CPSC 314
Assignment 2: Transformations

Due 11:59PM, February 14, 2022

1 Introduction

In this assignment you will utilize your knowledge of transformations to make things move. We will take a look at how to build and animate object hierarchies. As for our subject, we continue on from Assignment 1 with our armadillo.

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/cpsc314-2021w-t2/a2-release.git
```

1.2 Template

- The file `A2.html` is the launcher of the assignment. Open it in your preferred browser to run the assigment, to get started.
- The file `A2.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 sphere 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 `A2.html` in any modern browser. However, most browsers will prevent pages from accessing local files on your computer. If you simply open `A2.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:

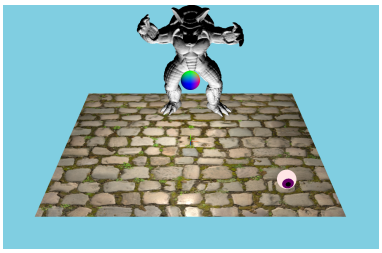
https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

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 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. For all parts, it will be helpful to look at Three.JS's documentation https://threejs.org/docs/, and lecture materials on scene graphs and hierarchies.

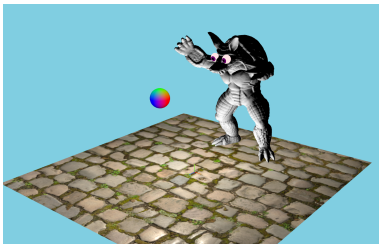


Part 1: Required Elements

(a) **15 pts** Adding eyes

For the first part of the assignment you will give the armadillo eyes. In the code you'll see that there are two eyeball meshes have been constructed. Your task is to add them to the scene and ensure their size and position is roughly as depicted in the image below.

Hint 1: This part can be done entirely in `A2.js` using the scene graph.



(b) **25 pts** Tracking the orb

The next step is to give our armadillo a moving target to look at with its new eyes. Make the sphere "orb" in the scene move by responding to keyboard input (WASD can be used for horizontal movement and QE can be used for vertical movement). Then point the eyes towards the orb to track its motion. Add code to the update function to adjust the transformations when the orb moves.

Hint: Remember that biological eyes are a lot like cameras. `THREE.Matrix4` and `THREE.Object3D` have a method called `lookAt` that may be of use.

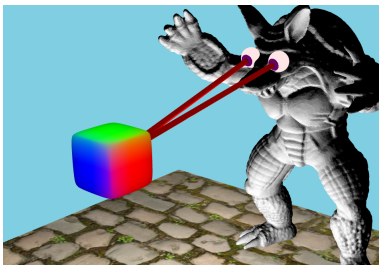
(c) **25 pts** Orb Throb

Add some deformation to the orb by transforming the vertices of the sphere orb. First make the orb "throb" by scaling the position of the vertices so that the orb's size changes sinusoidally with time. You can pass the time as a uniform variable to the shader for the orb. Next make the orb's shape change as it throbs. You may come up with and implement any interesting deformation for this part. There are only two requirements:

- The size of the orb must change sinusoidally with time.
- The deformation must be a function on the vertice's position on the sphere.

This part will require changes in shader code and `A2.js`

The following is an example of a deformation. This deformation changes the orb into a cube sinusoidally over time before changing back to an orb.



(d) **35 pts** Lasers

Now that our armadillo can see the dangerous looking orb, let's make it Superman. Make the armadillo shoot lasers out of its eyes at the orb, when the orb comes close enough (distance less than `LaserDistance`).

You will use the provided laser's geometry (a cylinder) and its material to build the laser. You can do this by appropriately scaling each eye's laser along the cylinder's axis to make it long enough to reach the orb. This can be done by exploiting Three.js's scene graph, and adding the laser as a child of the eye, with the proper scaling. Add code to the update function to adjust the length of the laser when the orb moves, and to turn it on/off.

Hint: Use Three.js' Matrix and Vector classes and associated functionality to define the transformation that will achieve the desired effect. E.g., Vector3 has a convenient `distanceTo` function, but make sure that all coordinates are already transformed to the same coordinate frame!

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:

- It's almost Valentine's day! Turn the orb into a heart when hit by the laser
- Or perhaps make the orb explode or melt when hit with the laser
- Animate the armadillo to reach the orb, and help the orb escape
- Add other objects to the scene that are animated
- Make a game out of all this!

3 Submission Instructions

3.1 Directory Structure

Under the root directory of your assignment, create two subdirectories named "part1" and "part2", and 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 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. Place README.txt under the root directory of your assigment.

3.2 Submission Methods

Please compress everything under the root directory of your assignment into `a2.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. Percentage wise, we use Part 1's total points as the denominator: e.g. if you get 95 out of 100 points from Part 1, but no points from Part 2, then your percentage grade would be 95/100. If you get full points from both Parts, then your percentage grade would be 110/100.

4.2 Face-to-face (F2F) Grading

For each assignment, you are required to meet face-to-face with a TA on Zoom or in person to demonstrate that you understand why your program works. Details regarding how to sign up a grading session with a TA 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 consider the time of only your last submission;

(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, we would count one late day.

No-show penalty. You are required to sign up a grading slot at least one day before F2F grading starts, and show up at your slot on time. So 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;

(b) Not showing up at your grading slot.

Note that we would not apply the penalty if you are unable to sign up/show up on time due to an emergency, or if you cannot sign up because none of the slots work for you. In those cases, please contact the course staff immediately on Piazza.