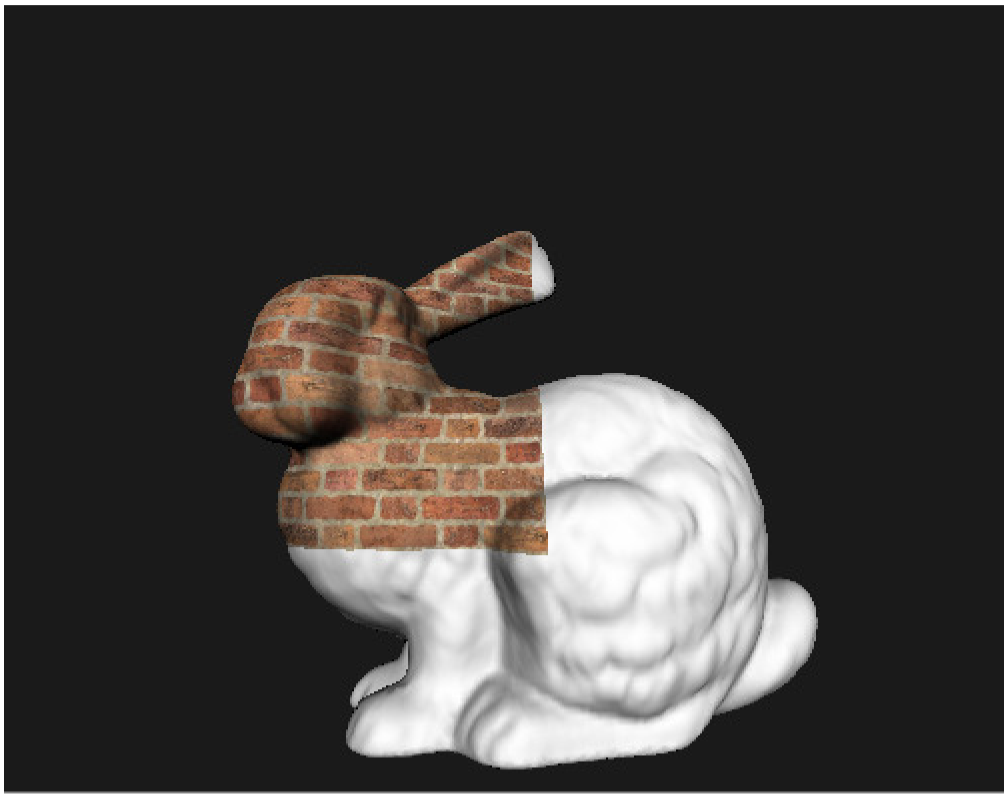
WebGL + “Shower Door” Effect



Description:
You’ve previously been introduced the GL Shader Language (GLSL) version 1.2 (in Lab 8). We now shift to doing WebGL (OpenGL ES) on the web. For this lab, your job is to utilize a combination of vertex and fragment shaders so that the user can interactively project a texture onto a 3D object.

Read Before You Start:
Some things that you should be aware of about WebGL (and Javascript caveats):

- First, you no longer have access to a matrix stack. That’s right, this means that you can no longer use glTranslate or glRotate, and no glPushMatrix and glPopMatrix. You are now responsible for maintaining your own stack.
  - For this lab, we are using the gl-matrix library (https://github.com/toji/gl-matrix)
  - You are welcome to use any other library (or write your own!)
- Since Javascript uses dynamic type checking, be careful with your strings and numbers. This is especially true with parsers… I had a bug where I did something like:
  - var num1 = Number(parse(filename)); //say, this returns 20
  - var num2 = 5;
  - var num3 = num1 + num2;

  - I was expecting num3 to 25, but instead, num3 came back to be the string "205".
- Opposite of Javascript is GLSL (the shaders), which is very strictly typed and requires precise type assignments. For example:
  - float num = 0; // this will throw an error!!
  - if (num > 0) … // similarly this will also throw an error!!
  - The correct syntax is to use 0.0. For example,
    - float num = 0.0;
- One common mistake in GLSL (that I have made) is to use operations on vec4. For example:
  - When dealing with positions:
    - vec4 position1 = vec4 (0.5, 0.5, 0.5, 1.0);
    - vec4 position2 = vec4 (0.5, 0.5, 0.5, 1.0);
    - gl_Position = position1 + position2;
    - One would assume that gl_Position would have the value of (1.0, 1.0, 1.0, 1.0).
    - However, the value of gl_Position turns out to be (0.5, 0.5, 0.5, 1.0) because position1+postion2 is technically (1.0, 1.0, 1.0, 2.0). Recall that GL uses a 4 dimensional homogeneous coordinate system, so GLSL automatically "normalizes" the w component to 1.0, resulting in (0.5, 0.5, 0.5, 1.0).
  - Similarly, for colors:
    - vec4 color1 = vec4 (1.0, 1.0, 1.0, 1.0);
    - color1 = color1 * dotProductForShading;
    - gl_FragColor = color1;
    - You would imagine that you will get the proper shading, but it turns out that you are manipulating the alpha channel as well, which will have potentially weird effects depending on the state of your GL's blending function. In my case, I get all white when the dotProduct is something close to 0.1.
  - **Bottom line: do not use vec4 unless you have to!** Stay with using vec3 and pad it with an extra 1.0 into a vec4 when needed.
- Lastly, there's this annoyance that happens because of compiler optimization in GLSL. The problem is that if you were to pass a variable to the shader (this affects both attribute and uniform variables), but in your shader code you don't use the information in any way, the compiler would strip out the calls and create both errors and warnings.
  - You can see the effect of this by commenting out line 32 in the vertex shader (in index.html). The line says:
    - vec3 normal = aVertexNormal
  - With the line commented out, you'll see the errors and warnings in the debugger console.

Your Tasks:
This lab is all about figuring out webGL by trying to implement the shower door effect, and **extending it to include 1 interesting feature**.

1. Implement the vertex and fragment shaders to get the shower door effect. This means that:
    a. Get the program to run in your browser. See the section "Programming Environment" below to get started.
    b. Once you're set to go, your first task should be to get shading to work. Note that, unlike Lab8, there is no "light vector" passed into the scene. You may assume that light is coming from a fixed location.
    c. You will need to pass in the mouse position and possibly the canvas size (see tutorial 4).
    d. The texture is read and sent to the shader for you, but you'll need to figure out how to use that texture image to create the shower door effect based on the mouse position.
2. Do something on top of this lab beyond the shower door! Be creative about what the feature is. Some possible suggestions include:
    a. Add keyboard interactions to rotate the 3D object.
    b. Turn on and off moving lights or animation (similar to Lab8).
    c. Do something in HTML to control the webGL scene (e.g. add buttons, sliders, etc.)
    d. Use multiple textures
    e. Fix my big in tutorial 7! This means that the 3D object will have a spherically projected texture, but as the user moves the mouse, the sphere rotates.
    f. etc.

Programming Environment:
Since we are doing web programming, you have your choices of your development environment. My personal choice is to keep the files on the CS Linux server and edit the files there directly. However, you are welcome to run a local web server and keep everything on your own computer (which might make collaboration easier). For a tutorial of how to set up a simple local web server using Python, see Mike's additional handout.