CS 174C: Assignment 1

Due Date: Check on BruinLearn

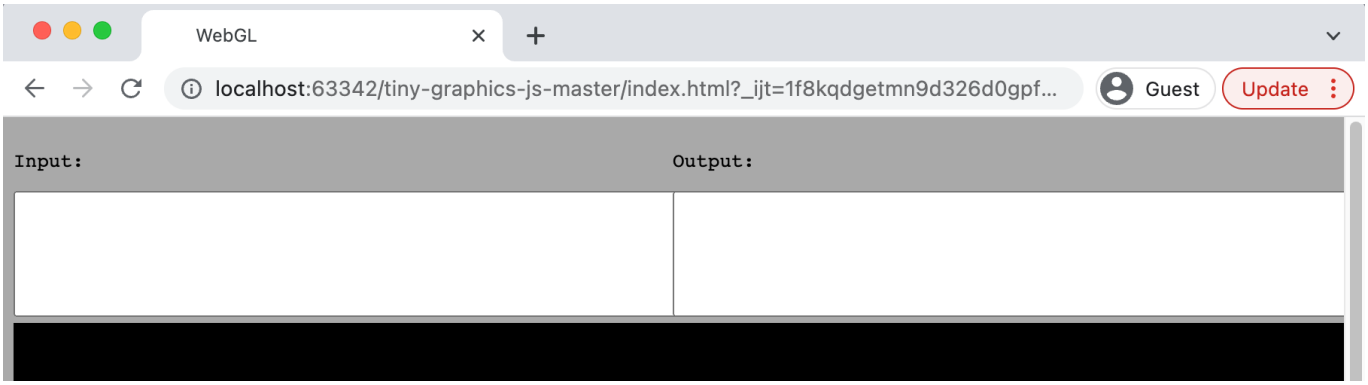
General Requirements

Your code must be implemented using the template code provided on BruinLearn. Make sure to implement all the **text-based commands** and **button events** specified below, as this is how we will test your code.

Create an introduction file named "README.md" (or in another format if you wish) and specify where you implement the bullet points (like "1-2: in line XXX-XXX", or "2-6: you should see XXX during animation"). This will help the grader to check your code and not miss a point.

Hermite Spline Modeling [20 points]

The code for this assignment should be written in `assign_one_hermite.js` (you can implement the spline and helper classes in another file, but if you do so please specify this in your README document). All input/output behaviors should be implemented using the top input and output text boxes, and operations triggered by clicking buttons.



Implement a Hermite spline class adhering to the following requirements:

1. [3 points] Your **Hermite spline** class should be able to handle at least **20** control points.
2. [3 points] Use a single global parameter ($t \in [0,1]$) for the entire spline. For n control points, the spline consists of $n-1$ Hermite segments, and control point P_i corresponds to $t_i = i/(n-1)$.
3. [3 points] Manipulation of the control points and tangents using the shell-like text interaction. Implement the following commands: (Click the "Parse Commands" button to run all commands in the input text box, and multiple commands should be correctly handled in multiple lines)

```
> add point x y z sx sy sz
```

Adds a point at position (x, y, z) with tangent (sx, sy, sz) at the end of the spline (this point should correspond to the parameter value t = 1).

```
> set tangent index sx sy sz
```

Set the components of the tangent specified by index.

```
> set point index x y z
```

Set the components of the point specified by index.

4. [3 points] Arc length parameterization using a piecewise linear approximation and a look-up table. The following shell command should print the (close estimate of) arc length of a spline, and a look up table with a reasonable number of rows. (Print to output)

```
> get_arc_length
```

5. [2 points] Load a spine from the input raw text and export the current spine to the output raw text. Read from the input text when the "Load" button is clicked and export to the output text when the "Export" button is clicked. Your code should be able to read and restore the previous session (spline) by "Load" from the previous "Export" output.

Implement exactly the following format for the spline text:

```
n
c_x1 c_y1 c_z1 t_x1 t_y1 t_z1
c_x2 c_y2 c_z2 t_x2 t_y2 t_z2
....
c_xn c_yn c_zn t_xn t_yn t_zn
```

where the prefix c_ refers to control points and the prefix t_ to tangents and n is the number of control points defined in the text.

6. [2 points] Preset drawings (button events) by implementing two buttons that generate preset splines and immediately draw them (i.e., overwrite the current spline data and update the scene on click).
    - Button "Straight!" : Create a straight line using 4 control points.
    - Button "Circle!" : Create a circle-like spline that appears as smooth as possible.
7. [4 points] Spline drawing. Update the scene after a button named "Draw" is clicked. The quality of the drawing also counts. Use sufficient sampling (e.g., ≥ 30 samples per segment) so the curve appears smooth. The rendered curve should not show obvious artifacts (e.g., jaggedness, kinks at joints, or unintended large overshoots caused by incorrect parameter/tangent handling).

Advice

1. Make sure you follow the exact syntax specified in the assignment. We will be using the same set of scripts to test your code.
2. Your code should use 0-indexing, which means indices start at 0.
3. Control points may be unevenly spaced in 3D even though *t* is uniform; tangent magnitude strongly affects overshoot.
4. A lot of code is already provided. You do not need to write your own vector libraries; however, if you already have one, you may use it.
5. You may NOT use existing code for the splines.
6. Do not use spaces in file names.
7. Test your code using an up-to-date Chrome browser.
8. **If your code doesn't compile or doesn't run, you will get 0 points.**
9. **If your code doesn't parse the input text correctly, you will get 0 points.**
10. You may want to implement additional commands to facilitate debugging.
11. Start the assignment right away and work incrementally!