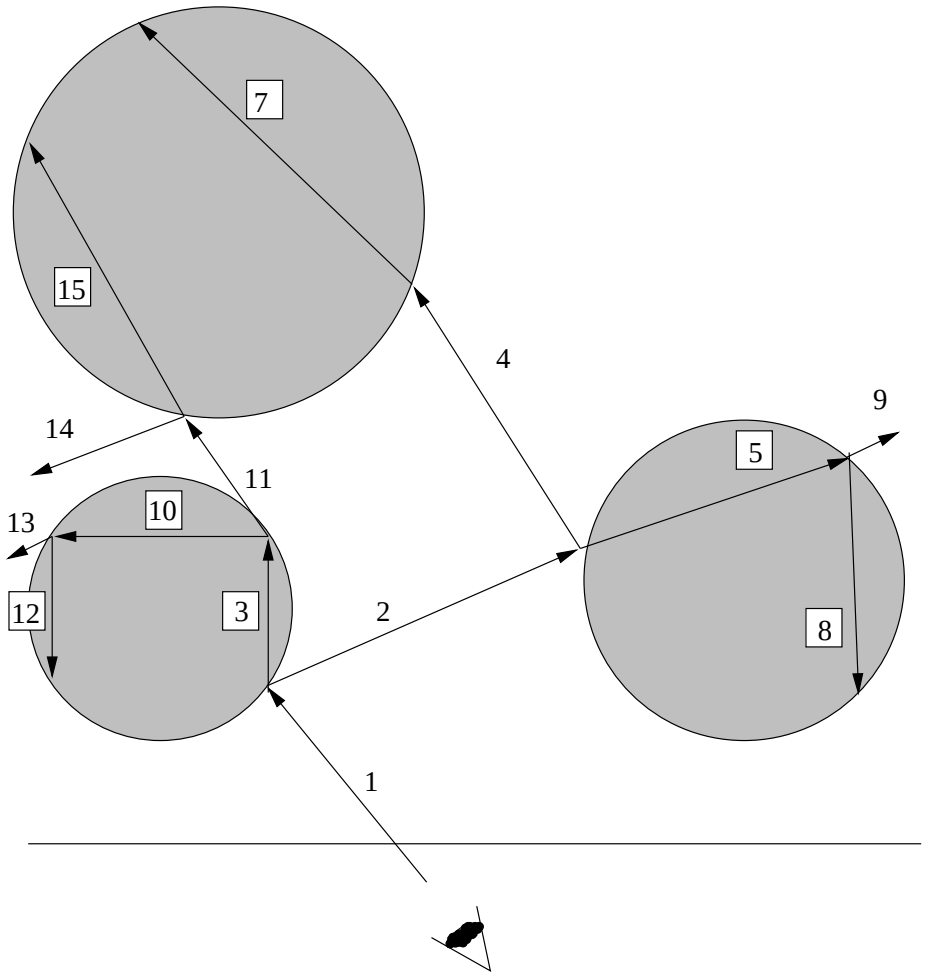
9.2 irudian ikus daitekeen bezala, 1 izpia ikuslearengandik irudiko pixel batera doa eta atzerantz hedatzen da erdi gardenak diren esferaz osatutako ingurune batean, esfera horiek argia isladatu eta errefraktatu egiten dute. Irudiko puntuan ikusi beharko litzatekeen kolorea kalkulatzeko, lehendabizi izpiak ebakitzen duen eta gertuen dagoen objektua zein den zehaztu beharko dugu. Objektua zein den jakinik, irudiko puntuari dagokion kolorea aukeratzeko, ondoko hiru eraginak ezin ditugu ahaztu:

1. Objektuak ebaketa-puntuan duen kolorea: hori kalkulatzeko, zuzeneko argiaren eta ingurune-argiaren eragina hartu behar dira kontuan.
2. Izpiaren isladaren ondorioz sortutako izpi berriaren (2 izpia) eragina.
3. Errefrakzioaren bitartez sortutako izpiaren (3 izpia) eragina.

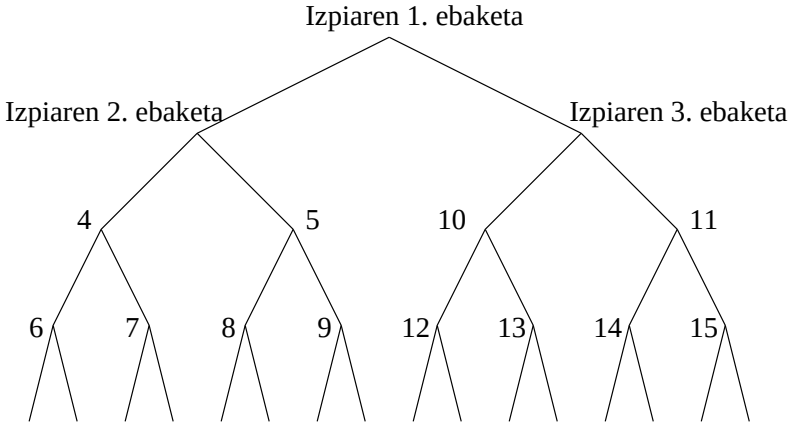
Isladatutako argi izpiaren eragina zein den jakiteko, atzerantz hedatu eta horrek duen lehenengo ebaketa, objekturik ebakitzen badu, kalkulatu beharko da. Ebaketarik badu, ebaketa-puntuaren kolorea kalkulatzeko berriz ere hiru eraginak aztertu beharko dira: argi-iturrien eragina, izpi isladatuarena eta objektua zeharkatzen duen izpiarena. Azken bi eraginak kalkulatzeko, 4 eta 5 izpiak atzerantz hedatu beharko lirateke. Beste izpiaren (3 izpia) kasuan, horren eragina kalkulatzeko gertuen dagoen objektuarekin daukan ebaketa-puntua kalkulatu beharko litzateke, etab.

9.3 irudian, puntuaren kolorea erabakitzeko egin beharko liratekeen ebaketen analisia azaltzen da. Lau aldiz hedatu ditugu izpiak, eta hortik aurrerako izpien eragina arbuiagarritzat hartu dugu.

1 izpia ezagutuz irudiko puntuaren kolorea kalkulatzeko, prozedura errekurtsiboari deitu beharko genioke. Parametroak hasierako puntua, izpiaren norabidea eta hedaketaren sakonera-maila izango lirateke.



Irudia 9.2: Oinarrizko izpi-hedaketa.



Irudia 9.3: Izpien eta objektuen arteko ebaketen zuhaitz bitarra.

9.5 irudiko zuhaitzak, erroko deiaren ondoren egingo liratekeen deiak deskribatzen ditu. Dei desberdinen aktibatzea goitik behera eta ezkerretik eskuinera emango litzateke. Beraz, bigarren izpiaren deia ematean bere ondorioz ematen diren dei guztiak bukatu arte ez litzateke hirugarren izpiaren deia egingo.

Azaldutako prozedura, irudiko pixel bakoitzeko behin eragingo litzateke.

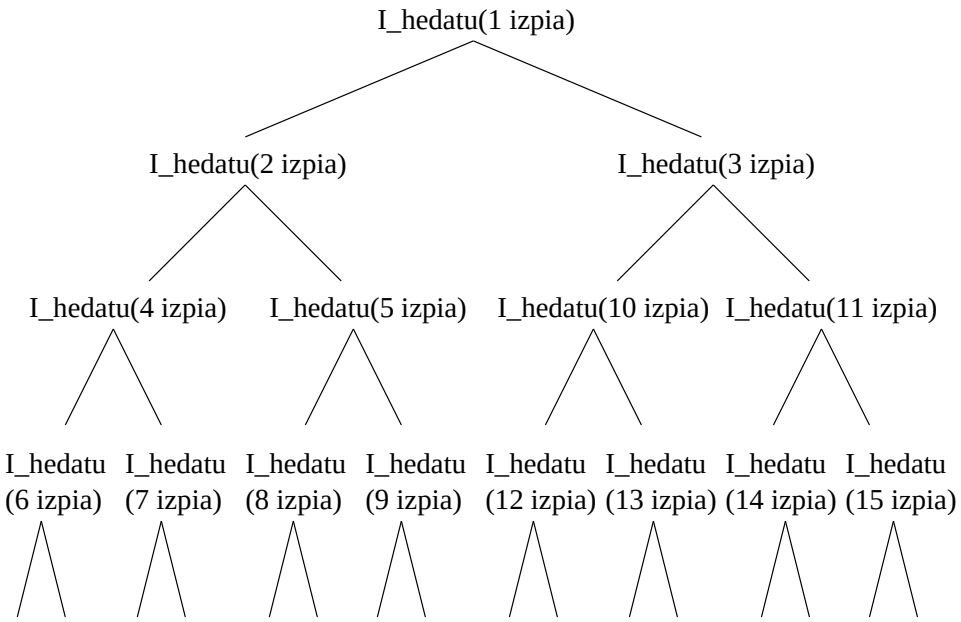
9.4.1 Prozeduraren eraginkortasuna

9.4 prozedura eraginkorragoa izan daiteke. Prozedura errekurtsibo askotan, prozeduraren hasieran oinarrizko kasua betetzen denentz begiratzen da, eta horrela bada, prozedura bukatutzat jotzen da. Beraz, zuhaitzaren hostoetan, segituan bukatzen diren eta beharrezkoak ez diren deiak egiten dira. Dei bakoitzean, pilarantz doazen parametroen neurria handiegia ez bada, prozeduraren argitasuna mantentzearren, onargarria da beharrezkoak ez diren hostoen deiak egitea. Izpi-hedaketaren kasuan, dei horiek egiteak islada- eta errefrakzio-norabideak kalkulatzea dakar. Kalkulu horiek zenbaki errealen aritmetika erabiltzen dute eta konputazio-kostua gehiegizkoa da.

9.6 prozedura berria erabiliz, hostoetako deiak ekiditea posible da. Prozedura horretan, lehenik sakonera-maila aztertzen da, dei errekurtsibo berririk egin behar denentz jakiteko; ondoren, "kolore lokalak"kalkulatzen dira.

```
Izpia_hedatu(bektoreak hasierako_puntua,norabidea,int sakonera,
                koloreak kolorea)
        {
        bektoreak ebaketa_puntua,
                        islada_norabidea, zeharkatze_norabidea;
        koloreak kolore_lokala,isladatutako_kolorea,kolore_zeharkatua;

       if (sakonera>sakonera_maximoa) kolorea=beltza;
       else {
               /* izpien eta objektu guztien arteko ebaketa-testa egin
               eta izpiaren hasierako puntutik gertuen dagoen
               ebaketa-puntua (halakorik balego) aurkitu */

              if (ebaketarik ez) kolorea=inguruko_kolorea;
              else {
                      kolore_lokala=(eredu lokalaren eragina ebaketa-puntuan);

                      /* isladatutako izpiaren norabidearen kalkulua */
                      Izpia_hedatu(ebaketa_puntua,islada_norabidea,
                                      sakonera+1, isladatutako_kolorea);

                      /* objektua igarotzen duen izpiaren norabidearen kalkulua */
                      Izpia_hedatu(ebaketa_puntua,zeharkatze_norabidea,
                                      sakonera+1, kolore_zeharkatua);

                      Konbinatu_koloreak(kolore_lokala,
                                              kolore_lokalaren_pisua,
                                              isladatutako_kolorea,
                                              isladatutako_kolorearen_pisua,
                                              kolore_zeharkatua,
                                              kolore_zeharkatuaren_pisua)
                     }
              }
       }
```

Irudia 9.4: Izpi-hedaketaren algoritmoa.



Irudia 9.5: Ebaketa guztien eragina kalkulatzeko egin beharreko prozedura-deien zuhaitza.

```
Izpia_hedatu(bektoreak hasierako_puntua,norabidea,int sakonera,
                koloreak kolorea)
        { bektoreak ebaketa_puntua,islada_norabidea,
                zeharkatze_norabidea;
         koloreak kolore_lokala,isladatutako_kolorea,kolore_zeharkatua;

         /* izpien eta objektu guztien arteko ebaketa-testa egin
         eta izpiaren hasierako puntutik gertuen dagoen
         ebaketa-puntua (halakorik balego) aurkitu */

        if (ebaketarik ez) kolorea=inguruko_kolorea;
        else
                {
                 kolore_lokala=(eredu lokalaren eragina);
                if (sakonera==sakonera_maximoa)
                        { kolore_zeharkatua=beltza;
                         isladatutako_kolorea=beltza;
                        }
                else
                        { /* isladatutako izpiaren norabidearen kalkulua */
                         Izpia_hedatu(ebaketa_puntua,islada_norabidea,
                                 sakonera+1,isladatutako_kolorea);
                         /* Objektua igarotzen duen izpiaren norabidearen kalkulua */
                         Izpia_hedatu(ebaketa_puntua,zeharkatze_norabidea,
                                 sakonera+1,kolore_zeharkatua);
                        }

                 Konbinatu_koloreak(kolore_lokala,kolore_lokalaren_pisua,
                                         isladatutako_kolorea,
                                         isladatutako_kolorearen_pisua,
                                         kolore_zeharkatua,
                                         kolore_zeharkatuaren_pisua)
                }
        }
```

Irudia 9.6: Izpi-hedaketaren algoritmo eraginkorra.

Dei errekurtsibo gehiago egin behar bada, islada- eta errefrakzio-norabideen kalkuluak egiten dira; bestela (lehen oinarrizko kasuan gertatzen zen bezala), kolore beltza esleitzen zaie, bai errefraktatutako izpiaren koloreari eta bai objektuan isladatutako izpiaren koloreari ere.

9.5 Izpi-hedaketaren geometria

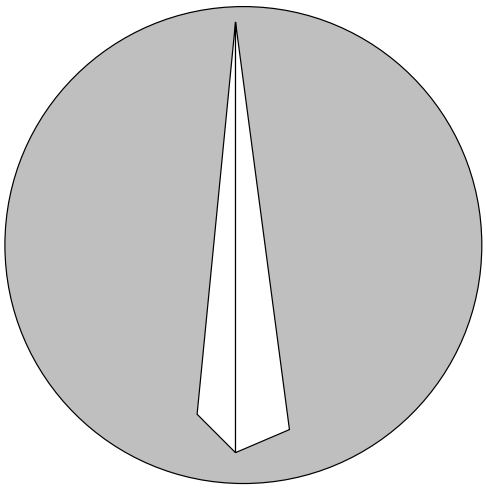
9.5.1 Ebaketak

Esan bezala, hasierako izpiak, ikuslearengandik irudiko puntura doanak, eszenako objekturik ebakitzen duenentz aztertzen da. Ebaketa-puntuen koordenatuak gorde egiten dira, eta ikuslearengandik gertuen dagoena aukeratzen da. Hori hasierako izpiaren lehen ebaketa da. Izpi horrek beharbada beste bi izpi sortaraziko ditu, eta bakoitzak eszenako objekturik ebakitzen ote duen aztertu behar da. Sortzen den izpi bakoitzeko eszenako objektu guztiekin, ebaketa-kalkuluak egin behar ditugu. Argi ikusten da, hortaz ebaketa-kalkuluak egiten programaren exekuzio-denbora handia galtzen dela.

Ebaketa-kalkuluak azkartzeko, borne-bolumenak erabili izan dira; hau da, eszenako objektu bakoitzari borne-bolumen bat egokitzen zaio, edozein objektu borne-bolumen batean barnera daiteke. Borne-bolumen modura, ebaketak era errazean kalkulatzeko aukera ematen duena aukeratu behar da. Izpiak borne-bolumena ebakitzen ez badu, barnean dagoen objektua ez du ebakiko. Poligono askoz osatutako objektu baten kasuan, kalkulu asko ekidin dezakegu; izan ere, izpiak borne-bolumena ebakitzen ez badu, izpiaren eta objektuko poligono guztien arteko ebaketarik ez baitugu kalkulatu beharko.

Batez ere esferak erabili izan dira borne-bolumen modura; hala ere, zenbait objekturen itxurara ez dira egokitzen. Hain zuzen, zenbait objekturen kasuan izpiak esfera ebaki arren, ez du barnean dagoen objektua ebakiko eta ezertarako balio ez duten kalkuluak egin beharko ditugu (izpiaren eta objektuko poligono bakoitzaren arteko ebaketa-kalkuluak). 9.7 irudian azaltzen den bezala, meheak eta luzeak diren objektuen kasuan, esfera ebakitzen duten izpi gehienek ez dute barneko objektua ebakiko.

Izpi-hedaketaz sortutako lehen irudiak esferaz osatuta zeuden, izpien eta esferaren arteko ebaketak kalkulatzea erraza baita.



Irudia 9.7: Kasu horretan esfera borne-bolumen modura erabiltzea ez litzateke eraginkorra izango.

Izpien eta esferen arteko ebaketak

Izpien eta esferen arteko ebaketak modu errazean kalkula daitezke. Izpi baten hasierako eta bukaerako puntuak (x_1,y_1,z_1) eta (x_2,y_2,z_2) badira, lehenengo pausoa izpia era parametrikoan adieraztea izango da:

$$\begin{aligned} x &= x_1 + (x_2 - x_1)t = x_1 + it \\ y &= y_1 + (y_2 - y_1)t = y_1 + jt \\ z &= z_1 + (z_2 - z_1)t = z_1 + kt \end{aligned} \tag{9.1}$$

Eta (l, m, n) zentroa eta r erradioa duen esferaren ekuazioa:

$$(x - l)^2 + (y - m)^2 + (z - n)^2 = r^2$$

x, y eta z ordezkatuz 2. mailako ekuazioa lortzen dugu:

$$at^2 + bt + c = 0$$

non:

$$\begin{aligned} a &= i^2 + j^2 + k^2 \\ b &= 2i(x_1 - l) + 2j(y_1 - m) + 2k(z_1 - n) \\ c &= l^2 + m^2 + n^2 + {x_1}^2 + {y_1}^2 + {z_1}^2 + 2(-lx_1 - my_1 - nz1) - r^2 \end{aligned}$$

Ekuazioak bi soluzio, soluzio bakarra ala soluziorik ez edukitzea gerta liteke. Bakarra badu, izpia esferaren ukitzailea izango da; bi baditu, izpia esferan sartu eta atera egiten dela adieraziko digu; eta soluziorik ez badu, izpiak esfera ukitzen ez duela esan nahi du. Askatutako t balioa izpiaren hasierako ekuazioetan ordezkatuz, ebaketa-puntuaren koordenatuak lor daitezke. t-ren balio positiboak soilik dira erabilgarriak, eta balio horrek ebaketa-puntuaren hurbiltasuna edo urruntasuna adierazten du.

Beharrezko beste datu bat, ebaketa-puntuan objektutik kanporanzko norabidea adierazten duen bektorea izaten da, horri bektore normala deritzo eta isladatutako eta errefraktatutako izpiak kalkulatu ahal izateko beharrezkoa da. Esfera borne-bolumen modura erabiltzen bada, esferako gainazalaren normala ez da beharrezkoa; kasu horretan interesatzen zaigun gauza bakarra, ebaketa eman den ala ez jakitea baita.

Ebaketa-puntua (x_i,y_i,z_i) eta esferaren zentroa (l, m, n) izanik, ebaketa-puntuko bektore normala hauxe da:

$$N = \left(\frac{x_i-l}{r}, \frac{y_i-m}{r}, \frac{z_i-n}{r} \right)$$

Izpien eta poliedro konbexoen arteko ebaketak

Objektu bat poligono-multzo batez adierazita badago eta konbexoa bada, izpia poligono bakoitzarekiko aztertzea izango litzateke hurbilketa-mota bat. Horrela egingo genuke:

1. Poligonoa dagoen planoaren ekuazioa lortu.
2. Izpiak planoa ebakitzen ote duen aztertu.
3. Ebaketarik badauka, ebaketa-puntua poligonoaren barnean dagoenentz aztertu.

Adibidez, poligonoa dagoen planoaren ekuazioa ondokoa bada:

$$ax + by + cz + d = 0$$

eta zuzena era parametrikoan definituta badago, ebaketa-puntua ondoko ekuaziotik lor daiteke:

$$t = -\frac{ax_1 + by_1 + cz_1 + d}{ai + bj + ck} \tag{9.2}$$

$t < 0$ bada, planoa izpiaren atzean dagoela esan daiteke.

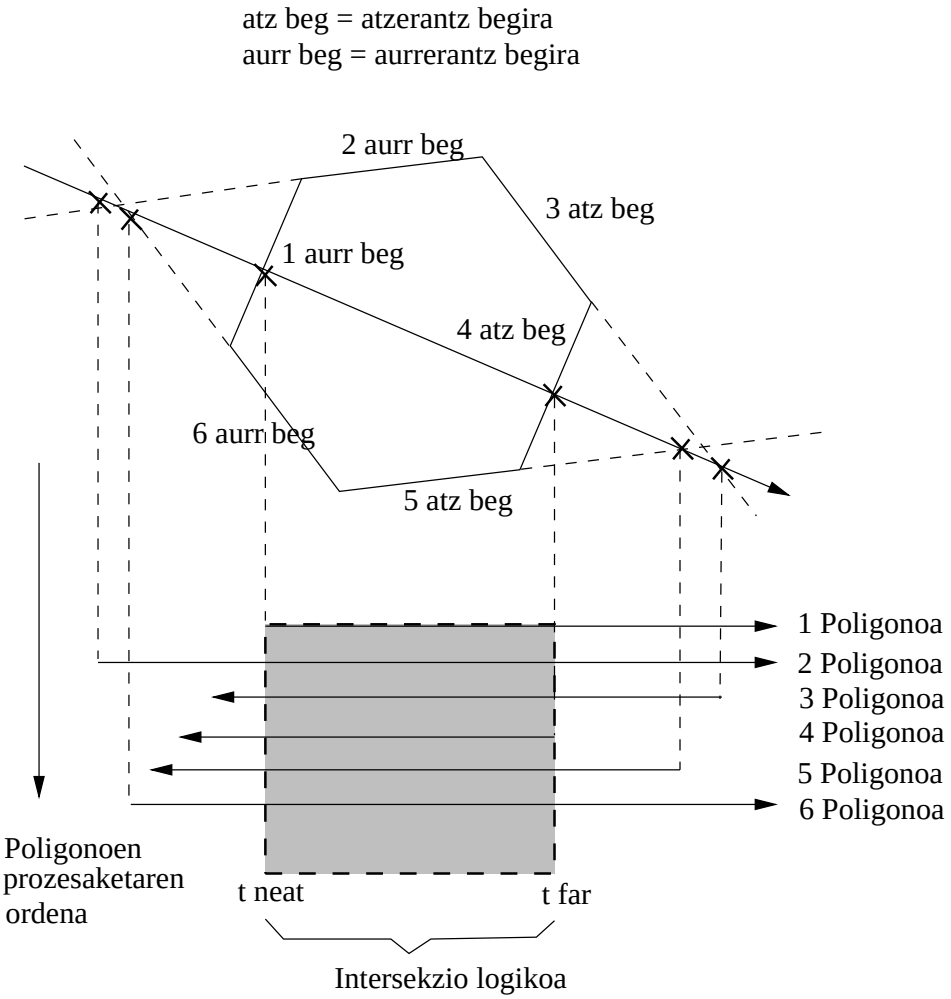
Izendatzailea 0 bada, zuzena eta planoa paraleloak dira. Kasu horretan, izpiaren hasierako puntua poligonoaren barnean edo kanpoan egon daiteke. Barnean ala kanpoan dagoen jakiteko, nahikoa da zenbakitzailearen ikurra aztertzea. Zenbakitzailea positiboa bada, izpia objektutik kanpo dagoen planoaren erdi-espazioan dago eta ez da azterketa gehiago behar.

Haines-ek (1991) garatu zuen algoritmoa erabiliz, izpi batek poligonoz osatutako objektua ebaki egiten duen ala ez jakin daiteke.

Horretarako, aurpegiak bi multzotan banatzen dira: izpia sortu den punturantz begira daudenak eta puntu horretarantz begira ez daudenak. Informazio hori 9.2 ekuazioaren izendatzailearen ikurraren bidez lor daiteke. Ondoren aurpegi bakoitzari dagokion planoa izpiak non ebakitzen duen kalkulatu behar da, eta izpiaren sorlekua ikusten duten aurpegien artetik ebaketa-puntua urrunen duena, t_{near}, aukeratzen da. Bestalde, izpiaren sorlekua ikusten ez dutenen artetik ebaketa-puntua gertuen duena, t_{far}, aukeratzen da. Bi ebaketa-puntu hauen arteko erlazioak, izpiak objektua zeharkatzen duen ala ez adierazten du; hau da, t_{near} t_{far} baino handiagoa bada, izpiak ez du objektua ukitzen, eta alderantziz gertatzen bada, izpia objektuan sartzen dela esan daiteke; gainera, sarrera-puntua t_{near} puntua izango da. Azterketa horri esker, ebaketa-puntua aurpegia definitzen duen poligonoaren barnean dagoela badakigu, eta azterketa hori egin beharrik ez dugu izango. 9.8 irudian algoritmo horren adibidea ikus daiteke, baina bi dimentsioetan; eta 9.9 algoritmoak azterketa egiteko sasi-kodea agertzen du.

Izpien eta kutxen arteko ebaketak

Izpien eta kutxen arteko ebaketen kalkulua garrantzi handikoa da, batez ere kutxa esfera baino borne-bolumen erabilgarriagoa izan daitekeelako. Kutxa borne-bolumen eraginkor gisa erabil daiteke, eta adibidez, kutxa orokortuak har ditzakegu. Kutxa-mota berezi horiek paralelo diren plano-bikotez osatzen dira, eta plano-bikote bakoitza beste bikoteekiko edozein angelutara koka daiteke. Izpiak horrelako kutxa bat ebakitzen ote duen jakitea sinplea da, eta algoritmoa izpien eta poligono konbexoen arteko algoritmoaren kasu berezia besterik ez da. Urrats bakoitzean plano-bikote bat tratatuko dugu, lehenengo planora arte dagoen distantzia (t_{near}) eta bigarren planora arte dagoen distantzia (t_{far}) kalkulatuz. Plano-bikote guztiak bata bestearen ondoren tratatu eta gero, t_{near} balio handiena eta t_{far} balio txikiena lortuko ditugu. t_{near} balio handiena t_{far} balio txikiena baino handiagoa bada, izpiak



Irudia 9.8: Izpien eta poligono konbexoen arteko ebaketen azterketa.

```
t_near=zenbaki_negatibo_handia;
t_far=zenbaki_positibo_handia;
for poliedroa osatzen duen plano bakoitzeko
        {
        t=plano eta izpiaren arteko azterketa parametrikoaren emaitza;
        if ((planoa atzerantz begira dago) && (t<t_far)) t_far=t;
        if ((planoa aurrerantz begira dago) && (t>t_near)) t_near=t;
        }
if (t_near>t_far) ez dago ebaketarik;
```

Irudia 9.9: Izpien eta poliedro konbexoen arteko ebaketen azterketa.

ez du kutxa ebakitzen. Hori guztia 9.10 irudian ikus daiteke. Ebaketarik balego, ebaketa-puntua t_{near} balioak adieraziko liguke.

Hori bera beste era batera ere uler daiteke. Plano-bikote bakoitzak t-ren tarte bat definitzen du; tarte horien arteko ebaketa hutsa bada, izpiak ez du kutxa zeharkatzen; baina, alderantziz, tarte horiek gainjarri egingo balira, ebaketa egongo litzateke.

Izpien eta koadriken arteko ebaketak

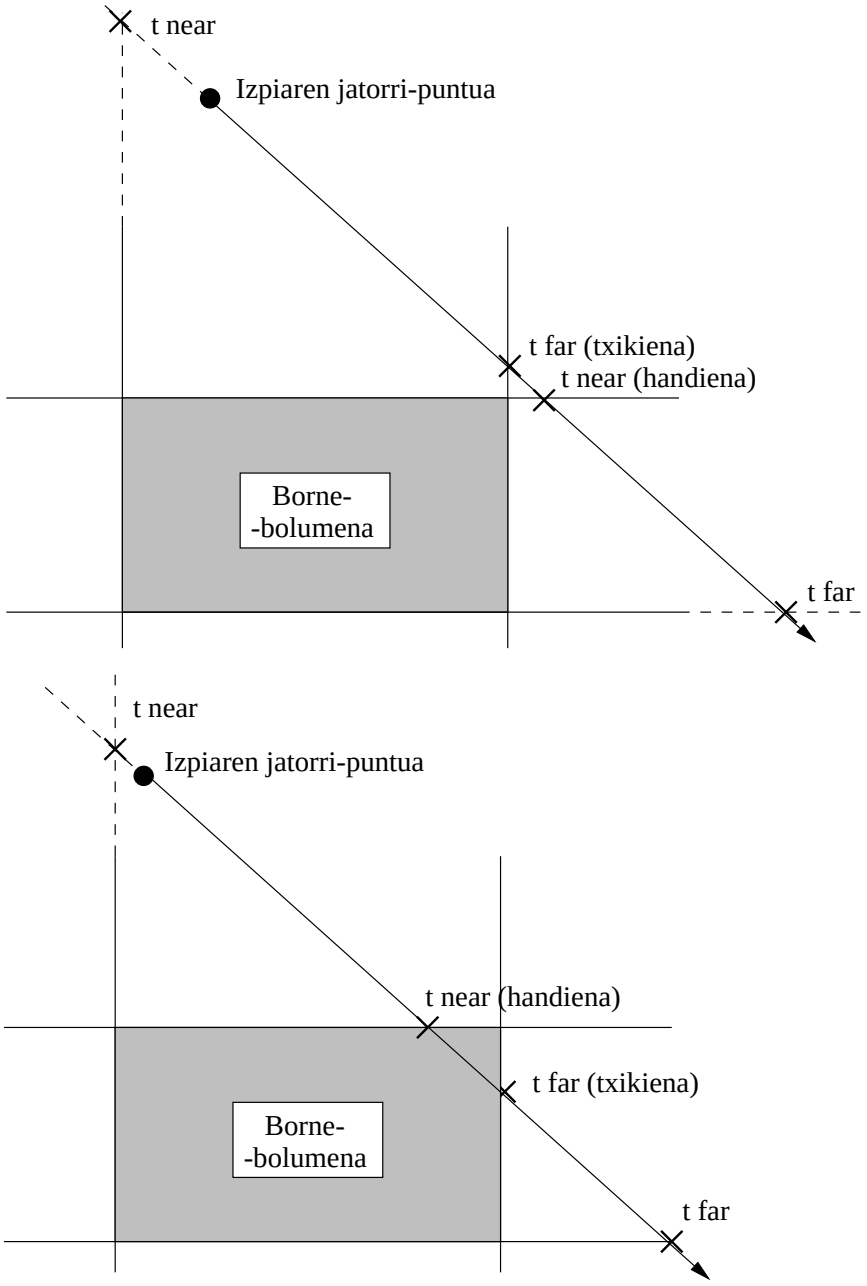
Izpien eta 2. graduko ekuazioen bidez adieraz daitezkeen objektuen arteko ebaketak, kasu orokor edo objektu berezien kasu (zilindroak, elipsoideak, paraboloideak, ...) modura ebatz daitezke. Kasu bereziak eraginkortasun-arrazoiengatik aztertzen dira. 2. graduko ekuazio inplizitu orokorra honako hau da:

$$Ax^2 + Ey^2 + Hz^2 + 2Bxy + 2Fyz + 2Cxz + 2Dx + 2Gy + 2Iz + J = 0$$

Edo matrize idazkera erabiliz:

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{pmatrix} \begin{pmatrix} x \\ y \\ x \\ 1 \end{pmatrix} = 0$$

Esferaren kasuan erabilitako prozesua jarraituz, 9.1 ekuazioko s, y eta z ordezkatu behar dira eta a, b eta c koefizienteak lortuko ditugu:



Irudia 9.10: Izpien eta kutxen arteko ebaketen azterketa.

$$\begin{aligned} a = \;& Ax_d{}^2 + Ey_d{}^2 + Hz_d{}^2 + 2Bx_dy_d + 2Cx_dz_d + 2Fx_dz_d \\ b = \;& 2(Ax_1x_d + B(x_1y_d + x_dy_1) + C(x_1z_d + x_dz_1) + \\ & Dx_d + Ey_1y_d + F(y_1z_d + y_dz_1) + Gy_d + Hz_1z_d + Iz_d) \\ c = \;& Ax_1{}^2 + Ey^2 + Hz_1{}^2 + 2Bx_1y_1 + \\ & 2Cx_1z_1 + 2Dx_1 + 2Fy_1z_1 + 2Gy_1 + 2Iz_1 + J \end{aligned}$$

non (x_d,y_d,z_d) izpiaren norabide normalizatua den.

2. graduko ekuazioen bidez adieraz daitezkeen objektu batzuk honako hauek dira:

- Esfera.

$$(x-l)^2 + (y-m)^2 + (z-n)^2 = r^2$$

  non (l, m, n) esferaren zentroaren koordenatuak diren.

- Zilindro infinitua.

$$(x-l)^2 + (y-m)^2 = r^2$$

- Elipsoidea.

$$\frac{(x-l)^2}{\alpha^2} + \frac{(y-m)^2}{\beta^2} + \frac{(z-n)^2}{\delta^2} - 1 = 0$$

  non α, β eta δ ardatzerdiak diren.

- Paraboloidea.

$$\frac{(x-l)^2}{\alpha^2} + \frac{(y-m)^2}{\beta^2} - z + n = 0$$

- Hiperboloidea.

$$\frac{(x-l)^2}{\alpha^2} + \frac{(y-m)^2}{\beta^2} - \frac{(z-n)^2}{\delta^2} - 1 = 0$$

Izpien eta txatal bikubikoen arteko ebaketak lortzeko, zenbait metodo garatu izan dira. Horien artean ondokoa aipa genezake: Borne-bolumen esferikoen ideia txatal bikubikoetara hedatzearen ondorioa da, eta gainazalaren txatal bakoitza esfera batean barneratzen da. Izpiak esfera ebakitzen badu, txatala txatal txikiagotan zatitzen da (txatal-zatiketarako algoritmo estandar bat erabiliz) eta lortutako txatal bakoitza esfera berri batean barneratzen da. Prozesu hori izpiak esfera ebakitzen ez duenean edo esferaren neurria (erradioa) aurretik zehaztatutako minimo batera heltzen denean bukatzen da. Prozesu horretatik lortutako ebaketa-puntuaren zehaztasuna, ezarritako esferaren erradio minimoaren araberakoa izango da.

9.6 Islada bidezko argiztatze-eredua

Zuzeneko argiztapen eta argiztapen globalaren arteko ezberdintasuna azpimarratzea komeni da. Zuzeneko argiztapena argi-iturri batetik edo gehiagotatik sortzen den eta eszenan eragina duen argia da. Zuzeneko argiztapenean, gainazal batek argia zuzenean jasotzen du argi-iturritik. Bestalde, objekturen batekin edo batzuekin harremanak izan ondoren izpien eta objektuen arteko elkarreraginaren ondorioz, argi izpiren bat gainazal batera iristen bada, argiztatze horri argiztatze globala deritzo. Beraz, argiztatze globala argiztatze lokalak objektu garden eta isladatzaileekin duen elkarreraginaren ondorioa da.

Phong-en zuzeneko argiztapen-ereduan, hori zuzeneko argiztatze-eredua dela kontuan izan behar da. Hiru gai erabiltzen dira: inguruneko argia, islada barreiatua eta ispilu-islada. Gai horietako bakoitza hiru osagaiz osatutako RGB bektorea da.

Bai islada barreiatua eta bai ispilu-islada, zuzeneko argiztapenaren ondorio dira. Inguruneko argia, berriz, eszena osotik sortutako argi barreiatuaren ondorioa da. Phong-en ereduan, inguruneko argia konstante modura hartzen da.

Argiztapenaren osagai lokala kalkulatzeko, Phong-en ereduaren hedapen bat erabil daiteke. Lehengo hiru osagaiei beste bat gehitzen zaie. Laugarren osagai berri hori objektu gardenak kontuan hartzeko txertatzen da; horren bidez, objektu garden batean zehar ikus daitezkeen argi-iturriak ere kontuan izango ditugu. Ingurune edo medio bat zeharkatzen duen argiaren sakabanatzeak, puntuzko argi-iturri baten irudia zirriborrotsu azaleraziko luke. Sakabanatze hori modelatu ahal izateko:

$$I_{\text{zehar}} = K_t(N\ H')^n$$

H erdibideko bektorea, aurpegiak ikuslearenganantz argi maximoa islada dezan eduki beharko lukeen norabidea da. Eta H' berriz, ikuslearenganantz argi maximoa errefraktatzeko aurpegiak izan beharko lukeen norabidea. H' beraz L, V eta errefrakzio erlatiboaren indizearen arteko funtzioa izango da:

$$H' = \frac{-L - (\eta_2/\eta_1)V}{\eta_2/\eta_1 - 1} = \frac{-L - \eta V}{\eta - 1}$$

Izpi-hedaketaren ereduan erabilgarria gertatzen da, konstantetzat hartzen den inguruneko argia izatez gai globala dela ahaztea, eta gai lokal bezala erabiltzea. Osagai lokalak ebaketa-puntu bakoitzean, zuzeneko argiztapen eta inguruneko argiaren arabera kalkulatzen dira. Beraz, gure eredu lokala ondokoa izango da:

$$I_{\text{lokala}} = K_a I_a + \sum_{i=1}^{n} I_{li}[K_d(N\ L_i) + K_s(N\ H_i)^{ns} + K_t(N\ H_i')^{n_t}]$$

Edozein argi-iturrirentzat azkeneko bi gaietako bakarra aktibatuko da. H eta H', V bektorea erabiliz lortzen dira.

Izpi-hedaketan, eredu horri globaltzat hartzen diren beste bi gai gehitzen zaizkio. Bi gai horiek **edonondik** hel daitezkeen izpien eraginez arduratzen dira:

1. Gainazalera heltzen den isladatutako izpiaren eragina.

2. Gainazalera heltzen den eta objekturen bat zeharkatzen duen izpiaren eragina.

Bi izpi horien eraginak, izpi-hedaketaren prozedura errekurtsiboaren bidez kalkula daitezke:

$$I = I_{\text{lokala}} + K_{rg} I_{\text{islada}} + K_{tg} I_{\text{zehar}}$$

Garrantzitsua da, espresio horrek, prozesu errekurtsiboaren puntu konkretu bateko hiru balioen batura adierazten duela konturatzea. I_{zehar} kalkulatzean, espresio baliokide bat sortzen da, objektua igarotzen duen izpiak ebakitzen duen hurrengo objektuan. Eta I_{islada} beste espresio baliokide bat izango da. Ondorioz, objektu anitzez osatutako eszena batean, ikuskorra den gainazal batean eman daitekeen islada, hiru iturriren eraginez azal daiteke:

1. Inguruneko argia. Gainazal batek beti inguruneko argia jasoko du.

2. Zuzeneko argi-iturriak. Gainazal batek argi-iturri horietatik argia jasoko du argi-iturritik, ikuskorra bada.

3. Objektuen arteko elkarreragina dela-eta norabide jakin baten inguruan kontzentratutako argi-izpi hedatuak.

Izpi-hedaketa, eredu hibrido bat da: izpi batzuk hedatu egiten dira, baina beste batzuk ez. Zuzeneko ispilu-isladari dagokion gaiaren sakabanaketa kalkulatzean, izpia hedatu beharrean, enpirikoki kalkulatzen da. Baina izpi-hedaketaren gai globalen kasuan, zeharkatutako eta isladatutako gaiak ez dira sakabanatzen. Beraz, zuzeneko argiztapenari dagozkion izpiak sakabanatu egiten dira (enpirikoki), baina ikuslearengandik datozen izpiak, izpi hedatuak, ez. Hori ereduari lotutako kontraesana da, eta ordenadore bidezko irudigintzako ereduak trikimailu enpiriko eta teorikoz osatuta daudela azaltzen du.

Izpi-hedaketan oinarritutako intentsitatearen kalkulurako azken ekuazioa hauxe da:

$$
\begin{aligned}
I &= I_{\text{lokala}} + I_{\text{globala}} = && (9.3)\\
&\quad K_a I_a + \sum_{i=1}^{n} I_{li}[K_d(N\ L_i) + K_s(N\ H_i)^{ns} + K_t(N; H_i')^{n_t}] + \\
&\quad K_{rg} I_{\text{islada}} + K_{tg} I_{\text{zehar}} && (9.4)
\end{aligned}
$$

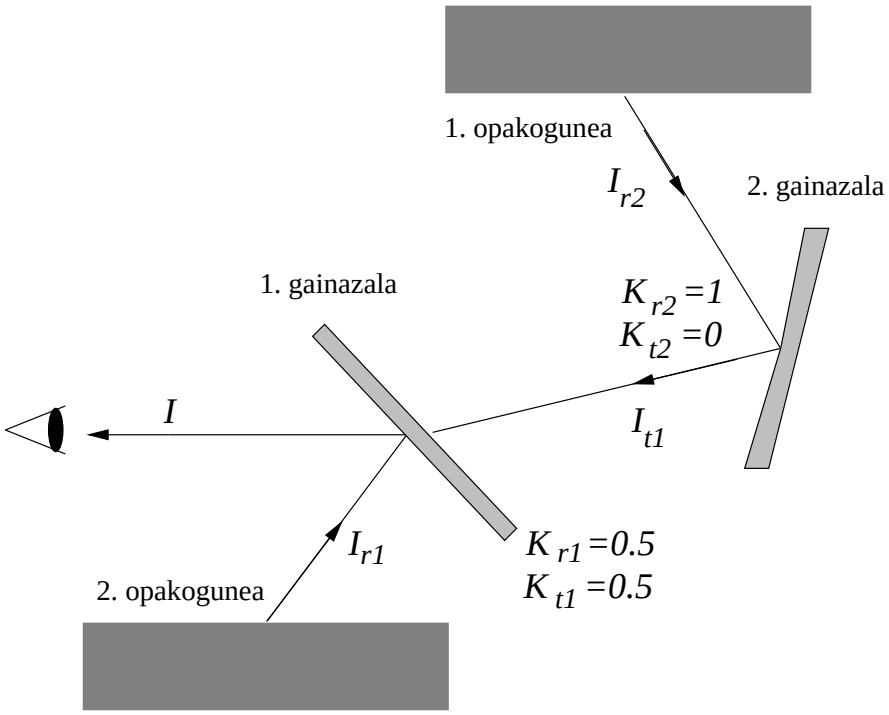
9.4 ekuazioko bi intentsitateak RGB bektoreak dira eta izaera errekurtsiboa daukate; hala ere, ekuazioan ez da izaera hori agertzen. Prozesuaren izaera errekurtsiboa azpimarratzeko, ekuazioa ondoko moduan idatz dezakegu:

$$I(P) = I_{\text{lokala}}(P) + K_{tg} I(P_{\text{zehar}}) + K_{rg} I(P_{\text{islada}})$$

non:

- P delakoa aztertzen ari garen ebaketa-puntua den.

- P_{islada}, P-tik isladatzen den hedatutako izpiaren lehenengo talka edo ebaketa den.



Irudia 9.11: Bi dimentsioko eszena sinple bateko izpi-hedaketaren azterketa.

- P_{zehar}, P-tik objektuan zehar hedatutako izpiaren lehenengo talka.

K_s-ren balioak K_{rg}-renaren berdina izan beharko luke; baina gure oinarrizko eredu horretan komenigarria da, K_s zuzeneko argiztapenaren eraginez sortutako puntu distiratsuen intentsitatea kontrolatzeko eta K_{rg} objektuen ispilu-islada kontrolatzeko erabiltzea. Gauza bera esan daiteke K_t eta K_{tg} parametroei buruz.

Azkenik, adibide erraz bat azalduko dugu. 9.11 irudiko bi dimentsioko eszena sinplea erabiliko dugu, bi objektu opakoz, gainazal erdi-garden batez eta guztiz isladatzailea den gainazal batez osatua. Kalkuluak 3. errekurtsibitate--mailaraino aztertuko ditugu:

$$I = I_{\text{gainazal1lokala}} + K_{r1} I_{\text{islada1}} + K_{t1} I_{\text{zehar1}}$$

Ondoren I_{islada1} eta I_{zehar1} kalkulatzen dira:

$$I_{\text{islada1}} = I_{\text{opako2}} + 0 + 0$$

$$I_{\text{zehar1}} = I_{\text{gainazal2lokala}} + K_{r2} I_{\text{islada2}}$$

Eta errekurtsibitatea I_{islada2}-ren kalkuluarekin bukatzen da:

$$I_{\text{islada2}} = I_{\text{opako1}} + 0 + 0$$

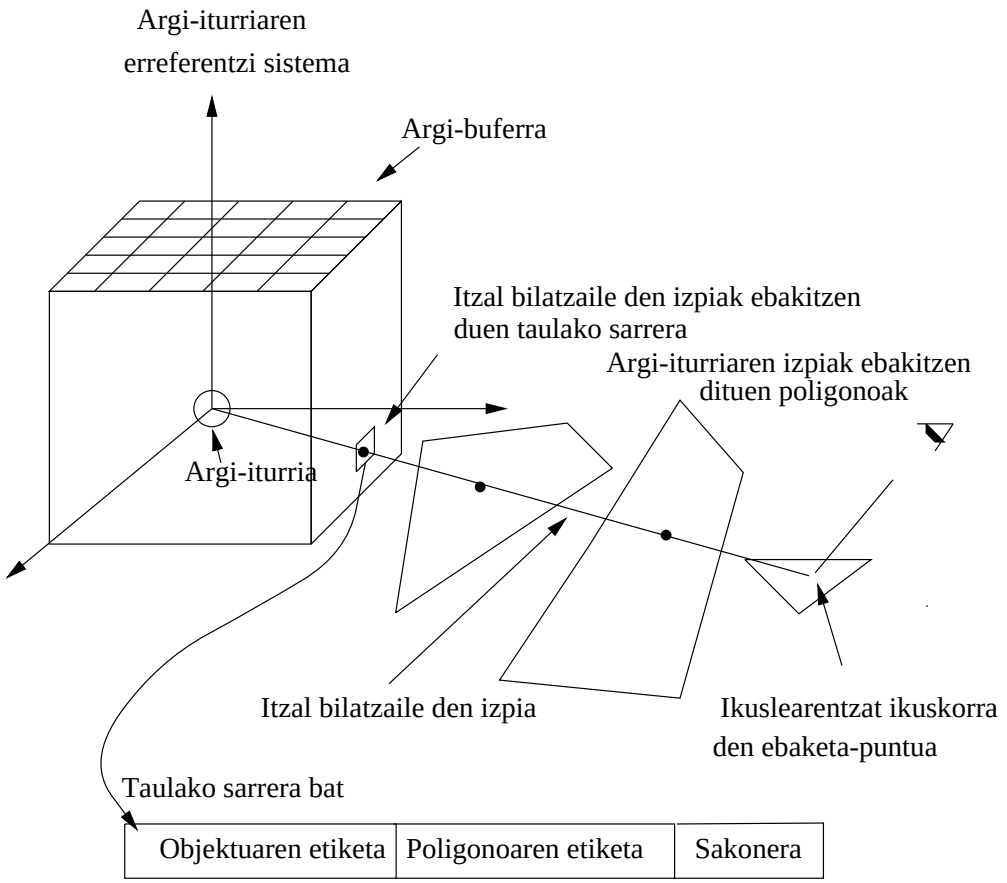
Zenbait eszenatan, eszenako zati batzuek besteek baino sakonera-maila handiagoa behar dute (objektu isladatzaile asko dagoen eszena-zatientzat sakonera-maila handiagoa aukeratzea izango litzateke egokiena). Aurrerago eszenako zati bakoitzari dagokion sakonera-maila egokia kalkulatzeko modua azalduko dugu.

9.7 Itzalak

Azaldutako islada bidezko argiztatze-ereduan itzalak era errazean txertatu daitezke. Izpi-ebaketa bakoitzeko I_{lokala} kalkulatzen da. Aurreko atalean esan bezala, L bektorea oztopa dezakeen objekturik ez dagoela suposatzen da. L delakoa objektu batek oztopatzen badu, tratatzen ari garen puntua itzalpean dago. L oztopatzen duen objekturik dagoenentz jakiteko, beste izpi bat sortzen da; izpi berri hori itzal-bilatzaile izenaz ezagutzen da, eta tratatzen ari garen puntutik argi-iturrira doa. Argi-iturri asko baleude, bakoitzera izpi bat bota beharko litzateke. Izpiak guztiz opakoa den objekturen bat aurkituko balu, puntuaren eta argi-iturriaren artean, I_{lokala} kalkulatzean argi-iturriari dagokion eragina ez luke kontuan hartu beharko. Objektua partzialki gardena bada, I_{lokala}-ren balioa ahuldu egingo da.

Itzal-bilatzailea den izpiak objektu garden bat ebakiko balu, errefraktatu egin beharko litzateke. Baina azaldutako eskema sinplean, ezinezkoa da itzal-bilatzailea den izpia errefraktatzea. Izpi hori, hasiera batean gainazal baten ebaketa-puntutik argi-iturri batera doan lerro zuzen modura definitzen baita, eta zaila izango bailitzateke puntu horretatik argi-iturrira doan izpia hedatzea eta, gainera, errefrakzioa kontuan hartzea.

Argi-iturri anitz baditugu, itzalaren testerako konputazio-kostua, beste izpien kostua adinakoa edo gehiagokoa izatera hel daiteke. Izpi nagusiak bat edo biko sakonera-mailaraino hedatzen dira; baina, izpien eta objektuen arteko ebaketa bakoitzak argi-iturri bakoitzeko izpi bana hedatu behar du, itzaletan dagoen ala ez jakiteko, eta izpi horiekin egin beharreko kalkuluak, beste izpiekin egin behar direnen berdinak dira.



Irudia 9.12: Itzalen kalkulurako erabiltzen diren argi bufferren egitura.

Haines eta Greenberg (1986) argi-buffer bat erabili zuten testa azkartzeko. Horrela, kasu txarrenean 4 aldiz azkartzea lortu zuten; eta kasu onenean 30 aldiz azkarrago bukatu zituzten kalkuluak.

Bufferraren erabileraren azalpena ulergarriagoa izan dadin, 9.12 irudia begiratzea komeni da. Metodo horretan, argi-iturri bakoitzeko zenbait aurre-kalkulu egiten da. Puntuzko argi-iturria barneratzen duen kuboa kalkulatzen da. Kuboaren alde bakoitza 2 dimentsioko taula bat da eta sarrera bakoitza hiru eremutako erregistroz osatzen da:

1. Objektu baten etiketa gordetzen duen eremua.

2. Poligono baten etiketa gordetzen duen eremua.

3. Sakonera gordetzen duen eremua.

Datu-egitura hori hasieratzeko, eszenako poligono guztiak kuboaren alde bakoitzean proiektatzen dira (proiekzio-zentro modura argi-iturriaren kokagunea erabiliz). Argi-bufferreko sarrera bakoitzeko, argi-iturritik ikus daitezkeen poligonoen lista edukiko dugu. Poligono bakoitzaren sakonera kalkulatzeko, argi-iturrian oinarritutako koordenatu-sistema erabiltzen da. Poligonoen lista, sakoneraren arabera, ordena gorakorrean ordenatzen da. Ondorioz, ikuslearengandik irteten den izpi bakoitzeko, berehala lor ditzakegu aztertzen ari garen ebaketa-puntua itzalpean laga dezaketen objektuen aurpegiak.

Itzal-testa egiteko, nahikoa da itzal-bilatzailea den izpiari kuboaren aurpegi baten taulako zein sarrera dagokion jakitea eta ordenatutako poligonoen lista aztertzea. Lista aztertu ondoren, ebaketa-puntua itzalpean jar dezakeen poligonoa aukeratuko dugu. Poligono horren sakonera ebaketa-puntuarena baino handiagoa bada, ez du puntua estaltzen eta puntuak argi-iturriaren eragina jasoko du; baina poligonoaren sakonera puntuarena baino txikiagoa bada, puntua itzalpean egongo da.

Metodo hori aplikatu ahal izateko, izugarrizko memoria behar da. Beharrezko memoria, puntuzko argi-iturri kopuruaren eta argi bufferren bereizmenaren araberakoa da.

9.8 Izpiak desbideratuz

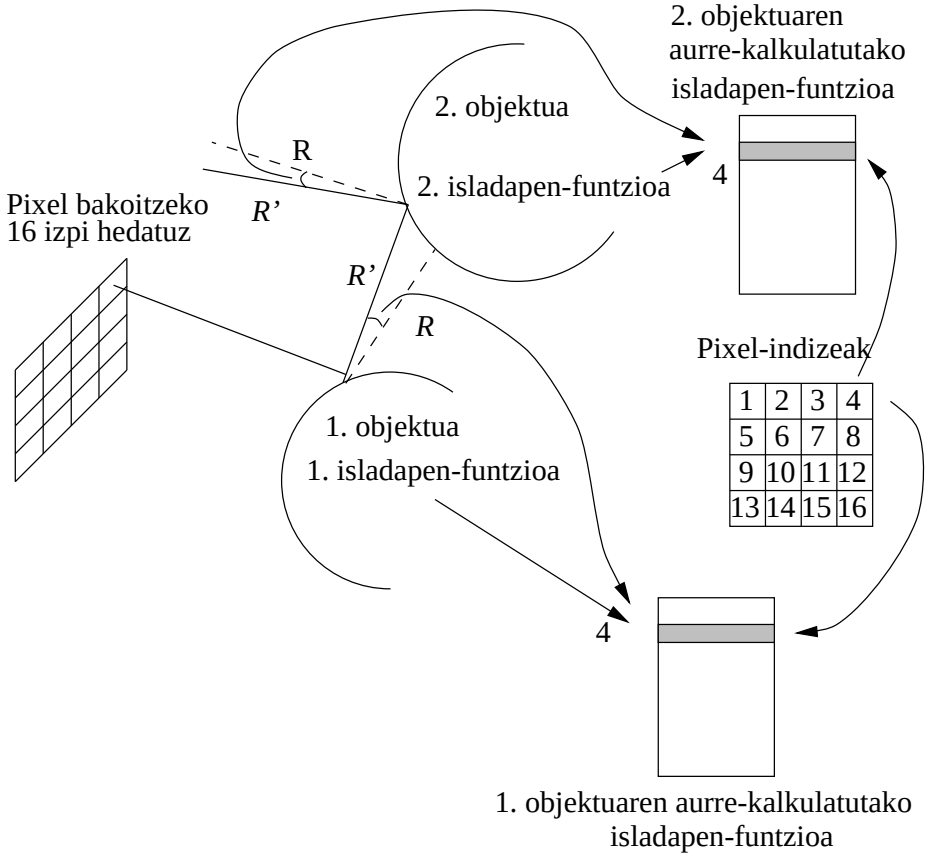
Izpi-hedaketaren algoritmoan, irudiko puntu edo pixel bakoitzeko izpi bat hedatzen da, eta, ondorioz, bata bestearen aldamenean dauden irudiko bi punturen arteko informazioa (irudiko bi punturen edo pixelen artean eszenako infinitu puntu daude) galtzen da. Azken irudia lortzeko, puntu-kopuru finitu batez baliatuko gara; puntu-kopurua, erabilitako bereizmenaren menpe egongo da. Eszenako lagin batez baliatuko garenez, pantailan lortutako irudian azalduko den aliasing efektua nabaria izango da.

Horietako arazo batzuk ekiditeko, lehen mailako izpi-kopurua handiagotzea, edo ebaketa bakoitzerako bi izpi hedatu beharrean gehiago hedatzea izango lirateke soluzio batzuk. Hala ere, soluzio horiek bideraezinak dira, konputazio-kostua gehiegizkoa gertatzen baita. Izpiak desbideratuz (Cook, Porter eta Carpenter (1984)), anti-aliasing efektuaren eta Cook-ek fuzzy fenomenoa deitzen zuenaren arazoak ebazten dira.

Izpi-hedaketaren bidez lortutako irudietan, ideal bezala hartzen diren errefrakzio eta islada globalaren ondorioz, garden edota isladatzaile perfektuak diren objektuak lortzen dira. Baina isladatzaileak diren gainazalek, islada zirriborrotsuak izaten dituzte, gainazalaren perfekzio ezak direla-eta. Era berean, gainazal gardenek irudi zirriborrotsuak azaltzen dituzte, materialaren ezaugarriengatik ematen den sakabanaketaren eraginez. Lehen esan bezala, izpi-hedaketaren izaera hibridoaren ondorioz, zuzeneko argi-iturri batetik sor daitekeen islada, islada zirriborrotsu bihur daiteke. Baina errefrakzio eta islada globala idealak dira. Izaera ideal hori alde batera uzteko, isladatutako eta objektua igarotzen duten izpien norabide ideala, hau da, orain arte kalkulatzen jardun duguna, zertxobait aldatzea proposatu zuten Cook, Porter eta Carpenter-ek. Horrela errefrakzio eta islada zirriborrotsuak lortuko ditugu. Eredu berdinaren bidez efektu berriak lor daitezke: itzal zirriborrotsuak, mugimendu zirriborrotsua, urrutiko objektu zirriborrotsuak eta anti-aliasing-a. Metodo horrek pixel bakoitzeko 16 izpi erabiltzen ditu, eta bere ezaugarri nagusiak ondokoak dira:

- Izpiak desbideratzearen prozesuak, anti-aliasing estokastikoa deritzon prozesuaren parte bihurtzea suposatzen du.
- Isladatutako izpiak desbideratzean, islada zirriborrotsuak azaltzen dira.
- Objektua igarotzen duten izpiak desbideratzean, zeharkako argiztapena azaltzen da.
- Itzal-bilatzaileak diren izpiak desbideratzean, erditzala edo argitzala azaltzen da.
- Izpien jatorria kameraren lentearen azaleran zehar desbideratzean, eremuaren sakonera-efektuak ikus daitezke.
- Denboran zehar izpiak desbideratzean, mugimendu zirriborrotsua lortzen du.

Pentsa dezagun nola tratatuko genituzkeen isladatutako izpiak. 9.13 irudian pixel batetik hedatzen den eta bi objektu topatzen dituen izpia ikus daiteke. Lehen talkaren ondorioz sortuko den izpi isladatuaren norabideak, ez dauka ideala izan beharrik. R-ren norabide ideala zertxobait desbideratuz, islada zirriborrotsuak lor daitezke. Norabide idealaren desbideraketaren eraginez, R desbideratuaren eta objektu baten arteko talka, posizio berri batean



Irudia 9.13: Izpi isladatuen norabide ideala aldatuz aztertutako eszena.

emango da. Talka ematen den posizioa ez da berdina izango izpi isladatu idealarentzat eta izpi desbideratuarentzat.

Objektu bakoitzeko, izpi idealaren desbideraketa itzuliko digun funtzioa izango dugu; funtzio horren balioak taula batean gordeta eduki daitezke. Izpi isladatuaren eta talka egindako gainazalaren normalaren arteko angelua ezagutuz eta objektuaren desbideraketa-funtzioa erabiliz, izpi isladatu desbideratuaren norabide berria kalkulatzeko aukera izango dugu.

Bestalde, pixel bakoitzetik hedatutako 16 izpietako bakoitzak bere garrantzia edo pisua edukiko du; balio hori, 16 izpien kolorea ezagutu ondoren pixelaren kolorea kalkulatzeko erabiliko da. Era berean, izpi bakoitzari bir-desbideratze-balio edo funtzioa ere jar diezaiokegu, eta berak sor ditzakeen izpiei beti aldaketa bera eragin diezaiekegu, guztien arteko batezbestekoa era egokian lortuko baita horrela.

Suposa dezagun lehen mailako izpi bat hedatzera goazela. Izpia pixelaren barneko bigarrena bada, izpi horri eta izpi horretatik islada eta errefrakzioagatik sortutako guztiei taulako bigarren sarrera egokitu beharko zaie. Horrela, izpi baten ondorioz sortuko diren izpi isladatu guztiei, izpi idealaren desbideraketa berdina eragingo diegu, eta lagina egokia dela ziurtasunez esan ahal izango dugu.

Objektu bakoitzak ispilu-isladaren funtzio desberdina eduki dezake. Objektua zeharkatzen duten izpientzat ere inplementazioa berbera izango litzateke, ispilu-isladaren funtzioen ordez errefrakzioaren funtzioak erabiliz.

9.9 Izpi-hedaketa eta anti-aliasing

Izpi-hedaketa, laginketan oinarritzen den metodoa da, eta, ondorioz, aliasing efektuak agertzen dira lortutako irudietan. Bestalde, aliasing-a ekiditeko, ezin daiteke edonolako gain-laginketa egin, horren konputazio-kostua handiegia izan baitaiteke.

Izpiak pixelen izkinetatik heda daitezke, gero pixelaren kolorea izkinetako izpiekin lortutako koloreen batezbesteko modura lortzeko. Teorikoki behintzat, ez da inongo irabazirik lortzen hurbilketa horren bidez; izpi-kopuru berdina hedatzen da eta, aliasing arazoa ebaztearen ondorioz, irudi zirriborrotsuak lortzen dira. Pixel bakoitzeko izpi gehiago hedatzearen hurbilketa garestiagoa da, hasierako exekuzio-denbora zatiketaren heinean biderkatzen baita. 100×100 pixeleko irudi batean izpi-hedaketaren oinarrizko algoritmoa aplikatuz lortzen den exekuzio-denbora unitate batekoa bada, pixel bakoi-

tzeko uniformeki banatutako 4 izpi hedatzen baditugu, exekuzio-denbora 4 unitatekoa izatera pasatuko da. Hurbilketa hobe bat, izkinetako laginak zatiketa kontrolatzeko erabiltzea izan daiteke. Bi izkinen arteko intentsitateen arteko diferentzia aurretik finkatutako balio bat baino handiagoa bada, pixela zatitu egiten da, pixel horrentzat 7 izpi gehiago hedatuz. Baina hurbilketa onena, geroago azalduko den laginketa estokastikoa izan daiteke.

Gain-laginketa ez uniformearen hurbilketa hobea azaldu zuen Mitchell-ek (1987). Metodoak bi ezaugarri garrantzitsu ditu:

- Ingurunearen menpe dagoen hurbilketa da, aliasing-arazo gehien dituzten irudiko zatietan efektu handiagoak lortzen dituena.

- Anti-aliasing modulua, algoritmo aldetik, izpi-hedaketaren modulutik kanpo dago. Beraz, edozein izpi-hedatzailerekin batera erabil daiteke.

Anti-aliasing motako eragiketak hiru urratsetan egiten dira:

1. Izpi-hedaketa lagin txiki batean oinarrituz egiten da; adibidez, pixel bakoitzeko izpi bat hedatuz.

2. Lortutako emaitza, lagin handiagoa erabiliz landu behar diren zatiak ezagutzeko erabiltzen da.

3. Aurreko irudia eta uniformeki banatu gabeko laginak erabiliz, azken irudia kalkulatzen da, iragazki egokiak aplikatuz.

Mitchell-ek bi mailako laginketa-hurbilketa aplikatzea aholkatzen du; bertan gain-laginketa egitean, pixel bakoitzeko 4 edo 9 izpi hedatzen dira. Gain-laginketa egin behar denentz jakiteko, bereizmen txikiko laginak aztertzen dira, ea intentsitate aldaketa handiak ematen diren ikusteko. Mitchell-ek hurbilketa horrek beti ez dituela emaitza onenak lortzen adierazi zuen, berak proposatutako 3×3 neurriko azalerak aztertuz ere informazioa galtzen da-eta.

Azken irudiaren kalkuluan erabili beharreko iragazkiaren aukera ez da erraza; laginaren dentsitatean ematen diren berehalako aldaketetan datza arazoa. Mitchell-ek maila anitzeko iragazkia erabili zuen.

9.10 Izpi-hedaketa eraginkor bihurtzen

Izpi-hedaketaren oinarrizko algoritmoa aplikatzen duen programa batek, denbora luzea igaro dezake kalkuluak egiten, azken irudia lortu arte. Ondoren, eraginkortasuna hobetzeko asmoz garatu diren zenbait teknika azaltzen dira; lehenengoa, sinpleena eta merkeena, ingurune bakoitzerako sakonera-maila egokia kalkulatzen duen algoritmoa da. Ondorengo hurbilketak inplementatzea zailagoa da.

9.10.1 Moldaerazko sakonera-kontrola

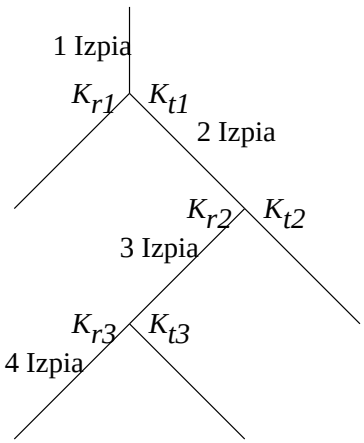
Oinarrizko izpi-hedatzaile batean, sakonera-maila maximoa finkatu behar da. Maila horretara iristean, errekurtsibitateak ez du aurrera egiten. Eszena konkretu bakoitzarentzat sakonera-maila bat finkatu behar da, eta eszenaren ezaugarrien arabera aukeratzea komeni da. Isladatzaile edo gardenak diren objektu asko dituen eszena baten kasuan, sakonera-maila altua beharko litzateke. Sakonera-maila unitate batekoa bada, izpi-hedaketaren algoritmoak errealismoaz arduratzen den beste edozein algoritmok bezala egingo du lan, argiztapen lokala aplikatuz eta ezkutuko gainazalak ezabatuz.

Hall eta Greenberg-ek (1983) diotenez, oso gardenak edo isladatzaile onak diren objektuez osatutako eszena-zatien portzentaia, oso txikia izaten da, eta ondorioz, ez da eraginkorra eszenako pixel guztientzat sakonera-maila berdina erabiltzea. Talka egindako materialaren ezaugarrien arabera, sakonera-maila bat edo bestea erabiltzea egokiagoa da. Sakonera-mailari eman diezaiokegun balioa, aurrez ezarritako maximoraino alda daiteke.

Izpiek eszena zeharkatzen duten heinean, beraien kolorea ahuldu egin daiteke. Izpi bat gainazal batean isladatzean, bere intentsitatea gainazalari dagokion ispilu-islada globalaren koefizientearen eraginez ahultzen da. Izpia gainazal batean errefraktatzerakoan, intentsitatea errefrakzio globalaren koefizientearen eraginez ahultzen da. Beraz, n-ko sakonera-maila kalkulatu ondoren, azken mailako izpien eragina zenbait koefizienteren eraginez ahultzen da.

$$K_{t1}K_{r2}K_{r3}$$

Balio hori finkatutako minimoa baino txikiagoa bada, ez du zentzurik izango izpiak hedatzen jarraitzea, ondoren hedatuko liratekeen izpien eragina arbuiagarritzat har baitaiteke.



Irudia 9.14: Laugarren izpiaren eragina $K_{t1} \times K_{r2} \times K_{r3}$ biderkaketak ahuldu egiten du, pixelaren kolorean duen eragina txikiagotuz.

Orokorrean, izpi bakoitzeko hiru koloreren kalkuluak egin beharko dira (RGB) eta hiru osagaien ahuldura-koefizienteak beharko ditugu.

9.15 sasikodeak puntu bakoitzeko sakonera-maila egokia kalkulatzen duen izpi-hedaketaren prozedura errekurtsiboa azaltzen du. Prozeduraren dei bakoitzean, izpiak pixelean duen pisua adierazten duen parametroa azalduko da. Dei berri bakoitzean pisu berriaren kalkulua egitea oso erraza da, pisua adierazten duen parametroari gainazalaren zeharkatze- edo islada-koefizientea biderkatzea nahikoa baita.

Metodo horri esker, beharrezkoak izango ez diren izpiei dagokien kalkulua egitea ekidingo dugu.

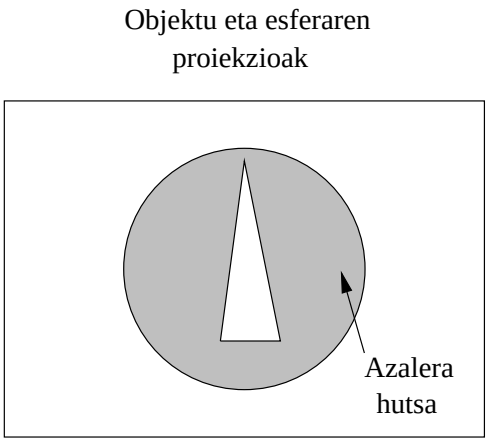
Oso isladatzaileak diren objektuez osatutako eszena batean, 15-eko sakonera maximoa erabiliz eta metodo hori aplikatuz, pixel bakoitzeko batezbesteko sakonera 1.71-koa izatea lortzen dela diote Hall eta Greenberg-ek (1983), konplexutasun handiko kalkulu asko ekidinez. Edozein eszenatan aurreztuko den kalkulua, eszenaren ezaugarrien araberakoa izango da.

9.10.2 Borne-bolumenak

Borne-bolumenak erabiliz lor daitekeen eraginkortasuna, borne-bolumen modura aukeratutako objektuaren eta eszenako objektuen arteko moldaera egokiaren araberakoa da. Esferak oso erabiliak dira borne-bolumen modura,

```
Izpia_hedatu(bektoreak hasierako_puntua,norabidea,
                int sakonera,
                koloreak kolorea,double pisua)
        { bektoreak ebaketa_puntua,islada_norabidea,
                        zeharkatze_norabidea;
          koloreak kolore_lokala,isladatutako_kolorea,kolore_zeharkatua;

          if ((pisua<pisu_minimoa) || (sakonera>sakonera_maximoa))
                kolorea=inguruko_kolorea;
          else
                { /* izpien eta objektu guztien arteko ebaketa-testa burutu
                eta izpiaren hasierako puntutik gertuen dagoen
                ebaketa-puntua (halakorik balego) aurkitu */

                if (ebaketarik ez) kolorea=inguruko_kolorea;
                else
                        { kolore_lokala=(eredu lokalaren eragina ebaketa-puntuan);
                         /* isladatutako izpiaren norabidearen kalkulua */
                         Izpia_hedatu(ebaketa_puntua,islada_norabidea,
                                        sakonera+1,isladatutako_kolorea,
                                        pisua*gainazalaren_islada_koef);

                         /* objektua igarotzen duen izpiaren norabidearen kalkulua */
                         Izpia_hedatu(ebaketa_puntua,zeharkatze_norabidea,
                                        sakonera+1,kolore_zeharkatua,
                                        pisua*gainazalaren_zeharkatze_koef);

                         Konbinatu_koloreak(kolore_lokala,
                             kolore_lokalaren_pisua,isladatutako_kolorea,
                             isladatutako_kolorearen_pisua,
                             kolore_zeharkatua,
                             kolore_zeharkatuaren_pisua)
                        }
                }
        }
```

Irudia 9.15: Izpi-hedaketaren algoritmo eraginkorra.



Irudia 9.16: Esfera borne-bolumen modura erabiltzean barnean egon daitekeen hutsunea.

batez ere izpien eta esferen arteko ebaketa-testaren sinpletasunagatik. Baina ikusi dugunez, objektu luze eta meheen kasuan esferak ez dira oso ondo moldatzen barneko objektuaren itxurari.

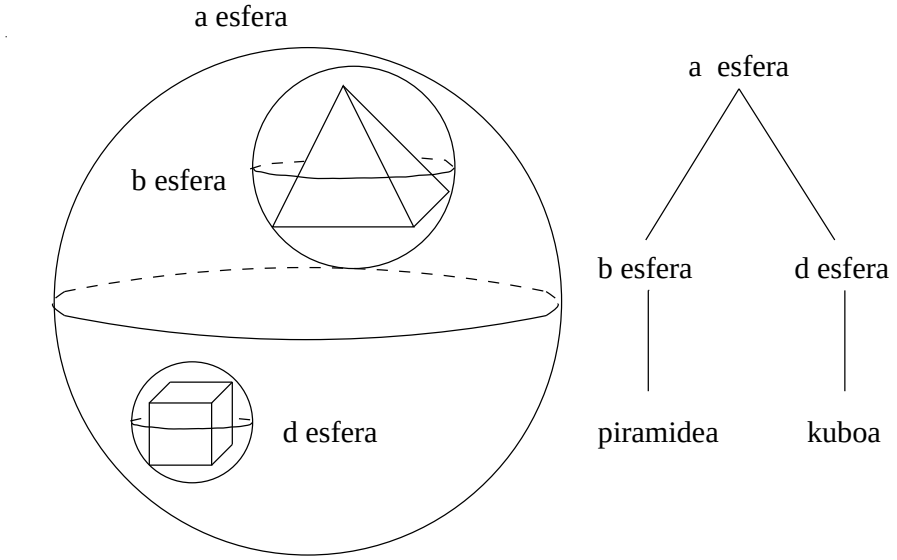
Weghorst, Hooper eta Greenberg-en (1984) iritziz, ebaketa-testaren sinpletasuna, ez da borne-bolumen bat aukeratzearen arrazoi bakarra izan behar. Borne-bolumen baten **azalera hutsa**, objektu eta borne-bolumenak izpiarekiko perpendikularra den eta bere hasierako puntutik pasatzen den planoan duten proiekzio ortogonalen azaleren arteko diferentzia modura definitu zuten.

Azalera hutsa, objektu, borne-bolumen eta izpiaren norabidearen arteko funtzio modura defini zitekeela frogatu zuten. Eta ebaketa-test batentzat kostu-funtzio bat definitu zuten:

$$T = bB + iI$$

T aldagaiaren balioak kostu totala adierazten du; b delakoak ebaketa bakoitzeko borne-bolumenarekin egiten den test-kopurua; B, ebaketa bakoitzeko borne-bolumena aztertzearen kostua da, i, ebaketa bakoitzeko barneko objektua aztertzen den kopurua da; eta I delakoak barneko objektua ebaketa bakoitzeko aztertzearen kostua adierazten du.

Orokorrean, bi biderkaketak independenteak dira. Borne-bolumenaren konplexutasuna txikiagotuz, B txikiagotuko dugu, baina era berean i handiagotuko dugu.



Irudia 9.17: Eszena sinple bati dagokion esfera-hierarkia eta zuhaitza.

Borne-bolumenen hedadura bat, eszenan borne-bolumenen arteko hierarkia bat antolatzea izan daiteke; ahal bada, elkarren artean gertu dauden objektuak multzotan biltzen dira, eta multzoak borne-bolumenetan barneratzen dira.

Objektuak multzotan bilduko dira, baldin eta elkarren arteko distantzia nahiko txikia bada, hau da, hierarkia batean antolatzeak eraginkortasunaren ikuspuntutik abantaila bat badakar. Bestela, ez dira multzotan bilduko, eta objektu bakoitzak borne-bolumen bakar bat izaten jarraituko du. Hierarkia bat antolatzearen ondorio gisa, ebaketa-testa errekurtsibitate bidez egin beharko da, hierarkian zehar mugituz, ebaketa bakoitzean hierarkian behera eginez. Beraz, eszena bat objektuen multzotan banatzen da, eta multzo horiek bata bestetik nahiko urrun egongo dira. Zenbat eta hierarkia sakonagoa, orduan eta denbora aurrerapen handiagoa lor daiteke.

9.17 irudian bi objektu esfera batean barneratu ditugu eta objektu bakoitza, besteetatik nahiko urrun dagoenez, esfera txikiagotan barneratzea erabaki dugu. Izpi askok, esfera handian talka egin arren, ez lukete objekturik ebakiko, eta esfera txikiekin hori era erraz eta azkarrean jakiteko modua edukiko dugu.

Hurbilketa horren desabantaila nagusia, eszenaren ezaugarriekiko duen

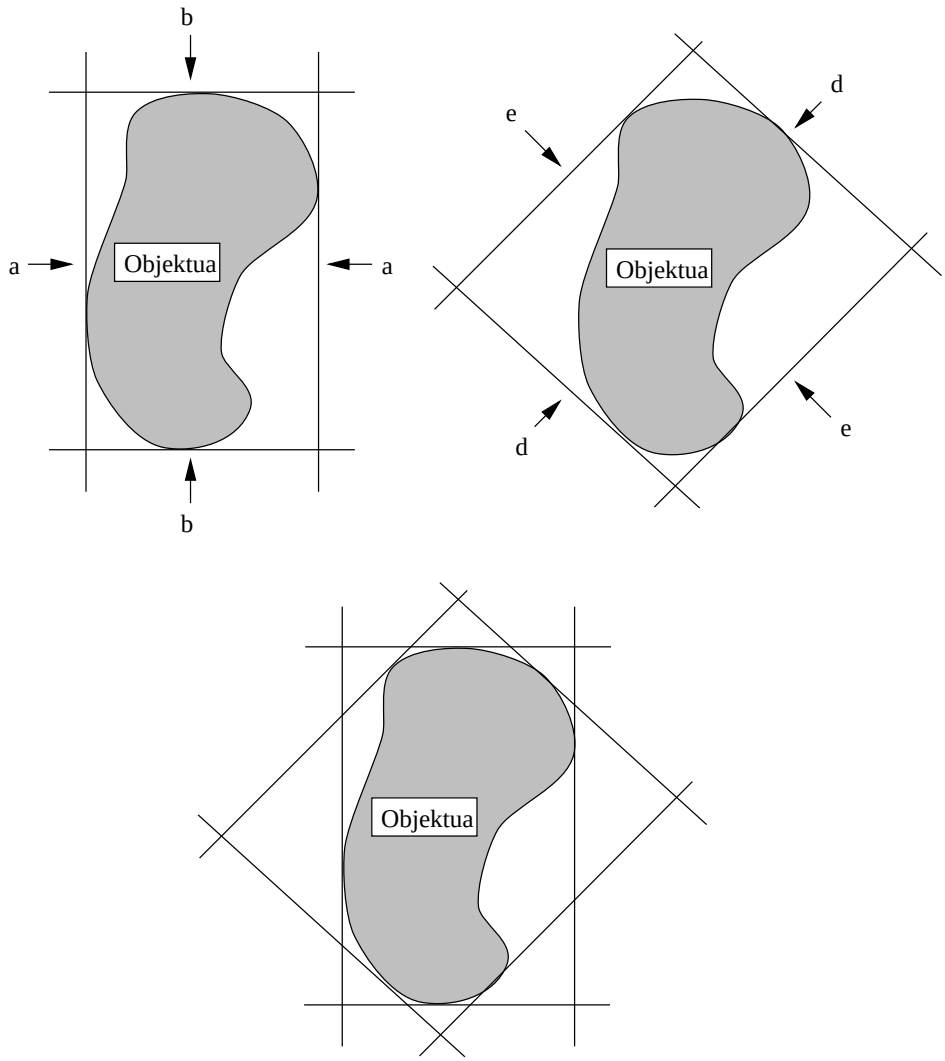
dependentzia da. Erabiltzailearen parte hartzea behar da, hierarkia egoki bat antolatzeko.

Kay eta Kajiya-k (1986) borne-bolumenen mota berri bat azaldu zuten; mota hori objektu konbexoen "zuloei"modu egokian moldatzeko erabil daiteke. Egileek izpien eta borne-bolumenen arteko ebaketa-azterketek kostu txikia zekartela adierazi zuten. Beraz, beraien metodoak azalera hutsaren arazoa ebazten du eta era berean ebaketa-kalkuluen kopuru txikia mantentzen du. Objektuak plano-bikotez osatutako poliedroetan barneratzen dira. Konputazio-kostua eta beharrezko memoria-tamaina handiegia izan ez dadin, planoen norabidea aurretik finkatutako norabide batzuetara mugatzen da. Plano-bikoteen kokapena, objektuak plano-bikotearekiko elkarzuta den proiekzio-plano batean duen hedadura maximo eta minimoaren arabera kalkulatzen da. Borne-bolumen berezi horien erabileraren adibidea 9.18 irudian azaltzen da.

9.10.3 Lehen talkaren azkartzea

Esan dugunez, isladatzailez osatutako eszenetan hedatzen diren izpien batezbesteko sakonera-maila 1 eta 2 artekoa da. Lortzen diren sakonera-maila horiek bultzatuta, Weghorst, Hooper eta Greenberg-ek (1984) izpi-hedatzaile hibrido bat azaldu zuten. Horiek hasierako izpien ebaketak aurreprozesatze bidez lortzen dituzte, ezkutuko gainazalentzat garatutako algoritmo bat aplikatuz; baina, jakina, horrek izpi-hedatzailea baino eraginkorragoa izan behar du. Weghorst-ek Z-buffer algoritmoa, zerbait aldatuta, erabiltzea aholkatzen du. Aldaketa hori, irudi planoko pixel bakoitzeko eszenako objektu batera doan erakusle bat gordetzean datza. Ondoren, sakonera-maila egokiaren kalkuluaren hobekuntzarekin lortutako izpi-hedaketaren algoritmoarekin bigarren mailatik gorako izpiak hedatu beharko dira. Horrela, lehenengo talkarekin ematen diren ebaketa-test garestiak ekiditen dira.

Weghorst, Hooper eta Greenberg-en (1984) emaitzen arabera, metodo horiek aplikatuz (moldaerazko sakonera-kontrola, borne-bolumen hierarkikoak eta lehen talkaren azkartzea) eszenaren konplexutasunarekiko alderantziz menpekoak diruditen hobekuntzak lortzen dira. Nahiko konplexua den eszena batentzako konputazio-kostua, borne-bolumenak erabiltzen dituen izpi-hedaketaren algoritmoarekin lortutako kostuaren erdia da.



Irudia 9.18: Objektuaren proiekzioen hedadura maximo eta minimoen bidez definitutako borne-bolumena, plano-bikotez osatua.

9.10.4 Espazio-koherentzia

Izpi-hedaketa azkartzeko erabilgarria da espazio-koherentzia. Metodoaren izaera dela-eta, edozein norabide duten izpiak eszenako edozein tokitara hedatzen dira. Hori dela-eta, izpi-hedaketan objektuen arteko koherentzia ez da gehiegi erabili, irudi-espazioko algoritmoek erabiltzen duten poligonoen lista ezin baitugu erabili. Izpi-hedaketaren algoritmoan ezinezkoa da lehenengo izpiaren eraginez sor daitezkeen ondorengo izpiak hasieratik ezagutzea. Hasierako izpi-hedaketaren algoritmoak, izpi bakoitzeko objektu guztien azterketa egiten du.

Espazio-koherentziaren atzean dagoen ideia, sinplea da. Eszenari dagokion espazioa, eskualdetan zatitzen da. Gero, izpi bakoitzak objektu edo borne-bolumen guztiekin eduki ditzakeen ebaketak aztertu beharrean, zeharkatzen ari den eskualdean objekturik ba ote dagoen jakin behar dugu. Eskualde bakoitzean zer dagoen badakigu, eta ez da gauza handirik egongo; egin beharrekoa, objektu horien eta izpiaren arteko ebaketa-testak dira. Eskualdeen neurria eta bertako objektuen kopurua, zatiketa egiteko erabilitako metodoaren araberakoa da.

Hurbilketa hori, espazio-koherentzia, espazioaren zatiketa edo espazio-hedaketa izenez ezagutzen da eta batez ere Glassner (1984), Kaplan (1985) eta Fujimoto, Tanaka eta Iwata-k garatu dute. Garatutako hurbilketa guztietan, espazioa aurreprozesatu egiten da, objektu bakoitzak betetzen duen espazioa deskribatzen duen datu-egitura laguntzaile bat hasieratzeko. Gero, izpiak hedatzen dira, eta zein objektu tratatu behar den jakiteko (izpiak zeharkatzen duen eskualdea jakinik), kalkulatutako egitura laguntzailea erabiltzen da. Ingurunea aurreprozesatzearen filosofia, lehen aldiz Schumaker-ek *et al.* (1969) erabili zuen. Schumaker-ek garatu zuen algoritmoan, eszenako objektuak multzotan taldekatzen ziren, eta espazioaren zatiketa planoak erabiliz burutzen zen. Espazioaren zatiketa zuhaitz bitar batez adierazten zen. Eskualde bateko edozein ikuspunturi zuhaitzaren hosto bat zegokion. Hosto batean kokatu ondoren, zuhaitzean zehar (aukeratutako hostotik abiatuz) gorantz mugituz, taldekatutako objektuak sakoneraren araberako lehentasunarekin lor zitezkeen. Algoritmo horretan espazioaren zatiketa behin egin ondoren (datu-egitura hasieratzerakoan), edozein ikuspuntutatik ikus daitekeena erabaki daiteke. Zuhaitz bitarraren bidez, objektu-talde edo multzoen ordenaren araberako lehentasuna jakin daiteke. Ikuspuntu batetik abiatuta, objektu-taldeak ikuspuntura dagoen distantziaren arabera ordenatuta izango ditugu. Algoritmo horren helburua, ezkutuko gainazalen ezabapena

azkartzea zen, irudiak denbora errealean sortu ahal izateko.

Espazio-koherentziaren garapena, borne-bolumenen edo izpien eta objektuen arteko testak ekiditeko hedatutako beste hurbilketen emaitza txarren edo oso onak ez diren emaitzen eraginez eman da batez ere. Dagoeneko aipatu da borne-bolumenen arazo nagusia. Beraien eraginkortasuna objektura moldatzeko duten gaitasunaren araberakoa da. Bestalde, borne-bolumenek izpien eta objektuen arteko ebaketen eraginkortasuna handiagotu dezakete (ebaketa asko ekidinez), baina eszenako objektu-kopuruaren menpekotasun handia daukate. Izpi bakoitza objektu guztien borne-bolumenarekin aztertu behar da, eta azterketa hori egiteak eskatzen duen denbora, eszenaren konplexutasunaren araberakoa da. Nahiz eta borne-bolumenak modu hierarkikoan antolatuz aurrerapen nabariak lortu, hierarkia egoki bat antolatzeak esfortzu handia eska dezake. Eta zenbait kasutan, objektuen ezaugarri eta kokapenengatik, hierarkia egokia antolatzea zaila edo ezinezkoa gerta daiteke. Atal honetan deskribatu diren metodoen berriztapen nagusia, azken irudiaren kalkulua konstantea eta eszenaren konplexutasunarekiko askea izatean datza.

Espazio-koherentziaren hurbilketa erabiltzen duten eskemen arteko desberdintasun nagusia, erabilitako datu-egitura laguntzailean aurkitzen da.

Datu-egitura laguntzaileen artean bi egitura dira nagusi. Lehenengoa octree adierazpenean oinarritzen da eta bigarrena BSP (Espazioaren partiketa bitarra, binary space partition) deituriko datu-egituraz baliatzen da.

Octree-egituraren erabilera

Octree bat, eszenako objektuak adierazteko erabiltzen den datu-egitura da. Horren bitartez espazio-koherentzia aplikatu ahal izango dugu. Espazioan bata bestetik gertu dauden bi objektu, octree-egituran gertu egongo diren bi erpinen bitartez adieraziko dira.

Izpi bat hedatzean, eszenako objektu guztiekin ebaketa-kalkuluak egin beharrean, izpiak zeharkatuko dituen eskualdeak jarraituko ditugu. Zeharkatzen duen eskualde bakoitzean, objektu gutxi egongo dira, bi edo hiru normalean, eta horiekin egin beharko dira ebaketa-kalkuluak. Eskualde bati dagokion erpina azkar aurki dezakegula suposatuz, bertan dauden objektuak, izpiarekin talka egingo dutenak edo bere norabidetik nahiko gertu egongo direnak, ziztu bizian atzitzeko ahalmena izango dugu. Espazioaren zatiketaren bereizmena nahiko handia bada, eskualde bakoitzeko egin beharko dugun ebaketa-testen kopurua txikia izango da, eta kopuru hori ez da handiagotuko

eszenaren konplexutasunarekin batera.

Izpia octree-egituran zehar

Izpiak espazioko zein eskualde zeharkatzen duen jakiteko gai izan behar dugu, gero eskualde horretan dauden objektuekin ebaketa-testak egiteko. Horrek eskualde batean izpia noiz eta nondik sartzen eta ateratzen den jakitea eskatzen du. Prozesu horretako eragiketa garrantzitsuena, (x, y, z) espazioko puntu bati octree-egituran dagokion erpin egokia aukeratzea da, eta beraz espazioko eskualde egokia aukeratzea.

Jarraipen-prozesu horren hasieran, izpiaren hasierako puntuari dagokion eskualdea zein den jakin beharko genuke. Hasierako eskualdean egon daitezkeen objektuekin ebaketa-testak egin beharko lirateke, eta ebaketarik balego, aurkitutako lehenengo objektua izango litzateke bilatzen ari ginena, hori izango baita izpiak bere norabidea jarraituz aurkitzen duen lehenengo objektua. Hasierako eskualde horretan ebaketarik ez balego, izpiak zeharkatuko lukeen hurrengo eskualdea zein izango litzatekeen kalkulatu beharko genuke. Horretarako, eskualdearen bornearen eta izpiaren arteko ebaketa-testak kalkulatu beharko lirateke. Horrela, izpia eskualdearen zein puntutik ateratzen den jakingo genuke. Gero, izpiak zeharkatuko lukeen hurrengo eskualdeari octree-egituran dagokion erpina identifikatzeko, izpiaren norabidean eta eskualde horretako hasierako puntutik gertu egongo litzatekeen puntu bat aukeratuz kalkulatuko genuke. Eskualde berri horretan objekturik balego, ebaketa-testa aplikatuko genuke. Prozesua eskualdetik eskualdera errepikatuko genuke, objektu batekin talka egin arte edo octree-egiturak adierazten duen espaziotik atera arte.

(x, y, z) puntu bati dagokion erpina kalkulatzeko erabil daitekeen hurbilketa sinpleenean, datu-egitura laguntzaile bat erabiltzen da. Datu-egitura horren bitartez, octree-zuhaitzeko erpin bakoitzak espazioko zein eskualde mugatzen duen jakingo genuke; horrela, puntu bat eskualde batean kokatzeko, nahikoa izango litzateke, zuhaitzaren errotik hasi eta erpin bakoitzarentzat puntua eskualdearen barnean dagoenentz begiratzea, konparazio sinpleak eginez. Nahiz eta puntua eskualde batean egon, erpinaren "semeak"aztertu beharko lirateke, eskualde bat azpieskualdetan banatuta egon baitaiteke. Horrela jarraituko genuke, hosto bateraino iritsi arte. Igarotako erpin-kopuru maximoa, zuhaitzaren sakoneraren berdina izango litzateke. Bereizmen handiko zatiketen kasuan, bilaketa-luzera edo igarotako erpin-kopurua txikia izango da. Adibidez, espazioaren zatiketaren bereizmena

$1024 \times 1024 \times 1024$ bada, octree-egituraren sakonera 10ekoa ($=\log_8(1024 \times 1024 \times 1024)$) izango da.

Azaldutakoa, espazioaren zatiketa adierazteko hurbilketa sinple bat da. Oinarrizko hurbilketa horren bi aldaketa deskribatu zituzten Glassner-ek (1984) eta Fujimoto, Tanaka eta Iwata-k (1986). Glassner-ek (x, y, z) puntu bati dagokion erpina aurkitzeko beste metodo bat deskribatu zuen. Glassner--ek ez du octree-egitura esplizituki gordetzen; voxel desberdinen informazioa atzitzeko, voxel bakoitzeko sarrera bat duen hash-taula bat erabiltzen du. (x, y, z) puntu bakoitzari kode bat egokitzen dio taulan, atzitu beharreko sarrera kalkulatzeko. Beste guztia, hau da, izpiaren jarraipena, aurreko metodoan bezala egiten da.

Fujimoto, Tanaka eta Iwata-k (1986) berriz, puntu higikorreko biderkaketa eta zatiketak ekiditeko metodo bat deskribatu zuten. Metodoa ulertzeko, octree-egituraren adierazpena ahaztea komenigarria da. Espazioaren zatiketaren adierazpenerako erabiltzen den datu-egitura, SEADS (spatially enumerated auxiliary data structure) izeneko datu-egitura, azalduko dugu. Eszenako objektuek betetzen duten espazio osoa, neurri berdineko voxeletan zatituko dugu, objektuek betetzen duten eskualde-bolumena kontuan hartu gabe. Modu horretan lortzen den hiru dimentsioko sarea, bi dimentsioko irudia pixeletan zatitzean sortzen den sarearen parekoa da. SEADS zatiketak, octree-zatiketak baino askoz ere voxel gehiago sortzen ditu, zatiketa egitean ez baita objektuek betetako bolumena kontuan hartzen. Beraz, erabilgarria ez den datu asko gordeko dugu memorian. Baina SEADS erabiliz, izpiaren jarraipena oso azkarra izango da. Erraza izango da izpiaren ibilbidea jarraitzea. Erabilitako jarraipen-algoritmoa DDA (digital diferential analyzer) algoritmoaren hedaketa da. DDA algoritmoaren bitartez, irudiko bi pixel lotzen zituen zuzenaren pixel-sekuentzia kalkulatzen genuen. Algoritmo hori zuzen batek ukitzen dituen pixel guztiak kalkula ditzan alda daiteke. Fujimoto-k algoritmo hori hiru dimentsioko espaziora hedatzeko era deskribatu zuen, ondoren hiru dimentsioko sare baten kasuan aplikatzeko, SEADS kasuan hain zuzen ere. 3D-DDA algoritmoan erabiltzen diren eragiketa bakarrak, batuketa, kenketa eta konparazioa dira; eta gehien egiten dena, batuketa da. Beraz, ez da biderkaketa eta zatiketarik erabiltzen, eta hori abantaila garrantzitsua da konputazio-kostuaren ikuspuntutik.

SEADS erabili beharrean, memoria gutxiago erabiltzeko, octree-adierazpenera bueltatuko gara. 3D-DDA algoritmoa alda daiteke, izpi baten jarraipena egiterakoan octree-egitura erabil dezan. Octree-adierazpenean "guraso"bera duten zortzi erpinetako multzoak, auzokide diren eta $2 \times 2 \times 2$ neu-

rriko sare bat osatzen duten zortzi eskualde kubikoren bloke bat adierazten du. Izpi bat multzo horretako eskualde batetik bestera jarraitzean, 3D-DDA algoritmoa erabil daiteke. Izpia hosto bat ez den eta "semeak"dituen erpin batez adierazitako eskualde batean barneratzen bada (azpieskualdez osatutako eskualde batean), eskualde berriari dagokion erpina aurkitzeko, nahikoa da zuhaitzean beherantz jotzea. Izpia $2 \times 2 \times 2$ eskualdez osatutako azpieskualdetik ateratzean, octree-egituran gorantz egin beharko dugu, berriro "gurasora"helduz. Ondoren, "gurasoa"daukan $2 \times 2 \times 2$ eskualdeko zein eskualde zeharkatuko dugun kalkulatu beharko da. Zuhaitzaren egituran gora eta behera mugitzeak, DDA algoritmoko kontrol-aldagaiak 2 balioaz zatitzea eta biderkatzea dakar. Eragiketa hori oso merkea da, bit desplazamendua baita.

Kapitulua 10

Erradiositatea

10.1 Sarrera

Izpi-hedaketaren metodoa erabiliz lortzen diren ispilu-islada eta gardenkiak oso errealistak dira. Hala ere argi globala kalkulatzean ingurune-argiaren koefizientea erabiltzen da, hau da, objektuen arteko argi-elkarreragiteak modelatzeko, koefiziente hori behar da. Hori bera egiteko, ingurune itxi bateko argi energiaren kontserbazioan oinarritzen den beste hurbilketa bat badago; gainera, zehatzagoa da.

Hurbilketa zehatz hori aplikatzeko, eszenako gainazal bakoitzak isladatzen eta sortzen duen argiaren energi kantitatea zein den jakin behar dugu. Gainazal-unitateko irteten den energi kantitateari **erradiositatea** deritzo, eta gainazalak isladatzen eta sortzen dituen energien batura da. Horregatik, erradiositatea konputatzen duten hurbilketei, erradiositate-metodo deritze. Erradiositate-metodoak argiaren elkarreragite guztiak kalkulatzen ditu, ikuslearen kokapena kontuan hartu gabe, horrek ez baitu inongo eraginik. Beraz, argi-elkarreragite guztiak kalkulatu ondoren, argiztapena ez da berriro kalkulatu beharko, ikuslea edonon dagoela ere gainazal bakoitzaren kolorea berdina izango baita. Ondorioz, eszenako irudia lortzeko eginbehar bakarrak, ikuskorrak diren gainazalak identifikatzea eta gainazal horiek interpolazioz itzaleztatzea izango lirateke.

10.2 Erradiositate-ekuazioa

Erradiositatearen filosofian ez da argi-iturri eta gainazalen arteko desberdintasunik egiten, gainazalek argia eman dezaketelako. Suposa dezagun eszena osoa azalera finituko txataletan zatitzen dugula. Txatal horiek azalera guztian zehar, argia uniformeki isladatzen eta ematen duten zati txikitzat hartu behar dira. Txatal bakoitzaren erradiositatea honako hau da:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{j-i} \frac{A_j}{A_i} \tag{10.1}$$

- E_i delakoa i gainazalak sortutako argia da, eta erradiositatearen unitate berdina du. Beste batugai guztiak gainazalak isladatzen duen argia adierazten du; jakina, argi hori eszenako edozein gainazaletatik hel dakioke.

- B_i eta B_j direlakoak i eta j txatalen erradiositatea adierazten duten aldagaiak dira, eta azalera-unitateko energi kantitatea neurtzen dute (W/m^2).

- ρ_i parametroak i txatalak jasotako "argi-unitateko"zenbat argi isladatzen duen adierazten du; ez du unitaterik, magnitude erlatiboa baita. Txatalaren islada-maila da.

- F_{j-i} balioak unitaterik ez du eta forma-faktore deritzo. j txatala uzten duen argi-unitate bakoitzeko i txatalera heltzen den argi-kantitatea adierazten du. Txatal batek bestetik jasoko duen eragina finkatzen du, eta hori kalkulatzeko, txatal bakoitzaren itxura, orientazioa eta bi txatalen artean beste txatalik dagoenentz jakin beharko dugu.

- A_i eta A_j aldagaiek txatal bakoitzaren azalera adierazten dute.

Azalera-unitateko irteten den energia, B_i, azalerak isladatzen eta sortzen dituen energien arteko batura da. Isladatutako argia kalkulatzeko, eszenako txatal guztietatik heldutako argi guztia kontuan hartu behar da. Heldutakoaren zati bat isladatu egingo da; hori ρ_i parametroak kontrolatzen du eta txatalaren ezaugarrien araberakoa da. $B_j F_{j-i}$ espresioak, j txataleko azalera-unitate bakoitzeko A_i guztira heltzen den argi-kantitatea adierazten du; beraz, j txatal osotik i txatal osora heltzen den argia zenbatekoa izan

den jakiteko, A_j biderkatu behar zaio. Baina guri interesatzen zaiguna i txatalaren azalera-unitateko argia da, eta ez azalera osokoa; beraz, A_i balioaz zatitu behar dugu.

Baina forma-faktoreen propietate bat honako hau da:

$$A_i F_{i-j} = A_j F_{j-i} \tag{10.2}$$

Beraz, 10.1 ekuazioa sinplifikatuz:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{i-j}$$

Azkenean, honela idatz dezakegu:

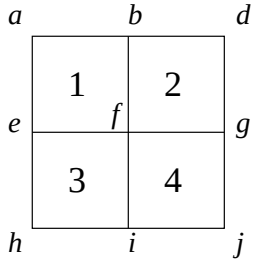
$$B_i - \rho_i \sum_{j=1}^{n} B_j F_{i-j} = E_i$$

Ingurunean ematen den argiaren eta txatalen arteko elkarreragina ekuazio-multzo batez idatz daiteke:

$$\begin{pmatrix} 1-\rho_1 F_{1-1} & -\rho_1 F_{1-2} & \dots & -\rho_1 F_{1-n} \\ -\rho_2 F_{2-1} & 1-\rho_2 F_{2-2} & \dots & -\rho_2 F_{2-n} \\ \vdots & \vdots & & \vdots \\ -\rho_n F_{n-1} & -\rho_n F_{n-2} & \dots & 1-\rho_n F_{n-n} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix} \tag{10.3}$$

Ekuazioetan azaltzen den bezala, txatal batek bere burua argi dezake, i txatalak isladatutako argiak i txatala jo dezake (txatala ahurra denean adibidez). Kasu orokorrean matrizeko diagonalak ez dauka batekoz osatuta egon beharrik. Matrizea hiru koloreen osagaientzat askatu behar da, ρ_i eta E_i balioek kolore osagai bakoitzarentzat (RGB) balio desberdina izan baitezakete. Forma-faktoreak, berriz, uhin-luzerekiko (koloreekiko) independenteak dira; horregatik, nahiz eta argien kolorea aldatu, ez dira birkalkulatu behar. Matrizea askatu ondoren, txatal bakoitzari dagokion kolorea egokitzeko gai izango ginateke, baina emaitza hobeak lortzeko, komenigarria izango litzateke txatalen erpinen erradiositatea kalkulatzea (txatalaren erradiositatea kalkulatu ondoren), gero erpinen kolorea txatalean zehar interpolatzeko.

Erpin bakoitzaren erradiositatea kalkulatzeko, algoritmo asko jarrai daitezke. Adibide gisa, ondorengo arauak jarraitzen dituen metodoa azalduko



Irudia 10.1: Txatalen erradiositatea ezaguna izanik, erpinen erradiositatearen kalkulua.

dugu: erpin batek, gainazalaren barnealdean badago, txatal batean baino gehiagotan hartuko du parte, eta horrelakoei txatal horien erradiositateen batezbestekoa egokitzen zaie. Erpina gainazalaren ertz batean badago, gertuen dagoen barnealdeko j erpina aurkitu behar da. Bi erpin horien erradiositatearen batezbestekoak, j erpina daukaten txatalen erradiositatearen batezbestekoaren berdina izan behar du.

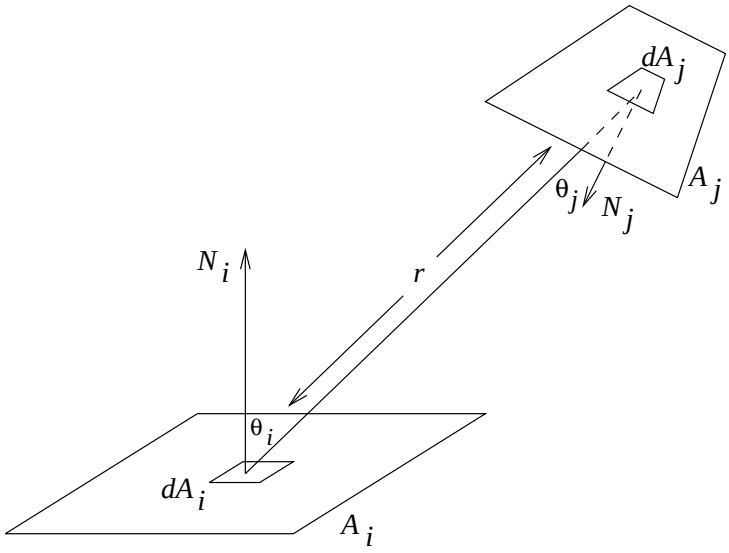
10.1 irudiaren kasuan, erpin bakoitzaren erradiositatea kalkulatzeko ondoko urratsak jarraitu behar dira. Barnealdeko f erpinari dagokion erradiositatea $B_f = (B_1 + B_2 + B_3 + B_4)/4$ izango da. Gainazalaren ertz batean dagoen b erpinaren erradiositatea kalkulatzeko, lehendabizi barnealdean eta b-tik gertuen dagoen erpina aurkitu behar da; kasu horretan f. Eta $(B_b + B_f)/2 = (B_1 + B_2)/2$ denez (algoritmoa aplikatuz), B_b askatzeko gai gara: $B_b = B_1 + B_2 - B_f$. Bestalde a-tik gertuen dagoen barnealdeko erpina f da, eta a erpina 1 txatalaren parte denez, $(B_a + B_f)/2 = B_1$; askatuz $B_a = 2B_1 - B_f$.

10.3 Forma-faktoreak kalkulatzen

Forma-faktoreak kalkulatzeko metodo desberdinak daude. Ondoren batzuk azalduko ditugu:

- Kontsidera ditzagun 10.2 irudian azaltzen diren bi txatalak.

  dA_i azalera diferentzialetik dA_j azalera diferentzialera heltzen den erradiositatea kalkulatzerakoan erabili beharko genukeen forma-faktorea hauxe da:



Irudia 10.2: Txatal baten forma-faktoreari dagokion balioaren kalkulua.

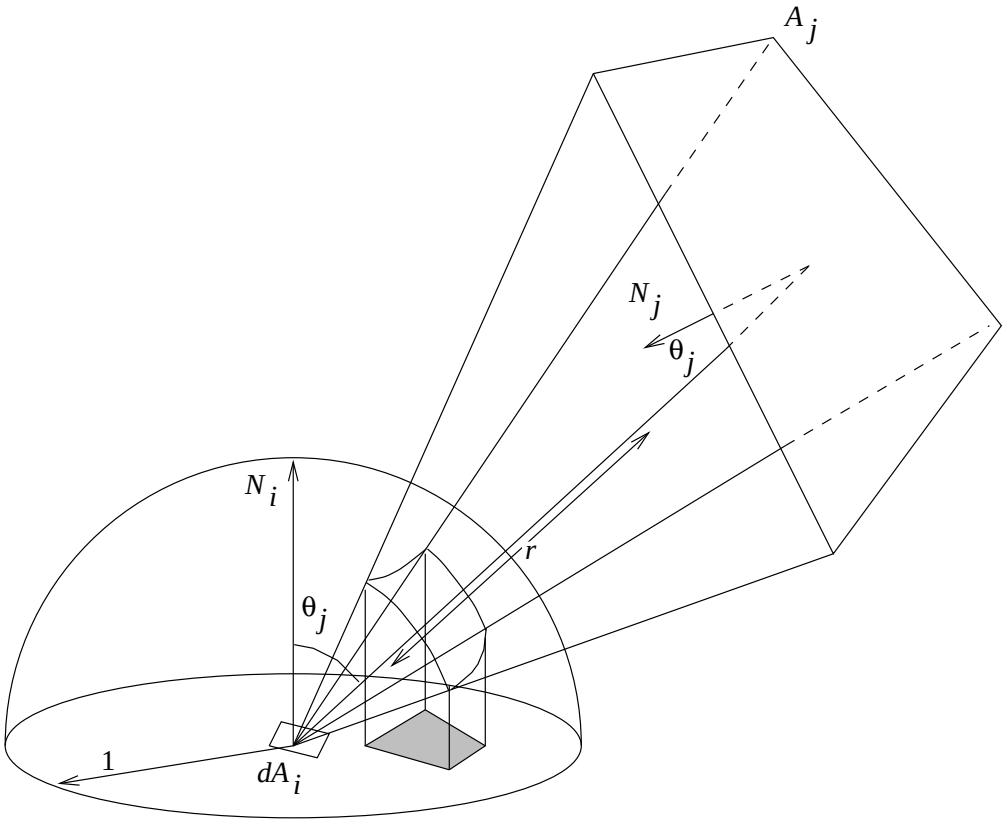
$$dF_{di-dj} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij} dA_j$$

dA_i eta dA_j azalera diferentzialen arteko izpiak A_i azaleraren bektore normalarekin osatzen duen angelua θ_i da, A_j-ren bektore normalarekin osatzen duena θ_j, eta dA_i eta dA_j elementuen arteko distantzia r. H_{ij}--ren balioa 1 edo 0 izan daiteke, 1 dA_j dA_i-tik ikuskorra bada; eta bestela, 0. F_{di-j} kalkulatzeko, j txatalaren azaleran zehar integratu beharko dugu:

$$F_{di-j} = \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij} dA_j$$

A_i-tik A_j-rako forma-faktorea, i txatala zeharkatuz, aurreko ekuazioaren azaleraren batezbestekoa izango da:

$$F_{i-j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij} dA_j$$

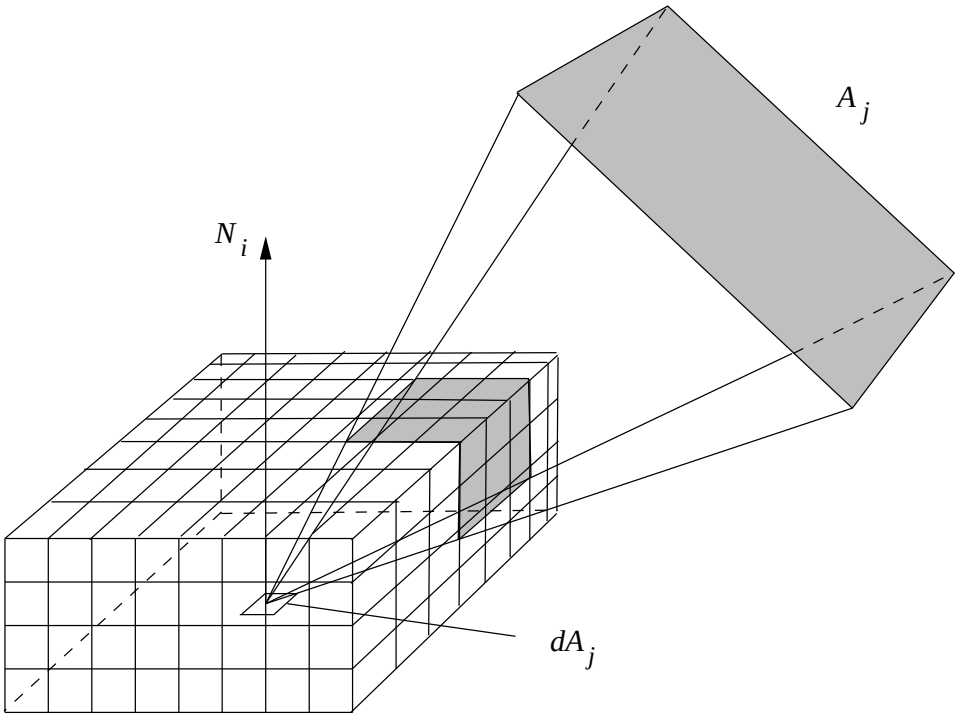


Irudia 10.3: Txatal eta azalera diferentzial baten arteko forma-faktorearen kalkulua. Forma-faktorea, esferaerdiaren eta bere oinarrian proiektatutako azaleraren arteko arrazoia da.

Txatal baten erdiko puntuak gainontzeko puntuak adierazten dituela suposatzen badugu, F_{di-j} delakoa txatalaren erdiko puntuan kalkulatuz, F_{i-j} ezagutu dezakegu.

Frogatuta dago, F_{di-j} konputatzea eta 10.3 irudian azaltzen dena kalkulatzea baliokide direla. Hau da, baliokidea dela ondoko kalkulua egitea: dA_i-tik ikuskorrak diren A_j-ko zatiak esferaerdi berezi batean proiektatu, ondoren, esferaerdian proiektatutakoa bere oinarrian ortogonalki proiektatu eta, azkenik, zirkunferentziaren azaleraz zatitzea. Esferaerdi berezi horren zentroa dA_i da, eta erradioa 1.

Horrelako esferaerdi baten gaineko proiekzioa $\cos\theta_j/r^2$-ren bidez kalku-



Irudia 10.4: Kuboerdia, txatalaren zentroan kokatzen den kubo baten goialdeko zatia da.

latzen da; eta oinarrian proiektatzeko, $\cos\theta_i$ balioaz biderkatzearekin nahikoa da. Azkenik, erradioa unitatekoa duen zirkunferentzia baten azalera π denez, π balioaz zatitzen da.

- Txatal bakoitza esfera baten gainean analitikoki proiektatu beharrean, kuboerdi batean proiekta dezakegu, 10.4 irudian ikus daitekeen bezala.

  Txatal batean zentratutako kuboerdi bat erabiliko dugu. Bere aurpegiak azalera berdineko karratu edo laukitan zatituko ditugu, eta horien neurri edo kopuruak bereizmenean eragina izango du; hau da, zenbat eta gehiagotan zatitu, orduan eta emaitza hobeak lortuko dira. Forma-faktoreak kalkulatzeko, eszenako txatalak dagokien kuboaren aldean proiektatu behar dira eta karratutxo bakoitzean gertuen dagoen txatalaren adierazlea gordeko dugu; horrez gain, karratutxoei forma-faktore bana egokituko diegu. Horrela, informazio hori erabiliz, j txatalari

dagokion F_{i-j} lortzeko j txatala erakusten duten karratutxoen forma--faktoreen batuketa egitearekin nahikoa izango da.

Kuboerdiko p karratu bakoitzari dagokion forma-faktorea kalkulatzeko, ondoko ekuazioa erabil dezakegu:

$$\Delta F_p = \frac{\cos\theta_i \cos\theta_p}{\pi r^2} \Delta A \tag{10.4}$$

non θ_p angelua p karratuaren normalaren eta p-tik dA_i-ra doan bektorearen arteko angelua, r delakoa bektore horren luzera, eta ΔA delakoa karratuaren azalera diren, 10.5 irudian ikus daitekeenez. Irudi horretan kuboerdiko goiko aurpegiko edo alboetako aurpegietako karratutxoen forma-faktoreak kalkulatzeko behar diren datuak azaltzen dira. Datu horiek darabilten erreferentzi sistemaren jatorria txatalaren zentroa da, eta z ardatza txatalarekiko elkarzuta da. Hori horrela izanik, 10.5 irudiko lehenengo kasurako, hau da, kuboaren goiko aurpegiko karratuen forma-faktoreen kasurako, datuak honako hauek izango dira:

$$r = \sqrt{x_p^2 + y_p^2 + 1}$$

$$\cos\theta_i = \cos\theta_p = \frac{1}{r}$$

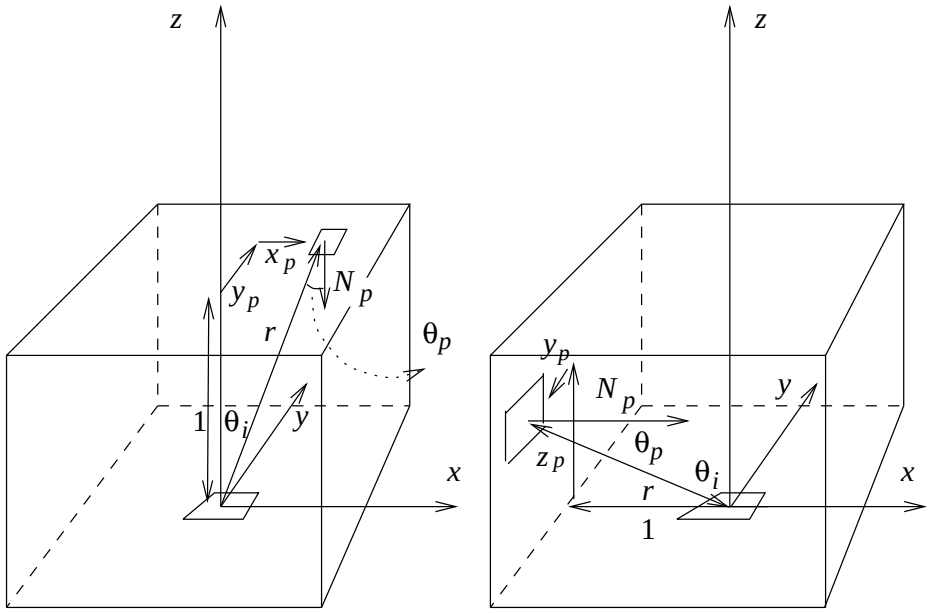
x_p eta y_p kuboerdiaren goiko aurpegiko karratuaren koordenatuak dira. Datu hauek 10.4 ekuazioan ordezkatuz, ondoko ekuazioa lortuko dugu karratutxo bakoitzaren forma-faktorearentzat:

$$\Delta F_p = \frac{1}{\pi(x_p^2 + y_p^2 + 1)^2} \Delta A$$

Kuboerdiaren x ardatzarekiko perpendikularra den aurpegi baten kasuan, hau da, kuboerdiaren alboetako karratuen kasuan:

$$r = \sqrt{y_p^2 + z_p^2 + 1}$$

$$\cos\theta_i = \frac{z_p}{r}$$



Irudia 10.5: ΔF_p edo forma-faktoreen gehikuntzaren kalkulua, kuboaren aurpegi horizontal eta bertikalaren kasuetan.

$$\cos\theta_p = \frac{1}{r}$$

Eta 10.4 ekuazioan ordezkatuz:

$$\Delta F_p = \frac{z_p}{\pi(x_p^2 + y_p^2 + 1)^2}\Delta A$$

10.4 Txatalak azpitxataletan zatitzen

Izpi-hedaketaren kasuan, errekurtsibitate-maila handiagotzeak kalkulu-kopurua handiagotzea zekarren. Erradiositatearen kasuan forma-faktoreen kalkulua da konputatze-kostua handitzen duena; txatal asko badaukagu, kalkulu-kopuru handia egin beharko da. Zenbat eta txatal gehiago erabili orduan eta emaitza hobeak lortuko ditugu, baina azken irudia lortzeko itxaron beharko dugun denbora, orduan eta handiagoa izango da.

Erradiositate-metodoan, erradiositate-aldaketa handia duten txatal-sareetan egin beharko dugu txatal-zatiketa, txatal-sareko txatal bakoitza azpitxataletan zatituz. Zatiketa hori egitean, zenbait ideia argi edukitzea komeni da:

1. Azpitxatal bakoitza ez da benetako txataltzat hartzen. Badakigu txatalen kopuruaren arabera $n \times n$ neurriko ekuazio-sistema askatu behar dela, eta azpitxatalekin ekuazio-sistema asko handituko litzateke; horregatik, era berezian tratatzen dira.

2. Txatala azpitxataletan zatitzean, azpitxatal bakoitzak eszenako beste txatal bakoitzarekiko duen F_{s-j} forma-faktorea, kuboerdiaren teknika erabiliz kalkulatu beharko da.

3. Eszenako txataletatik azpitxatal berrietara doazen forma-faktoreak ez dira kalkulatu behar.

4. i txatala zatitu eta s azpitxatalen forma-faktoreak kalkulatu ondoren, i txataletik eszenako txatal guztietara (azpitxataletara ezik) doazen forma-faktoreak birkalkulatu beharko dira. Oraingoan forma-faktore zehatzagoak kalkulatuz, azpitxatalen forma-faktoreak erabiliz:

$$F_{i-j} = \frac{1}{A_i} \sum_{s=1}^{m} F_{s-j} A_s$$

   Azpitxatal bakoitzaren forma-faktoreari, azpitxatalaren azalerarekiko proportzionala den garrantzia emango zaio: A_s garrantzia dagokio bakoitzari.

Forma-faktore berrien arabera txatalen erradiositate berriak kalkulatu eta gero, azpitxatalen erradiositatea ondoko moduan kalkula daiteke:

$$B_s = E_i + \rho_i \sum_{j=1}^{n} B_j F_{s-j}$$

Algoritmo hori erabiliz, azpitxatalak sortzen (txatal-sareak zatitzen) jarraituko genuke, txatal-sare bateko erradiositate aldaketak finkatutako balio bat baino txikiagoak izan arte. Algoritmoa aplikatuz, forma-faktoreak lortzeko erabiltzen den matrizearen dimentsioa aldatu gabe kalkula daiteke azpitxatalen erradiositatea.

10.5 Urratsez urratseko hobekuntza

Txatal bakoitzaren erradiositatea lortzeko, 10.3 ekuazio-sistema ebatzi behar da. Baina ekuazio-sistema hori askatzeak kalkulu-kopuru handia egitera behartzen gaitu; hori dela eta, irudia lortu artean denbora luzea igaro daiteke, eta erabiltzaileekin elkarrekintzan diharduten aplikazioek ezin dute halakorik onartu. Arazo horri irtenbide egokia ematearren, urratsez urratseko hurbilketan oinarritutako algoritmoa erabil genezake. Azken finean, kalkuluak alderantziz egitean oinarritzen da metodo hori; hau da, txatal bakoitzaren erradiositatea kalkulatzeko, orain arte, txatal guztien eragina hartzen genuen kontuan; baina alderantziz ere pentsa daiteke, hots, txatal batek beste guztiengan duen eragina kalkulatzea plantea daiteke. Horrela, berez argi-iturri diren txataletatik hasita, beste guztietan duten eragina kalkula daiteke. Eragin hori dela eta, zenbait txatalen erradiositatea aldatu egingo da, eta aldaketa jasan duten txatalek beste txataletan duten eragina kalkulatu beharko da. Bigarren kalkulu hori egin aurretik, eszenaren irudia marraz daiteke, eta erabiltzaileak hasierako irudi ilun samarra ikusi ahal izango du. Urratsez urrats irudi hori hobetuz joango da, txatalek jasandako aldaketa oso txikia denean prozesua gelditzea erabaki arte.

10.3 ekuazio-sistemako i lerroa konputatzea i txatalaren erradiositatea kalkulatzearen baliokidea da; hori beste txatalen erradiositatearen balioztapenean oinarrituz kalkulatzen da. Ekuazio-sistemako lerro bakoitzarentzat lortuko dugun terminoaren batugai bakoitza $\rho_i B_j F_{i-j}$ modukoa izango da, eta j txatalak i txatalaren erradiositatean duen eragina adierazten du:

$$B_i \quad \text{gaineko } j \text{ txatalaren eragina} \quad = \rho_i B_j F_{i-j} \quad j \text{ guztientzat}$$

Hurbilketa horrekin, txatal bakoitzak eszenako beste txatalek jaurtitako argia *biltzen* duela esan daiteke. Txatal bakoitzaren erradiositatea kalkulatzeko, txatal guztien erradiositateek ezagunak izan behar dute. Urratsez urratseko hobekuntza-metodoan, ordea, txatal bakoitzaren erradiositatea kalkulatu ondoren ingurunera *jaurtitzen* da eta bere eragina beste guztiei hedatzen zaie:

$$B_j \quad \text{gaineko } i \text{ txatalaren eragina} \quad = \rho_j B_i F_{j-i} \quad j \text{ guztientzat} \tag{10.5}$$

Azken ekuazioaren bidez, B_i-ren balioa ezaguna bada, txatal horrek beste txatal guztietan duen eragina kalkula daiteke. Horretarako, kuboerdia erabiliz kalkulatuko diren F_{j-i} balioek ezagunak izan beharko dute, j guztientzat.

i. txatala aukeratu
j txatal bakoitzeko F_{i-j} kalkulatu
for j txatal bakoitzeko
{
ΔErradiositate=$\rho_j\ \Delta B_i\ F_{i-j}\ A_i/A_j$;
ΔB_j=ΔB_j+ΔErradiositate;
B_j=B_j+ΔErradiositate;
}
ΔB_i=0;

Irudia 10.6: Txatal baten erradiositatea ingurura jaurtitzeko algoritmoa.

Balio horien kalkuluak denbora eta memoriaren erabileraren gain karga handia ekarriko luke. Baina 10.2 ekuazioan finkatutako erlazioa erabiliz:

$$B_j \text{ gaineko } i \text{ txatalaren eragina } = \rho_j B_i F_{i-j} \frac{A_i}{A_j} \quad j \text{ guztientzat} \qquad (10.6)$$

10.5 ekuazioaren bidez, F_{j-i} kalkulatzeko j kuboerdi erabili beharko lirateke. Baina 10.6 ekuazioa aplikatuz, F_{i-j} kalkulatzeko i txatalean zentraturik egongo litzatekeen kuboerdi bakar bat erabili beharko genuke. Kuboerdi bakar bat erabiltzearen abantaila argi dago, bai memoria eta bai abiadura aldetik.

Beraz, 10.6 ekuazioa j bakoitzarentzat ebaluatzea, i txatalean zentratutako kuboerdi bakar bat erabiliz forma-faktoreak kalkulatzea izango litzateke. Forma-faktore horiek azkar kalkulatzeko gai izango bagina (adibidez Z-buffer algoritmoa hardware bidez inplementatuz eta kuboerdi teknikarako aplikatuz), urrats bakoitzean erradiositate berriak azkar kalkulatzeko gai izango ginateke, eta gainera, i jakin horrentzat F_{i-j} guztiez ahaztu ahal izango dugu, behin kalkulatuz gero memorian gorde ahal izango baititugu. Ondorioz, urrats bakoitzean kuboerdi bakar bat erabili beharko dugu, agian bat ere ez, zeren, aurreko urrats batean i txatala erabili behar izan badugu, memorian edukiko baititugu forma-faktoreak.

Txatal bateko erradiositatea ingurura jaurti ondoren, beste txatal bat aukeratu behar da. Eszenan eragin handiena izango duena, hots, emateko erradiositate gehien duena aukeratzea da egokiena. Horrela, edozein txatal ausaz aukeratu beharrean, $\Delta B_i A_i$ balio handiena duena izango da aukeratutakoa. Hasieran txatal bakoitzaren erradiositatea B_i=ΔB_i=E_i izango da;

txatal arrunten E_i zero da, baina argi-iturri direnenak balioren bat edukiko dute. Finkatutako tolerantzia lortu arte eragin beharko da algoritmoa.

Iterazio bakoitzaren bukaeran irudi bat sortuko bagenu, lehenengo irudia nahiko iluna izango litzateke eta iterazio bakoitzean argiago ikusiko genuke. Hasierako irudiak erabilgarriago bihurtzeko, ingurune-terminoa gehi diezaiokegu erradiositateari; eta iterazio bakoitzean ingurune-aldagaia txikiagotuz joango litzateke, bere eragina desagertu arte. Gai horrek jaurti gabeko txatalen erradiositatearen menpeko balioa izango luke. Baina hori kalkulatu aurretik, ingurunearen propietatea izango den islada barreiatuaren batezbesteko faktore bat kalkulatu beharko litzateke. Bere kalkulua txatalen islada-faktorean oinarritzen da, txataletako islada-faktore bakoitzari txataLaren azaleraren araberako garrantzia egokituz.

$$\rho_{\text{batezb}} = \frac{\sum_{i=1}^{n} \rho_i A_i}{\sum_{i=1}^{n} A_i}$$

Ekuazio hori R islada-faktore globala konputatzeko erabiltzen da:

$$R = 1 + \rho_{batezb} + \rho_{batezb}^2 + \rho_{batezb}^3 + \ldots = \frac{1}{1 - \rho_{batezb}}$$

Beraz, jaurti gabeko txatalen erradiositatea kontuan izango duen ingurune-terminoaren balioa hauxe izango da:

$$Ingurunea = \frac{R \sum_{i=1}^{n} (\Delta B_i A_i)}{\sum_{i=1}^{n} A_i}$$

Termino horren helburu bakarra txatal bakoitzaren erradiositatea handitzean datza, hasierako irudiak ilunegiak izan ez daitezen:

$$B_i^{'} = B_i + \rho_i Ingurunea$$

10.6 Forma-faktore zehatzagoen kalkulua

Nahiz eta kuboerdiaren teknika, hardware mailan inplementatutako Z-buffer algoritmoa aplikatuz eraginkorra izan, zenbait kasutan ez ditu emaitza egokiak lortzen:

for i txatal bakoitzeko
{ ΔB_i=E_i;
for i txataleren s azpitxatal bakoitzeko
B_s=E_i;
}
Azaleren_batura=$\sum_{i=1}^{n} A_i$;
Ingurune=R $\sum_{i=1}^{n}$ (ΔB_i A_i)/Azaleren_batura;
while ez dute bat egiten
{ $\Delta B_i A_i$ handien daukan i txatala aukeratu;
txatal guztietako s azpitxatalen F_{i-s} finkatu;
/* ΔEnergia jaurti beharreko energiarekin hasieratzen da */
ΔEnergia=$\Delta B_i A_i$;
/* i txataleren erradiositatea jaurti */
for i txataletik ikuskorra den j txatal bakoitzeko
{ ΔB_zaharra=ΔB_j;
for i-tik ikusten den j-ren s azpitxatal bakoitzeko
{ ΔErradiositate=ρ_j ΔB_i F_{i-s} A_i/A_s;
B_s=B_s+ΔErradiositate;
ΔB_j=ΔB_j+ΔErradiositate A_s/A_j;
}
/* ΔEnergia aldagaiari j txatalak zurgatutakoa kendu */
ΔEnergia = ΔEnergia - ($\Delta B_j - \Delta B$_zaharra)A_j;
}
Erpinen erradiositatea azpitxatalena erabiliz: $B_s + \rho_j$ Ingurune;
j txatal bateko s azpitxataleren erradiositatea ezagutuz
if Bi erpin auzokideren arteko erradiositate diferentzia altuegia
Azpitxatalak zatitu eta erradiositatea jaurti i txataletik
azpitxatal berrietara.
ΔB_i=0;
Gainazal ikuskorrak identifikatu eta argiztapena burutu
/* Ingurunearen balio berria kalkulatzeko ΔEnergia
(txatal guztiek zurgatutako energia) erabili */
Ingurune=Ingurune−$R\Delta$Energia/Azaleren_batura;
}

Irudia 10.7: Pixkanakako hurbilketa aplikatzen eta txatalak azpitxataletan zatitzen dituen algoritmoa. Ingurune-argia erabiliz.

1. Kuboerdiaren aurpegietako karratutxo bakoitzean gertuen dagoen txatal bakar baten identifikadorea gordetzen da. Karratutxo bakoitzean txatal bat baino gehiago proiektatzen bada, horietako zenbait ez lirateke kontuan hartuko. Bestalde, gertuen dagoen txatala oso txikia bada eta karratutxoaren azalera baino proiekzio txikiagoa badauka, bere eragina handiagotu egingo da, karratutxo osoan proiektatzen dela suposatzen baita.

2. Kuboerdiaren teknika erabiltzean txataleko erdiko puntua txatalaren gainazal guztiaren ordezkari bihurtzen da, puntu horretatik ikus daitekeena txatal guztitik ikus daitekeela suposatuz. Suposaketa hori faltsua gertatzen bada (txataleko zenbait tokitatik txatalaren erdiko puntutik ikuskorrak ez diren txatalak ikuskorrak direnean adibidez), gainazala azpitxataletan zatitu beharko da. Baina i txatal bat ezin da maila ezberdinetan zatitu j txatal bakoitzarentzat.

3. Kuboerdiaren hurbilketa zuzena izan dadin, txatalek batak bestetik nahiko urrun egon behar dute. Bi txatal auzokideak badira, kalkulu guztiak txatalaren erdiko puntuan oinarrituz egiten direnez, forma-faktorearen kalkulua behar baino txikiagoa izango da, kalkuluek txatalen zati auzokideen arteko hurbiltasuna ez baitute kontuan hartzen.

Izpi-hedaketa erabil daiteke forma-faktoreak kalkulatzeko; horretarako, txatal baten erradiositatea jaurtitzerakoan, jaurtitzailea azpigainazal txikitan zatitzen da eta argia jaso behar duen txataleko erpin bakoitzetik zatitutako azpigainazaletara doazen izpiak hedatzen dira. Erpinen eta azpitxatal bakoitzaren artean gainazalik ez badago, hau da, azpitxatala erpinetik ikuskorra bada, forma-faktorea kalkulatzen da. Erpinen eta azpitxatal guztien arteko forma-faktoreak kalkulatu ondoren, erpin bakoitzari dagokion forma-faktorea kalkulatu ahal izango dugu. Horretarako, azpitxatal bakoitzari bere azaleraren araberako garrantzia emanez, guztien arteko batezbestekoa kalkulatu beharko da.

Hurbilketa horrek zenbait abantaila ditu:

1. Txatal baten erradiositatea kalkulatu beharrean, txatalaren erpinen erradiositatea kalkulatuko dugu;eta argiztapenaz arduratzen diren algoritmoek horixe bera behar izaten dute, ez txatalaren erradiositatea. Hau da, txatal baten barnealdeko pixelen intentsitatea, erpinen intentsitateak interpolatuz lortzen dute algoritmo horiek.

2. Erpinen normalak erabiltzeko aukera dugu, poligonozko sareek gainazal biribilen itxura har dezaten.

3. Errealitatean ez dauden puntuzko argi-iturriak erabil daitezke. Erpinetik argi-iturrira izpi bat hedatu eta argi-iturriaren iluminazio-ekuazioa aplikatzearekin nahikoa izango litzateke.

4. Erpin bakoitzaren erradiositatearen kalkulurako, emailea den txatala gainazal-kopuru desberdinetan zati daiteke.

Kapitulua 11

Kolore-espazioak

11.1 Sarrera

Gaur egun, kolorearen ikuspuntutik argi-objektu elkarreraginaren simulazio zehatza egitea beharrezkoa duten aplikazioak garatzen dira. Bistaratze arloan ere, beharrezkoa da kolorearen tratamendu zehatza, kolorea informazio numerikoa adierazteko erabiltzen baita.

Ordenadore bidezko irudigintzaren industriak garapen-ahalegin guztiak fotorrealismorantz zuzendu dituen arren, kolorearen tratamendu zehatzaz ez da gehiegi arduratu, eta alde batera utzi duela esan dezakegu, nahiz eta 16 miloi kolore irudikatzeko gai diren ordenadoreak aspalditik erabili. Kolorearen simulazio zehatza baztertzearen arrazoiak ondokoak izan daitezke:

- Phong argiztapenean, izpi-hedaketan eta erradiositatean nagusi den RGB ereduaren erabilera eta prozesu horietan hiru uhin-luzera baino gehiago erabiltzeak dakarren gehiegizko kostua.
- Kolorearen simulazio zehatza baino garrantzitsuagoak diren akatsak.
- Kolorearen tratamendu zehatza eskatzen duten aplikazioen kopuru txikia.

Hala ere, aplikazio askok kolorearen tratamendu zehatzari esker hobekuntza handiak lortuko lituzkete. Tratamendu zehatz horren bidez irabazi handia lortuko lukeen errealismo-eredu bat ere badago, erradiositate-eredua alegia. Ordenadorez sortutako diseinu arkitektoniko baten irudia bistaratzean, begiratzen ari garen irudia argazkia den ala ordenadore bidez lortutakoa den

bereiz dezakegu. Bereizketa hori, batez ere, ordenadoreko irudiaren zehaztasun geometriko mugatuagatik egin dezakegu. Zehaztasun geometrikoaren eza, konputazio-kostuen eta gehiegizko kostu horien eraginez erabiltzen diren hurbilketen ondorioa da. Baina badaude garrantzitsuak diren "bigarren mailako"arrazoiak ere: itzal artifizialak, erabilitako argiztapena ...

Koloreen garrantzia handia den beste arlo bat ViSC (bistaratze zientifikoa) da; bertan bolumen-errealismorako eredua erabiltzen da, eta hiru dimentsiotan adierazten diren datuen aldaketak nabaritzeko erabiltzen dira koloreak. Kolorea, informazioa modu hobezinean komunikatzeko erabiltzen da.

Koloreen simulazio zehatza aplikatzea erabakitzen badugu, zenbait arazorekin egingo dugu topo. Kolorearen adierazpenerako eta kalkulurako hiru uhin-luzera erabili beharrean n uhin-luzera erabiltzea eta horrek dakarren kostuaren arazoa alde batera utziz,

- Nola adieraziko dugu kolorea?
- Nola konpondu kolore berdinen adierazpena monitore desberdinetan irudikatzerakoan lor daitezkeen kolore desberdinen arazoa?

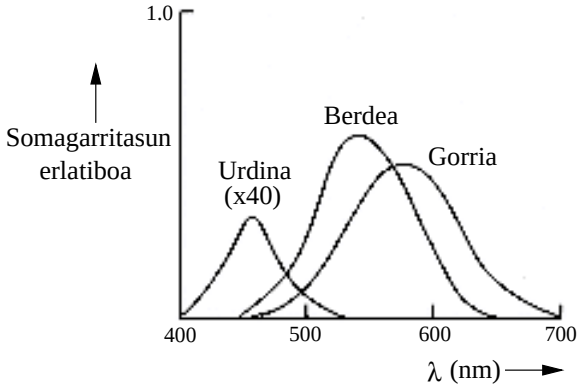
11.2 Hiru osagairen bidezko adierazpena

Koloreen zenbakizko espezifikazioa, oinarrizkoak diren hiru kolore batuz egiten da. Gizakientzat somagarriak diren kolore gehienak, baina ez guztiak, oinarrizko hiru kolore (gorria, berdea eta urdina) nahastuz adieraz daitezke. C kolore bat adierazteko:

$$C = rR + gG + bB$$

Ordenadore-monitore batean, kolore bat irudikatzeko bata bestearen aldamenean dauden hiru fosforo-puntu (gorria, berdea eta urdina) kitzikatzearekin nahikoa da. Fosforo puntu horiek oso txikiak direnez, begiak kolore bakarreko puntu modura somatzen ditu.

Giza erretinak, batez ere, hiru uhin-luzera somatzen ditu. Erretinan hiru errezeptore-mota dauzkagu eta errezeptore-mota bakoitzak uhin-luzera ezberdinen aurrean era ezberdinean jokatzen du. Errezeptore bakoitzak uhin-luzera ezberdinak somatzeko daukan ahalmena, 11.1 irudian ikus daiteke. Grafiko horien arabera, bakoitza uhin-luzera jakin batzuekin espezializatu



Irudia 11.1: Giza erretinako errezeptore-mota bakoitzak uhin-luzera ezberdinak somatzeko daukan ahalmena.

dela esan daiteke; hala ere, horrek ez du esan nahi, beste uhin-luzerak somatzeko gai ez direnik.

Eszena bateko argi-objektu elkarreragina zehazki simulatu nahi bada, elkarreragina hiru uhin-luzera baino gehiagotan aztertu behar da. Bestela, aliasing efektua emango da kolorearen eremuan, objektuaren isladapen-funtzioa eta argiaren distribuzioaren azpilaginketa dela-eta. Kolore-eremuan aliasing efektua gertatzeak, kolore egokiaren aldaketa dakar, eta, ondorioz, efektu hori ez da nabarmena. Aliasing efektuaren eraginez, ez ditugu kolore egokiak irudikatuko.

Errealismoa lortu nahi duen eredu batek kolorearen simulazio zehatza egiteko n uhin-luzera erabiliz egin beharko lituzke argiztapenaren kalkuluak. Bai argi-iturri erreal baten intentsitate-distribuzioa eta bai objektu erreal baten isladapen-ezaugarriak ere uhin-luzeren menpe daudenez, $F(\lambda)$ funtzioa erabili beharko genuke, eta funtzio hori n uhin-luzera diskreturen gain definitu beharko litzateke. Koloreari dagokion RGB hirukotea lortu eta gero, eta transformazio egokiak eragin ondoren, kalkulatutako RGB datuak RGB monitorera bidaltzeko gai izango ginateke. Prozesu hori 11.2 irudian azaltzen da. Beraz, puntu baten kolorea kalkulatzeko:

$$\begin{aligned} r &= \textstyle\sum_\lambda F(\lambda) r(\lambda) \\ g &= \textstyle\sum_\lambda F(\lambda) g(\lambda) \\ b &= \textstyle\sum_\lambda F(\lambda) b(\lambda) \end{aligned} \tag{11.1}$$

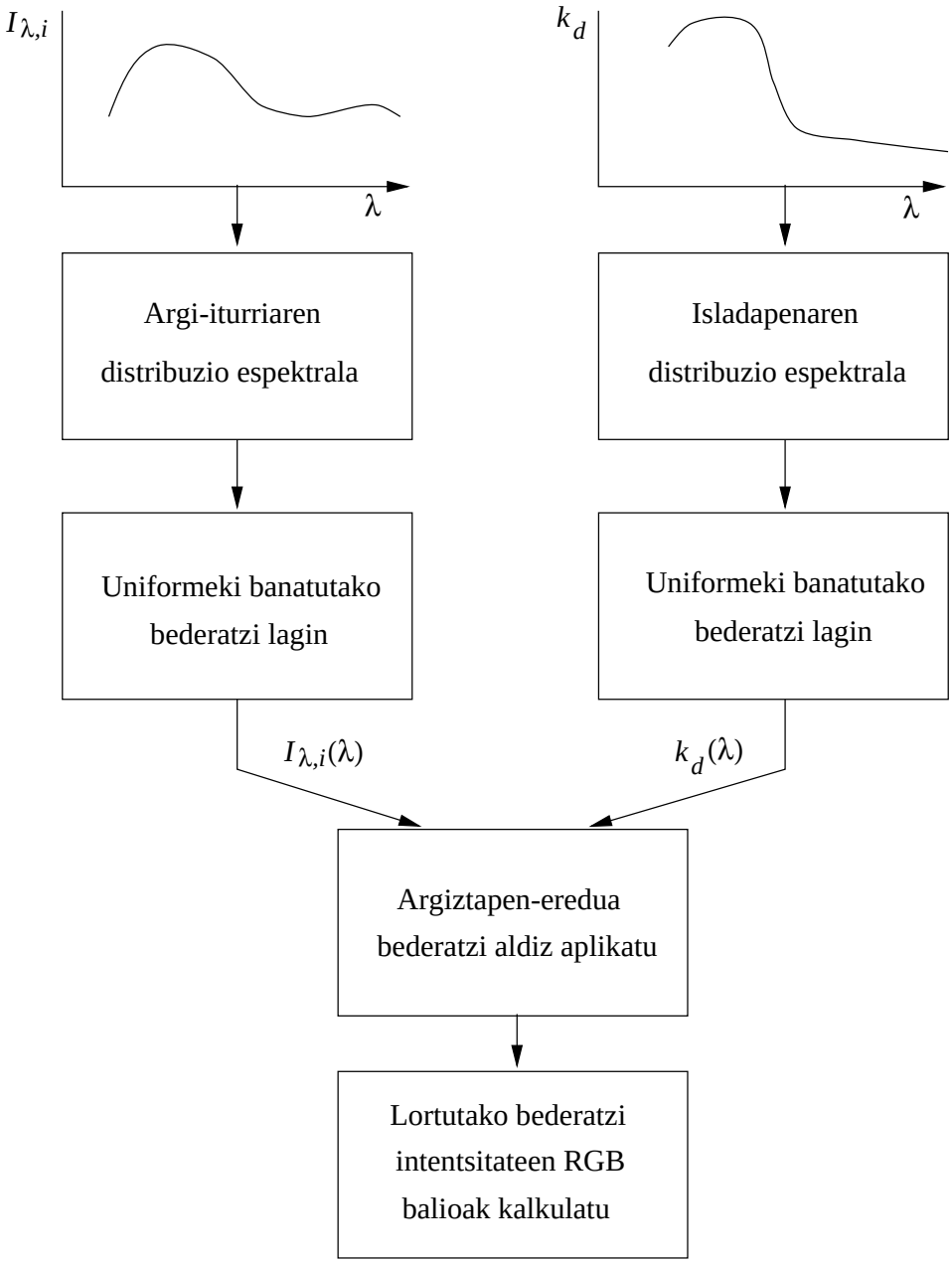
$r(\lambda)$, $g(\lambda)$ eta $b(\lambda)$ kolore-parekatzearen funtzioak dira.

Hall eta Greenberg-ek (1983) kolore-aliasing efektuari buruzko lan bat egin zuten. 360 nm eta 830 nm artean eta 1 nm diferentziako irudi errealistak erabili zituzten, ondoren RGB hirukote eta beste metodoen arteko konparaketa egiteko. Hallek, RGB hirukotean oinarritutako metodoak sortzen zuten distortsioa esanguratsua zela-eta bederatzi lagin erabiltzen zituen hurbilketaren bidez lortutako emaitzak, onak zirela adierazi zuen.

11.3 Kolore-espazioak

Ordenadore bidezko irudigintzan, ondoko kolore-espazioak erabiltzen dira:

- **Espazio espektrala:** Objektuaren isladapena eta argi-iturriak, intentsitate-distribuzio baten n uhin-luzerako lagin gisa definitzen dira.
- **RGB espazioa:** Ordenadore bidezko irudigintzan estandar bihurtu den kolore-adierazpena da. Espazio espektralean oinarritutako hiru laginetako espazioa da. Objektuaren isladapena eta argi-iturriak hiru uhin-luzeraren bidez deskribatzen dira, gorria, berdea eta urdina. Bestalde, n uhin-luzeraren bitartez egindako kalkuluak RGB eredura itzul daitezke 11.1 ekuazioan bezala. R, G eta B oinarrizko koloreak, kolore asetu edo puru modura hartzen dira.
- **$\mathbf{RGB_{monitore}}$ espazioa:** Espazio horretako hirukote batek kolore partikular bat irudikatzen du monitore partikular batean. Hirukote berdinak ez dauka monitore desberdinetan kolore berdina irudikatu beharrik. Edozein eredutako n laginetan oinarritutako kolore bat kalkulatu ondoren, monitorearen menpe dagoen aldaketa aplikatzen zaio, kolorea RGB_{monitore} espazioko puntu batera itzultzeko.
- **HSV espazioa:** RGB espazioaren transformazio ez-lineal bat da. Kolorea ñabardura, asetasun eta balioa hirukotearen bitartez definitzeko aukera ematen digu.
- **HLS espazioa:** HSV espazioaren antzekoa da. Erabiltzailearekin elkarreragiteko bide desberdina erabiltzeko posibilitatea eskaintzen digu.
- **CIE XYZ espazioa:** Koloreen espezifikaziorako estandar internazional nagusia da. Kolore bat eragingarri-hirukoteen multzo modura definitzen da.



Irudia 11.2: Intentsitatearen bederatzi lagin erabiltzen dituen argiztapen--eredua.

- **CIE xyY espazioa:** CIE XYZ ereduaren ondorioa da, koloreak definitzeko (x, y) koordenatu kromatikoak eta Y argitasuna erabiltzen dira.

- **CIE LUV espazioa:** CIE XYZ kolore-espazioaren bariante bat da, ordenadore bidezko irudigintzarako aproposa dena.

11.3.1 RGB eredua

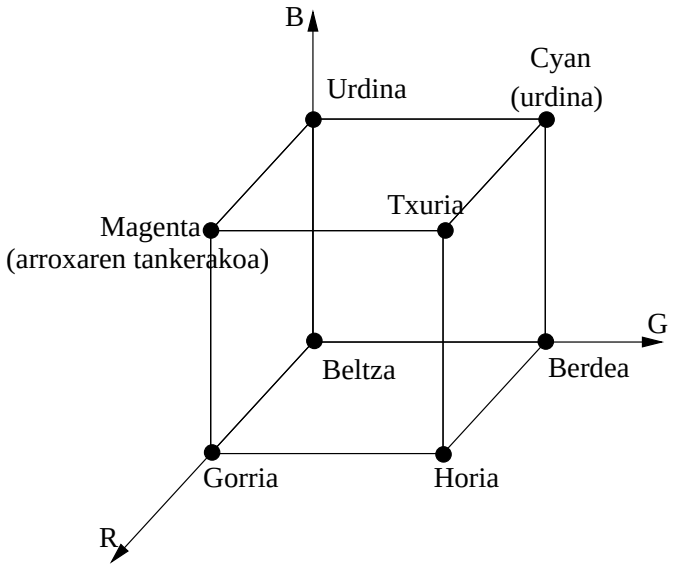
ordenadore bidezko irudigintzan erabili ohi den kolore-espazioa da. Kolore bat adierazteko, oinarrizko hiru koloreen kantitate desberdinak batzen dira, oinarrizko koloreak "nahastuz". $(0, 0, 0)$ hirukoteak kolore beltza adierazten du, ez kolore gorririk, ez kolore berderik, ezta kolore urdinik ere ez duen kolorea alegia. $(1, 1, 1)$ delakoa, berriz, kolore zuria da. Kolore-adierazpen horrek mugatzen duen kolore-espazioa 11.3 irudian ikus daiteke. RGB espazioaren ezaugarri garrantzitsuak:

1. Somagarritasun ez-lineala azaltzen du. Hau da, kolore-espazioko bi eskualde ezberdinetan gehikuntza bera aplikatuz, eskualde batek jasandako kolore-aldaketa beste eskualdeak jasandakoa baino askoz ere somagarriagoa izan daiteke.

2. Somagarritasun ez-lineala dela-eta, RGB adierazpeneko balio edo intentsitate baxuetan aplikatutako aldaketak ez dira irudian azaldutako koloreetan somatzen. RGB balio baxuetan 20 unitateko aldaketa bat eragitea beharrezkoa gerta daiteke, monitoreko irudian kolore-aldaketa somagarria izan dadin. Balio edo intentsitate altuetan, berriz, unitate bakarreko aldaketak oso nabariak diren kolore-aldaketak sortzen dituzte.

3. Ordenadore-monitore batean ikus daitezkeen kolore guztiak, RGB espazioko koloreak, giza begiarentzat somagarriak diren koloreen azpimultzo bat osatzen dute. Hori ez da RGB espazioan bakarrik gertatzen. Oinarrizko edozein hiru koloreren batuketek, gizakientzat somagarriak diren koloreen azpimultzo bat besterik ezin dute adierazi.

Ez da kolore-adierazpen ona. Esperientzia gutxiko erabiltzaileentzat zaila gertatzen da kolore zehatz baten RGB adierazpena jakitea. Ez da adierazpen intuikorra.



Irudia 11.3: RGB adierazpideak mugatzen duen kolore-espazioa

11.3.2 HSV eredua

Ñabardura, asetasun eta balio (Hue, Saturation, Value) hirukotearen bidezko adierazpena A. R. Smith-ek (1978) proposatu zuen. Adierazpenaren helburua erabiltzailearentzat intuikorra izatean datza. Adierazpen horrek mugatzen duen eremuak edo kolore-espazioak kono hexagonalaren itxura du. HSV konoa, RGB kuboaren transformazio ez-lineala da. Adibidez, kolore urdina izanik, H, S edo V osagaiaren aldaketak kolorean izango duen eragina ezaguna izango da aldaketa eragin baino lehen, adierazpena intuikorra baita. Kolore gaineko aldaketak erraz egin daitezke, RGB adierazpenean ez bezala.

HSV eredua koordenatu polarretan oinarritzen da. H gradutan adierazten da ($0 \ldots 360$). H parametroari balio bat esleitzea kolore bat aukeratzearen baliokidea da; S txikitzea koloreari kolore zuria gehitzearen baliokidea da; eta V txikitzea, berriz, beltza gehitzearen baliokidea. 11.4 programaren bidez RGB hirukotea HSV adierazpenera itzul dezakegu.

RGB kuboaren diagonal nagusiarekiko elkarzuta den planoan RGB kuboa proiektatuz, disko hexagonal bat lortuko dugu. Ideia horretan oinarritzen da azaldutako programa.

Ondoren RGB kuboaren sei erpinen eta HSV ereduko kono hexagonalaren

```
double maximoa(double Red, double Green, double Blue)
    { double max;
    if (Red>Green) max=Red; else max=Green;
    if (Blue>max) max=Blue;
    return(max);
    }

double minimoa(double Red, double Green, double Blue)
    { double min;
    if (Red<Green) min=Red; else min=Green;
    if (Blue<min) min=Blue;
    return(min);
    }

RGBtik_HSVra(double R, double G, double B, double *H,double *S,
double *V)
    { int ez_definitua=adieraz_daitekeen_int_handiena_maxint;
    double balio_maximoa,balio_minimoa,dif,r_dist,g_dist,b_dist;

    balio_maximoa=maximoa(R,G,B);
    balio_minimoa=minimoa(R,G,B);
    dif=balio_maximoa-balio_minimoa;
    *V=balio_maximoa;
    if (balio_maximoa!=0) *S=dif/balio_maximoa; else *S=0;
    if (*S==0) *H=ez_definitua;
    else {
        r_dist=(balio_maximoa-R)/dif;
        g_dist=(balio_maximoa-G)/dif;
        b_dist=(balio_maximoa-B)/dif;
        if (R==balio_maximoa) *H=b_dist-g_dist;
        else if (G==balio_maximoa) *H=2+r_dist-b_dist;
            else if (B==balio_maximoa) *H=4+g_dist-r_dist;
        *H=*H*60;
        if (*H<0) *H=*H+360;
        }
    }
```

Irudia 11.4: RGB hirukotea HSV adierazpenera itzultzeko algoritmoa.

```
HSVtik_RGBra(double H,S,V, double *R,*G,*B)
    { int i;
    double f,p,q,t;
    if (S==0)
        { *R=V; *G=V; *B=V; }
    else
        { if (H==360) H=0;
        H=H/60;
        i=int(H);
        f=H-i;
        p=V*(1-S);
        q=V*(1-(S*f));
        t=V*(1-(S*(1-f)));
        switch (i)
            { case 0:
              *R=V; *G=t; *B=p;
              break;
            case 1:
              *R=q; *G=V; *B=p;
              break;
            case 2:
              *R=p; *G=V; *B=t;
              break;
            case 3:
              *R=p; *G=q; *B=V;
              break;
            case 4:
              *R=t; *G=p; *B=V;
              break;
            case 5:
              *R=V; *G=p; *B=q;
              break;
            }
        }
    }
```

Irudia 11.5: HSV hirukotea RGB adierazpenera itzultzeko algoritmoa.

sei puntuen arteko baliokidetzak azaltzen dira:

RGB		HSV
$(1,0,0)$	gorria	$(0,1,1)$
$(1,1,0)$	horia	$(60,1,1)$
$(0,1,0)$	berdea	$(120,1,1)$
$(0,1,1)$	cyan	$(180,1,1)$
$(0,0,1)$	urdina	$(240,1,1)$
$(1,0,1)$	magenta	$(300,1,1)$

Hemen H gradutan neurtzen da. Disko hexagonal hori, HSV ereduan V=1 planoak barneratzen duen hexagonoa izango litzateke. Kuboaren diagonal nagusian zehar mugituz, pauso bakoitzean azpikubo bat definituko genuke, eta azpikubo bakoitzak disko hexagonal bat definituko luke. Modu horretan lortutako disko hexagonalen pilak, HSV ereduko kono hexagonala osatzen du.

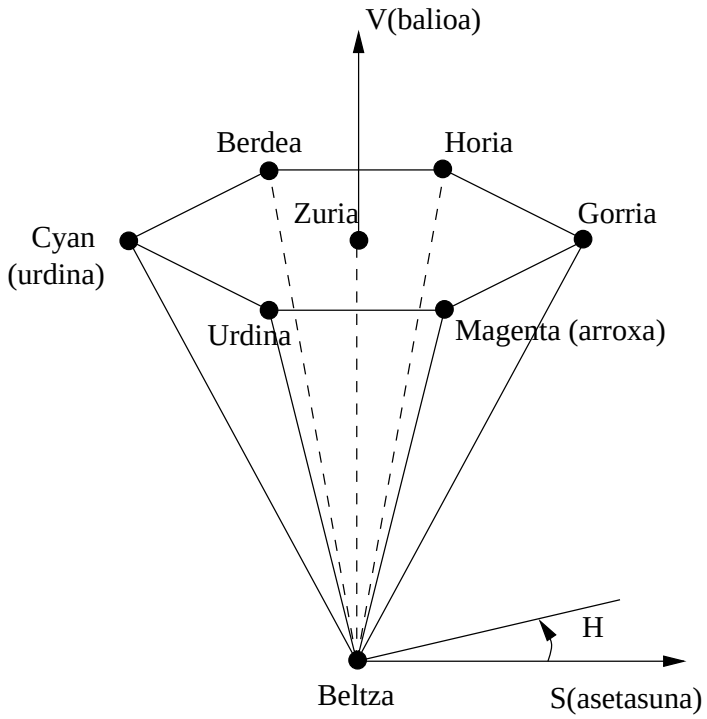
11.6 irudian HSV adierazpenak mugatzen duen kolore-espazioa ikus daiteke.

Baina HSV adierazpenak ere zenbait desabantaila ditu:

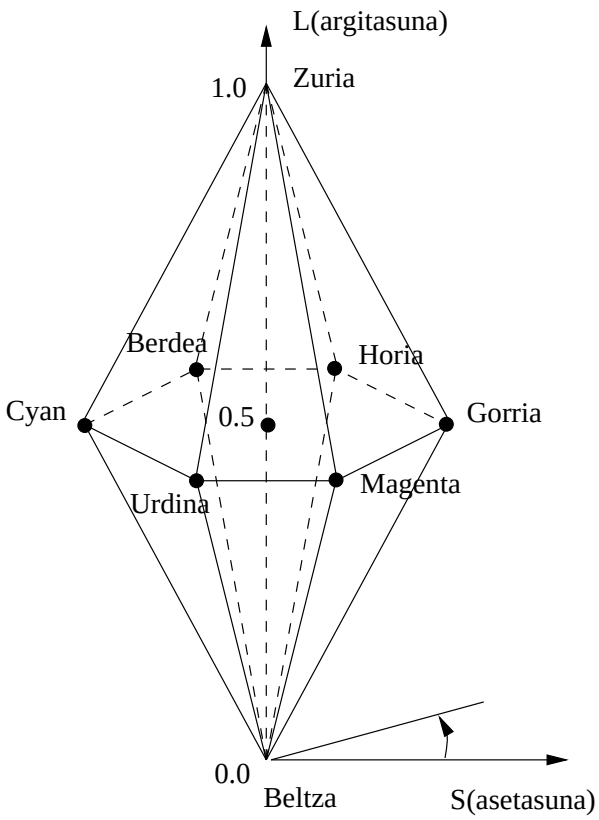
- Somagarritasun ez-linealaren arazoa ematen da.
- Ereduko atributuak ez dira somagarriak. Posible da H parametroaren gaineko aldaketa bat somatzea, benetan aldatzen ari den parametroa V denean, adibidez.
- HSV kolore-espazioan, monitorean irudika daitezkeen kolore guztiak etiketatzen dira, eta V konstante duen plano bateko kolore guztiak intentsitate berdina dutela suposatzen da. Nahiz eta suposaketa hori egia ez izan, plano berdineko eta intentsitate maximoa duten urdina eta horia irudikatzen baditugu, horia urdina baino distiratsuago somatuko dugu.

11.3.3 HLS eredua

HLS (Hue, Lightness, Saturation) eredua HSV ereduarekin erlazionatuta dago. HSV ereduak mugatzen zuen espazioa kono hexagonal baten bidez adierazten genuen. HLS ereduak mugatzen duena berriz, kono hexagonal bikoitz batekin adierazten da.



Irudia 11.6: HSV adierazpideak deskribatzen duen kolore-espazioa, kono hexagonala.



Irudia 11.7: HLS kolore-ereduak adierazten duen kono hexagonal bikoitza.

Erdiko ardatzak L izena du (Lightness, argitasuna) eta ñabardura, orain ere, gradutan neurtzen da. Eredu horretan, bai kolore beltza bai zuria izkinetan dauden bi puntu dira. Hori dela eta, HSV eredua baino intuikorragoa da. Baina ñabardura asetuak agertzen diren planoa L=0.5 posizioan dago, eta horrek ereduarentzat desabantaila dakar, kolore aukeraketarako erabiltzean. 11.8 eta 11.9 programek HLS eta RGB adierazpenen arteko itzulpenak egiten dituzte.

11.3.4 CIE XYZ espazioa

CIE XYZ, kolore-parekatzearen funtzioetan oinarrituriko eredu bat da, kolore bat adierazteko, (X, Y, Z) hirukote bat erabiltzen da, eta giza begiarentzat somagarriak diren kolore guztiak adieraz daitezke. Beraz, aurreko hiru adierazpenak ez bezala, eredu hori ez da monitorean irudika daitezkeen koloretara mugatzen.

Estandar horren oinarriak 1931. urtean finkatu ziren. Eredu horren helburua, edozein kolore hirukote modura adieraztea da, non:

$$C = rR + gG + bB \tag{11.2}$$

11.2 ekuazioa lortzeko asmotan esperimentu asko egin zen eta horien ondorioz, 11.10 irudian ikus daitezkeen kolore-parekatzearen funtzioak lortu ziren. λ uhin-luzerako edozein kolore lortzeko, batu beharreko balioak itzultzen dituzten funtzioak dira kolore-parekatzearen funtzioak. Hau da:

$$C_\lambda = r(\lambda) + g(\lambda) + b(\lambda)$$

λ uhin-luzerako koloreari egokituko zaizkion r, g eta b balioak:

$$r = k \int_\lambda P(\lambda) r(\lambda) d\lambda$$

$$g = k \int_\lambda P(\lambda) g(\lambda) d\lambda$$

$$b = k \int_\lambda P(\lambda) b(\lambda) d\lambda$$

$P(\lambda)$ energia espektralaren distribuzioa da. Kolore-parekatzearen funtzioak erabiliz edozein C kolore (r, g, b) hirukote batera itzul dezakegu.

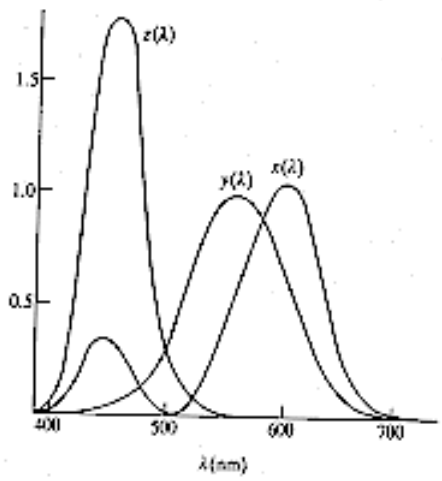
Baina edozein kolore oinarrizko diren hiru koloreen batuketa modura adierazteak zenbait arazo dakartza. Hirukote positiboak (r, g eta $b > 0$) erabiliz

```
RGBtik_HLSra(double R, double G, double B, double *H,double *L,
double *S)
        {
        int ez_definitua=adieraz_daitekeen_int_handiena_maxint;
        int balio_baxua=0.0000001;
        double balio_maximoa,balio_minimoa,dif,r_dist,g_dist,b_dist;

        balio_maximoa=maximoa(R,G,B);
        balio_minimoa=minimoa(R,G,B);
        dif=balio_maximoa-balio_minimoa;
        *L=(balio_maximoa-balio_minimoa)/2;
        if (abs(dif)<balio_baxua)
                {
                *S=0; *H=ez_definitua;
                }
        else
                {
                if (*L<=0.5) *S=dif/(balio_maximoa+balio_minimoa);
                else *S=dif/(2-balio_maximoa-balio_minimoa);
                r_dist=(balio_maximoa-R)/dif;
                g_dist=(balio_maximoa-G)/dif;
                b_dist=(balio_maximoa-B)/dif;
                if (R==balio_maximoa) *H=b_dist-g_dist;
                else
                        {
                        if (G==balio_maximoa) *H=2+r_dist-b_dist;
                        else
                        {
                        if (B==balio_maximoa) *H=4+g_dist-r_dist;
                        }
                *H=*H*60;
                if (*H<0) *H=*H+360;
                }
        }
```

Irudia 11.8: RGB hirukotea HLS adierazpenera itzultzeko algoritmoa.

```
double RGB(double q1,q2,H)
        {
        if (H>360) H=H-360;
        if (H<0) H=H+360;
        if (H<60) return (q1+(q2-q1)*H/60);
        else
                {
                if (H<180) return (q2);
                else
                        {
                        if (H<240) return (q1+(q2-q1)*(240-H)/60);
                        else return (q1);
                        }
                }
        }
HLStik_RGBra(double H,S,V, double *R,*G,*B)
double p1,p2;
        {
        if (L<=0.5) p2=L*(1+S);
        else p2=L+S-L*S;
        p1=2*L-p2;
        if (S==0)
                {
                *R=L; *G=L; *B=L;
                }
        else
                {
                *R=RGB(p1,p2,H+120);
                *G=RGB(p1,p2,H);
                *B=RGB(p1,p2,H-120);
                }
        }
```

Irudia 11.9: HLS hirukotea RGB adierazpenera itzultzeko algoritmoa.



Irudia 11.10: CIE kolore-parekatzearen funtzioak.

somagarriak diren koloreen azpimultzo bat soilik adieraz dezakegu, izan ere, bi kolore nahastean lortzen den kolorea, jatorrizko koloreak bezain asetua izatea ezinezkoa baita, hau da, koloreen nahasketaren bidez ezinezkoa baita oso asetua den kolore bat lortzea.

Hori dela eta, r, g eta b parametroen balio negatiboak ekiditeko asmoz, oso asetuak diren hiru kolore-parekatzearen funtzioak erabiltzea erabaki zen. Kolore-parekatzearen funtzioak, $x(\lambda)$, $y(\lambda)$ eta $z(\lambda)$, esperimentuen bidez lortutako funtzioen gain aplikatutako transformazioen bidez kalkulatu ziren. Funtzio berri horiek erabiliz, X, Y, Z hiru balio lortuko ditugu, horiek batuz edozein kolore lortuko dugularik. Kolore-parekatzearen funtzio horiek 11.11 irudian agertzen dira eta beti balio positiboak itzultzen dituzte. Beraz:

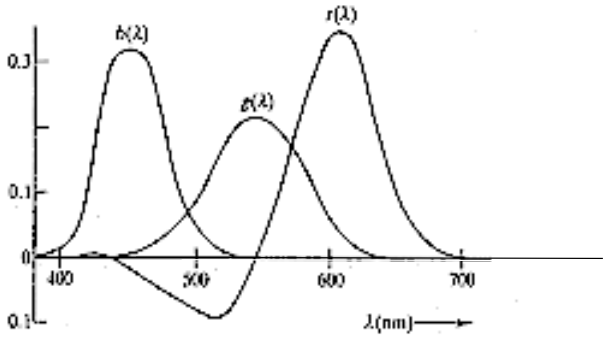
$$X = k \int_{\lambda} P(\lambda)x(\lambda)d\lambda$$

$$Y = k \int_{\lambda} P(\lambda)y(\lambda)d\lambda$$

$$Z = k \int_{\lambda} P(\lambda)z(\lambda)d\lambda$$

Argia ematen duten objektuentzat $k = 680$ da.

X, Y eta Z balioek mugatzen duten kolore-espazioa, CIE XYZ espazioa da. Kolore-espazio horrek muturra jatorri-puntuan daukan kono baten itxura



Irudia 11.11: kolore-parekatzearen funtzioak.

du, eta monitoreak irudika ditzakeen koloreek, CIE espazioaren azpiespazio bat osatzen dute, 11.12 irudian azaltzen den bezala.

11.3.5 CIE xyY espazioa

(X, Y, Z) hirukotea adierazteko beste modu bat (x, y, Y) da; hemen, (x, y) direlakoak koordenatu kromatikoak dira.

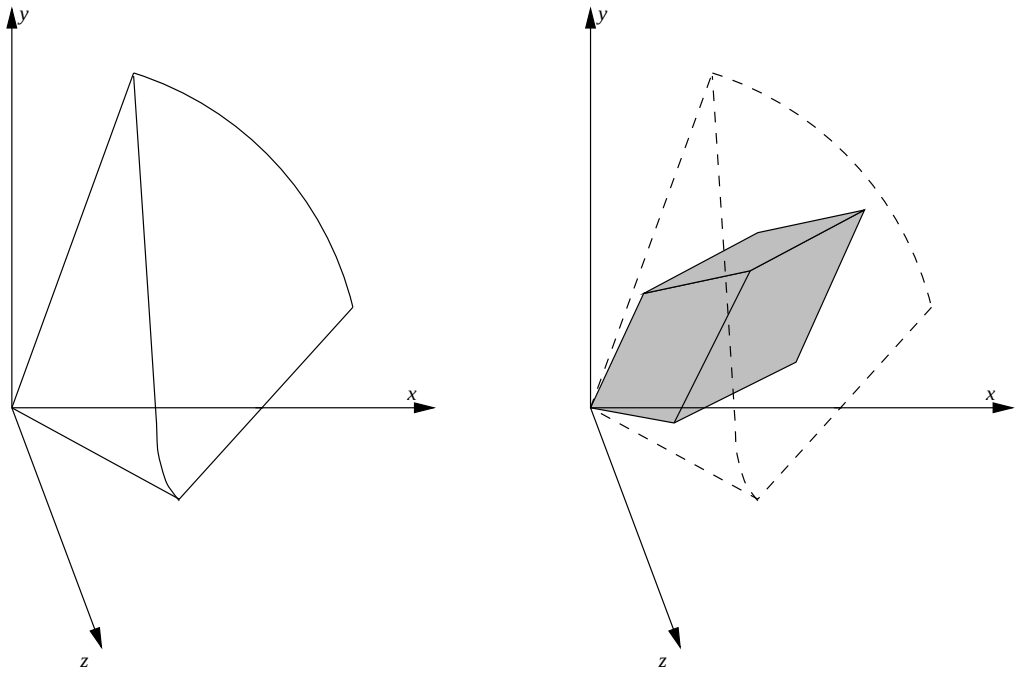
$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

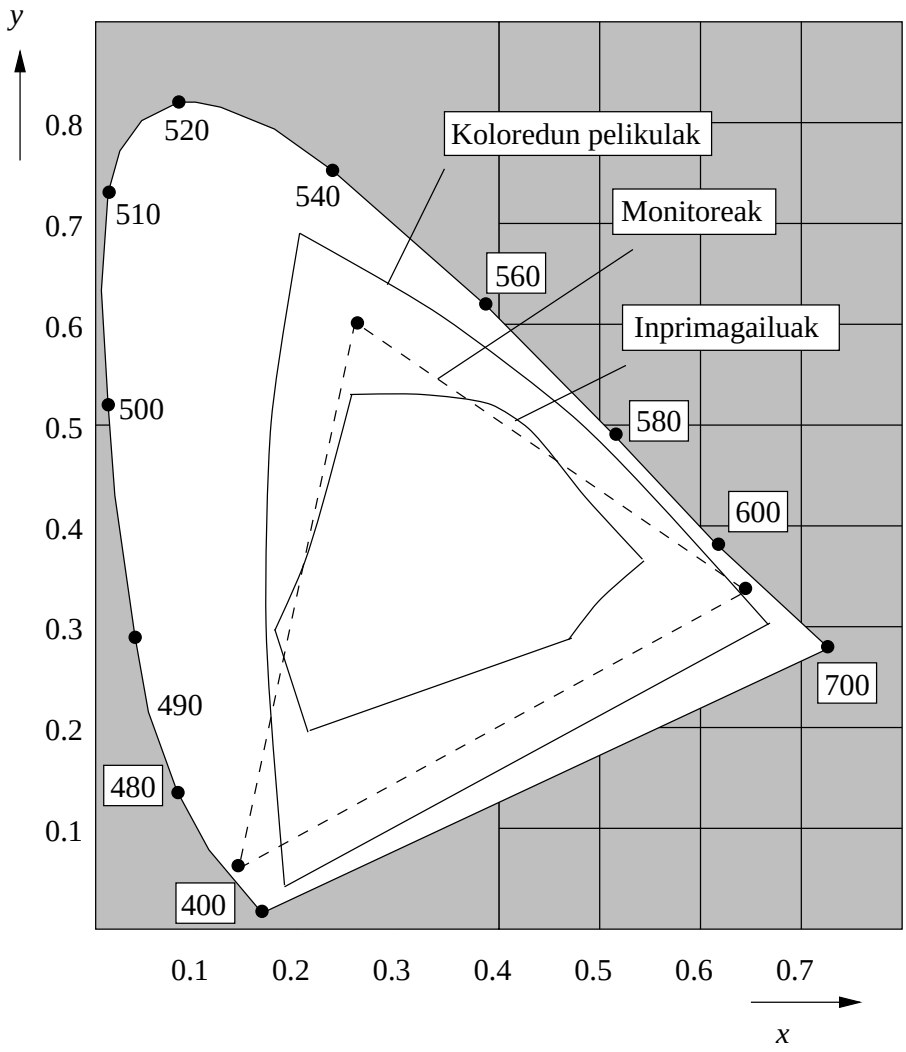
Koordenatu horiek erabiliz, diagrama kromatikoa deritzon bi dimentsiotako 11.13 irudia kalkula dezakegu (kolore ikuskor guztien (x, y) balioak formuletan ordezkatuz). Diagrama horrek, imajinak irudikatzeko erabil daitezkeen gailu desberdinek irudikatu edo marraz ditzaketen koloreak adierazten ditu. Diagrama hori ikuskatze-gailu desberdinen arteko konparaketarako erabilgarria da, eta adibidez, ordenadorean dugun irudi bat gailu desberdinak erabiliz marraztu edo agertu behar dugunean erabil daiteke.

11.3.6 CIE LUV espazioa

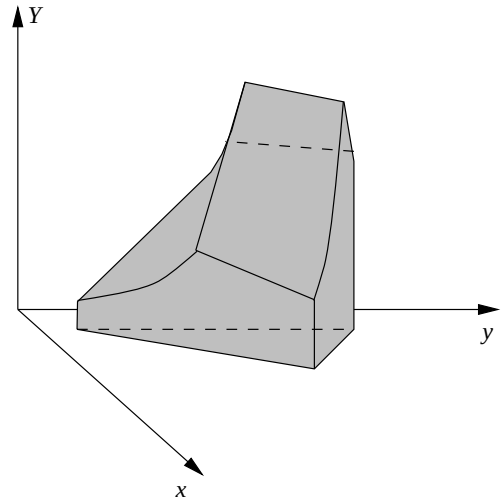
Monitorearen espazioak CIE xyY espazioan duen itxura dela-eta, ez da komenigarria ordenadore bidezko irudigintzan CIE xyY adierazpidea erabiltzea. CIE espazioaren abantailak galdu gabe (kolore-sistema estandarrekiko



Irudia 11.12: CIE XYZ ereduak mugatzen duen kolore-espazioa eta monitoreak irudika ditzakeen koloreen azpiespazioa.



Irudia 11.13: CIE diagraman koloredun pelikulek, monitoreek eta inprimagailuek adieraz ditzaketen kolore-azpiespazioak ikus daitezke.



Irudia 11.14: Monitoreak irudika ditzakeen CIE xyY espazioko koloreen azpiespazioa.

duen zuzeneko erlazioa, kolore-komunikaziorako lengoaiaren espezifikazioaren zehaztasuna etab.), beste bi abantaila lortzeko erabil dezakegu CIE LUV espazioa:

- CIE LUV espazioak somagarritasun lineala azaltzen du.
- CIE LUV adierazpideak mugatzen duen kolore-espazioa koordenatu sistema zilindrikoan deskriba daiteke. Eta ondorioz, erabiltzailearekin lanean jarduteko aproposa da.

CIE XYZ adierazpeneko abantailei beste bi abantaila horiek gehitu ondoren, CIE LUV kolore-espazioa, kolorearen tratamendu zehatza behar duten ordenadore bidezko irudi aplikazioentzat komenigarria dela esan daiteke.

CIE xy diagraman, berdin somatzen ditugun bi kolorek ez dute (x, y) espazioko distantzia uniformetara egon beharrik. Berdinak somatzen ditugun aldaketak, eskualde morean txikiagoak dira eskualde berdean baino. Desabantaila hori zuzentzeko, badago (x, y) espazioa distortsionatzen duen transformazio bat. Transformazio hori 1960. urtean onartu zen:

$$u = 2x/(6y - x + 1.5)$$

$$v = 3y/(6y - x + 1.5)$$

Eta 1974. urtean beste hau onartzea erabaki zen:

$$u' = 2x/(6y - x + 1.5)$$

$$v' = 4.5y/(6y - x + 1.5)$$

Y aldatu gabe mantentzen da. 1976. urtean, CIE LUV espazioaren definizioa finkatuko zuen azken transformazioa aplikatu zen. Azken aldaketa horrek bi zuzenketa egiten ditu: ardatz akromatikoa $(0, 0)$ puntuan kokatzen du eta $L^* = 0$ puntua bihurtzen da. Hona hemen $L^*u^*v^*$ hirukotearen definizioa:

$$L^* = 116(Y/Y_w)^{1/3} - 16$$

$$u^* = 13L^*(u' - u'_w)$$

$$v^* = 13L^*(v' - v'_w)$$

adierazpide horietan u', v' eta $Y(u', y', Y)$ espazioko kolore-koordenatuak eta u_w, v_w eta Y_w espazio horretako kolore zuriaren koordenatuak direlarik. Espazio horrek mugatzen duen kolore-espazioa 11.15 irudian ikus daiteke.

11.15 irudian ikusten den kolore-espazioa kono bikoitz distortsionatu bat da. Beraz, nahiz eta HLS eredua ez izan, eredu horri dagozkion H, L eta S parametroak defini daitezke:

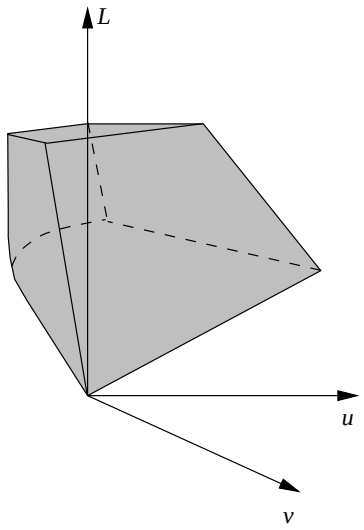
$$H(\tilde{n}abardura) = \arctan(v^*/u^*)$$

$$L(argitasuna) = L^*$$

$$S(asetasuna) = (u^{*^2} + v^{*^2})^{1/2}$$

H CIE 1976 uv ñabardura-angelu eta L CIE 1976 argitasun psikometriko gisa ezagutzen dira. S CIE 1976 uv kroma da. Asetasuna, argitasunaren arabera definitu beharko litzateke. CIE 1976 asetasun psikometrikoaren definizioa hauxe da:

$$S_{uv} = \frac{\text{CIE 1976 uv kroma}}{L^*} = \frac{(u^{*^2} + v^{*^2})^{1/2}}{L^*}$$



Irudia 11.15: CIE $L^*u^*v^*$ kolore-espazioa.

Filosofia horretan oinarritzen den eredu berriago bat ere badago, Tektronix HVC sistema deiturikoa (Taylor, Murch eta McManus, 1988). CIE $L^*u^*v^*$ espazioan oinarritzen da eta H(ñabardura), V(balioa) eta C(kroma) ardatzek 0-360, 0-100 eta 0-100 tarteko balioak izan ditzakete, hurrenez hurren. Eredu berri horrek, HLS ereduak erabiltzeko duen erraztasuna eta CIE espazioko somagarritasun-neurriak konbinatzen ditu. Gainera, HLS ereduak dituen zenbait desabantaila ekiditen ditu.

11.4 Monitore desberdinen azterketa

11.4.1 Monitore desberdinak eta kolore berdinak

Bi monitore desberdinetan RGB hirukote bati dagokion kolorea irudikatuz, kolore desberdinak ager daitezke, monitore bakoitzak bere ezaugarriak baititu. RGB hirukote bat CIE XYZ adierazpenera (askoz ere zehatzagoa dena eta monitorean benetan irudikatuko dena jakiteko erabiliko duguna) itzultzeko, ondoko ekuazioa daukagu:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} = T \begin{pmatrix} r \\ g \\ b \end{pmatrix}$$

Ekuazio horretan T parametroa desberdina izango da monitore bakoitzarentzat. T hori finkatzeko, RGB balioen eta monitorearen irteeraren artean erlazio lineal bat dagoela suposatzen da. (X_r, X_g, X_b) balioak, R oinarrizko kolorearen unitate bat irudikatzeko behar diren hiru eragingarrien balioak dira, (Y_r, Y_g, Y_b) balioak G unitate bat irudikatzeko etab. T_1 eta T_2 parametroak ezaugarri desberdinak dituzten bi monitoreei aplikatzen zaien transformazioak badira, orduan, lehenengo monitoreko RGB balioak bigarrenera itzultzeko, $T_2^{-1}T_1$ aplikatu beharko da. T parametroa ondoko moduan kalkula daiteke:

$$D_r = X_r + Y_r + Z_r$$

$$D_g = X_g + Y_g + Z_g$$

$$D_b = X_b + Y_b + Z_b$$

Ondorioz:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} D_r x_r & D_g x_g & D_b x_b \\ D_r y_r & D_g y_g & D_b y_b \\ D_r z_r & D_g z_g & D_b z_b \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix}$$

non:

$$x_r = X_r/D_r \quad y_r = Y_r/D_r \quad z_r = Z_r/D_r \quad \ldots$$

Aurreko T matrizea matrizeen arteko biderkaketan deskonposatuz:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{pmatrix} \begin{pmatrix} D_r & 0 & 0 \\ 0 & D_g & 0 \\ 0 & 0 & D_b \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix}$$

Lehenengo matrizeak monitorearen koordenatu kromatikoak adierazten ditu. RGB espazioko $(1, 1, 1)$ hirukoteak kolore zuria irudikatu beharko lukeela adierazteko honako hau idatz dezakegu:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{pmatrix} \begin{pmatrix} D_r \\ D_g \\ D_b \end{pmatrix}$$

11.4.2 Koloreen mugaketa

Monitore batzuk irudika ditzaketen zenbait kolore, ezaugarri desberdinetako beste monitoreetan irudikatzea ezinezkoa da. Zenbait monitoretan, osagairen bat 0 baino txikiagoa edo 1 baino altuagoa duten RGB hirukoteak irudikatzea ezinezkoa da. Agian, monitore batek $(1.2, 0.5, 0.5)$ hirukotea onartuko luke baina beste batek ($T_2^{-1}T_1$ itzulpena aplikatu ondoren) ez luke irudikatuko. Arazo berdina eman daiteke eszena baten irudi errealista lortzerakoan. Monitorea, eszena irudikatzeko beharko genituzkeen koloreak irudikatzeko gai ez izatea gerta daiteke. Inprimagailuen kasuan, kolore gutxi marrazteko gai izaten direnez, arazoa larriagoa da.

Arazo hori ekiditeko kolore-mugaketa aplikatuko dugu. Horren helburua, irudien koloreak irudika daitezkeen koloreetara mugatzea da, irudiaren kalitatea galdu gabe noski. Kolore-mugaketaren eraginkortasuna irudiaren ezaugarrien araberakoa izango da.

Osagairen bat 0 baino txikiagoa eta 1 baino handiagoa duten koloreak mugatuko ditugu. Horretarako, bi multzotan sailka daitezke irudika ezin daitezkeen koloreak:

1. Monitorearentzat onargarriak ez diren koloreak (RGB negatiboak).

2. Onargarriak izanik monitorearentzat gehiegizko intentsitatea duten koloreak (1 baino balio altuagoko zenbakiak dituzten RGB hirukoteak).

Edozein kolore-zuzenketak, kolore-aldaketa bat dakar. Kolorearen gaineko aldaketa, ñabardura-, asetasun- edota balio-atributuetan eragin daiteke.

Lehenengo multzoko koloreen kasuan, egokiena koloreari zuria gehitzea edo asetasuna txikiagotzea da, aldatutako kolorea irudikatzea posible izan arte. Horrela, kolorearen ñabardura eta argitasuna ez dira galduko. Bigarren multzoko koloreen zuzenketarako zenbait aukera ditugu. Irudi guztiko koloreak eskalatu egin daitezke, intentsitate handieneko kolorea irudikatzeko gai izan arte. Bestalde, kolorea mantenduz, intentsitatea eskala dezakegu. Eta azkenik, irudian gehien agertzen diren ñabardura eta intentsitatea mantenduz, koloreen asetasuna txikiagotu dezakegu, zuria gehituz.

Kapitulua 12

Animazio-teknikak

12.1 Sarrera

Ordenadore bidezko animazioa irudi jarraien kalkuluan oinarritzen da, irudi batetik besterako aldea bai objektu eta bai kameraren mugimenduaren ondoriozkoa izan daiteke. Animazio-mota hori filmeetan garatu den animazio-motaren parekoa da, eta adibideak ugari dira: plastilinazko pertsonaien animazioa, maketazioa erabiliz eraikitako espazio-untzien animazioa (gaur egun ordenadoreak erabiliz eraiki eta animatzen direnak), etab.

Ordenadore bidezko animazioak bi abantaila nagusi eskaintzen ditu:

1. Animazioan parte hartuko duten objektuak ez dira fisikoki eraiki behar, ordenadorea erabiliz eraikitzen baitira. Batzuetan, objektua ordenadorearen bidez eraikitzea, fisikoki eraikitzea bezain zaila izan daiteke; beste batzuetan, berriz, objektua fisikoki eraiki eta hiru dimentsiotako digitalizatzaile bat erabiliz, ordenadorera pasatzen da.

2. Ordenadorearen kamera edonon koka daiteke, kamera fisikoak ez bezala.

Orokorrean, filmeen kasuan behintzat, animazio bat egitean teknika ezberdinen konbinazioa erabiltzen da:

1. Ingurunea (mendiak, hiriak, ...) edo objektu batzuk maketen bidez eraikitzen dira, eta horien aldaketak, aldaketarik jasaten badute, frame edo irudi desberdinetan grabatzen dira. Maketak edo ingurune errealak grabatzen dira.

2. Ordenadorean eraikitako objektuak animatzen dira.

3. Maketen animazioari dagokion filmean, ordenadorearen animazioa integratzen da.

Nahiz eta gaur egun ordenadore bidezko animazioa gehien bat filme--industrian eta telebistako iragarkietan erabili, badirudi zientifikoen tresna ere bilakatzen ari dela.

12.2 Ordenadore bidezko animazio-teknikak

Ordenadorearen barnean ditugun objektuen animazioa garatzeko, metodo desberdinak jarrai daitezke:

- Behe-mailako kontrola: metodo horren bidez objektuek jasango dituzten aldaketak deskribatuko ditugu. Aldaketa horiek deskribatzeko, teknika desberdinez balia gaitezke: gidoi-sistemak edo scripting systems, keyframing edota spline bidez zuzendutako animazioa.

- Adierazpenaren animazioa: metodo horri esker, objektua definitzen duten datuak aldatuz, objektuak animazioan zehar jasango duen itxura-aldaketa defini dezakegu. Metodo horretan bi animazio-mota bereiz ditzakegu:

    1. Objektu artikulatuen animazioa: objektu artikulatua elkarren artean konektatutako zatiz osatzen da eta zati bakoitzaren mugimenduak muga batzuk ditu. Animazio-mota hori kalkulatzeko, bi teknika erabiltzen dira: alderantzizko zinematika eta aurreranzko zinematika.
    2. Objektu bigunen animazioa: animazio-mota horren bidez, objektuak deformatzea posible da, eta deformazio horiek animatzea ere bai.

- Prozedura bidezko animazioa: animazioa, programazio-lengoaia bat erabiliz kodetutako prozedura baten bidez definitzen da. Prozedura horren bidez, animazioa denboraren funtzio modura deskribatzen da.

- Portaera-animazioa: metodo hori objektuen portaeraren deskribapenean oinarritzen da. Objektuen portaera eta ingurunearekin duten elkarrekintza definitzen da.

- Animazio estokastikoa: kasu horretan, animazioaren ezaugarriak behe-mailako datuak itzultzen dituzten prozesu estokastikoek definitzen dituzte.

12.3 Behe-mailako kontrola

12.3.1 Gidoi-sistemak

Gidoi-sistemen teknika zaharrenetakoa da. Ordenadore bidezko animazioaren lehen pausoetan erabili zen eta gaur egun ez da gehiegi erabiltzen. Gidoia animazio-lengoaia batean idatzitako programa bat da. Beraz, teknika hori programatzaileentzat egokia izan daiteke baina ez artistentzat, artistek abstrakzio maila altuagoa behar baitute animazioak egiterakoan. Gaur egun denbora errealeko elkarrekintza eskaintzen duten sistemekin egiten da lan. Hala ere, teknika horren abantaila bat, animazio-errutinen bildumak edo liburutegiak garatzeko eskaintzen duen ahalmena da; hots, animazio-prozeduren liburutegiak garatzea posible da.

12.3.2 Keyframing

Teknika hori, Walt Disney-k erabiltzen zuen produkzio-sistemaren baliokide informatikoa da. Walt Disney-ren produkzio-sisteman, margolari trebeenek pertsonaia baten mugimendu- edo animazio-sekuentzia zailenak marrazten zituzten eta ondoren hain trebeak ez ziren marrazkilariek mugimendu horien arteko animazioa marrazten zuten. Marrazkilari trebeek margotzen zituzten irudiak, animazioko keyframe edo irudi garrantzitsuenak dira.

Produkzio-sistema hori informatikan aplikatzean, erabiltzaileak marrazkilari trebeen papera jokatzen du, eta adierazten dituen keyframeen arteko irudiak, ordenadoreak kalkulatzen ditu, interpolazioa erabiliz. Teknika hori objektuen edozein ezaugarriren aldaketak deskribatzeko erabil daiteke: kokapena, gardentasuna, kolorea ...

Keyframeen Teknika kasu oso sinpleetan bakarrik gertatuko zaigu erabilgarria. Gainera, interpolazioaren bidez kalkulatutako tarteko mugimenduak espero genituenak direnentz aztertu beharko dugu beti.

12.3.3 Spline bidez zuzendutako animazioa

Teknika horren bidez animazio osoa defini dezakegu, objektuen egoera splineen bidez adieraziz.

Adibide gisa, animazioa zuzentzeko splineen aplikazio sinple bat azalduko dugu, aplikazioak objektu bat $Q(u)$ bidetik $V(u)$ abiaduraz mugitzeko ahalmena emango digu. Animazio-sekuentzia bat lortzeko, momentu jakin batzuetan, objektuak, definitutako bidean duen kokapena kalkulatu beharko genuke.

Metodo horren abantaila nagusiak:

1. Mugimendu desberdinak deskribatzen dituzten splineen liburutegiak gara daitezke; horrela, animazio berri bat egitean, aurrez definitutako mugimenduak erabiltzeko aukera izango genuke.

2. Bidea eta abiadura, spline desberdinak erabiliz definitzen direnez, bide batentzat abiadura-kurba desberdinak proba daitezke, eta alderantziz.

Teknika horren bidez beste parametroak ere anima daitezke: objektuaren kolorea, gardentasuna, kokapena ... Zenbakien bidez adieraz daitekeen edozer anima daiteke.

12.4 Adierazpenaren animazioa

12.4.1 Objektu artikulatuen animazioa

Objektu artikulatuen animaziorako, bi teknika desberdin garatu dira: aurreranzko zinematika eta alderantzizko zinematika. Teknika horiek mekanikaren arloarekin erlazionatuta daude eta robotikan oso erabiliak dira. Lehenik eta behin, animatu nahi den gorputz artikulatuaren egitura definitzen da, ondoren, aipatutako teknikaren batekin animazioa edo puntu desberdinen kokapena kalkulatzeko.

Teknika horiek azaltzeko, pertsona baten animazioa erabiliko dugu eredutzat. Pertsona deskribatuko lukeen objektu artikulatua, pertsona baten eskeletoaren antzekoa izango litzateke.

Aurreranzko zinematika erabiltzean, egitura artikulatua zuhaitz moduan antolatzen da. Zuhaitzeko erpin batek jasaten dituen aldaketak, errekurtsiboki hedatzen dira bere semeetan: erpinetik semeetara, semeetatik horien semeetara etab. Erabiltzaileak, kokapen konkretu bat lortzeko zuhaitzaren

erroari aldaketa bat eragin eta jarraian, zati artikulatu desberdinak doituko ditu. Pertsonaren adibidean, zuhaitzaren erro gisa burua hartzen badugu, burua eskuinerantz biratzean, gorputz osoa eskuinerantz biratuko da. Kokapena gehiago zehaztu nahi izanez gero, adibidez, aldaka biratuko genuke eta ondorioz hankek automatikoki biratuko lukete, zuhaitzean aldakaren seme modura agertuko lirateke-eta. Hori guztia objektuaren kokapen bakar bat adierazteko egin behar da; objektua animatzeko, zuhaitzeko osagai bakoitzari denboran zehar eragin beharreko aldaketak definitu beharko genituzke, B-splineak erabiliz adibidez.

Erabiltzaileak datu asko eman behar dizkio ordenadoreari. Beraz, teknika hori erosoa ez dela esan daiteke; era berean, animazio-egileari askatasun gehiago ematen dio, erabaki gehienak animazio-egileak hartzen baititu eta ez ordenadoreak.

Alderantzizko zinematikak, berriz, objektu artikulatuaren gorputzadarren kokapenak adieraziz egiten du lan. Erabiltzailea da gorputzadarren kokapenak ematen dituena, eta ondoren ordenadoreak egitura artikulatua osatzen duten gainontzeko puntuen kokapena kalkulatzen du. Horretarako, erabiltzaileak finkatutako murrizketak erabiltzen dira. Pertsonaren kasuan, murrizketa bat bi puntu lotzen dituen zatiaren luzeran oinarrituko litzateke. Adibidez, besoa luzeegia edo laburregia ez izateko. Horrela, pertsona deskribatzen duen egitura artikulatuaren gorputzadarrak (hankak, besoak eta burua) non edo non finkatuz, gainontzeko puntuen kokapena kalkulatzen da, murrizketak ahaztu gabe. Pertsona animatzeko, keyframeen teknika erabil genezake. Zenbait momentutan gorputzadarren kokapenak adierazi-ko genituzke, eta gainontzekoak lortzeko, interpolazioaz baliatuko ginateke. Ondoren, gorputzadarren kokapenak ezagutuz beste puntuenak kalkulatzeko, alderantzizko zinematika erabili beharko genuke.

12.4.2 Objektu bigunen animazioa

Oraingoan, objektu deformagarriekin egingo dugu lan. Objektu horien animazioa egiteko, deformazio desberdinak eragingo dizkiogu objektuari denboran zehar. Deformazio sinpleenak, poligonozko objektu baten erpinak edo gainazal parametriko baten kontrol-puntuak mugituz lor ditzakegu. Eragin daitezkeen deformazioak, objektuaren adierazpenaren menpe daude.

Lehenik, objektuen adierazpenak deformazioa eragiteko moduan duen zerikusia azalduko dugu. Gero, adierazpenarekiko independenteak diren deformazio-teknikak aipatuko ditugu. Eta azkenik, deformazioa animatzeko

teknikak ikusiko ditugu.

Objektuen adierazpena eta deformazioa

Poligono bidez adierazitakoaren deformazioa: Poligonozko objektu bat deformatzeko, bere erpinak mugitu beharko ditugu. Hala ere, erpinak independenteki mugituz, zentzugabeko objektu deformatuak lor ditzakegu. Hori gerta ez dadin, erpinen arteko auzokidetza erlazioak mantentzea komeni da.

Poligono bidezko adierazpena, benetako objektuaren lagin bezala har daiteke, lagin hori erpin konektatuen multzoa besterik ez baita. Zenbat eta erpin gehiago edo lagin handiagoa erabili, orduan eta adierazpen hobea lortuko dugu benetako objektuari dagokionez. Deformazioa, lagin horietan eragindako funtzio modura har daiteke. Erpin gutxiko objektu baten gain deformazio konplexu bat eragitean aurpegi ez planarrak lor daitezke; horregatik, kontuz ibili behar da.

Gainazal parametrikoen deformazioa: Deformazioari buruz hitz egitean, adierazpide parametrikoak abantaila handi bat azaltzen du poligono bidezkoaren aurrean. Ez du axola deformazioaren konplexutasunak, objektu parametrikoei dagokien irudia nahi bezain zehatz lor baitezakegu; poligono bidezko adierazpenean ez bezala, objektuak ez du inolako degradaziorik jasango.

Kontrol-puntuak, adierazi nahi den benetako gainazal edo objektuaren lagintzat har daitezke. Kontrol-puntu gutxiegi erabiliz gero, zenbait deformazio lortzea ezinezkoa izango da, hau da, bi kontrol-punturen arteko gainazal-zatitxoetako deformazioak hain zuzen ere.

Deformaziorako erabil daitezkeen zenbait gainazal:

1. Bezier txatalak: Bezier txatal bakar batekin lan egitean, bere kontrol-puntuak edozein tokitan koka ditzakegu. Bezier txatalen multzo batekin, hau da, Bezier gainazal batekin lan egitean, berriz, gainazala definitzen duten kontrol-puntuak ezin ditugu edonora mugitu, txatalen arteko jarraitasun-baldintzak betetzea beharrezkoa baita.

   Azaldutako jarraitasun-murrizketak kontuan hartu beharraren desabantailak hauexek dira:

   (a) Deformazioa egitean plater-efektua agertzen da (ikus 4.45 irudia).

(b) Ezin dira deformazio lokalak, hots, gainazaleko eskualde konkretu batean ematen diren deformazioak, lortu. Eskualde konkretu bateko kontrol-puntuak mugitzean, jarraitasuna mantentzeko, inguruko kontrol-puntuak mugitu beharko genituzke, eta deformazioa inguruko txataletara hedatuko litzateke.

2. B-spline txatalak: Gainazalak eraikitzean B-splineak erabiltzea Bezier oinarriak erabiltzea baino komenigarriagoa da. Arrazoi berdinengatik, deformazioak eragitean B-splineak Bezierren oinarriak baino egokiagoak dira. Gogora ditzagun arrazoi horiek:

   (a) Jarraitasun-murrizketarik erabili gabe, C^2 jarraitasuna lortzen dugu, B-spline oinarriaren ezaugarriei esker.

   (b) Jarraitasun-murrizketa ezari eta B-spline oinarriaren ezaugarriei esker, deformazio lokalak eragitea posible da.

Adierazpidearekiko independenteak diren deformazio-teknikak

Bi teknika azalduko ditugu. Lehenengoa, deformazio ez-linealean oinarritzen da. Bigarrenak, berriz, espazioa deformatuz itxuraldatzen ditu objektuak.

1. Orain arte, objektu bati aldaketaren bat (leku-aldaketa, biraketa, neurri-aldaketa ...) eragitean, aldaketa inplementatzen zuen matrizea, objektuaren erpin bakoitzaren koordenatuei biderkatzen genion. Aldaketa eragiten ari ginen bitartean, ez genuen matrizearen definizioa aldatzen. **Deformazio ez-lineala**, aldaketa eragiten duen matrizearen aldaketan oinarritzen da; aldaketa eragiten goazen heinean matrizearen definizioa aldatzen dugu. Horrela, objektuaren erpin guztietan ez da aldaketa berdina eragiten. Aldaketa-matrizea, erpinaren kokapenarekiko funtzio modura kalkula daiteke:

$$(X, Y, Z) = F(x, y, z)$$

(x, y, z) deformatu gabeko objektuaren erpin baten koordenatuak dira, eta koordenatuen gainean eragindako aldaketa, erpinaren koordenatuen araberako funtzioa da. Adibidez, neurri-aldaketa:

$$(X, Y, Z) = (s_x x, s_y y, s_z z)$$

Eta objektu bat z ardatzean zehar meheago egiteko:

$$(X, Y, Z) = (rx, ry, z) \quad \text{non} \quad r = f(z)$$

$f(z)$ funtzioa matematikoki definitu beharrean, kurba grafikoen bidez marraz dezakegu, kurben erabilera definizio matematikoarena baino intuikorragoa baita. Bestalde, aipatutako aldaketak denboran zehar eraldatuz, animaziorako oso erabilgarri izango den deformazio-tresna gara dezakegu.

2. **FFD** edo forma askeko deformazioa: FFD (free form deformation) teknika erabiliz, objektua zuzenean deformatu beharrean, espazioa deformatzen dugu, eta objektua espazio horren barnean dagoenez, deformatu egiten da. Suposa dezagun objektu bigun bat plastikozko kutxa batean sartzen dugula; kutxa deformatzean, barneko objektua ere deformatu egingo da.

   Deformatu behar den espazioa mugatzeko eta deformatzeko, espazio-zati bat hipertxatal trikubiko baten bidez inguratzen dugu. Bezier hipertxatal trikubiko baten definizioa hauxe da:

$$Q(u, v, w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} p_{ijk} B_i(u) B_j(v) B_k(w)$$

   non $B_i(u)$, $B_j(v)$ eta $B_k(w)$ direlakoak 3. mailako Bernstein-en polinomioak diren. Hipertxatala 64 kontrol-puntuz osatuta dago eta u, v, w parametroen bidez parametrizatuta dago, $u, v, w \in [0, 1]$. Hipertxatala, deformagarria den kubo modura ikus daiteke. Horren kontrol-puntuak mugituz (64 kontrol-puntu), espazioa aldatuko dugu eta espazio berriari dagozkion objektuaren erpin berriak kalkula daitezke.

   FFD-en teknika erabiliz, emaitza ikusgarriak lortzen dira. Ideia horri jarraituz, zenbait hedaketa garatu dira. Horiek, deformazio-espazioa definitzeko kubo baten ordez, beste objektu bat erabiltzean oinarritzen dira.

Deformazioaren animazioa

Deformazioak animatzeko, erpinak edo kontrol-puntuak modu egokian mugitu beharko ditugu denboran zehar. Bi teknika azalduko ditugu (biak FFD-etan oinarritzen dira):

1. Objektu bat animatzeko modu egoki bat, objektua inguratzen duen FFD-a animatzean datza. FFD-a animatzeko deformazio ez-linealen animazioa, keyframeak, spline bidezko animazioa etab. erabil ditzakegu. Teknika hori benetan ahaltsua da.

2. Bigarren aukerak ere, FFD-ak erabiltzen ditu. Edozein espazio-zatitan eragin daiteke FFD motako deformazioa, eta objektu bat espazio-zati horretan sartzen ari garen heinean, barruan dagoen zatiak deformazioa jasango du. Bigarren teknika, objektuak FFD bat eragiten diogun espazio-zatia zeharkatzean jasaten duen deformazioan oinarritzen da. FFD-ari itxura egokia emanez, nahi diogun deformazioa eragingo diogu objektuari. Adibide batez azalduko dugu. Har dezagun plano batez adierazitako paper bat; espazioan kokapena eduki beza, eta paperetik urrun S itxurako FFD bat jar dezagun. Papera dagoen tokitik FFD-ra mugitzen dugu, eta FFD-a zeharkatzen ari den bitartean, deformazio bat jasango du. Papera S itxura hartuz joango da, berriro plano bihurtuz FFD-tik ateratzean.

Beti gertatzen den bezala, emaitza egokienak teknika desberdinak konbinatuz lortzen dira. Horren adibide bat, pertsonen animaziorako garatu zen metodo batean ikus daiteke: Pertsonaren eskeletoa objektu artikulatu moduan eraiki zen, bere gain aurreranzko edo alderantzizko zinematika-teknikak eragiteko. Eskeletoaren animazioa egin ondoren, eskeletoaren konfigurazioa edo kokapena erabiliz, gorputzaren giharren itxura kalkulatu zen, eta giharren itxura FFD deformazioak erabiliz kalkulatu zen. FFD-ak kalkulatu ondoren, gorputza adierazten zuen poligonozko objektua deformatu zen.

12.5 Prozedura bidezko animazioa

Animazioa, programazio-lengoaia bat erabiliz kodetutako prozedura baten bidez definitzen da. Hiru teknika-mota azalduko ditugu.

12.5.1 Partikula-multzoen animazioa

Animazio-mota hori sua, hodeiak, etab. modelatzeko erabili da. Animazioetan sortuz eta desagertuz doazen partikula asko erabiltzen dira, eta partikula bakoitza argi-iturri modura hartzen da. Animazio bat diseinatzean, sortze-desagertzearen prozesua eta iraupen-prozesua definitu behar dira.

- Sortze-desagertzearen prozesuak honako hauek kontrolatzen ditu: partikulak sortzen diren eskualdearen itxura (zirkunferentzi itxurakoa, elipsoidala, ...) eta kokapena, sortzen diren partikulen ezaugarriak, desagertu behar diren partikulen ezabapena. Eta datu horiekin, bizirik dirauten partikulei dagokien irudia lortu behar da. Momentu jakin bakoitzean, zenbait partikula sortu beharko da, eta horretarako, ausazko formulak edo denboraren araberako formulak erabili beharko dira.

- Eta iraupen-prozesuak bizirik dirauten partikulen ezaugarriak aldatzen ditu: kokapena, abiadura, mugimenduaren norabide-bektorea, gardentasuna, neurria eta hiltzeko geratzen den denbora.

Leherketa edo eztanda baten kasuan partikulek jarraituko luketen mugimendua kalkulatzeko, fisikaren tiro parabolikoaren formulak erabiliko genituzke.

12.5.2 Portaeraren animazioa prozedura bidez

Objektuen portaera deskribatuz, animazioa definituko dugu. Animazioan parte hartzen duten objektuak partikulatzat har ditzakegu. Horien kokapena eta abiadura, jarraituko duten portaeraren bidez kalkulatu ahal izango dugu. Teknika horren bidez, animalia-taldeen mugimenduak modelatu izan dira. Teknika horrek aurrekoarekiko agertzen dituen desberdintasunak hauexek dira:

1. Partikulak ez dira independenteak. Partikula bakoitzak inguruan dituenekin elkarreragiten du, taldeari portaera bat egokituz.

2. Partikulak ez dira argi-iturri modura hartzen, hots, partikulak objektuak dira (poligonoz osatuak, gainazal parametrikoak ...).

Argi dago, desberdintasun garrantzitsuena lehenengoa dela. Adibide gisa, txori-taldeen animazioa azalduko dugu. Taldea mugitzeko helburu-puntu bat mugituko dugu. Txori bakoitzaren portaera honakoa izan daiteke: txoriek ez dute elkarren arten talka egingo eta, era berean, batak bestearengatik gertu egon nahi du. Txori bakoitzaren portaera lehentasun desberdineko erregelen bidez lor daiteke:

1. Inguruan dauden txoriekin eman daitezken talkak ekiditen dira.

2. Txoria, inguruan dauden txorien abiadurarekin mugituko dugu.

3. Txoria, inguruan dauden txoriekiko gertu mantenduko dugu.

Erregela horiek talde bateko txorien portaera definitzen dute. Erregela gehiago gehituz, talde desberdinak biltzea (gertu daudenean) edo talde bat zatitzea (oztopo batekin aurkitzen denean) lor dezakegu.

12.5.3 Eredu analitikoen animazioa

Animazio desberdinak formula matematikoen bidez modelatzean oinarritzen da teknika hori. Adibide gisa, uhinen edota sugeen mugimendua aipa dezakegu. Uhinen mugimendua maiztasun, fase eta anplitude desberdineko sinuen konposizio modura modela daiteke. Sugeen kasuan, gorputzaren hiru zatik mugimendu desberdinak deskribatzen dituzte. Sugearen buruaren mugimendua anplitude txikiko eta maiztasun handiko sinu funtzio modura modela dezakegu; gorputzaren mugimendua, anplitude handiagoko eta maiztasun txikiagoko sinu modura; eta buztana, anplitude handiko eta maiztasun oso txikiko sinu modura. Sinuak erabili beharrean, B-splineak ere erabil daitezke.

Eranskina A

Spline marrazketarako programa

A.1 Zenbait definizio: globala.def

Programak erabiltzen dituen zenbait datu: leiho-neurriak, splinearen puntu--kopurua, kontrol-puntuen kopurua ...

```
#define TRUE 1
#define FALSE 0
#define PI 3.14159
#define KORAPILO_KOP_MAX 40
#define KONTROL_PUNTU_KOP_MAX KORAPILO_KOP_MAX-4
#define PANTAILA_TARTEA 250.0 /* 250 pixel marraztuko ditugu */
#define PUNTU_KOP 2000.0 /* splinearen 2000 puntu kalkulatuko ditugu*/
#define Y_ESKALA 50
#define KARAKTERE_ZABALERA 4
#define KONTROL_PUNTUAREN_ERRADIOA 5
#define XJATORRIA 210
#define YJATORRIA 150
#define LEIHO_ZABALERA 2*XJATORRIA
#define LEIHO_ALTUERA 2*YJATORRIA
#define YARD 100
#define YARDATZA YJATORRIA+YARD /* oinarriaren marrazgunea */
```

A.2 Erazagupen globalak: globala.h

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include "globala.def"
typedef struct {
    int xkoord,ykoord;
} xykoord;
extern int KorapiloKop;
extern int KontrolPuntuKop;
extern int Korapilo[];
extern xykoord KontrolPuntu[];
extern unsigned short int Koordy;
extern Display *Konexioa;
extern Window Leihoa;
extern GC TIG1,TIG2; /* leihoak erabiliko dituen ingurune grafikoak */
extern int Pantaila;
extern unsigned long ArkatzKolorea,PaperKolorea;
void PoligonoaMarraztu(void); /* funtzio erazagupena */
```

A.3 Poligonoa marrazteko funtzioa

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include "globala.h"
void PoligonoaMarraztu(void)
    {
    int puntua;
    xykoord azkenpuntua;
    azkenpuntua.xkoord = KontrolPuntu[0].xkoord;
    azkenpuntua.ykoord = KontrolPuntu[0].ykoord;
    for(puntua=1;puntua<KontrolPuntuKop;puntua++)
       {
       XDrawLine(Konexioa,Leihoa,TIG2,
                  azkenpuntua.xkoord+XJATORRIA,
                  azkenpuntua.ykoord+YJATORRIA,
                  KontrolPuntu[puntua].xkoord+XJATORRIA,
```

```
                KontrolPuntu[puntua].ykoord+YJATORRIA);
        azkenpuntua.xkoord = KontrolPuntu[puntua].xkoord;
        azkenpuntua.ykoord = KontrolPuntu[puntua].ykoord;
        }
    for(puntua=0;puntua<KontrolPuntuKop;puntua++)
        {
        XFillArc(Konexioa,Leihoa,TIG2,
                KontrolPuntu[puntua].xkoord+XJATORRIA-
                KONTROL_PUNTUAREN_ERRADIOA/2,
                KontrolPuntu[puntua].ykoord+YJATORRIA-
                KONTROL_PUNTUAREN_ERRADIOA/2,
                KONTROL_PUNTUAREN_ERRADIOA,
                KONTROL_PUNTUAREN_ERRADIOA,0,64*360);
        }
    }
```

A.4 B-splinearen hasieraketa

```
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Xutilh>
#include "globala.def"
#include "globala.h"
#include "poligono.h"
/* Oharra: korapiloak.cfg fitxategiaren barnean spline ez razional
bat definitzen duen korapilo-bektorea gordeko dugu */
int KorapiloKop;
int KontrolPuntuKop;
int Korapilo[KORAPILO_KOP_MAX+1];
xykoord KontrolPuntu[KONTROL_PUNTU_KOP_MAX];
unsigned short int Koordx=0;
unsigned short int Koordy=0;
void ArdatzaMarraztu(void);
void OinarriaMarraztu(void);
void BsplineaMarraztu(void);
double Bspline_Cox(int i,int j,double t);
```

```
void BsplineaHasieratu(void)

   {
   int korapiloa,puntua,x,y;
   FILE *korapiloak,*puntuak;
   char fitxategikolerroa[60];
   int knot;
   /* korapilo-bektoreen konfigurazio-fitxategia irakurri behar dugu;
   lerro bakoitzak ezin du 60 karaktere baino gehiago eduki */
   korapiloak = fopen("korapiloak.cfg","rt");
   /* lehenengo lerroa = komentarioak */
   fgets(fitxategikolerroa,sizeof(char)*60,korapiloak);
   fgets(fitxategikolerroa,sizeof(char)*60,korapiloak);
   KorapiloKop = atoi(fitxategikolerroa);
   KontrolPuntuKop = KorapiloKop-4;
   /* komentariozko beste lerro bat */
   fgets(fitxategikolerroa,sizeof(char)*60,korapiloak);
   for(korapiloa=0;korapiloa<KorapiloKop;korapiloa++)
      {
      fgets(fitxategikolerroa,sizeof(char)*60,korapiloak);
      knot = atoi(fitxategikolerroa);
      Korapilo[korapiloa] = knot;
      }
   fclose(korapiloak);
   puntuak = fopen("puntuak.cfg","rt");
   /* lehenengo lerroa eta bigarren lerroa = komentarioak */
   fgets(fitxategikolerroa,sizeof(char)*60,puntuak);
   fgets(fitxategikolerroa,sizeof(char)*60,puntuak);
   for(puntua=0;puntua<KontrolPuntuKop;puntua++)
      {
      fgets(fitxategikolerroa,sizeof(char)*60,puntuak);
      sscanf(fitxategikolerroa,"%d %d",&x,&y);
      KontrolPuntu[puntua].xkoord=x;
      KontrolPuntu[puntua].ykoord=y;
      }
   fclose(puntuak);
   }
```

A.5 Ardatz- eta oinarri-marrazketa

Programak leihoan kontrol-puntuen araberako splinea marrazteaz gain, oinarriko splineak ere marrazten ditu; oinarria korapilo-bektorearen araberakoa da eta hori ere kalkulatu behar da. Horretarako, 4.6.1 ataleko algoritmo errekurtsiboa erabiltzen du, eta ardatz batean KorapiloKop-4 oinarrizko splineak kokatzen ditu. Hasteko, ardatza marrazten du eta bertan tarte bakoitzeko marratxoa jartzen du.

```
void ArdatzaMarraztu(void)
    {
    double x_lag;
    int i,x;
    char karaktereak[5];
    x_lag = -(PANTAILA_TARTEA/2.0);
    XDrawLine(Konexioa,Leihoa,TIG1,x_lag+XJATORRIA,YARDATZA,
                -x_lag+XJATORRIA, YARDATZA);
    for(i=0;i<=Korapilo[KorapiloKop-1];i++)
       {
       x = x_lag;
       XDrawLine(Konexioa,Leihoa,TIG1,x+XJATORRIA,YARDATZA,
                   x+XJATORRIA, YARDATZA+3);
       sprintf(karaktereak,"%d",i);
       XDrawString(Konexioa,Leihoa,TIG1,
                     x-(KARAKTERE_ZABALERA/2)+XJATORRIA,
                     YARDATZA+16, karaktereak,strlen(karaktereak));
       if (i==Korapilo[3])
          XDrawLine(Konexioa,Leihoa,TIG2,x+XJATORRIA,YARDATZA,
                            x+XJATORRIA, YARDATZA-Y_ESKALA);
       if (i==Korapilo[KorapiloKop-4])
          XDrawLine(Konexioa,Leihoa,TIG2,x+XJATORRIA,YARDATZA,
                            x+XJATORRIA, YARDATZA-Y_ESKALA);
       x_lag += PANTAILA_TARTEA*
                (1.0/Korapilo[KorapiloKop-1]);
       }
    }
```

Ardatza marrazteaz gain, oinarriko KorapiloKop-4 splineak marraztu behar ditu; spline bakoitzeko PUNTU_KOP adina puntu kalkulatzen ditu, baina, ondoko funtzioak spline bakoitzeko puntu bana kalulatu eta marrazten ditu, hurrengo puntura pasa aurretik.

```
void OinarriaMarraztu(void)
    {
    double t,x_lag,y_lag;
    int i;
    unsigned short x,y;
    ArdatzaMarraztu();
    x_lag = -(PANTAILA_TARTEA/2.0);
    for(t=0;t<Korapilo[KorapiloKop-1];
            t+=Korapilo[KorapiloKop-1]/PUNTU_KOP)
       {
       /* oinarriko funtzio bakoitzeko puntu bat marrazten dugu */
       for(i=0;i<KorapiloKop-4;i++)
          {
          y_lag = -(Y_ESKALA*Bspline_Cox(i,4,t));
          y = YARD+y_lag;
          x = x_lag;
          XDrawPoint(Konexioa,Leihoa,TIG1,XJATORRIA+x,
                  YJATORRIA+y);
          }
       x_lag += PANTAILA_TARTEA/PUNTU_KOP;
       }
    }
```

A.6 B-splinearen marrazketarako funtzioa

Splinea marrazteko, PUNTU_KOP adina puntu kalkulatzen ditu; horretarako, oinarriko spline bakoitzak bertan duen balioa ezagutu behar da, eta hori 4.6.1 ataleko algoritmo errekurtsiboaren bidez kalkulatzen du; ondoren, balio horietako bakoitzari kontrol-puntuaren koordenatua biderkatzen dio.

```
void BsplineaMarraztu(void)
{
double t,t_gehikuntza;
double x,y;
int puntua;
t_gehikuntza = Korapilo[KorapiloKop-1]/PUNTU_KOP;
/* korapilo[3]==0 -> t=0 -> lehenengo puntua,
lehenengo kontrol-puntuaren posizioan */
/* marraztuko litzateke, posizio okerrean */
for(t=Korapilo[3]+t_gehikuntza;t<Korapilo[KorapiloKop-4];
   t+=t_gehikuntza)
   {
   x=y=0;
   for(puntua=0;puntua<KontrolPuntuKop;puntua++)
      {
      x += KontrolPuntu[puntua].xkoord*Bspline_Cox(puntua,4,t);
      y += KontrolPuntu[puntua].ykoord*Bspline_Cox(puntua,4,t);
      }
   XDrawPoint(Konexioa,Leihoa,TIG1,XJATORRIA+x,
               YJATORRIA+y);
   }
}
```

A.7 Oinarriko splineen kalkulua

Oinarriko funtzioen balioak kalkulatzeko, 4.6.1 ataleko algoritmoa erabiltzen da. Errekurtsibitatean oinarritzen den algoritmoa da eta, gainera, kontuz ibili behar da, zatitzaile modura 0 zenbakia eduki baitezake.

```
double Bspline_Cox(int i,int j,double t)
{
double zatik1,zatik2;
if(j==1)
   {
   if((Korapilo[i]<=t) && (t<=Korapilo[i+1]))
      return(1.0);
```

```
    else
        return(0.0);
    }
  else if(j==2)
    {
    if(Korapilo[i+1]-Korapilo[i]==0)
        zatik1 = 0;
    else
        zatik1 = (t-Korapilo[i])/(Korapilo[i+1]-Korapilo[i]);
    if(Korapilo[i+2]-Korapilo[i+1]==0)
        zatik2 = 0;
    else
        zatik2 = (Korapilo[i+2]-t)/(Korapilo[i+2]-Korapilo[i+1]);
    return (zatik1*Bspline_Cox(i,j-1,t)+
                    zatik2*Bspline_Cox(i+1,j-1,t));
    }
  else if(j==3)
    {
    if(Korapilo[i+2]-Korapilo[i]==0)
        zatik1 = 0;
    else
        zatik1 = (t-Korapilo[i])/(Korapilo[i+2]-Korapilo[i]);
    if(Korapilo[i+3]-Korapilo[i+1]==0)
        zatik2 = 0;
    else
        zatik2 = (Korapilo[i+3]-t)/(Korapilo[i+3]-Korapilo[i+1]);
    return (zatik1*Bspline_Cox(i,j-1,t)+
                    zatik2*Bspline_Cox(i+1,j-1,t));
    }
  else /*j==4*/
    {
    if(Korapilo[i+3]-Korapilo[i]==0)
        zatik1 = 0;
    else
        zatik1 = (t-Korapilo[i])/(Korapilo[i+3]-Korapilo[i]);
    if(Korapilo[i+4]-Korapilo[i+1]==0)
        zatik2 = 0;
    else
```

```
        zatik2 = (Korapilo[i+4]-t)/(Korapilo[i+4]-Korapilo[i+1]);
    return (zatik1*Bspline_Cox(i,j-1,t)+
                zatik2*Bspline_Cox(i+1,j-1,t));
    }
}
```

A.8 Programaren hasieraketa

Marrazteko erabiliko den leihatilaren hasieraketaz arduratzen da ondoko funtzioa. Leihoa sortzeaz gain, bere ezaugarriak ezartzen ditu: izena, koloreak eta bertan xaguarekin eragin daitezkeen gertaerak.

```
#include <stdio.h>
#include <limits.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include "globala.h"
#include "bspline.h"
Display *Konexioa;
Window Leihoa;
GC TIG1,TIG2; /* leihoak erabiliko dituen
                 ingurune grafikoak */
int Pantaila;
XSizeHints LeihoEzaugarriak;
unsigned long ArkatzKolorea,PaperKolorea;


void Hasieraketa(void)
    {
    char *leiho_izena,*ikono_izena;
    XTextProperty leiho_titulua,ikono_titulua;
    /* Zerbitzariarekin konektatu, leihoa sortu, */
    /* grafiko-ingurunea definitu eta jaso nahi ditudan
    gertaerak adierazi */
    leiho_izena=(char *)malloc(sizeof(char)*20);
```

```
ikono_izena=(char *)malloc(sizeof(char)*20);
strcpy(leiho_izena,"Nere Leihoa");
strcpy(ikono_izena,"Nere Leihoa");
Konexioa = XOpenDisplay(NULL);
Pantaila = DefaultScreen(Konexioa);
ArkatzKolorea = BlackPixel(Konexioa,Pantaila);
PaperKolorea = WhitePixel(Konexioa,Pantaila);
LeihoEzaugarriak.x = LeihoEzaugarriak.y = 90;
LeihoEzaugarriak.width = LEIHO_ZABALERA;
LeihoEzaugarriak.height = LEIHO_ALTUERA;
LeihoEzaugarriak.flags = PPosition | PSize;
Leihoa = XCreateSimpleWindow(Konexioa,
      DefaultRootWindow(Konexioa),
      LeihoEzaugarriak.x,LeihoEzaugarriak.y,LeihoEzaugarriak.width,
      LeihoEzaugarriak.height,5,ArkatzKolorea,PaperKolorea);
/* Izenak testu propietatetara itzuli */
XStringListToTextProperty(&leiho_izena,1,&leiho_titulua);
XStringListToTextProperty(&ikono_izena,1,&ikono_titulua);
/* Ikonoari eta leihoari izena eman */
XSetWMName(Konexioa,Leihoa,&leiho_titulua);
XSetWMIconName(Konexioa,Leihoa,&ikono_titulua);
TIG1 = XCreateGC(Konexioa,Leihoa,0,0);
TIG2 = XCreateGC(Konexioa,Leihoa,0,0);
XSetForeground(Konexioa,TIG1,ArkatzKolorea);
XSetForeground(Konexioa,TIG1,ArkatzKolorea);
XSetBackground(Konexioa,TIG2,PaperKolorea);
XSetBackground(Konexioa,TIG2,PaperKolorea);
XSetLineAttributes(Konexioa,TIG1,0,LineSolid,CapButt,
      JoinMiter);
XSetLineAttributes(Konexioa,TIG2,0,LineOnOffDash,CapButt,
      JoinMiter);
XSelectInput(Konexioa,Leihoa,ExposureMask | ButtonPressMask
            | PointerMotionMask);
XMapWindow(Konexioa,Leihoa);
}
```

A.9 Kontrol-puntuak mugitzeko funtzioak

Erabiltzaileak mugitu nahi duen kontrol-puntua zein den jakiteaz arduratzen da ondoko funtzioa. Kontrol-puntu horiek tokiz aldatu nahi dituenean xaguarekin bat aukeratu eta kokapen berrira eramateko lehenengo zein aukeratu duen jakin behar da, eta horretaz arduratzen da funtzioa.

```
int XagutikGertuena(int *x, int *y)
{
int puntua,dist1,dist2,puntumugitua;
int leiho_nagusiax,leiho_nagusiay,leiho_semeax,leiho_semeay;
unsigned int xaguaren_teklak;
Window leiho_nagusia,leiho_semea;
XQueryPointer(Konexioa,Leihoa,&leiho_nagusia,&leiho_semea,
                  &leiho_nagusiax,&leiho_nagusiay,&leiho_semeax,
                  &leiho_semeay,&xaguaren_teklak);
/* funtzio honetan sartu bada, nire leihoan gertaera bat
sortu delako da; beraz, ez daukat xagua zein leihotan
dagoen aztertu beharrik. Goiko funtzioak itzulitako
koordenatuak nire leihoarekiko erlatiboak izango
dira, eta ez beste leihoarekiko erlatiboak */
puntumugitua=-1;
dist1 = INT_MAX;
for(puntua=0;puntua<KontrolPuntuKop;puntua++)
   {
   dist2 = abs(KontrolPuntu[puntua].xkoord-(leiho_semeax-
         XJATORRIA))+abs(KontrolPuntu[puntua].ykoord-
         (leiho_semeay-YJATORRIA));
   if (dist2 < dist1)
      {
      *x = KontrolPuntu[puntua].xkoord;
      *y = KontrolPuntu[puntua].ykoord;
      dist1 = dist2;
      puntumugitua = puntua;
      }
   }
return(puntumugitua);
}
```

Ondoko funtzioa aukeratutako kontrol-puntua zein den jakiteaz eta kokapen berrian jartzeaz arduratzen da, horretarako XagutikGertuena deituriko funtzioa erabiltzen du.

```
void KontrolPuntuakEguneratu(int kontrolpuntumugitua, int *x, int *y)
    {
    int leiho_nagusiax,leiho_nagusiay,xagu_x,xagu_y;
    unsigned int xaguaren_teklak;
    Window leiho_nagusia,leiho_semea;
    int puntua;
    /* erabiltzailea mugitzen ari den puntua kontrol-puntu errepikatu bat
    izan daiteke. Xaguaren posizioa irakurtzen dugu */
    XQueryPointer(Konexioa,Leihoa,&leiho_nagusia,&leiho_semea,
                      &leiho_nagusiax,&leiho_nagusiay,&xagu_x,
                      &xagu_y,&xaguaren_teklak);
    for(puntua=kontrolpuntumugitua;puntua<KontrolPuntuKop;
          puntua++)
       {
       if((KontrolPuntu[puntua].xkoord==*x) &&
          (KontrolPuntu[puntua].ykoord==*y))
          {
          KontrolPuntu[puntua].xkoord=xagu_x-XJATORRIA;
          KontrolPuntu[puntua].ykoord=xagu_y-YJATORRIA;
          }
       }
    *x = xagu_x-XJATORRIA;
    *y = xagu_y-YJATORRIA;
    }
```

A.10 Programa nagusia

Programa nagusiak, X sistema erabiltzen duenez, leiho bat irekitzen du Hasieraketa funtzioa erabiliz; ondoren, splinea hasieratzen du, fitxategietan dagoen informazioarekin, eta azkenik, erabiltzailearen aginduak betetzeko zikloan jartzen da. Agindu horiek hiru motatakoak izan daitezke: lehenengo

aginduak leihoaren birmarrazketa egin behar dela adierazten du, bigarrenak xagua leiho barnean mugitzen ari garela adierazten du; kasu horretan, kontrol-puntua mugitzen ari bagara, xaguaren kokapenera eramango dugu eta ondoren splinea birmarraztu. Hirugarren aginduak botoiren bat sakatu duela adierazten du; eskuineko botoia sakatu badu, splineari dagokion irudia fitxategi batean gorde eta bukatu egiten da programa; beste botoiren bat sakatu badu, kontrol-puntua mugitu nahi duela ulertzen du programak, eta horri ekiten dio, berriz botoia sakatu bitartean.

```
int main(void)
    {
    XEvent ebentua,ebentua2;
    Pixmap pixmap;
    XWindowAttributes leiho_atributuak;
    int kont,bukatu;
    int kontrolpuntuamugitzen=0;
    int kontrolpuntumugitua,puntuax,puntuay;
    bukatu=0;
    Hasieraketa();
    XClearWindow(Konexioa,Leihoa);
    BsplineaHasieratu();
    while(!bukatu)
       {
       XNextEvent(Konexioa,&ebentua);
       switch(ebentua.type)
          {
          case Expose:
             OinarriaMarraztu();
             BsplineaMarraztu();
             PoligonoaMarraztu();
             break;
          case MotionNotify:
             if (kontrolpuntuamugitzen)
                {
                kont=0;
                if (XEventsQueued(Konexioa,QueuedAfterReading)>5)
                      {
```

```
                while(kont<5)
                    {
                    XPeekEvent(Konexioa,&ebentua2);
                    if(ebentua2.type!=MotionNotify) kont=5;
                       else XNextEvent(Konexioa,&ebentua);
                    kont++;
                    }
                }
            XFlush(Konexioa);
            sleep(2); /* itxaron, erabiltzaileak marrazkia
                         ikus dezan */
            XClearArea(Konexioa,Leihoa,0,0,LEIHO_ZABALERA,
                       YARDATZA-Y_ESKALA,0);
            KontrolPuntuakEguneratu(kontrolpuntumugitua,&puntuax,
                                    &puntuay);
            BsplineaMarraztu();
            PoligonoaMarraztu();
            }
        break;
    case ButtonPress:
        if (ebentua.xbutton.button==3)
            {
            XGetWindowAttributes(Konexioa,Leihoa,
                                 &leiho_atributuak);
            pixmap = XCreatePixmap(Konexioa,Leihoa,
                                   LEIHO_ZABALERA, LEIHO_ALTUERA,
                                   leiho_atributuak.depth);
            XSetFunction(Konexioa,TIG1,GXcopyInverted);
            XCopyArea(Konexioa,Leihoa,pixmap,TIG1,0,0,
                  LEIHO_ZABALERA,LEIHO_ALTUERA,0,0);
            XWriteBitmapFile(Konexioa,"bitmap.bmp",pixmap,
                  LEIHO_ZABALERA,LEIHO_ALTUERA,-1,-1);
            XFreePixmap(Konexioa,pixmap);
            bukatu=1;
            }
        else
            {
            kontrolpuntuamugitzen = !kontrolpuntuamugitzen;
```

```
                if (kontrolpuntuamugitzen)
                   kontrolpuntumugitua = XagutikGertuena(&puntuax,
                                                         &puntuay);
                }
             break;
          default:
             break;
          }
       }
    XFreeGC(Konexioa,TIG1);
    XFreeGC(Konexioa,TIG2);
    XDestroyWindow(Konexioa,Leihoa);
    XCloseDisplay(Konexioa);
    }
```

Kontzeptuen Aurkibidea

β-splinea, 124, 126
β-splinea, 127

A-bufferra, 197
adierazpenaren animazioa, 322
adierazpidea, 33
  espazioaren partiketa, 34, 42
  parametrikoa, 33
  poligonozkoa, 33–35
alborapena, 125, 126
aldaketa
  2D, 140
  3D, 151
aldaketen kateaketa, 148
alderantzizko zinematika, 320, 323
algoritmoa
  Brasenham, 16
  Cohen-Sutherland, 163
  Cox-De Boor, 98
  erradiositatearena
    urratsez urratsekoa, 290
  gehikuntzazkoak, 13
  H-test, 234
  izpi-hedaketa, 244
    eraginkorra, 246
  lerroko tartekakoa, 202
  lerroz lerrokoa, 25, 32
  marrazketarako oinarrizkoak, 13
  poligonoa betetzekoa, 13
  Shutherland-Hodgman, 165
  spaning scanline, 202
  Z-bufferra, 196
aliasing, 12, 27
animazioa, 12, 27, 85, 319
  adierazpenarena, 320
  behe-mailakoa, 320
  estokastikoa, 321
  objektu artikulatuena, 320
  objektu bigunena, 320
  portaerarena, 320
  prozedura bidezkoa, 320, 327
  teknikak, 320
anti-aliasing, 28
  Fourierren teoria, 28
  gain-laginketa, 28
  iragazkia, 28
  laginketa estokastikoa, 28
  lausotzea, 28
  supersampling, 28
argi-iturria, 212
  anitz, 218
  kono-motakoa, 212, 219
  norabide hutsezkoa, 212
  puntuzkoa, 212
argiztatze-eredua, 211
  islada bidezkoa, 255
  koloretakoa, 221
  Phong-en eredua, 214
  Warn-en eredua, 218
argiztatzea

globala, 237
Gouraud versus Phong, 233
Gouraud-en interpolazioa, 227
parametroak, 212
Phong-en interpolazioa, 227
poligonoena, 227
zuzenekoa, 237, 238
atze-aurpegia, 55
ezabatzea, 191
aurreranzko zinematika, 320, 322

B-rep, 34
B-splinea, 80, 83, 89
B-splineen oinarria, 89
B-splineen txatala, 115
ez-uniformea, 90, 92
ezugarriak, 84
interpolazioa, 85, 90, 93
kontrol-puntua, 84
interpolazioa, 89
korapilo-balioa, 86, 87
razionala, 125
uniformea, 86
bektore normala
erpinarena, 228
bereizmena, 29, 38
txatal parametrikoena, 119
Bezier
Bezierren oinarria, 93
gainazal lotura, 112
gainazala, 107, 110
jarraitasun-baldintza, 75
kokapen-jarraitasuna, 112
kontrol-puntua, 71
kurba, 71, 74
txatala, 107, 113
bideoa, 12
biraketa
2D, 142
3D, 154
edozein ardatzekikoa, 157
x ardatzarekikoa, 156
y ardatzarekikoa, 156
z ardatzarekikoa, 155
bista-aldaketa, 173
PHIGS, 186
borne-bolumena, 267
Brasenham, 16, 20
BSP, 45, 47

CAD, 38, 39
B-splinea, 98
Bezier, 113
CIE LUV, 311
CIE xyY, 311
CIE XYZ, 307
CSG, 34, 41, 56

DDA, 15, 16
deformatzea, 134, 136
deformazioa, 323
ez-lineala, 325
forma askekoa, 326
gainazal parametrikoena, 324
poligonozko objektuena, 324
deformazioaren animazioa, 326
digitalizatzailea, 36
diseinua, 133
diskretua
izaera, 12
distortsioa, 138

ebaketa, 247
izpi/esfera artekoa, 248
izpi/kuadrika artekoa, 252
izpi/kutxa artekoa, 250
izpi/poliedro artekoa, 249

ebakiduren mugimendua, 39
egokitzea, 62, 65, 102
  txatal parametrikoena, 127
ehundura, 22
ekuazio diferentziala, 15
ekuazioa
  esplizitua, 13, 61, 104
  inplizitua, 61, 62, 104
  parametrikoa, 61, 62, 104
erabateko barne-islada, 226
erdibideko bektorea, 216
erradiositatea, 279
  argi-iturria, 280
  txatal-zatiketa, 287
  urratsez urratsekoa, 289
erreferentzi sistema, 48
  aldaketa, 173, 175
  dispositiboarena, 162
  ikuslearena, 48, 172
  kamerarena, 173
    PHIGS, 182
  mundukoa, 48
  objektuarena, 48
  PHIGS
    kalkulatzea, 186
errefrakzio-bektorea, 225
errorea, 16
  biribiltze-errorea, 17
espazioaren zatiketa, 43
eulerren metodoa, 15
euskarria, 15
ezkutuko aurpegia, 55
ezkutuko azalera, 193

FFD, 138, 326
forma-faktorea, 280
  kalkulua, 282
  kuboerdikoa, 285
  zehatza, 291
Fourierren transformatua, 28
funtzioen bektore-espazioa, 63, 68

gain-laginketa, 29
gainazalen egokitzea, 38
gardentasuna, 222
  errefrakzioduna, 224
  errefrakziorik gabea, 223
    interpolatua, 223
    iragazia, 224
gidoi-sistema, 321
Gouraud-en interpolazioa, 228

HLS, 304
Housenholder
  matrize isladatzailea, 146
HSV, 301
hutsunea, 16

ikuste-bolumena, 55, 162
ingurune-argia, 212
intentsitatearen ahuldura, 220
interpolazioa, 62, 69, 102, 104
  B-splinea, 85
iragazkia
  pisudun iragazkia, 29
irudia, 12
irudia sortzea, 34, 35, 51, 52, 55
  CSG, 40, 56
  PHIGS, 184
  poligonozko objektuena, 38, 40, 48
  txatal parametrikoena, 38, 55
irudigintza
  2D, 161
  3D, 170
islada barreiatua, 213
islada-aldaketa

2D, 146
3D, 152
islada-barreiapenaren koefizientea, 213
ispilu-islada, 214
itxura
aldaketa, 136
itzala, 227, 259
buffer bidez, 260
itzaleztatze-eredua, 211
itzaleztatze-prozesua, 55
itzulpena
txatal-saretik poligonora, 56
izpi-hedaketa, 42, 45, 56, 237
anti-aliasing, 261, 264
BSP egiturarekin, 274
eraginkorra, 243, 266
lehen talka, 271
octree egiturarekin, 274
oinarrizko algoritmoa, 238
sakonera-kontrola, 266

jarraia
izaera, 11
jarraitasun-baldintza, 39, 75, 133
jarraitasuna, 13
β-splineena, 127
B-splineena, 89
deribatuaren jarraitasuna, 95
jarraitasun eza, 12, 14, 16
kokapen-jarraitasuna, 75, 95
ukitzaileen jarraitasuna, 75

kamera, 172
parametroak, 174
PHIGS
begiratze-norabidea, 184
ikuste-bolumena, 184
parametroak, 182
proiekzio-planoa, 184
keyframing, 321
koherentzia
azaleren artekoa, 193
ertzen artekoa, 194
espazioarena, 273
irudi artekoa, 194
lerro artekoa, 25, 194
objektuen artekoa, 24, 25, 193
pixel artekoa, 194
sakonerarena, 194
tarte artekoa, 194
kolore-espazioa, 298
CIE LUV, 300
CIE xyY, 300
CIE XYZ, 298
espektrala, 298
HLS, 298
HSV, 298
RGB, 298
RGB monitorea, 298
kolore-mugaketa, 318
kolorea, 11, 295
adierazpena, 296
koma higikorreko aritmetika, 18
konboluzioa, 28
kontrol-puntua, 37, 62
kontrola
globala, 74
gutxi gora-beherakoa, 136
itxuraren kontrola, 62, 68, 71, 80, 102, 107
lokala, 87, 133
zehatza, 133
koordenatu homogeneoa, 150
korapilo-balioa, 90, 92
korapilo-bektorea, 92, 93
kurba

konikoa, 125

laginketa, 27
azalera-laginketa, 31
estokastikoa, 32
laser-aztertzailea, 36
laukiraketa, 162, 168
PHIGS, 191
lautasun-maila, 56
txatal parametrikoena, 119
leku-aldaketa
2D, 141
3D, 151
lerroko tartekako algoritmoa, 198

marra zuzena, 13, 16, 20
marrazki diskretua, 11
marrazkilari metodoa, 203
moldaerazko zatiketa, 47, 48
mugaketa, 161, 162
2D, 162
3D, 162
marrena, 163
poligonoena, 165

neurri-aldaketa
2D, 141
3D, 151
NURB, 124, 125

objektu artikulatuen animazioa, 322
objektu bigunen animazioa, 323
objektua eraikitzea, 36
ebakidura mugituz, 36, 131
txatal parametriko bidez, 38, 127
octree, 42, 43
izpi-hedaketa, 274
oinarria
Bernstein-en oinarria, 70, 75
berrekizun oinarria, 69
funtzioen oinarria, 63, 68
Lagrange, 69
Newton, 70
polinomioen oinarria, 69, 74
ordenazio-algoritmoa, 203

pam, 170
partikula-multzoen animazioa, 327
PHIGS, 172
Phong-en interpolazioa, 231
pixel-ilara, 24
pixel-matrizea, 27
pixela, 11
poligonoa
zuloa, 22
poligonoak betetzea, 22
poligonozko sarea, 35, 119
proiektatzea, 162, 173
PHIGS, 186
proiekzio zeiharra, 191
proiekzio-planoa, 171
leihatila, 173

quadtree, 42

Ray-casting, 207
RGB, 300

seinale-teoria, 28
simulazioa, 28
solidoen geometria eraikitzailea, 34, 40
spaning scanline, 198
spline bidezko animazioa, 322
splinea, 77, 80
kontrol-puntua, 77
lotura-puntua, 77
oinarrizko splinea, 80, 82, 86

tentsioa, 125, 126
teorema
    Jordanen teorema, 22
txatal-sarea, 37
txatal-zatiketa, 119, 121, 122
txatala, 37

ViSC, 296
voxela, 42, 276

Z-bufferra, 32, 195
    gardentasuna, 197
zarata, 32
zenbaki osoko aritmetika, 20, 26
zenbakizko metodoa, 15
zirkuitu integratua, 13
zirkunferentzia, 14
zoom, 170