Mission 5: Curve Manipulation

Start date: 04 September 2017
Due: 09 September 2017, 23:59

Readings:

- Textbook Sections 1.3 to 1.3.1

This mission has **two** tasks.

Task 1:

In addition to the direct construction of curves such as `unit_circle` or `unit_line`, we can use elementary Cartesian geometry in designing Source functions which *operate* on curves. For example, the mapping `(x,y) -> (-y,x)` rotates the plane by `PI/2` (anti-clockwise), so the following program

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

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

`Unary-Curve-Operator : Curve -> Curve`

Write a definition of a `Unary-Curve-Operator`, `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 2:

It is useful to have operations which combine curves into new ones. We let `Binary-Curve-Operator` be the type of binary operations on curves:

`Binary-Curve-Operator : (Curve, Curve) -> Curve`

The function `connect_rigidly` is a simple `Binary-Curve-Operator`. 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: `connect_ends`. The curve returned by `connect_ends(curve1, curve2)` should consist 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.

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

Hint

You may want to use the following function in your solution:

- `translate(x, y)` returns a `Unary-Curve-Operator` which takes in a curve and returns another curve that is identical to the original one except that it is moved to the right by `x` units and up by `y` units.

Submission

To submit your work for this mission, copy the url on your browser and email it to your respective Avengers. Strictly follow to the deadlines set at the start of this file.

IMPORTANT: Make sure that everything for your programs to work is on the left hand side and **not** in the interpreter pane on the right! This is because only that program is preserved in the url you have emailed to us.