Graphics Pipeline

(Slides courtesy of Tamar Shinar)
Graphics Pipeline

Geometric Pipeline
- Transform
- Project
- Clip

Pixel Pipeline
- OpenGL application program
- Pixel operations
- Rasterizer
- Frame buffer
Transform
“Modelview” Transformation

- **Object coordinates**
  - Model

- **World coordinates**
  - View

- **Eye coordinates**
Project

The diagram illustrates the pipeline for rendering a graphics application using OpenGL. The pipeline consists of two main parts: the Geometric Pipeline and the Pixel Pipeline.

- **Geometric Pipeline:**
  - Transform
  - Project
  - Clip

- **Pixel Pipeline:**
  - Pixel operations
  - Rasterizer
  - Frame buffer

The OpenGL application program interacts with the transform process, which then passes the data to the project stage. After project transformation, the data is clipped to ensure only valid data is fed into the pixel operations. The rasterizer then converts the pixel data into a form that can be stored in the frame buffer.
Projection: map 3D scene to 2D image
Orthographic projection

Top

Side 1

Front

Side 2

=
OpenGL Orthogonal Viewing

glOrtho(left, right, bottom, top, near, far)
Perspective projection

[Image: Diagram of perspective projection with red lines converging at a single point, illustrating how objects appear smaller as they recede into the distance.]
OpenGL Perspective Viewing

\texttt{glFrustum(xmin,xmax,ymin,ymax,near,far)}

View volume (frustrum) for a perspective projection
Clip

Geometric Pipeline

Transform → Project → Clip

Pixel Pipeline

OpenGL application program → Pixel operations → Rasterizer → Frame buffer
Clip against view volume
Hidden Surface Removal
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

problem:
triangle intersection
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

**problem:**
occlusion cycle
Use a *z-buffer* for hidden surface removal

test depth on a pixel by pixel basis

red drawn last
Use a *z-buffer* for hidden surface removal at each pixel, record distance to the closest object that has been drawn in a *depth* buffer.
Use a **z-buffer** for hidden surface removal

http://www.beyond3d.com/content/articles/41/
Backface culling: another way to eliminate hidden geometry
Hidden Surface Removal in OpenGL

```gl
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

enable(GL_DEPTH_TEST);

enable(GL_CULL_FACE);
```

For a perspective transformation, there is more precision in the depth buffer for z-values closer to the near plane