Unlocking the Use of Coding to Understand Geometric Concepts

Ann Wheeler, Texas Woman's University
Hannah Barrera, Texas Woman's University
Sarah Cooley, Texas Woman's University

Abstract

In this paper, the authors detail a fifth and a sixth grade geometry-based lesson involving Python coding. Both problem-based tasks involve the use of graphing various two-dimensional figures in the first quadrant with justification of shapes drawn. Helpful coding tips, sample prompts with example work, as well as teacher modifications, are included.

Keywords: Geometry, problem-solving, middle school, coding, computer science

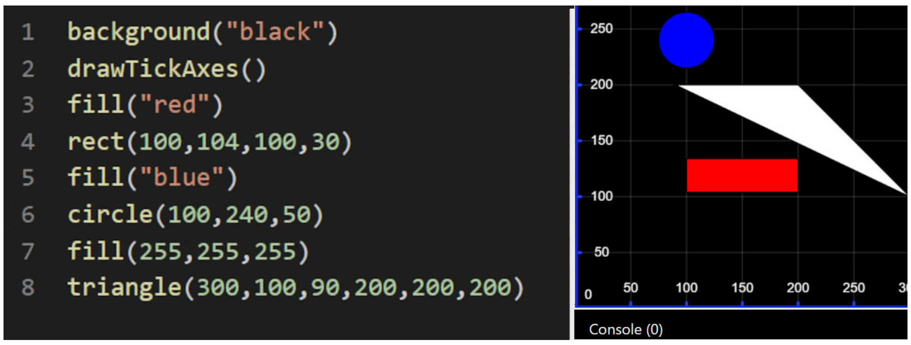
1 Introduction

Even though coding might sound like an atypical mathematical activity, it can be an innovative way to incorporate a STEM-based technology into a classroom to create engaging mathematically infused tasks. We, as mathematics educators, need to learn ways in which to provide deep learning opportunities that include suitable technology for students (National Council of Teachers of Mathematics [NCTM], 2000). The use of coding can be done at various grade levels and with multiple mathematical standards. In the following paper, we will investigate the use of Python coding to enrich student's understanding of geometry aligned to the Common Core Mathematics Standards at the fifth and sixth grade levels (CCSSM, 2010). Before we look at the projects, though, let's explore Python.

Let's Understand the Code: Python Logistics

For this work, we had students use a free Python editor, Strive Math (https://code.strivemath.com/). When writing Python code, students will need to learn some basic syntax, which the website tutorials will be helpful for student comprehension. One can see from the given tutorial materials that the code is not too difficult for upper elementary and middle school students to grasp. The teacher could even run through a simple example, such as creating a blue circle, red rectangle, and white triangle to help aid students' understanding of the coding (see Figure 1).

Figure 1: Sample discussion photo with Python code.



```
background("black")
drawTickAxes()
fill("red")
rect(100,104,100,30)
fill("blue")
circle(100,240,50)
fill(255,255,255)
triangle(300,100,90,200,200,200)
```

The teacher can demonstrate constructing the three shapes with the class and discuss the language needed to input into their computers. For example, to change the color of a shape, the student would need to type `fill()` and insert the color with quotes or the three grayscale values associated with the color and commas, which can be accessed through the above website link. Since just typing the name of the color is typically easier, we suggest using the generic color (red, blue, yellow, green, etc.) unless a student desires to challenge themselves with color variations.

With specific shapes, students need to write the name of the two-dimensional figure, such as `rect()` for rectangle, and then certain other values, depending on the shape. For example, the rectangle code would require the coordinates for the bottom left vertex, as well as what the program calls the "width" and "height" of the rectangle, inputted in the parentheses. [Note: In creating a generic quadrilateral, one can type `quad()` and then the x- and y-coordinates of each of the four vertices inside the parentheses.] The triangle code consists of the word `triangle()` and then the x- and y-coordinates of the three vertices inside the parentheses. With the "quad" and "triangle" codes, make sure to state vertices in clockwise or counterclockwise order, or there will be gaps in the coloring inside the shape, which will distort their pictures. If constructing circles, you would input `circle()`, the x- and y-coordinates of the center, and then the diameter in the parentheses. For additional figures, refer to the website mentioned above.

Once students are comfortable with simple coding of shapes, they can move onto project-based learning with their work. In the following paragraphs, we discuss two main tasks students could complete using coding. Sample pictures and teacher prompts are provided with helpful hints to aid classes in being successful with these lessons.

2 Fifth Grade Fun: Graph that Shape!

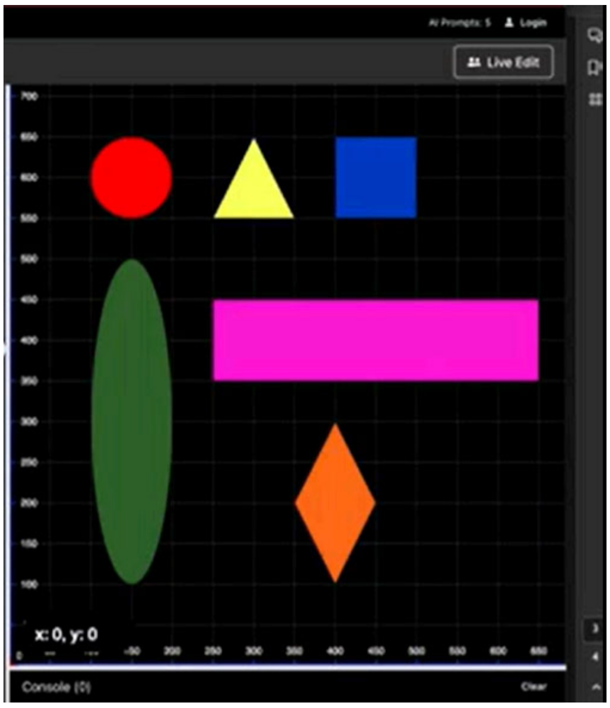
In fifth grade, students are learning to graph in the first quadrant (CCSS.MATH.CONTENT.5.G.A.2), as well as discussing two-dimensional shape relationships including their hierarchical connections (CCSS.MATH.CONTENT.5.G.B.4). Thus, having a lesson where students are tasked with generating code to create pictures including various shapes in the first quadrant can be an engaging experience.

At the beginning of class, the teacher should review the concepts of graphing in the first quadrant, as well as the basic attributes of common two-dimensional shapes like triangles, squares, rectangles, circles, and rhombuses. Once students feel confident about their understanding of these concepts, the teacher can task the class with the following prompt:

> *Using your coding program, create a picture of at least four different shapes of four different colors. You can either create separate shapes or combine them all to make an innovative masterpiece! Below your picture, identify each shape, as well as your Justification for each shape's name. For example, if you draw a blue square, how do you know it is a square?*

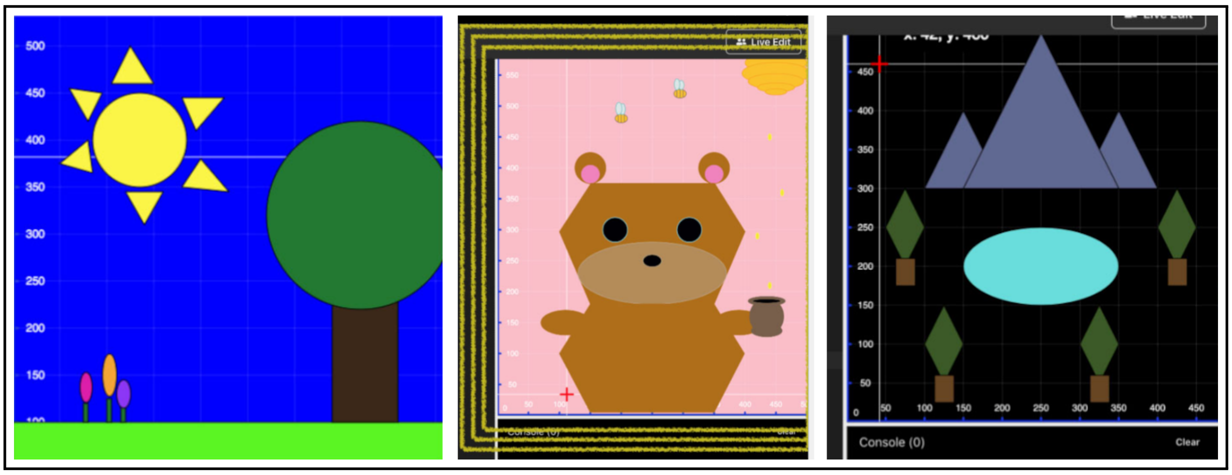
A sample picture is displayed in Figure 2.

Figure 2: Shapes created using Python.



Simple shape pictures can be accepted, but if students want to be more creative, they can use their imagination and create various illustrations, such as animals or landscape scenes (see Figure 3).

Figure 3: Sample student pictures.



Teachers can also require more restrictions on the drawing, such as requiring students to include a rhombus that is not a square or maybe have students create a specific triangle, such as a right triangle. Being specific with shapes can make for more challenging learning to take place.

3 Sixth Grade Sleuth: Name that Shape!

When connecting to sixth grade content, students can begin to plot polygons in the plane with specified vertices (CCSS.MATH.CONTENT.6.G.A.3). For example, students can plot given vertices and then identify the shapes they have constructed. This can be done in various ways, but creating a detective game on how to figure out the given two-dimensional figures could help to engage students.

To start the lesson, the teacher should review polygons with the class such as various quadrilaterals (e.g., squares, rhombuses, trapezoids, rectangles, etc.) and triangles (e.g., acute, right, obtuse, scalene, equilateral, isosceles). Once the teacher feels the class is ready to move on, have students pair up and provide the following prompt:

> *You are a detective trying to figure out the secret code to open this safe. Rumor has it that there is a treasure hidden inside, but I need your help! Using your program code and the first clue, create the given polygon. When your group has created the polygon and figured out its name, take your computer to me, verify its identity, and obtain your first digit of the passcode. Continue until you have figured out the three clues and unlock the treasure.*

Sample Clues:

- **Clue 1:** *Draw a yellow shape with vertices at* $(100, 100)$, $(100, 500)$, $(300, 300)$, *and* $(300, 100)$. *What am I? How do you know?*
- **Clue 2:** *Draw a white shape with vertices at* $(200, 200)$, $(200, 30)$, *and* $(30, 30)$. *What am I? How do you know?*
- **Clue 3:** *Draw a green rhombus that is not a square with side lengths of* 100 *units each. How do you know you have a rhombus that is not a square?*

With the above sample clues, you can see that you can make codes that are more or less challenging, depending on the class and/or student group. With the first and second clues, if students tell you they made a quadrilateral and triangle, respectively, you can ask them to be more specific, such as the type of quadrilateral and triangle created with justification. For the third clue, you can have the students think more deeply about what they would have to draw in Python to create a rhombus. What properties confirm that the shape is a rhombus? How do they know?

The teacher can have a box or lock that has three digits which correspond to the answer to the puzzle. Giving students small prizes (e.g., toys or homework passes) for winning can be a motivating factor for students to complete the task.

4 Conclusion

Using coding through mathematical tasks can make for appealing projects that teachers can use as review exercises in their classes. With both activities, instructors can differentiate the learning by having the problems easier or more challenging for students they feel might need certain accommodations. These variances could consist of modifying the types and number of shapes that students would need to create. Students might also want some time to plan their work outside of the coding program. For these students, use Appendix A as a planning page. Others struggling to create picture ideas or understand how to code could use Appendix B for some additional sample pictures with corresponding codes that instructors can use to help students understand how to draw more complicated pictures. However you choose to have students complete this activity, you will find this is a worthwhile use of a STEM-based task to open students' minds to coding and mathematical problem-solving.

References

Common Core State Standards Initiative (2010). *Standards of Mathematical Practice.* Available online at http://www.corestandards.org/Math

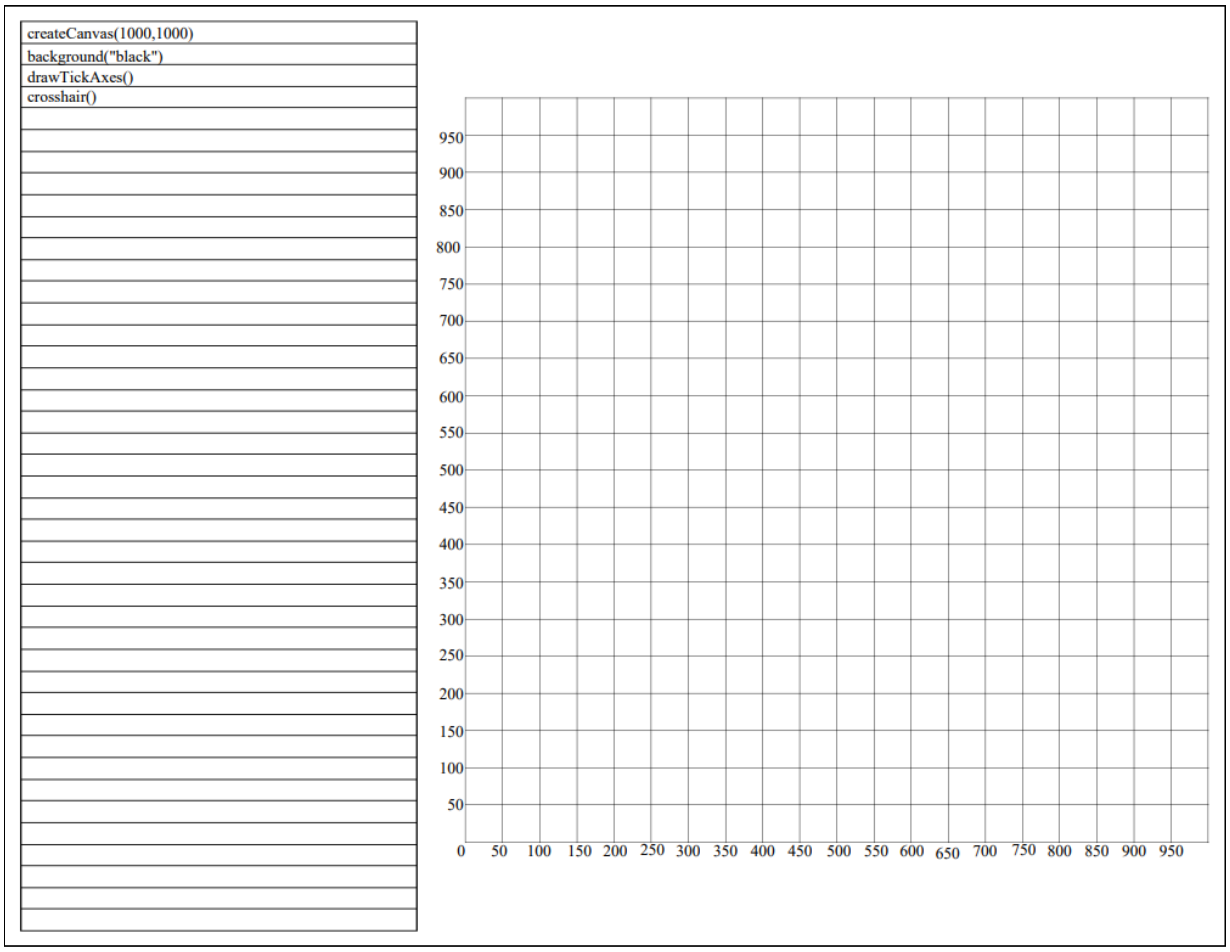
National Council of Teachers of Mathematics (2000). *Principles and Standards for School Mathematics*. Reston, VA.

Ann Wheeler is a Professor of Mathematics at Texas Woman's University in Denton, TX. Dr. Wheeler's teaching interests include mathematics education, where she emphasizes innovative and engaging teaching methods. Her research focuses on the use of technology in mathematics education and the development of mathematical problem-solving skills. Dr. Wheeler is actively involved in mentoring students and contributing to the academic community at TWU.

Hannah Barrera is a dedicated college student pursuing a degree in education, passionate about helping students achieve their full potential. With experience as a tutor, supplemental instruction leader, course assistant, and substitute teacher, she excels in creating engaging lesson plans, adapting to diverse needs, and fostering inclusive learning environments.

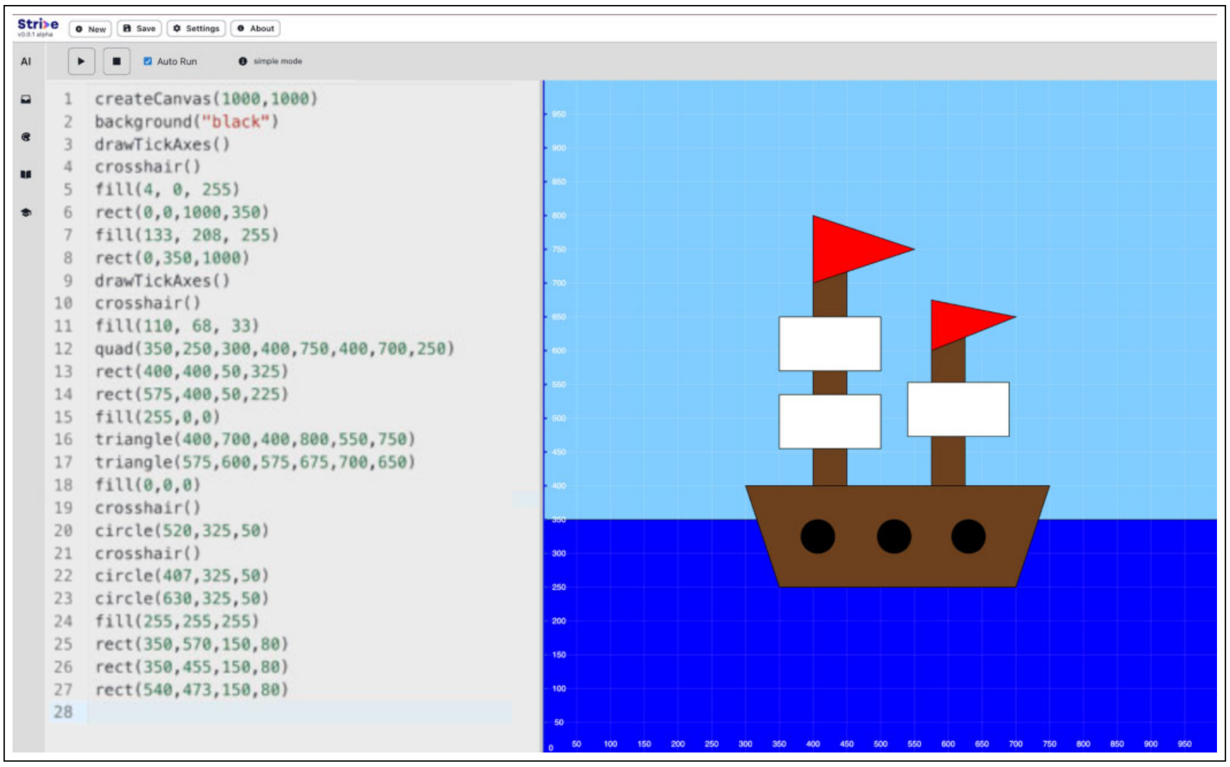
Sarah Cooley is an adjunct faculty member in the Department of Mathematics at Texas Woman's University in Denton, TX. She teaches courses such as College Algebra, Discrete Mathematics, and Statistics. Her professional interests include integrating coding to enhance the understanding of geometric concepts, as well as fostering innovative teaching methods that promote student engagement and comprehension in mathematics.

Appendix A: Blank Coding Page

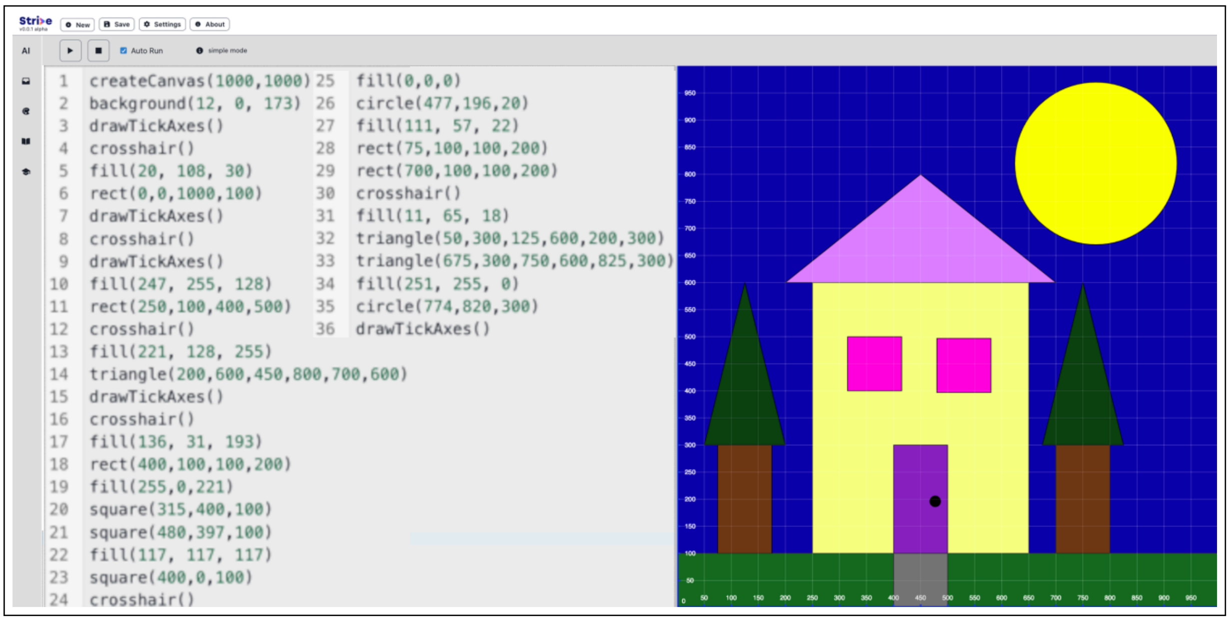


```
createCanvas(1000,1000)
background("black")
drawTickAxes()
crosshair()
```

Appendix B: Additional Sample Pictures with Generating Code

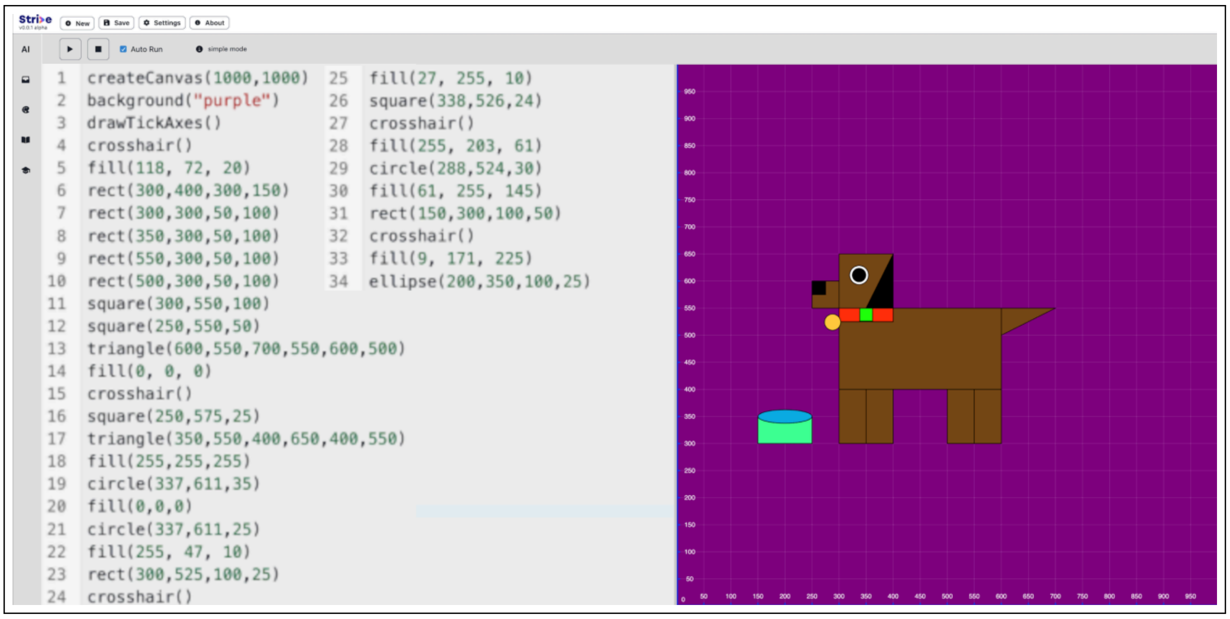


```
createCanvas(1000,1000)
background("black")
drawTickAxes()
crosshair()
fill(4, 0, 255)
rect(0,0,1000,350)
fill(133, 208, 255)
rect(0,350,1000)
drawTickAxes()
crosshair()
fill(110, 68, 33)
quad(350,250,300,400,750,400,700,250)
rect(400,400,50,325)
rect(575,400,50,225)
fill(255,0,0)
triangle(400,700,400,800,550,750)
triangle(575,600,575,675,700,650)
fill(0,0,0)
crosshair()
circle(520,325,50)
crosshair()
circle(407,325,50)
circle(630,325,50)
fill(255,255,255)
rect(350,570,150,80)
rect(350,455,150,80)
rect(540,473,150,80)
```

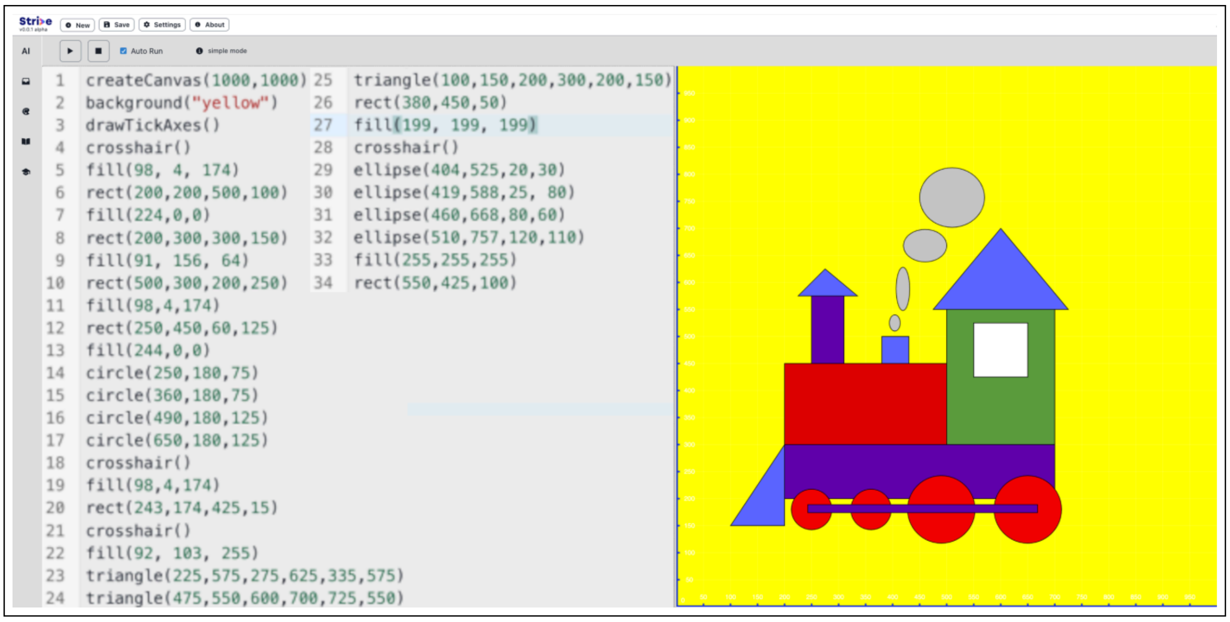


```
createCanvas(1000,1000)
background(12, 0, 173)
drawTickAxes()
crosshair()
fill(20, 108, 30)
rect(0,0,1000,100)
drawTickAxes()
crosshair()
drawTickAxes()
fill(247, 255, 128)
rect(250,100,400,500)
crosshair()
fill(221, 128, 255)
triangle(200,600,450,800,700,600)
drawTickAxes()
crosshair()
fill(136, 31, 193)
rect(400,100,100,200)
fill(255,0,221)
square(315,400,100)
square(480,397,100)
fill(117, 117, 117)
square(400,0,100)
crosshair()
fill(0,0,0)
circle(477,196,20)
fill(111, 57, 22)
rect(75,100,100,200)
rect(700,100,100,200)
crosshair()
fill(11, 65, 18)
triangle(50,300,125,600,200,300)
triangle(675,300,750,600,825,300)
fill(251, 255, 0)
circle(774,820,300)
drawTickAxes()
```

Appendix B (continued)



```
createCanvas(1000,1000)
background("purple")
drawTickAxes()
crosshair()
fill(118, 72, 20)
rect(300,400,300,150)
rect(300,300,50,100)
rect(350,300,50,100)
rect(550,300,50,100)
rect(500,300,50,100)
square(300,550,100)
square(250,550,50)
triangle(600,550,700,550,600,500)
fill(0, 0, 0)
crosshair()
square(250,575,25)
triangle(350,550,400,650,400,550)
fill(255,255,255)
circle(337,611,35)
fill(0,0,0)
circle(337,611,25)
fill(255, 47, 10)
rect(300,525,100,25)
crosshair()
fill(27, 255, 10)
square(338,526,24)
crosshair()
fill(255, 203, 61)
circle(288,524,30)
fill(61, 255, 145)
rect(150,300,100,50)
crosshair()
fill(9, 171, 225)
ellipse(200,350,100,25)
```



```
createCanvas(1000,1000)
background("yellow")
drawTickAxes()
crosshair()
fill(98, 4, 174)
rect(200,200,500,100)
fill(224,0,0)
rect(200,300,300,150)
fill(91, 156, 64)
rect(500,300,200,250)
fill(98,4,174)
rect(250,450,60,125)
fill(244,0,0)
circle(250,180,75)
circle(360,180,75)
circle(490,180,125)
circle(650,180,125)
crosshair()
fill(98,4,174)
rect(243,174,425,15)
crosshair()
fill(92, 103, 255)
triangle(225,575,275,625,335,575)
triangle(475,550,600,700,725,550)
triangle(100,150,200,300,200,150)
rect(380,450,50)
fill(199, 199, 199)
crosshair()
ellipse(404,525,20,30)
ellipse(419,588,25, 80)
ellipse(460,668,80,60)
ellipse(510,757,120,110)
fill(255,255,255)
rect(550,425,100)
```