Dibujar 3D en Processing Cheat Sheet

by César Navarro (CesarANS) via cheatography.com/143077/cs/30971/

Tipos primitivos en 3D

En processing hay 2 tipos primitivos en 3D

box(dim) , box(x,y,z)

> Genera una caja con ancho de **x**, alto de **y** y profundidad **z** o un cubo de dimensiones **dim**

sphere(r)

> Crea una esfera con radio **r**

Crear figuras 3D

Para crear figuras en 3D, es necesario el uso de **vertex(x,y,z)** siendo los parámetros las coordenadas de la figura, tal es el caso siguiente:

Ejemplo:

```
translate(width/2, height/2, 0);
rotate X(P I/2);
rotate Z(- PI/6);
beginS hape();
vertex (-100, -100, -100);
vertex (100, -100, -100);
vertex(0, 0, 100);
vertex (100, -100, -100);
vertex (100, 100, -100);
vertex(0, 0, 100);
vertex (100, 100, -100);
vertex (-100, 100, -100);
vertex(0, 0, 100);
vertex (-100, 100, -100);
vertex (-100, -100, -100);
vertex(0, 0, 100);
endSha pe();
```

Texturas

Para aplicar las texturas a una imagen, podemos usar **texture(img)**
Siendo **img** la imagen que se aplicará a la figura.
Se debe invocar **texture(img)** despues de **beginShape()**
Aunque para definir cómo se ajustará la imagen en la figura se usa **vertex(x,y,z,u,v)** donde los parametros **u** y **v** definen las coordenadas a mapear de la imagen en ese vertice.

Luces

Para activar las luces por defecto podemos usar **lights()**
Aunque también exiten funciones más especificas donde **v1** se refiere a **hue** o **rojo**, **v2** se refiere a **verde** o **saturación** y **v3** se refiere a **azul** o **brillo:**
ambientLight(v1,v2,v3[x,y,z])
Inidca la luz del ambiente, así como su origen
directionalLight(v1,v2,v3,x,y,z)
Indica la luz con origen en **x,y, z** qu eimpacta con más fuerza en nuna superficie plana que en una curva.
spotLight(v1,v2,v3,x,y,z,nx,ny,nz,deg,fuerza)
Similar a **directionalLight** pero incluye la dirección del cono de iluminación **(nx,ny,nz)**, así como la rotación de este **(deg)** y su fuerza **(fuerza)**.

Luces

pointLight(v1,v2,v3,x,y,z)
Crea spotlight con un cono de 180 grados con origen en **x,y,z**

Es importante mencionar que estos métodos deben estar en el método **draw** pues la escena 3D se reinicia cada frame.

Perspectivas

Para las perspectivas podemos usar la perspectiva normal
perspective()
perspective(fovy,aspect,zNear,zFar)
La cual sin argumentos pone la perspectiva predeterminada, y con argumentos puede ajustar el ángulo del campo de visión **(fovy)**, la razón de ancho y alto **(aspect)**, la posición en **z** del plano más cercano **(zNear)** y el plano más lejano **(zFar)**

O también la perspectiva ortogonal donde un elemento aparece de igual tamaño independientemente de qué tan lejos esté **ortho()**.
ortho(left,right,bottom,top,[near,far])
Donde sin parámetros se establece la configuración por defecto, o se pueden definir los planos lpimite en las 4 direcciones **(left, right,bottom, top)** así como la distancia máxima del origen a la vista**(near)** y la distancia mácima de lejanía desde el origen hasta la vista.

By **César Navarro** (CesarANS)
cheatography.com/cesarans/

Published 19th March, 2022.
Last updated 19th March, 2022.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com