CPSC314
Assignment 1: Introduction to Three.js, WebGL, and Shaders

Due 23:59:59, Oct 2, 2020

0 Plagiarism Policy

By handing this assignment in, you agree to the course plagiarism policy as stated here: `https://www.students.cs.ubc.ca/~cs-314/Vjan2013/cheat.html`. Please read it, and make sure you understand it.

1 Introduction

The main goals of this assignment are to set up your graphics development environment, including checking your browser compatibility, enabling the use of local files, and an initial exploration of the uses of vertex and fragment shaders. For this exploration, you will be using a template provided by the instructor, including shader code (.glsl files).

Your main task will be to understand how the provided code works overall, to modify the shaders to perform simple fragment and vertex-based operations, and to explore rudimentary communication between the JavaScript program and the shaders. Some of the details of what is going on in the rest of the code will only become clear a bit later in the course. You are of course, welcome to take a peek now, especially for the last part of the assignment. Some of the concepts are explained in Appendix A of your textbook ("Foundations of 3D Computer Graphics"), and in the web resources listed on the course web page.

To program a shader, you will use a programming language called GLSL (OpenGL ES Shading Language version 3.0). Note that there are several versions of GLSL, with more advanced features available in regular OpenGL. Make sure that any code you find while trying to learn GLSL is the correct version.

The template provides a simple scene with a 3D raccoon model and a red "remote-control" sphere. You can move the camera around the scene by dragging a mouse. Your task will be to change a number of scene properties and to *activate* the remote control.

1.1 Template

- The file `A1.html` is the launcher of the assignment. Open it in your preferred browser to run the assignment to get started.
- The file `A1.js` contains the JavaScript code used to set up the scene and the rendering environment. You will need to make minor changes to answer the questions.
- The folder `glsl` contains the vertex and fragment shaders for the character and the remote control. This is where you will do most of your coding.
- The folder `js` contains the required JavaScript libraries. You do not need to change anything here.
- The folder `obj` contains the geometric models loaded in the scene. We will discuss the format encoding them later in the course.
- The folder `images` contains the texture images used.

1.2 Execution

As mentioned above, the assignment can be run by opening the file `A1.html` in any modern browser. However, most browsers will, for security reasons, prevent pages from accessing local files on your computer. If you simply open `A1.html`, you may get a blank screen and an error message on the console similar to this:

```
XMLHttpRequest cannot load raccoon.vs.glsl . Cross origin requests are only supported
for protocol schemes: http, data, https.
```

We highly recommend that you run a local server, instead of changing browser security settings. Please see this web page for options on how to run things locally:

`https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally`

You may also follow these steps to run a local server through Visual Studio Code:

1. Follow the link `https://code.visualstudio.com/Download` to download and install VSC.

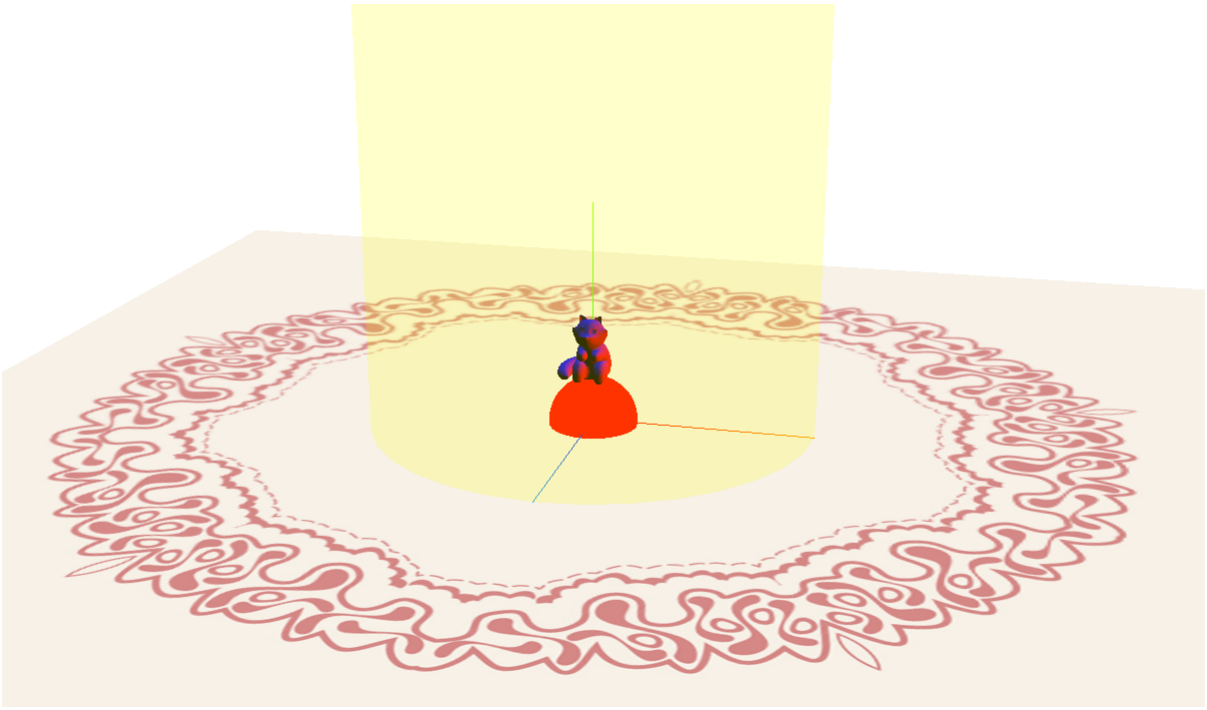
2. Open VSC and search for the Live Server extension. Download and install it. You may also install it here: `https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer`

3. In VSC, open the assignment root folder.

4. Open the `A1.html` file and right-click anywhere in the window to Open with Live Server, or click the Go Live button on the bottom. You should see the assignment open on the browser of your choice.

Please note that if you are using Chrome, you may need to do a "hard reload" or "empty cache and hard reload" in the browser (Inspect page to enable these settings) to see any changes made to `.glsl` or `.js` files updated on your screen.

2 Work to be done (100 pts)



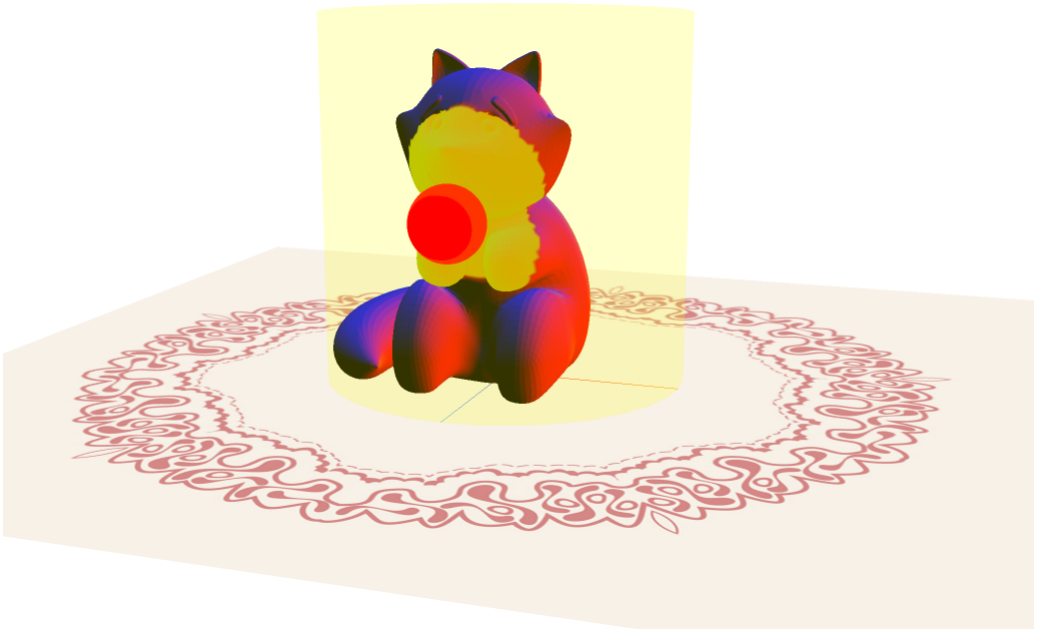
1. **Part 1:** (75 points)

   (a) **15 pts** Raccoon placement.
   Modify the raccoon vertex shader (`raccoon.vs.glsl`), to scale and translate the raccoon so that a) its bottom is on the floor (roughly) and b) its paws are roughly the same scale as the remote control orb. **5 points** will be given to those who compute the exact translation for a) algorithmically.

   (b) **15 pts** Changing remote control color (state)
   Modify the remote control's fragment shader (`remote.fs.glsl`), to change the remote's color based on the value of the variable `rcState`. Note that you have to pass this uniform variable to the shader from the JavaScript, where it is already defined and changed by keyboard keys. Then you will be able to switch states/colors using your keyboard (keys 1-3) and see the remote react.

   (c) **15 pts** Moving the remote control
   The variable `remotePosition` (the position of the remote control center in world coordinates) declared in `A1.js` is changed using the keyboard, and passed to the remote control shader (`remote.vs.glsl`) using `uniform` variables. Modify the remote control shader to move the remote in response to keyboard input.
   Important: **do not** *use Three.js functions; you must modify the shader for credit.*

   (d) **30 pts** Interaction
   Here you will explore the interaction between different models in the scene, where each is a prior described in its own coordinate system. Specifically, you should modify the raccoon's shaders so that the fragments that are within a small (fixed) distance from the remote, change color to a color of your choice.
   Hint 1: You might do something along the lines of the `interpolatedNormal` variable, defined in the raccoon vertex shader, and passed on to its fragment shader.



Hint 2: In the vertex shader, you may want to use one of the predefined transformation matrices, listed below.

```
uniform mat4 modelMatrix;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;
uniform mat4 viewMatrix;
uniform mat3 normalMatrix;
```

2. **Part 2:** (25 points) States - Creative License

   Now it's time for the program to render the raccoon (and possibly other objects in the scene) differently for each of the three different remote control states. Make the raccoon look cool! Possible options may include:

   - deforming the vertices in the raccoon in a wave over time,
   - adding noise,
   - exploding/offsetting the model along face normals,
   - coloring it differently with colors evolving over time,

   or anything of similar complexity.

   Bonus marks may be given at the discretion of the marker for, particularly noteworthy explorations.

3 Hand-in Instructions

You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called "a1" under your "cs314" directory. Put all the source files and your README.txt file in there. Do not use further sub-directories. The assignment should be handed in with the exact command:

```
handin cs314 a1
```

Alternatively, you can use the web interface (`https://my.cs.ubc.ca/docs/hand-in`), which does exactly the same thing. Log in with your CWL credentials, write "cs-314" for the course, "A1" for the assignment name, and submit your zipped assignment folder.

It is always in your best interest to make sure your assignment was successfully handed in. To do this, you may either use the `check submissions` button in the online handin, or using the -c flag on the command line:

```
handin -c cs314 a1
```