CPSC 314
Assignment 1: Attacking the Coronavirus! Introduction to Three.js, WebGL, and Shaders

Due 11:59PM, Jan 27, 2021

1 Introduction

The main goals of this assignment are to setup your graphics development environment, including checking your browser compatibility, setting up a local server, 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 in the `glsl/` folder). Your main work will be to develop a high level understanding of how the code works, to modify or write shaders, and to use 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, 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.
This assignment uses a simple scene consisting of an "Armadillo" character and a disassembled "Coronavirus". Your task for this assignment will be to move the virus around, detect how close it is to the poor Armadillo, and to assemble a more realistic looking virus with spikes. You can move the camera around the scene by dragging with a mouse, pan by holding down the right mouse button while dragging, and zoom by scrolling the mouse wheel.

1.1 Getting the Code

Assignment code is hosted on the UBC Students GitHub. To retrieve it onto your local machine navigate to the folder on your machine where you intend to keep your assignment code, and run the following command from the terminal or command line:

```
git clone https://github.students.cs.ubc.ca/cpsc314-2020w-t2/a1-release.git
```

1.2 Template

- The file `A1.html` is the launcher of the assignment. Open it in your preferred browser to run the assigment, 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 in it to answer the questions.
- The folder `glsl` contains the vertex and fragment shaders for the armadillo and light-bulb geometry. 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.
- The folder `images` contains the texture images used.

1.3 Execution

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

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

Please see this web page for options on how to run things locally:

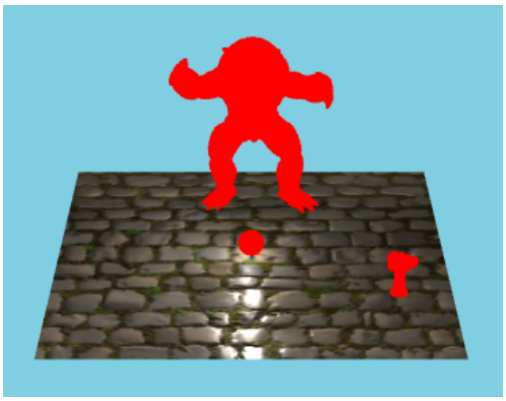
https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

We highly recommend that you run a local server, instead of changing browser security settings.

1. Follow the link https://nodejs.org/en/ to download and install Node.js, which comes packaged with npm.
2. Open the link https://www.npmjs.com/package/http-server and follow the instructions to download and install a local command-line http server.
3. Go to the command-line or terminal and run `http-server [path]` where [path] is the path to the assignment folder.
4. Open your preferred browser and copy and paste the URL of the local server specified by the http-server on your command-line.

2 Work to be done (100 pts)

First, ensure that you can run the template code in your browser. See the instructions above. Study the template to get a sense of how it works. The initial configuration should look as it does in the figure below.

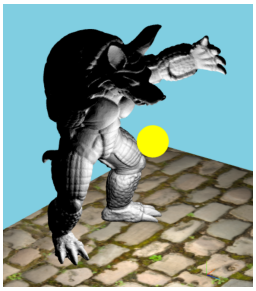


Part 1: Required Elements

(a) **20 pts** Moving & Coloring a Simple Virus. We initially represent the virus using the sphere, and manipulate it using the vertex shader `sphere.vs.glsl`. The variable `virusPosition` (the position of the virus center in world coordinates) is declared in `A1.js`. It is changed using the keyboard, and passed to the sphere vertex shader using a `uniform` variable. First, modify the sphere shader to move the sphere in response to keyboard input. Then, change the color of the sphere to yellow in the sphere fragment shader (in `sphere.fs.glsl`). Important: **do not** use Three.js functions; you must modify the shader for credit.

(b) **20 pts** Activate the Virus.

Modify `A1.js` and the armadillo shaders (`armadillo.vs.glsl` and `armadillo.fs.glsl`), to color points on the surface of the armadillo based on the cosine of the angle between its normal and the direction vector to the center of the sphere. When correctly coded, the virus will be "activated", affecting different parts of the armadillo as it's moved around, as illustrated in the figure below.



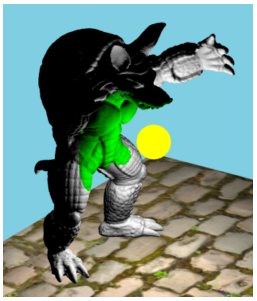
Hint 1: See how uniforms are passed to the sphere shader.

Hint 2: You should pass the necessary information about the sphere to the armadillo shaders.

Hint 3: See how varying variables are passed to the armadillo fragment shader.

(c) **30 pts** Infectious virus.

For this part you will need to modify `armadillo.fs.glsl` to further color the armadillo fragments green when in close proximity to the sphere, as illustrated in the figures below. One simple way is to check if a fragment is within a specified distance to the sphere, and if it is, set its color to green.



Hint: You should use the appropriate uniform variable in the armadillo shader.

(d) **30 pts** Assemble the fancy Coronavirus. The real coronavirus is decorated with multiple "spikes." In this part you will assemble the spikes on the spherical virus body, and make the spikes move with the sphere. You will modify `A1.js` to attach a number of spikes to the sphere to form a realistic looking virus. A function called `generateSphereSampling`, located in 'js/setup.js', is provided to help you position each instanced spike around the sphere. Study the code used for adding the spike to the scene and replace the provided transforms with transforms generated by `generateSphereSampling`. You will also have to modify the spike vertex shader (in `spike.vs.glsl`) to move the spikes with the virus, using the `virusPosition` uniform.

Part 2: Creative License (Optional)

You have many opportunities to unleash your creativity in computer graphics! In this **optional** section, and you are invited to extend the assignment in fun and creative ways. We'll

highlight some of the best work in class. A small number of exceptional contributions may be awarded bonus points. Some possible suggestions might be:

- deform the vertices in response to the virus, or as a function of time.
- explode the armadillo or virus along face normals.
- animate colors, lights, in fun ways.
- add interesting objects to the scene.

2.1 Hand-in Instructions

You must write a clear README.txt file that includes your name, student number, and CWL username, instructions on how to use the program (keyboard actions, etc.) and any information you would like to pass on to the marker. Create a folder called "A1" under your "cs314" directory. Within this directory have two subdirectories named "part1," and "part2", and put all the source files, your makefile, and your README.txt file for each part in the respective folder. Do not use further sub-directories. The assignment should be handed in with the exact command:

`handin cs314 a1`
on a department computer, which you can SSH into.
You may also use `Web-Handin` by following this link https://my.cs.ubc.ca/docs/hand-in, logging in with your CWL credentials, and writing cs-314 for the course, A1 for the assignment name, and zipping your assignment folder for submission.
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 Web-Handin, or using the -c flag on the command line `handin -c cs314 a1`.