National University of Singapore
School of Computing
CS1101S: Programming Methodology (JavaScript)
Semester I, 2012/2013

Mission 5
Curve Manipulation

Start date: 29 August 2012
Due: 03 September 2012, 23:59

Readings:

- Textbook Sections 1.3 to 1.3.1

Background:

Grandmaster Martin is pleased that the disciples are putting in their effort to learn the Force. He prepares them for their next lesson and cautions them that this is the only beginning. They have to be mentally prepared to deal with the increasing difficulty as the training of curves proceeds.

Grandmaster Martin begins the second lesson by presenting some recursive problems as warm-up exercises. He then demonstrates some difficult curve manipulation techniques. He instructs his disciplines to follow him and practice as he demonstrates.

After some practice, the disciples begin to get the hang of curve manipulation. Grandmaster Martin is pleased and encourages them further:

"Yes and now I can feel that you have some control over your curve drawing. Now try to imagine it floating around in that empty space of your mind, twisting and warping, forming a connected image..."

This mission has **two** tasks.

Task 1:

In addition to the direct construction of curves such as `unit_circle` or `unit_line`[1], we can use elementary Cartesian geometry in designing JavaScript functions which *operate* on curves. For example, the mapping $(x, y) \longrightarrow (-y, x)$ rotates the plane by $\pi/2$ (anti-clockwise), so the following code

```
function rotate_pi_over_2(curve){
    return function(t){
               var ct = curve(t);
               return make_point(-y_of(ct),
                                 x_of(ct));
           }
}
```

[1]the curves unit_circle and unit_line have been defined in hi_graph.js

defines a function which takes a curve and transforms it into another, rotated, curve. The type of `rotate_pi_over_2` is

$$\text{Curve-Transform} : \text{Curve} \to \text{Curve}.$$

Write a definition of a Curve-Transform `reflect_through_y_axis`, which turns a curve into its mirror image.

Note:

It is actually fine if the curve reflects in the y-axis and disappears from the viewport. To view the effect and the curve in the viewport, you might try `draw_points_squeezed_to_window` or `draw_connected_squeezed_to_window`.

Task Files

- lib/list.js
- lib/misc.js
- lib/graphics.js
- lib/hi_graph.js
- mission_5_1.html
- **mission_5_1.js**

Task 2:

It is useful to have operations which combine curves into new ones. We let Binary-Transform be the type of binary operations on curves,

$$\text{Binary-Transform} : (\text{Curve}, \text{Curve}) \to \text{Curve}.$$

The function `connect_rigidly` is a simple Binary-Transform. Evaluation of `connect_rigidly (curve1, curve2)` returns a curve consisting of `curve1` followed by `curve2`; the starting point of the curve returned by `connect_rigidly(curve1, curve2)` is the same as that of `curve1` and the end point is the same as that of `curve2`. (curve1 and curve2 can be disconnected)

```
function connect_rigidly(curve1, curve2){
    return function(t){
               if(t < 1/2){
                   return curve1(2 * t);
               }else{
                   return curve2(2 * t - 1);
               }
           }
}
```

There is another, possibly more natural, way of connecting curves. The curve returned by `connect_ends(curve1, curve2)` consists of a copy of `curve1` followed by a copy of `curve2` after it has been rigidly translated so its starting point coincides with the end point of `curve1`. The end product is a continuous curve.

It is important to note that, in order to make the starting point of `curve2` coincide with the end point of `curve1`, you can only shift or scale `curve2`, which means that you *cannot* rotate `curve2`.

Write a definition of the Binary-Transform `connect_ends`. It is **recommended** that you use `connect_rigidly` in your `connect_ends` function.

Hint

You may want to use the following functions provided in `hi_graph.js` in your solution. (Note: You are **NOT** required to use both these methods)

- `translate` returns a Curve-Transform which rigidly moves a curve given distances along the x and y axes.
- `scale_x_y` returns a Curve-Transform which stretches a curve along the x and y coordinates by given scale factors.

Task Files

- lib/list.js
- lib/misc.js
- lib/graphics.js
- lib/hi_graph.js
- mission_5_2.html
- **mission_5_2.js**

Submission

To submit your work to the Academy, copy the contents from the template file(s) into the box that says ”Your submission” on the mission page, click ”Save Code”, then click ”Finalize Submission”. Note that submission is final and that any mistakes in submission requires extra effort from a tutor or the lecturer himself to fix.