CSC 240 Computer Graphics
Day 17: Texture Mapping

Nick Howe
Smith College
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 = [
        // array of Vector3 giving vertex coordinates
        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

Rotate these to make orbit

Add these at offset position

↑ = add
Alternate Solar System Hierarchy

Rotate these to make orbit
Add these at offset position

↑ = add
Q. I don't understand how Phong shading creates the highlight with weighted values. Can you explain in detail the differences between Gouraud and Phong again?

A. *Both start with normal vectors at the corners.*

Gouraud computes shadings at the three corners using their normal vectors, and then interpolates colors across the triangle. Because the shading is only computed at the corners, it may miss central highlights.

Phong interpolates the normal vectors across the triangle and uses the interpolated normal to compute the shading at each point. Because the normal vector differs everywhere, you can get specular highlights.
Q. When using barycentric coordinates to compute the average color or normal, will alpha always correspond to red/x, and beta to green/y, and gamma to blue/z (i.e. each weight corresponds to a specific component), or do you have to compute the average for each component individually?

A. I made the corners pure colors just for an easy example. In most cases you will have different amounts of all three components, and average them.

Q. Why are the barycentric coordinates computed with the distance from the corner point to the side of the triangle? Do the distances always have to be perpendicular to the side opposite the point?

A. The coordinates are the distance to the side line (always computed as a perpendicular), scaled by the distance to the opposite corner. The scaling ensures that the coordinate value ranges from 0 to 1 across the triangle.

Q. What do you mean by the coefficients are used as "weights"? Just to confirm, barycentric coordinates are only used with Phong and Gouraud shading?

A. Since the barycentric coordinate values sum to 1, they can be used to compute a weighted average.
Your Questions

Q. Why do texture coordinates only range from 0 to 1?
A. That is the chosen convention.

Q. How did you find such specific numbers for the coordinates of the corners of the green cube face? Or was it just an estimate?
A. The green face is 1/3 of the total image size. The decimals given are approximations of the fractions 1/3 and 2/3.

Q. Can you explain the uv dimension assigning code? Do we have to individually map out all of the coordinates for 20 faces or is there a more efficient way of doing this?
A. If you use a built-in Three.js geometry, they also come with predefined UV coordinates. You can override the provided coordinates if you wish. Also, if we create our own geometries, we need to provide the UV coordinates ourselves for all the faces.
Texture Mapping in Three.js

Three.js offers functionality to implement texture maps

- A class to load textures:

```javascript
var loader = new THREE.TextureLoader();
var myTexture = loader.load("myTexture.jpg");
```

- Add the texture map to a Phong material:

```javascript
var myMaterial = new THREE.MeshPhongMaterial(
  { map: myTexture }
);
```

- Add \((u, v)\) coordinates to each face of a geometry:

```javascript
var uvcoords = [new THREE.Vector2(0, 0),
    new THREE.Vector2(1, 0),
    new THREE.Vector2(0, 1)];
myGeom.faceVertexUvs[0].push([uvcoords[0], uvcoords[1], uvcoords[2]]);
```
Your Questions

Q. Why are geometries in three.js always composed of triangles?  
A. *Triangles are universal.* Any complex polygon can always be broken down into triangles. We have efficient algorithms for rendering triangles. Therefore, why use anything else?

Q. Can you elaborate on what texels are and how they are used?  
A. *“Texel” stands for “texture element.”* In practice, each texel is one pixel in a texture map.

Q. Can you elaborate on what a mipmap is? I’m having trouble understanding why they are useful.  
A. Answer on next slide.
Mipmap Usage

Large fill area with many pixels. Use big texture in mipmap

Small fill area with few pixels. Use tiny texture in mipmap
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.
Your Questions

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 $\gamma$.
   $f_{23} = (y_2 - y_3)x + (x_3 - x_2)y + x_2y_3 - x_3y_2$ \hspace{1cm} \alpha = f_{23}(x, y)/f_{23}(x_1, y_1)$
   $f_{31} = (y_3 - y_1)x + (x_1 - x_3)y + x_3y_1 - x_1y_3$ \hspace{1cm} \beta = f_{31}(x, y)/f_{31}(x_2, y_2)$
   $f_{12} = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1$ \hspace{1cm} \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
   $I_d = k_d C_m C_l (\hat{L} \cdot \hat{N})$
   Step 2: compute specularity
   $\hat{R} = 2(\hat{L} \cdot \hat{N}) \hat{N} - \hat{L}$ \hspace{1cm} I_s = k_s (\hat{R} \cdot \hat{V})^\alpha$
   Step 3: add ambient, emissive light
   $I = I_d + I_s + I_a + I_e$
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://3dtexutres.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).
Your Questions

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. $20 \times 3 = 60$