Model Optimization
Outline

• Motivation
• Mesh Simplification
• Level of Detail
Motivation

Common Screen Dimension: 1920x1080 (1080p)
Motivation

Common Screen Dimension: 1920x1080 (1080p)
Total Number of Pixels: 2,073,600
Motivation

Common Screen Dimension: 1920x1080 (1080p)
Total Number of Pixels: 2,073,600

• So a 1080p screen can display contributions from at most ~2m triangles (1px per face)
Motivation

Common Screen Dimension: 1920x1080 (1080p)
Total Number of Pixels: 2,073,600

• So a 1080p screen can display contributions from at most ~2m triangles (1px per face).
• In reality this number will be much lower.
  – We rarely display all triangles at once.
  – Visible triangles often contribute to more than one pixel.
  – Images often have “empty” pixels where no triangles are contributing.
So what happens when we have more triangles than pixels on which to display them?
Motivation

So what happens when we have more triangles than pixels on which to display them?

Choices have to be made as to which triangles get to contribute to the final pixel.
Motivation

But that’s fine, we already know how to do this. So what’s the problem?
But that’s fine, we already know how to do this. So what’s the problem?

If we can only rasterize a fraction of the visible triangles in a mesh but still have to load the entire mesh (for example when we are zoomed way out), then we’re doing a lot of work for nothing.
Motivation

Wouldn’t it be better to use just enough faces to get all the necessary details?
Motivation

Wouldn’t it be better to use just enough faces to get all the necessary details?

Calculating exactly which triangles we need isn’t realistic, but we can approximate it.
Outline

• Motivation
• Mesh Simplification
• Level of Detail
Mesh Simplification

Techniques for reducing the number of triangles in a mesh while maintaining the general shape.

- Resampling
- Edge Collapse
- Re-triangulation
- Adaptive Mesh Subdivision
Resampling

• Construct an $i*j*k$ size grid and, for each grid cell, store the signed distance from the cell to the nearest surface.

• March through each cell and use interpolation to determine if the cell contains triangles and, if so, what edges of the cell they are connected to.
Resampling

- **Pros:**
  - Fast for reasonable grid sizes
  - Easy to implement

- **Cons:**
  - Can miss important details

Note: there are other techniques referred to as resampling:
- Visualization method that is very similar to this
- Alternative simplification method that uses a point cloud
Resampling

- **Pros:**
  - Fast for reasonable grid sizes
  - Easy to implement

- **Cons:**
  - Can miss important details

- **Note:** there are other techniques referred to as resampling.
  - Visualization method that is very similar to this
  - Alternative simplification method that uses a point cloud
Edge Collapse

- Select an edge
- Merge its two vertices into a single new vertex
Edge Collapse

• Pros:
  – Conceptually simple
  – Maintains mesh quality well

• Cons:
  – Programmatically hard
Edge Collapse

- How do we select the right edges?
- How do we deal with creating degenerate triangles?
- How do we know what vertices go with which edges/triangles?
Edge Collapse

Answer:

• There isn’t an easy one
• Must consider possible cases and deal with them
• You also need the right data structure
  – Our old triangle class isn’t set up for doing edge collapse efficiently
Re-Triangulation (Decimation)

- Pick a vertex to remove
- Delete the vertex and all adjacent triangles
- Fill the new gap by generating fresh triangles
Re-Triangulation (Decimation)

• Pros:
  – Don’t have to deal with degenerate triangles
  – Maintains mesh quality well

• Cons:
  – Shares edge collapse’s issues in needing to know which vertices to select and which triangles are adjacent to a vertex
  – Hard to implement (you need some ugly math to fully describe what’s going on)
Adaptive Mesh Subdivision

- Some areas of a mesh can be more complex than others.
Adaptive Mesh Subdivision

- Some areas of a mesh can be more complex than others.
- So why not keep the triangles in places with lots of detail and remove triangles from relatively simple areas?
Adaptive Mesh Subdivision

- Some areas of a mesh can be more complex than others.
- So why not keep the triangles in places with lots of detail and remove triangles from relatively simple areas?
- This reduces our geometry count without noticeably impacting mesh quality.
Adaptive Mesh Subdivision

This technique generally works in reverse from the others in that it starts from a simple mesh and builds up complexity where needed.
Mesh Simplification

Note that there are numerous other methods for mesh simplification beyond just these. We could devote an entire class to all the ways to do this.

The important detail they all share is that they allow us to get rid of unnecessary geometry.
Outline

• Motivation
• Mesh Simplification
• Level of Detail
So we can reduce our mesh such that it contains just enough detail to render all the pixels in a static image.
So we can reduce our mesh such that it contains just enough detail to render all the pixels in a static image.

What about if we’re moving the camera around dynamically?
Level of Detail

So we can reduce our mesh such that it contains just enough detail to render all the pixels in a static image.

What about if we’re moving the camera around dynamically?

This thought serves as the motivation for something known as level of detail.
The Idea?

• Pre-generate meshes of various quality levels using one of the aforementioned methods
Level of Detail

The Idea?

• Pre-generate meshes of various quality levels using one of the aforementioned methods

• At render time use the camera’s distance to the object (or another appropriate heuristic) to pick the mesh with just enough detail to maintain image quality without “wasting” a ton of geometry.
Level of Detail

By passing the smallest level of detail needed, we can decrease the amount of geometry the graphics card has to obtain and use without causing a reduction in image quality.
Problem: With a discrete number of meshes to choose between, moving towards or away from the object can cause the detail level to "pop"
Level of Detail

Problem: With a discrete number of meshes to choose between, moving towards or away from the object can cause the detail level to “pop”

Common issue seen in many computer games
Level of Detail

We can fix this issue by using continuous level of detail algorithms to refine the mesh on the fly.
We can fix this issue by using continuous level of detail algorithms to refine the mesh on the fly.

These are more complex than we’re going to get for this lecture, but they do exist if needed.
Final Thoughts

Model optimization is a complex topic with numerous other things to consider, some of which we’ve gone over in class, some of which we will go over later, and others that you will have to learn on your own.

In the end it’s all a matter of balancing quality against processing time while looking for ways to avoid calculating anything that doesn’t contribute to the final image.
Sources

• Bunny and dragon models courtesy of the Stanford Computer Graphics Laboratory scanning repository (graphics.stanford.edu/data/3Dscanrep/)

• Resampling images (slides 13-15) courtesy of Hank Childs’ scientific visualization lectures (www.cs.uoregon.edu/Classes/15F/cis410visualization/)

• Models generated with VisIt visualization tool (wci.llnl.gov/simulation/computer-codes/visit) and MeshLab mesh processing tool (meshlab.sourceforge.net/)