Interactive Graphics Programming with JavaScript

East House Enrichment Program @ RIT, June 2023

Overview

This workshop will provide an introduction to core software programming concepts. Each student will work individually and in small groups to write programs that generate pictures of their own design. On day two, we'll work together to add interactivity to the images by changing what the program does depending on the user's mouse and/or keyboard actions.

We'll be writing *JavaScript* programs that utilize the *p5.js library* to support drawing shapes onto a digital canvas.

Since our time is limited, the primary goal is to give everyone a chance to:

- See what is possible with p5.js
- Run some pre-made programs
- Learn a few programming concepts so that you can customize the programs
- Edit the demo code a bit to play with how things work
- Try your hand at designing your own program

Workshop activities

We'll get as far as we can on Day 1 while leaving plenty of time for you to play around and brainstorm things you'd like to try. On Day 2, we'll wrap up some demos and then focus on helping you make your own programs!

Demos

1. Getting Started
2. Hello East House!
3. Shapes & Colors
4. Letting the computer do the math
5. Adding interactions & movement!

Things to Try: A collection of suggested adjustments to the demo code + an outline on designing your own project!

Quick Links

- Workshop Website: https://esmesh.github.io/east-house-intro-programming/
- Online p5.js Editor: https://editor.p5js.org/
- Quick Reference
- Other References

p5.js Quick Reference

Shapes

p5.js provides many pre-written functions for drawing common shapes. We can control how and where the shape is drawn by providing different data values to the function parameters.

Points are drawn as single dots given as two parameters to the `point` function: X, then Y

```
// x = 4, y = 5
point(4, 5);
```

Lines are drawn between two coordinates given as four parameters to the `line` function: Start X, Start Y, End X, End Y

```
// Line between 0, 0 & 4, 5
line(0, 0, 4, 5);
```

Rectangles are drawn with their top left corner at a given coordinate using a given width and height.

```
// Rectangle starting at 1, 2
// with a width of 4 and height of 3
rect(1, 2, 4, 3);
```

Ellipses (ovals) are drawn with their *center* at a given coordinate using a given width and height. *Circles are ellipses with only a width (diameter).*

```
// Ellipse centered at 3, 3
// with a width of 4 and height of 6
ellipse(3, 3, 4, 6);

// Circle centered at 3, 3
// with a width (diameter) of 4
ellipse(3, 3, 4);
```

Triangles and **Quads** are drawn as connections between the given coordinates. (3 coordinates for triangles, 4 for quads.)

```
// Triangle with corners at:
// - 4, 2
// - 7, 8
// - 2, 7
triangle(4, 2, 7, 8, 2, 7);
```

```
// Quad with corners at:
// - 2, 1
// - 8, 2
// - 7, 7
// - 1, 4
quad(2, 1, 8, 2, 7, 7, 1, 4);
```

Colors

The simplest way to refer to colors when using p5.js is to use pre-defined colors names. For example: “Red”, “Yellow”, “Purple”, … To see a list of available color names, see https://www.w3schools.com/colors/colors_names.asp

To change the background color of the canvas, we use the command `background` with a color name:

```
background("Aqua");
```

In order to fill the various shapes in the scene with color, we use the command `fill` with a color name. **However**, once we set a fill color **ALL** shapes will become that color until we change the color again.

```
  // Sun
  fill("Yellow");
  ellipse(350, 20, 40, 40); // this is now Yellow inside.
```

Black outlines on each shape are distracting. To make them go away, we can add `noStroke();`. We can also change the outline color using `stroke`:

```
noStroke(); // Hide ALL outlines.
stroke("Purple"); // Make the outlines purple.
```

Variables & Expressions

A **variable** is a storage location in computer memory that contains a value.

- Variables can contain any type of value.
- They are called “variables” because the value that they store can change - vary - and be updated.

An **expression** lets us use common math **operators** (+, -, *, /) to tell the computer to perform calculations for us.

```
let canvasSize = 300;

// Make the sun size based on the canvas size AND start it down a bit from
// the top left corner.
let sunSize = canvasSize/3;
let sunX = 0;
let sunY = sunSize/4;
```

We can use both variables & expressions as parameters to functions!

```
// Make the top 3/4 of the canvas blue for the sky
// (so start at 0,0 and full canvas width but not the full height down)
fill("LightSkyBlue");
rect(0, 0, canvasSize, 3*canvasSize/4);
```

If we change a variable each time we draw, our picture changes too!

```
  ellipse(sunX, sunY, sunSize);
  sunX = sunX + 1; // ADD this line of code to slowly move the sun!
```

Global Variables Defined by p5.js

p5.js defines some variables for our use that store properties of the enviroment. A fun one is the current location of the mouse pointer!

- `mouseX` (number): x-coordinate of the mouse
- `mouseY` (number): y-coordinate of the mouse

```
  // Add a "bee" where the mouse is
  fill("Yellow");
  ellipse(mouseX, mouseY, 5);
```

Things to try!

In the Hello World:

- Change the text to have your name
- Use a different background color *(See https://www.w3schools.com/colors/colors_names.asp for the possible color names.)*
- Add new text somewhere else on the canvas

In the Smiley demo:

- Change some colors
- Make the face, eyes, and/or nose bigger
- Add some freckles
- Change the ORDER of some statements *(What happens if the call to draw the face is last?!)*
- Change the sizes/placement of the sample shapes at the bottom

In the Flower demo:

- Change ***only*** the `canvasSize` variables.
- Change the flower height or width
- Try adding a bee 1 `flowerWidth` to the right of the top of the flower stem
    - *This is abstract art, the "bee" can be a simple circle or oval. ;)*
    - The top of the flower stem has variables for its X & Y coordinates: `stemTopX` and `stemTopY`

Implement the heart example

- Start with the diagram from the Getting Started page
- Draw out the layout in your notebook
- What shapes do you need?
- What are their coordinates/sizes?
- *See a solution here*

Design your own project!

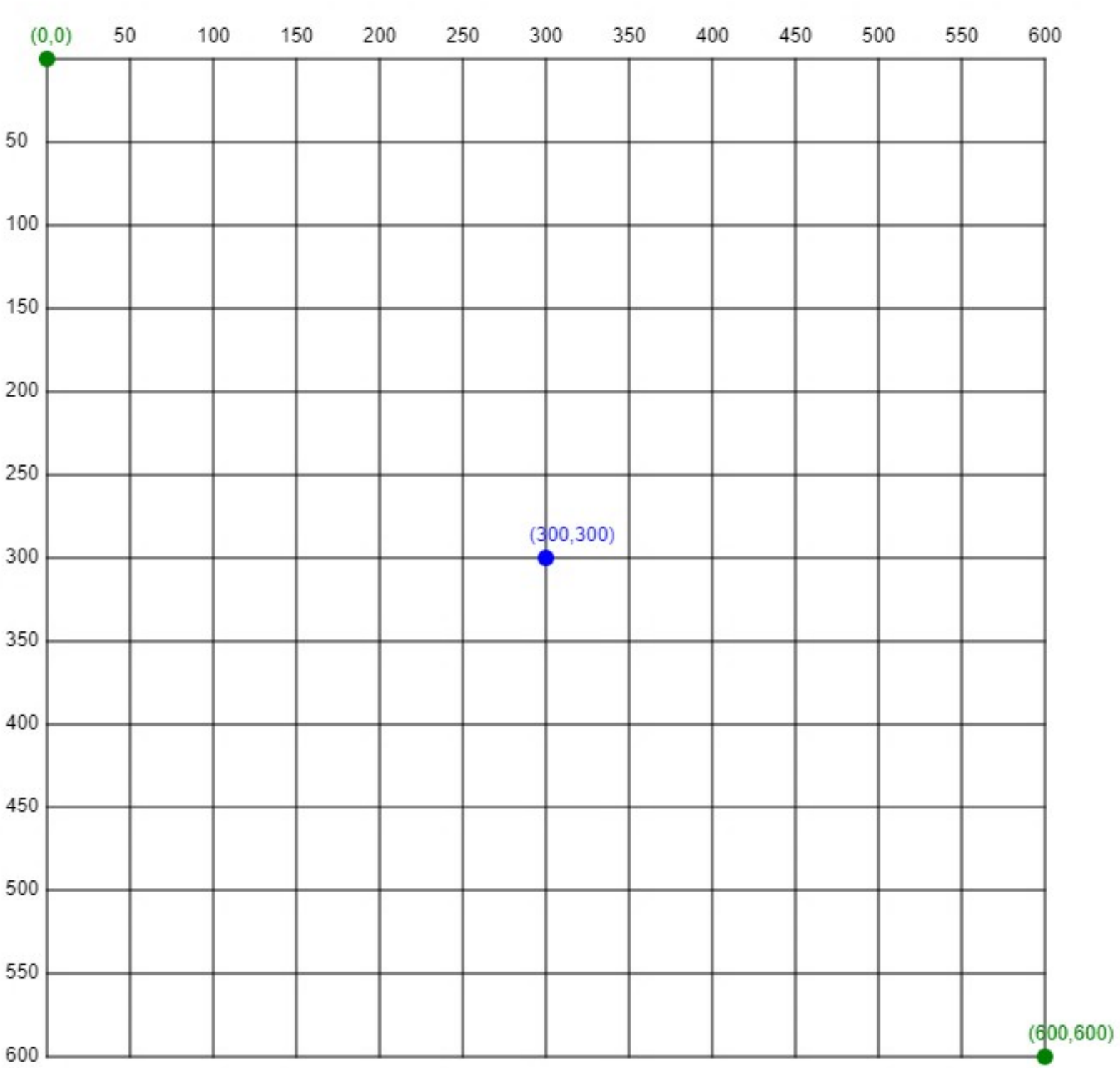
Feel free to ask us for help at any point along the way!

Planning

1. Sketch out your vision on graph paper
2. Identify the major sub-shapes & colors
3. Label with coordinates (or with the math you want the computer to do to figure out the coordinates)

Implementation

1. Create a new p5.js sketch project
2. Set the canvas size in `setup()`
3. Set the background color in `draw()`
4. Start writing the code to draw the shapes you need.


(0,0)
50
100
150
200
250
300
350
400
450
500
550
600
(300,300)
(600,600)