PPEM 2022

3D

3D

Coordenadas y transformaciones en 3D

Formas geométricas

Texturas

Iluminación

Visión de la escena

Texto

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);
}
```

Texturas de imagen // textureWrap

```
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));
 if (mousePressed) {
  textureWrap(REPEAT);  //repetir imagen
 } else {
  textureWrap(CLAMP); // repetir últimos pixeles
 }
 beginShape();
 texture(img);
 vertex(-100, -100, 0,0,0 );
 vertex(100, -100, 0,img.width*2,0);
 vertex( 100, 100, 0,img.width*2,img.height*2);
 vertex( -100,  100, 0,0,img.height*2);
 endShape(CLOSE);
}
```

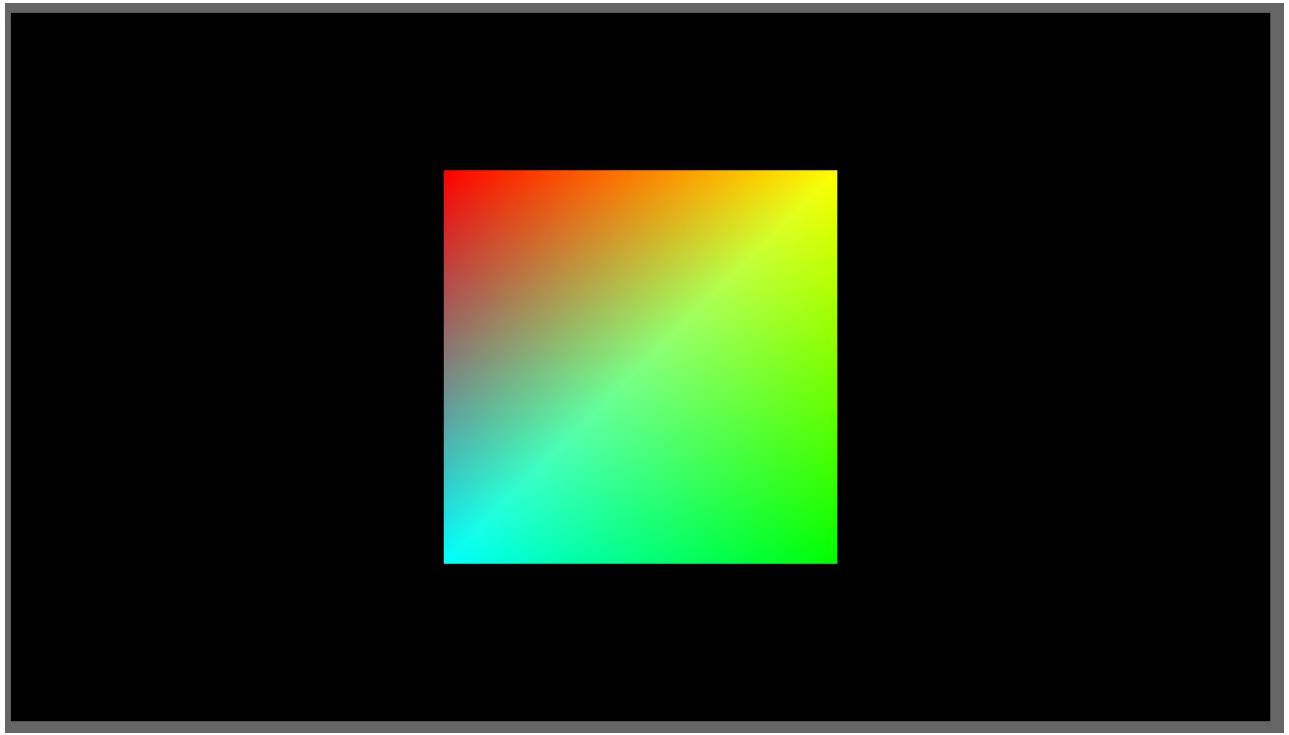
Texturas y PShape

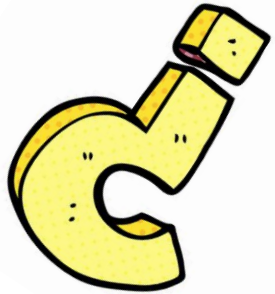
```
float rotx = PI/4;
float roty = PI/4;
PShape b;
void setup() {
  size(640, 360, P3D);
  PImage tex = loadImage("we.jpg");
  b = createShape(BOX,100,200,300);
  b.setTexture(tex);
}
void draw() {
  background(0);
  translate(width/2.0, height/2.0, -20);
  rotateX(rotx);
  rotateY(roty);
  scale(-1);
  shape(b);
}
void mouseDragged() {
  rotx += (pmouseY-mouseY) * 0.01;
  roty += (mouseX-pmouseX) * 0.01;
}
```

Textura de color

`vertex(x, y, z)`

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



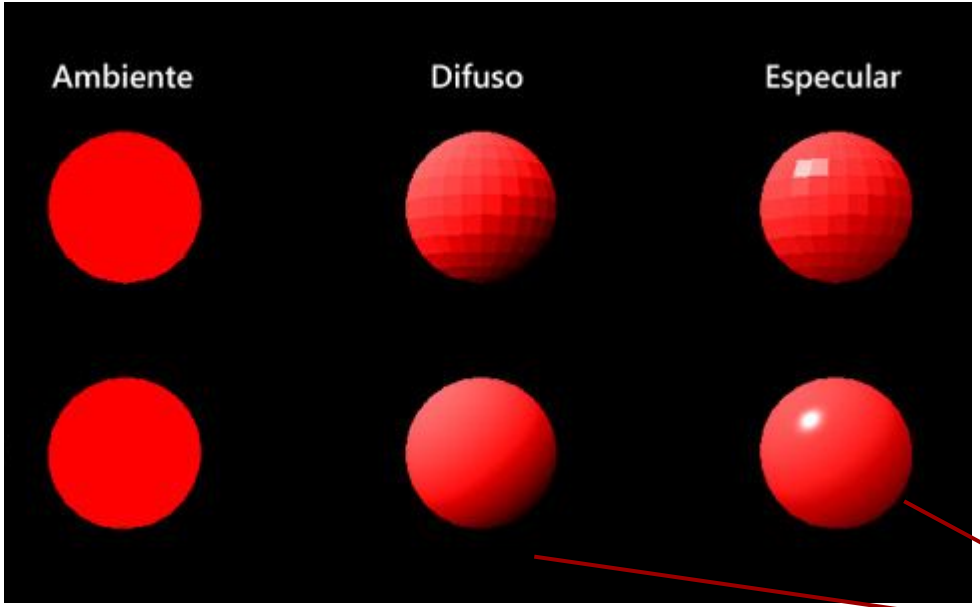


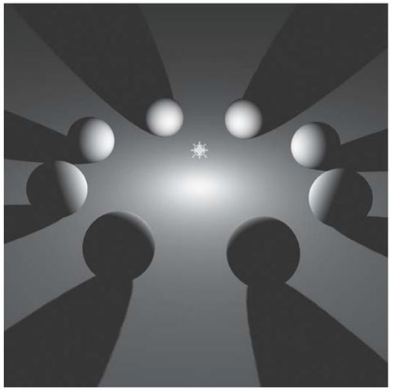
¿Dudas?

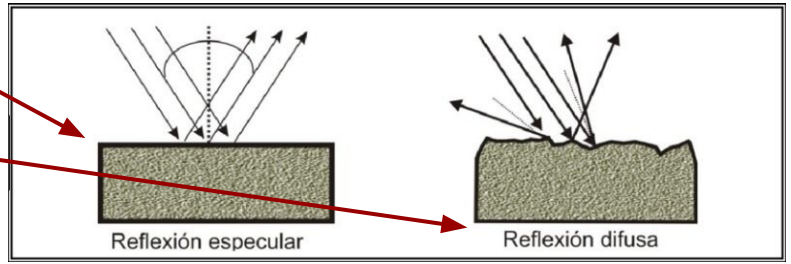
Iluminación

Iluminación de la escena 3D

ambientLight directionalLight lightSpecular spotLight pointLight







Iluminación de la escena 3D

ambientLight(v1, v2, v3)

Color plano que “tinie” de todos lados los objetos en la escena. No tiene dirección. Normalmente combinada con otros tipos de luces.

directionalLight(v1, v2, v3, nx, ny, nz)

Viene de una dirección. Después de rebotar en un objeto se dispersa en todas las direcciones. -> nx, ny, nz van de -1 a 1 e indican la dirección hacia donde va la luz. V1,v2 y v3 indican el color de la luz (RGB o HSB).

spotLight(v1, v2, v3, x, y, z, nx, ny, nz, ángulo, concentración)

Foco de luz . V1,v2 y v3 indican el color de la luz (RGB o HSB). X, y y z la posición del foco. Nx, ny, nz indican hacia donde apunta la luz. Ángulo define el ángulo del cono de la luz. Concentración define la concentración de luz enfocando hacia el centro del cono.

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

Emite luz en todos los sentidos. X, Y y Z posicionan la luz en la escena. Es como spotLight con un cono de 180 grados.

Iluminación de la escena 3D

lightSpecular(v1, v2, v3)

Rebota en una dirección, no en todas. Se utiliza para generar resaltados. Uno define el color de la luz especular que se emite con directionalLight, spotLight y pointLight.

lightFalloff(constant, linear, quadratic)

Establece las tasas de caída de pointLight, spotLight y ambientLight.

d = distancia desde la fuente de la luz hasta la posición
falloff = 1 / (CONSTANT + d * LINEAR + (d*d) * QUADRATIC)

lights() = ambientLight(128, 128, 128) and directionalLight(128, 128, 128, 0, 0, -1), lightFalloff(1, 0, 0), y lightSpecular(0, 0, 0).

Propiedades de materiales

Shininess: Establece la cantidad de brillo en la superficie de las formas.

Emissive: Establece el color emisivo del material.

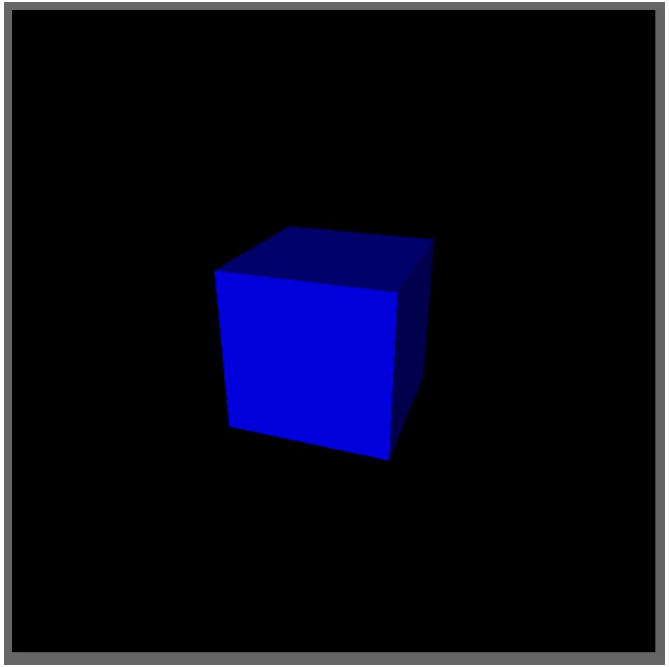
Ambient: Define la reflectancia del ambiente para las formas dibujadas en la pantalla. Esto se combina con el componente de luz ambiental del entorno (ambientLight).

Specular: Establece el color especular de los materiales. Especular se refiere a la luz que rebote de una superficie en una dirección preferida. Se combina con lightSpecular().

directionalLight

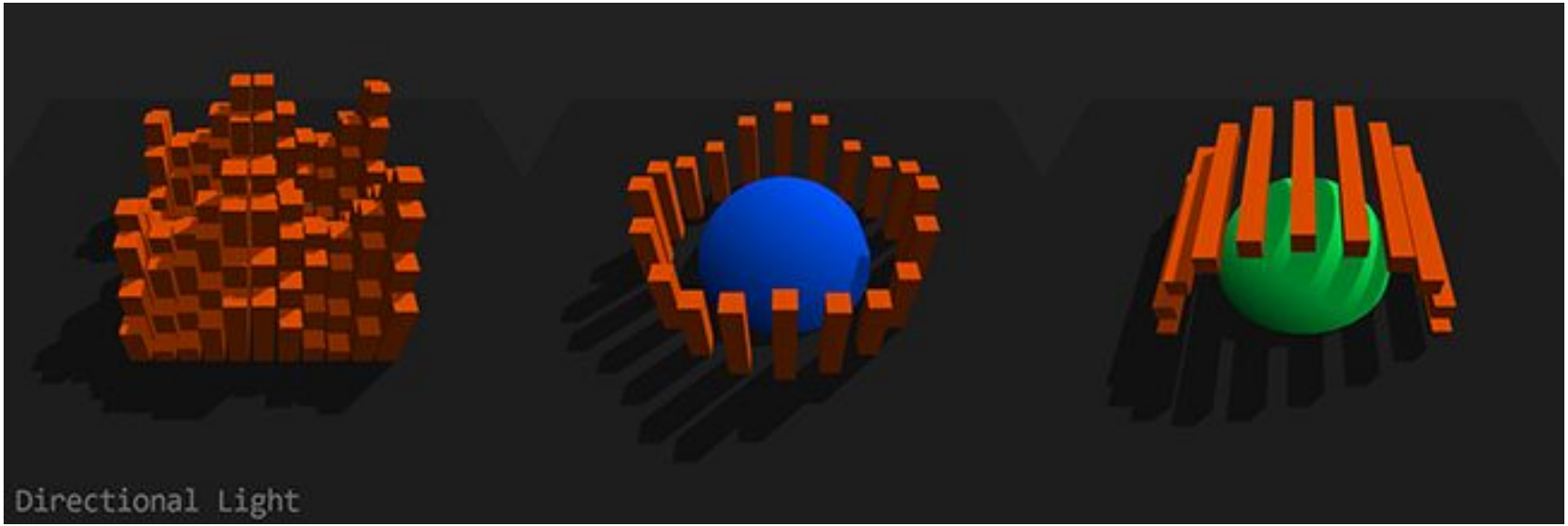
```
directionalLight(v1, v2, v3, nx, ny, nz)
```

```
float rotx = PI/4;
float roty = PI/4;

void setup() {
 size(400, 400, P3D);
 noStroke();
}
void draw() {
 background(0);
  if (mousePressed) {
   directionalLight(0,0,255,0,0,-1); // -1 =  viene desde adelante hacia atrás (eje z)
 }
 translate(width/2.0, height/2.0, 0);
 rotateX(rotx);
 rotateY(roty);
 box(100);
}
void mouseDragged() {
 float rate = 0.01;
 rotx += (pmouseY-mouseY) * rate;
 roty += (mouseX-pmouseX) * rate;
}
```



directionalLight

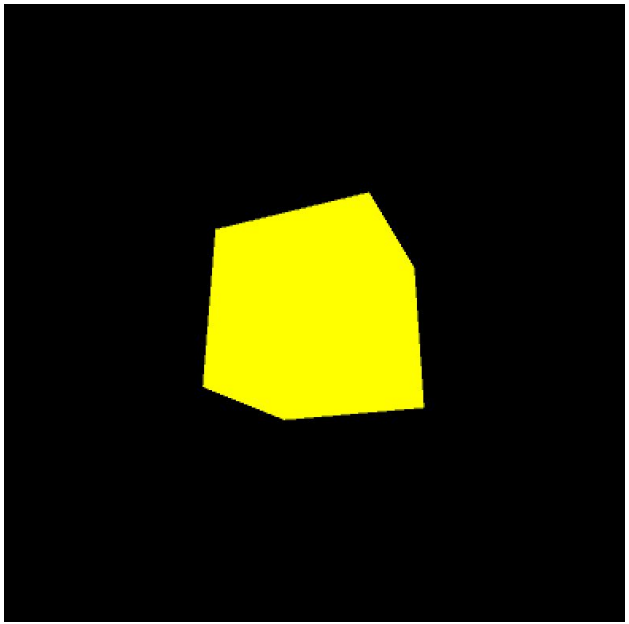
https://forum.processing.org/two/discussion/12775/simple-shadow-mapping



ambientLight

```
ambientLight(v1, v2, v3)
```

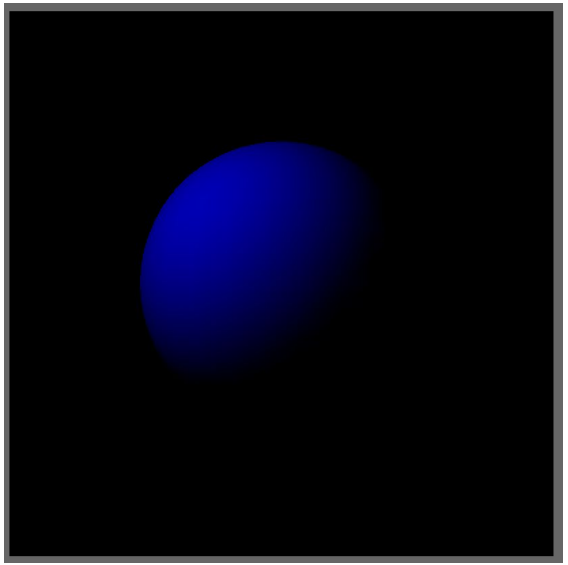
```
float rotx = PI/4;
float roty = PI/4;

void setup() {
 size(400, 400, P3D);
 noStroke();
}
void draw() {
 background(0);
 ambientLight(255,255,0); // luz ambiente amarilla
  if (mousePressed) {
   directionalLight(0,0,255,0,0,-1); // -1 =  viene desde adelante hacia atrás (eje z)
 }
 translate(width/2.0, height/2.0, 0);
 rotateX(rotx);
 rotateY(roty);
 box(100);
}
void mouseDragged() {
 float rate = 0.01;
 rotx += (pmouseY-mouseY) * rate;
 roty += (mouseX-pmouseX) * rate;
}
```



spotLight

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

```
void setup() {
  size(400, 400, P3D);
  noStroke();
}
void draw() {
  background(0);
  translate(width/2.0, height/2.0, 0);
  spotLight(0, 0, 255, map(mouseX,0,width,-200,200),
map(mouseY,0,height,-200,200), 200, 0, 0, -1, PI/2, 2);
  sphere(100);
}
```

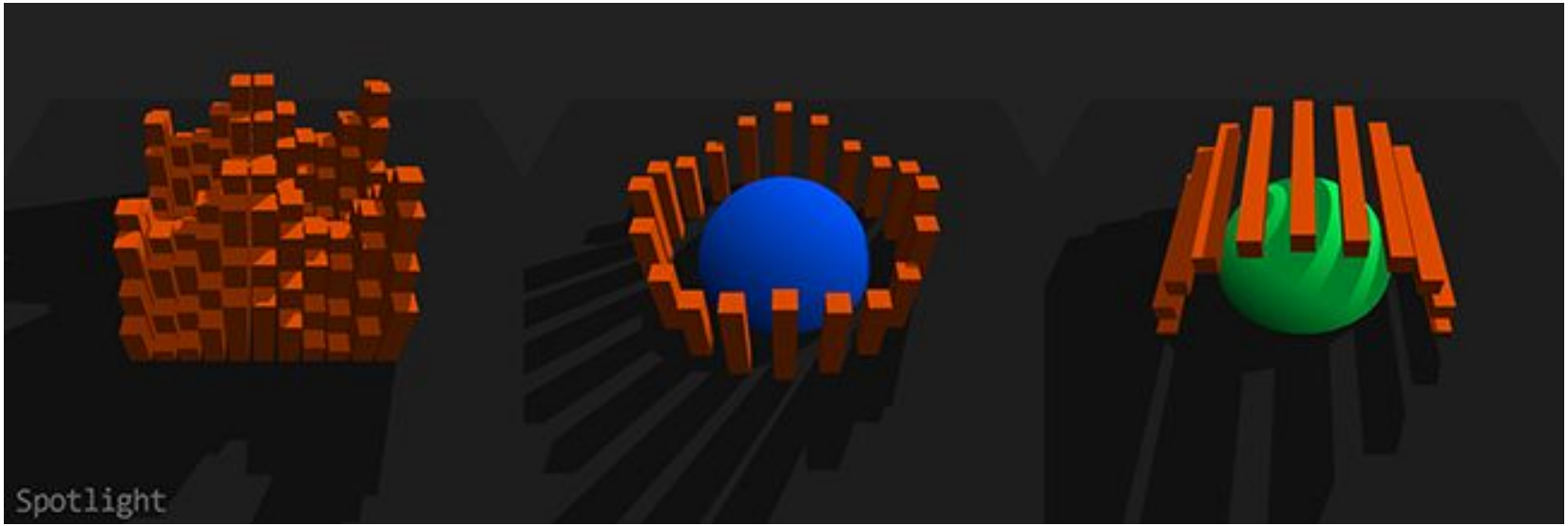


Cono de 90 grados

Poca concentración hacia el centro

spotLight

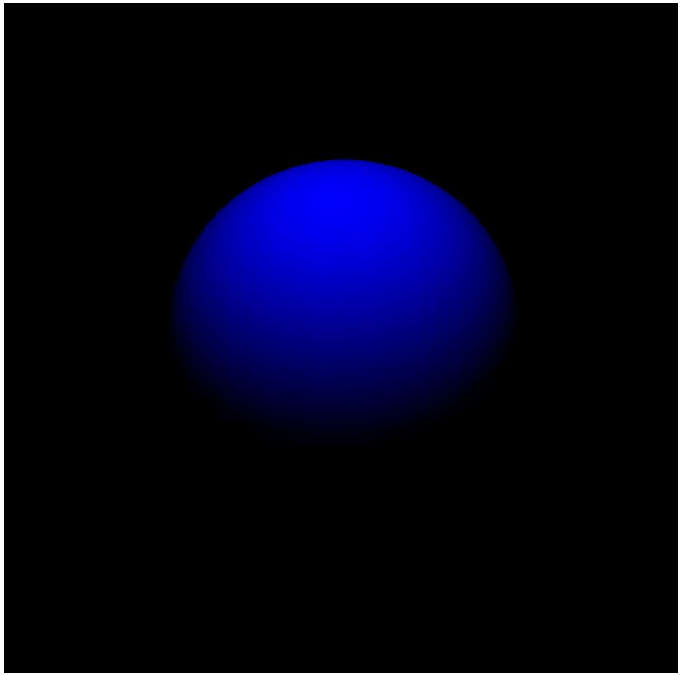
https://forum.processing.org/two/discussion/12775/simple-shadow-mapping



pointLight

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

```
void setup() {
  size(400, 400, P3D);
  noStroke();
}
void draw() {
  background(0);
  translate(width/2.0, height/2.0, 0);
  pointLight(0, 0, 255, map(mouseX,0,width,-200,200),
map(mouseY,0,height,-200,200), 200);
  sphere(100);
}
```

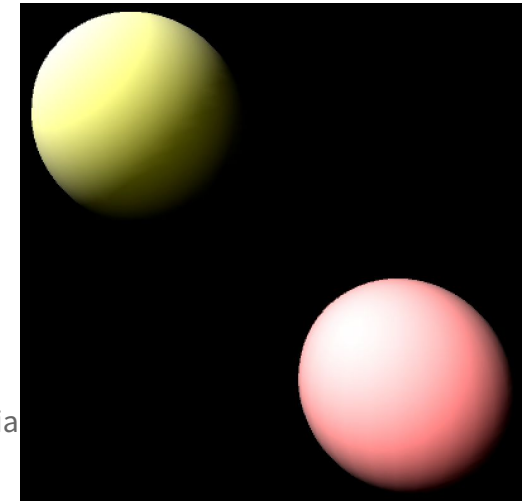


lightSpecular

```
lightSpecular(v1, v2, v3)
```

```
specular(rgb)
specular(gray)
specular(v1, v2, v3)
```

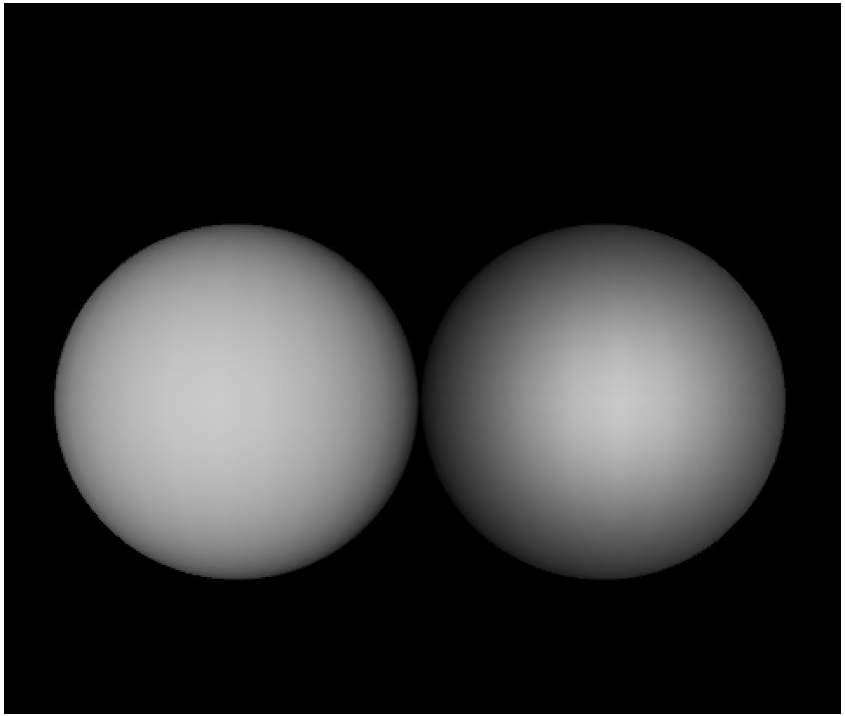
```
void setup() {
 size(400, 400, P3D);
 noStroke();
 fill(255);
}
void draw() {
 background(0);
 translate(width / 4, height / 4);
 lightSpecular(255, 255, 0); // color especular emitido con la luz (amarillo)
 directionalLight(255, 255, 255, 1, 1, -1); // luz blanca direccional hacia izquierda, arriba y hacia
 sphere(80);  // el material no tiene color especular
 translate(width / 2, height / 2);
 float s = map(mouseX,0,width,0,255) ;
 pushStyle();
  specular(s, 0, 0); // color especular del material (rojo)
  sphere(80);
 popStyle();
}
```



shininess

`shininess(shine)`

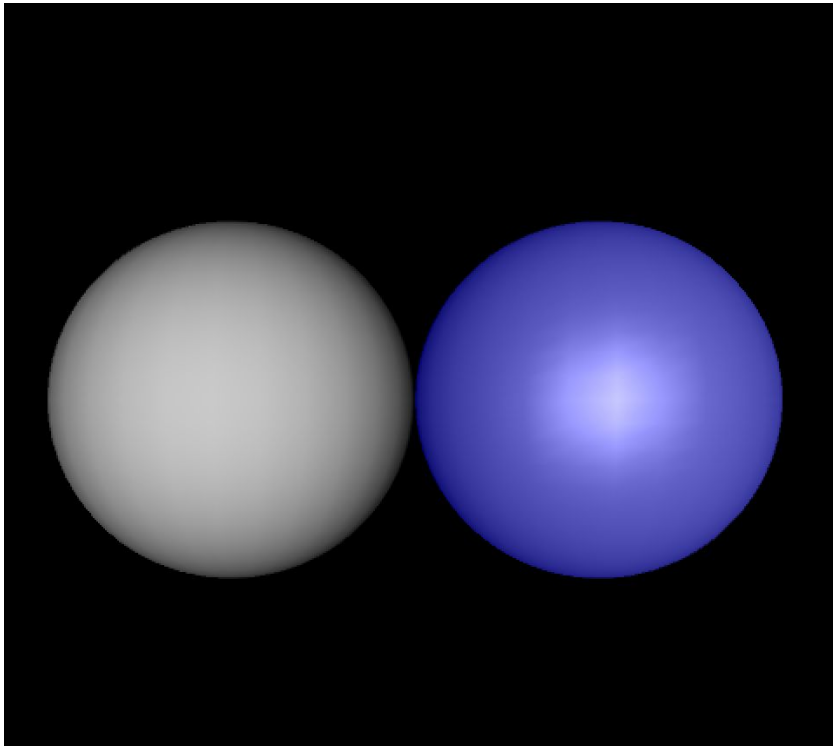
```
size(500, 500, P3D);
background(0);
noStroke();
scale(5);
fill(102, 102, 102);
lightSpecular(204, 204, 204);
directionalLight(255, 255, 255, 0, 0, -1);
translate(30, 50, 0);
shininess(1.0);
sphere(20);  // izquierda
translate(40, 0, 0);
shininess(5.0);
sphere(20);  // derecha
```



emissive

```
emissive(rgb)
emissive(gray)
emissive(v1, v2, v3)
```

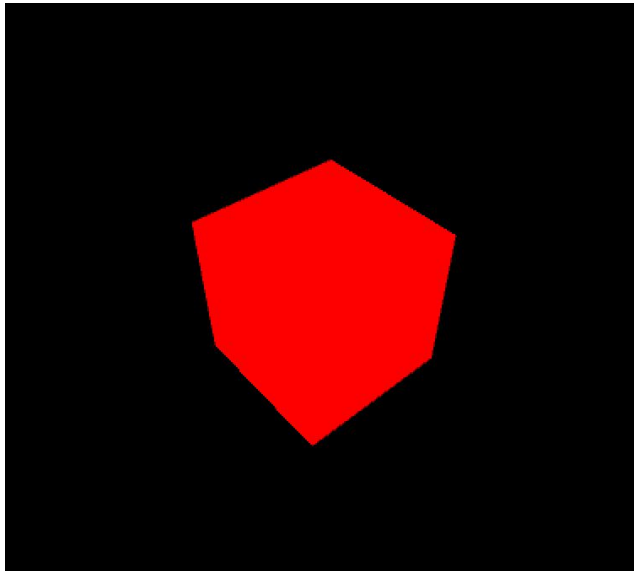
```
size(500, 500, P3D);
background(0);
noStroke();
scale(5);
fill(102, 102, 102);
lightSpecular(204, 204, 204);
directionalLight(255, 255, 255, 0, 0, -1);
translate(30, 50, 0);
shininess(1.0);
sphere(20);  // izquierda
translate(40, 0, 0);
shininess(5.0);
emissive(0,0,102);
sphere(20);  // derecha
```

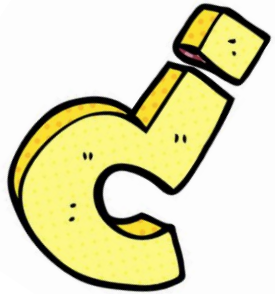


ambient

```
ambient(rgb)
ambient(gray)
ambient(v1, v2, v3)
```

```
float rotx = PI/4;
float roty = PI/4;
void setup() {
 size(400, 400, P3D);
 noStroke();
}
void draw() {
 background(0);
 ambientLight(255,255,0); // luz ambiente amarilla
  if (mousePressed) {
   directionalLight(0,0,255,0,0,-1); // -1 =  viene desde adelante hacia atrás (eje z)
 }
 translate(width/2.0, height/2.0, 0);
 rotateX(rotx);
 rotateY(roty);
 ambient(255,0,0); // reflejo luz roja ambiental
 box(100);
}
void mouseDragged() {
 float rate = 0.01;
 rotx += (pmouseY-mouseY) * rate;
 roty += (mouseX-pmouseX) * rate;
}
```

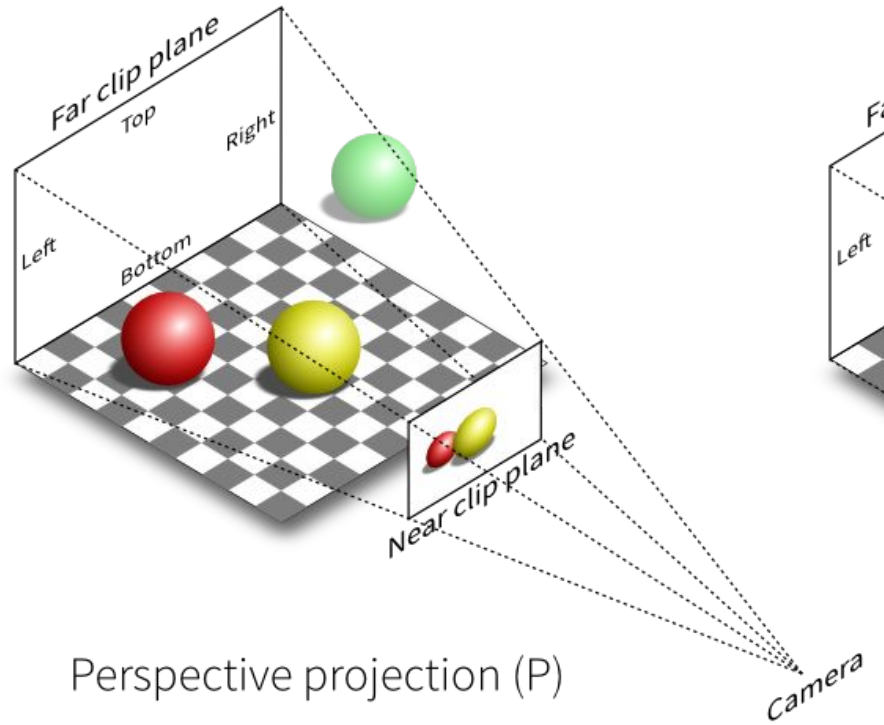




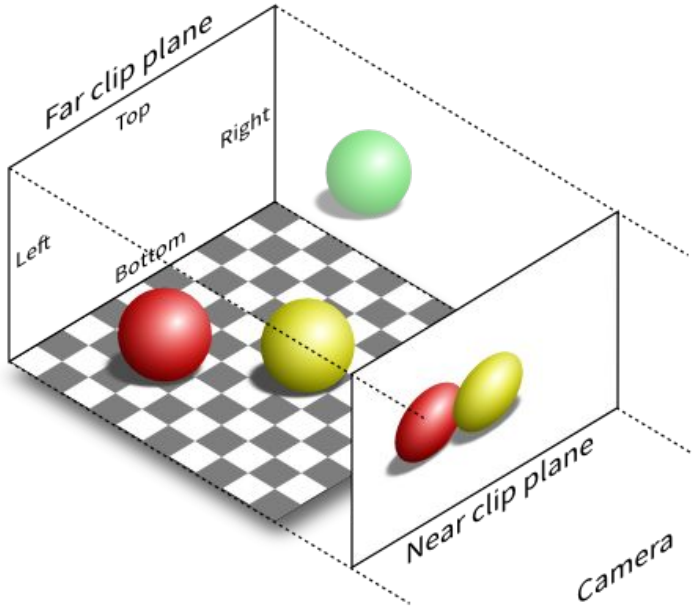
¿Dudas?

Visión de la escena

Visión en 3D



Perspective projection (P)



Orthographic projection (O)

Visión en 3D

Cámara

camera()
camera(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ) // donde está (eyeX, eyeY, eyeZ), hacia donde mira (centerX, centerY, centerZ) y cuál eje es el eje vertical (upX, upY, upZ)

Perspectiva

perspective()
perspective(fovy, aspect, zNear, zFar) // ángulo de campo de visión (en radianes) para dirección vertical, relación de ancho a alto, posición z del plano de recorte más cercano, posición z del plano de recorte más alejado.
Los parámetros definen un volumen de visualización con forma de pirámide truncada.

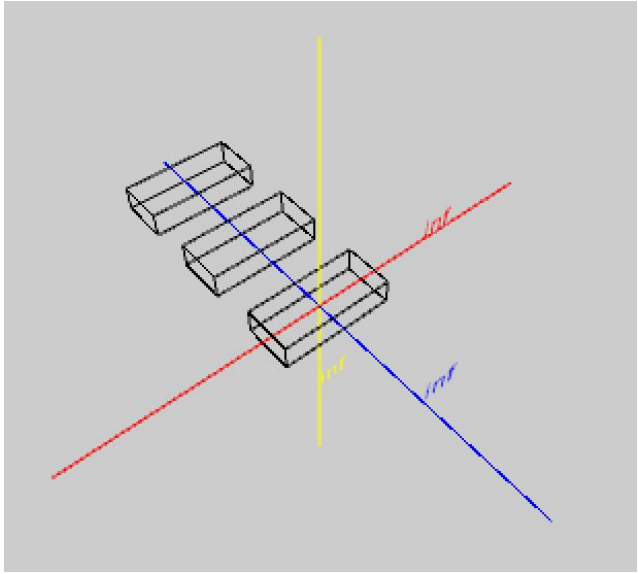
Proyección ortográfica

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

Ejemplo de la cámara

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

```
void setup(){
  size(400, 400, P3D);
}
void draw(){
  background(204);
  camera(100, 10, 120.0,  // donde esta
       width/2, height/2, 0, // hacia donde mira
      0.0, 1.0, 0.0); // el eje vertical es el eje Y
  translate(width/2, height/2, 0);
  stroke(255,0,0);
  line(-100,0,100,0); // línea roja en el eje x
  fill(255,0,0);
  text("inf",50,0,0);
  stroke(255,255,0);
  line(0,-100,0,100); //línea amarilla en el eje y
  fill(255,255,0);
  text("inf",0,50,0);
  stroke(0,0,255);
  line(0,0,-100,0,0,100);//línea azul en el eje z
  fill(0,0,255);
  text("inf",0,0,50);
  stroke(0); // contorno negro
  noFill();
  box(45,10,20);
  translate(0, 0, -40);
  box(45,10,20);
  translate(0, 0, -40);
  box(45,10,20);
}
```



Cámara por defecto

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

Los valores por defecto para la cámara son:

camera(width/2.0, height/2.0, (height/2.0) / tan(PI*30.0 / 180.0), width/2.0, height/2.0, 0, 0, 1, 0).

0.57

La cámara está en el centro de la escena, en un z > 0.

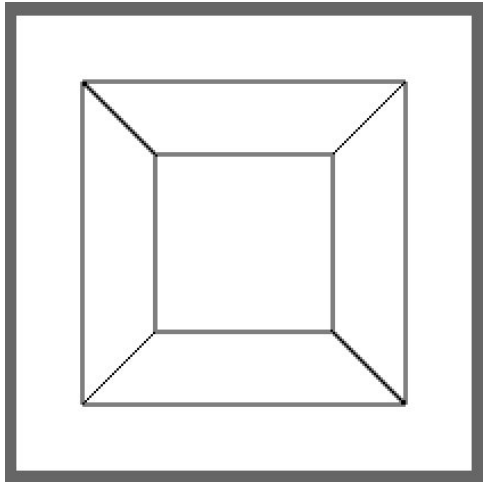
Mira al centro de la escena y z = 0.

El eje vertical es el eje Y.

**peasycam -> biblioteca de manejo de la cámara con el mouse

Perspectiva

`perspective(fovy, aspect, zNear, zFar)`

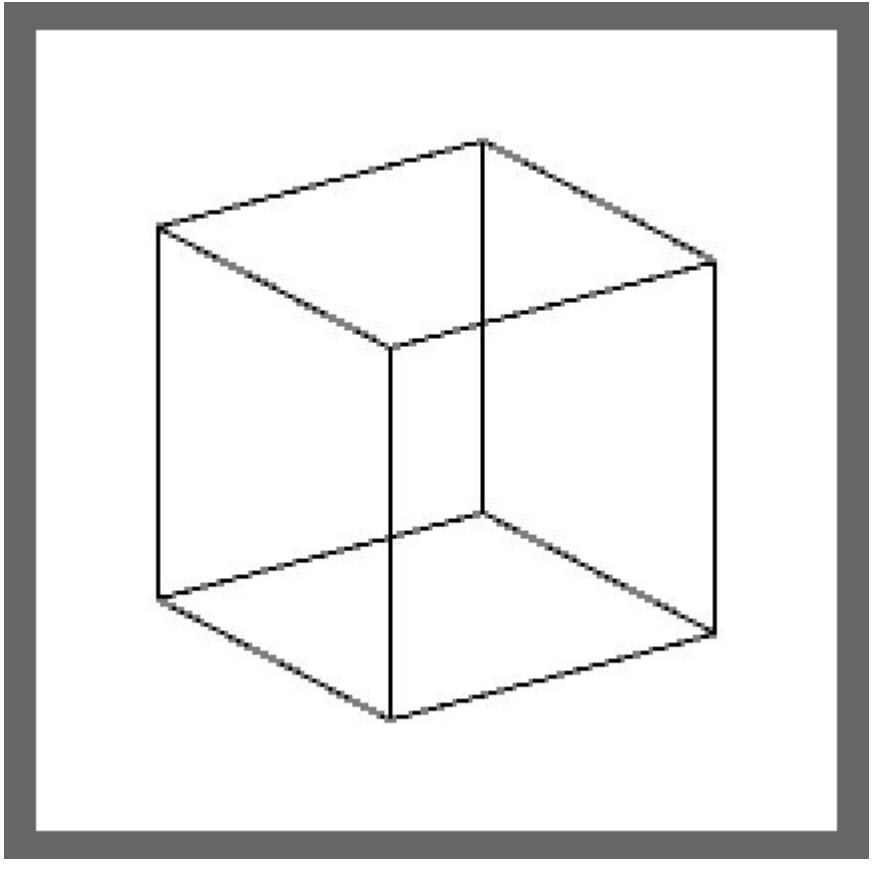


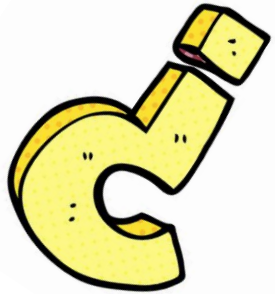
```
void setup() {
  size(200, 200, P3D);
  noFill();
}
void draw() {
  background(255);
  float fov = 1.04; // valor por defecto = PI/3.0 = 1.04;
  float cameraZ = 175; // valor por defecto (height/2.0) / tan(fov/2.0);
  float aspect = 1.0; // por defecto float(width)/float(height)
  perspective(fov, aspect, 17.5, 1750); // cameraZ/10.0, cameraZ*10.0
  translate(width/2, height/2, 0);
  box(100);
}
```

Proyección ortográfica

```
ortho(left, right, bottom, top)
```

```
float rotx = 0;
float roty = 0;
void setup(){
  size(200, 200, P3D);
  noFill();
}
void draw(){
  background(255);
  ortho(-100,100, -100, 100);
  translate(width/2, height/2, 0);
  rotateX(rotx);
  rotateY(roty);
  box(100);
}

void mouseDragged() {
  float rate = 0.01;
  rotx += (pmouseY-mouseY) * rate;
  roty += (mouseX-pmouseX) * rate;
}
```





¿Dudas?

Texto

Texto en 3D

text(c, x, y, z) // un char en 3D

text(str, x, y, z) // un string en 3D

text(chars, start, stop, x, y, z) // array de chars en 3D

text(num, x, y, z) // número en 3D

Ejemplo de texto en 3D

```
void setup(){
  size(400, 200, P3D);
  textSize(64);
  fill(255,255,0);
  textAlign(CENTER);
  stroke(255,0,0);
}
void draw(){
  background(0);
  translate(width/2,height/2);
  rotateY(radians(frameCount%360));
  text("hola 3D", 0, 15, 0);
  noFill();
  box(40);
}
```