3D Graphics and OpenGl

First Steps

Rendering of 3D Graphics

Objects defined in (virtual/mathematical) 3D space.

Rendering of 3D Graphics

Objects defined in (virtual/mathematical) 3D space.

We see surfaces of objects $\Rightarrow$ define surfaces.

Rendering of 3D Graphics

Objects defined in (virtual/mathematical) 3D space.

We see surfaces of objects $\Rightarrow$ define surfaces.

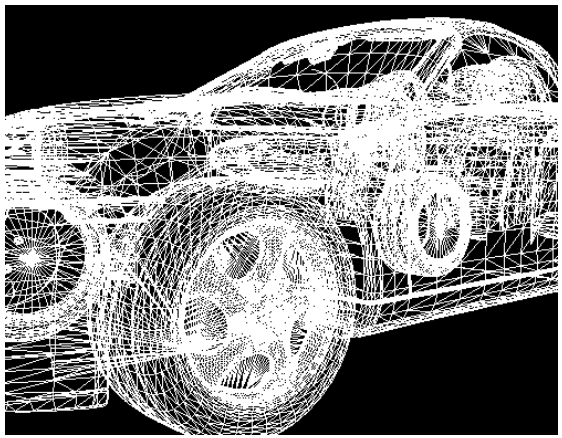
Triangles will be the fundamental element.

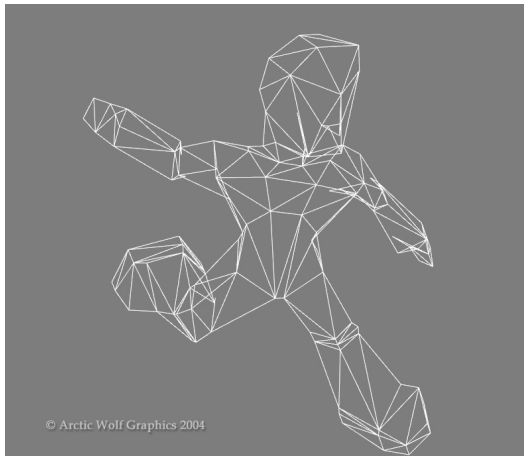
Rendering of 3D Graphics

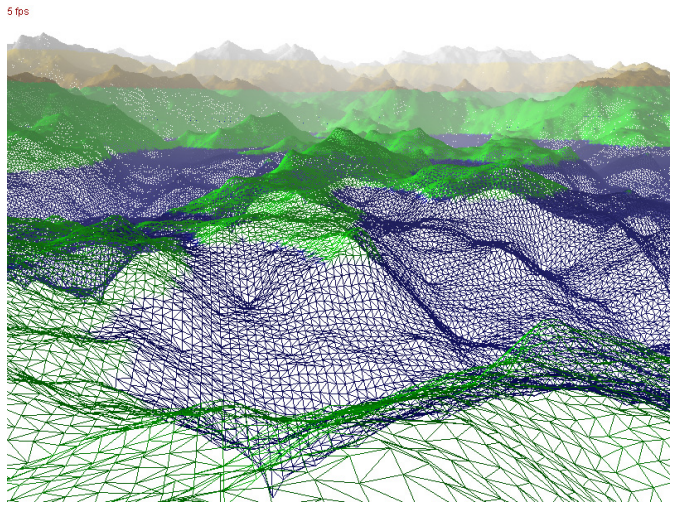
Objects defined in (virtual/mathematical) 3D space.

We see surfaces of objects $\Rightarrow$ define surfaces.

Triangles will be the fundamental element.





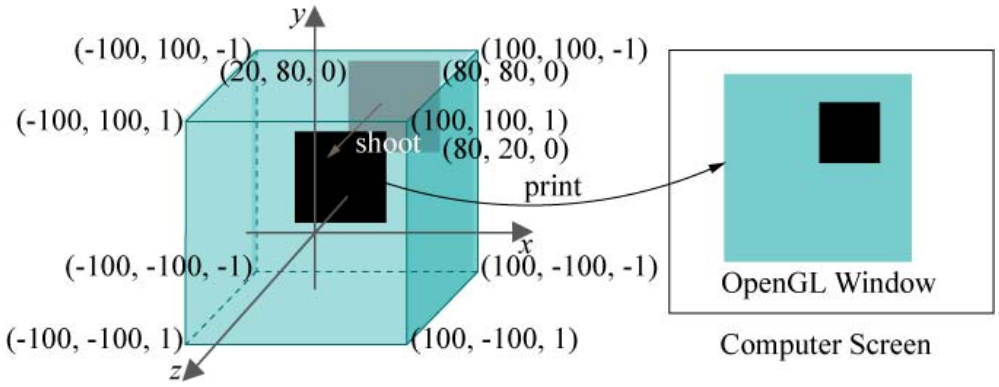


Rendering of 3D Graphics

Main objective: transfer (models built of) triangles from 3D space to 2D screen space. Add colors to the screen pixels covered by triangle (shading).

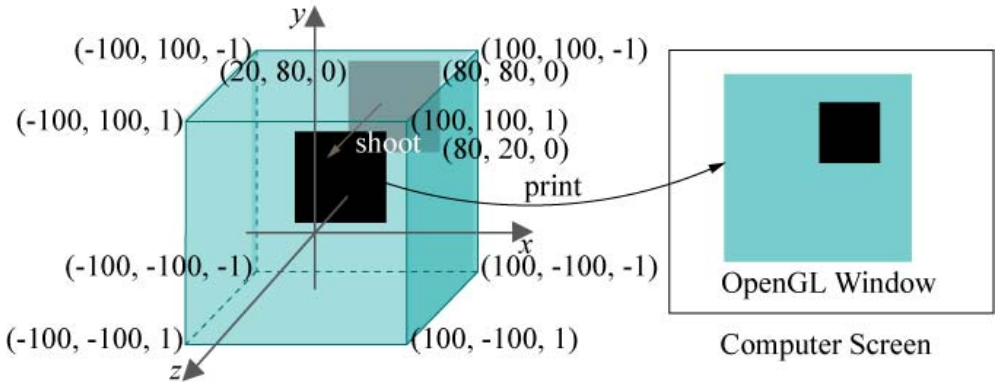
Rendering of 3D Graphics

Main objective: transfer (models built of) triangles from 3D space to 2D screen space. Add colors to the screen pixels covered by triangle (shading).

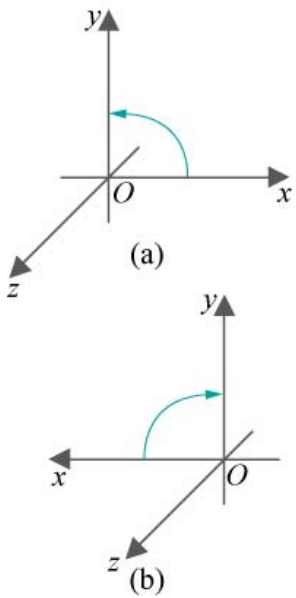


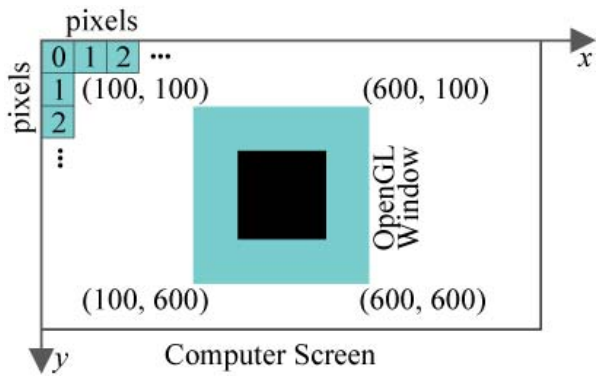
Rendering of 3D Graphics

Main objective: transfer (models built of) triangles from 3D space to 2D screen space. Add colors to the screen pixels covered by triangle (shading).



Coordinate systems:



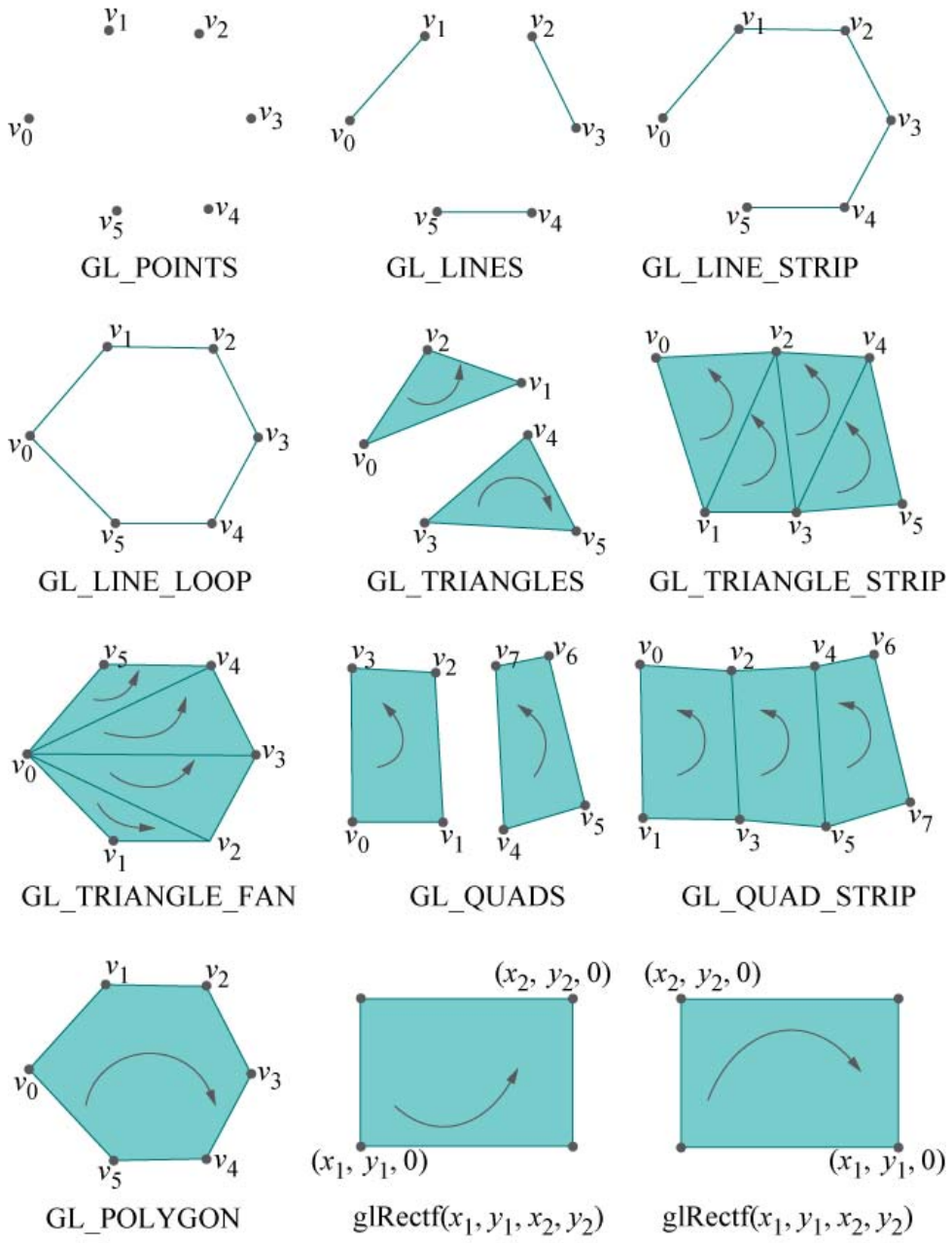


Vertices

Core data: vertices of triangles.

```
glBegin(GL_TRIANGLES);
   glVertex3f(20.0, 20.0, 0.0);
   glVertex3f(80.0, 20.0, 0.0);
   glVertex3f(80.0, 80.0, 0.0);
   .
   .
glEnd();
```

Other OpenGL Primitives

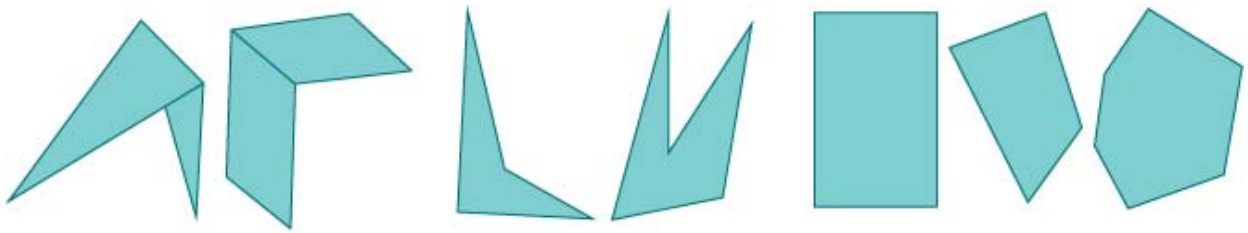


OpenGL Primitives

Polygons and quads are divided into triangles by OpenGL before rendering. Must be plane and convex

OpenGL Primitives

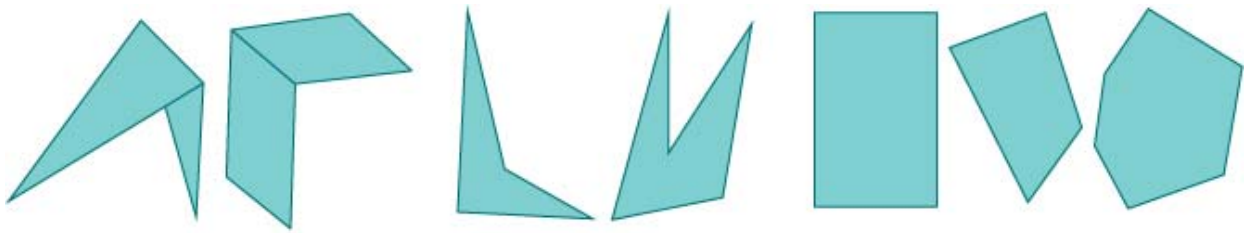
Polygons and quads are divided into triangles by OpenGL before rendering. Must be plane and convex



Not planar, not convex Planar, not convex Planar and convex

OpenGL Primitives

Polygons and quads are divided into triangles by OpenGL before rendering. Must be plane and convex



For efficiency, use array lists (single rendering call accessing array of many points) and display lists (precompiled and stored groups of OpenGL commands, including declarations of geometry/primitives). See sections 3.1 and 3.2.

Geometry

Core data: triangles

Geometry

Core data: triangles

Triangle vertices and associated data:

- Position
- Color
- Normal vector
- Texture coordinate

Geometry

Core data: triangles

Triangle vertices and associated data:

- Position
- Color
- Normal vector
- Texture coordinate

Vertex data are interpolated across triangle at rendering time (details of interpolation and other parts of rendering later).

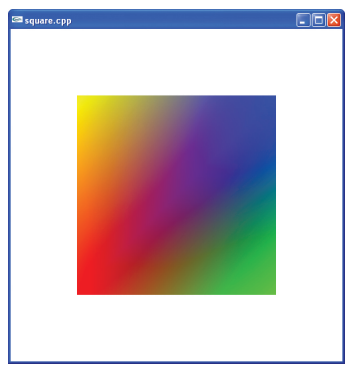
Geometry

Core data: triangles

Triangle vertices and associated data:

- Position
- Color
- Normal vector
- Texture coordinate

Vertex data are interpolated across triangle at rendering time (details of interpolation and other parts of rendering later).



OpenGL has a state

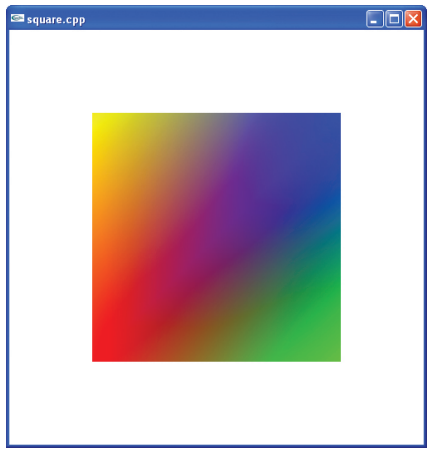
State machine: Long list of set variables affecting rendering. Value fixed after initialization until changed. (Alternative would be to give long list of parameters for all rendering calls).

OpenGL has a state

State machine: Long list of set variables affecting rendering. Value fixed after initialization until changed. (Alternative would be to give long list of parameters for all rendering calls).

E.g., setting (foreground/vertex) color using `glColor`:

```
glBegin(GL_QUADS);
   glColor3f(1.0, 0.0, 0.0);
   glVertex3f(20.0, 20.0, 0.0);
   glColor3f(0.0, 1.0, 0.0);
   glVertex3f(80.0, 20.0, 0.0);
   glColor3f(0.0, 0.0, 1.0);
   glVertex3f(80.0, 80.0, 0.0);
   glColor3f(1.0, 1.0, 0.0);
   glVertex3f(20.0, 80.0, 0.0);
glEnd()
```



Projections

Transfer (models built of triangles built of vertex) points from 3D space to 2D screen space.

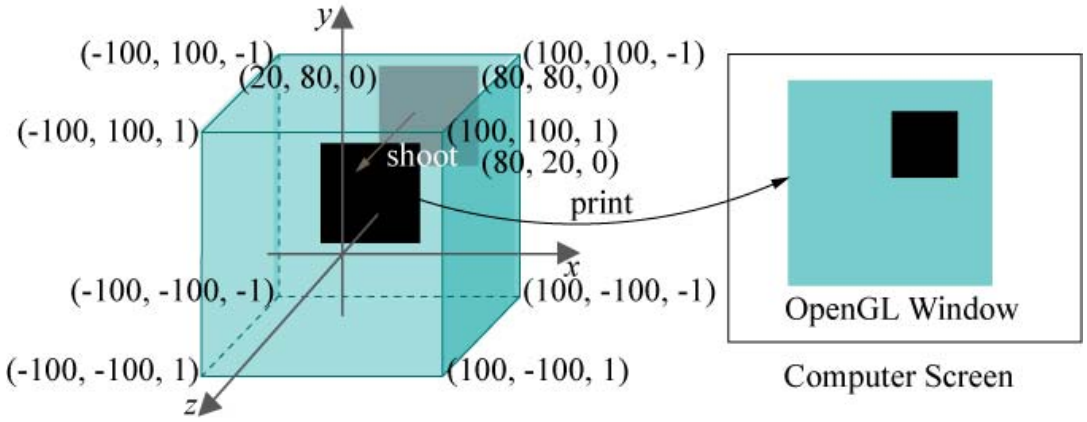
Projections

Transfer (models built of triangles built of vertex) points from 3D space to 2D screen space.

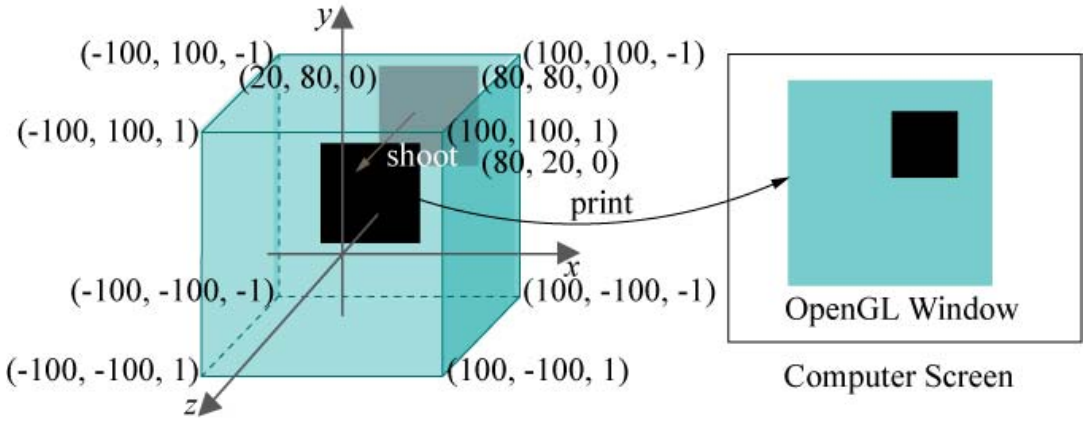
Two types:

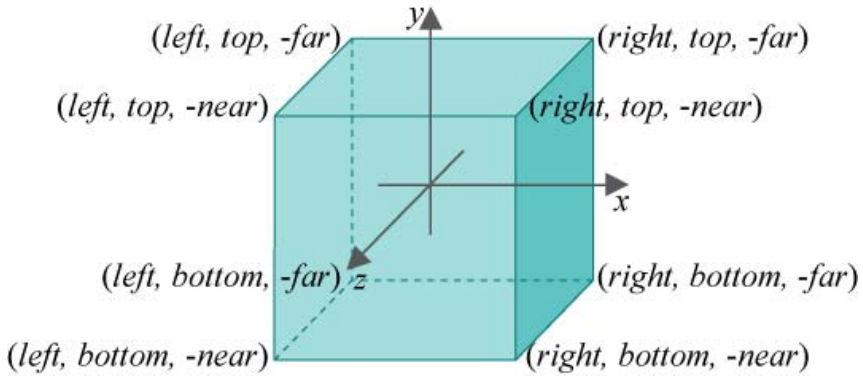
- Orthographic
- Perspective

Orthographic Projection

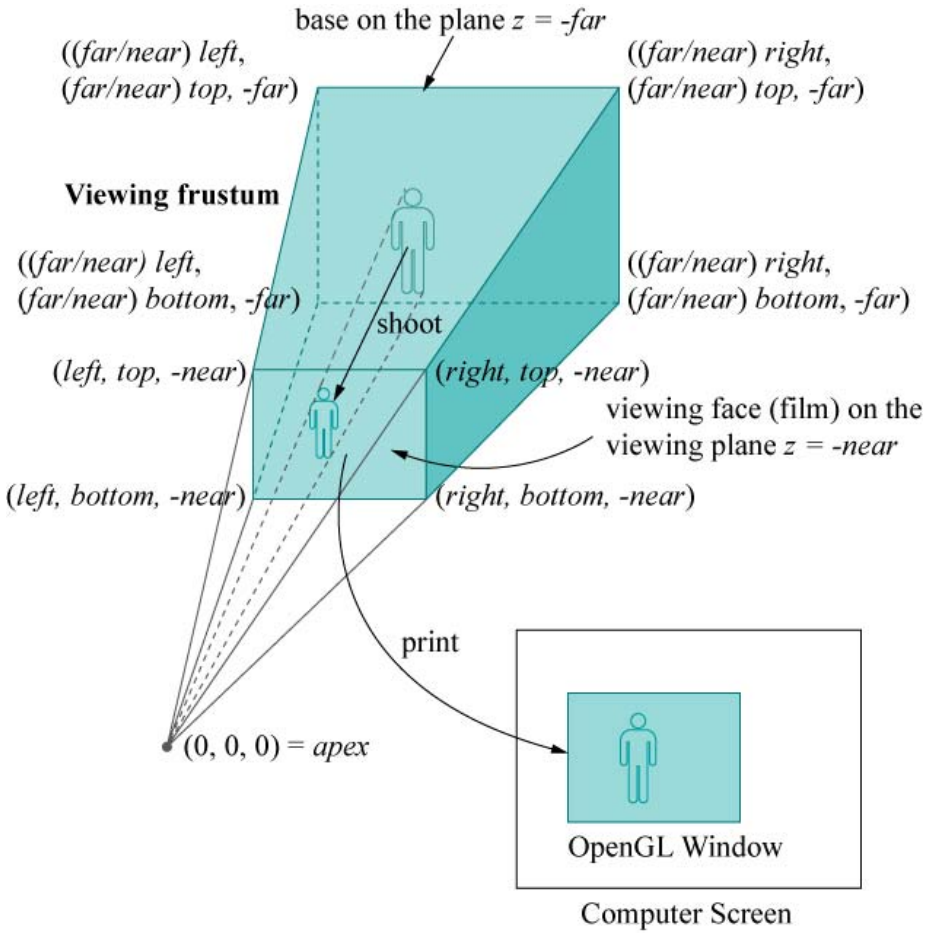


Orthographic Projection

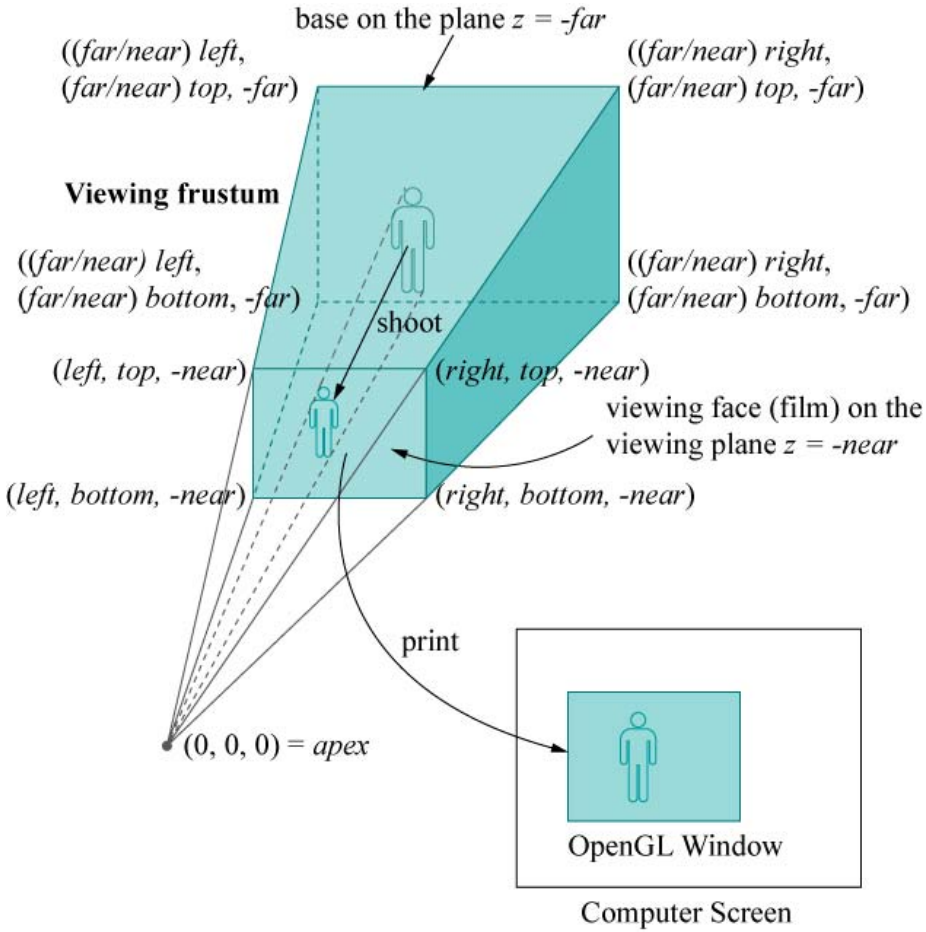


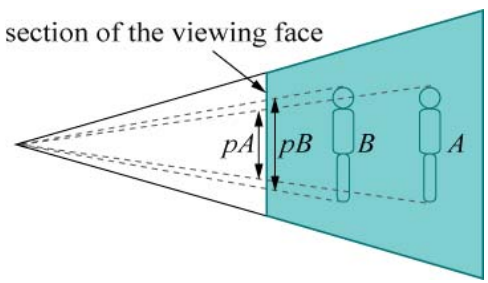


Perspective Projection



Perspective Projection

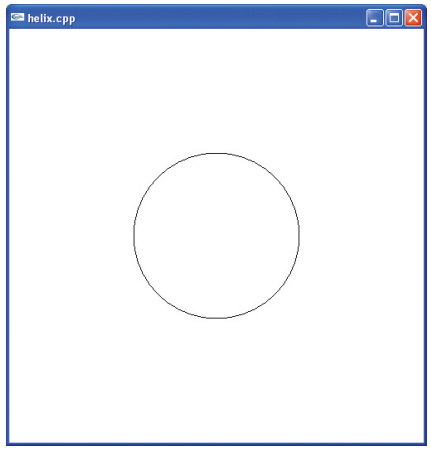


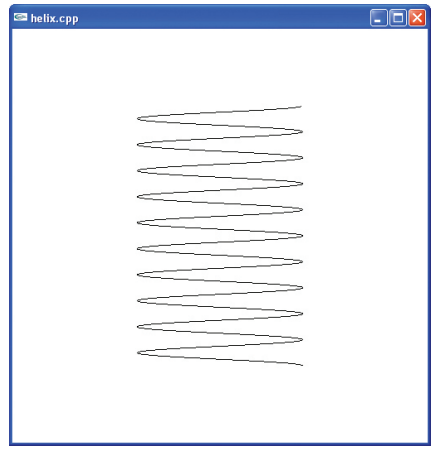


Perspective

Helix curve:

Orthographic:

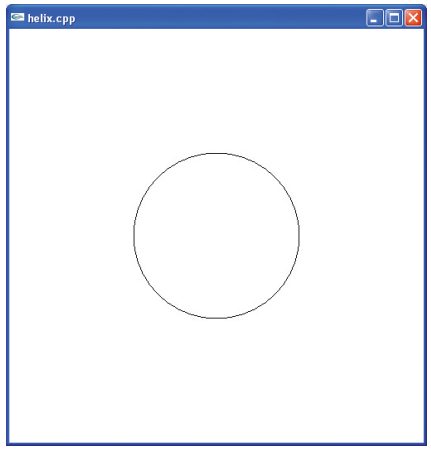


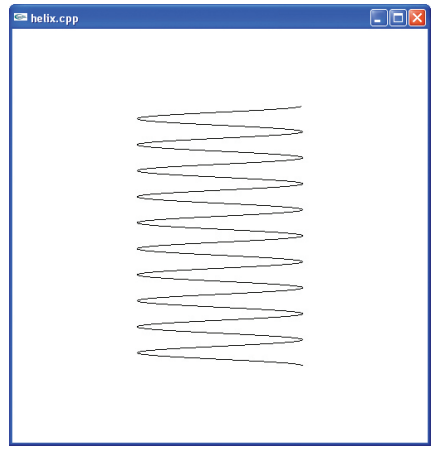


Perspective

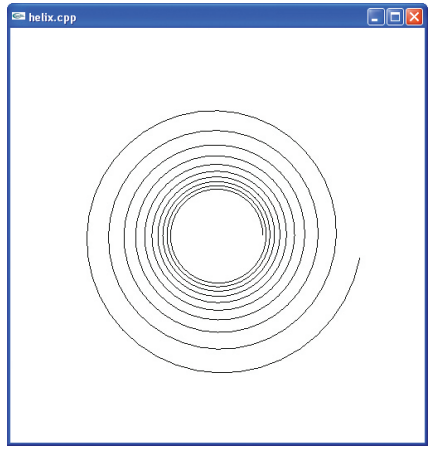
Helix curve:

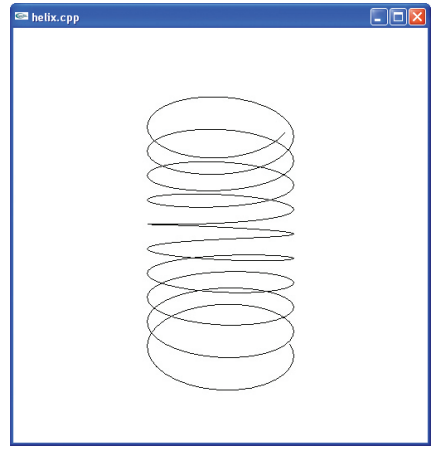
Orthographic:





Projective:



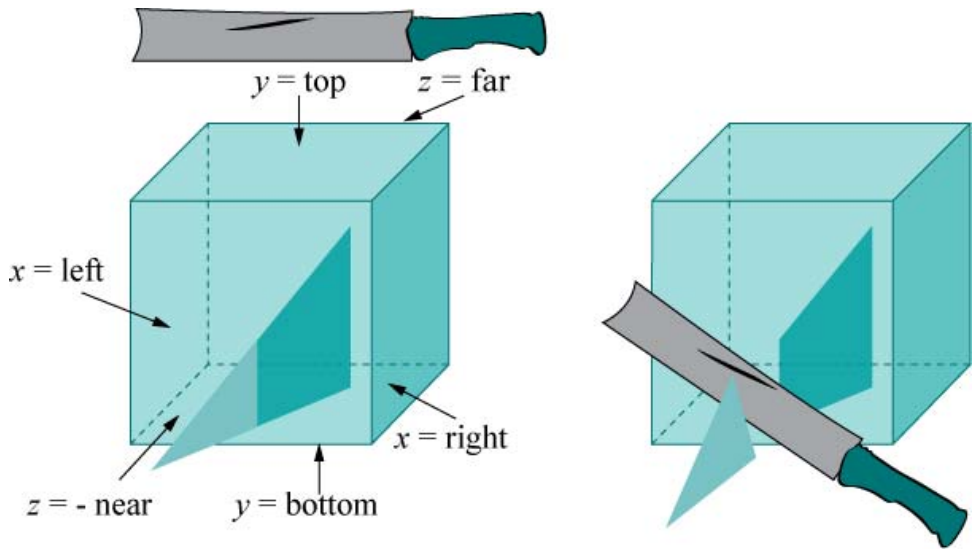


Clipping before Projection

The geometry is clipped against the viewing area planes before projection. Further clipping planes can be specified manually.

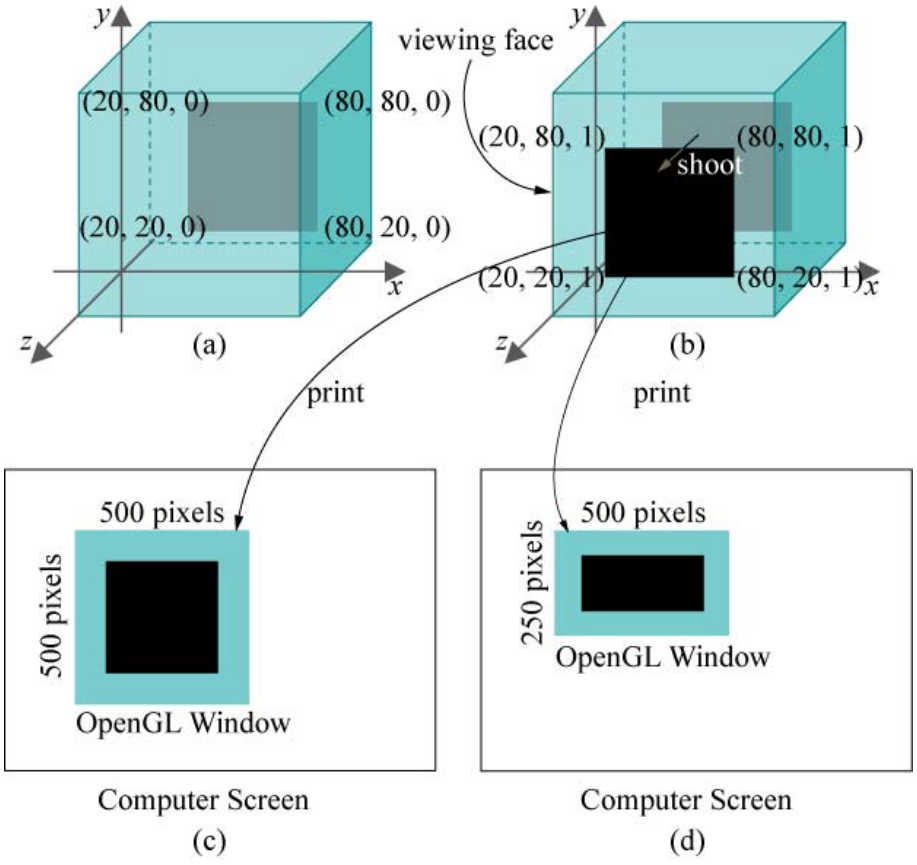
Clipping before Projection

The geometry is clipped against the viewing area planes before projection. Further clipping planes can be specified manually.



Stretch after Projection

The projected image is stretched to the screen/window size after projection.



OpenGL Buffers

A buffer is a screensize 2D array of (pixel) data. Several buffers are available in OpenGL (collectively called the framebuffer).

OpenGL Buffers

A buffer is a screensize 2D array of (pixel) data. Several buffers are available in OpenGL (collectively called the framebuffer).

Two important buffer types:

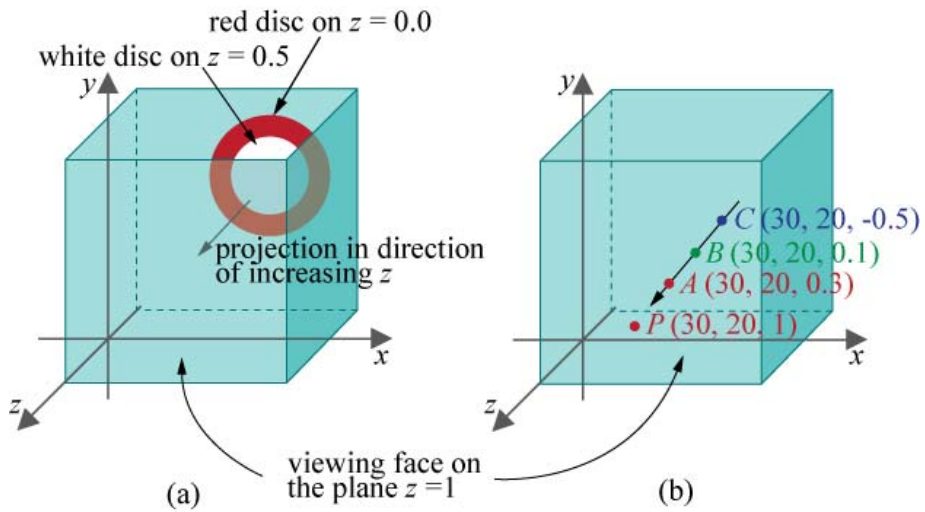
- Color buffers. Hold the color values to be shown on screen.
- Depth buffer. Resolves hidden surface removal.

OpenGL Buffers

A buffer is a screensize 2D array of (pixel) data. Several buffers are available in OpenGL (collectively called the framebuffer).

Two important buffer types:

- Color buffers. Hold the color values to be shown on screen.
- Depth buffer. Resolves hidden surface removal.



(Free)GLUT

- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).

(Free)GLUT

- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).
- Handles keyboard/mouse input, windowing management.

(Free)GLUT

- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).
- Handles keyboard/mouse input, windowing management.
- Event loop.

(Free)GLUT

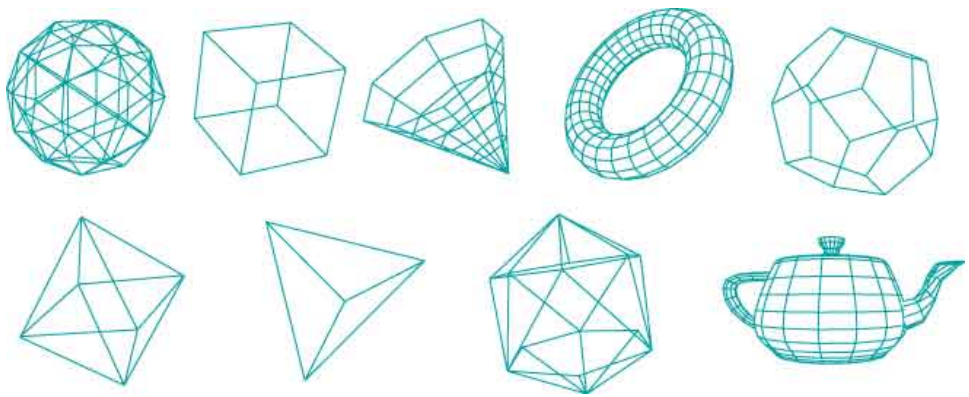
- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).
- Handles keyboard/mouse input, windowing management.
- Event loop.
- OpenGL programmer associates callback functions with events.

(Free)GLUT

- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).
- Handles keyboard/mouse input, windowing management.
- Event loop.
- OpenGL programmer associates callback functions with events.
- Animation through timed callbacks (`glutTimerFunc()`) or idle time callback (`glutIdleFunc()`).

(Free)GLUT

- Library that abstracts away OS-specific interface/libraries between OpenGL and OS (incl. creation of framebuffer and double buffering swaps).
- Handles keyboard/mouse input, windowing management.
- Event loop.
- OpenGL programmer associates callback functions with events.
- Animation through timed callbacks (`glutTimerFunc()`) or idle time callback (`glutIdleFunc()`).
- Commands for triangles for basic models (cube, cone, sphere, torus, teapot,...).



GLU is a lower level utility library (may also appear as command name prefix).