Real-time 3D Graphics
Doug A. Bowman
Outline

- (Very) brief review of 3D image synthesis
- Real-time 3D graphics for VEs
Rendering 3D scenes

- Most 3D graphics is based on scenes composed of objects built of polygons.
- Goal: from scene specification and viewing specification, produce a 2D image of the scene with proper 3D perspective by determining the color for each pixel on the display.

3D teapot animation…
Overview of 3D image synthesis (rendering)

- **3D model**
  - Includes scene definition, camera properties, lighting characteristics, etc.
  - Process of mapping the desired 3D primitives into a 2D space

- **2D projection**
  - 2D versions of 3D primitives
  - Process of changing primitive definitions into pixel locations/colors

- **Pixel values**
  - Color value for each pixel on the screen, stored in the frame buffer
  - Process of taking information from frame buffer and producing physical representation on a display device

- **Displayed image**

(C) 2008 Doug Bowman, Virginia Tech
3D Viewing Example

Object model
Coordinate systems

- Right- vs. left-handed
- World vs. object (object trees, scenegraphs)
- World vs. camera
- Screen/display

- Transformations (mappings) between CSs
Graphics pipeline

Local space: Object definition

World space: Compose scene
             Define view parameters
             Perform culling

View space: Clip to 3D view volume

3D screen space: Hidden surface removal
                Shading

Display space: Rasterization

Modeling transformation

Viewing transformation

Perspective projection

3D screen space

Display space
Modeling transformation

- For each vertex: transform object to world coords.
- Each M represents a transformation matrix.
- Where is chair2 relative to room?
Viewing transformation

- Transform world to camera coords.
- Where are the objects with respect to the camera?

View frustum

Back clipping plane

Front clipping plane

Angle of view, A

Camera

$2h$

D F

Viewing transformation animation…
Clipping

- Process of removing polygons and parts of polygons that fall outside the view frustum
- Basically, comparing each vertex with plane equations for the six view frustum sides
Projections

- Projection: mapping 3D to 2D space
- Center of projection = camera = eye
- Projection plane = viewplane: 2D plane onto which objects are projected
Mathematics of a one-point projection

\[ x_{\text{projected}} = \frac{dx}{z} = \frac{x}{(z/d)} \]

\[ y_{\text{projected}} = \frac{dy}{z} = \frac{y}{(z/d)} \]
Graphics pipeline

Local space
- Object definition

World space
- Compose scene
- Define view parameters
- Perform culling

View space
- Clip to 3D view volume

3D screen space
- Hidden surface removal
- Shading

Display space
- Rasterization

Modeling transformation
- Viewing transformation
- Perspective projection

Display space
- Display space

(C) 2008 Doug Bowman, Virginia Tech
Hidden-surface elimination

- Problem: we only need to display polygons that are visible
- Example approaches:
  - z-buffer (hardware)
  - BSP tree
  - Spatial subdivision
We can increase realism by adding shading/lighting, but more complex approaches take longer to calculate.
Rasterization

- Once vertices have been projected, we only need to “fill in” the primitives between the vertices
- Rasterization (sometimes called scan conversion) is the process of determining which pixels are part of a primitive and “turning on” those pixels.

Polygon fill animation…
Graphics pipeline

- Local space: Object definition
- World space: Compose scene, Define view parameters, Perform culling
- View space: Clip to 3D view volume
- 3D screen space: Hidden surface removal, Shading
- Display space: Rasterization

Spaces:
- Local space
- World space
- View space
- 3D screen space
- Display space

Transformations:
- Modeling transformation
- Viewing transformation
- Perspective projection
- Rasterization

(C) 2008 Doug Bowman, Virginia Tech
What is real time?

• “Fast enough” that the user can
  ○ Perceive smooth motion
  ○ Interact with the environment

• Often stated in frames per second (fps) – number of views of the scene rendered in one second

• A frame rate of approx. 10-30 fps is the minimum for “real-time”
Speed vs. Quality tradeoff

- More complex models ⇒ slower rendering
- We must manage this tradeoff to achieve maximum quality with an acceptable frame rate
Techniques for real-time 3D graphics

- Polygonal simplification
- View culling
- Occlusion culling
- Level of detail management
- Texture substitution
- Lightmaps
Polygonal simplification

- Decrease the number of polygons without sacrificing visual quality (meet polygon budget)

30944 triangles  2502 triangles  621 triangles  251 triangles
Level of detail management

- LOD: change the complexity of the model/image dynamically to maintain real-time performance
- Ex: use simplified models when objects are at a great distance
Level of detail - another example

- Match characteristics of human eye
- Use high detail in fovea, low detail in periphery
View culling

- Not the same as clipping!

- Easy: remove all objects that are behind the plane of the camera
- Harder: remove all objects outside the view frustum
Occlusion culling

- Hidden-surface elimination techniques: still render all polygons (e.g. z-buffer)
- Occlusion culling: remove polygons from consideration before rendering
Texture substitution (*imposters*)

- Textures can cause objects to appear much more detailed than they actually are.
- Problem: when the user is close.
Lightmaps

- Pre-calculate lighting for a scene, then texture lighting onto objects
Lightmapped VEs
Other graphics issues for VEs

- Stereo rendering
- Off-axis projections
- Inserting the user into the scenegraph
- Parallel / distributed rendering