CPSC 314
Assignment 2: Transformations

Due 11:59PM, October 22, 2021

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-t1/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:

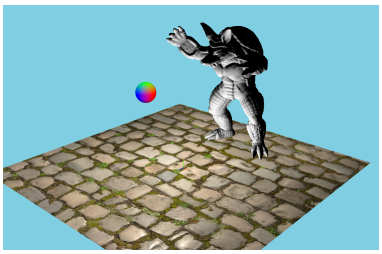
https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

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.

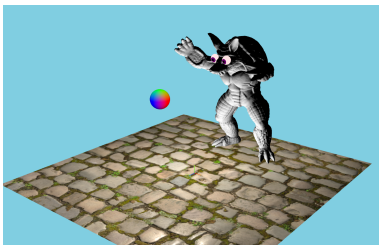


Part 1: Required Elements

(a) **15 pts** Adding eyes

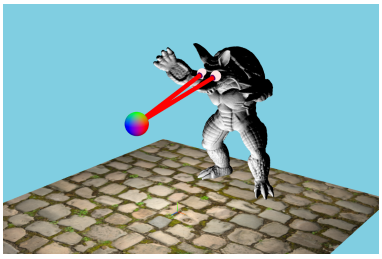
For the first part of the assignment you will give the armadillo eyes. In the code you'll see that there are two eyeball meshes provided. Your task is to add them to the scene and ensure their position is roughly as depicted in the image below.

Hint 1: This part can be done entirely in `A2.js`



(b) **15 pts** Lasers

Now that our armadillo can see, let's make him Superman. You will use the provided cylinder geometry, along with the beam materials, to make the armadillo shoot lasers out of its eyes at the orb. You should use three.js' Matrix class and its associated functionality to define the transformation that will achieve the desired effect, then pass that Matrix (as a uniform) into the shaders where it will be applied to the model. *Hint 1:* `THREE.Matrix4` and `THREE.Object3D` have a method called `lookAt` that may be of use.



(c) **35 pts** Tracking. The next step is to give our armadillo a moving target and the ability to hit it. 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). Add an update function which will adjust the transformation of the lasers so that they are continuously pointing from the eye to the orb.

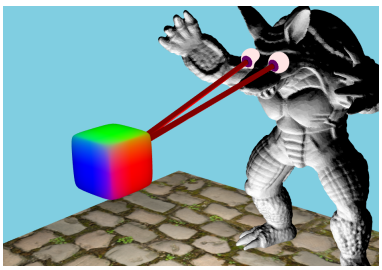
(d) **35 pts** Orb Deformation

Add some deformation to the orb by transforming the vertices of the sphere orb. Scale the position of the vertices so that the orb changes sinusoidally with time. You can pass the time as a uniform variable to the shader for the orb. You may come up with and implement any interesting deformation for this section. 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.



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:

- Animate other body parts of the armadillo
- Detect if the armadillo actually reached the ice cream and give him a reward

- Add other objects to the scene that are animated
- Make a game out of all this!

3 Hand-in 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, your makefile, 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

You can choose one of the two ways to submit your assignment: (a) running `handin` command on a departmental server, or (b) submitting through `Web-Handin`.

3.2.1 Submit from departmental server

1. SSH into a departmental server.
2. Create a directory named "cs-314" under your home directory if you have not yet done so.
3. Create a folder called "a2" under `cs-314/`.
4. Upload everything under the root directory of your assignment to "a2".
5. Run the exact command: `handin cs-314 a2`.
6. Check if the submission was successful: run `handin -c cs-314 a2`.

3.2.2 Submit via Web-Handin

1. Compress everything under the root directory of your assignment into `a2.zip`.
2. Log into `Web-Handin` with your CWL credentials, by following this link https://my.cs.ubc.ca/docs/hand-in
3. Write "cs-314" for the course name, "a2" for the assignment name.
4. If you're trying to overwrite a previous submission, check the box "Overwrite previous".

5. Upload the zip file.
6. Click "Handin assignment".
7. Check if the submission was successful: use the `Check submissions` button.