CSC 240 Computer Graphics
Day 17: Texture Mapping

Nick Howe
Smith College
Homework 6 Notes

How to fix size & location of pyramid?

• Option 1: embed in coordinates of the geometry

// Create a pyramid as specified
function addPyramid(cx, cy, cz, h, sr) {
    var pyramidGeom = new THREE.Geometry();

    pyramidGeom.vertices = [
        new THREE.Vector3(cx+sr, cy+sr, cz+0), // vertex number 0
        new THREE.Vector3(cx+sr, cy-sr, cz+0), // vertex number 1
        new THREE.Vector3(cx-sr, cy-sr, cz+0), // vertex number 2
        new THREE.Vector3(cx-sr, cy+sr, cz+0), // vertex number 3
        new THREE.Vector3(cx+0, cy+0, cz+h) // vertex number 4
    ];

    ...
How to fix size & location of pyramid?

• Option 2: use default geometry with transformations

```javascript
// Create a pyramid as specified
function addPyramid(cx, cy, cz, h, sr) {
  var pyramidGeom = new THREE.Geometry();

  pyramidGeom.vertices = [
    new THREE.Vector3(1,1,0), // vertex number 0
    new THREE.Vector3(1,-1,0), // vertex number 1
    new THREE.Vector3(-1,-1,0), // vertex number 2
    new THREE.Vector3(-1,1,0), // vertex number 3
    new THREE.Vector3(0,0,1) // vertex number 4
  ];

  var pyramid = new THREE.Mesh( pyramidGeom, pyramidFaceMaterial );
  pyramid.position.set(cx, cy, cz);
  pyramid.scale.set(sr, sr, h);
}```
if (ticks == 0) {
    // time to pick a new action
} else {
    // carry out the action
    if (da == 0) {
        // we are moving
        camera.position.x += step*dx;
        camera.position.y += step*dy;
    } else {
        // we are turning
        camera.rotation.y += da*astep;
    }
}
ticks--; // count down
if (ticks == 0) {
    // time to pick a new action
    if (da == 0) {
        // we were moving so now begin turning
        da = (Math.random() > 0.5) ? -1:1;
        ticks = Math.floor(120*Math.random())+30;
    } else {
        // we were turning so now begin moving
        da = 0;
        dx = -Math.sin(camera.rotation.y);
        dy = Math.cos(camera.rotation.y);
        ticks = 90+Math.floor(120*Math.random());
    }
} else {
    // carry out the action
    if (da == 0) {
        // we are moving
        camera.position.x += step*dx;
        camera.position.y += step*dy;
    } else {
        // we are turning
        camera.rotation.y += da*astep;
    }
}
ticks--; // count down
Solar System Hierarchy

Scene

Sun

Earth Orbit

Earth

Moon Orbit

Moon

↑ = add

Rotate these to make orbit

Add these at offset position
Alternate Solar System Hierarchy

Scene

Sun center

Earth center

Moon Center

Sun

Earth

Moon

Rotate these to make orbit

Add these at offset position

↑ = add
Q. I didn't quite understand what you meant by exterior points and interior points in the barycentric coordinates

A. Interior points are inside the triangle. Exterior are outside.
Q. Can you give an example of how coefficients of barycentric coordinates are calculated?

A. Step 1: Compute $f_{12}$, $f_{23}$, and $f_{31}$.

Step 2: Compute $\alpha$, $\beta$, and $\gamma$.

$$f_{23} = (y_2 - y_3)x + (x_3 - x_2)y + x_2y_3 - x_3y_2$$

$$\alpha = f_{23}(x, y)/f_{23}(x_1, y_1)$$

$$f_{31} = (y_3 - y_1)x + (x_1 - x_3)y + x_3y_1 - x_1y_3$$

$$\beta = f_{31}(x, y)/f_{31}(x_2, y_2)$$

$$f_{12} = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1$$

$$\gamma = f_{12}(x, y)/f_{12}(x_3, y_3)$$

Q. Can you do a math example of shading?

A. Step 1: compute diffuse shading

Step 2: compute specularity

$$\hat{R} = 2(\hat{L} \cdot \mathbf{N})\mathbf{N} - \hat{L}$$

$$I_s = k_s(\hat{R} \cdot \hat{D})^\alpha$$

Step 3: add ambient, emissive light

$$I = I_d + I_s + I_a + I_e$$
Your Questions

Q. Is there a particular order when copying colors from the texture image onto rendered surfaces?

A. The colors will be copied according to the order of pixel shading in the half-triangle fill algorithm. The corresponding barycentric coordinate is computed and the converted to (u,v) coordinates and read from the mipmap of appropriate resolution.
Your Questions

Q. Why the texture image should be square? What if we want to put a texture image on a non-square polygon?

A. Square images are the standard, and simplify the \((u,v)\) math. Portions of the texture map can be left unused for irregular shapes.
Your Questions

Q. Why are squares the best?
A. They are standard. It is easier to rescale a square for the mipmap.

Q. What happens if texture images are not square?
A. Older versions of Three.js wouldn’t display it. More recent versions rescale the image automatically.

Q. If we only want one texture image for a triangle - do we still apply it to the 3 vertices? if so, do they overlap?
A. We define three UV coordinates for a triangle.
Your Questions

Q. Do textures need to be JPG or can it be other formats?
A. Any web image format is fine (PNG, GIF, etc.)

Q. How do you know the uv coordinates of the geometries already in Three.js?
A. They are mostly simple and “do what you would want”.
You can write a program to print them out if you are curious.

Q. What are some examples for implementation of texture mapping?
A. We’ll look at this in the lab today.
Q. How do we get the detailed texture image in the first place?

A. For purposes of this course, we will not create our own textures. You can download textures from free sites like https://3dtextures.me/. These may be created by laser scanning or high-precision renders.

Q. And how to map it onto the rendered surface? Does the order of mapping matter? How to determine the order?

A. The UV coordinates determine how the texture is mapped. Premade geometries in Three.js have default UV coordinates already specified. If you create your own geometry, you have to define the UV coordinates yourself (as we will do in the lab).
Q. I don't understand the last question "This icosahedron has 20 faces and 12 vertices total. How many $uv$ coordinates will you need to provide in order to fully map it with a texture?"

Q. I didn't quite understand why the 3rd question was 60?

A. The icosahedron has 20 triangular sides. For each triangle you need to provide $(u,v)$ coordinates at each of the three corners. $20 \times 3 = 60$
Lab 10