WORCESTER POLYTECHNIC INSTITUTE

p5.Polar - Programming For Geometric Patterns

by

Liz Shihching Peng

A thesis submitted in partial fulfillment for the
degree of Master of Science
in
Interactive Media and Game Development

May 2020

APPROVED:

Professor Charles Davis Roberts, Major Thesis Advisor

Professor Brian J. Moriarty

Professor Lane T. Harrison

Abstract

Traditional teaching methods are often passive and do not interactively engage students, and this is even more challenging when teaching programming to beginners. In recent years, tech companies such as Google, and academic institutions like MIT, have introduced online learning environments to schools for teaching programming. Most of these learning environments are web-based, interactive, and provide visual feedback. Our project follows these trends and builds on p5.js, a JavaScript library that provides software sketching features and rapid visual feedback to reduce the barrier for learning programming languages. We designed and implemented a new library for drawing geometric patterns using polar coordinate systems, p5.Polar. We then developed a game that incrementally teaches our library to players, and evaluated it with an online user study.

Introduction

Conventional programming language courses focus on theoretical concepts before students see anything visual. This learning process usually frustrates the beginner and is uninteresting to students (Bhatti, Dewani, Maqbool & Memon, 2019). Their initial curiosity might be lost because they could not see the path from what they have to learn to what they want to create (McCarthy, Reas & Fry, 2016). Compared to teaching programming languages such as C++, the programming languages that provide visual feedback, reduced unnecessary syntax and were closer to human language (Lye & Koh, 2014). Processing is an example of a programming language created to teach the fundamentals of programming within a visual context. The Processing creators were inspired by Design By Numbers (DBN), a programming environment and language (McCarthy, Reas & Fry, 2016); Design By Numbers was also created to teach the ideas of computational thinking within a visual context (Maeda, 1999). Dots, lines, and fields can be combined with the idea of variables and conditional statements to generate procedural drawings . The goal of Processing is to make programming easier for art and design students as well as make programming interactive graphics easier for technical students (McCarthy, Reas & Fry, 2016). The core language and additional libraries of Processing were originally built with Java but it also has a twin sister, p5.js, a web version of Processing.

p5.js starts with the same goal of Processing and works on the web by using JavaScript, to make programming reachable for artists, designers, educators, and beginners (Ibid., 5.). The syntax between JavaScript and p5.js are identical, but p5.js adds abstractions for graphics and interactivity. p5.js also provides an online web editor environment that enables users to practice programming and create drawings.

p5.js offers a set of drawing functionalities which use the Cartesian coordinate system to position points, lines, and shapes on the screen's canvas. When drawing with p5.js, a user often starts by writing a single line of code using its built-in functions; the corresponding shape will then appear on the screen. By adding a few more lines of code, the user can create a complex pattern or animation.

As part of our research, we created a JavaScript library, *p5.Polar*, which draws using a polar coordinate system. The library enables users to draw shapes by defining an angle, a radius, and a distance (see Figure 1).

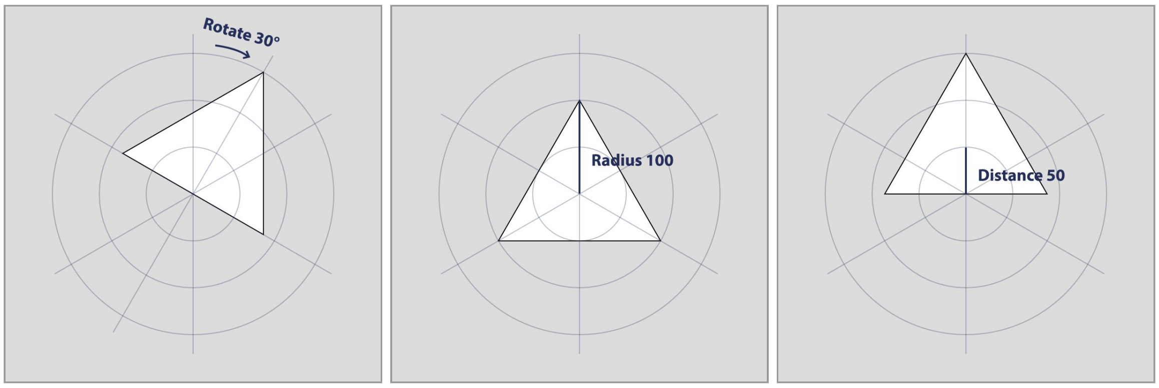


Figure 1. Drawing with p5.Polar library by defining an angle, a radius, and a distance

Instead of setting pairs of Cartesian coordinate values, the library converts polar coordinates to Cartesian coordinates, and reduces the barrier of trigonometry required when drawing with native p5.js functions for drawing patterns that radiate outward from a center point. Drawing such patterns with the p5.Polar library almost always requires fewer lines of code compared to the p5.js built-in functions. Besides this new library, our research also includes the design and development of a tutorial / pattern matching game to incrementally teach our library, inspired by several online interactive learning experiences, as discussed in the next section of this document. The study contains a web application implementation to provide visual feedback for library users to practice the drawing functionality and make creative patterns. The web application gamifies the coding editor and offers a series of programming challenges for users to recreate geometric patterns while understanding the concept of drawing with polar coordinates. Our goals for our research are to enable users to experiment with pattern creation, and to create patterns at a higher level of abstraction. The accompanying pattern redrawing games teaches players to use the library as well as encourages them to use the library for their future creative work.

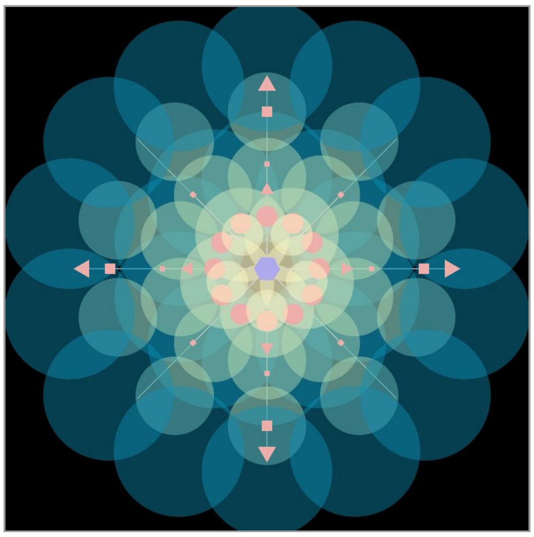
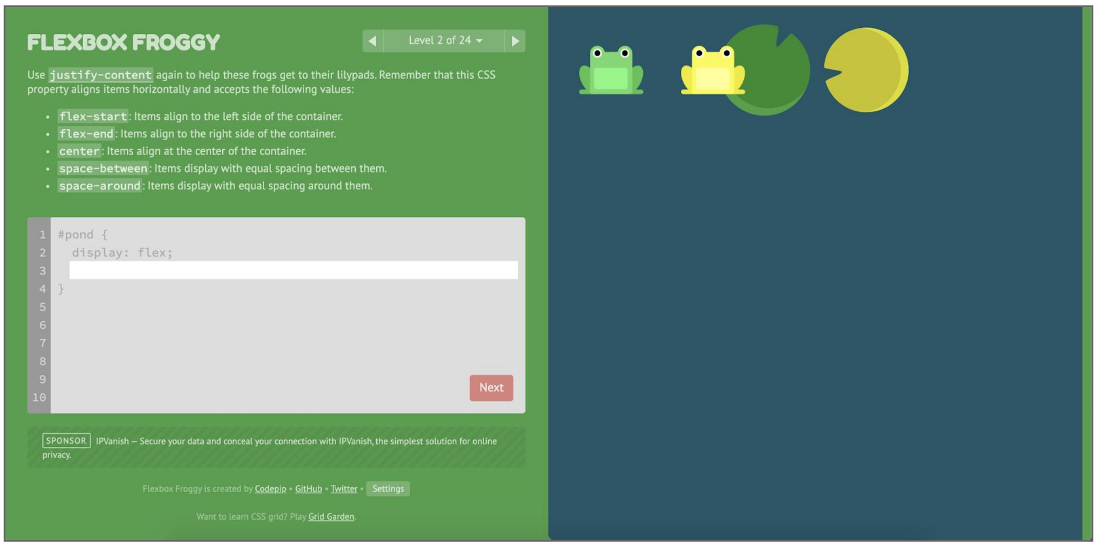


Figure 2. Draw abstract patterns with p5.Polar library

Background

For most high school students, traditional mathematical teaching methods are not intuitive and are mysterious (Feurzeig & Papert, 2011). Back in the 1960s, the programming language LOGO was first introduced as a potential framework for teaching mathematics (Ibid., 498.). LOGO is a text-based programming language and is well known for its novel use of *turtle graphics*. Students type commands to move a robotic turtle and create vector graphics such as lines or shapes. The creators of LOGO believe that teaching programming language can reduce formal academic barriers and enable students to understand key concepts in mathematics and improve their logical thinking (Ibid., 487.).

In recent years, several new programming languages have been introduced to traditional classroom to K-12 students such as Lye & Koh (2014) listed, Scratch (Burke, 2012; Lee, 2010), Toontalk (as cited in Kahn, Sendova, Sacristán & Noss, 2011), Stagecast Creator (Denner, Werner & Ortiz, 2012) and Alice (Graczyn ́ ska, 2010). Many of these are inspired by the concepts of the LOGO programming language and provide visual feedback that is easy to understand (Ibid., 53.). Scratch is one of the most popular programming languages that is used in K-12 classes (Ibid., 52.). It is a block-based visual programming language. Students can create interactive stories, games, and animations through dragging nodes that represent programming functions to complete the logical code blocks. p5.js is a text-based programming language but offers rapid visual feedback. It serves as a software sketchbook and is accessible for beginners and non-technical students. Students who use it learn programming through creating interactive graphics (McCarthy, Reas & Fry, 2016). Traditional programming languages such as Java, are closer to the computer's way of thinking (Smith, Cypher, & Tesler, 2000), while programming languages that provide visual feedback are likely to be suitable for teaching computational thinking (Lye & Koh, 2014). The creators of p5.js also found that students can find motivation and encouragement by providing immediate visual feedback (McCarthy, Reas & Fry, 2016). Our research draws inspiration from several interactive experiences that teach coding through games and provide immediate visual feedback, including the Flexbox Froggy (see Figure 3), CSS Grid Garden (see Figure 4), Khan Academy (see Figure 5), Turtle Academy (see Figure 6), and the p5.js Web Editor (see Figure 7).


FLEXBOX FROGGY

Level 2 of 24

Use `justify-content` again to help these frogs get to their lilypads. Remember that this CSS property aligns items horizontally and accepts the following values:

- `flex-start`: Items align to the left side of the container.
- `flex-end`: Items align to the right side of the container.
- `center`: Items align at the center of the container.
- `space-between`: Items display with equal spacing between them.
- `space-around`: Items display with equal spacing around them.

```
#pond {
  display: flex;

}
```

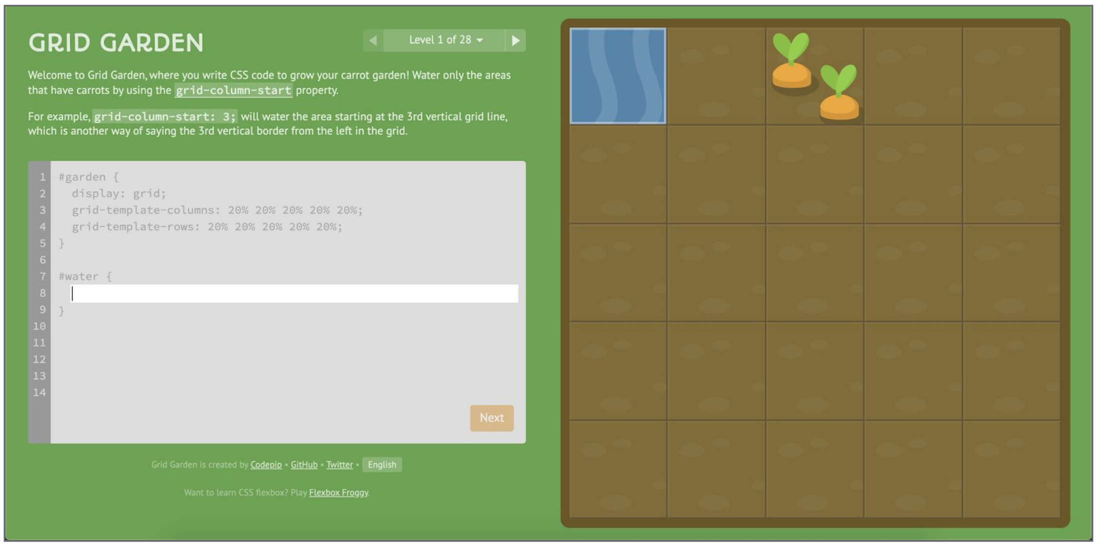
Next

SPONSOR IPVanish – Secure your data and conceal your connection with IPVanish, the simplest solution for online privacy.

Flexbox Froggy is created by Codepip • GitHub • Twitter • Settings

Want to learn CSS grid? Play Grid Garden.

Figure 3. Interactive experience - Flexbox Froggy


GRID GARDEN

Level 1 of 28

Welcome to Grid Garden, where you write CSS code to grow your carrot garden! Water only the areas that have carrots by using the `grid-column-start` property.

For example, `grid-column-start: 3;` will water the area starting at the 3rd vertical grid line, which is another way of saying the 3rd vertical border from the left in the grid.

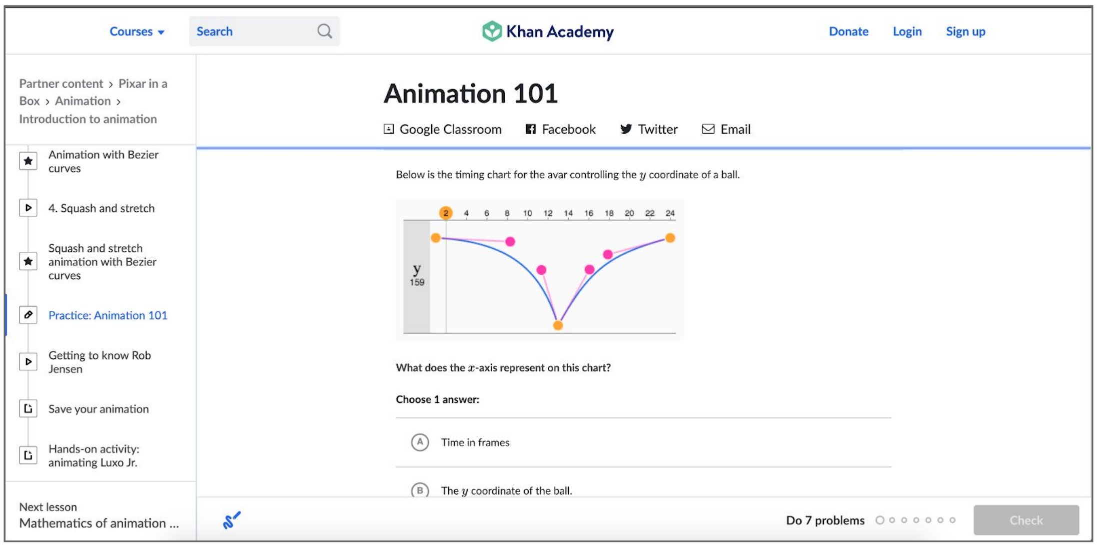
```
#garden {
  display: grid;
  grid-template-columns: 20% 20% 20% 20% 20%;
  grid-template-rows: 20% 20% 20% 20% 20%;
}

#water {

}
```

Next

Grid Garden is created by Codepip • GitHub • Twitter • English

Want to learn CSS flexbox? Play Flexbox Froggy.

Figure 4. Interactive experience - CSS Grid Garden


Courses
Search
Khan Academy
Donate
Login
Sign up
Partner content › Pixar in a Box › Animation › Introduction to animation
Animation with Bezier curves
4. Squash and stretch
Squash and stretch animation with Bezier curves
Practice: Animation 101
Getting to know Rob Jensen
Save your animation
Hands-on activity: animating Luxo Jr.
Next lesson
Mathematics of animation ...
Google Classroom
Facebook
Twitter
Email
y
159
Do 7 problems
Check

Animation 101

Below is the timing chart for the avar controlling the y coordinate of a ball.

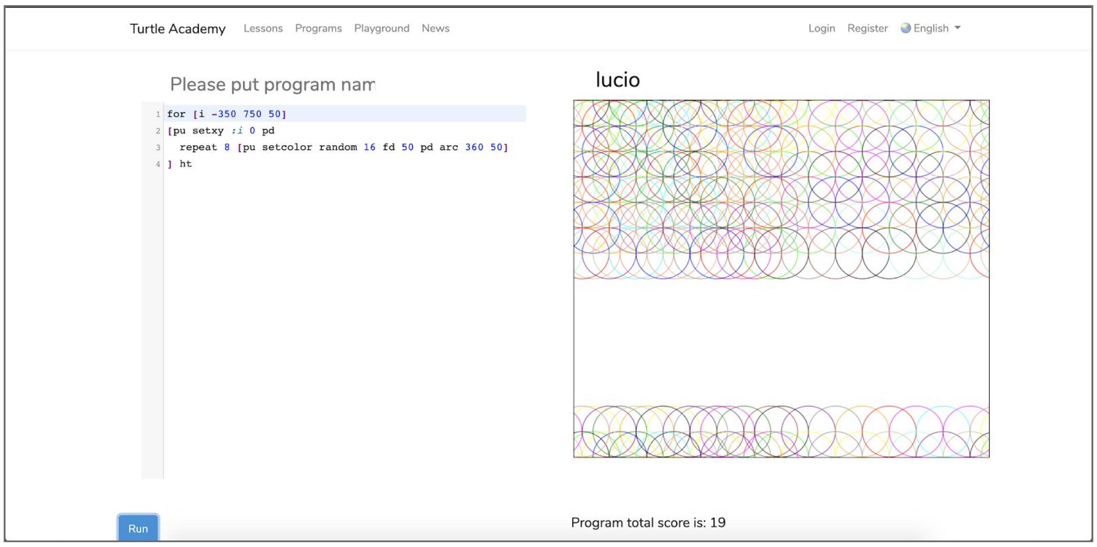
What does the x-axis represent on this chart?

Choose 1 answer:

(A) Time in frames

(B) The y coordinate of the ball.

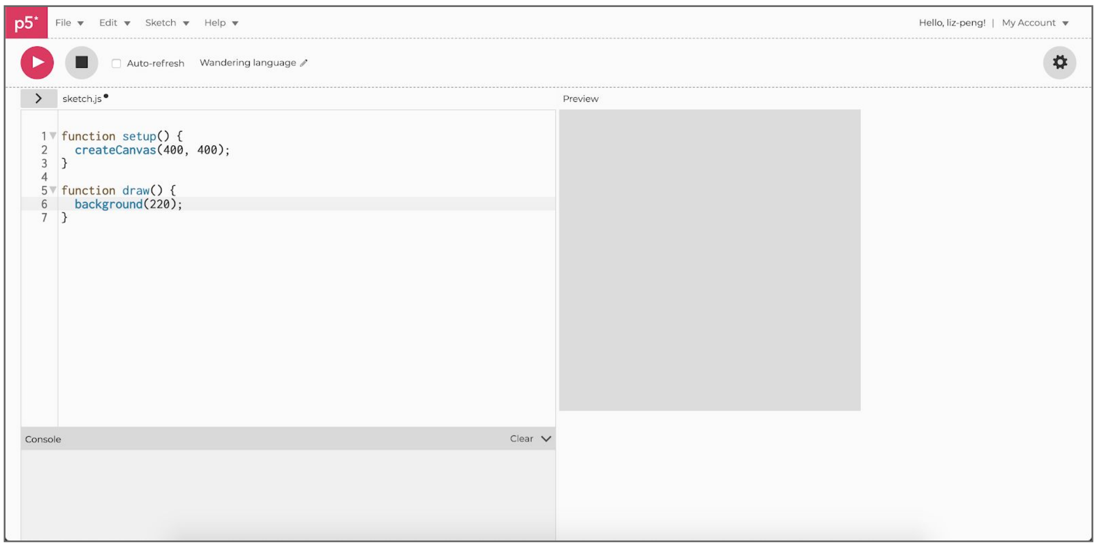
Figure 5. Interactive experience - Khan Academy



```
for [i -350 750 50]
[pu setxy :i 0 pd
  repeat 8 [pu setcolor random 16 fd 50 pd arc 360 50]
] ht
```

Program total score is: 19

Figure 6. Interactive experience - Turtle Academy



```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

Figure 7. Interactive experience - p5.js Web Editor

Methodology

In this section we begin by illustrating typical drawing functions found in p5.js, and the difficulties associated with creating procedural, radial patterns using cartesian coordinates. We then describe the abstractions our library contains for drawing with polar coordinates, and describe the iterative design / development process used to create our gamified tutorial for the library.

Drawing geometric patterns with p5.js (without using p5.Polar)

p5's built-in functions use the Cartesian coordinate system to position points, lines and shapes on the canvas. For example, giving x-coordinate and y-coordinate values, we can plot a point at 2D dimension space with p5.js built-in `point()` function:

```
function draw() {
  point(15, 20);
}
```

The feature of p5.js serves as a blank canvas that allows a user to sketch by layering lines of code to create drawings. Add one line of code to plot a point, add a few more lines to form a shape, and change the color of each shape by adding more lines of code. Take the `ellipse()` function as an example. When drawing ellipses starting from the middle of the canvas, we need at least two p5.js built-in functions, `translate()` and `ellipse()`. Specify the x and y parameters can translate the vector graphics to left or right, and up or down.

Setting a pair of x and y parameters along with width and height values to `ellipse()` function, draws an ellipse on the middle of the screen as shown in Figure 8:

```
function draw() {
  translate(width/2, height/2);
  ellipse(0, 0, 100, 100);
}
```

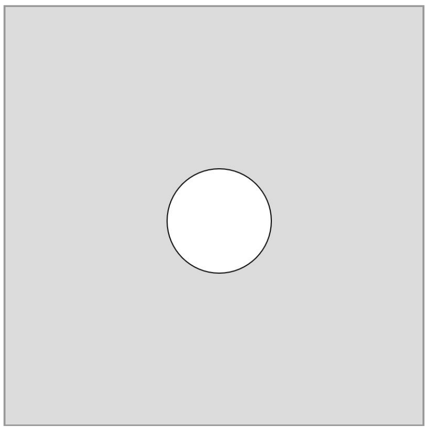


Figure 8. Draw an ellipse in the middle of canvas with p5.js built-in function

Add one more line of code to set the color of the shape. We can use the built-in `fill()` function and give RGB values from the range 0 to 255 as shown in Figure 9:

```
function draw() {
  translate(width/2, height/2);
  fill(238, 175, 170);
  ellipse(0, 0, 100, 100);
}
```

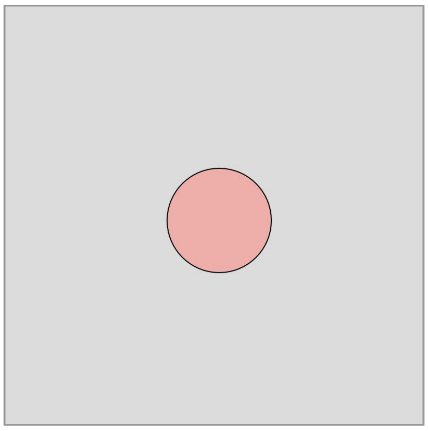


Figure 9. Draw a colored ellipse with p5.js built-in function

Layering multiple lines of code allows us to draw much complicated patterns just like drawing on a sketchbook. It is the greatest feature of p5.js. Draw a ring of ellipses by layering six `ellipse()` functions as shown in Figure 10:

```
function draw() {
  translate(width/2, height/2);
  fill(238, 175, 170);
  ellipse(0, -100, 100, 100);
  ellipse(0, 100, 100, 100);
  ellipse(86, -50, 100, 100);
  ellipse(86, 50, 100, 100);
  ellipse(-86, 50, 100, 100);
  ellipse(-86, -50, 100, 100);
}
```

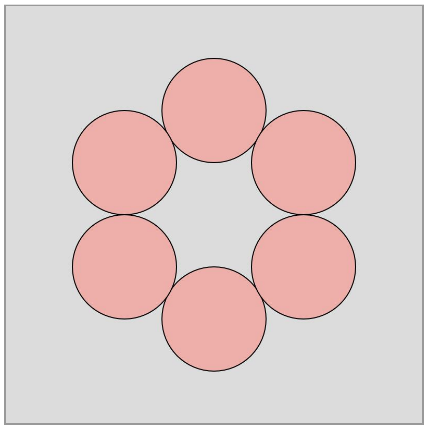


Figure 10. Use brute force method to draw a ring of ellipses with p5.js built-in function

p5.js enables what its authors describe as “sketching with code”, layering lines of code one after another to gradually develop sketches. (McCarthy, Reas & Fry, 2016). When drawing with the p5s built-in functions, programmers can either guess to set pairs of x-coordinate and y-coordinate parameters or set them with mathematical calculations. If a user feels manually positioning pairs of x and y values is too verbose, there is another way to draw patterns that requires better understanding of trigonometry, as shown in Figure 11:

```
function draw() {
  for(let i=0; i<6; i++) {
    resetMatrix();
    translate(width/2, height/2);
    let angle = (TWO_PI/6)*i;
    translate(sin(angle)*100, cos(angle)*100);
    ellipse(0, 0, 100, 100);
  }
}
```

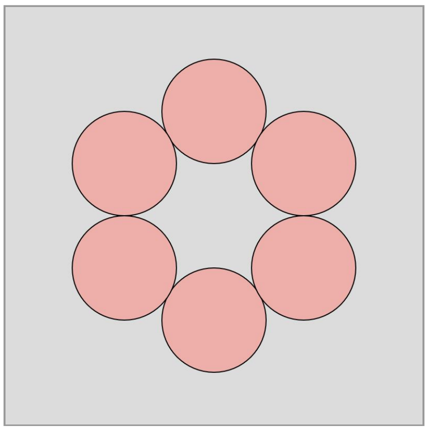


Figure 11. Use for loop method to draw a ring of ellipses with p5.js built-in function

With trigonometry and the `translate()` function users can create a for loop that draws the same pattern on the screen as calling `ellipse()` six times with different coordinates.

The ease of sketching with p5.js is also a downside. Imagine drawing a complex abstract pattern with the built-in functions, where users would have to type tens or hundreds of individual calls to the `ellipse()` function, and give each of them different pairs of x-coordinate and y-coordinate values. While using a loop is terser, it requires familiarity with underlying trigonometry calculations in order to draw complicated abstract patterns. Both methods might frustrate beginners doing creative work.

A polar coordinate system for p5.js

In order to minimize the barriers of mathematics requirement, and reduce the verbosity of numerical Cartesian coordinate settings for creating patterns with p5.js built-in functions, we built a new JavaScript library, *p5.Polar*, which implements drawing using a polar coordinate system. (https://github.com/liz-peng/p5.Polar). The library converts polar coordinates to Cartesian coordinates for use with the built-in functions of p5.js, and abstracts the mathematics required for drawing abstract patterns. The source code of p5.Polar library basically starts with resetting the current matrix with the identity matrix. Then programmers select a center point for the polar coordinate system. The library abstract mathematics required by converting degree measurements to their corresponding values in radians, and then draws shapes that are placed using trigonometry calculations. It often requires fewer lines of code to draw radial patterns with the p5.Polar library as compared to the built-in functions of p5.js.

The p5.Polar library offers two sets of drawing functions, one set for drawing single shapes and another set for drawing rings of shapes.. There are nine types of shapes in p5.Polar: line, ellipse, triangle, square, pentagon, hexagon, heptagon, octagon, and polygon.

- `polarLine(angle, radius, [distance])`
- `polarTriangle(angle, radius, [distance])`
- `polarSquare(angle, radius, [distance])`
- `polarPentagon(angle, radius, [distance])`

- `polarHexagon(angle, radius, [distance])`
- `polarHeptagon(angle, radius, [distance])`
- `polarOctagon(angle, radius, [distance])`
- `polarEllipse(angle, widthRadius, heightRadius, [distance])`
- `polarPolygon(number, angle, radius, [distance])`

The line, triangle, square, pentagon, hexagon, heptagon, and octagon drawing functions require at least two parameters: angle and radius.. Distance is an optional parameter for every function.

Drawing a triangle with `polarTriangle()` function as shown in Figure 12:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangle(0, 100, 0);
}
```

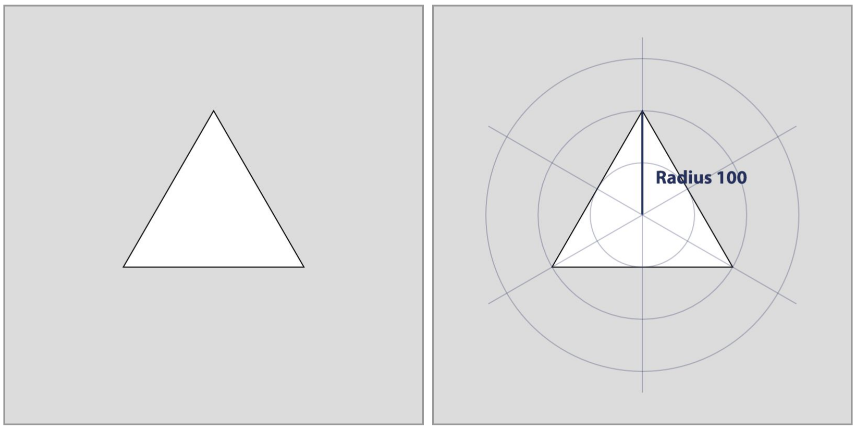


Figure 12. Draw a triangle with p5.Polar library

Moving the shape from center point by defining the distance as shown in Figure 13:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangle(0, 100, 50);
}
```

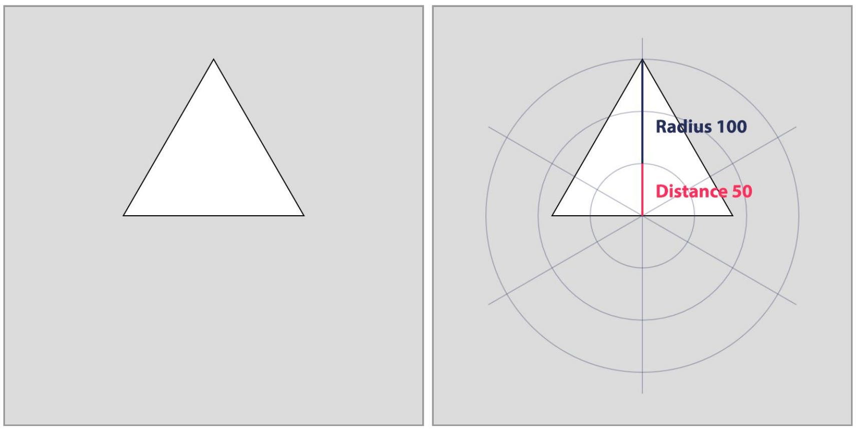


Figure 13. Move a triangle with p5.Polar library

Rotating the shape by defining the angle as shown in Figure 14:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangle(30, 100, 50);
}
```

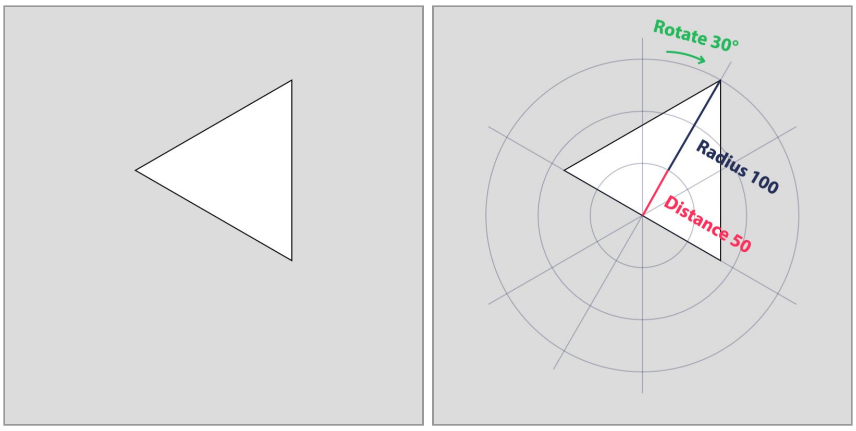


Figure 14. Rotate a triangle with p5.Polar library

The basic usage of p5.Polar library is defining the angle, radius and distance of a shape by following the order. Distance is an optional parameter, therefore, `polarTriangle(0, 100)` is functionally the same as `polarTriangle(0, 100, 0)`.

The `polarEllipse()` function requires four parameters: angle, width, height, and distance, in order to draw an oval shape as shown in Figure 15:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipse(0, 50, 100, 0);
}
```

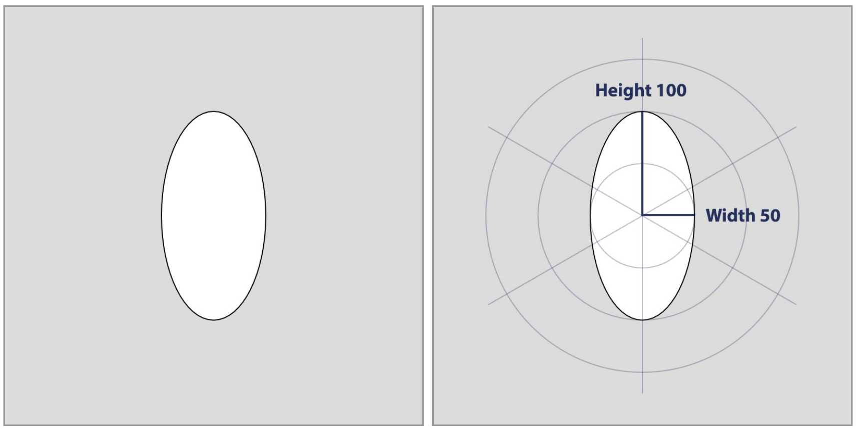


Figure 15. Draw an ellipse with p5.Polar library

The `polarPolygon()` enables drawing shapes with an arbitrary number of edges by specifying the number of edges, an angle, a radius, and an optional distance to the function.

Drawing a twelve-sided shape with the `polarPolygon()` function as shown in Figure 16:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarPolygon(12, 0, 100);
}
```

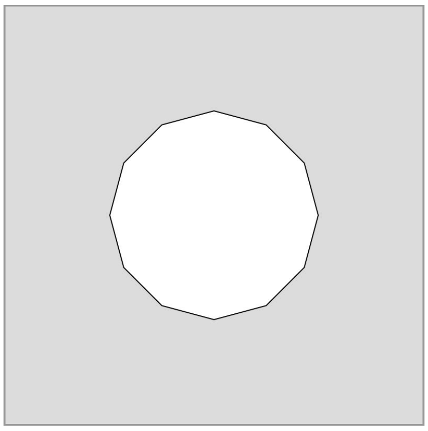


Figure 16. Draw a twelve edges shape with p5.Polar library

The p5.Polar library enables users to draw geometric patterns with polar coordinates, but the library still retains the features of p5.js, helping users sketch with code by layering lines of code to create drawings.

By manipulating the angles, radii, and distances used in our polar coordinate system, we can quickly create simple patterns, as shown in Figure 17:

```
function draw() {
  polar.setCenter(width/2, height/2);
  noFill();
  polarEllipse(0, 10, 5, 120);
  polarEllipse(-60, 5, 10, -120);
  polarEllipse(60, 5, 10, -120);
  polarEllipse(0, 10, 5, -120);
  polarEllipse(-120, 5, 10, -120);
  polarEllipse(120, 5, 10, -120);
  polarEllipse(0, 50, 50, 0);
  polarTriangle(0, 100, 0);
  polarTriangle(180, 100, 0);
}
```

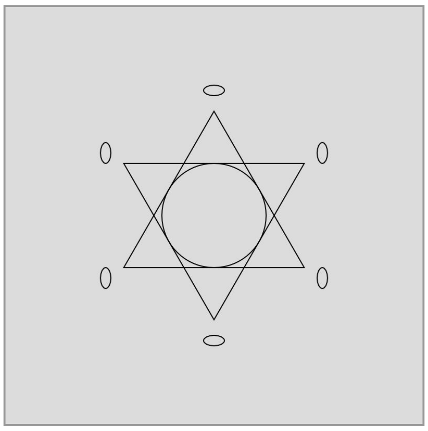


Figure 17. Draw a geometric pattern with p5.Polar library

Combining this with the built-in functions (such as `fill()`) of p5.js enables us to create I. colorful patterns, such as shown in Figure 18:

```
function draw() {
  polar.setCenter(width/2, height/2);
  noStroke();
  fill(175, 170, 238);
  polarEllipse(0, 10, 5, 120);
  polarEllipse(-60, 5, 10, -120);
  polarEllipse(60, 5, 10, -120);
  fill(238, 175, 170);
  polarEllipse(0, 10, 5, -120);
  polarEllipse(-120, 5, 10, -120);
  polarEllipse(120, 5, 10, -120);
  polarEllipse(0, 50, 50, 0);
  fill(175, 170, 238, 120);
  polarTriangle(0, 100, 0);
  fill(238, 175, 170, 120);
  polarTriangle(180, 100, 0);
}
```

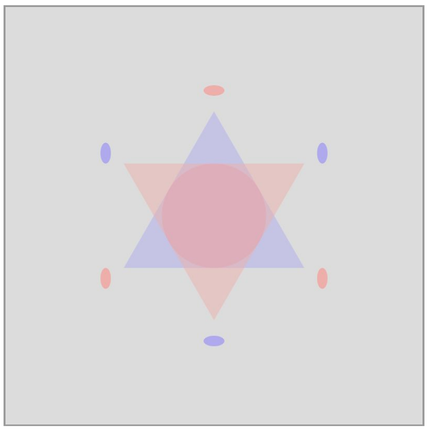


Figure 18. Draw a geometric pattern and fill the color with p5.js built-in function

Layering lines of code to see how shapes changing on the screen is a more visualize way of learning programming. The p5.Polar library also provides drawing functions for advanced users to create multiple copies of a shape, laid out in a ring, with only one line of code. There are nine such drawing functions: lines, ellipses, triangles, squares, pentagons, hexagons, heptagons, octagons, and polygons.

- `polarLines(number, radius, distance, [callback])`
- `polarTriangles(number, radius, distance, [callback])`
- `polarSquares(number, radius, distance, [callback])`
- `polarPentagons(number, radius, distance, [callback])`
- `polarHexagons(number, radius, distance, [callback])`
- `polarHeptagons(number, radius, distance, [callback])`
- `polarOctogons(number, radius, distance, [callback])`
- `polarEllipses(number, widthRadius, heightRadius, distance, [callback])`
- `polarPolygons(number, number of edges, radius, distance, [callback])`

The naming is exactly the same as the functions that draw a single shape but with a ‘s’ appended to the function name. These functions require at least three parameters: number of shapes, radius, and distance. For example, the code below would draw six triangles using the `polarTriangles()` function by specifying the number of shapes, radius, and distance from the center point as the parameters (see Figure 19):

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangles(6, 50, 100);
}
```

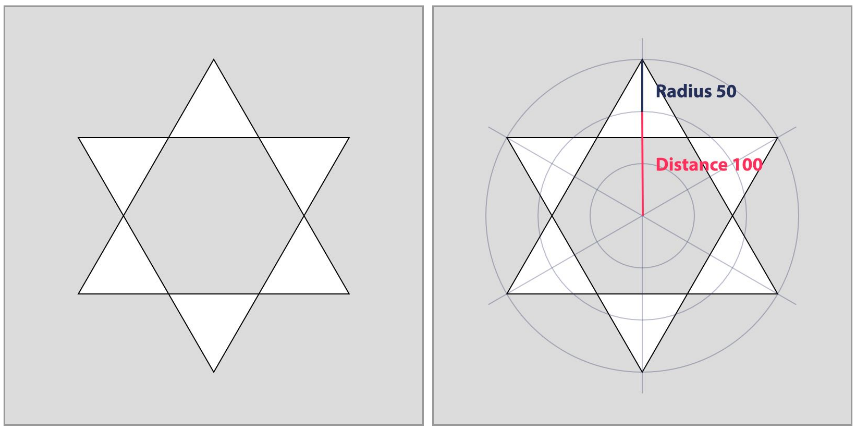


Figure 19. Draw six triangles with p5.Polar library

p5.Polar library automatically performs the necessary calculations for users to create symmetric geometric patterns. Giving an odd number to the multiple drawing function will also yield symmetric pattern (see Figure 20):

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarPentagons(7, 50, 100);
}
```

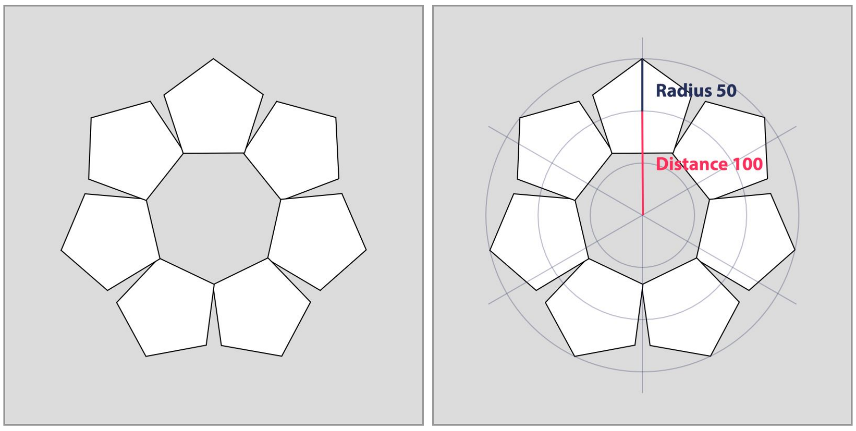


Figure 20. Draw seven pentagons with p5.Polar library

The flexibility of p5.Polar library allows users to layer calls to functions that draw single shapes, rings of shapes, and calls to the built-in functions provided by p5.js (see Figure 21):

```
function draw() {
  polar.setCenter(width/2, height/2);
  strokeWeight(0.3);
  polarLines(6, 150, 100);
  noStroke();
  fill(175, 170, 238);
  polarHexagon(30, 50, 0);
  fill(252, 248, 200);
  polarEllipses(8, 10, 10, 100);
  fill(238, 175, 170);
  polarEllipses(12, 40, 40, 200);
  fill(252, 248, 200, 120);
  polarEllipses(5, 80, 80, 160);
}
```

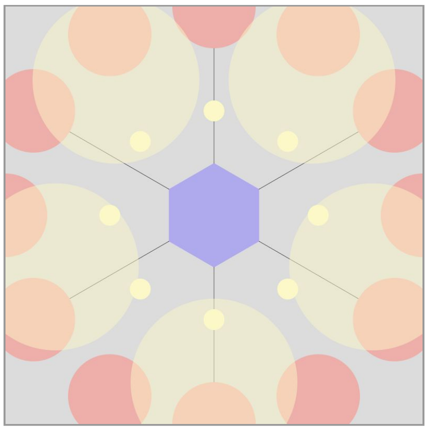


Figure 21. Draw a geometric pattern and fill the color with p5.js built-in function

As one additional more advanced feature, the p5.Polar library supports customization by passing an optional callback function as the last parameter to functions that draw multiple shapes.. The callback function allows users to manipulate the value of each parameter and create asymmetric patterns.

Use the callback function by simply giving the optional parameter to any one of the multiple drawing functions:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangles(6, 50, 100, function(...args) {
    return args;
  });
}
```

`...args` means the rest parameter syntax in JavaScript which indicates an indefinite number of arguments stored as an array. In our library, the value of each member of `args` will be:

- `args[0] // the current shape iteration, from 1 to n`
- `args[1] // angle`
- `args[2] // radius`
- `args[3] // distance`

For example, if we have a function `polarTriangles(6, 50, 100, function(...args))`, our callback function will be passed six sets of arguments:

```
[1, 60, 50, 100]
[2, 60, 50, 100]
[3, 60, 50, 100]
[4, 60, 50, 100]
[5, 60, 50, 100]
[6, 60, 50, 100]
```

The iteration advances from 1 to 6, the angle which is based on the polar coordinate system has been divided by 360 degrees which is 60 (360/6 = 60), the radius of every triangle is 50, and the distance from the center point which is always 100. Since `args[0]` is not a constant value, it allows users to make creative patterns with some mathematical calculations.

Take a ring of regular ellipses as an example as shown in Figure 22:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipses(6, 50, 50, 100, function(...args) {
    return args;
  });
}
```

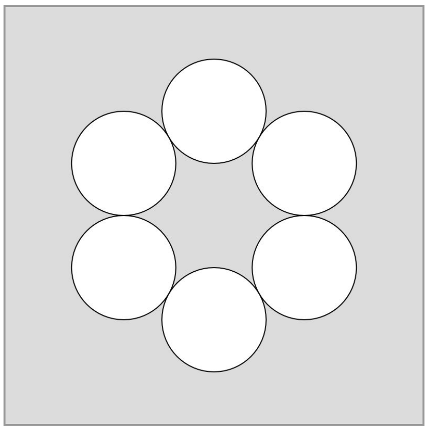


Figure 22. Draw a ring of ellipses with p5.Polar library

Resetting the width and height of radius by giving new values to the second and third arguments as shown in Figure 23:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipses(6, 50, 50, 100, function(...args) {
    args[2] = 80;
    args[3] = 80;
    return args;
  });
}
```

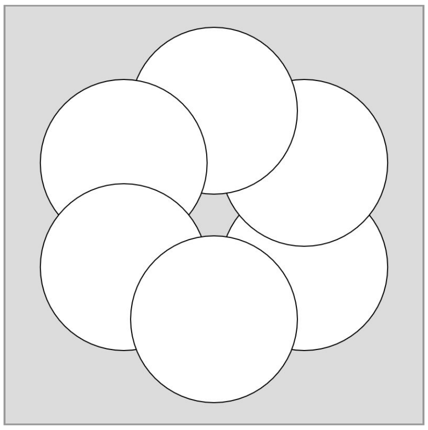


Figure 23. Draw a ring of ellipses with p5.Polar library

Setting the width and height of each individual shape by referencing the first argument as shown in Figure 24:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipses(6, 50, 50, 100, function(...args) {
    args[2] = args[0]*6;
    args[3] = args[0]*6;
    return args;
  });
}
```

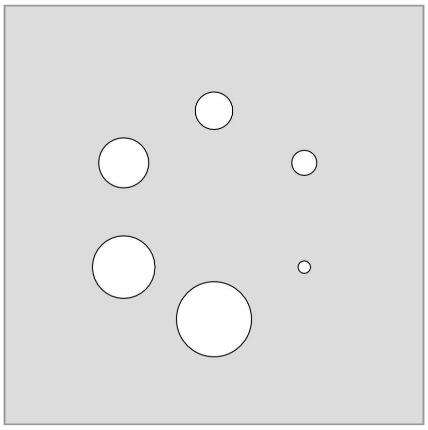


Figure 24. Draw a ring of ellipses with p5.Polar library

As a final example, below we create a gradient of color by using the first argument to change RGB values, as shown in Figure 25:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipses(6, 50, 50, 100, function(...args) {
    fill(args[0]*60, args[0]*60, args[0]*60);
    args[2] = args[0]*6;
    args[3] = args[0]*6;
    return args;
  });
}
```

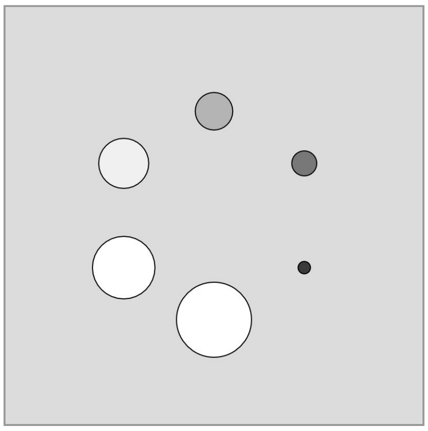


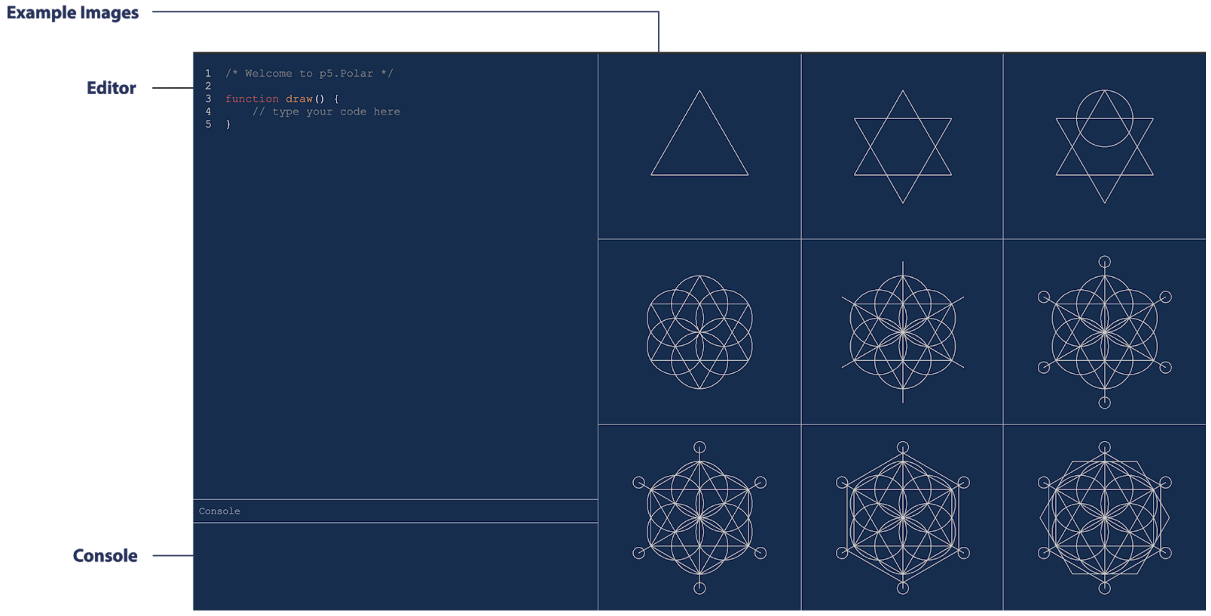
Figure 25. Draw a ring of ellipses with p5.Polar library

We use `args[0]` multiplied by different numbers to set each shapes' stroke and fill color, and also use it to re-calculate the radius and distance of each ellipse. Each value of the `args array` can be used with both p5.Polar functions and p5.js built-in functions.

A gaming application for learning p5.Polar library

Traditional programming courses typically focus on teaching algorithms, methods and theories first before students see anything visual (McCarthy, Reas & Fry, 2016). It is a difficult way for beginners to learn a programming language. Teaching programming by offering instant visual feedback often gives encouragement and motivation for students (ibid.). Inspired by several online interactive learning experiences include Flexbox Froggy, CSS Grid Garden, Khan Academy, Turtle Academy, and the p5.js Web Editor. The study comes along with a gaming application for creative coding, with a focus on re-creating pattern examples through each game level while learning to draw geometric patterns with polar coordinate concept (https://liz-peng.github.io/p5.Polar/). The application is designed with the goal of giving immediate visual feedback while programming and gamifying the learning process for p5.Polar library users.

The initial prototype of the game—as shown in Figure 26— gives example images that tell players how to draw patterns by gradually increasing their complexity.

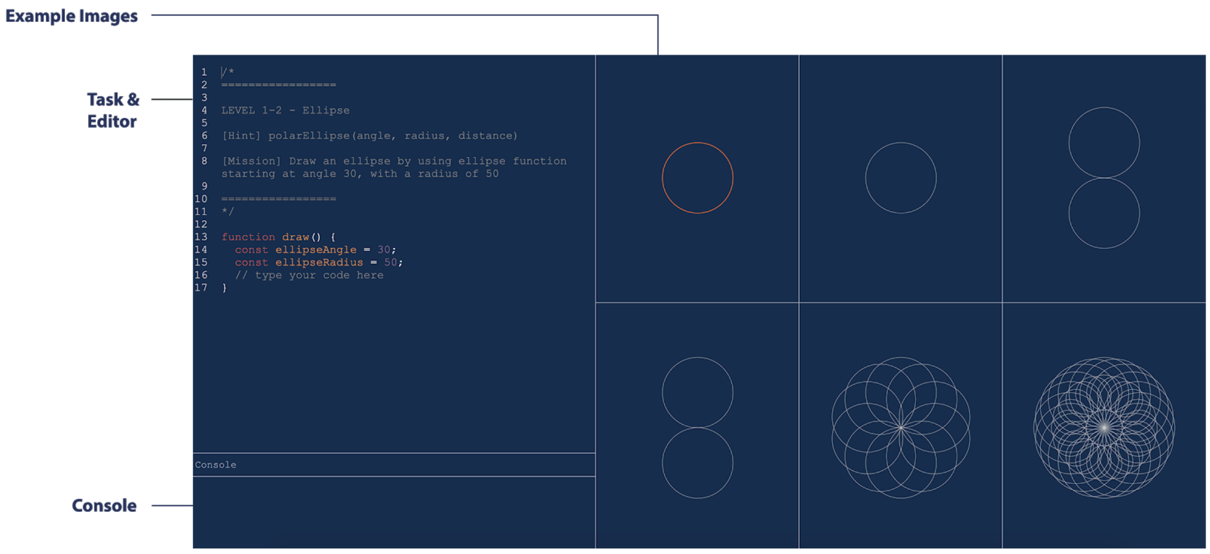


```
/* Welcome to p5.Polar */

function draw() {
    // type your code here
}
```

Figure 26. p5.Polar learning application prototype

The idea of the game shares the same goal with p5.js and Processing that makes coding accessible for non-technical students. Start drawing with a circle, and gradually draw the whole pattern while learning to code at the same time. The game separates into three parts, one part for coding, one for showing the error message after compiling, and one for displaying the example images.

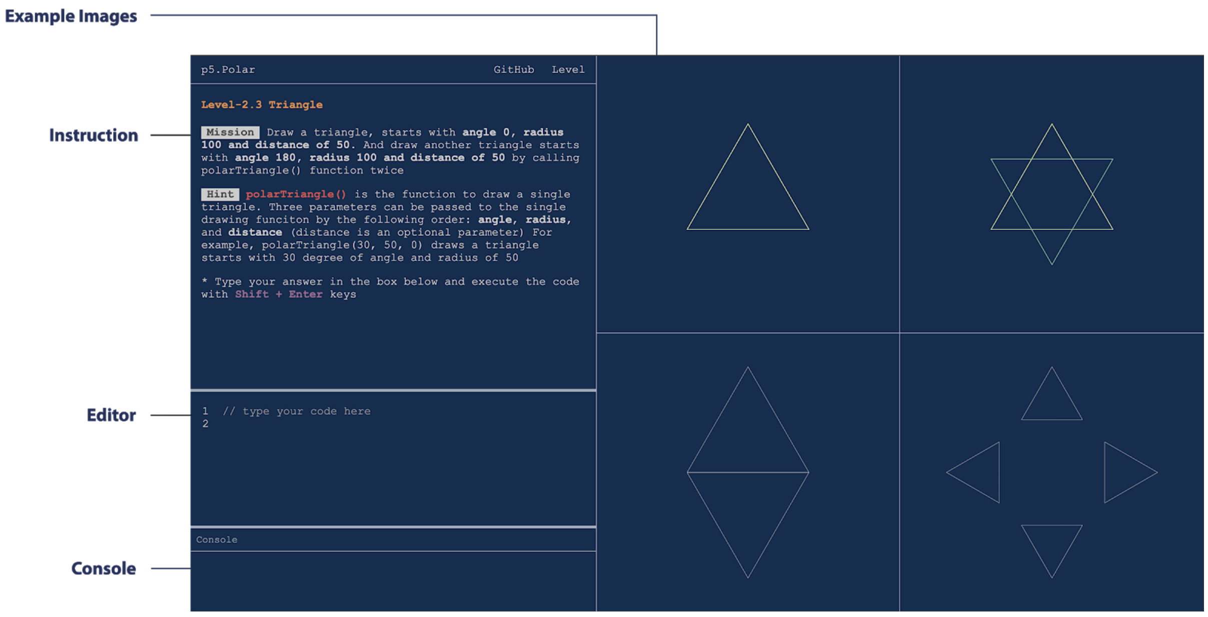
To introduce different types of shapes with their corresponding drawing functions, we changed the game focus on one shape per level, and added a feature where the pattern changes color when it is drawn correctly. The second prototype focus on teaching single drawing and multiple drawing functions with different types of shape in one level (see Figure 27):



```
/*
==================

LEVEL 1-2 - Ellipse

[Hint] polarEllipse(angle, radius, distance)

[Mission] Draw an ellipse by using ellipse function
starting at angle 30, with a radius of 50

==================
*/

function draw() {
  const ellipseAngle = 30;
  const ellipseRadius = 50;
  // type your code here
}
```

Figure 27. p5.Polar learning application prototype

In order to simplify the learning process, the game separates teaching functions that draw individual versus multiple shapes into separate levels. Our third prototype begins by teaching how to draw individual triangles, as shown in Figure 28.



Level-2.3 Triangle

Mission Draw a triangle, starts with **angle 0, radius 100 and distance of 50.** And draw another triangle starts with **angle 180, radius 100 and distance of 50** by calling polarTriangle() function twice

Hint **polarTriangle()** is the function to draw a single triangle. Three parameters can be passed to the single drawing funciton by the following order: **angle**, **radius**, and **distance** (distance is an optional parameter) For example, polarTriangle(30, 50, 0) draws a triangle starts with 30 degree of angle and radius of 50

* Type your answer in the box below and execute the code with **Shift + Enter** keys

```
// type your code here
```

Figure 28. p5.Polar learning application prototype

This third prototype was inspired by Flexbox Froggy, the CSS learning game. In this prototype, the code editor and instructions are separated. It also adds thorough method explanations for the library so users can reference each p5.Polar function. This interface and the overall user experience of the game is evaluated in the next section of this document.

We use string matching and regular expressions to check the correctness of user-coded patterns. Unfortunately, this limits the available coding styles that can be used to solve the pattern puzzles; in future work we will move to using pixel comparisons between the target pattern and the user-coded pattern, so that players will be free to use any coding methodologies.

Evaluation

Our evaluation takes two forms. First, we compare the abstractions of p5.Polar to alternative methods of achieving the same results in p5.js, with concerns for expressiveness and avoiding math that—while trivial for many—might be difficult to understand for some beginning programmers. Second, we evaluate our tutorial game via user-testing accompanied by an online survey.

p5.Polar library evaluation

The p5.Polar library keeps the sketching style of p5.js while abstracting the underlying trigonometry that p5.js users need to draw complicated geometric patterns. This section compares drawing with the p5.js framework's Cartesian coordinate system to drawing with the p5.Polar library's polar coordinate system.

Drawing an ellipse with p5.js built-in function requires a pair of x-coordinate and y-coordinate values, and a pair of width and height of the shape (see Figure 29):

```
function draw() {
  translate(width/2, height/2);
  ellipse(0, 0, 50, 50);
}
```

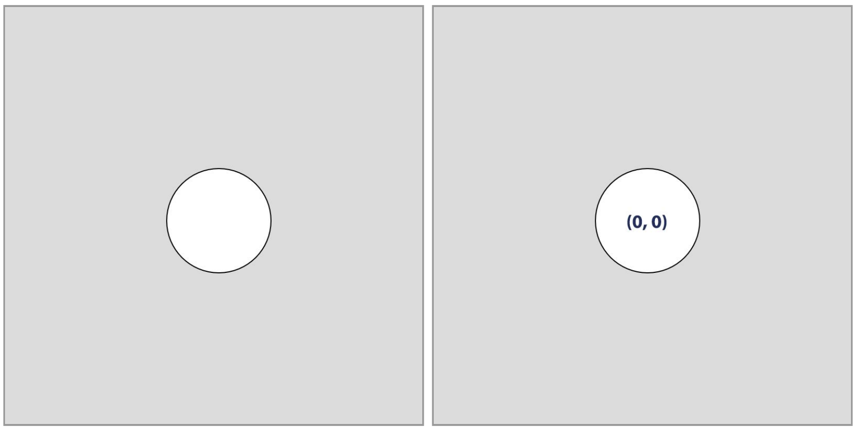


Figure 29. Draw an ellipse in the middle of canvas with Cartesian coordinate system

Drawing an ellipse with p5.Polar library requires the angle, width and height of radius, and distance from the center point (see Figure 30). At this point, the code is fairly similar to what is required in p5.js:

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipse(0, 50, 50, 0);
}
```

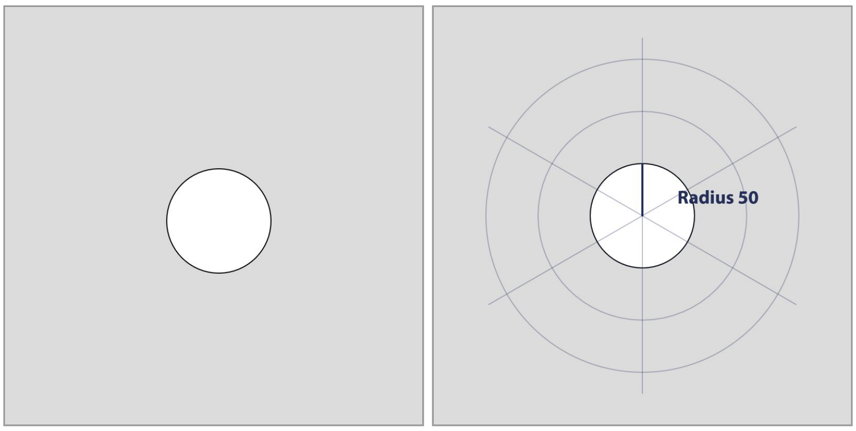


Figure 30. Draw an ellipse in the middle of canvas with polar coordinate system

The `polar.setCenter()` function provided by the p5.Polar library works the same as the p5.js `translate()` function but uses a more intuitive naming Users can and should set the center of the polar coordinate system before drawing any shapes.

Drawing a triangle with the p5.js built-in `triangle()` function requires three pairs of xy coordinates to represent three connected points on the canvas (see Figure 31):

```
function draw() {
  translate(width/2, height/2);
  triangle(-43, 26, 0, -50, 43, 26);
}
```

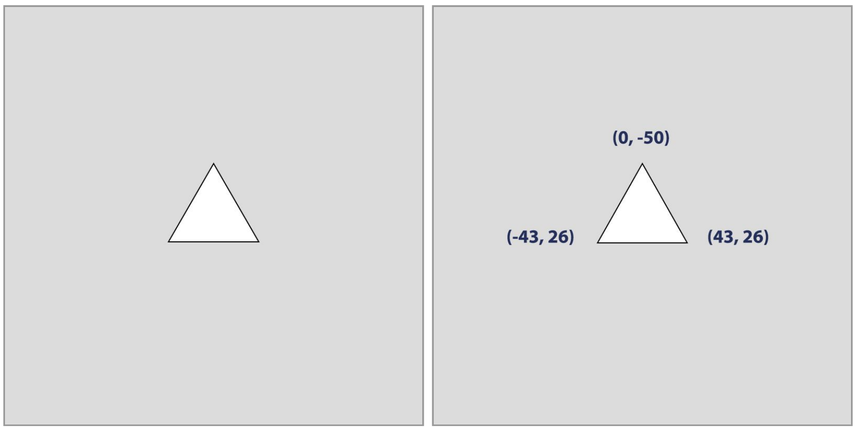


Figure 31. Draw a triangle with Cartesian coordinate system

Drawing a triangle with p5.Polar library requires the angle, radius, and distance from the center point (see Figure 32):

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarTriangle(0, 50, 0);
}
```

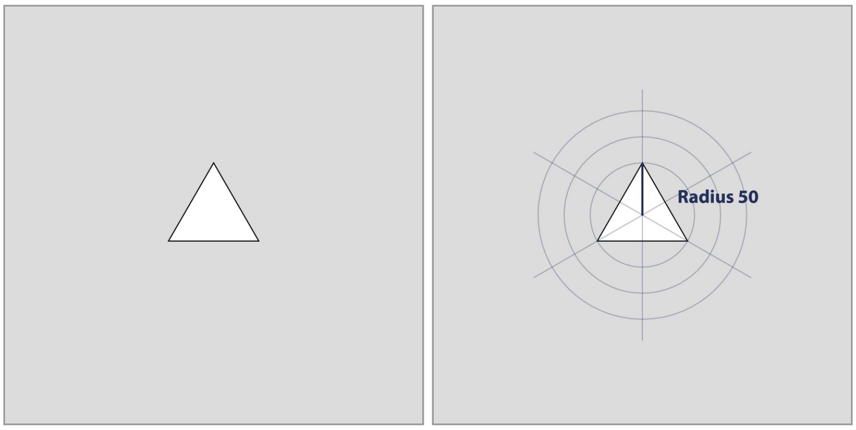


Figure 32. Draw a triangle with polar coordinate system

In the example using Cartesian coordinates, even though we're not using trigonometry to draw a triangle with p5.js, the complexity of giving three pairs of x-coordinate and y-coordinate values is time consuming for programmers to determine.. And while drawing a triangle requires three pairs of Cartesian coordinates, polygons with more edges requires users to come up with even more pairs of x and y coordinates, increasing the difficulty of the task. Drawing with polar coordinates simplifies this process, although the current implementation in p5.Polar only supports equilateral triangles.

Drawing a ring of ellipses with p5.js built-in `ellipse()` function requires six pairs of xy coordinates, and six pairs specifying the width and height of each ellipse see Figure 33):

```
function draw() {
  translate(width/2, height/2);
  ellipse(0, -100, 100, 100);
  ellipse(0, 100, 100, 100);
  ellipse(86, -50, 100, 100);
  ellipse(86, 50, 100, 100);
  ellipse(-86, 50, 100, 100);
  ellipse(-86, -50, 100, 100);
}
```

Adding a for loop shortens the required code but requires knowledge of trigonometry to correctly position the ellipses in a ring (see Figure 33):

```
function draw() {
  for(let i=0; i<6; i++) {
    resetMatrix();
    translate(width/2, height/2);
    let angle = (TWO_PI/6)*i;
    translate(sin(angle)*100, cos(angle)*100);
    ellipse(0, 0, 100, 100);
  }
}
```

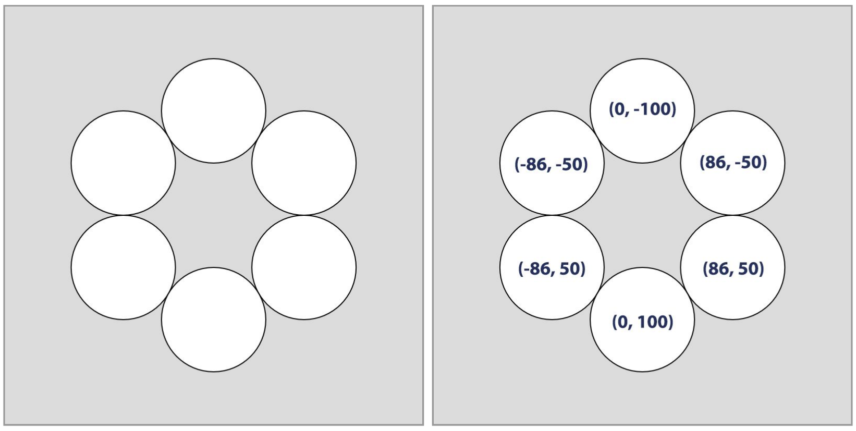


Figure 33. Draw a ring of ellipses with Cartesian coordinate system

Drawing a ring of ellipses with p5.Polar library only requires the number of ellipses, the width and height of the ellipses, and the distance of each ellipse from the center point, greatly simplifying the resulting code (see Figure 34):

```
function draw() {
  polar.setCenter(width/2, height/2);
  polarEllipses(6, 50, 50, 100);
}
```

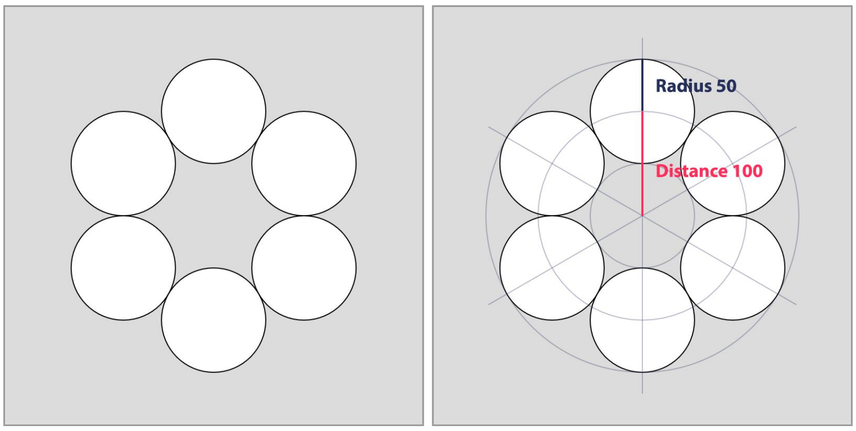


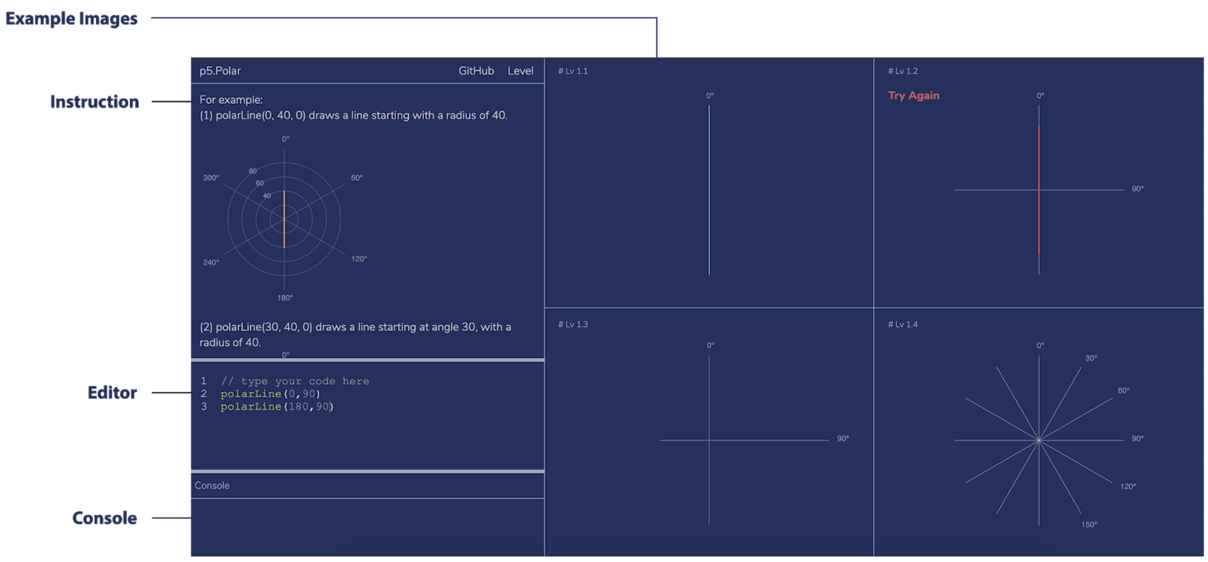
Figure 34. Draw a ring of ellipses with polar coordinate system

p5.Polar web application evaluation

The first formative playtesting of the tutorial game involved four friends that are traditional programming language users (C++, Java) and had never heard of p5.js or Processing. This playtesting focused on getting user interface and functionality feedback. During the playtesting, we asked the testers to play 3 levels of the game, and verbally share what they were thinking after finishing the third level. The feedback we got from the first playtesting session included:

Changing the hint section's color to improve the visibility.
Need detailed explanations for each function's parameters.
Hope the game can provide an example of a polar coordinate drawing concept for a person who couldn't associate drawing lines with radius.
Improve the visibility of lines and shapes after users compile the code.
Simplify the introduction page.
Provide a pure space for drawing without limitation by gaming mechanics.

Based on feedback from the first playtesting, we made a few tweaks to improve the color visibility by changing the color scheme, retyping task information, and providing polar coordinate drawing examples to clarify the task of each level and improve the way we teach p5.Polar drawing functions; these changes can be seen in Figures 35 and 36. We also added a playground for experimenting with pattern creation outside the context of the tutorial game, as shown in Figures 37:

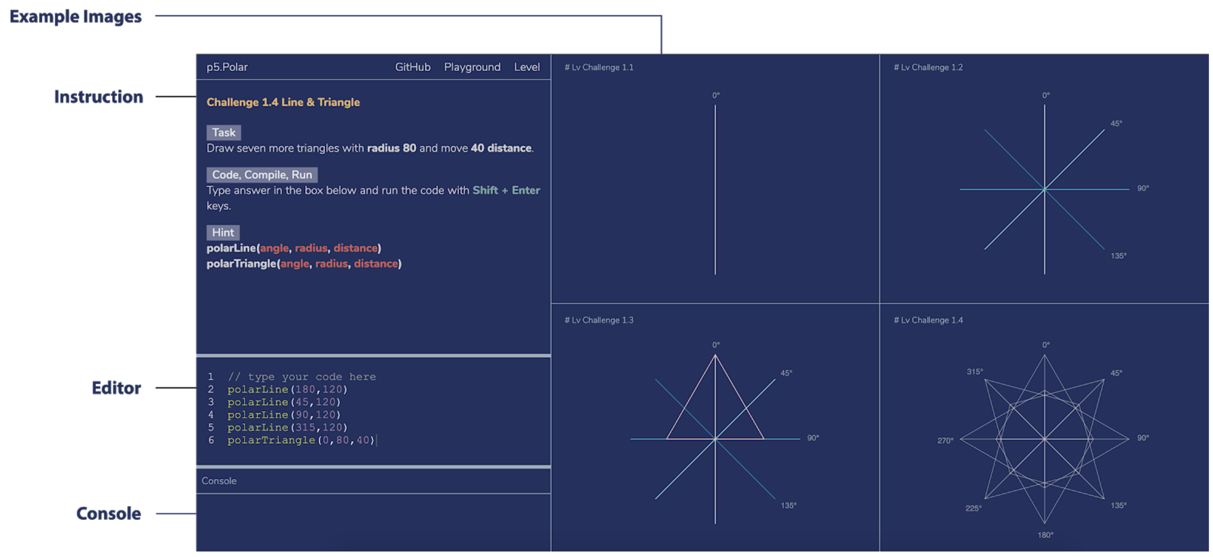


For example:
(1) polarLine(0, 40, 0) draws a line starting with a radius of 40.

(2) polarLine(30, 40, 0) draws a line starting at angle 30, with a radius of 40.

```
// type your code here
polarLine(0,90)
polarLine(180,90)
```

Figure 35. p5.Polar learning application prototype



Challenge 1.4 Line & Triangle

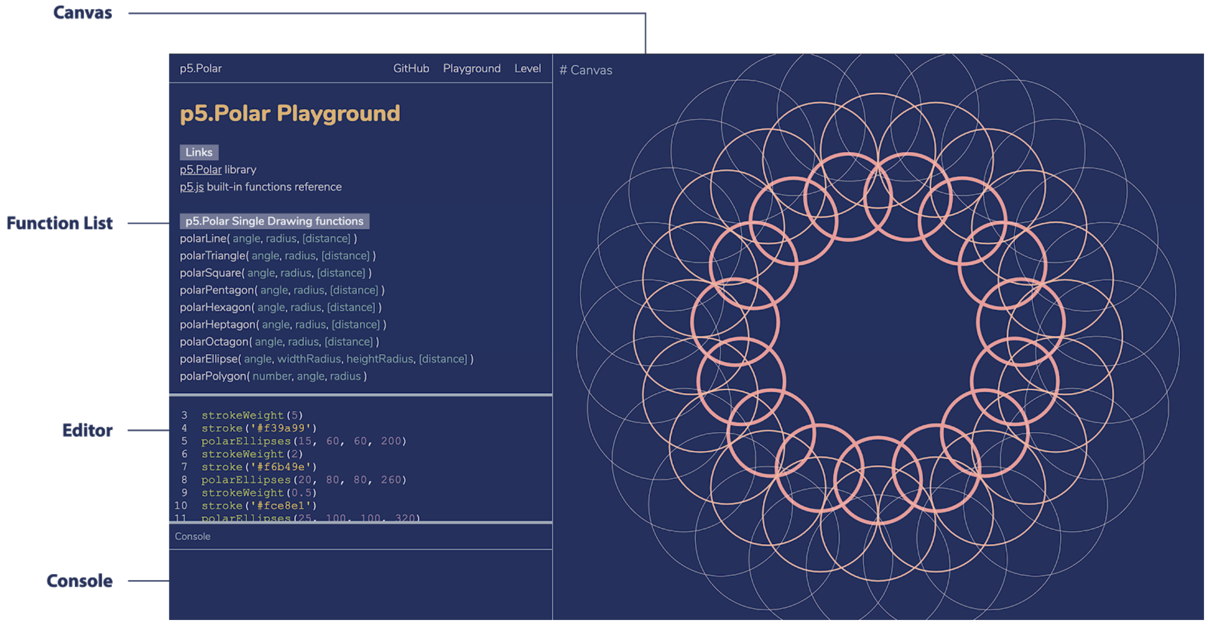
Draw seven more triangles with **radius 80** and move **40 distance**.

Type answer in the box below and run the code with **Shift + Enter** keys.

polarLine(angle, radius, distance)
polarTriangle(angle, radius, distance)

```
// type your code here
polarLine(180,120)
polarLine(45,120)
polarLine(90,120)
polarLine(315,120)
polarTriangle(0,80,40)
```

Figure 36. p5.Polar learning application prototype



p5.Polar Playground

p5.Polar library
p5.js built-in functions reference

polarLine(angle, radius, [distance])
polarTriangle(angle, radius, [distance])
polarSquare(angle, radius, [distance])
polarPentagon(angle, radius, [distance])
polarHexagon(angle, radius, [distance])
polarHeptagon(angle, radius, [distance])
polarOctagon(angle, radius, [distance])
polarEllipse(angle, widthRadius, heightRadius, [distance])
polarPolygon(number, angle, radius)

```
strokeWeight(5)
stroke('#f39a99')
polarEllipses(15, 60, 60, 200)
strokeWeight(2)
stroke('#f6b49e')
polarEllipses(20, 80, 80, 260)
strokeWeight(0.5)
stroke('#fce8e1')
polarEllipses(25, 100, 100, 320)
```

Figure 37. p5.Polar learning application - playground prototype

After improving the user interface and functionality of the game and obtaining IRB approval, we conducted more formal playtesting. This was constrained by the necessity of doing online playtesting due to COVID-19, and focused on a user experience study. Seven people participated in the second round of playtesting including five college students (see Figure 38). The survey was conducted after playtesters finished playing three levels of the game, including levels teaching how to use lines, triangles, and one challenge level where players were asked to replicate a more complex pattern combining multiple shapes. All three levels of the game only required players to use functions that draw individual shapes. To summarize the playtester's programming background, 5 of the testers code about 1–3 times a week, while two testers code 1–3 times a month (see Figure 39); thus, all the participants had coding experience. When examining p5.js coding experience, only one playtester uses p5.js 1–3 times a month, the rest of them had never heard of p5.js (see Figure 40). When examining Processing (the Java ancestor of p5.js) coding experience, one playtester uses Processing 1–3 times a month, one uses Processing 1–3 times a year, while the remaining five playtesters had never heard of Processing (see Figure 41).

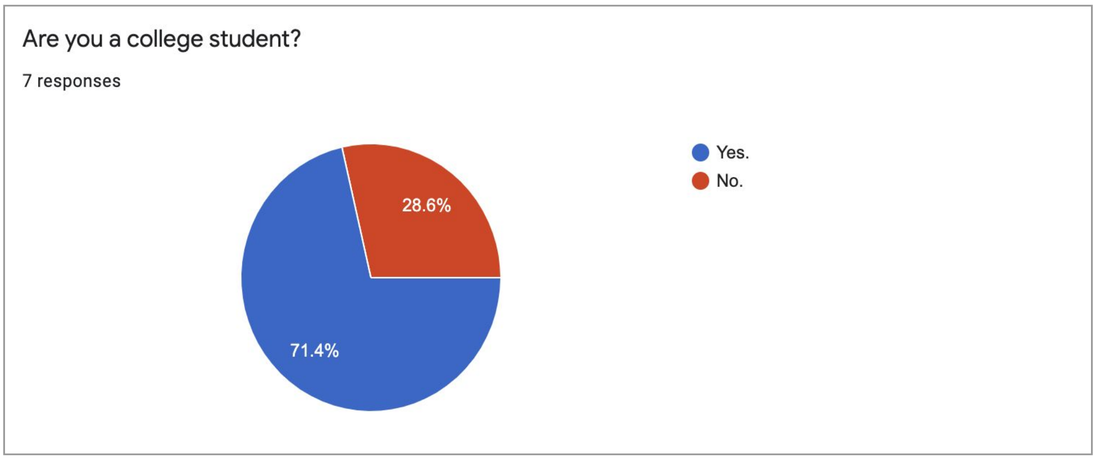


Figure 38. p5.Polar web application survey

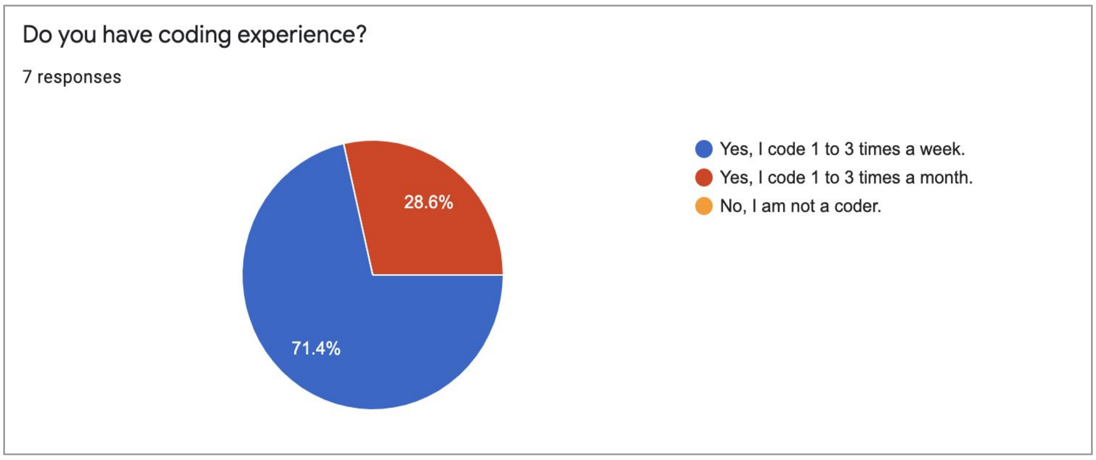


Figure 39. p5.Polar web application survey

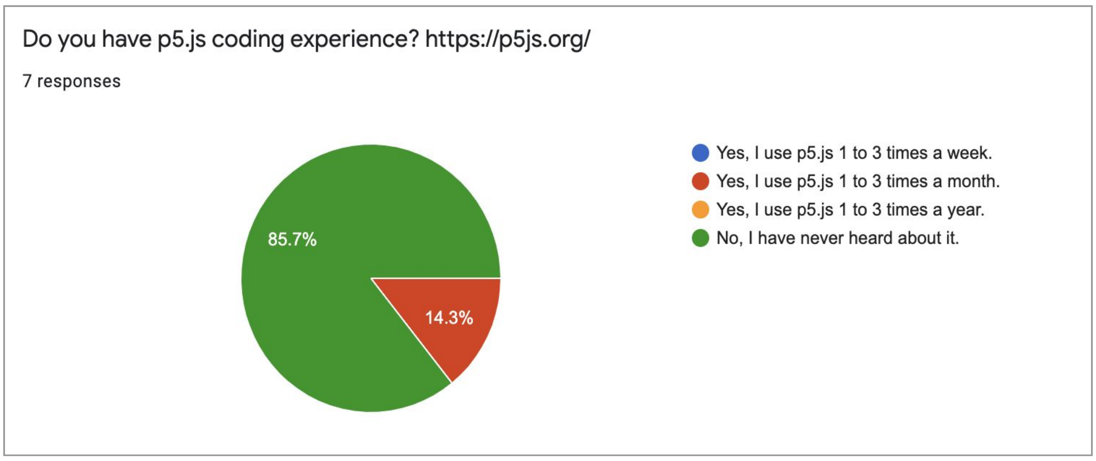


Figure 40. p5.Polar web application survey

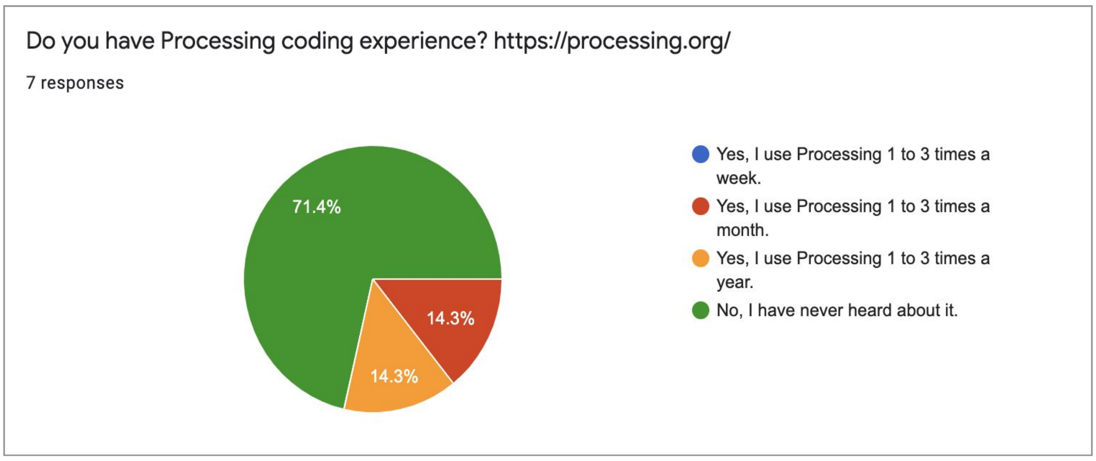


Figure 41. p5.Polar web application survey

We wanted to assess if playtesters understood the concept of drawing with a polar coordinate system after playing the game. We asked them to examine two images, one of the images indicates the polar coordinate system and the other one is the Cartesian coordinate system (see Figure 42). Then we asked the testers which one of the images represents the polar coordinate system. All the 7 playtesters got the correct answer (see Figure 43); this is by no means a definitive assessment but is a strong indication that players had basic intuition about the concept of polar coordinates.

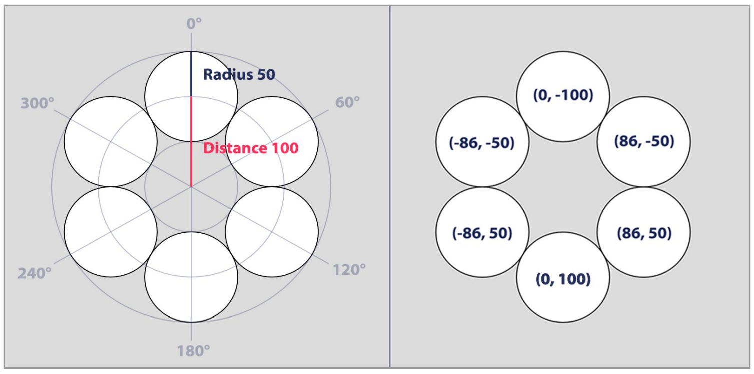


Figure 42. p5.Polar web application survey

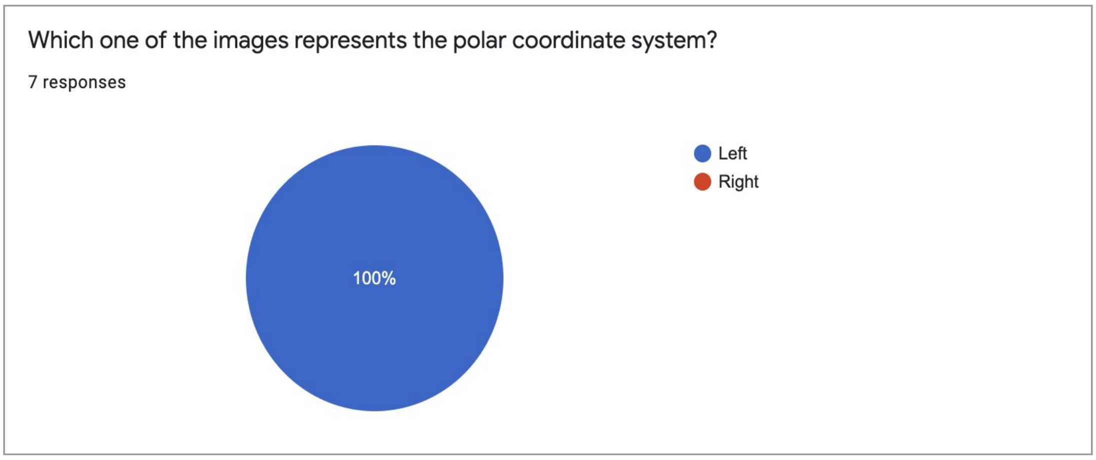
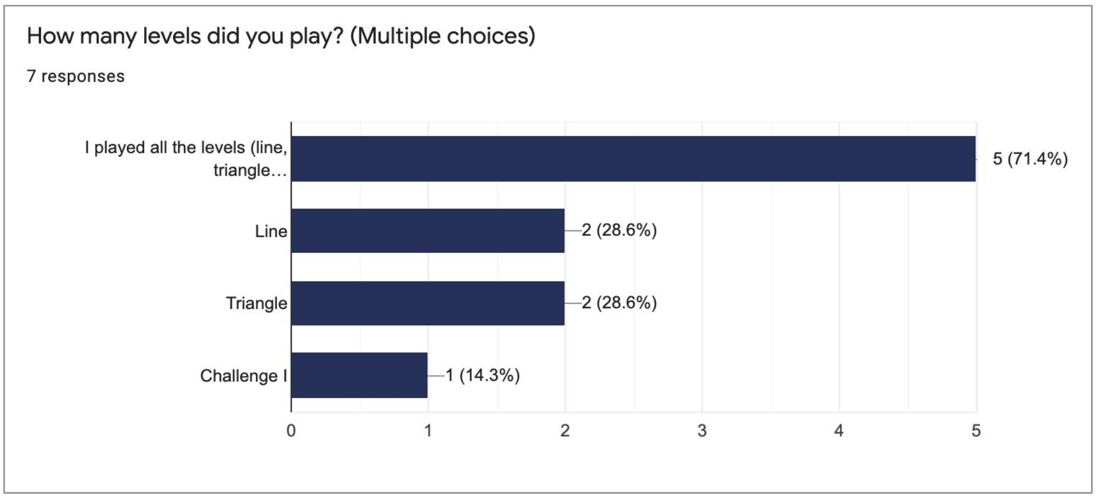


Figure 43. p5.Polar web application survey

After the general background questions, playtesters headed over to the second part, which assessed the user interface and user experience. As shown in Figure 44, Six playtesters played through all three levels, while one playtester encountered technical problems and only played two levels (these problems will be discussed later in this section).



I played all the levels (line, triangle and challenge I)
I played all the levels (line, triangle and challenge I)
I played all the levels (line, triangle and challenge I)
I played all the levels (line, triangle and challenge I)
Line, Triangle, Challenge I
Line, Triangle
I played all the levels (line, triangle and challenge I)

Figure 44. p5.Polar web application survey

The survey separated each component and functionality in the game, and asked about the experience of the introduction section first. For the “Task” section (see Figure 45), 4 playtesters felt the task section was clear, while 3 of them chose neutral (see Figure 46). In the "Code, Compile, Run" section (see Figure 47), all the playtesters feltl the instruction was clear (see Figure 48). For the “Hint” section (see Figure 49), 4 playtesters felt the explanation was clear, while 3 of them chose neutral (see Figure 50). In the “example” section (see Figure 51), all playtesters felt the information was clear (see Figure 52).

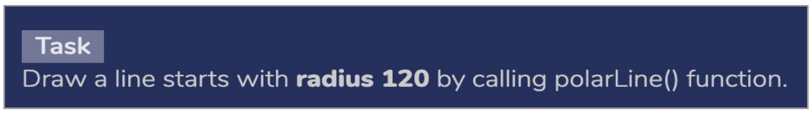

Task
Draw a line starts with **radius 120** by calling polarLine() function.

Figure 45. p5.Polar web application survey

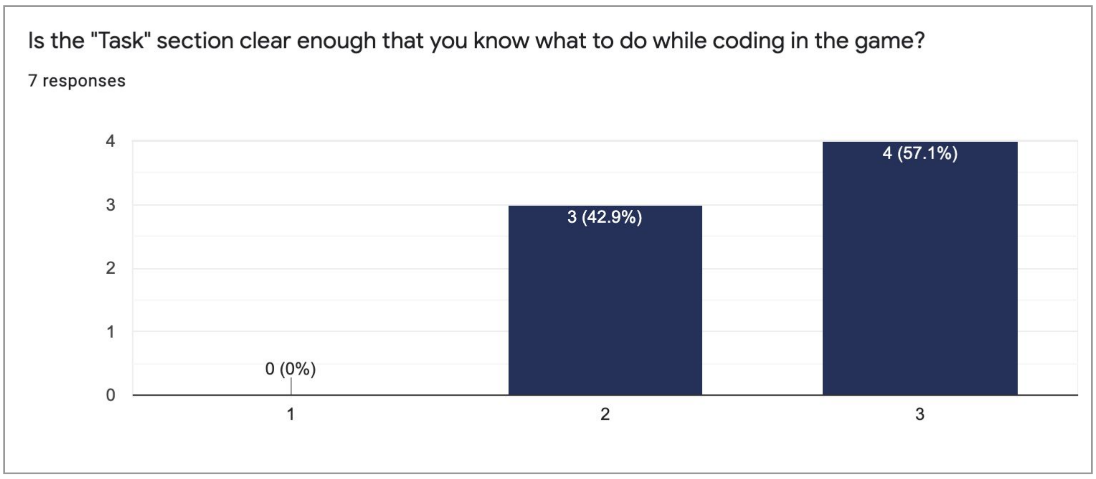


Figure 46. p5.Polar web application survey

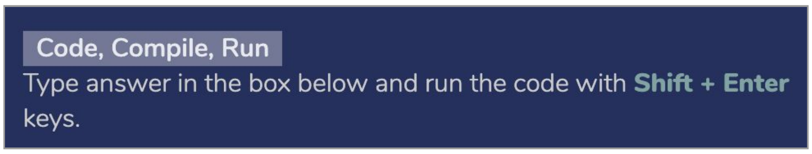

Code, Compile, Run
Type answer in the box below and run the code with **Shift + Enter** keys.

Figure 47. p5.Polar web application survey

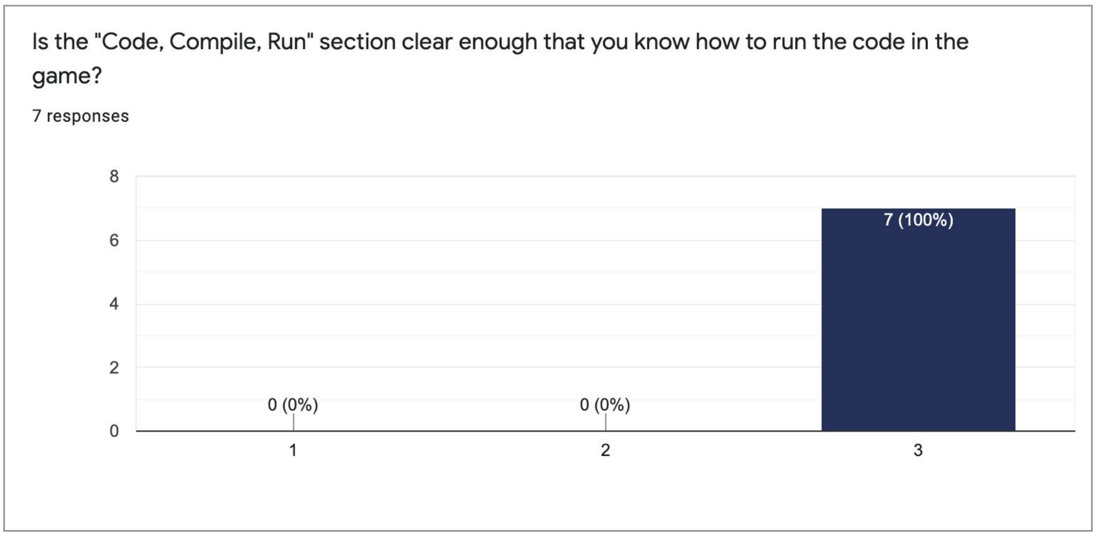


Figure 48. p5.Polar web application survey

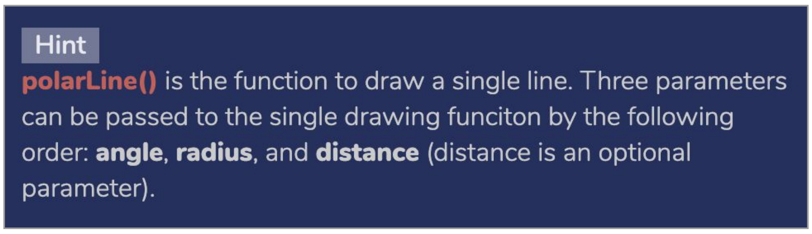

Hint
polarLine() is the function to draw a single line. Three parameters can be passed to the single drawing funciton by the following order: **angle**, **radius**, and **distance** (distance is an optional parameter).

Figure 49. p5.Polar web application survey

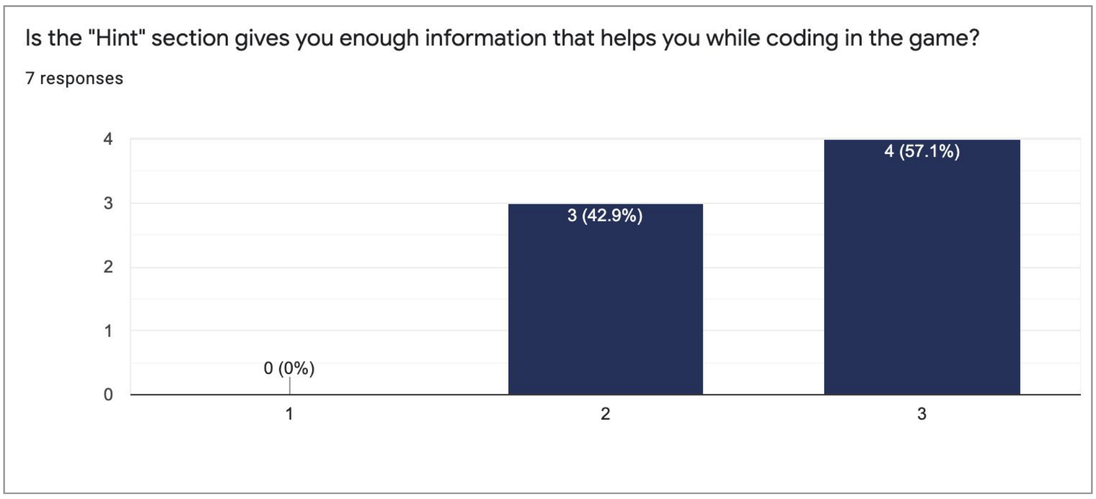
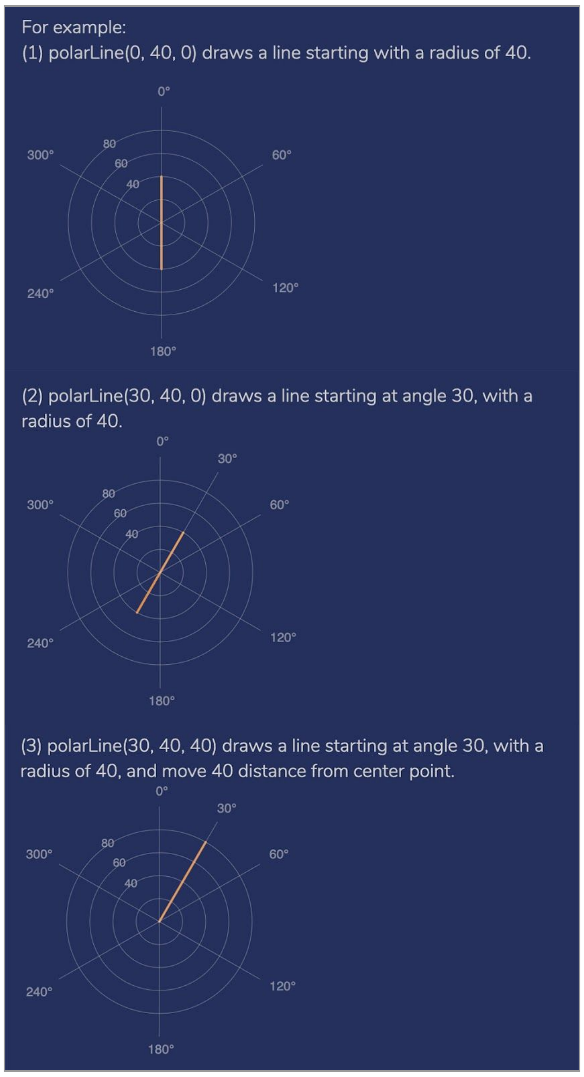


Figure 50. p5.Polar web application survey


For example:
(1) polarLine(0, 40, 0) draws a line starting with a radius of 40.

(2) polarLine(30, 40, 0) draws a line starting at angle 30, with a radius of 40.

(3) polarLine(30, 40, 40) draws a line starting at angle 30, with a radius of 40, and move 40 distance from center point.

Figure 51. p5.Polar web application survey

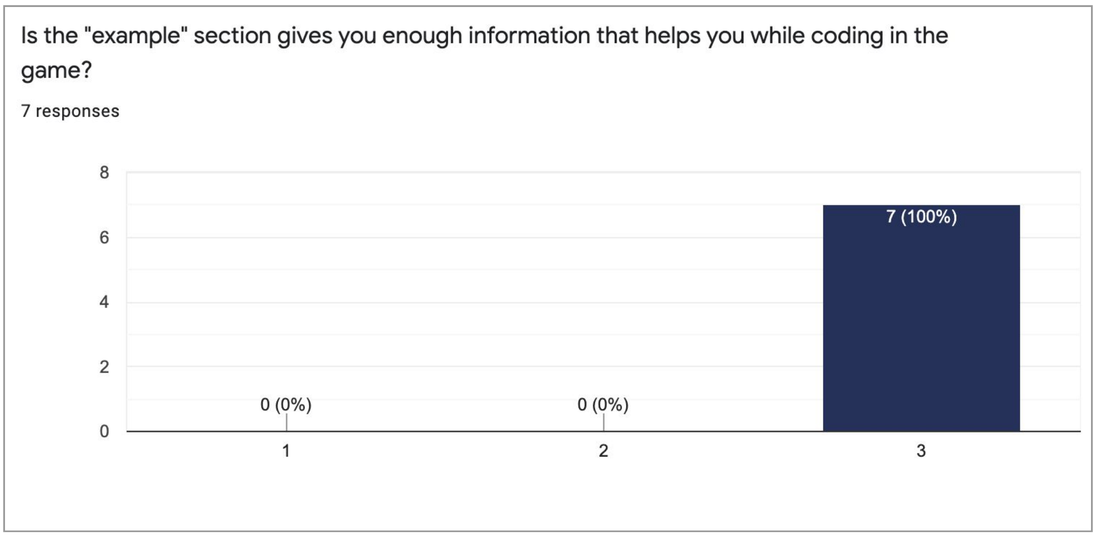
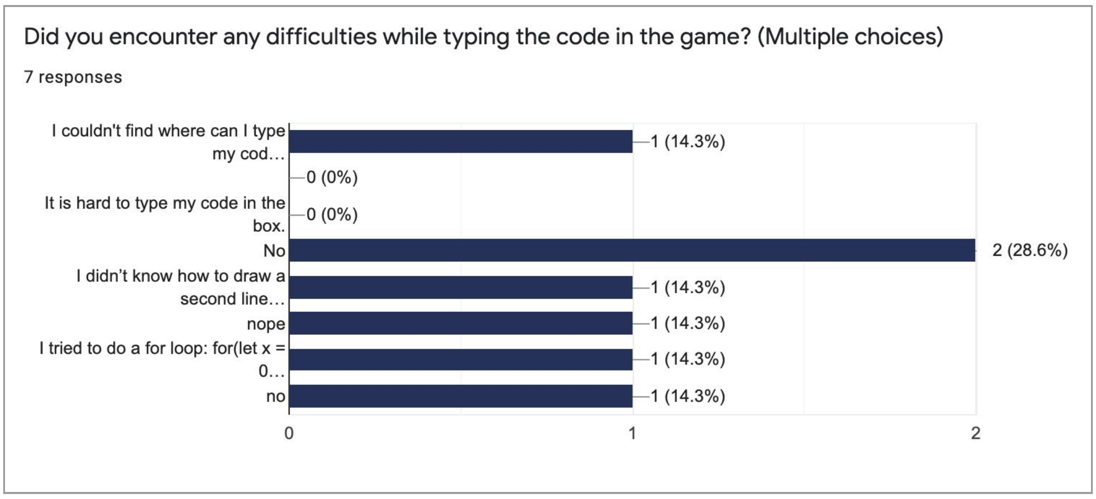


Figure 52. p5.Polar web application survey

In the rest of the second part of the survey, we asked about their overall experience using the tutorial game and playground. First, we wanted to know if playtesters encountered any difficulties in the game (see Figure 53). One playtester mentioned that he/she didn't know how to draw the second line in the first level, one playtester mentioned about getting the error when coding with for loops, and one playtester couldn’t find the place to type the code. In the “Console” section (see Figure 54), two playtesters ran into error messages and the resulting error messages helped them with debugging, while the other two playtesters also ran into error messages but felt the error messages did not give enough information to help them debug. The other three playtesters reported not seeing any error messages (see Figure 55).

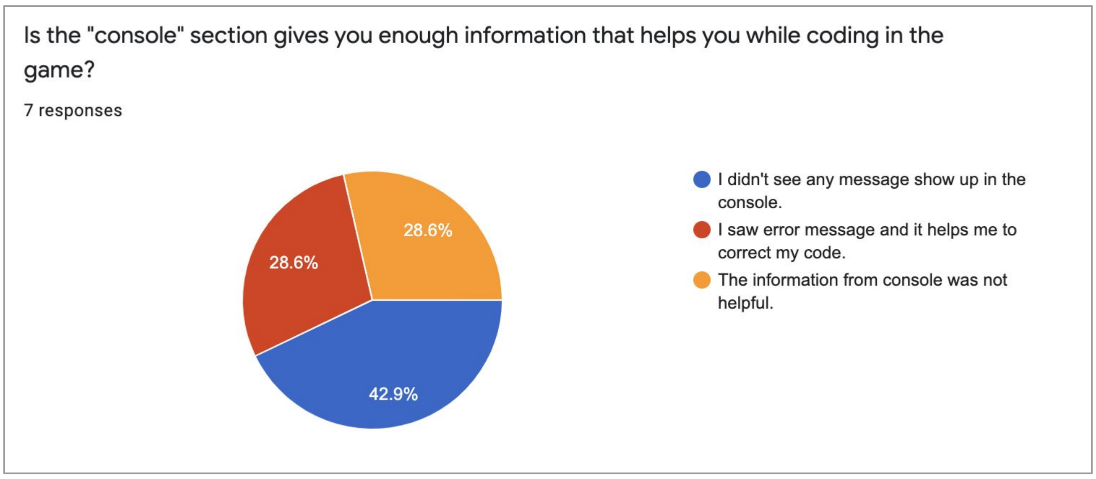


I didn’t know how to draw a second line at first; but I tried and erred and figured it out at the end.
No
No
nope

I tried to do a for loop: for(let x = 0; x <=5; x++){polarLine(x*30, 120, 0)}, but it didn't compile correctly. I checked and the for loop was written correctly
I couldn't find where can I type my code.
no

Figure 53. p5.Polar web application survey

Figure 54. p5.Polar web application survey


Is the "console" section gives you enough information that helps you while coding in the game?

7 responses

Figure 55. p5.Polar web application survey

For the “example images” experience (see Figure 56), all the playtesters felt the angle indications helped them while drawing. Six playtesters felt the images generally helped them with drawing and none of the playtesters stated that the angle or images were not helpful; one playtester did not have an opinion on the matter (see Figure 57). In the “Try Again” and “Pass” section (see Figure 58), all the playtesters agreed that the information is clearly indicated if they passed a given level (see Figure 59).

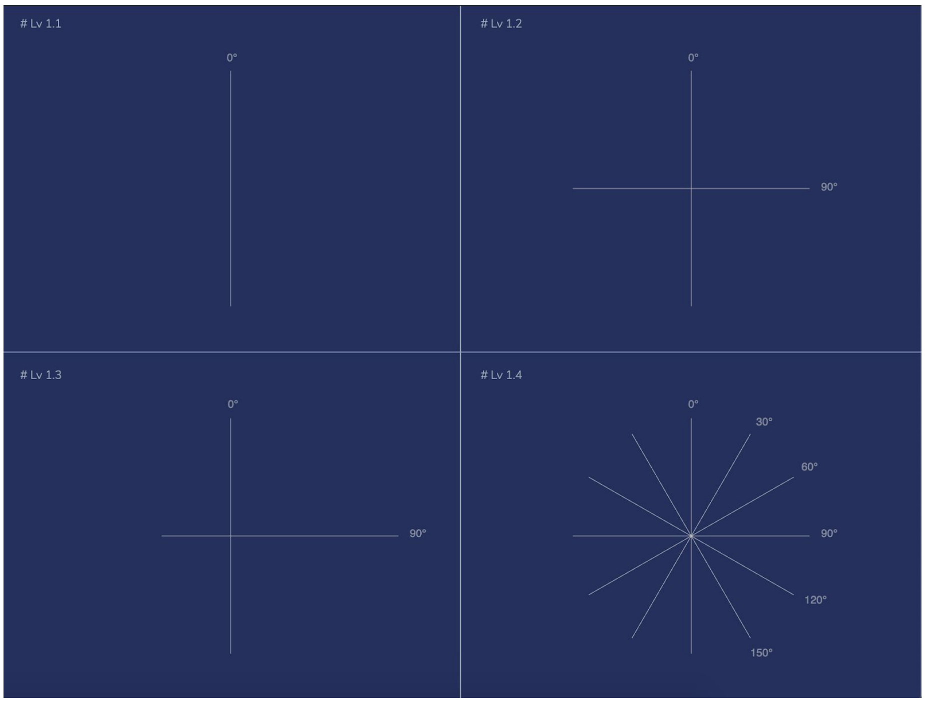
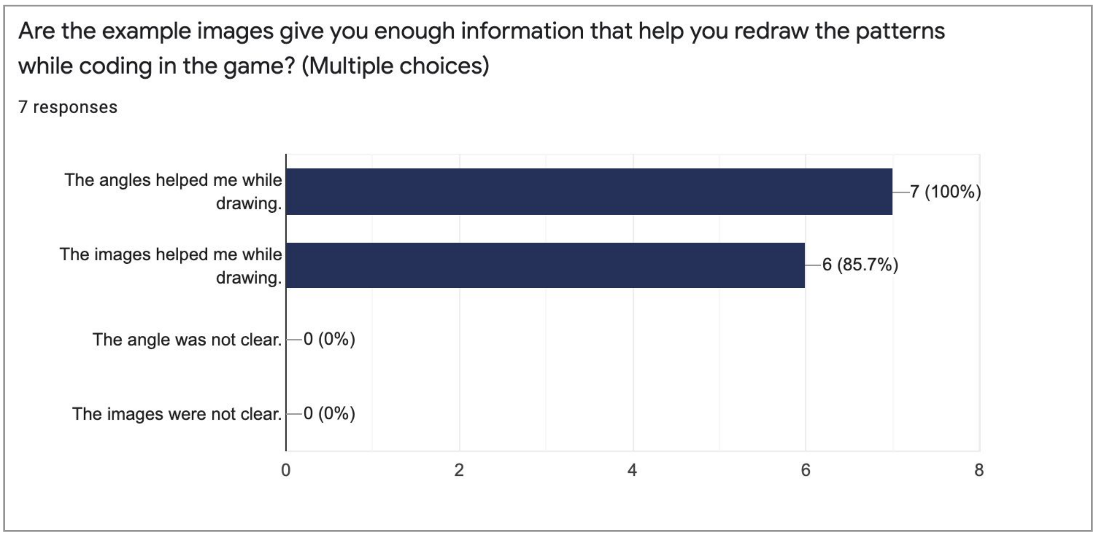


Figure 56. p5.Polar web application survey


Are the example images give you enough information that help you redraw the patterns while coding in the game? (Multiple choices)

7 responses

The angles helped me while drawing.	7 (100%)
The images helped me while drawing.	6 (85.7%)
The angle was not clear.	0 (0%)
The images were not clear.	0 (0%)

Figure 57. p5.Polar web application survey

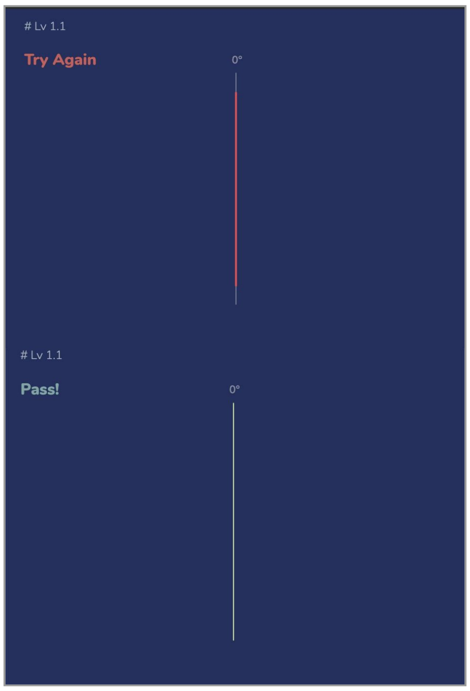
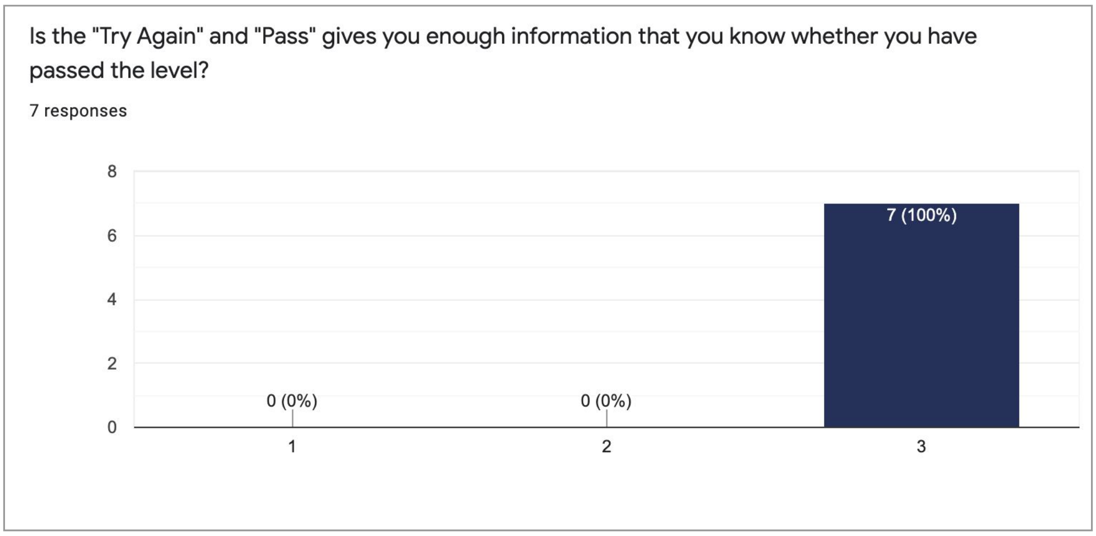


Figure 58. p5.Polar web application survey



Is the "Try Again" and "Pass" gives you enough information that you know whether you have passed the level?

7 responses

Figure 59. p5.Polar web application survey

After the functionality questions, we wanted to assess playtesters enjoyment of the system. Scaled from 1 to 5, four playtesters enjoyed the game with a rating of 4, while three players really enjoyed the game and gave it a rating of 5 (see Figure 60). We also wanted to assess if the game encourages playtesters to think about drawing in a new way. Five playtesters agreed with that statement, while one playtester disagreed and the final playtester had been using polar coordinates for many years; drawing with polar coordinate system was not new for them (see Figure 61).

The second to last question asked what kind of project would playtesters would use the p5.Polar library for in the future. Responses included creating a Mandala drawing, creating a project related to kids, and kaleidoscope patterns (see Figure 62). The last question is asked playtesters for unstructured thoughts and feedback. One playtester suggested creating more levels, one playtester felt the drawing was boring, and one playtester pointed out a bug in the game. Finally, one playtester encounters a layout bug (see Figure 63).

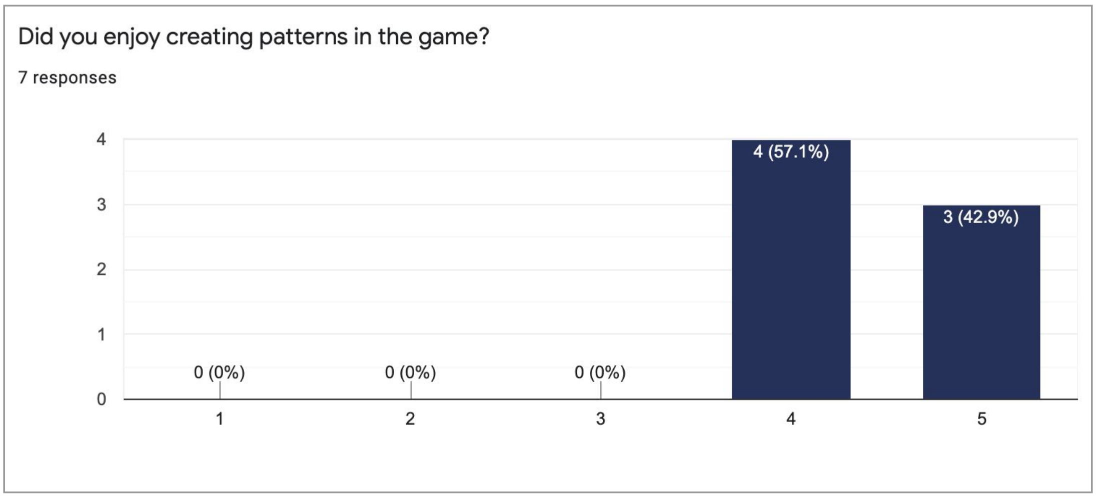


Figure 60. p5.Polar web application survey

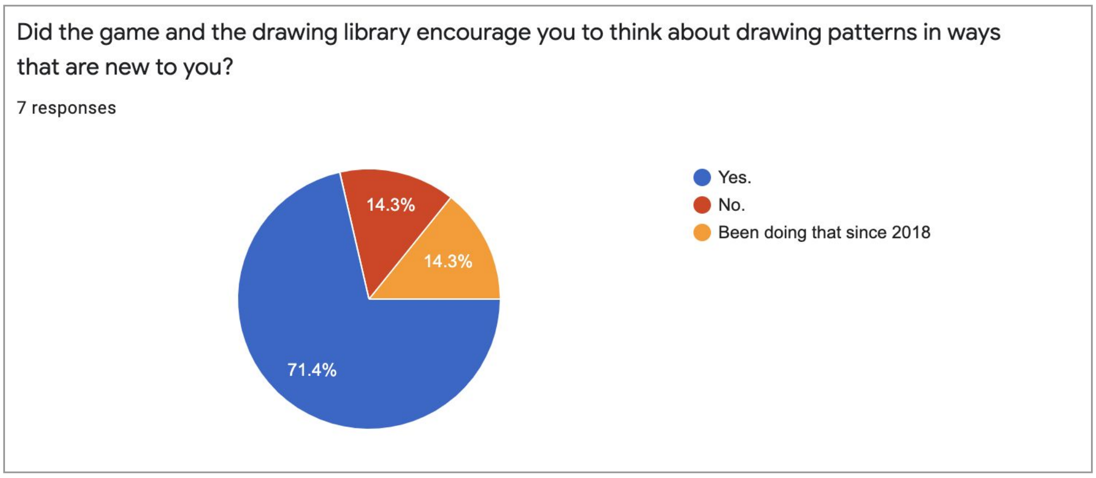
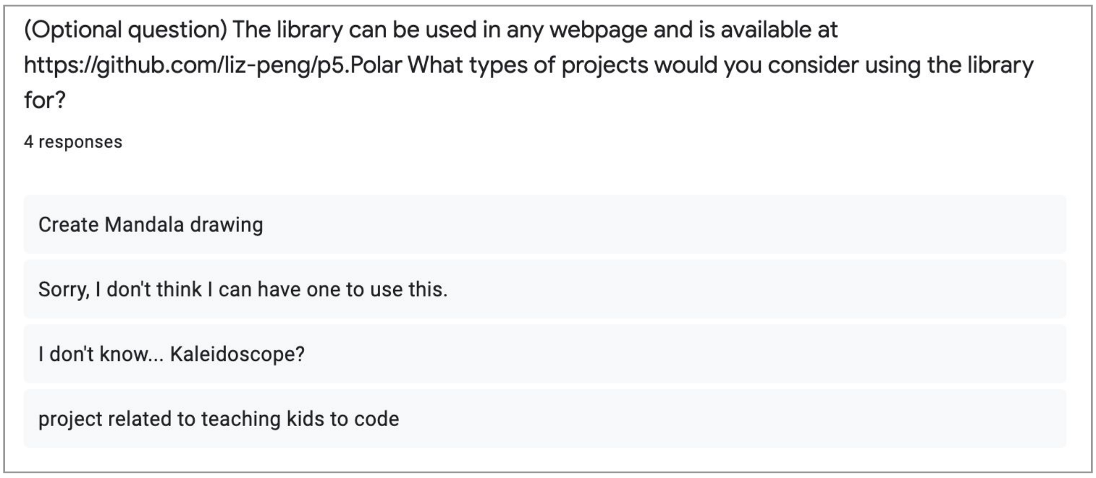


Figure 61. p5.Polar web application survey


(Optional question) The library can be used in any webpage and is available at https://github.com/liz-peng/p5.Polar What types of projects would you consider using the library for?

4 responses

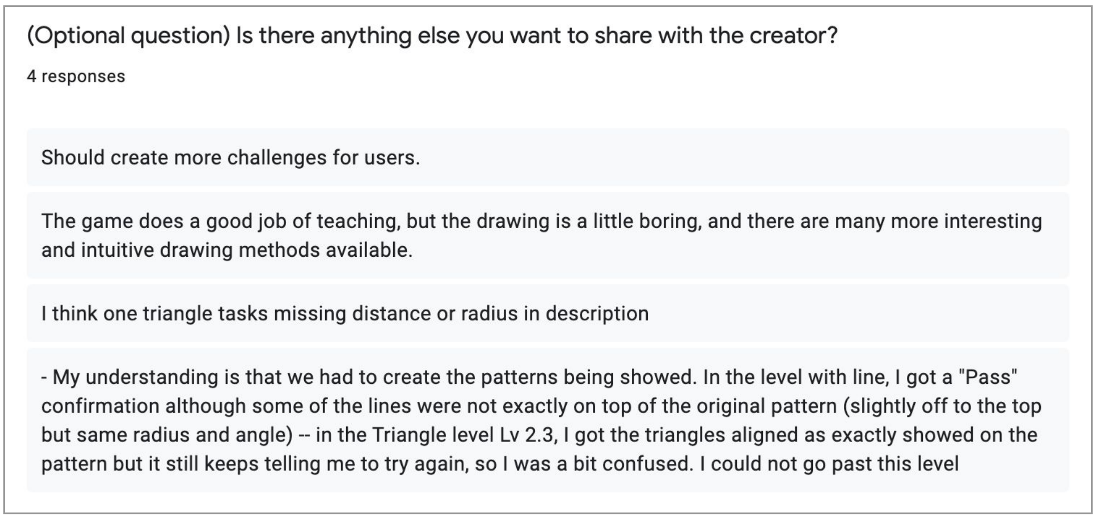
Create Mandala drawing

Sorry, I don't think I can have one to use this.

I don't know... Kaleidoscope?

project related to teaching kids to code

Figure 62. p5.Polar web application survey


(Optional question) Is there anything else you want to share with the creator?

4 responses

Should create more challenges for users.

The game does a good job of teaching, but the drawing is a little boring, and there are many more interesting and intuitive drawing methods available.

I think one triangle tasks missing distance or radius in description

- My understanding is that we had to create the patterns being showed. In the level with line, I got a "Pass" confirmation although some of the lines were not exactly on top of the original pattern (slightly off to the top but same radius and angle) -- in the Triangle level Lv 2.3, I got the triangles aligned as exactly showed on the pattern but it still keeps telling me to try again, so I was a bit confused. I could not go past this level

Figure 63. p5.Polar web application survey

Overall, we were happy that playtesters enjoyed the application, and that it encouraged them to think about procedural drawing in a new way. As one playtester suggested creating projects for kids, an interesting future goal for p5.Pola would be to create a tutorial system that both encouraged children to play with generating patterns and also taught them the underlying trigonometry involved.

Conclusion

Web-based, interactive environments have become popular in recent years for teaching programming and computational thinking. For those that teach graphics programming and procedural drawing, the immediate visual feedback that they provide encourages and motivates students to learn, and reduces the barrier for learning programming languages. p5.js offers an online learning environment which encourages sketching with code, just like layering with paints on paper. The study contributes to the p5.js community with a new JavaScript library, p5.Polar, that provides a way of drawing patterns with a polar coordinate system to reduce lines of code and abstract mathematics away from beginning programmers. The project has been formally submitted to the Processing foundation in the hope that it will become an official p5.js community library. Our study also presents a gamified web-based learning environment that aims to provide an easily accessible space for drawing creative patterns with polar coordinate systems, and uses game mechanics to encourage players to make abstract patterns in an interactive yet interesting way.

Based on our evaluation results, we plan a number of future improvements to p5.Polar. Eventually, the game will support for loops / while loops for advanced users, instead of manually typing repetitive lines of code, and shall add indications to clarify the user interface and ensure the code typing box is easy to find. To improve the error console, we shall add customized error messages to help players debugging with the p5.Polar library; the error message currently only outputs JavaScript syntax related errors. We shall create more levels that might increase user enjoyment. Finally, we shall improve our software testing workflow, so that pushing new versions of our game will avoid user confusion.

References

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5), 487–501.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61.

McCarthy, L., Reas, C., & Fry, B. (2016). *Getting started with p5.js.* San Francisco, CA: Maker Media.

Reas, C., & Fry, B. (2006). Processing: programming for the media arts. *Ai & Society*, 20(4), 526–538.

Bhatti, S., Dewani, A., Maqbool, S., & Memon, M. A. (2019). A Web based Approach for Teaching and Learning Programming Concepts at Middle School Level. *International Journal of Modern Education and Computer Science*, 11(4), 46–53.

Asad, K., Tibi, M., & Raiyn, J. (2016). Primary School Pupils' Attitudes toward Learning Programming through Visual Interactive Environments. *World Journal of Education*, 6(5).

Maeda, J. (1999). *Design by numbers*. Cambridge, MA: MIT Press.

Reas, C., & Fry, B. (n.d.). Processing.org. Retrieved from https://processing.org/

McCarthy, L. (n.d.). p5.js. Retrieved from https://p5js.org/

Flexbox Froggy. (n.d.). Retrieved from https://flexboxfroggy.com/

CSS Grid Garden. (n.d.). Retrieved from https://cssgridgarden.com/

Turtle Academy. (n.d.). Retrieved from https://turtleacademy.com/