CSS 452: Programming Assignment #3
Working with Source Code Base

Due time: Please refer to our course web-site

Objective
There are two main purposes for this assignment. First, for you to take a deep dive into the source code base. You will modify almost every file (in my implementation, I did not modify the following three files: gl.js, vertex_buffer.js, and transform.js) in the source code base where you should become familiar with the entire code base. Second, for you to develop a more in-depth appreciation for the shader interface between the JavaScript in the CPU and the GLSL shader code. This will be accomplished by extending the Renderable class to support the drawing of an optional *shadow*. The details are as followed.

Assignment Specification:
Here is screen shot of the results from the assignment:

You must modify and extend the **Renderable** class to support the drawing of an optional *Shadow*. In our very simple case, a *shadow* is simply an additional instance of the same rectangle with user specified color and x/y offset drawn before the actual primitive is drawn. For example, on the left screen shot, the exact same two squares as Example 3.3 are drawn with black shadows. In this case, the shadows are in black, and 5% offset in both x and y from the actual squares. Here are the required modification to the Renderable class:

- Constructor: expects a boolean parameter indicating if shadow drawing is enabled for the instance:
  - True: indicates a shadow should be drawn,
  - False: Renderable should be drawn as before
- setShadowColor(color):
  - color: is a vec4, indicating the color of the shadow to be drawn
- setShadowOffset(offset):
  - offset: is a vec2, indicating the x/y offset of the shadow square.

You must support the offset in object space. This means, an offset of 0.1 is 10% of the size of the object that you will observed on the screen. It is recommended that you support this functionality by defining a new GLSL vertex shader where, the shader will accumulate the offset before applying the TRS model transform. The source code for the GLSL shadow shader can be as listed:

```
//
// This is the vertex shader
attribute vec3 aVertexPosition;  // Vertex shader expects one vertex position

// to transform the vertex position
uniform mat4 uModelXformMatrix;

// to support shadow offset
uniform float uOffsetX;
uniform float uOffsetY;

void main(void) {
    vec3 p = aVertexPosition + vec3(uOffsetX, uOffsetY, 0);
    gl_Position = uModelXformMatrix * vec4(p, 1.0);
}
```

With the new vertex shader, you can define a new ShadowShader (using the same source code in simple_fs.glsl), and then, you can draw the same square twice, first with ShadowShader, and then, with the ConstColorShader.

Please use (***and do not modify***) the provided user source code: index.html, my_game.js, and my_shadow.test.js. If you would like create extra something, please define a new Canvas for that.

Hints

- Please use Example 3.3 (transform_objects) as the base of your implementation.
- This is what I did:
    - Create a new shadow_vs.glsl (with the above given source code)
    - Modify shader_resources.js: to create a new ShadowShader
    - Modify simple_shader.js: to create a subclass from SimpleShader to handle the new ShadowShader. This is a subclass of SimpleShader where the constructor and activate() functions are all reused.
    - Modify renderable.js to support the new shadow
        - Hint: my new Renderable now has two shaders, ShadowShader for drawing the shadow, and the SimpleShader for drawing the normal primitive
- In general, the number of lines of code that you need to create is relatively slow, but, it requires you have a good understanding of the existing system, and, you will have to integrate your changes in to a non-trivial source code system. This can be intimidating, and when your solution does not work it is very likely nothing will show. Start early and bring questions to class or email me for help!
- Javascrip help:
    - To create a subclass:
      ```
      class ShadowShader extends SimpleShader {}
      ```
    - To call super class methods (including constructor):
      ```
      super(/* with appropriate arguments */ );
      ```

Credit Distribution

Here is how the credits are distributed in this assignment:

1.	Defined ShadowShader class		25%
	a. Include shader_vs.glsl b. Support shadow color setting c. Support x/y offset in Object Coordinate	5% 10% 10%	
2.	Proper integration of ShadowShader		20%

	a. Definition in shader_resource.js b. Proper initialization c. Proper access methods defined	10% 10% 10%	
3.	Renderable class update		30%
	a. Shadow on/off b. Shadow color c. Shadow offset d. Shadow drawn with ShadowShader	5% 10% 10% 10%	
4.	Support the defined API		15%
	a. No changes to my_game.js b. No changes to my_shadow_test.js c. No changes to index.html	5% 5% 5%	
5.	Proper submission		10%
	a. Zip file names with **NO SPACES** b. No extra unused files/folders (E.g., Test folder) c. Styles (project name, variable names, etc.)	10% 10% 10%	

This programming assignment will count 8% towards your final grade for this class.

Creativity and Extra Credits: Can you draw something interesting with the shadow?