PPEM 2022

3D

3D

Coordenadas y transformaciones en 3D

Formas geométricas

Texturas

Iluminación

Visión de la escena

Texto

Coordenadas y transformaciones

Coordenadas y transformaciones en 3D

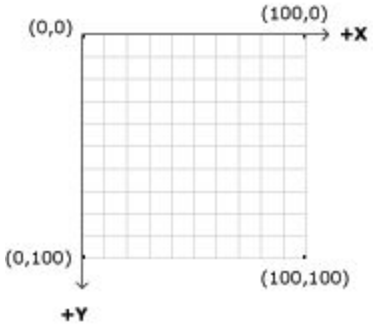
translate(x,y,z)

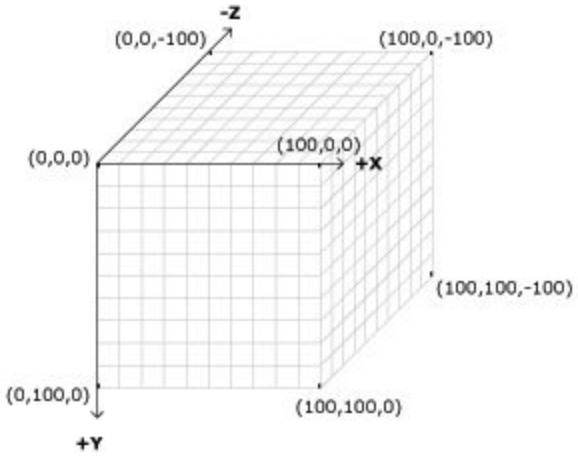
rotateX

rotateY

rotateZ

scale

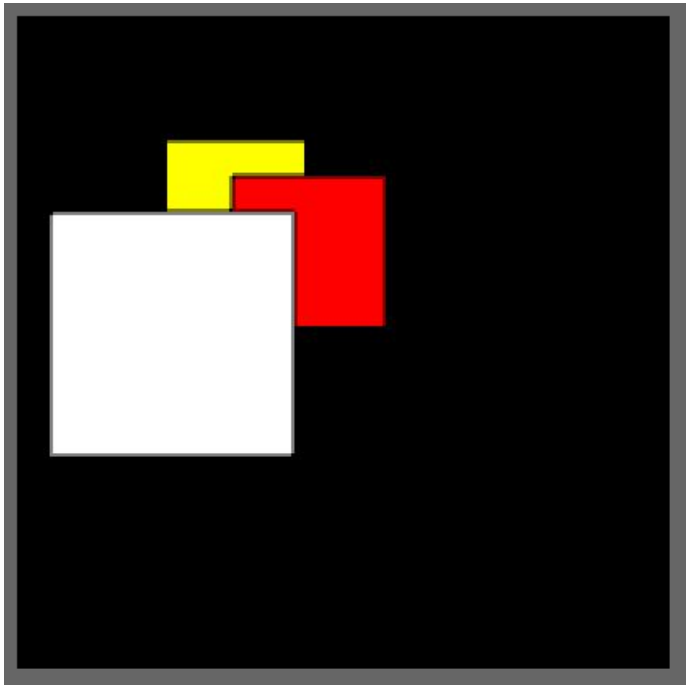




translate

```
translate(x, y, z)
```

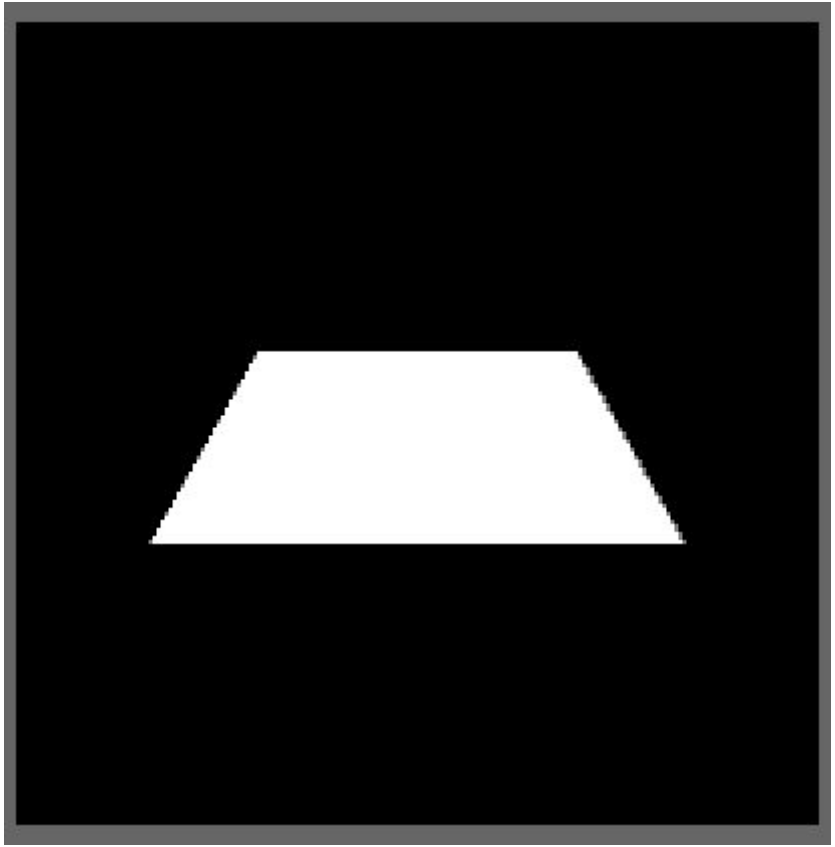
```
size(200, 200, P3D);
background(0);
fill(255); // blanco
rect(10, 60, 75, 75);

translate(30, 20, -50);
fill(255,255,0); // amarillo
rect(0, 0, 55, 55);

translate(30, 20, 20); // -50 + 20 = -30
fill(255,0,0); // rojo
rect(0, 0, 55, 55);
```



rotateX

`rotateX(angle)`

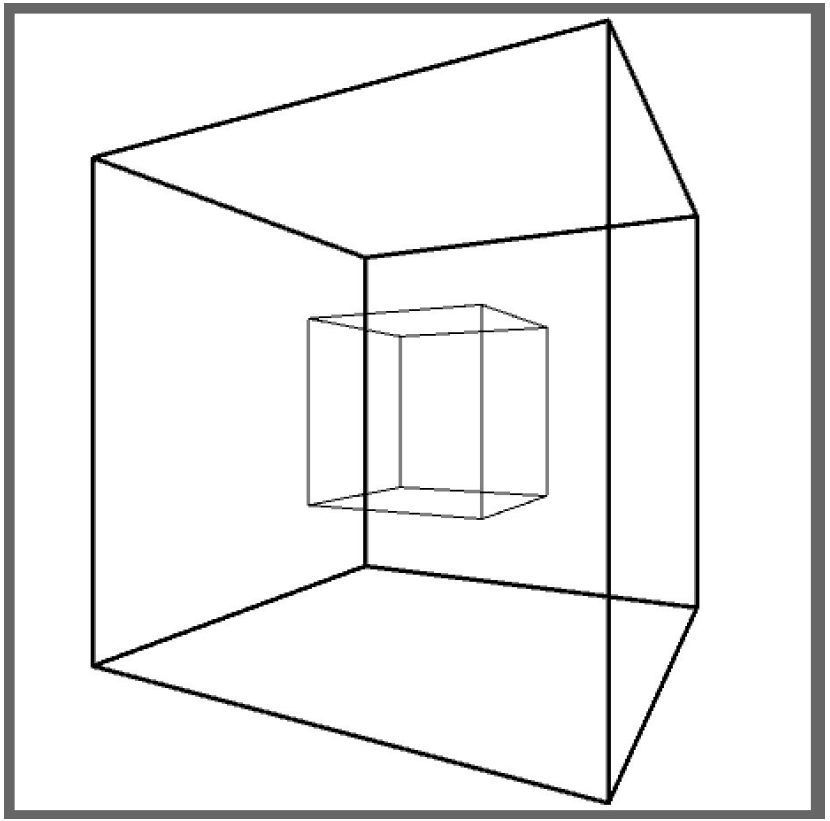
```
float x;
void setup() {
  size(200,200,P3D);
  x = 0;
  rectMode(CENTER);
}
void draw() {
  background(0);
  x=map(mouseX,0,width,0,TWO_PI);
  translate(width/2,height/2);
  rotateX(x);
  rect(0,0,100,100);
}
```



scale

```
scale(x, y, z)
```

```
void setup(){
  size(450, 450, P3D);
  noFill();
}
void draw(){
  background(255);
  translate(width/2, height/2);
  rotateY(map(mouseX,0,width,0,TWO_PI));
  box(100, 100, 100);
  scale(2.5, 2.5, 2.5);
  box(100, 100, 100);
}
```



Formas geométricas

Formas geométricas

box

sphere

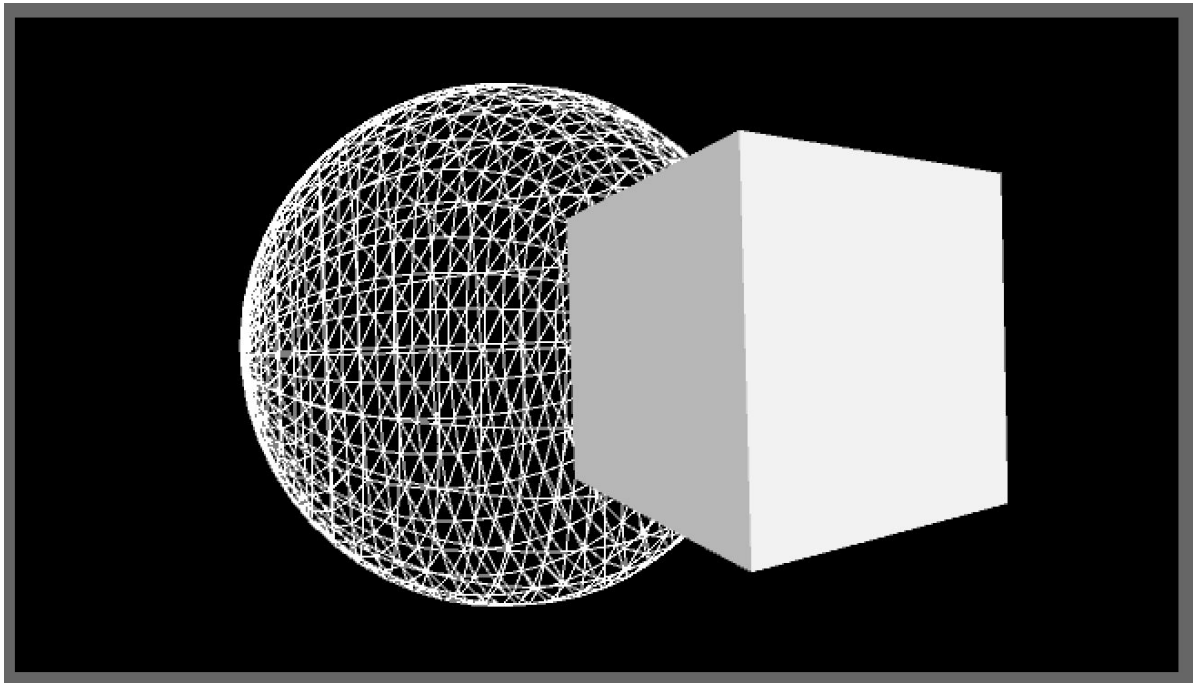
PShape (sphere y box)

vertex()

box y shpere

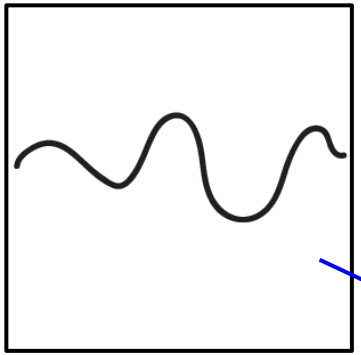
```
void setup(){
  size(640,360,P3D);
}
void draw(){
  lights();
  background(0);
  translate(width/2,height/2);
  rotateY(map(mouseX,0,width,0,TWO_PI));
  rotateX(map(mouseY,0,height,0,TWO_PI));
  pushMatrix();
    translate(-130, 0);
    noStroke();
    fill(255);
    box(100); // igual a box (100,100,100)
  popMatrix();
  pushMatrix();
    translate(130, 0);
    noFill();
    stroke(255);
    sphere(180);
  popMatrix();
}
```

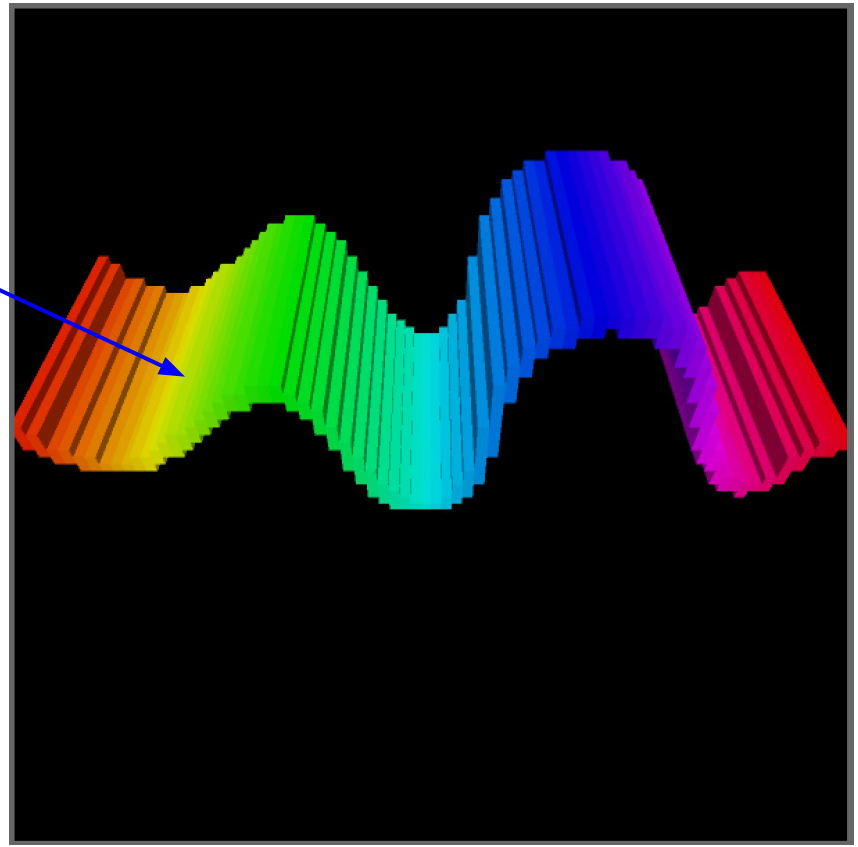
radio



box

```
PImage img;
ArrayList<PVector> points;
void setup(){
  size(600, 600, P3D);
  img=loadImage("line.png");
  noStroke();
  points = new ArrayList<PVector>();
  for(int i=0;i<img.width;i+=4){
    for(int j=0;j<img.height;j+=4){
      if(brightness(img.get(i,j))<40){ //negro
        points.add(new PVector(i,j));
      }
    }
  }
  colorMode(HSB,img.width,100,100);
}
void draw(){
 background(0);
 lights();
 translate(150,height/2,200);
 rotateX(map(mouseX,0,width,0,TWO_PI));
 for(int i=0;i<points.size();i++){
   fill(points.get(i).x,100,100);
   pushMatrix();
   translate(points.get(i).x, points.get(i).y);
   box(4,4,100);
   popMatrix();
 }
}
```

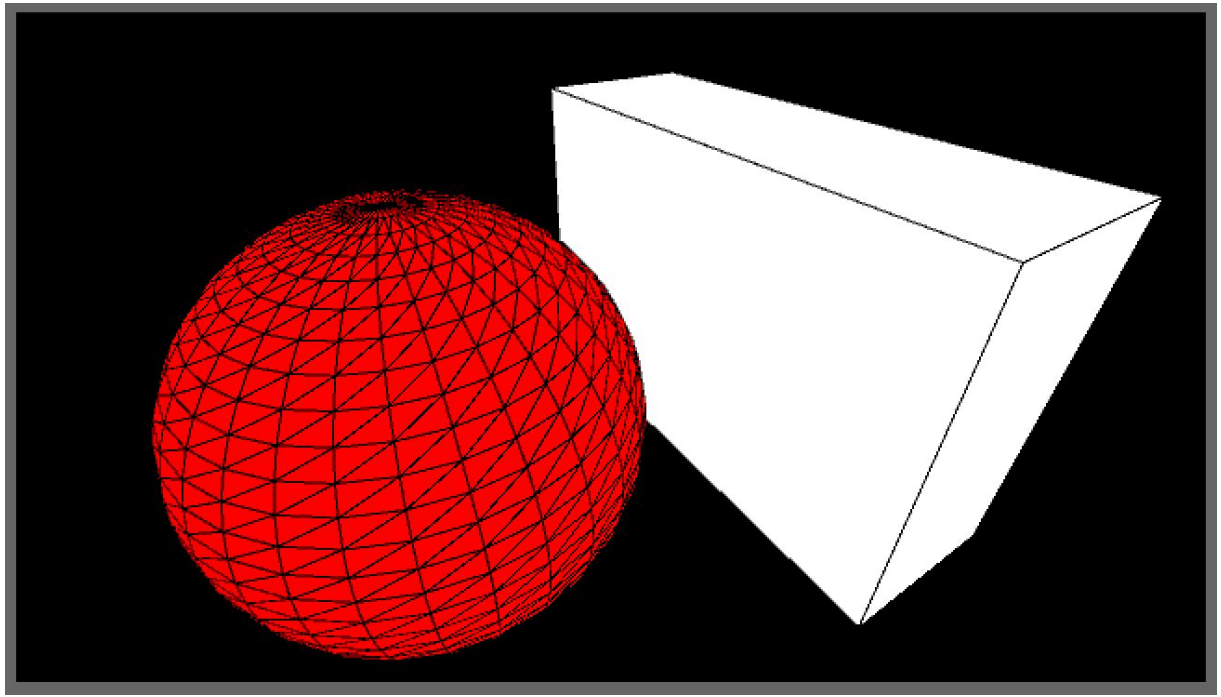




PShape

`createShape(kind, p)`

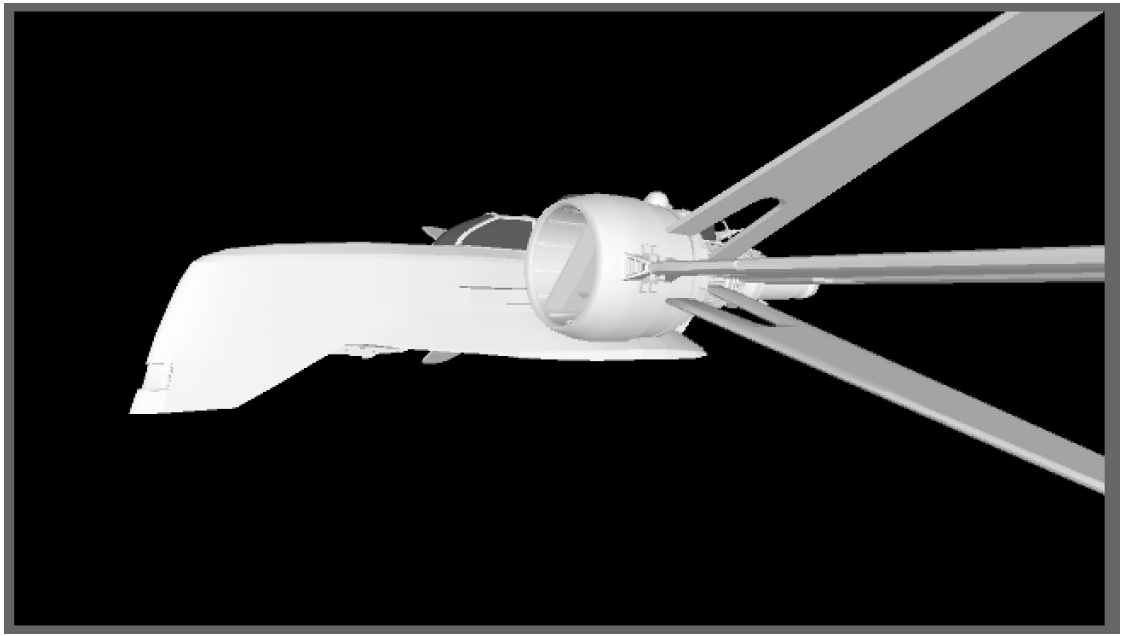
```
float rotx = 0;
float roty = 0;
PShape box,sphere;
void setup() {
  size(640, 360, P3D);
  box = createShape(BOX,100,200,300);
  sphere= createShape(SPHERE,100);
  color r = color(255,0,0);
  sphere.setFill(r);
}
void draw() {
  background(0);
  translate(width/2.0, height/2.0);
  rotateX(rotx);
  rotateY(roty);
  translate(-100,0);
  shape(box);
  translate(200,0);
  shape(sphere);
}
void mouseDragged() {
  rotx += (pmouseY-mouseY) * 0.01;
  roty += (mouseX-pmouseX) * 0.01;
}
```



Cargar un modelo 3D como PShape

`loadShape(filename)`

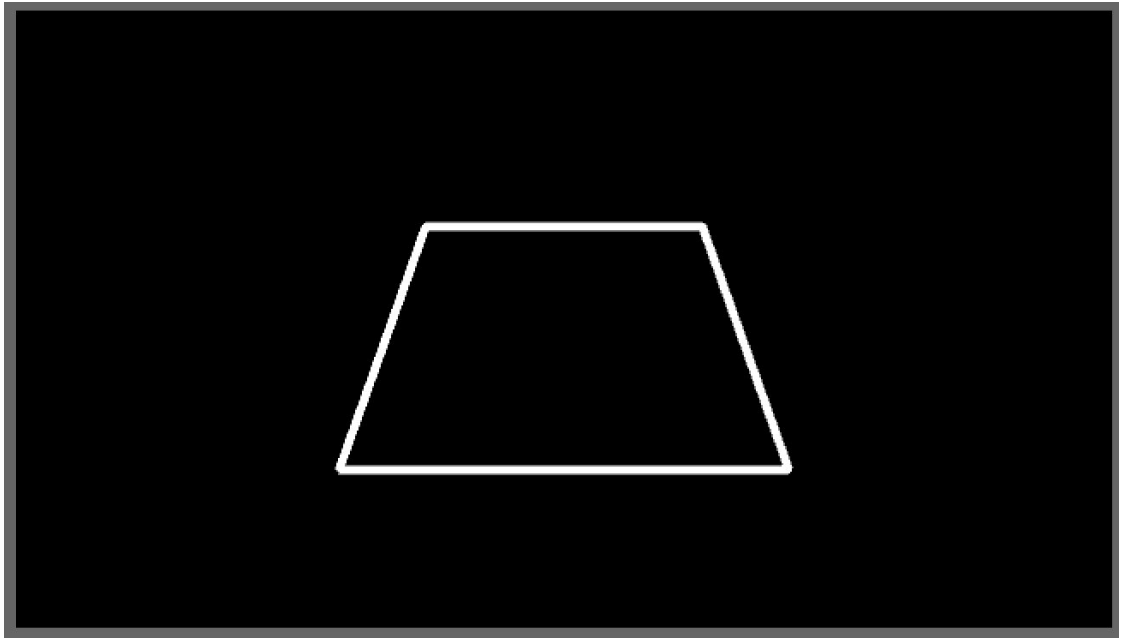
```
PShape arc170;
float ry;
public void setup() {
  size(640, 360, P3D);
  arc170 = loadShape("Arc170.obj");
}
public void draw() {
  background(0);
  lights();
  translate(width/2, height/2+100, -400);
  rotateZ(PI);
  rotateY(ry);
  shape(arc170);
  ry += 0.02;
}
```



Vertex

`vertex(x, y, z)`

```
void setup(){
  size(640, 360, P3D);
  noFill();
  stroke(255);
  strokeWeight(5);
}
void draw(){
  background(0);
  translate(width/2, height/2, 0);
  rotateX(radians(frameCount%360));
  beginShape();
  vertex(-100, -100, 0);
  vertex(100, -100, 0);
  vertex( 100, 100, 0);
  vertex( -100,  100, 0);
  endShape(CLOSE);
}
```



Texturas

Texturas de imagen

```
vertex(x, y, z, u, v)
```

```
PImage img;
void setup(){
  size(640, 360, P3D);
  noStroke();
  img=loadImage("we.jpg");
}
void draw(){
  background(0);
  translate(width/2, height/2, 0);
  rotateX(radians(frameCount%360));
  beginShape();
  texture(img);
  vertex(-100, -100, 0,0,0 );
  vertex(100, -100, 0,img.width,0);
  vertex( 100, 100, 0,img.width,img.height);
  vertex( -100,  100, 0,0,img.height);
  endShape(CLOSE);
}
```