Drawing Things

Instructor Guide

Overview

Learning to draw rectangles and circles but in a fun way! (in tkinter)

Learning Goals

- Coordinate system (x, y, up is down)
- Functions for drawing shapes
- Using loops to draw shapes

Personal Growth Goals

- Computational Thinking: how to think about drawings in a step-by-step, methodical fashion

Skills Required

- Basic Geometry
- How to input code into a file and run code
- Conditionals
- Basic Loops

Resources Required

- Personal Computers for each student
- Paper and pencil so mentors and students can draw out what the predicted drawings will be
- Small groups working on the labs together with 2-3 students and a mentor

Instructor Preparation

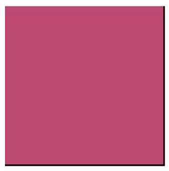
Pre-load the computers with the programming files

In Depth Description of Lab Activities

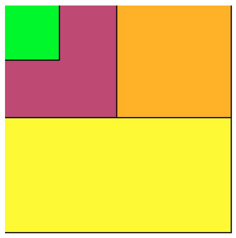
This lab is made up of six activities based around the six programming files.

Part 1 - Drawing Rectangles (rectangles1.py) (20 minutes)

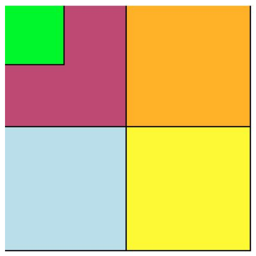
1. Run rectangles1.py
    a. Explain graphics:
        i. We draw things on a canvas
        ii. Our canvas is 200 pixels wide and 200 pixels tall
        iii. X is across and y is vertical
        iv. In computer graphics, “up is down”! (explain with call below)
        v. Thus (0,0) is in the top-left.
        vi. ignore the “runDrawing” code at the bottom, it just gets the “canvas” ready for us
            1. the important part is the width=200, height=200
    b. Explain each part of the rectangle drawing call: canvas.create_rectangle(0,0,100,100, fill="maroon")
    c. **Write out on their paper** (for their reference), that generally the call looks like: canvas.create_rectangle(x1, y1, x2, y2, fill=”colorName”)



2. For each of the next lines, ask them to predict what it will draw next (**have them draw each one out on paper before** you run the new version of code for each step) [Teaching Tip] **teach them to trace x1 and x2 on the top axis first, then y1 and y2 on the left axis**, *then* draw what the rectangle will be)
    a. canvas.create_rectangle(0,0,50,50, fill="green")
        i. size will grow smaller
    b. canvas.create_rectangle(100,0,200,200, fill="orange")
        i. Shape is right half of screen (x1 is offset)
    c. canvas.create_rectangle(0,100,200,200, fill="yellow")
        i. Shape is bottom of screen (y1 is offset)

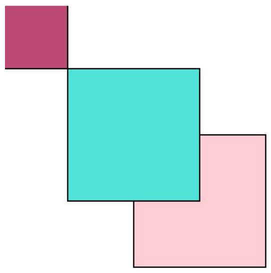


3. Then uncomment the last line and have them plan **on paper** how to draw a "lightblue" rectangle in the bottom left quadrant

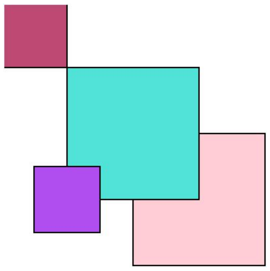


Part 2 - Guessing Rectangles (rectangles2.py) (10 minutes)

4. Before running rectangles2.py, have them read the code and draw out what it will be
    a. Then run rectangles2.py

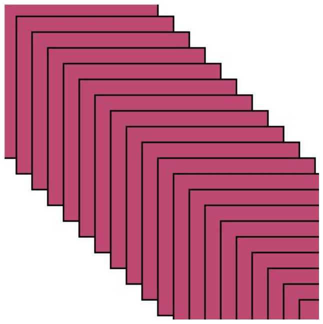


5. Then uncomment the last rectangle and trace out where it will go (tracing x1-to-x2 and y1-to-y2 on each axis first will help here!!)
    a. Then run the new rectangles2.py



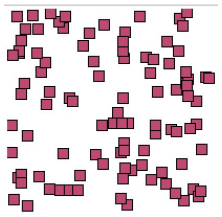
Part 3 - Looping Rectangles (rectangles3.py) (10 minutes)

6. Before running rectangles3.py
    a. Explain what a loop is (if they don't already know)
    b. Draw the first rectangle on the paper (mark down x = ? for each step)
    c. Draw the second rectangle
    d. Draw the third rectangle, and then see if they can extrapolate the pattern
7. Then run rectangles3.py (see pic below)
    a. Explain how the coordinates can go off the page, but only the 200x200 area is visible



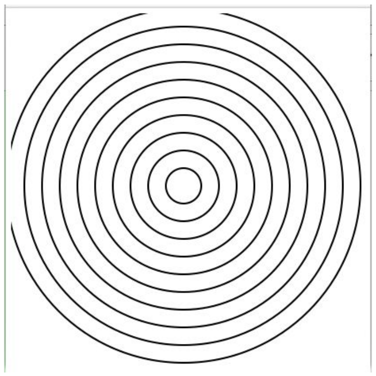
Part 4 - Random Rectangles (rectangles4.py) (15 minutes)

8. "So, that was just a few rectangles - what if we wanna draw them all over the place!"
    a. Run the code, get their curiosity going
    b. Explain for-loop (how many times does it run? 100. Where does i start? 0. Where does i end in the last time the loop runs? Not 100, but 99!)
    c. Explain randint, picks a number from 1 to 190
    d. Follow EACH of the comments, change up the code, and have fun with it!!



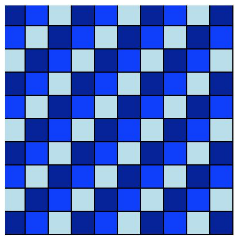
Part 5 - Bullseye (bullseye.py) (15 minutes)

9. Show them the picture below (also in Code Files folder) - “Next, let’s try drawing this bullseye shape.”
    a. All black circles are 10 pixels apart.
    b. Ask them for their ideas as how to tackle it - let them struggle for a bit.
    c. What's an easy way?
    d. Bonus if they use a loop instead of “hard-coding” it!
    e. Hints to give as they need:
        i. “Why do you think the starter code draws the outside circle first?”
        ii. “How do you draw the second circle? The third?”
        iii. “Do you see a pattern?”
        iv. “How do you get the coordinates to keep the center of each circle the same? Can the changing diameter help you here?”
    f. Show “bullseye_solution.py” if you run out of time



Part 6 - Challenge Question: Color Board (colorboard.py) (20 minutes)

10. Show them the picture below (in color, also in Code Files folder) - “Now for a tough challenge! Let’s try drawing this checkerboard with alternating colors, called a Color Board.”



11. Start with their ideas
    a. “We have to use loops here, right? It would take a long time to ‘hard-code’ it like this code is starting to do. Let’s not do that.”
    b. “Any initial ideas?”
12. Draw just one row of the same rectangle first
    a. “Let’s try just drawing the first row first.”

    b. “How many squares are in the first row?”
    c. “How do we make a loop that loops 10 times?”
    d. “Let’s try just drawing 10 of the same rectangle. What coordinates need to change over the row? X or Y?”
    e. “How much does the X change by for each square?”
    f. “Can we use multiplication to change the X value? Like the i times the width of each square?”
13. Alternate the colors in one row
    a. “How can we get the colors to alternate? There are three of them.”
    b. “We have to ‘remember’ the last color, right? How do we do that?”
    c. “You ‘remember’ using variables, what is a good variable name for that?”
    d. “How could we switch the variable based on the last color? If-statements maybe?” (explain if-statements if they do not know)
14. Loop over multiple rows
    a. “Okay, so for one row we changed the X. Now we want a loop to go over each ROW! How could we do that?”
    b. “We can actually put a loop inside a loop - loop inception - WHOA!” (you may need to show them / code this if it is too difficult) “Now we want to loop over the Ys, how many times?” (write for “j” or “row” in range(10))
    c. “Wow, just adding that filled in the rest! Isn’t it cool how by doing one row, doing many rows was pretty easy. Wouldn’t be like that if we were hard-coding it, right??”
    d. “You often want to do this when coding - break down big problems into smaller problems. Good job!”

Lesson Plan

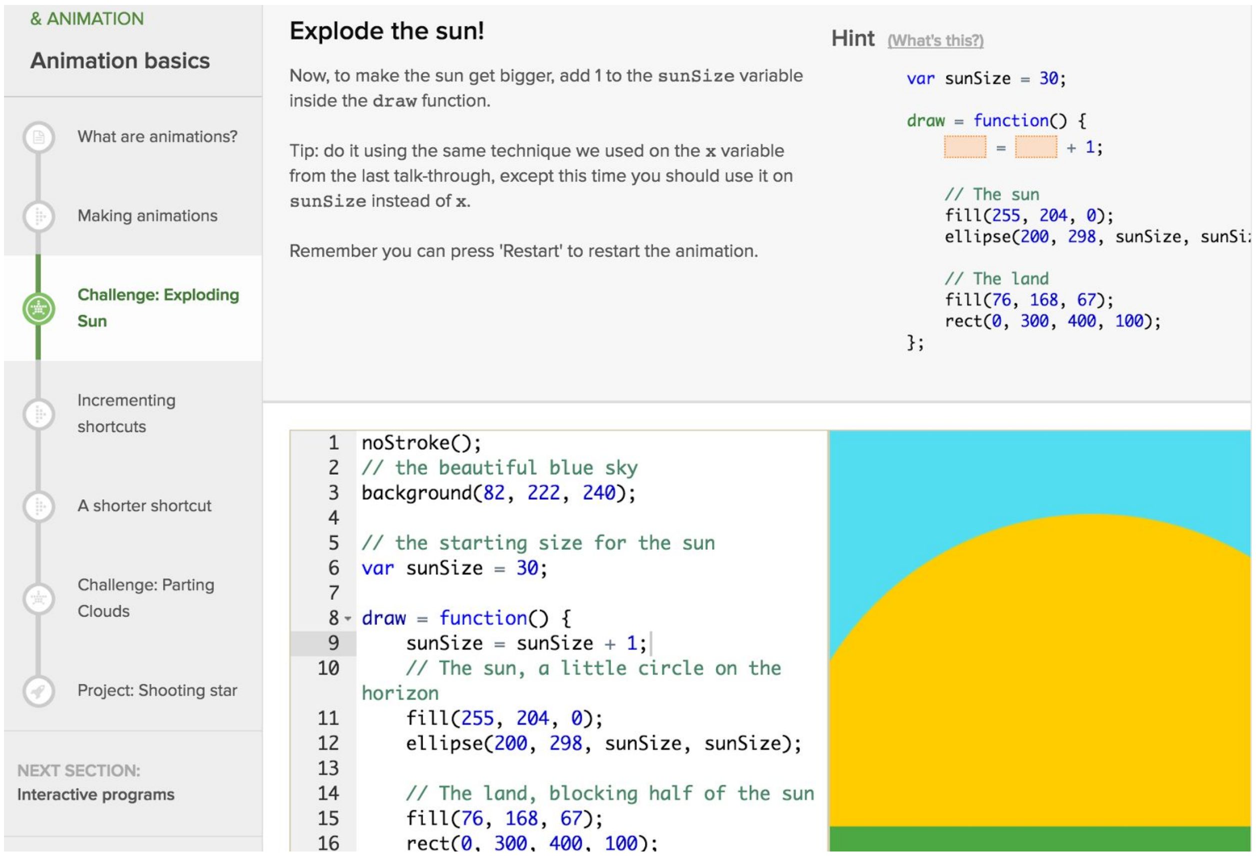
(:10) means that this part should be done by the tenth minute of the lesson

1. Setup (:15)
    a. Divide into groups
    b. Get ready to code (open IDLE all the “adventure” Python files.)
2. Part 1 - Drawing Rectangles (:35)
3. Part 2 - Guessing Rectangles (:45)
4. Part 3 - Looping Rectangles (:55)
5. Part 4 - Random Rectangles (:70)
6. Part 5 - Bullseye (:85)
7. Part 6 - Challenge Question: Color Board (:105 / end)

Additional Resources / Further Drawing Fun!

If the students like the lab, recommend they go through Khan Academy's Drawing with Code course online - it's well-structured and a lot of fun with drawing and animation too!

https://www.khanacademy.org/computing/computer-programming/programming



Explode the sun!

Now, to make the sun get bigger, add 1 to the `sunSize` variable inside the `draw` function.

Tip: do it using the same technique we used on the `x` variable from the last talk-through, except this time you should use it on `sunSize` instead of `x`.

Remember you can press 'Restart' to restart the animation.

```
var sunSize = 30;

draw = function() {
    [ ] = [ ] + 1;

    // The sun
    fill(255, 204, 0);
    ellipse(200, 298, sunSize, sunSi

    // The land
    fill(76, 168, 67);
    rect(0, 300, 400, 100);
};
```

```
noStroke();
// the beautiful blue sky
background(82, 222, 240);

// the starting size for the sun
var sunSize = 30;

draw = function() {
    sunSize = sunSize + 1;
    // The sun, a little circle on the horizon
    fill(255, 204, 0);
    ellipse(200, 298, sunSize, sunSize);

    // The land, blocking half of the sun
    fill(76, 168, 67);
    rect(0, 300, 400, 100);
```

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.