COMP250: Computer Graphics

Jérôme Waldispühl
School of Computer Science
McGill University
Slides assembled from M. Blanchette (McGill), T. Thorne (U. of Edinburgh)
Computer Graphics Rendering

• World is represented by a set of 3D objects, with colors, reflectivity, transparency, etc.
  – Primitive objects: Polygons, spheres, cones
  – Complex objects: Mesh of triangles

• Goal: Produce a realistic 2D picture of the world
Computer Graphics is about animation (films)
Games are very important in Computer Graphics
Medical Imaging is another driving force
Computer Aided Design too
Scientific Visualisation

Slide by T. Thorne (U. of Edinburgh)
Graphics / Rendering Pipeline

There are three stages:
- Application Stage
- Geometry Stage
- Rasterization Stage

Slide by T. Thorne (U. of Edinburgh)
Batch Computer Graphics

- Today, still use non-interactive *batch mode* for final production-quality video and film (special effects – FX). Rendering a single frame of Monsters University (a 24 fps movie) averaged 29 hours on a 24,000-core render farm!
Application stage
(for interactive computer graphics)

Entirely done in software by the CPU

Read Data
the world geometry database,
User’s input by mice, trackballs, trackers, or sensing gloves

In response to the user’s input, the application stage change the view or scene
Geometry Pipeline

- Loaded 3D Models
- Model Transformation
- Transformation: viewing
- Hidden Surface Elimination
- Shading: reflection and lighting

Slide by T. Thorne (U. of Edinburgh)
Preparing Shape Models

Designed by polygons, parametric curves/surfaces, implicit surfaces and etc.

Defined in its own coordinate system

Slide by T. Thorne (U. of Edinburgh)
Model Transformation

Objects put into the scene by applying translation, scaling and rotation

Linear transformation called homogeneous transformation is used

The location of all the vertices are updated by this transformation

Slide by T. Thorne (U. of Edinburgh)
The input

2D image

3D World

eye (x,y,z)
Ray-tracing
Ray-tracing algorithm

Input: - world: set of 3D objects
  - (x,y,z) position of the eye
  - Position of the 2D screen
Output: Image: array of colors of size nPixels by mPixels
For i = 1...nPixels
  For j = 1...mPixels
    r = ray( eye -> pixel(i,j) )
    object = getClosestIntersection(r, world)
    if (object! = null) then
      image[l,j] = object.getColor();
Recursive Ray-tracing
Finding intersections

- Suppose your world consists of Millions of objects
- How can you calculate closest intersection quickly?
  - Computing intersection between ray and each object is much too slow
- Idea: Store your objects in a data structure that allows you to quickly discard objects that can’t have intersection
Quad trees

For a 2D-world, subdivide the world into four quadrants.

Keep subdividing as long there is more than one object per square.

For 3D-world, subdivide world into eight octants.
Quad trees

Subdivision is represented as a tree:
Root = complete world
Children = four quadrants
Fast ray intersection problem

To quickly find intersection between ray and world:

Find which main quadrant is intersected

Find which of its subquadrant is intersected

... Keep going down the tree until a leaf is found

If leaf contains an object, then test intersection

Continue until intersection is found

Slide by M. Blanchette (McGill)
Hidden Surface Removal

Objects occluded by other objects must not be drawn
Shading

Now we need to decide the colour of each pixels taking into account the object’s colour, lighting condition and the camera position.
Shading : Constant Shading - Ambient

Objects colours by its own colour
Shading – Flat Shading

Objects coloured based on its own colour and the lighting condition

One colour for one face
Gouraud shading, no specular highlights

Lighting calculation per vertex
Specular highlights added

Light perfectly reflected in a mirror-like way
Phong shading
Next, the Imaging Pipeline

1. Geometry
2. Rasterization and Sampling
3. Texture Mapping
4. Image Composition
5. Intensity and Colour Quantization
6. Framebuffer/Display
Rasterization

Converts the vertex information output by the geometry pipeline into pixel information needed by the video display.

Aliasing: distortion artifacts produced when representing a high-resolution signal at a lower resolution.

Anti-aliasing: technique to remove aliasing.
Anti-aliasing

Aliased polygons (jagged edges)

Anti-aliased polygons

Slide by T. Thorne (U. of Edinburgh)
How is *anti-aliasing* done? Each pixel is subdivided (sub-sampled) in n regions, and each sub-pixel has a color;
Compute the average color value

![Diagram showing anti-aliasing process with a single multisampled pixel (16 samples). Composite Color = 5/16 Blue + 11/16 Green.](image)
Texture mapping
Other topics:
Reflections, shadows & Bump mapping