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

Due 11:59PM, January 21, 2026

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.

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, his boxing gloves, 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 course's GitHub repository:

`https://github.students.cs.ubc.ca/CPSC314-2025W-T2/a1-release`

Students registered in this course should be able to access the repo by logging in with their CWL ID. If you have problems with logging in, please head to the CS departmental account

setup page (https://www.cs.ubc.ca/getacct/), log in with your CWL ID and check if your student account has been activated. And if you still have problems, please create a private post on Piazza so the teaching team can have a look into the issue.

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:

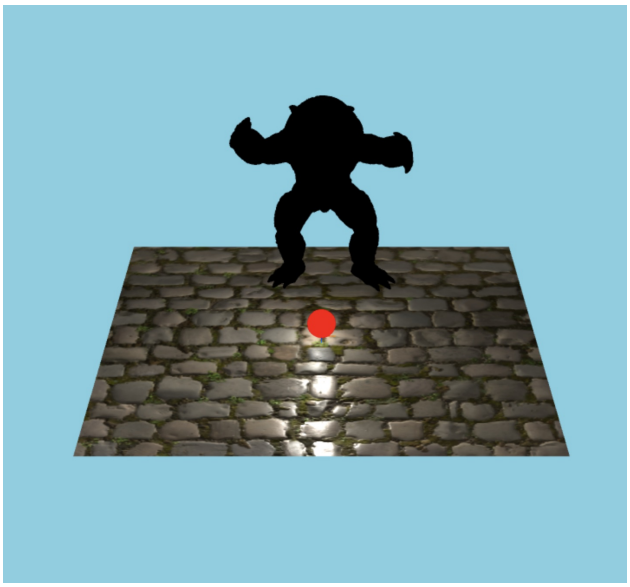
```
XMLHttpRequest cannot load... Cross origin requests are
  only supported for protocol schemes: http, data, https.
```

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.



(a) **15 pts** Moving & Coloring the Orb and Boxing Gloves.

For the boxing gloves, look at how the `loadAndPlaceOBJ` function is used to place the armadillo in the scene, and use it to load the boxing gloves onto the scene modifying the position, rotation and scale of the boxing gloves to fit the armadillo's hands. This may require some manual trial and error. You can use the `boxingGloveMaterial` to color the boxing gloves.

For 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 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) **15 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) **25 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 cyan when in close proximity to the sphere, as illustrated in the figures below. One simple way is to check if an armadillo material fragment is within a specified distance to the sphere, and if it is, set its color to cyan.

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

(d) **30 pts** Body Deformation.

In this part you will make the Orb behave like a strong magnet, attracting and deforming the parts of the armadillo's mesh that get close to 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 away from the surface to the orb. You should pass the necessary information about the Orb to the armadillo and glove shaders.

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

(e) **15 pts** Feature Extension. In this part, you task is to extend the assignment to add features of your own choosing. The goal is to encourage you to explore the capabilities of Three.js and WebGL, and more importantly, unleash your creativity! The extension has to be non-trival to receive credit, but for full credit, the features have to be truly creative. See Sec. 4.2 for grading expectations. A small number of exceptional extensions will be selected from this group and shown in class, with the student's permission. Here are some ideas to get you started, but feel free to come up with your own:

- **Procedural Portal System:** Create inter-dimensional portals that render the scene from different viewpoints, with seamless teleportation mechanics.
- **Dynamic Fluid Simulation:** Implement a real-time fluid effect where the orb creates rippling water/energy waves across the floor that interact with the armadillo's movement.
- **Interactive Sound Visualization:** Create a system that analyzes audio input and generates synchronized 3D visualizations, with geometry that dances to music frequencies.
- **Physically-Based Destruction:** Implement a destruction system where objects can be shattered into realistic fragments with simple physics simulation and debris interaction.
- **Interactive game mechanics:** Design a simple game around the armadillo and orb, such as a collection game where the armadillo must gather items while

avoiding obstacles, with scoring and levels.

First duplicate your current work into a new directory named 'part2', then implement your feature in this new directory. Write a brief description of your feature in the README file. You will be graded on both how it works and how well you can explain your feature.

3 Submission Instructions

3.1 Directory Structure

Your submission should contain two subdirectories - the first should be named 'part1' and should contain all parts except part e (no feature extension), the second subdirectory should be named 'part2' and should contain your feature extenstion. Under the root directory of your assignment, please add both subdirectories including 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 Face-to-face (F2F) Grading

Submitting the assignment is not the end of the game; to get a grade for the assignment, you must meet face-to-face with a TA in a 12-min slot during or outside lab hours, on Zoom, to demonstrate that you understand how your program works; Grading slots and detailed instructions on how to sign up and conduct F2F will be announced on Canvas and on Piazza. During the meeting, the TA will (1) ask you to run your code and inspect the correctness of the program; (2) ask you to explain parts of your code; (3) ask you some questions about the assignment, and you will need to answer them in a limited timeframe. The questions will mostly be based on ThreeJS or WebGL concepts that you must have come across while working on the assignment; but also you may get conceptual questions based on lecture materials that are relevant to the assignment, or technicalities that you may not have thought about unless you really "digged deep" into the assignment. But no need to be nerveous! We evaluate your response based mainly if not entirely on the coherence of your thoughts, rather than how complete or long your response is: e.g. if the full answer to a question includes A+B+C+D and you mentioned A+B only in the provided time, but your thread of thought is logical, then you may still get full marks.

4.2 Point Allocation

All questions before Feature Extension have a total of 85 points. The points are warranted based on

- The functional correctness of your program, i.e., how visually close your results are to expected results;
- The algorithmic correctness of your program, e.g., applying transformation matrices in the right order;
- Your demonstrating your understanding of the code with your answers to TAs' questions during face-to-face (F2F) grading.

The feature extension grade is based on creativity and sophistication of your submission. We warrant partial marks on four scales:

- 15: Outstandingly creative and sophisticated feature extension, e.g. the suggestions provided in Sec. 2 e. But note that only a very small number of extensions may get this score, and we cannot guarantee that if you implement one of the suggestions as is then you will definitely get full marks;

- 10: Average creativity and sophistication, e.g. adding a few interesting objects into the scene and animating, deforming the objects in visually appealing ways;
- 5: Basic creativity and sophistication, e.g. adding a few static objects into the scene, changing the orb's shape in simple ways;
- 0: Trivial or no feature extension, e.g. just changing the color of the orb.

4.3 Collaboration Policy

Submissions must be individual, but you are encouraged to discuss the assignment with your classmates, and you are allowed to use generative AI tools, e.g. ChatGPT, Claude to help with constructing your solution. **In the `README.txt` file, include names of the people you discussed the assignment with, websites you acquired information from, and if you used generative AI tools, include the names of the tools you used.** And you must understand your code and be able to explain them during F2F grading. If you cannot demonstrate at least a basic understanding of your code during face-to-face grading, you may receive a zero for the assignment—even if your code runs perfectly. Hiring a tutor to write your assignment, or copy-pasting others' code without proper referencing in `README.txt` are considered as plagiarism, and will be reported to the dean's office.

4.4 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.

No-show penalty. Please sign up for a grading slot on the provided sign-up spreadsheet (link will be posted later on Piazza) before the submission deadline of the assignment, and show up to your slot on time. A 10-point deduction would be applied to each of the following circumstances:

(a) Not signing up a grading slot before the sign-up period closes. Unless otherwise stated, the period closes at the same time as the submission deadline.

(b) Not showing up at your grading slot.

If none of the provided slots work for you, or if you have already missed your slot, contact the course staff via Piazza or Canvas to arrange a late grading slot. Also, please note that

(a) you'll need to explain to your TA why you're getting graded late, and may be asked to present documents to justify your hardship. The TA may remove the no-show penalty as long as he/she deems the justification to be reasonable.

(b) In the past some students reported that their names disappeared mysteriously due to technical glitches, and the spreadsheet's edit history has no trace of it. So double check that your name is on the sign-up sheet after you sign up by refreshing the page.