CPSC 314
Assignment 2: Transformations and Skinning

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 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 `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:

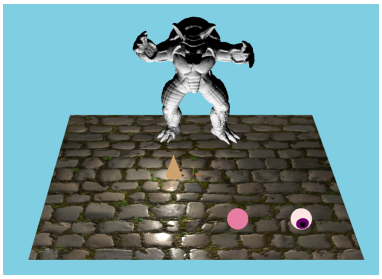
https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

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** Making Some Ice Cream.

Notice there are now more objects for us to play with. Ignoring the eye balls for now, take a look at the cone and sphere we have. Use Three.JS to combine the cone and sphere to make an ice cream cone. Make sure that the cone and sphere move together when controlled. You can do this using hierarchies.

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

(b) **15 pts** Animation

Now that we have a nice looking ice cream cone in our scene, lets animate it. Make the ice cream float around in a figure-eight ∞ symbol. Edit the `update` method and use the variable `time` which has been provided to do this.

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

(c) **25 pts** Eyeballs. Your task here is to bless our Armadillo with a pair of eyes. One eyeball model is already provided. Your job is to place two copies of this eyeball onto

where the eyeballs should be on the Armadillo using an appropriate transformation. Finally, make the eye balls track the ice cream.

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

Hint 2: `THREE.Matrix4` and `THREE.Object3D` have a method called `lookAt` that may be of use.

(d) **15 pts** Armadillo Controls Let us add controls to the armadillo. Add controls to move the armadillo forward and backwards using the keys 'W' and 'S'. Additionally, add controls to make the armadillo rotate left and right, forwards and backwards, using the keys 'A', 'D', 'Q', and 'E'. armadillo as it moves or rotates. Be careful to transform the armadillo's normal vectors properly so the lighting updates as well.

Hint 1: Use the provided `armadilloFrame` in A2.js to transform the armadillo.

(e) **30 pts** Skinning.

Use skinning to make the armadillo always keep its legs planted on the ground even while rotated. You can do this by applying the rotation in the previous section to only the part above the hip (commonly known as "pelvis" in graphics). Be careful to transform each vertex by some weight depending on its height above the pelvis. A good pelvis height is defined in `armadillo.vs.glsl` as `PELVIS_HEIGHT`.

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

Hint 1: Use the provided `rotationMatrix` in A2.js and `armadillo.vs.glsl` to rotate each vertex above the pelvis.

Hint 2: You might find it useful to use `rotationFrame` to make sure the armadillo's eyes follow its body properly.

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
- Make the ice cream more interesting
- 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.