COMP30019 Graphics and Interaction
Rendering pipeline & object modelling

Adrian Pearce

Department of Computing and Information Systems
University of Melbourne

The University of Melbourne
Lecture outline

- Introduction to Modelling
- Polygonal geometry
- The rendering pipeline
- High Performance Visible Surface Detection
How are objects modelled in computer graphics?

Aim: understanding polygonal geometry and the basics of the rendering pipeline.

Reading:

▶ Foley Section 9.1.1 Representing polygon meshes.
▶ Foley Sections 14.4 Shadows, 14.9 The rendering pipeline and 14.3 Surface detail.

Additional Reading:

▶ Section 23.3 Level of detail in Computer graphics and virtual environments: from realism to real-time by Mel Slater, Anthony Steed and Yiorgos Crysanthou, Addison Wesley 2002 ISBN: 0201624206.
Introduction to modelling

The first problem in modelling geometric objects and their motion is to formulate a suitable mathematical model (does it have a solution? is the solution unique?). A *model* based on general theories and laws of particular discipline (e.g. luminosity model of a 3D surface or physical model of a soccer ball).

- You may need to *simplify* the model and approximate to ensure tractability (use polyhedra model of soccer ball instead of parametric surface.)
- There will be a balance between what to include for completeness, and what to remove for tractability (don’t model the effect of wind on the soccer ball).
In general, a polygon is any plane figure bounded by straight line segments and but can comprise these forms:

- polygonal arcs (polylines)
- polygonal boundaries (closed polylines)
- and filled polygons

Polygons are very useful, both in themselves and as building blocks for approximating arbitrary curved arcs and regions.
Representation

- Either as a set of line segments,
  ![Diagram of a polygon]

- or an ordered sequence of vertices using absolute or relative coordinates,
  ![Diagram of a polygon with vertices connected in a sequence]

- Walking order convention often applies, e.g. anti-clockwise for outer and clockwise for inner
Polygon mesh: vertex list

\[ V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4)) \]

\[ P_1 = (1, 2, 4) \]

\[ P_2 = (4, 2, 3) \]
Polygon mesh: edge list

Polygons represented as line segments (edges) makes polygon clipping and scan-line filling operations easier (we will see later).
Introduction to Modelling

Polygonal geometry

The rendering pipeline

High Performance Visible Surface Detection

Polygon types

- Convex
- Concave
- Non-simple
- Multiple-boundary
- Star

Adrian Pearce
University of Melbourne

COMP30019 Graphics and Interaction
Rendering pipeline & object modelling
Polygon Properties

- In a convex polygon no internal angle is greater than 180 degrees
- In a concave polygon there are internal angles that can be greater than 180 degrees
- Concave polygons can be represented as a conjunction of convex polygons, sometimes desired as convex polygons have certain properties that simplify some geometric operations and tesselations.
- The multiple boundary polygons would need some classification of region (like Tasmania connected to the mainland of Australia).
The Rendering pipeline (Simplified)

(Foley Figure 14.41)
Double buffering

The problem: animation and motion can cause the display to flicker.

The solution: double buffering can be used to reduce the effect of flickering, by sending all drawing commands to an off-screen buffer, then swapping buffers before redrawing.

In addition, only those pixels recently drawn are selectively erased at each step in the off-screen buffer.
Rendering pipeline for z-buffer (revisited)

(Foley Figure 14.41)
Rendering pipeline - the steps

See page 521 of Foley:

- **db traversal & modelling transformation**: transform all (relatively defined) polygons or polygonal meshes to their correct location.
- **Trivial accept/reject**: Entirely outside & back face culling
- **Lighting**: calculate intensity (for vertices only: need to do in 3D before perspective projection)
- **Viewing transformation**: perspective transformation (e.g. perspective foreshortening).
- **Clipping**: clip to viewport (involves creation of new vertices on border).
- **Map to viewport**: Change of coordinate systems (Divide by W: based on homogeneous coordinate system).
- **Rasterisation**: scan-line drawing polygons (including z-buffering & shading interpolation)
Level of detail

Varying quality with distance - programmer provides several different versions of an object for viewing at different distances.
Example from Stanford University Computer Graphics Lab

Despite the difference in the number of used to model the figures (on the left), at a distance (on the right) they appear very similar (Figure from Slater text).
Vertex split method

(Figure from Slater text)
Example of levels of detail
Texture and bump mapping

- A *texture map* is used to scale one or more of the surfaces material properties, such as diffuse colour (RGB) components.
- A *bump map* is an array of displacements, each which perturbs the surface normal before it is used in the illumination model.
Summary

- Polygons are one of the most widely used models in computer graphics and are useful for representing both two-dimensional shapes and three-dimensional objects.
- The Rendering pipeline is the basis of OpenGL.
- The rendering pipeline facilitates the systematic modelling, transformation and rendering of polygons.
- Various techniques exist for achieving high performance, including level-of-detail, texture and bump mapping.