How to make a simple 3D in processing

Student ID: 111268 Name: Xin Wang

Advanced Computational Design

content

overview

P3D Box();

Sphere();

lights();

ambientLight();

directionalLight(): piontLight();

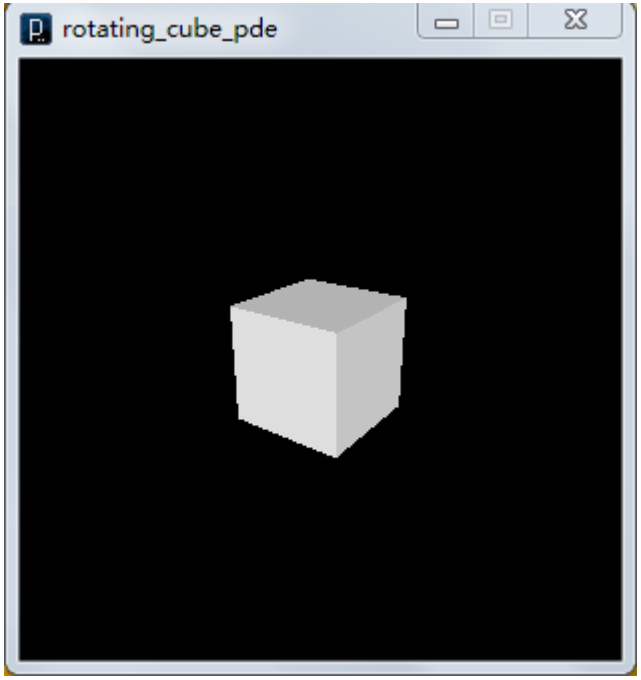
sportLight();

Camera(); perspective();

Vertex();

And so on

Make simple instance

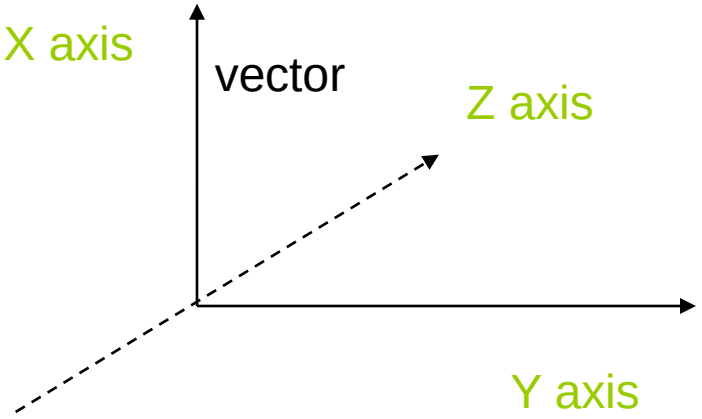


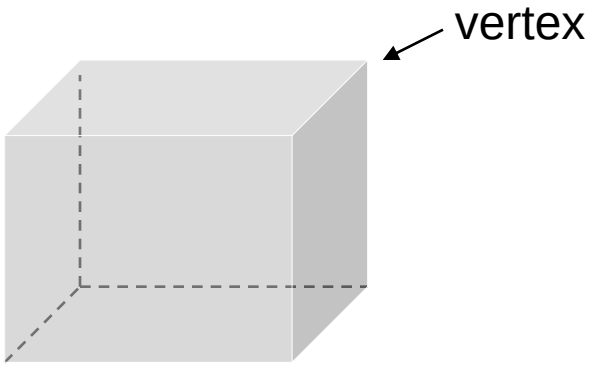
Understanding3D

There are a few important but simple concepts to understand about Three dimensional objects and three-dimensional space.

Any objects in 3D space will x ,y, and z coordinates. That is ,the object will be located at a 3D point

Vectors any objects in 3D space can also have a heading, which is direction that it is moving in or looking toward.





P3D

P3D (processing 3D)

This is a faster 3D renderer for the web that sacrifices rendering quickly for quick 3D drawing

OPENGL

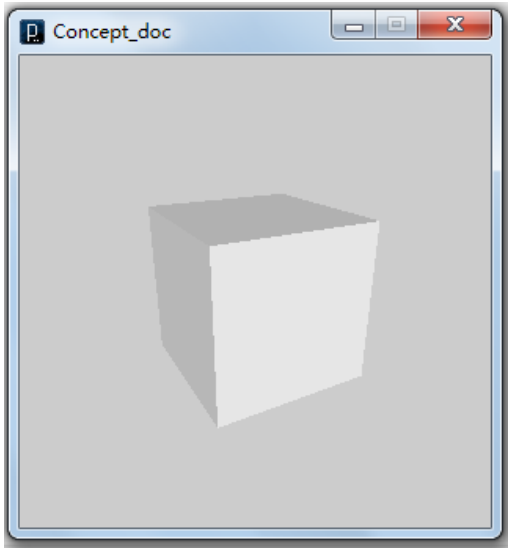
This is a high speed 3D renderer for that use OpenGL-compatible graphics hardware if available .

```
//import processing.opengl.*;
void setup() {
size(400, 400, P3D/ OPENGL); // set up the 3D renderer
}
void draw() {
noStroke(); // commenting this line out will show the lines in the box
//lights(); // uncommenting this will show the model with lights
fill(255);
translate(200, 200, 0);
box(100);
                                                    }
```

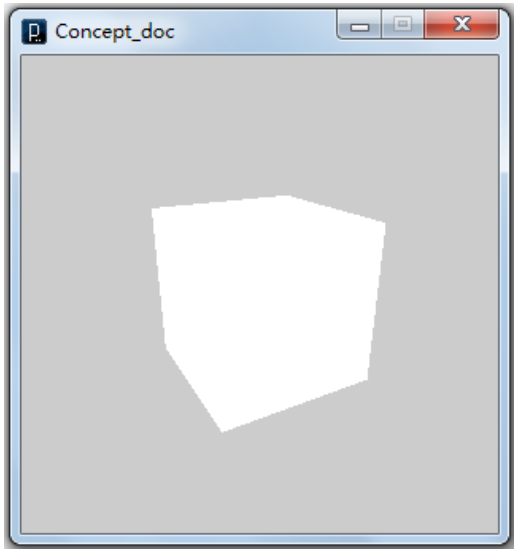
Lighting in processing

Lighting is important thing in 3D space because without lights positioned anywhere ,there is no way for the renderer to know which parts of the objects are darker and which parts are lighter. Lighting is one of the keys of representing three dimensionality ,without it, there is no way for a viewer to know what the three dimensional shape of an object is .

With lights



Without lights

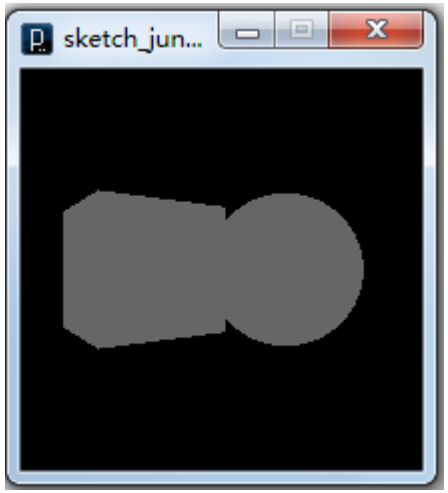


Different lights

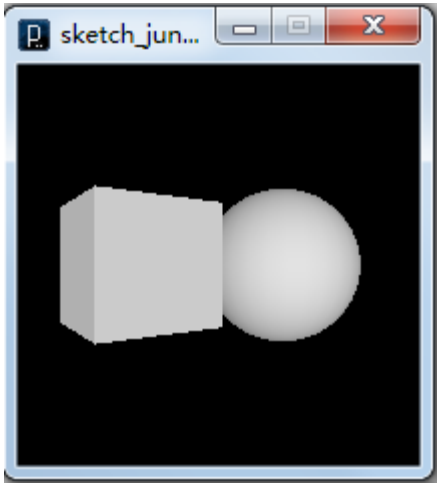
ambientLight(); directionalLight(): piontLight(); sportLight();

Ambient light doesn't come from specific direction the rays have light have bounced around so much that objects are evenly lit from all sides. Ambient lights are almost always used in combination with other types of lights.

ambientLight(102, 102, 102);



directionalLight(126, 126, 126, 0, 0, -1);
ambientLight(102, 102, 102);



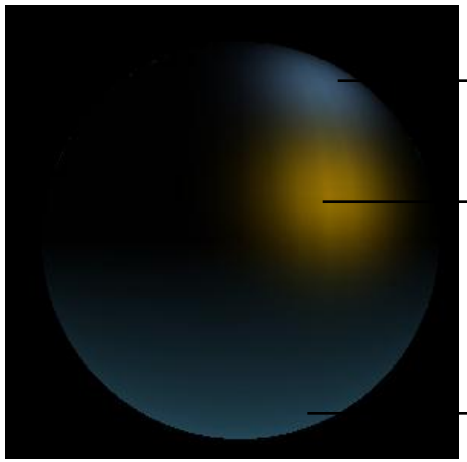
Lighting in processing

pointLight(v1, v2, v3, x, y, z) ;

The affect of the **v1**, **v2**, and **v3** parameters is determined by the current color mode. The **x**, **y**, and **z** parameters set the position of the light

spotLight(v1, v2, v3, x, y, z, nx, ny, nz, angle, concentration) ;

The **nx**, **ny**, **nz** specify the direction or light. The **angle** parameter affects angle of the spotlight cone. The concentration parameters how much brighter the light is at the center of the cone.



spotLight(102, 153, 204, 360, mouseY, 600, 0, 0, -1, PI/2, 600);

spotLight(204, 153, 0, 360, 160, 600, 0, 0, -1, PI/2, 600);

directionalLight(51, 102, 126, 0, -1, 0);

Camera

camera(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)

upX, upY, upZ the x,y,z of the camer relative to the word usually (0.0,1.0,0.0);

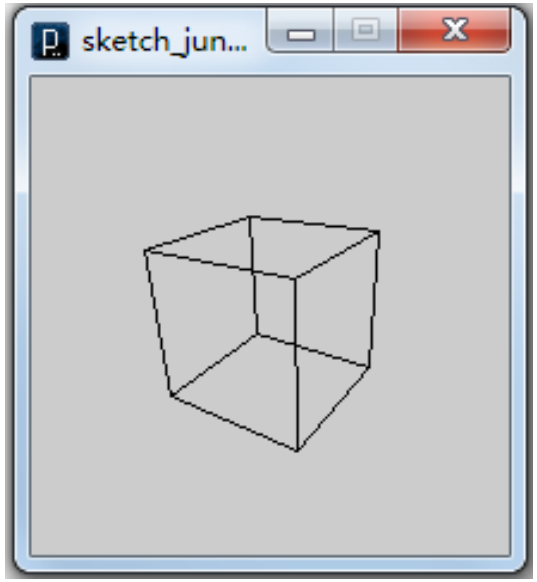
centerX, centerY, centerZ

the x ,y z coordinate for the center of the scene

eyeX, eyeY,eyeZ the position of camera

Camera

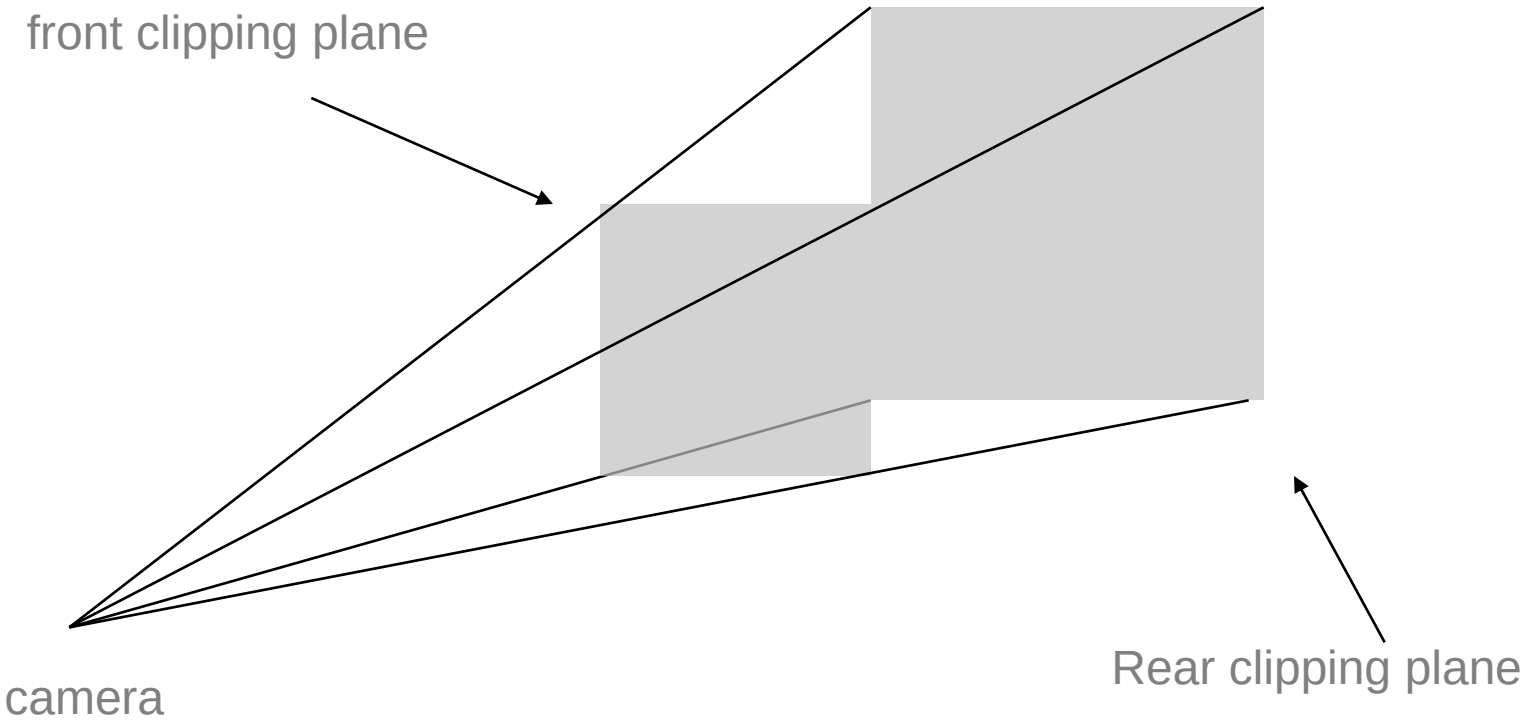
Sets the position of the camera through setting the eye position, the center of the scene, and which axis is facing upward. Moving the eye position and the direction it is pointing (the center of the scene) allows the images to be seen from different angles.




```
size(200, 200, P3D);
noFill();
background(204);
camera(60.0, 60.0, 120.0, 50.0, 50.0,
0.0,
       0.0, 1.0, 0.0);
translate(50, 50, 0);
rotateX(-PI/6);
rotateY(PI/3);
box(45);
```

Frustum()

Objects between the front clipping and the rear clipping plane will be displayed



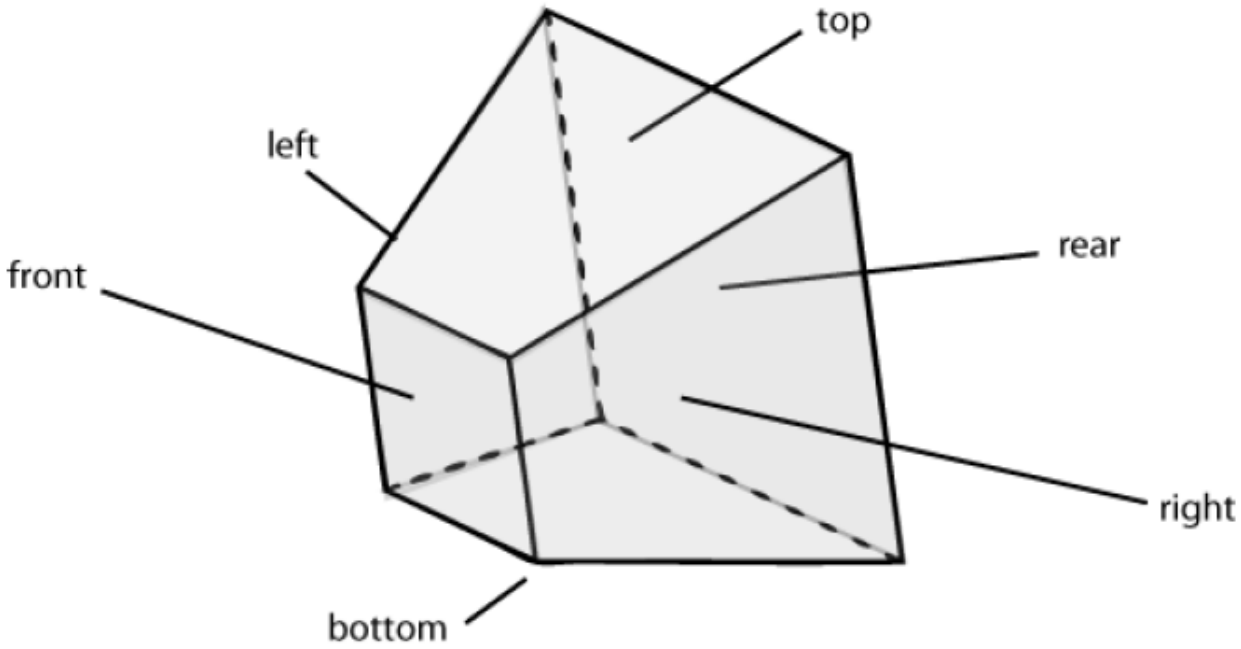
Frustum()

frustum (left, right, bottom, top, near, far) ;

Left, right, bottom ,top float: different face component of the clipping plane

Near float: near component of the clipping plane

Far float: far component of the clipping plane



perspective() VS orthographic ()

perspective()

perspective(fov, aspect, zNear, zFar)// default

The default values are: perspective(PI/3.0, width/height, cameraZ/10.0, cameraZ*10.0)

Perspective (fov, aspect, zNear, zFar)

Fov float: field-of-view angle (in radians) for vertical direction **aspect** float: ratio of width to height **zNear** float: z-position of nearest clipping plane **zFar** float: z-position of farthest clipping plane

ortho()

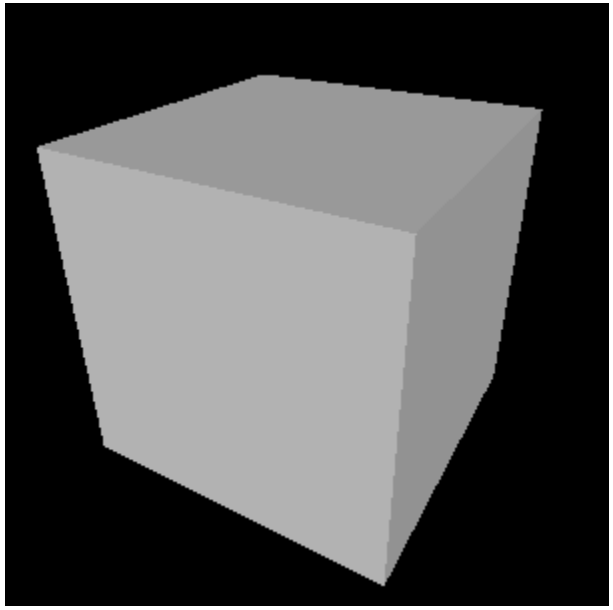
ortho(left, right, bottom, top, near, far)

the default is used: ortho(0, width, 0, height, -10, 10).

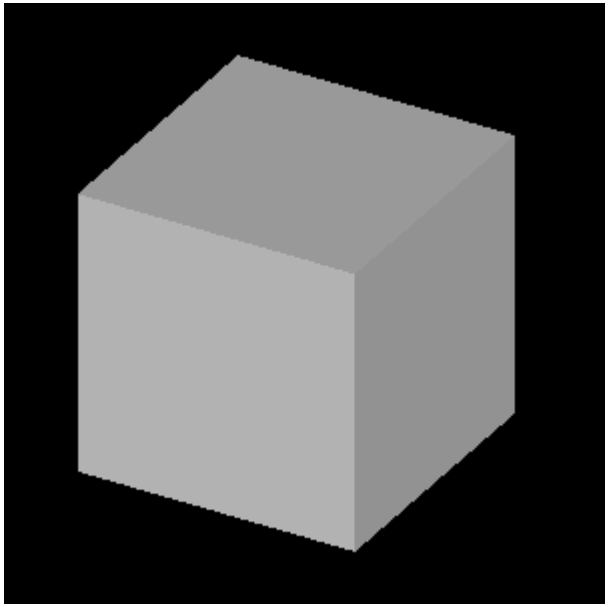
perspective() VS orthographic ()

We can use mousepressed method to see difference between orthographic projection and perspective projection

Using a perspective view



Using an orthographic view



perspective() VS orthographic ()

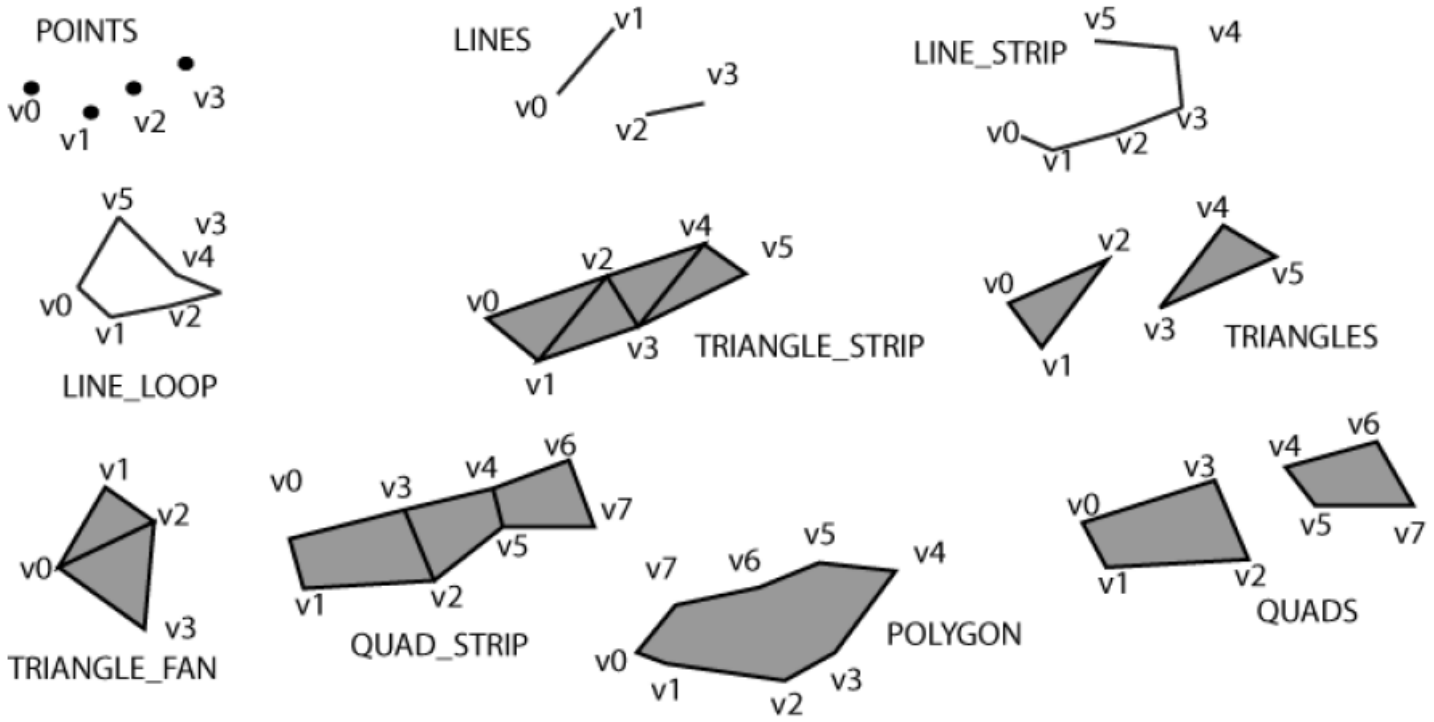
```
void setup()
{
  size(640, 360, P3D);
  noStroke();
  fill(204);
}
void draw()
{ background(0);
  lights();
   translate(width/2, height/2, 0);
  if(mousePressed) {
perspective();//default
  } else {
    ortho(-width/2, width/2, -height/2, height/2, -10, 10);
    //ortho();
  }
   rotateX(-PI/6);
  rotateY(PI/3);
  box(160);
}
```

We cam use perspective(); method in stead of following code

```
//float fov = PI/3.0;
// float cameraZ = (height/2.0) /
tan(PI * fov / 360.0);
 // perspective(fov,
float(width)/float(height),
          //  cameraZ/2.0,
cameraZ*2.0);
```

Vertex

All shapes are constructed by connecting a series of vertices. **vertex()** is used to specify the vertex coordinates for points, lines, triangles, quads, and polygons and is used exclusively within the **beginShape()** and **endShape()** function.



Vertex (3D)

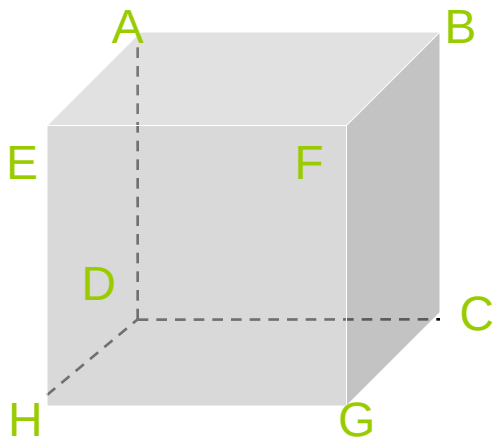
A vertex is a point in 3D space so you can use vertex(x, y, z); and beginShape(QUADS); methods endShape(); to make s simple 3D

A(-1, 1, 1);

E(-1, 1, -1);

D(-1, -1, 1);

H(-1, -1, -1);



B(1, 1, 1);

F(1, 1, -1);

C(1, -1, 1);

G(1, -1, -1);

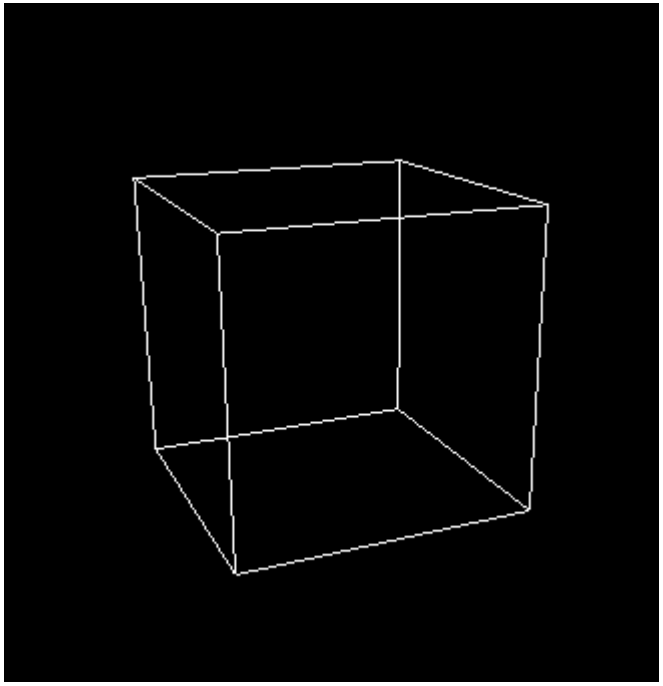
Rear right front left top bottom

```
void setup()
{
  size(640, 360, P3D);
  noStroke();
 // colorMode(RGB, 1);
}

void draw()
{
  background(0.5);
  lights();
  pushMatrix();
  translate(width/2, height/2, -30);
   rotateX(-PI/6);
  rotateY(PI/3);

  scale(90);
  beginShape(QUADS);
```

```
 ABCD
vertex(-1,  1,  1);
 vertex( 1,  1,  1);
 vertex( 1, -1,  1);
 vertex(-1, -1,  1);
 BFGC
 vertex( 1,  1,  1);
 vertex( 1,  1, -1);
 vertex( 1, -1, -1);
 vertex( 1, -1,  1);
  FEHG
 vertex( 1,  1, -1);
 vertex(-1,  1, -1);
 vertex(-1, -1, -1);
 vertex( 1, -1, -1);
  EADH
 vertex(-1,  1, -1);
 vertex(-1,  1,  1);
 vertex(-1, -1,  1);
vertex(-1, -1, -1);
```

```
  EFBA
vertex(-1,  1, -1);
vertex( 1,  1, -1);
vertex( 1,  1,  1);
vertex(-1,  1,  1);
  HGCD
vertex(-1, -1, -1);
vertex( 1, -1, -1);
vertex( 1, -1,  1);
vertex(-1, -1,  1);
    endShape();
     popMatrix();
          }
```

Make a simple 3D in processing



Thanks