CPSC 314
Assignment 1: Hello Armadillo! Introduction to Three.js, WebGL, and Shaders

Due 11:59PM, September 24, 2024

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 magical "Orb" that it interacts with. 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. Your task for this assignment will be to write simple shaders to move the Orb around, turn it on to illuminate the armadillo, detect how close it is to the poor Armadillo, and make it interact with the armadillo's body.

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/CPSC314-2024W-T1/a1-release
```

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 lightbulb 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:

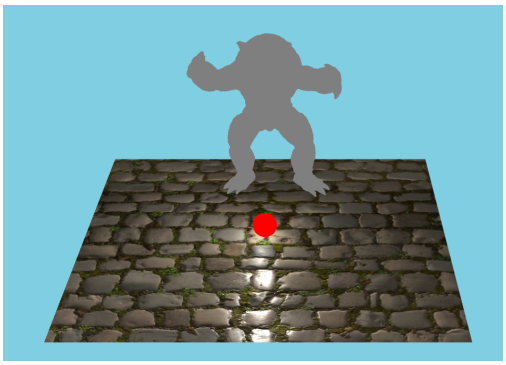
```
XMLHttpRequest cannot load... Cross origin requests are
   only supported for protocol schemes: http, data, https.
```

We highly recommend that you run a local server, instead of changing browser security settings. For example you can do this with VS Code:

1. Follow the link https://code.visualstudio.com/Download to download and install VS Code.
2. Open VS Code and install the Live Server extension. You may also install it from here: https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
3. In VS Code, open the assignment's root folder.
4. Open `A1.html`, right-click in the editor, click "Open with Live Server".

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 script `js/setup.js` creates the basic scene with the floor, and provides a utility function for loading 3D models. The initial configuration should look as it does in the figure below.

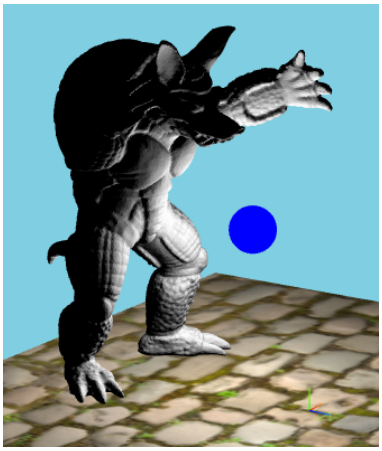


Part 1: Required Elements

(a) **20 pts** Moving & Coloring the Orb. The shape of the Orb is represented by `SphereGeometry`, and manipulated using the vertex shader `sphere.vs.glsl`. The variable `orbPosition` (the position of the orb 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 blue 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** Lighting the Armadillo.

The light from the orb should light up the armadillo. Here you will implement a simple model of how light from the orb would interact with the armadillo, a simple shading model called "Gouraud shading." We will study more realistic models later in the course. Modify `A1.js` and `armadillo.vs.glsl` to color each vertex 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 orb will be "activated", lighting up different parts of the armadillo as it's moved around, as illustrated in the figure below.



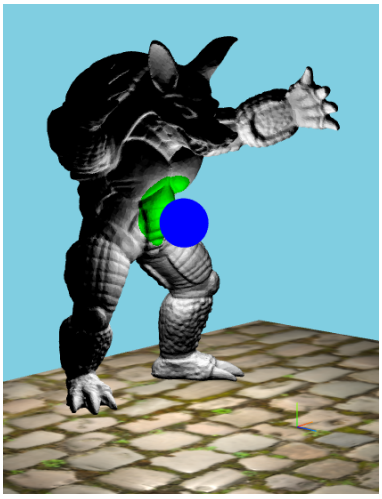
Hint 1: See how uniforms are passed to the sphere shader.

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** Proximity detection.

The armadillo has sensors on its skin that can detect objects in close proximity. 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 an armadillo 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** Body Deformation.

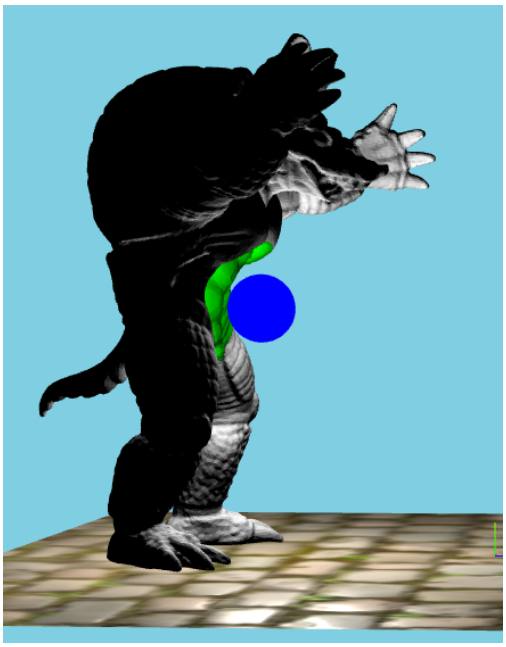
In this part you will indent the armadillo's mesh when pushed in by the Orb, as illustrated in the figure below. This is a preview of how vertex shaders can be used for changing a shape. For this you will need to change `armadillo.vs.glsl` and `A1.js`. One simple way is to check if a vertex is within the Orb, and if it is, move the vertex to the surface. You should pass the necessary information about the Orb to the armadillo shader.

Hint 1: See how uniforms are passed on the sphere shaders.

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:

- explode the armadillo or orb along face normals.



- animate colors, lights, in fun ways.
- add interesting objects to the scene.

3 Submission Instructions

3.1 Directory Structure

Under the root directory of your assignment, create two subdirectories named "part1" and "part2", put all the source files and everything else required to run each part in the respective folder. Do not create more sub-directories than the ones already provided.
You must also write a clear `README.txt` file which 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. Place the file under the root directory of your assigment.

3.2 Submission Methods

Please compress everything under the root directory of your assignment into `a1.zip` and submit it on Canvas. You can make multiple submissions, but we will grade only the last one.

4 Grading

4.1 Point Allocation

Each assignment has 100 points for Part 1. Part 2 is optional and you can get bonus points (0-10 points) at the instruction team's discretion. The max score for each assignment is 110 points.

4.2 Face-to-face (F2F) Grading

For each assignment, you are required to meet face-to-face with a TA during lab hours to demonstrate that you understand how your program works. Details regarding when and how will be announced on Canvas and on Piazza.

4.3 Penalties

Aside from penalties from incorrect solution or plagiarism, we may apply the following penalties to each assignment:
Late penalty. You are entitled up to three grace (calendar) days in total throughout the term. No penalties would be applied for using them. However once you have used up the grace days, a deduction of 10 points would be applied to each extra late day. Note that

(a) The three grace days are given for all assignments, **not per assignment**, so please use them wisely;

(b) We check the time of your last submission to determine if you are late or not;

(c) We do not consider Part 1 and Part 2 submissions separately. Say if you submitted Part 1 on time but updated your submission for Part 2 one day after the deadline, that counts one late day.

No-show penalty. Please sign up for a grading at least one day before F2F grading starts, and show up to your slot on time. So a 10-point deduction would be applied for not showing up at your grading slot.
If you cannot make it to F2F grading, please contact the course staff on Piazza before the sign-up closes. Also, please note that

(a) we would not apply the no-show penalty only if you are unable to show up on time because of a personal health emergency, and in such cases we would like to see a written proof of the situation.

(b) In the past some students reported that they got their names overwritten by others, or their names disappeared mysteriously due to technical glitches. Therefore we strongly advice students to take a screenshot of your slot after sign-up just to prove you have done it.