

Geometric Optimization

COMP 3003 Autumn 2005

Geometric Optimization

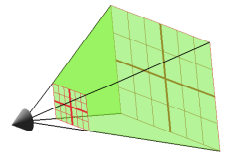
- We use *geometry* to describe our world
- Our world is *huge*
 - so we can't render *all* of it
 - we only render *part* of it
 - or we render it less *accurately*
- This is *geometric* optimization

Geometric Options

- We can:
 - *discard* geometry that's not visible
 - *clipping* it outside the field of view
 - *culling* objects behind other objects
 - *simplify* geometry that is unimportant
 - *e.g.* textures

Clipping to View Frustum

- Defined by 6 planes
- Geometry *outside* view frustum is discarded
- Geometry *intersecting* frustum is modified
- Geometry *inside* frustum is kept



Additional Clipping

- Clipping also performed on fragments
 - *scissor* test, *stencil* test, &c.
- OpenGL allows 6 extra *clipping planes*
 - `glClipPlane()` specifies plane
 - Implicit eqn in homogeneous coords
- *Cap* clipped objects using stencil buffer

Culling

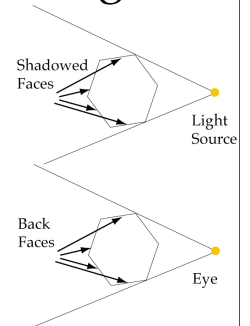
- *Culling* discards known invisible objects
- Includes:
 - *visibility culling*: geometric computation
 - *backface culling*: in geometric processing
 - *depth buffering*: discards invisible pixels

Visibility Culling

- Visibility culling is done by programmer
- *Portals* (division of scene into *rooms*)
- *Binary Space Partition (BSP) Trees*: division of scene along face planes
- *Visibility Region* computation
- Usually computed for static objects only

Backface Culling

- Similar to *shadowed faces*
- Normals *away* from eye
- Discarded by OpenGL during geometric processing



Simplification

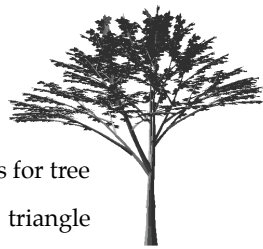
- Even fast video cards are limited
- What do we do if they're too slow?
 - Make them do *less*
 - i.e. *simplify* our models of objects
 - especially complex background objects

The Tree Again



A Geometric Tree

- 45,000 triangles
- 512 x 512 image
 - about 90,000 pixels for tree
 - or about 2 pixels / triangle
 - and this is nearby with few leaves!



A Distant Tree

- Twice as far
- Foreshortened, so only 256 x 2
 - about 22,500 pixels used
 - 0.5 pixels / triangle
 - but each triangle uses c. 100B
 - so roughly 200B information / pixel
 - but only 3B information shown



Level of Detail

- Nearby objects should be *high detail*
- Distant objects can be *low detail*
- So have *several* models for the object
 - high detail up close
 - low detail far away

Level of Detail

- Mostly a geometric optimization
- Faraway objects are smaller (*foreshortening*)
 - Don't need as much geometric detail
 - So use a simpler cheaper model
 - Or use textures instead of geometry

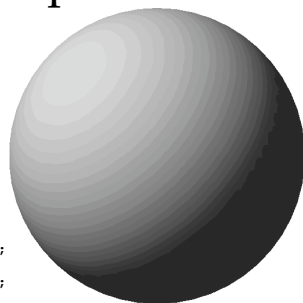
Choosing a Level

- Level of detail depends on *foreshortening*
 - i.e. on *distance*
- We need distance from camera to object
 - OpenGL doesn't do this
 - CPU (programmer) has to do it

Object Distance

- To use level of detail, keep track of:
 - camera location
 - object locations
- Before rendering object
 - compute distance
 - choose appropriate model

LoD Sphere

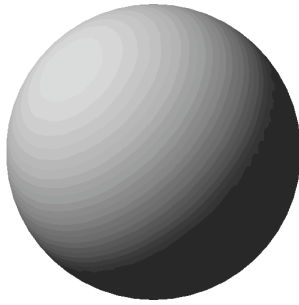


```
if (scale > 25.0)
  glutSolidSphere(1.0, 6, 6);
else if (scale > 5.0)
  glutSolidSphere(1.0, 8, 8);
else
  glutSolidSphere(1.0, 50, 50);
```

LoD Problems

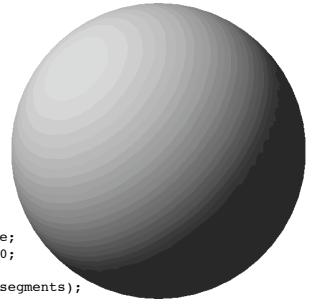
- Choosing the *level* can be tricky
 - not always easy to predict
 - especially with smooth objects
 - and objects in *foreground*

Bad LoD



Smooth LoD

- Solution: *compute* the level of detail



```
int segments = (int) 300 / scale;  
if (segments > 50) segments = 50;  
if (segments < 6) segments = 6;  
glutSolidSphere(1.0, segments, segments);
```

Textured Objects

- Remember why we use textures
 - Cheaper than geometry
 - Can also simplify objects this way
- Close objects in geometric detail
- Distant objects using textures

Mipmaps

- *Mipmaps* are level-of-detail textures
 - Supported by OpenGL
 - Smoother interpolation of textures
 - at extreme magnification
 - or extreme minification

Substituting Textures

- Trees are very complex
- Often need a lot of them
- 45,000 triangles = 13.5MB
- Use textures instead
 - *billboards*, *skyboxes*, *scenery boxes*, &c., &c.



Billboards

- Two textured quads
- Approximates a tree
- Uses alpha blending
- Texture is 1MB
- Much cheaper

