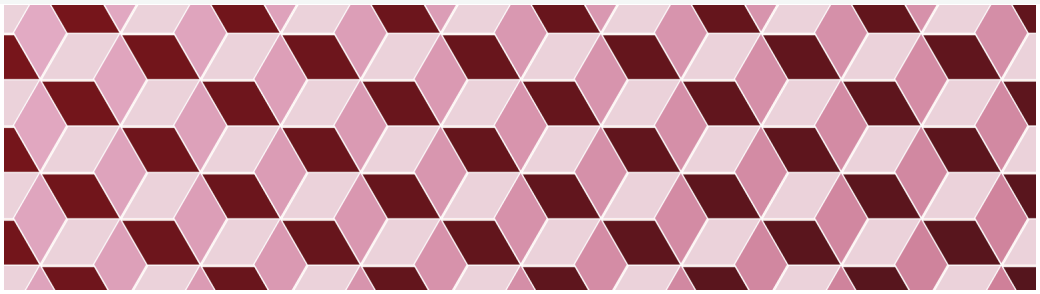


15-104 Introduction to Computing for Creative Practice

Fall 2020

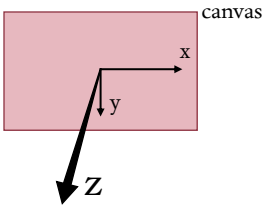
29 Three Dimensions

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1

Drawing in 3D

- We will use the WEBGL renderer to draw in "3D space". (We used the P2D renderer by default all semester.)
- In the WEBGL renderer, there is an x and y axis as before, but there is also a z (depth) axis!
- The origin in WEBGL is in the center of the canvas by default.
  - x increases left to right (as before)
  - y increases top to bottom (as before)
  - z increases toward us.
- By default, the canvas is the z=0 plane.



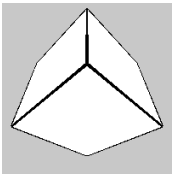
2

Example: Rotations

```
function setup() {
    createCanvas(400, 400, WEBGL);
}

function draw() {
    background(160);
    rotateX(frameCount * 0.01);
    rotateY(frameCount * 0.01);
    rotateZ(frameCount * 0.01);
    box(200, 200, 200);
}
```

A box of width 200, height 200 and depth 200.



`rotateX` rotates around the x axis the given amount in radians.
(similar for `rotateY` and `rotateZ`)

Try each rotation by itself and see if the visual makes sense. Combine them two at at time and see if it makes sense.

3

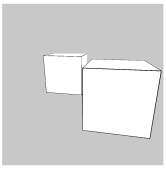
Camera and View

- The default camera view in WEBGL mode is ***perspective*** with a 60-degree field of view. (The other camera view is ***orthographic***.)
  - In a perspective view, objects closer to the viewer in the z-plane appear larger than those farther away.
  - In orthographic view, objects of the same dimensions appear to be the same size even if they are farther away on the z-plane.

4

Perspective

- camera frustum vertical field of view, from bottom to top of view
- camera frustum aspect ratio
- frustum near plane length
- frustum far plane length



```
function setup() {
  createCanvas(400, 400, WEBGL);
  perspective(PI/3.0, width/height, 0.1, 500);
}
function draw() {
  background(200);
  rotateX(0.3); rotateY(-0.2);
  push();
  translate(-80, 0, sin(frameCount / 30) * 90); box(120);
  pop();
  push();
  translate(80, 0, sin(frameCount / 30 + PI) * 90); box(120);
  pop();
}
```

5

Orthographic

- camera frustum left plane
- camera frustum right plane
- camera frustum bottom plane
- camera frustum top plane
- camera frustum near plane
- camera frustum far plane

```
function setup() {
  createCanvas(400, 400, WEBGL);
  ortho(-width/2, width/2, height/2, -height/2, 0, 500);
}
function draw() {
  background(200);
  rotateX(0.3); rotateY(-0.2);
  push();
  translate(-80, 0, sin(frameCount / 30) * 90); box(120);
  pop();
  push();
  translate(80, 0, sin(frameCount / 30 + PI) * 90); box(120);
  pop();
}
```

6

Camera

```
function setup() {
  createCanvas(400, 400, WEBGL);
}

function draw() {
  background(204);
  camera(0, 0, 80 + sin(frameCount * 0.01) * 40, 0, 0, 0, 0, 1, 0);
  box(50, 50, 50);
}
```

- camera position on x-, y- and z-axes
- center of sketch (x, y, z)
- camera up vector (in x, y, z)

Essentially, you indicate where the camera is, where the center of the sketch is, and the camera's orientation (which way is up for the camera).

Camera defaults: 0, 0, (height/2.0)/tan(PI*30.0 / 180.0), 0, 0, 0, 0, 1, 0

7

3D Shape Primitives

- There are seven 3D shapes you can use in your drawings:
  - `box, sphere, cone, cylinder, ellipsoid, torus, plane`
- Each primitive has parameters that specify its size, not position.
- An object is always drawn with respect to the origin of the 3D space.
- In order to draw an object elsewhere, use translate first.

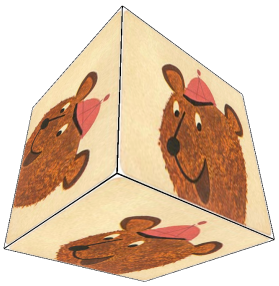
```
box(10,20,30); // width: 10, height: 20, depth: 30
cone(40, 100); // radius: 40, height: 100
```

8

Texture

```
var img;
function preload() {
  img = loadImage("https://i.imgur.com/my3TqY7.jpg");
}
function setup() {
  createCanvas(500,500,WEBGL);
}
function draw() {
  background(255);
  texture(img);
  orbitControl();
  box(200, 200, 200);
}
```

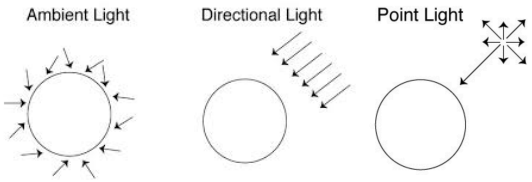
Allows movement around a 3D sketch using a mouse or trackpad



9

Lighting

- There are 3 types of light functions to provide depth and realism:
  - Ambient light – provides even omnidirectional light
  - Directional light - rays shine in a given direction, but they do not have a specific point of origin
  - Point light – rays shine from a specific point of origin



10

Lighting Examples

```
ambientLight(mouseX / width * 255);
sphere(160);
```

parameter: radius

```
var dirX = (mouseX / width - 0.5) * 2;
var dirY = (mouseY / height - 0.5) * 2;
directionalLight(250, 250, 250, -dirX, -dirY, -1);
sphere(160);

var locX = mouseX - width / 2;
var locY = mouseY - height / 2;
pointLight(250, 250, 250, locX, locY, 200);
sphere(160);
```

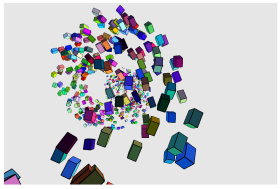
11

Materials (See: https://p5js.org/examples/3d-materials.html)

- In the real world, light reflects off objects differently, depending on their angle of reflection as well as the object's' material.
- There are four types of materials:
  - `fill(r,g,b)` – basic material, not affected by lighting
  - `normalMaterial()` - automatically maps a geometry's normal vectors to RGB colors (surfaces facing x axis become red, y become green, z blue)
  - `ambientMaterial(r,g,b)` – like `fill`, however the total color is affected by light functions that precede it.
  - `specularMaterial(r,g,b)` – simulates a shiny, reflective material – defines the color the object reflects under ambient lighting, or reflects the color of the light source to the viewer for other types of lighting.

12

Migrating Boxes



```
var boxes;
var zlimit = -4000;
function randomColor() {
    return color(random(255), random(255), random(255));
}
function setup() {
    createCanvas(600, 400, WEBGL);
    boxes = [];
    for (var i = 0; i < 1000; i++) {
        boxes.push({x: random(-100, 100), y: random(-100, 100),
                    z: random(zlimit, 300), color: randomColor()});
    }
    frameRate(20); // to reduce cpu load
}
```

13

Migrating Boxes (cont'd)

```
function draw() {
    background(230); orbitControl(); ambientLight(100, 100, 200);
    pointLight(255, 100, 100, 255, 300, 0, 0);
    pointLight(100, 255, 100, 255, 0, 300, 0);
    for (var i = 0; i < boxes.length; i++) {
        var b = boxes[i];
        push();
        translate(-200, 0, b.z); rotateZ(b.z*0.005);
        translate(b.x + 200, b.y, 0); b.z -= 4;
        ambientMaterial(b.color); rotateX(b.z*0.02); box(15,20,25);
        pop();
        if (b.z < zlimit) { b.z = 300; }
    }
}
```

14