CSCI3260 PRINCIPLES OF COMPUTER GRAPHICS

Tutorial 3

Colman Leung

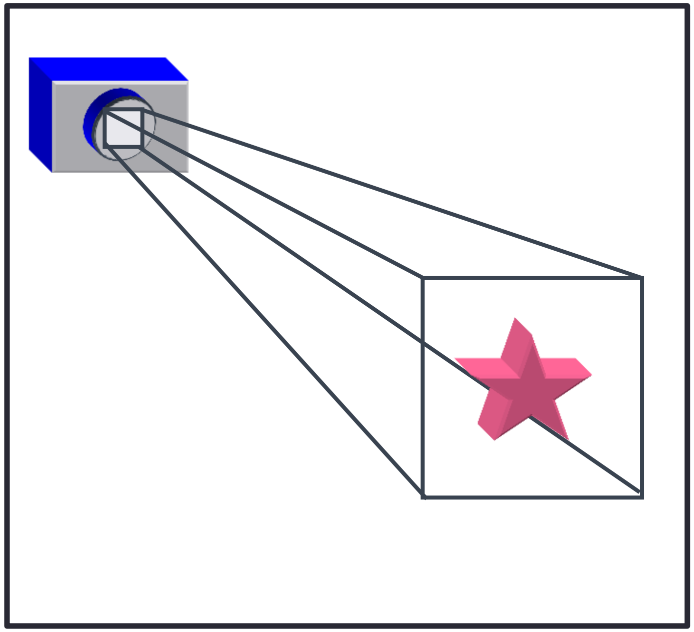
Outline

- Complex 3D objects drawing (GLU & GLUT lib)
- Perspective and Orthographic Projection
- Viewport Transformation

Recall ModelView Transformation

Arrange the scene into the desired composition: Modeling transformation

Point the camera at the scene: Viewing transformation



Drawing GLU objects

- First create a quadric object

```
GLUquadricObj *obj = gluNewQuadric();
```

It describes **how** to draw geometric shapes.
Default is to fill shapes using polygon and strip primitives

- You can change the style using these functions:

```
void gluQuadricDrawStyle(GLUquadricObj *obj, GLenum drawStyle)
void gluQuadricNormals(GLUquadricObj *obj, GLenum normals)
void gluQuadricOrientation(GLUquadricObj *obj, GLenum orientation)
void gluQuadricTexture(GLUquadricObj *obj, GLboolean textureCoords)
```

Drawing GLU objects

Cylinder

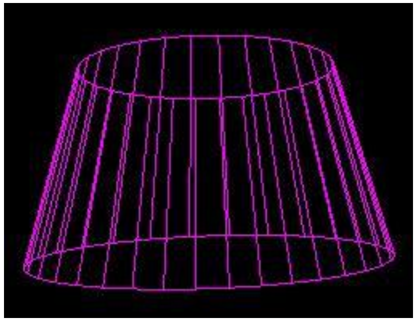
gluCylinder(obj, baseRadius, topRadius, height, slices, stacks)

Disk

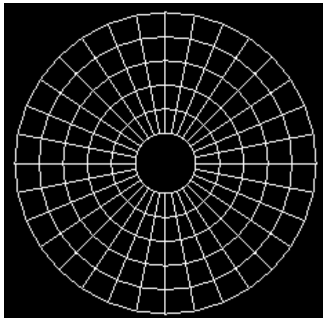
gluDisk(obj, innerRadius, outerRadius, slices, loops)

Sphere

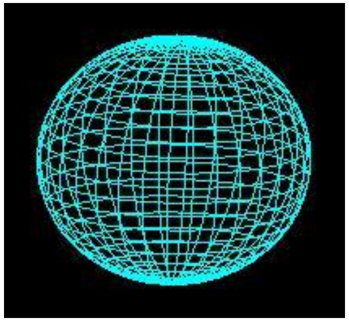
gluSphere(obj, radius, slices, stacks)



Slices: 32
Stacks: 1



Slices: 32
Loops: 5

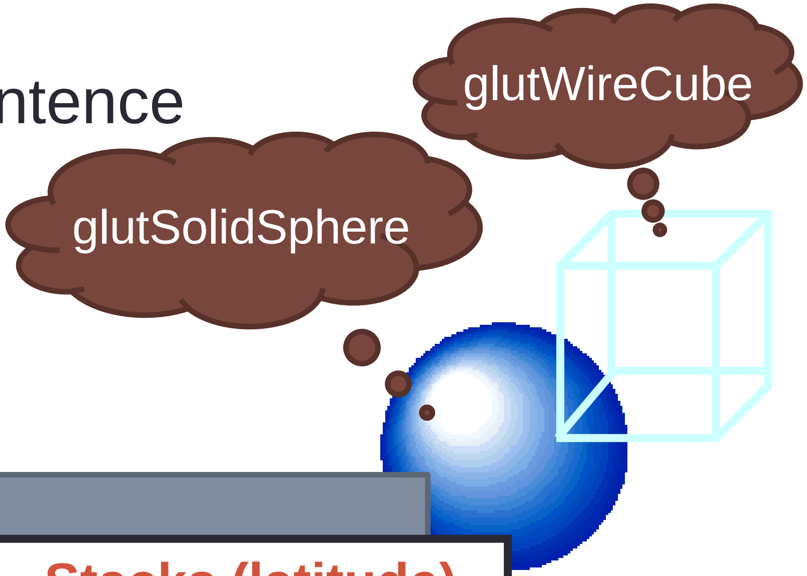


Slices: 32
Stacks: 32

Drawing GLUT objects

- GLUT objects are drawn in one sentence

```
glutWireCube(1.0);
glutSolidSphere(2, 10, 10);
```



Radius

Stacks (latitude)

```
glutSolidSphere(2, 50, 10);
```

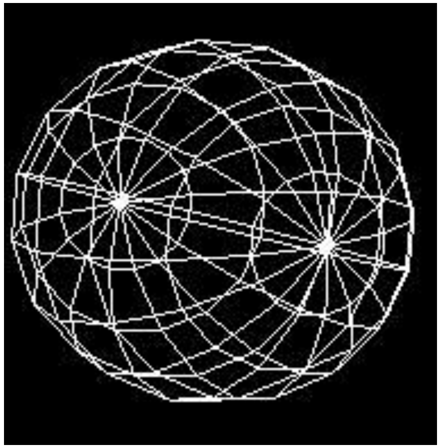
Slices (longitude)

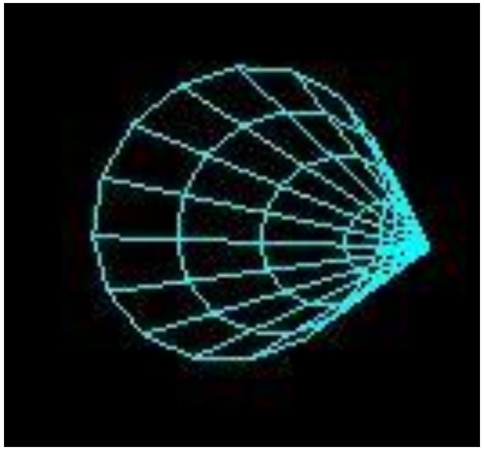
GLUT flavors

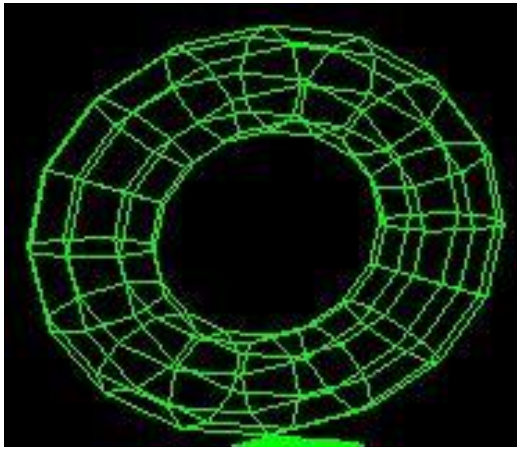
GLUT object

Glut shapes

- Sphere glutWireSphere(radius, slices, stacks)
- Cone glutWireCone(baseRadius, height, slices, stacks)
- Torus glutWireTorus(innerRadius, outerRadis, sides, rings)

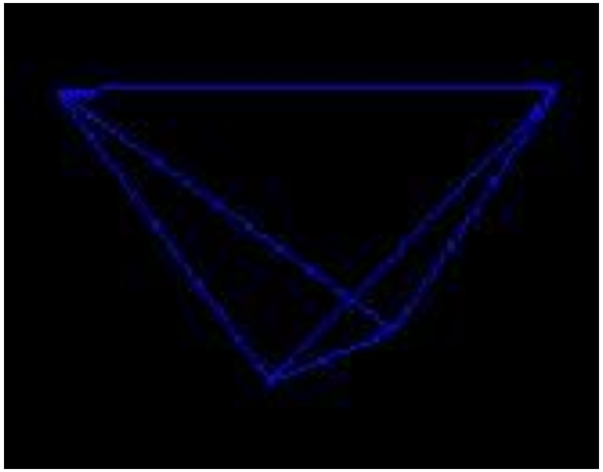


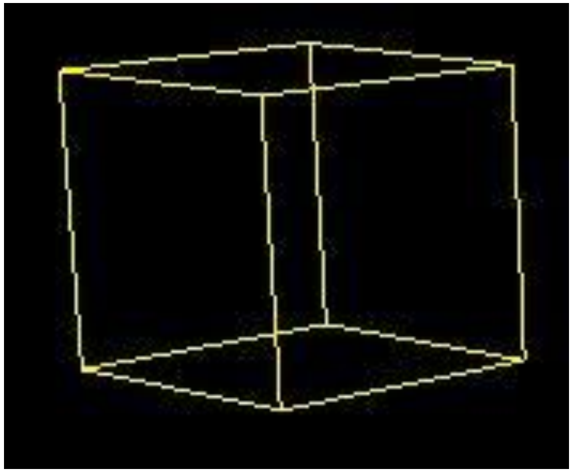




Glut shapes

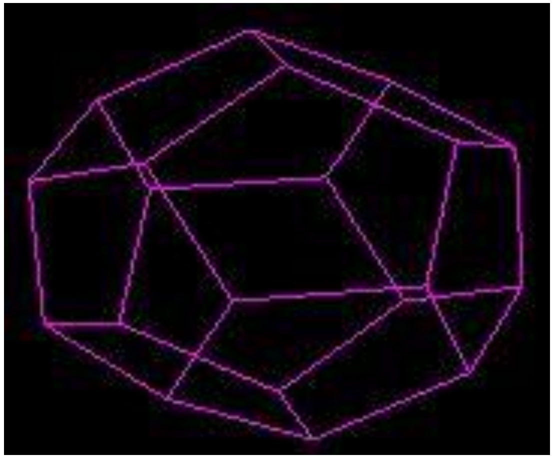
- Tetrahedron glutWireTetrahedron()
- Cube glutWireCube(size)
- Octahedron glutWireOctahedron()

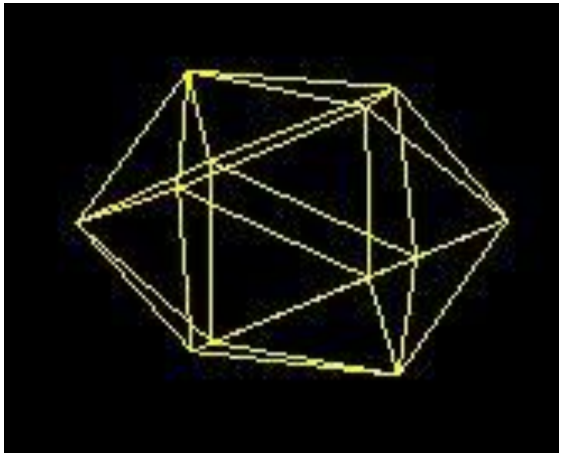


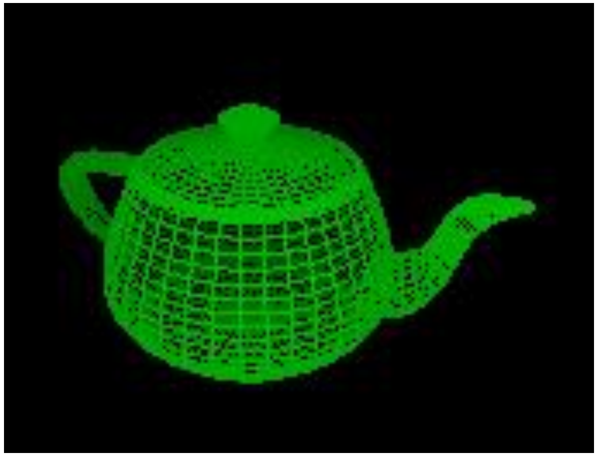


Glut shapes

- Dodecahedron glutWireDodecahedron()
- Icosahedron glutWireIcosahedron()
- Teapot glutWireTeapot(size)

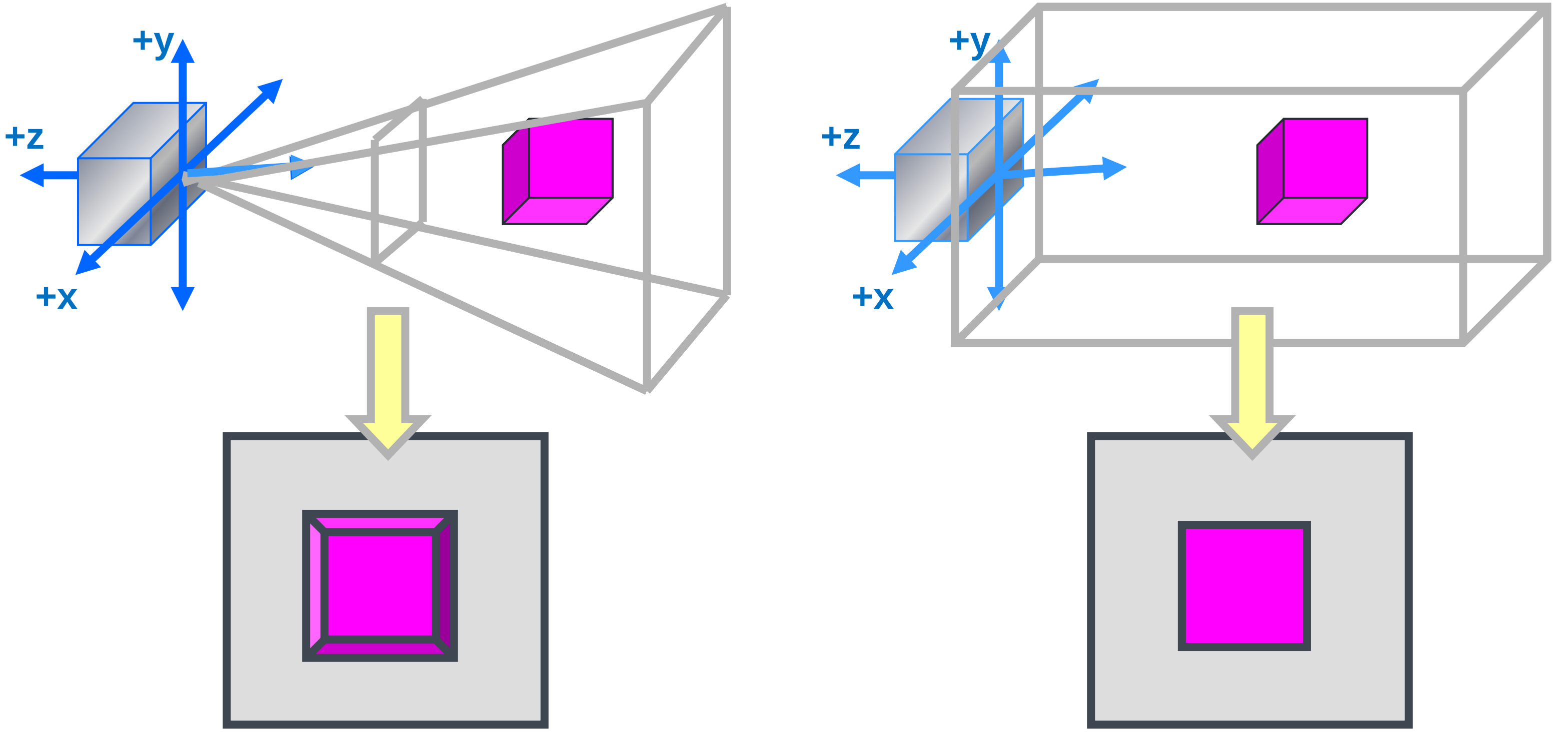






Projection Transformation

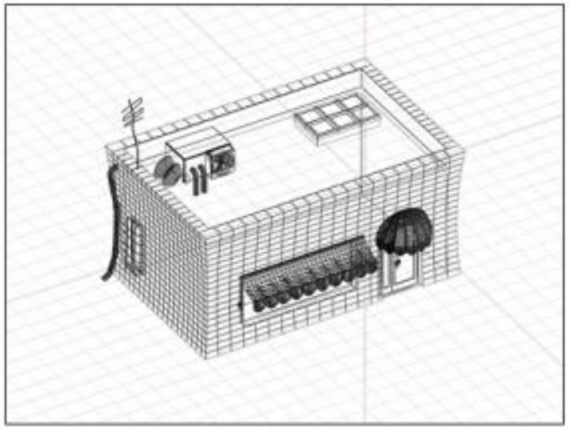
- In OpenGL, you can specify the projection type (perspective or orthographic) and the parameters of the projection



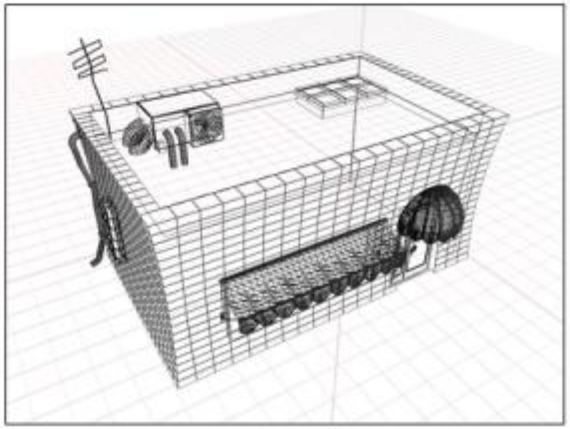
- Viewing Volume Clipping – Any primitives that lie outside the viewing volume are clipped and will not be displayed in final scene

Perspective / Orthographic views

- In the perspective view, objects which are far away are smaller than those nearby.
- In the orthographic view, all objects appear at the same scale.



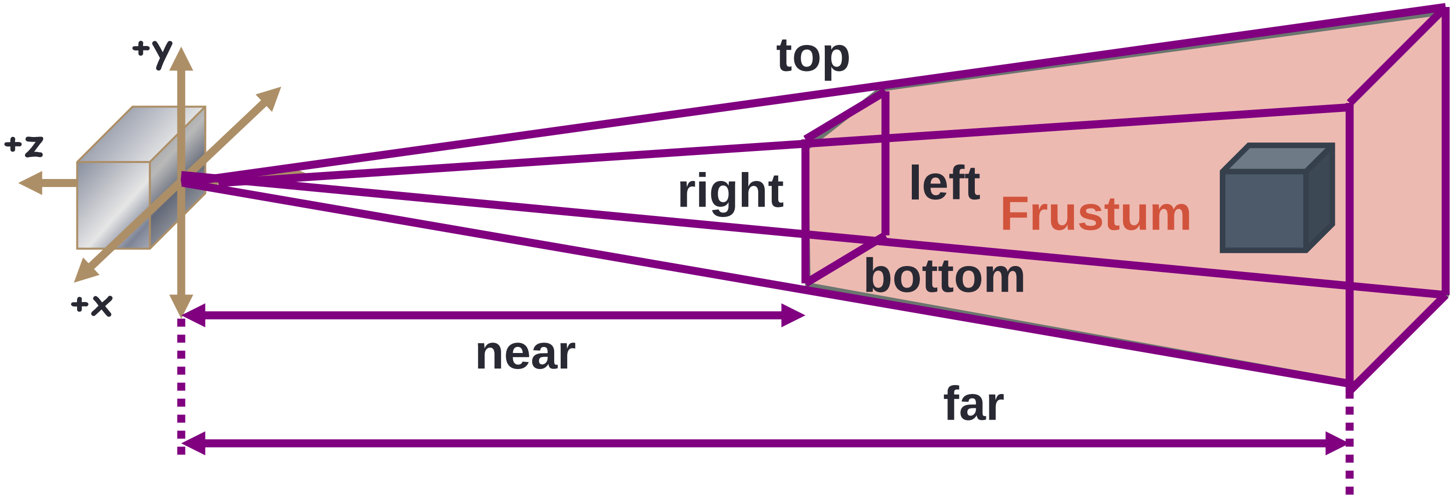
orthographic



perspective

Perspective Projection using glFrustum

```
void glFrustum(GLdouble left, GLdouble right, GLdouble bottom,
GLdouble top, GLdouble near, GLdouble far);
```



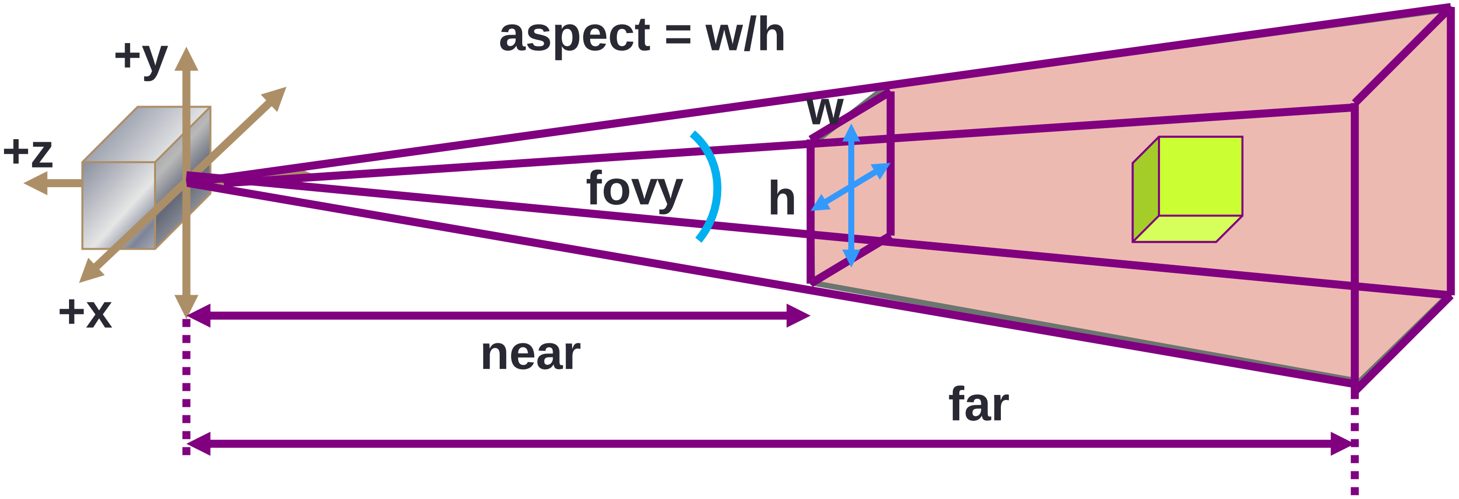
Specify the viewing volume using glFrustum()

Perspective Projection using gluPerspective

```
gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble
zNear, GLdouble zFar);
```

fovy: the field of view angle, in degrees, in the y direction.

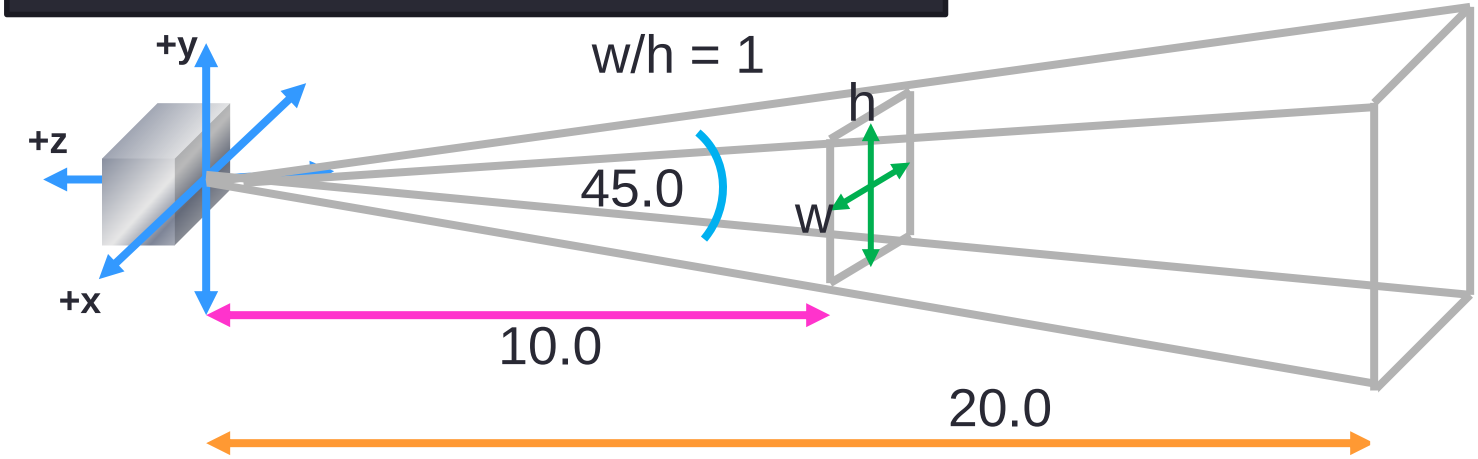
aspect: the aspect ratio that determines the field of view in the x direction. The aspect ratio is the ratio of x (width) to y (height).



Specify the viewing volume using gluPerspective()

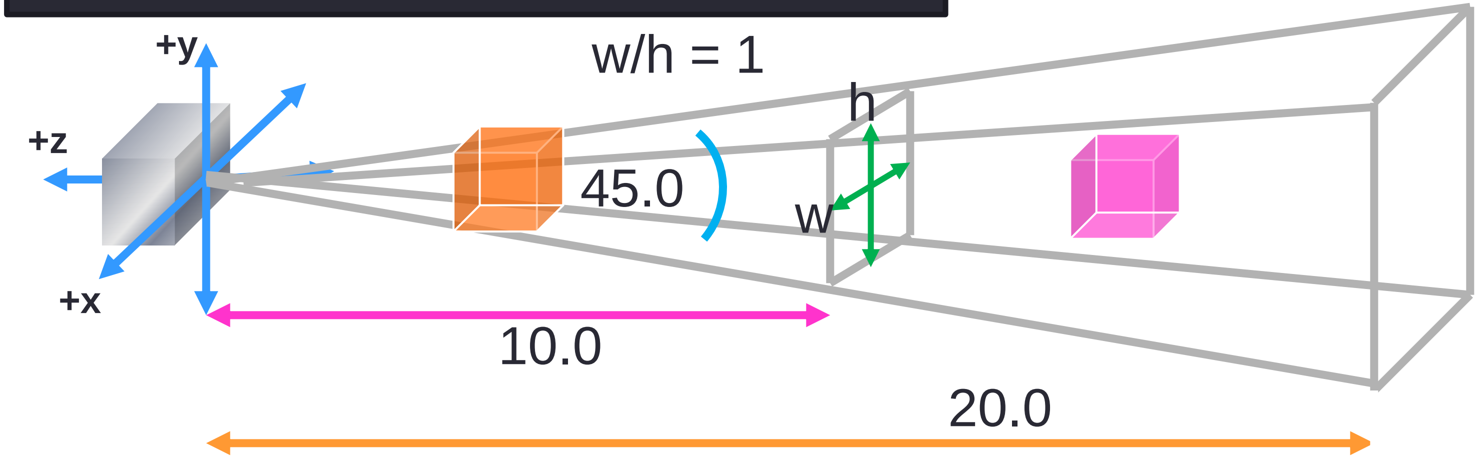
Perspective Projection using gluPerspective (An Example)

```
gluPerspective(45.0, 1.0, 10.0, 20.0);
```



Perspective Projection using gluPerspective (An Example)

```
gluPerspective(45.0, 1.0, 10.0, 20.0);
```

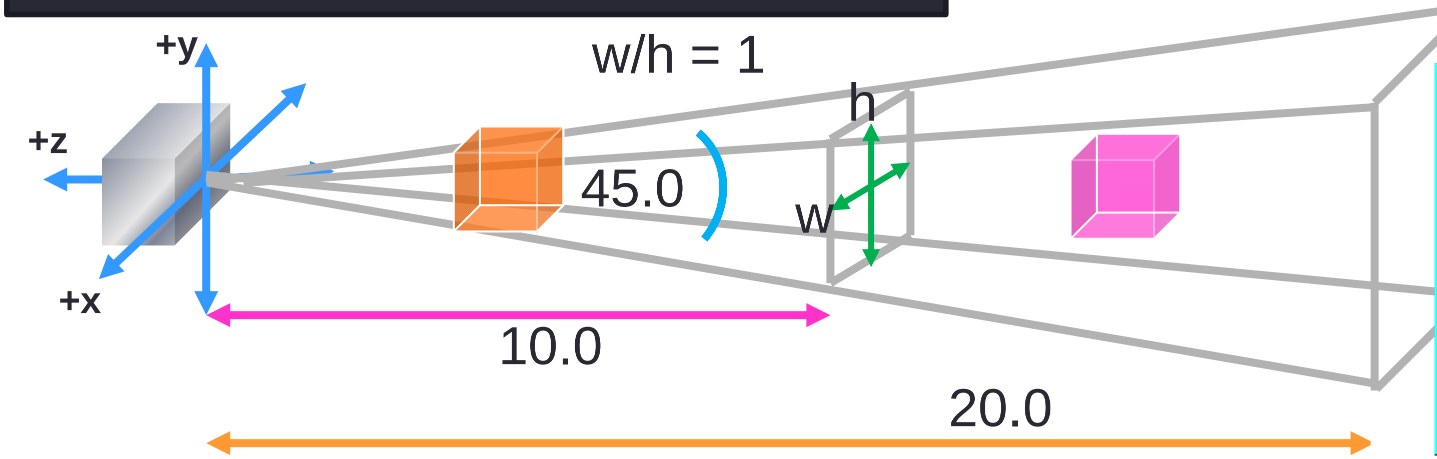


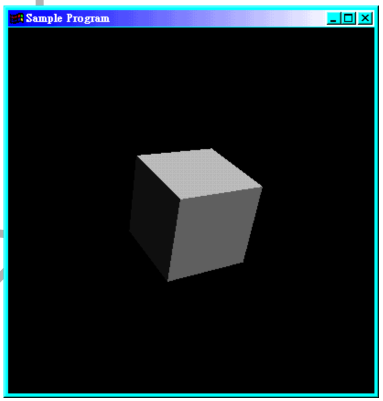
```
glMatrixMode(GL_MODELVIEW);
glTranslated(0.0, 0.0, -5.0);
glutSolidCube(1.0);
```

```
glMatrixMode(GL_MODELVIEW);
glTranslated(0.0, 0.0, -15.0);
glutSolidCube(1.0);
```

Perspective Projection using gluPerspective (An Example)

```
gluPerspective(45.0, 1.0, 10.0, 20.0);
```



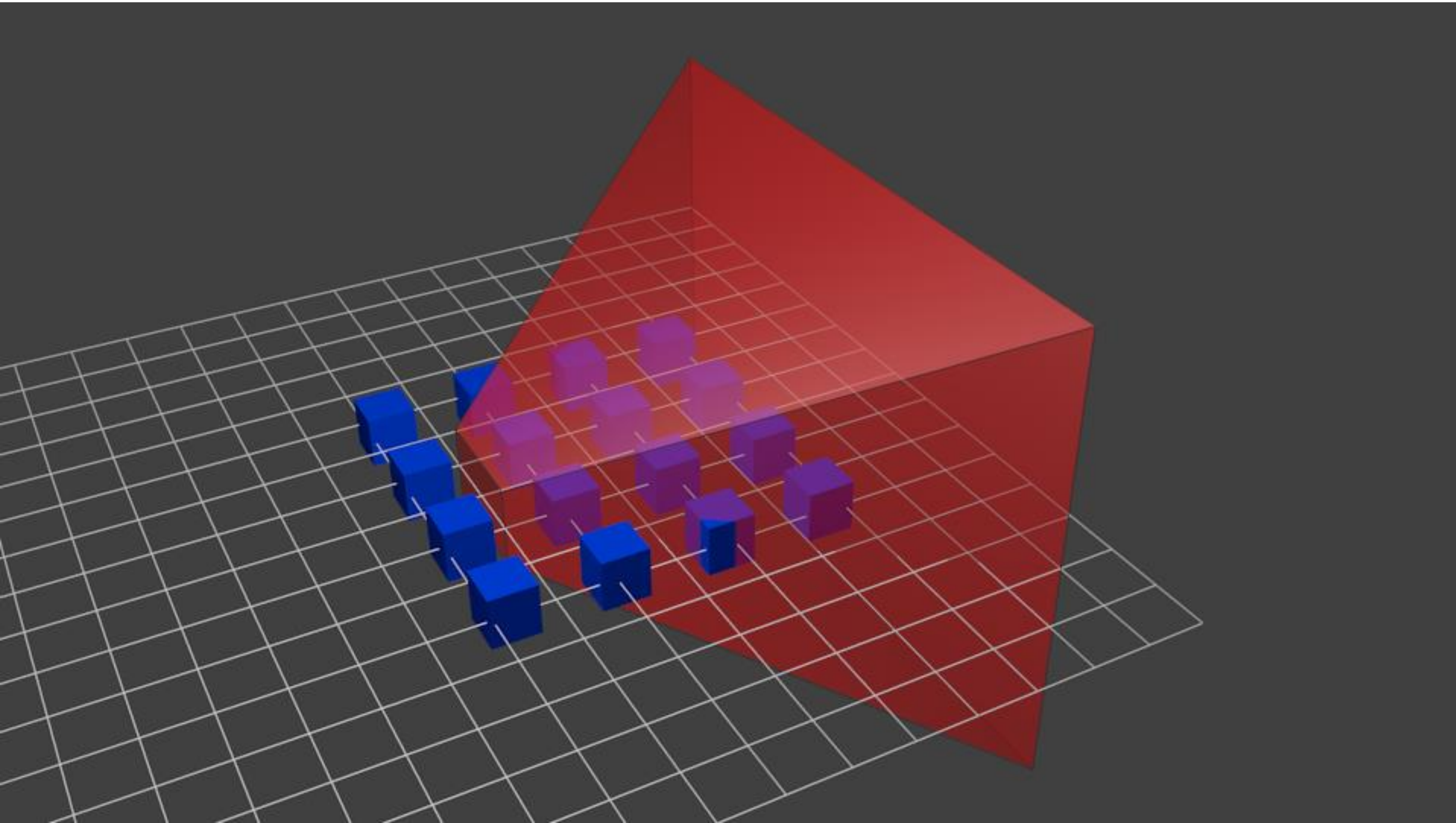


```
glMatrixMode(GL_MODELVIEW);
glTranslated(0.0, 0.0, -5.0);
glutSolidCube(1.0);
```

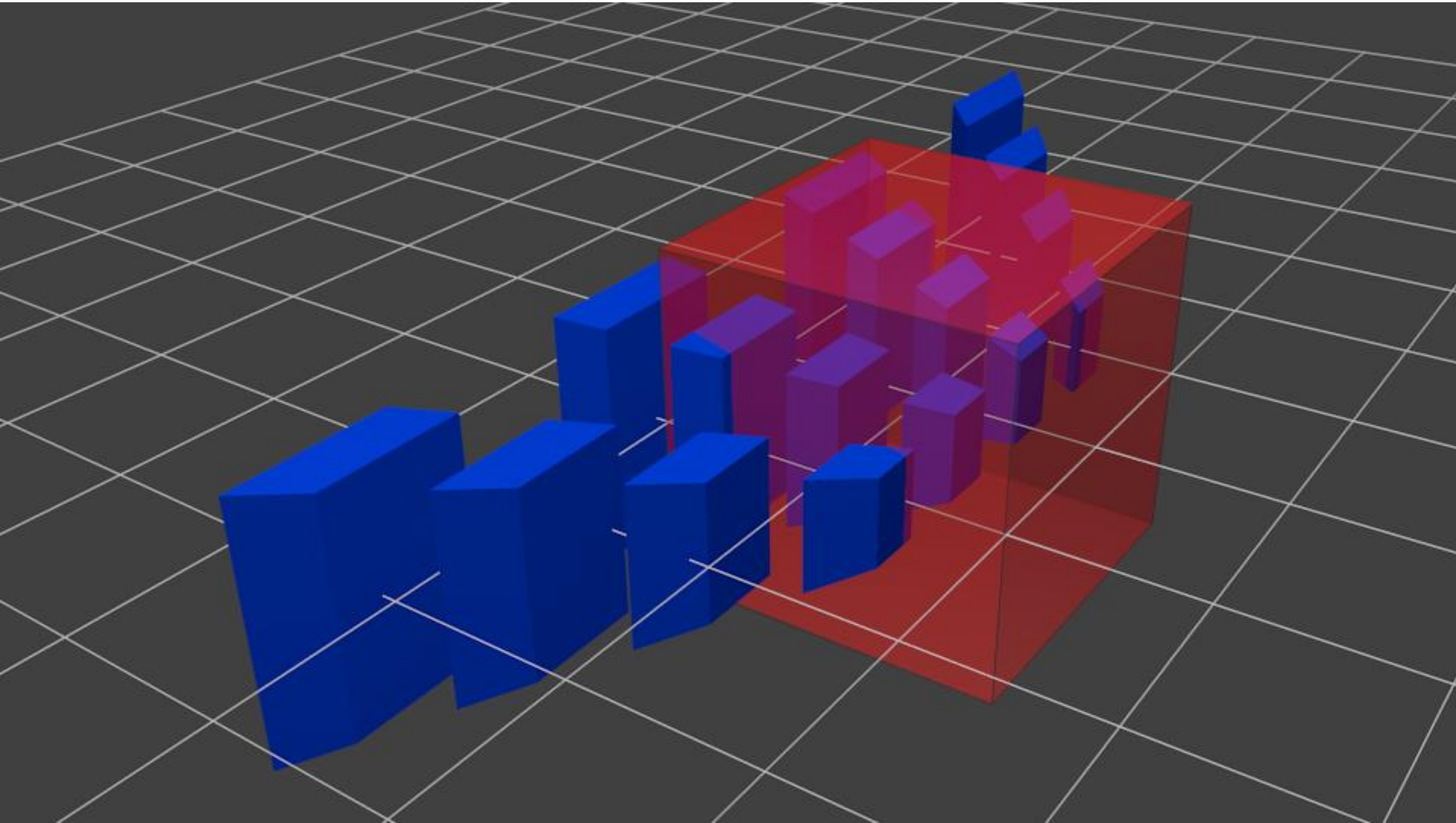
Nothing can be seen on screen!!

```
glMatrixMode(GL_MODELVIEW);
glTranslated(0.0, 0.0, -15.0);
glutSolidCube(1.0);
```

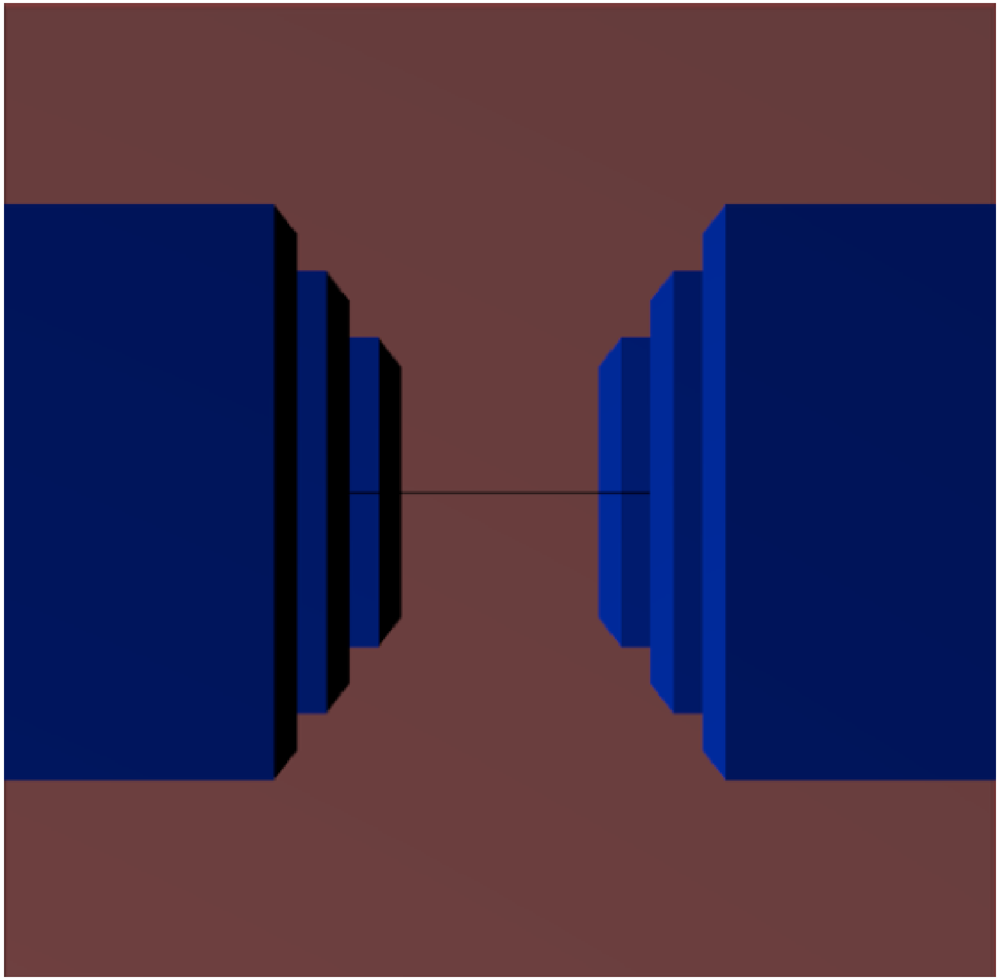
Before multiplying with Projection Matrix



After multiplying with Projection Matrix



What the camera sees

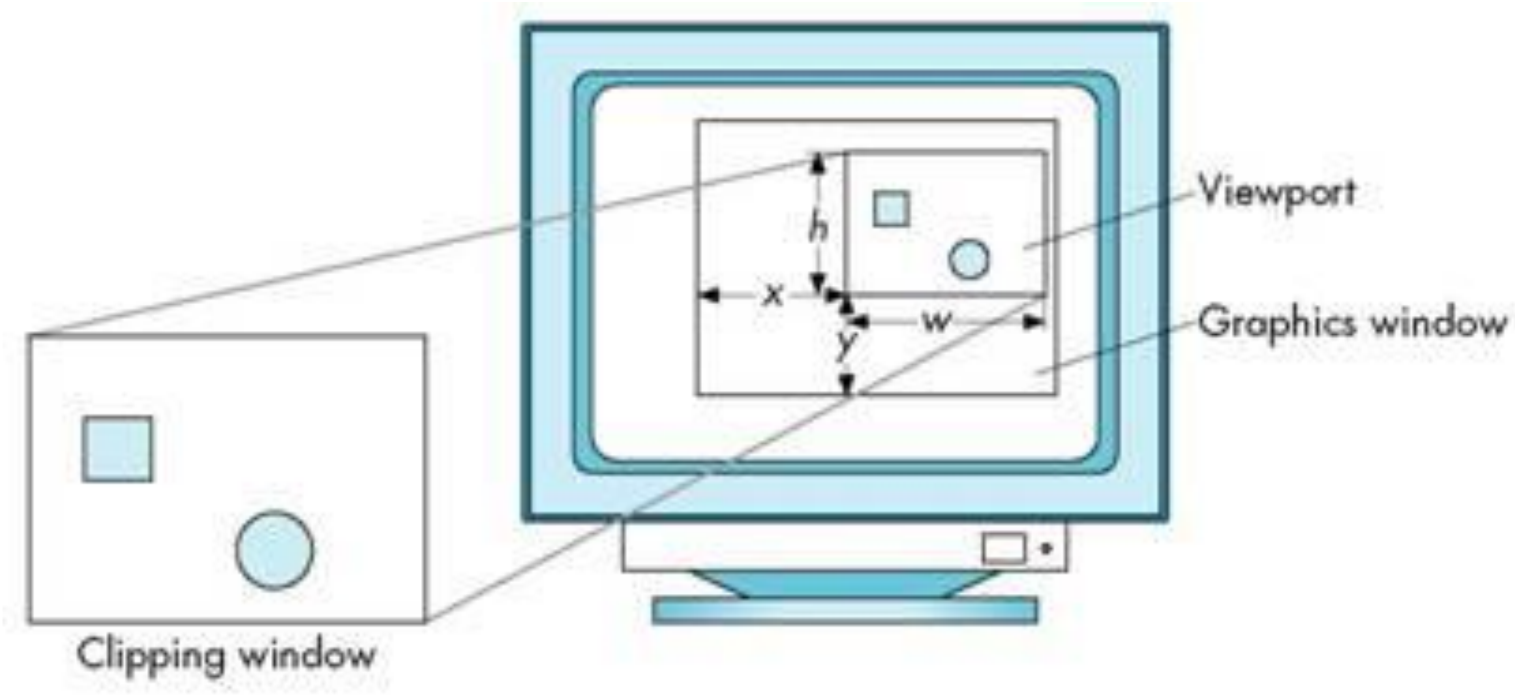


Viewport transformation

- void glViewport(GLint x, GLint y, GLsizei width, GLsizei height) - set the viewport
  - x, y - Specify the lower left corner of the viewport rectangle, in pixels. The default is (0, 0).
  - width, height - Specify the width and height of the viewport. When a GL context is first attached to a window, *width* and *height* are set to the dimensions of that window.

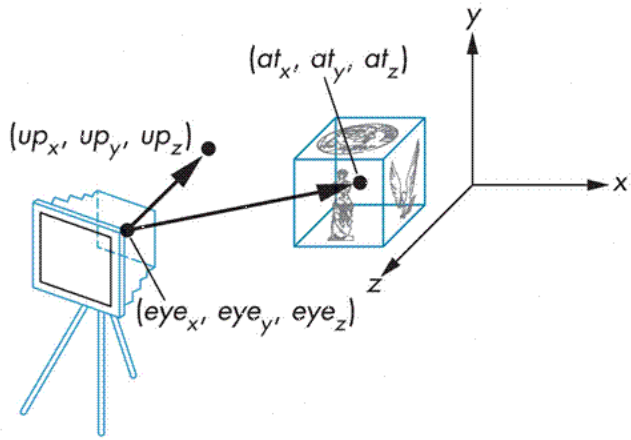
Viewport transformation

- glViewport() + gluPerspective();



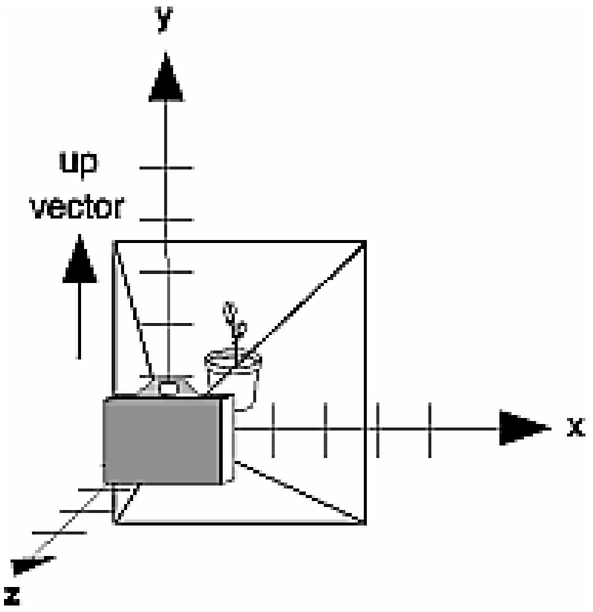
gluLookAt

- gluLookAt (*eyeX* , *eyeY* , *eyeZ* , *centerX* , *centerY* , *centerZ* , *upX* , *upY* , *upZ*)
  - *eyeX*, *eyeY*, *eyeZ* Specifies the position of the eye point.
  - *centerX*, *centerY*, *centerZ* Specifies the position of the reference point.
  - *upX*, *upY*, *upZ* Specifies the direction of the *up* vector.



About gluLookAt()

- Default Setting for gluLookAt() function
  - In default condition, the camera is located at original point; points to the -z axis; up direction is y axis.
  - gluLookAt(0, 0, 0, //position
    0, 0, -100, // direction
    0, 1.0, 0); // up direction



End