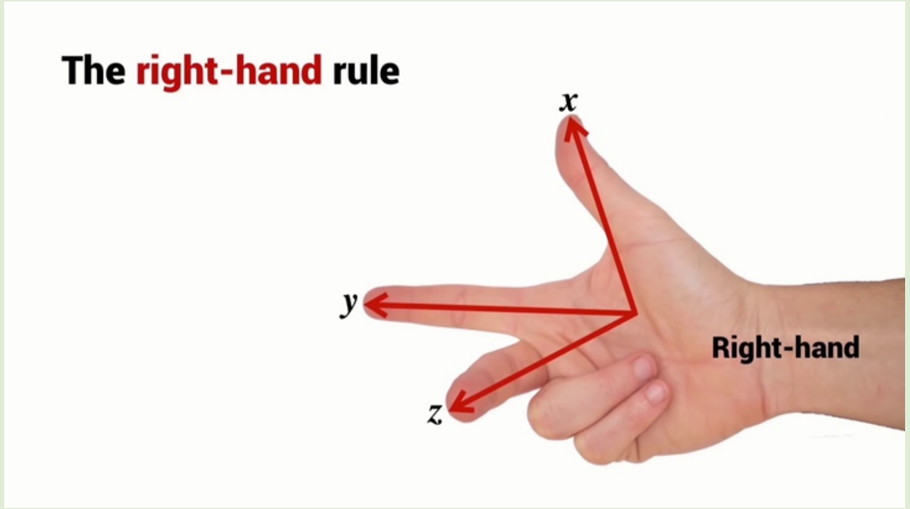
GOING 3D

OpenGL Fundamentals

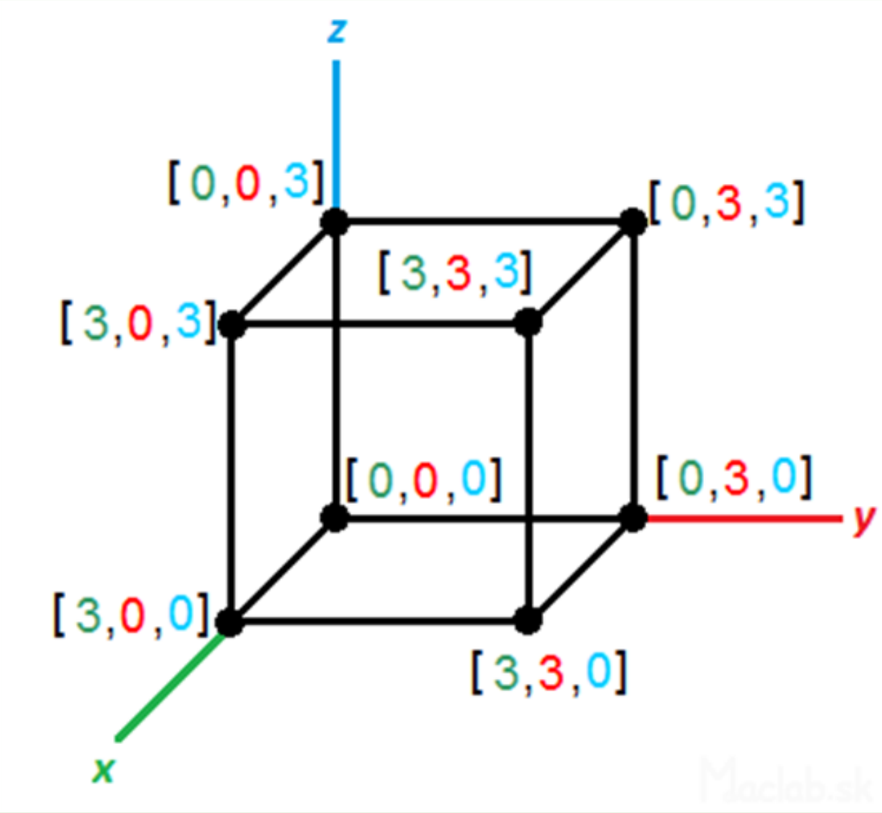
RIGHT-HANDED SYSTEM

- By convention, OpenGL is a right-handed system. What this basically says is that the positive x-axis is to your right, the positive y-axis is up and the positive z-axis is backwards. Think of your screen being the center of the 3 axes and the positive z-axis going through your screen towards you. The axes are drawn as follows:



USING 3 AXIS VERTICES

- Using glVertex3f(x, y , z)



FROM 2D TO 3D TRANSITION

- adding these paremeters and functions

```
//Initializes 3D rendering
void init()
{

    glClearColor(0.3,0.3f,0.3f,1.0f); //set background to black

}

int main(int argc,char**argv)
```

```
35
36  float _angle = -70.0f;
37
38  //Initializes 3D rendering
39  void initRendering() {
40
41      glClearColor(0.5f,0.0f,0.5f,1.0f); //set background to black
42      glEnable(GL_DEPTH_TEST);
43
44  }
```

```
int main(int argc,char**argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE); //display modes

    glutInitWindowPosition(200,100); //window postion
```

```
46
47  int main(int argc, char** argv) {
48      //Initialize GLUT
49      glutInit(&argc, argv);
50      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
51      glutInitWindowSize(400, 400);
52
```

```
void display()
{

    glClear(GL_COLOR_BUFFER_BIT);//clear color

    glLoadIdentity();//clear transforms

    glPushMatrix();
    glTranslatef(5.0f, 5.0, 0.0f); // move object from origin
    glRotatef(angle, 0.0f, 0.0f, 1.0f); //animated rotation
```

```
66
67  //Draws the 3D scene
68  void display() {
69      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
70
71      glMatrixMode(GL_MODELVIEW);
72      glLoadIdentity();
73
74      glTranslatef(0.0f, 0.0f, -15.0f); //move object in -z axis to seen in
75
76      //pyramid
```

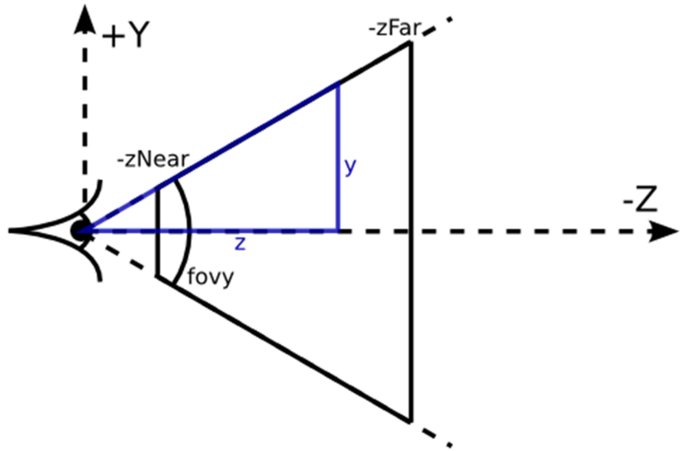
FROM 2D TO 3D TRANSITION

- Replace gluOrtho2D with gluPerpective

```
void reshape(int w, int h)
{
    glViewport(0,0,(GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-10,10,-10,10);//size of the world
    glMatrixMode(GL_MODELVIEW);
}
```

```
121  //Called when the window is resized
122  void reshape(int w, int h) {
123      glViewport(0, 0, w, h);
124      glMatrixMode(GL_PROJECTION);
125      glLoadIdentity();
126      gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
127  }
128
```

- gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect = 2.0 means the viewer's angle of view is twice as wide in x as it is in y. If the viewport is twice as wide as it is tall, it displays the image without distortion.



GLUPERSPECTIVE()

- **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).
- **zNear** - The distance from the viewer to the near clipping plane (always positive).
- **zFar** - The distance from the viewer to the far clipping plane (always positive).

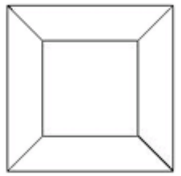
GLUT 3D MODELS

Two main categories

Wireframe Models and Solid Models

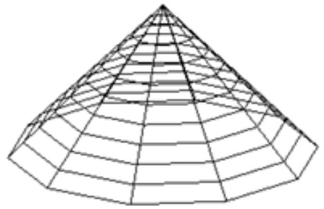
- Basic Shapes–Cube: glutWireCube( ), glutSolidCube()–Cone: glutWireCone( ), glutSolidCone( )–Sphere, Torus, Tetrahedron
- More advanced shapes–Octahedron, Dodecahedron, Icosahedron–Teapot (symbolic)

BASIC 3D GLUT OBJECTS

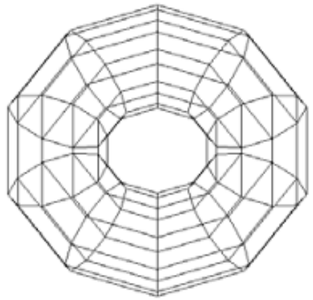


```
glutWireCube(1.0);
```

```
glutWireSphere(0.5,10,10);
```

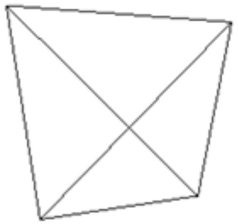


```
glutWireCone(1, 1, 10, 10);
```

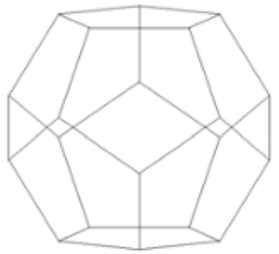


```
glutWireTorus(0.5,1.5,10,10)
```

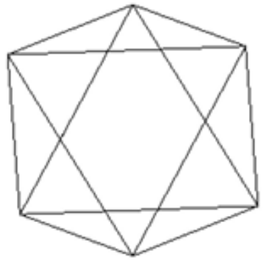
GLUT PLATONIC OBJECTS



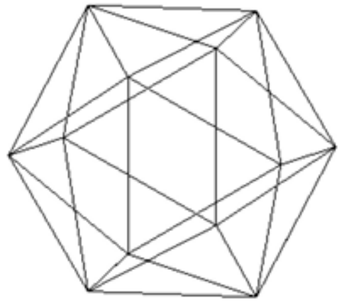
`glutWireTetrahedron();`



`glutWireDodecahedron();`

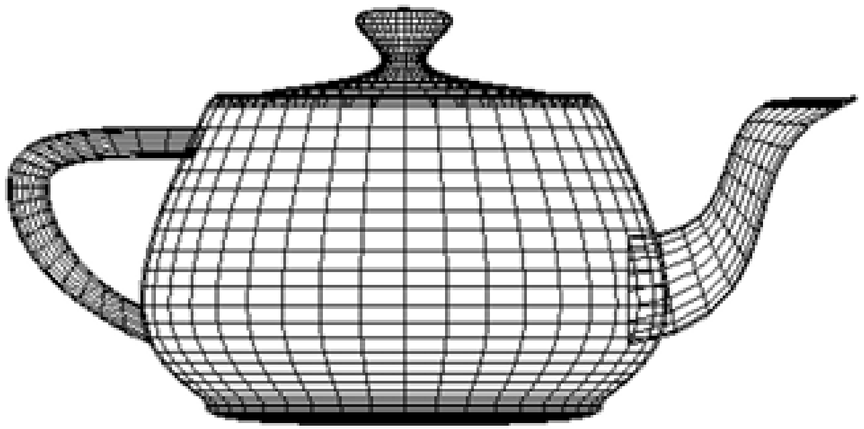


`glutWireOctahedron();`



`glutWireIcosahedron();`

GLUT OBJECT (SYMBOLIC



```
glutWireTeapot(1.0);
```

LEARNING LINKS

- https://www.opengl.org/resources/libraries/glut/spec3/node80.html
- http://www.cs.uccs.edu/~ssemwal/geometric.html