Ikuspena

<a.soroa@si.ehu.es>

EHU

Ikuspena

- Lerro eta gainazal-zatien ikuspena

Abiadura algoritmoak

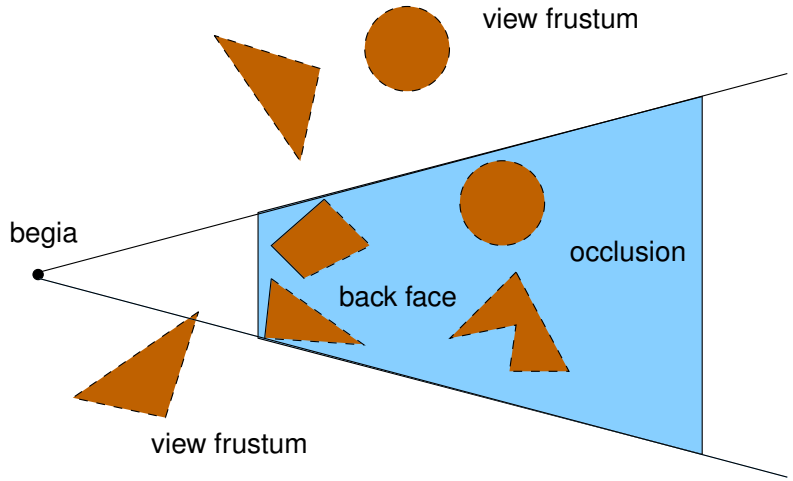
- CPUak gero eta azkarragoak dira, baina inoiz ez azkarregiak
- Beti dago informazio gehiago sartzerik
  - Modelo aberatsagoak: ehundaka mila poligono
  - Animazio leuna: 60-85 fps (monitorearen frekuentzia)
  - Erresoluzio handiagoak
- Ezin zaio dena OpenGL-ri eman
- Konklusioa: abiadura algoritmoak behar dira

Ikuspena. Definizioak.

- "Exact Visible Set" *EVS*: une batean ikusiko diren primitibak
  - Oso garestiak identifikatzen $O(n^9)$
- "Potentially Visible Set" *PVS*: une batean ikus daitezkeen primitibak
  - *EVS*ren aurreikuspena
    - $EVS \subset PVS \rightarrow$ aurreikuspen kontserbakorra
    - bestela, $\exists p \in \mathrm{EVS} \wedge p \notin \mathrm{PVS} \rightarrow$ minimizatu p
- *PVS* lorrtzeak eraginkorra izan behar du

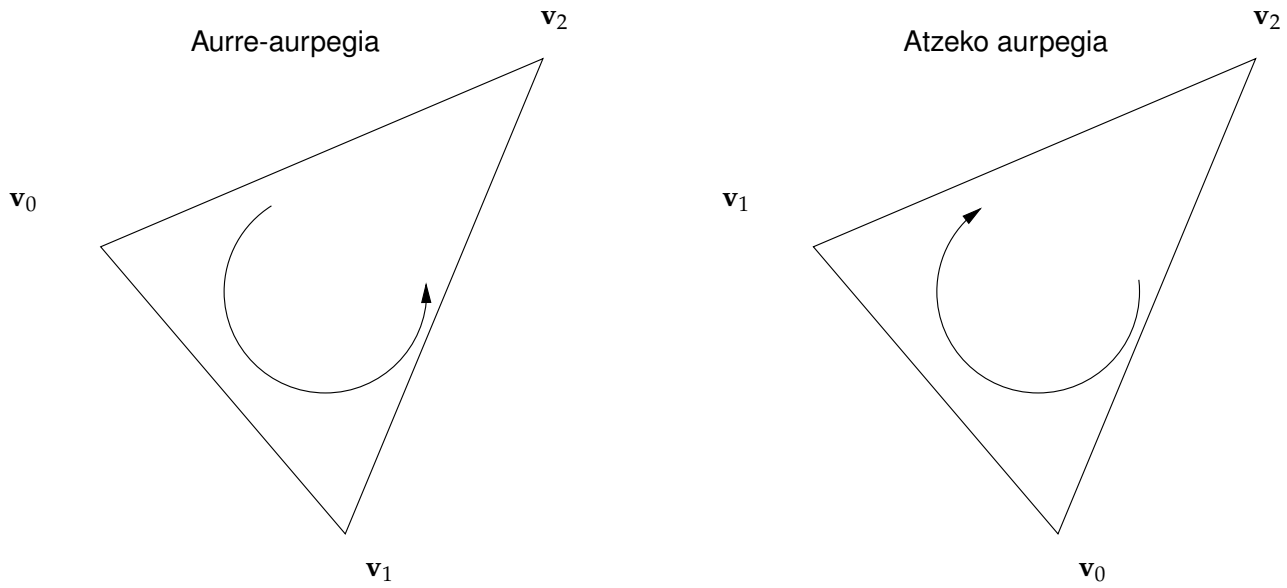
Culling

- Eszena osoa dugu memorian
- Une batean eszenaren zati bat besterik ez da ikusiko
- *Culling*
  - *Frustum culling*: kamarak ikusten dituen objektuak
  - *Backface culling*: ikusiko ez diren poligonoak
  - *Occlusion culling*: ezkutuan geratzen diren objektuak



Backface culling

- Eszena baten poligono-kopuru erdia baino ez da ikusten
- Objektuen atzeko aurpegiak ezkutatuta daude
- Jakin behar da zer den atzeko aurpegi bat
- Adib: Opengl-n

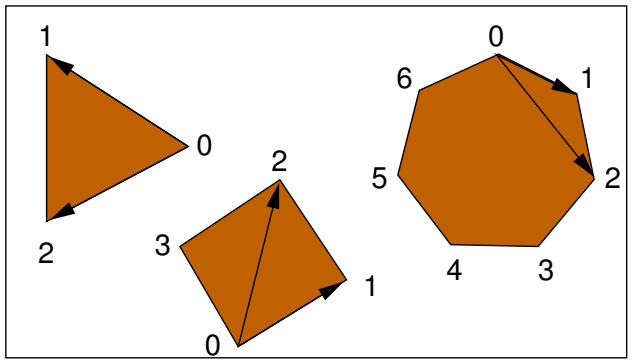


Backface culling

- Pantaila-espazioan
  - poligonoa: $(v_0, \ldots, v_{n-1})$
  - normala: $\mathbf{n} = (v_1 - v_0) \times (v_2 - v_0)$

$$\begin{cases} \mathbf{n} = (0, 0, -a) & \text{atzeko aurpegia} \\ \mathbf{n} = (0, 0, a) & \text{aurre-aurpegia} \end{cases}$$

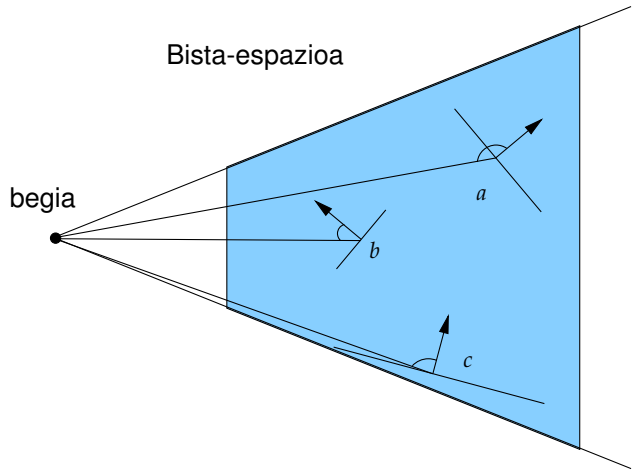
Pantailaren espazioa



Backface culling

- Bista-espazioan
  - ikuslearen posizioa $\mathbf{e}$ eta poligonoko puntu bat $\mathbf{v}_0$
  - poligonoaren normala: $\mathbf{n}_p$ (ez du normalizatua izan behar)
  - $\mathbf{ev}_0$: $\mathbf{e}$tik $\mathbf{v}_0$ra doan bektorea = $\mathbf{v}_0 - \mathbf{e}$
  - $a = \mathbf{ev}_0 \cdot \mathbf{n}_p = \frac{\cos(\alpha)}{\|\mathbf{ev}_0\|\|\mathbf{n}_p\|}$

$$\begin{cases} a < 0 & \text{atzeko aurpegia } (\alpha > \frac{\pi}{2}) \\ \text{bestela} & \text{aurre-aurpegia } (\alpha \leq \frac{\pi}{2}) \end{cases}$$



Backface culling

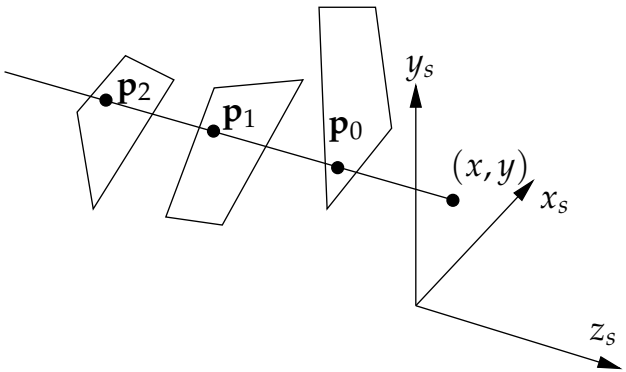
- Normalean pantaila-espazioan gauzatzen da
  - Geometria-fasea ez da aldatzen
    - Objektuen poligono guztiak doaz geometria-fasera
  - Diskretizazio-fasea azkarragoa izango da
  - Kontuz isla-aldaketekin: lateralitatearen inbertsioa
- OpenGL-k hala egiten du

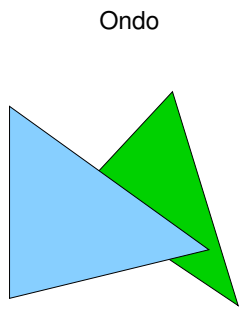
Backface culling eta OpenGL

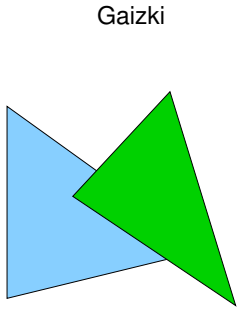
- Aktibatzeko: `glEnable(GL_CULL_FACE)`
- Aurreko aurpegia/ atzeko aurpegiak zeintzuk diren alda daiteke:`glCullFace()`
  - `GL_FRONT`: aurre-aurpegiak aurrean
  - `GL_BACK`: atzeko aurpegiak aurrean. Baliagarriak objektuaren barruan bagaude.
- Poligonoko erpinen ordenak garrantzia du
  - Eskuin eskuko erregela
  - baina alda daiteke: `glFrontFace()`
    - `GL_CCW`: counterclockwise (lehenetsia)
    - `GL_CW`: clockwise
  - Berriro ere: kontuz isla-aldaketarekin

Z-buffer

- Pantailako pixel batean hainbat objektu proiektatzen da
- Perspektiba gauzatu ondoren, objektu gertuen z osagaia urrunekoena baino txikiagoa da (OpenGL)
- Irudian $\mathbf{p}_0$ da objektu gertuena







Z-buffer

- *Z-buffer*-ak arazoa konpontzen lagunden du
  - Pixel bakoitzaren sakonera (z) gorde
  - Triangelu bat marrazterakoan, kalkulatu barruko pixel bakoitzeko sakonera
  - Alderatu pixelaren z osagaia *Z-buffer*-ek duenarekin
  - Triangeluarena txikiagoa bada, ordezkatu pixela (eta *Z-buffer*-a)
  - Bestela, ez egin ezer

  Edozein ordenetan marraz daitezke primitibak

Z-buffer. Inplementazioa OpenGL-en.

- *Z-buffer*-a hasieratu (GLUT)
  - `glutInitDisplayMode ( ...| GLUT_DEPTH )`
- *Z-buffer* test-a aktibatu:
  - `glEnable(GL_DEPTH_TEST)`
- Agindu osagarriak:
  - `glClear( ...| GL_DEPTH_BUFFER_BIT)`: *Z-buffer*-a garbitu
  - `glClearDepth([0,1])`: zein baliorekin garbituko den
  - `glDepthFunc(GL_LESS)`: Konparazio-funtzioa. Balio posibleak:
    - `GL_NEVER, GL_LESS, GL_EQUAL, GL_NOTEQUAL, GL_GEQUAL, GL_ALWAYS`
  - `glDepthRange(near[0,1], far[0,1])`: dispositiboaren koordenatu normalizatutik [-1, +1] mapaketa lineala [0,1]-era
  - `glDepthMask(0 disabled, ≠ 0 enabled)`: *Z-buffer*-ean idatzi edo ez

Frustum culling

- Objektu geometrikoak borne-bolumen(BB) batzuez estaliak daude.
- BBa ikuspenaren piramide-enborretik kanpo badago, objektu guztia kanpoan egongo da.
  - Ez da ikusiko.
- Benetan eraginkorra izateko, objektuen hierarkiak erabili behar dira.

Frustum culling

- Borne-bolumen tipikoak
  - AABB: ardatzekiko lerrokatutako borne-kutxa
    - Bi puntu: $\mathbf{a}^{\min}$ eta $\mathbf{a}^{\max}$, non $\mathbf{a}_i^{\min} \leq \mathbf{a}_i^{\max}, \forall i \in \{x, y, z\}$
  - OBB: Edozein orientazioko borne-kurtxa
    - $\mathbf{b}^c$ puntua, eta $\mathbf{b}^u, \mathbf{b}^v, \mathbf{b}^w$ hiru norabide-bektore.
  - Esferak
    - $\mathbf{c}$ puntua, eta r erradioa
  - k-DOP, etc.

Frustum culling

- Beraz, jakin behar da objektu baten borne-bolumena (BB) kamarak definitutako piramide-enborrean barruan dagoenetz.
- Arazoa: zein espaziotan gauzatu *cullinga*?
  - Objektuen eredu lokalan: kamarako piramide-enborra objektu bakoitzaren eredura bihurtu
  - Munduko ereduan: objektuaren BBa eta kamarako piramide-enborra mundura bihurtu
  - Kamara ereduan: objektuaren BBa kamararen eredura eraman.
- Guk munduko ereduan gauzatuko dugu *cullinga*.