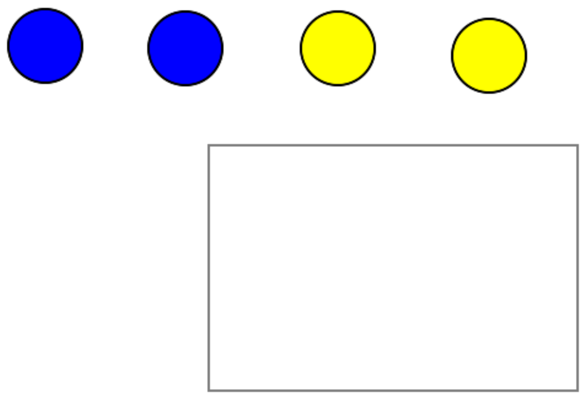
Making Models with JS-Eden

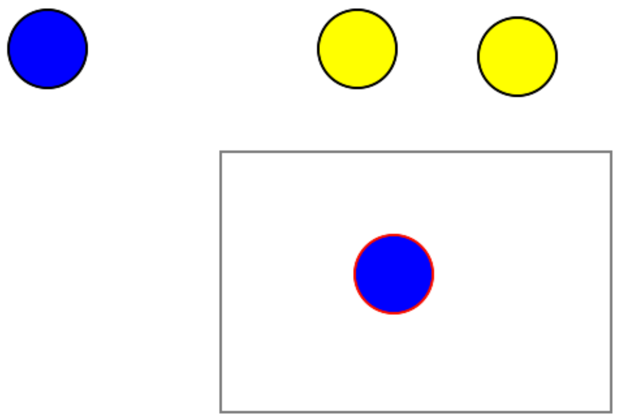
In this activity we will build a game that is suitable for 3-5 year olds. We will use the JS-Eden Canvas: http://harfield.org.uk/jsedencanvas/

The Game

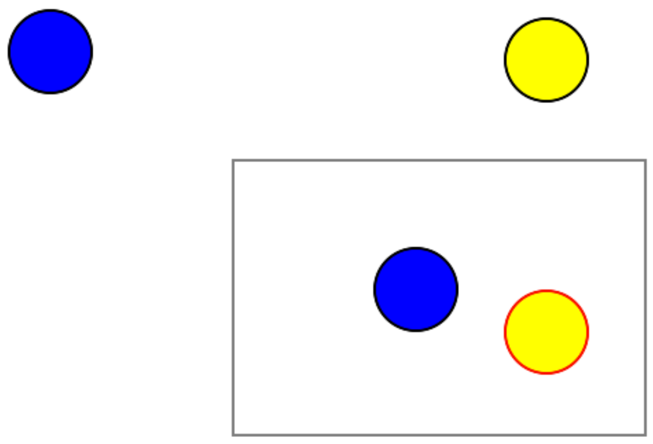
The aim of the game is to place 2 circles of the same colour inside the rectangle. We will set up the UI as follows, with 2 blue circles and 2 yellow circles:



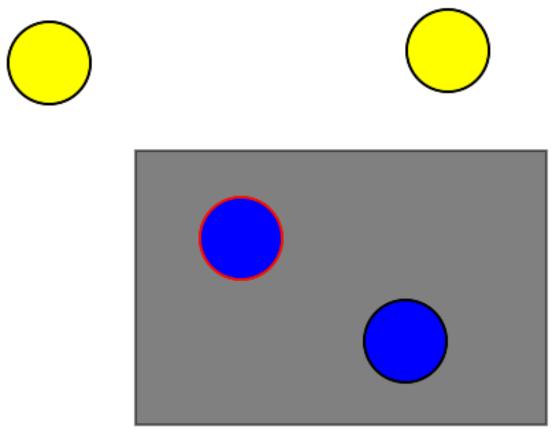
When you click on a circle then that circle will become selected (indicated by a red outline), and when you click into an empty space the circle will move to that point:



You can click on another circle and move it:



If the rectangle contains 2 circles of the same colour and no other circles then the background changes colour indicating that the game is complete:



Building the UI

Start by building the rectangle:

```
rectX = 250;
rectY = 150;
rectWidth = 300;
rectHeight = 200;
rect is Rectangle(rectX,rectY,rectWidth,rectHeight, "white");
canvas is [rect];
```

Then build the 4 circles:

```
circCurrent = "";
circRadius = 30;
circB1_x = 70;
circB1_y = 50;
circB1 is Circle(circB1_x,circB1_y,circRadius,"blue",circCurrent ==
"B1" ? "red" : "black");
```

```
circB2_x = 220;
circB2_y = 50;
circB2 is Circle(circB2_x,circB2_y,circRadius,"blue",circCurrent ==
"B2" ? "red" : "black");
circY1_x = 370;
circY1_y = 50;
circY1 is Circle(circY1_x,circY1_y,circRadius,"yellow",circCurrent
== "Y1" ? "red" : "black");
circY2_x = 520;
circY2_y = 50;
circY2 is Circle(circY2_x,circY2_y,circRadius,"yellow",circCurrent
== "Y2" ? "red" : "black");
canvas is [rect, circB1, circB2, circY1, circY2];
```

You should now see the rectangle and 4 circles.

Selecting and moving a circle

The first part of the agency is to select the circle:

```
proc mouseClicked : mouseDown {
      if (mouseDown == 0) return;
      if (mouseX > circB1.x - circB1.radius && mouseX < circB1.x +
circB1.radius && mouseY > circB1.y - circB1.radius && mouseY <
circB1.y + circB1.radius)
            circCurrent = "B1";
      else if (mouseX > circB2.x - circB2.radius && mouseX <
circB2.x + circB2.radius && mouseY > circB2.y - circB2.radius &&
mouseY < circB2.y + circB2.radius)
            circCurrent = "B2";
      else if (mouseX > circY1.x - circY1.radius && mouseX <
circY1.x + circY1.radius && mouseY > circY1.y - circY1.radius &&
mouseY < circY1.y + circY1.radius)
            circCurrent = "Y1";
      else if (mouseX > circY2.x - circY2.radius && mouseX <
circY2.x + circY2.radius && mouseY > circY2.y - circY2.radius &&
mouseY < circY2.y + circY2.radius)
            circCurrent = "Y2";
}
```

This will allow you to select a particular circle. Next we need to modify this method to move the circle:

```
proc mouseClicked : mouseDown {

      ... from above ...

      else if (circCurrent != "") {
            if (circCurrent == "B1") {
                circB1_x = mouseX;
                circB1_y = mouseY;
            }
            else if (circCurrent == "B2") {
                circB2_x = mouseX;
                circB2_y = mouseY;
            }
            else if (circCurrent == "Y1") {
                circY1_x = mouseX;
                circY1_y = mouseY;
            }
            else if (circCurrent == "Y2") {
                circY2_x = mouseX;
                circY2_y = mouseY;
            }
      }
}
```

Try selecting and moving a circle now.

Detecting the end game

The next part of the game is to detect when 2 same colour circles are inside the rectangle. First we need to detect when each circle is inside (using dependency):

```
circB1_inside is circB1_x > rectX && circB1_x < rectX+rectWidth &&
circB1_y > rectY && circB1_y < rectY+rectHeight;

circB2_inside is circB2_x > rectX && circB2_x < rectX+rectWidth &&
circB2_y > rectY && circB2_y < rectY+rectHeight;

circY1_inside is circY1_x > rectX && circY1_x < rectX+rectWidth &&
circY1_y > rectY && circY1_y < rectY+rectHeight;

circY2_inside is circY2_x > rectX && circY2_x < rectX+rectWidth &&
circY2_y > rectY && circY2_y < rectY+rectHeight;
```

Now we can write the logic (also using dependency):

```
isComplete is (circB1_inside && circB2_inside && !circY1_inside
&& !circY2_inside) || (!circB1_inside && !circB2_inside &&
circY1_inside && circY2_inside);
```

Finally, modify the rectangle definition to change the background colour when complete:

```
rect is Rectangle(rectX,rectY,rectWidth,rectHeight, isComplete ?
"grey" : "white");
```

Extensions

Examine the full set of definitions for this model. Can you say which are plain observables, dependencies or agency.

You could add a button to the model that resets the model by moving the circles to random positions on the canva.

You could add a new game button that changes the colours of the circles.