Introduction to Graphics

Lecture 8:

Polygon Rendering

Lecture Overview

- Polygon Renders
- Data Flow
- Scene Layout
- Terrain
- Texture
- Shading

Polygon Renders

Most real time animation systems today are based on polygon rendering,

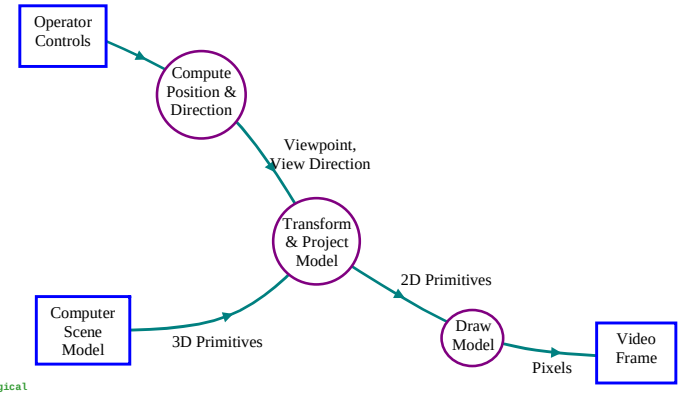
i.e. they use only 3D planar polygons to build a scene.

Examples:

Flight Simulation

Games

Data flow in a computer Game

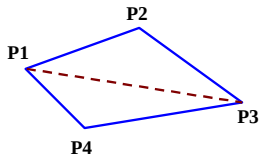


Scene Layout

Most scenes are built from planar polygons

Quadrilaterals are a common choice,

Triangles are safer!

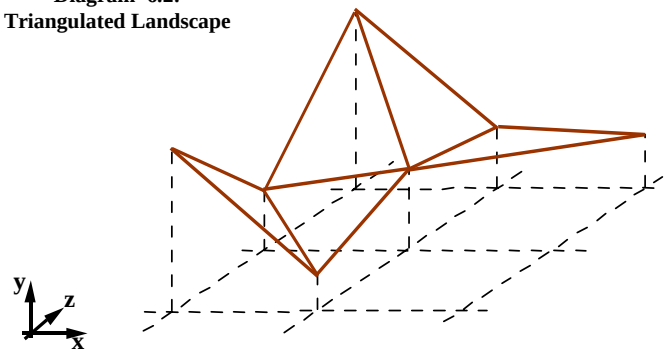


Four points are not necessarily coplanar

Triangles always are.

Terrain

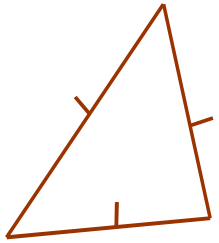
Diagram 6.2:
Triangulated Landscape



Making more complex terrain

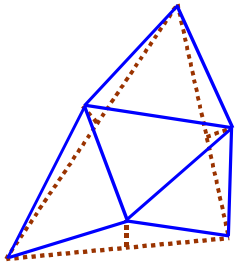
- Defining a complex terrain is time consuming
- One trick is to define a simple terrain and make it more complex using 'fractals'
- A simple geometric algorithm does it

For each triangle of the simple terrain



Displace each triangle mid-point

Join up the new points and the old vertices



Join up to form four new triangles

Texture

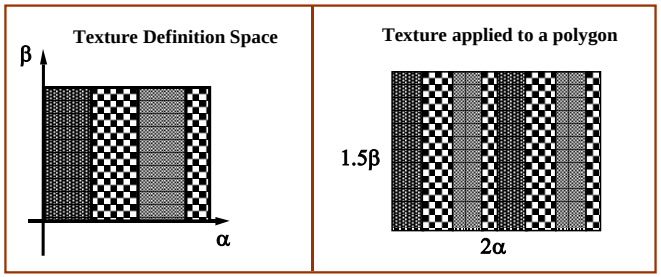
- The visual appearance of a graphics scene can be greatly enhanced by the use of texture.
- Consider a brick building, using a polygon for every brick require a huge effort in scene design.
- So why not use one polygon and draw a repeating brick pattern (texture) onto it?

Texture Definition

Textures may be defined as:

Bitmaps - Arrays containing the actual pixel values to be mapped to the polygon. The data can be derived from photographs for example.

Procedures - Suitable for repeating patterns.



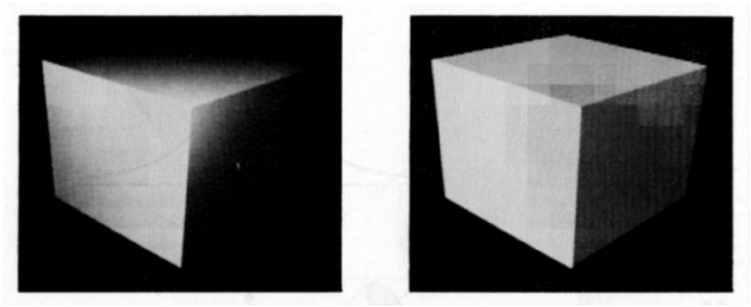
Photographs as Textures

- Photographs can be used to enhance reality with virtually no design effort.
- For a flight simulator landing at an airport the distant landscape can be presented as a photograph which forms the back clipping plane
- This is similar to the "Blue Screen" technique used in films

Shading

- Adding a shade can add considerable realism to graphics scene.
- The shade at each point on a surface is dependent on the positions of the light sources and the position of the viewer.
- The basic law for light rays hiting a physical object is called Lambert's cosine law.

Shading from different light sources

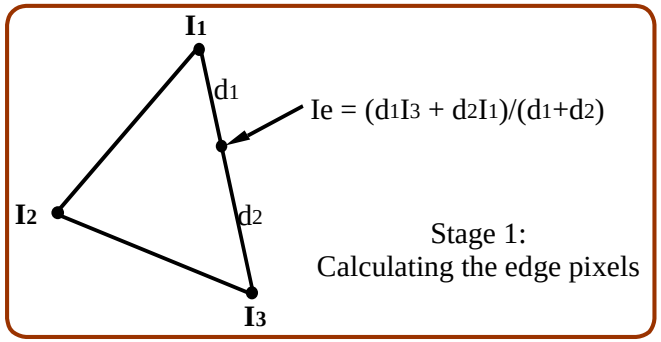


Gouraud Shading

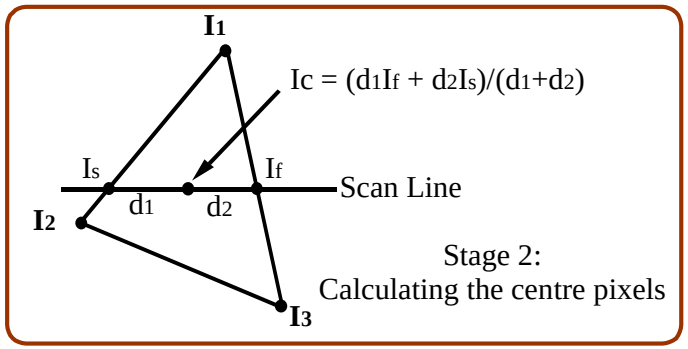
- Suppose that the designer of a game can choose / calculate the shade intensity at each vertex of a 3D object
- The shade at all other pixels of the polygon can be found as follows:

1. Interpolate to find the shade value at the boundary
2. Interpolate to find the shade values in the middle

Calculating the shades at the edges



Calculating the internal shades



Hardware support for polygon rendering

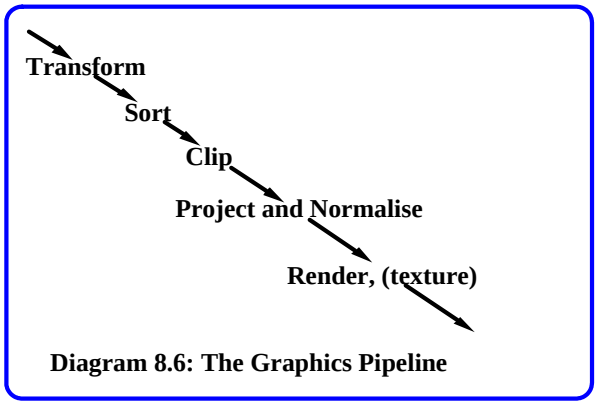


Diagram 8.6: The Graphics Pipeline