CPSC 314
Assignment 2: Fun with Transformations

Due 23:59:59, Oct 21th, 2016

1 Introduction

In this assignment you will create, articulate and animate an octopus character.
The main goal of this assignment is to practice rotation, translation and scaling transformations and transformation hierarchies. You will also get better understanding of how posing and character animation are implemented in typical graphical setups.
You are provided with a template code that has the rendering environment and a portion of the octopus already built in. There are three main tasks to be done: compete the octopus geometry, animate its tentacles, and add some further creative components to the scene. To perform those, you will need to create, apply, and update a set of transformation matrices in javascript without three.js matrix creation helper functions. Note that the overall transformation matrices should remain relatively simple.

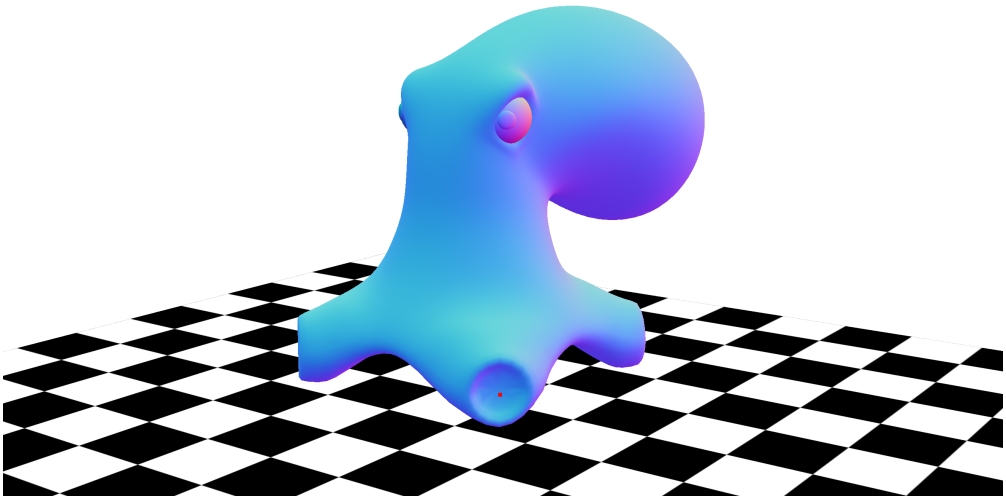

Figure 1: The provided template.(The red dot represents the position of the tentacle socket.)

The template code is very similar to the previous assignment template. The main differences are in the javascript file:

- The dragon is no longer present. Instead, a part of an octopus has been constructed for you. There is the head, (placed with respect to the world coordinate frame) two eyes, (placed with respect to the head coordinate frame) and two pupils, (placed with respect to each eye coordinate frame).
- A new incomplete function called `updateBody` is provided. It is called every frame and you should complete it to animate the octopus.

2 Important Rules

Three.js provides several tools for easy parenting, rotation, translation and scaling of objects. Yet, as the purpose of this assignment is to understand the principles behind all of this methods, **you are not allowed to use them**. More specifically, you must:

1. Explicitly declare new matrices using the `Matrix4().set` method. Creating matrices through basic operations on other existing matrices is also permitted (eg: multiplication). Generating matrices through utility methods such as `Matrix4().makeRotationX` is not allowed.
2. Objects must be moved by changing their coordinate frame using the `object.setMatrix` method. Operations on their `position`, `rotation` and `scale` attributes are not allowed.
3. Explicitly parenting objects using the `parent` attribute is not allowed.

You can find examples of matrix initialization and object positioning as described above in the template code.

3 Work to be done (100 pts)

First, ensure that you can run the template code in your browser. See the instructions from the previous assignment. **Remember to follow the requirements described in the previous section.** Study the template to get a sense of how it works.

1. Basic Tasks (70 points)

   (a) **15 pts** Build tentacles.
   Add four tentacles to the octopus. Each tentacle should consist of two or more links, and be placed with respect to the head coordinate frame. Use the tentacle socket red dot placement as a cue. Note that the tentacle placements are symmetric around the head coordinate frame - use this observation to avoid replicating code for each tentacle and place tentacles algorithmically to gain extra points.
   Potential link geometries have already been built for you. You can use those geometries or create your own. You do not need to write your own shaders because you can re-use the default material that has already been applied to the torso, neck and head. You can create cylinders with `THREE.CylinderGeometry` *constructor.*

(b) **35 pts** Animate the octopus.

Rotate the link coordinate frames so that their motion simulates swimming in place (see video). Have each link change its orientation with respect to its parent. Make sure the swimming animation is cyclical and continues to move back and forth. Also, make sure the motion speed allows the observer to see the animation details. You should use a timer to keep the animation speed consistent instead of animating the tentacles every frame.

The variable t has been declared for you already and contains total elapsed program time. Obviously, this value will always increase and we need a cyclical animation. You will, therefore, need a function to turn a constant value into a cyclical one and use it to change the orientation and position of the objects you want to animate.

(c) **20 pts** Add poses.

Produce 3 more different static poses for your octopus (for instance make it scratch its ears). Include head and/or eye motion in your poses. Pressing each number key between 0 and 2 should show a different pose. You can move and rotate the head, the tentacles, and the eyes.

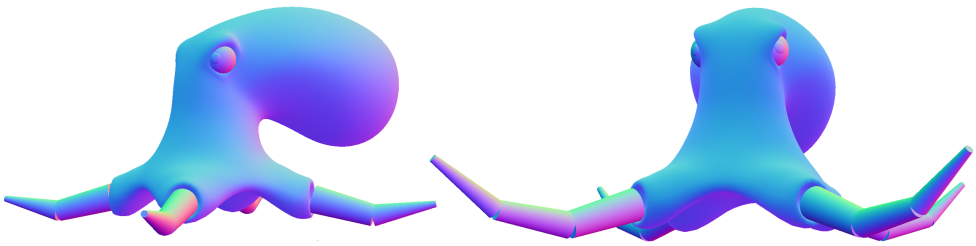

Figure 2: Example octopus poses.

2. Creative License (30 pts)

Here it is time to unleash your creativity and explore a larger range of interesting animations. Your idea should be of a similar complexity to the previous tasks. If you have any doubts, make sure to OK it with the prof or a TA. Some possible suggestions might be:

- Make vertex shaders for some of the moving objects and make use of their dynamic coordinate frames (defined in glsl code as `modelMatrix`).
- Instead of the 4 tentacle head loaded from the file `Octopus_4_A.obj` use the 8 tentacles octopus head provided in an `.obj` file named `Octopus_8_A.obj` and create/animate all the tentacles without replicating your code eight times.
- Add circular joints or extra links to the tentacles or more details to the scene.
- Create a partner or kids for your octopus.

Figure 3: 8 tentacle octopus template.

Bonus marks may be given at the discretion of the marker for particularly noteworthy explorations.

3.1 Hand-in Instructions

Handin is a command-line program that the CS department use to process assignments. It exists as a web form and as a command-line utility accessible from any undergrad or department Linux machine. That is, if you're not using the web form, you must either SSH into an undergraduate server (eg. remote.ugrad.cs.ubc.ca), or use one of the CS lab computers to login to a Linux environment and use handin from the command-line.
You must be able to access your undergraduate CS account. All students who are registered in a CPSC course in one of these two winter terms will have access to an undergraduate account. (Graduate students automatically get one as well.) You must re-activate this account every year, please do so at https://www.cs.ubc.ca/getacct/ .
The command-line command for using handin is `handin course assignment` For the second assignment, the command is `handin cs314 a2`. You must have a folder in your home directory named cs314, and a subfolder named a2 which contains the directory tree for the assignment. Place the corresponding source files in these folders, together with a README.txt that includes you name, student number, cs ID and any information you would like to pass on to the marker. Do not include any files in that directory that are not part of the assignment (e.g. do not include the example video). Everything inside this folder will be submitted when you run the handin command.