Exercises with JS-Eden 2

This activity follows on from *Getting Started with JS-Eden*.

More about canvas 2D features ....

Rectangles

To make a rectange:

```
myRect is Rectangle(20,20,220,100);
```

You will also need to add the rectangle to the canvas. The minimum parameters for a rectangle are x, y, width and height. Rectangle has 2 optional parameters which are similar to the Circle properties: fill colour and outline colour.

Task A: Can you make a simple game that the user controls the position of a circle by clicking on the Canvas? If the circle is inside a rectangle then colour of the circle will be green, else it is red.

Divs (a special kind of text)

The Text observable is for simple labels. For richer text there is the Div:

```
myDiv is Div("<h1>Title</h1><p>Some text</p>", 10,100,300, 200);
```

The basic parameters of a Div are: html, x, y, width, height.

A clever way to use a Div might be to show the value of several observables, e.g.:

```
 a = 7;
 b = 13;
myDiv is Div("<p>a="//a//"</p><p>b="//b//"</p>",10,100,300,200);
```

Remember that double-slash (//) is the string concatenation operator.

Buttons

Another important user interface component is the Button. It is different from other elements in that clicking a button usually involves triggering an action.

To create a button:

```
      myButton is Button("Press me",50,300,true);
```

Parameters: text, x, y, enabled

The first parameter name is required because when there is user interaction with a button then it will update an observable. In the above example, the name is "myButton" and so when the user clicks the button there is an observable "myButton_clicked" that is up-dated. This means we can write a trigger:

```
      proc myButtonAction : myButton_clicked {
            if (myButton_clicked {
                  label = "Button was clicked!";
                  a = a+2;
            }
      }
```

A trigger (or triggered procedure) can be thought of as an agent that constantly watches one or more observables, and when they change iexecutes a change to the state of the model. Recall that ‘=’, unlike ‘is’, assigns a fixed value to an observable.

Consider the following code:

```
buttonState = "Off";

onOffButton is Button(buttonState, 20, 20, true);

proc onOffButton_trigger : onOffButton_clicked {
      if (onOffButton_clicked) {
            if (buttonState == "On") buttonState = "Off";
            else buttonState = "On";
      }
}

picture is [onOffButton];
```

Task B: Create a button that counts the number of times it is clicked. Every time it is clicked, increase the click count by 1. Show the click count on the button.

More exercises

Task C: Create a circle anywhere on the screen with the coordinates x and y and radius r. Use mouseClickX and mouseClickY to fill the circle yellow when the user has clicked inside the circle.

Task D: Create a button that moves to a random position when clicked.

Task E: Can you make a plan of your bedroom including a bed, a table, chairs, windows and doors? Use dependency so that the plan resizes to the size of the canvas.