Design and Evaluation Report

Luke Hammond
Digital Media
University of the West of England
Coldharbour Lane
Bristol, UK
luke2.hammond@live.uwe.ac.uk

submitted 21/01/2022

Abstract
This report summarizes research in the area of the generative drawing project. This contains three separate p5,js sketches, the code for the images as well as a weekly research and design journal completed over 5 entries.

Author Keywords
translate(); rotate(); MODE(); loops; variable; function(); setup(); point(); beginShape(); endShape(); push(); pop(); array;

Introduction
This project is an opportunity to test my programming skills that have developed over the autumn term. The project allows for creative freedom while honing the skills I have acquired. For my project I have chosen the theme of space and through my research as well as workshops I have designed images, I believe suitable for that theme.

My research method contained looking at tutorials on YouTube, especially those made by 'The Coding Train' as well as looking into code on w3schools and p5.js reference page.

Research

Most of my research was completed via primary research, research I collected myself, as well as secondary research, using research that already exists. I used these methods as they were the most suitable and effective for my task. Most of my research was secondary and was achieved utilising the p5.js reference page as well as YouTube tutorials. I chose this method of research as I am still developing my programming skills and thus my research helped me learn new skills as well as further my development. In addition to this, my base level learning in the workshop also boosted my skills and was the initial development process. A significant amount of research was me referring back to old workshops. The people involved in this research were my lecturers, Dr Simon Emberton, Dr Dave Meckin as well Tom Garne.

Reference Image

This image was the basis of one of my works it is a very basic and simple outline of the solar system.

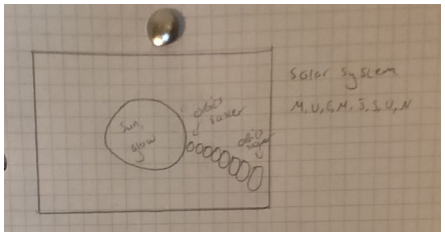


Figure 1. Initial design for SOLAR SYSTEM

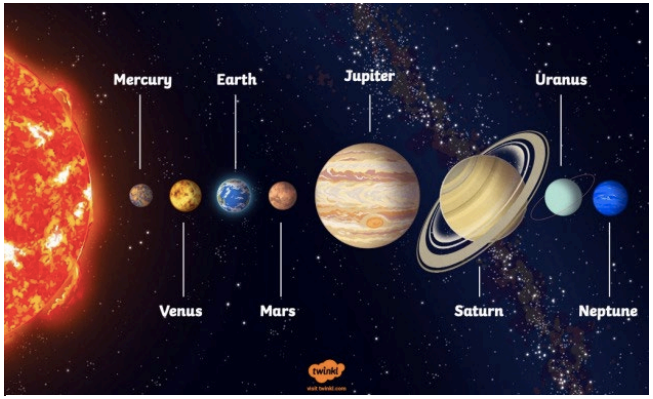


Figure 2. Reference image for SOLAR SYSTEM

manageable. The only problem with the image was the details on each planet, I was unsuccessful in making detailed planets, especially the rings that surround Saturn and Uranus.

Gradient

Gradients had been something I had tried in the past in one of the creative coding workshops however it did not suit my needs. Thus, I looked into it and my in my findings I found a YouTube tutorial.

```
beginShape();
vertex(20, 20);
vertex(45, 20);
vertex(45, 80);
endShape(CLOSE);

beginShape();
vertex(50, 20);
vertex(75, 20);
vertex(75, 80);
endShape();
```

Description

The endShape() function is the companion to beginShape() and may only be called after beginShape(). When endShape() is called, all of image data defined since the previous call to beginShape() is written into the image buffer. The constant CLOSE as the value for the MODE parameter to close the shape (to connect the beginning and the end).

Syntax

```
endShape([mode])
```

Parameters

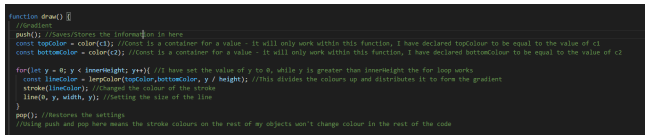
mode	Constant: use CLOSE to close the shape (Optional)

Figure 3. p5js reference page for beginShape() and endShape()

Figure 4. Canvas showcasing a gradient

The colour of the gradient is black to grey because of my theme being space.

The code for this gradient was simple and straightforward, it involved creating two variables – container for storing data and setting a value for them. Then using a for loop (a statement that is executed if the instructions are true) you make the gradient appear as each line of the canvas changes colour.



```
function draw() {
  //Gradient
  push(); //Saves/Stores the information is here
  const topColor = color(c1); //Const is a container for a value - it will only work within this function, I have declared topColour to be equal to the value of c1
  const bottomColor = color(c2); //Const is a container for a value - it will only work within this function, I have declared bottomColour to be equal to the value of c2

  for(let y = 0; y < innerHeight; y++){ //I have set the value of y to 0, while y is greater than innerHeight the for loop works
    const lineColor = lerpColor(topColor,bottomColor, y / height); //This divides the colours up and distributes it to form the gradient
    stroke(lineColor); //Changed the colour of the stroke
    line(0, y, width, y); //Setting the size of the line
  }
  pop(); //Restores the settings
  //Using push and pop here means the stroke colours on the rest of my objects won't change colour in the rest of the code
```

Figure 5. Code for the gradient background

Custom Shapes

For one of my images, I wanted to draw an irregular shape that wasn't a preset, to do this I went and watched 'The Coding Train' on YouTube.

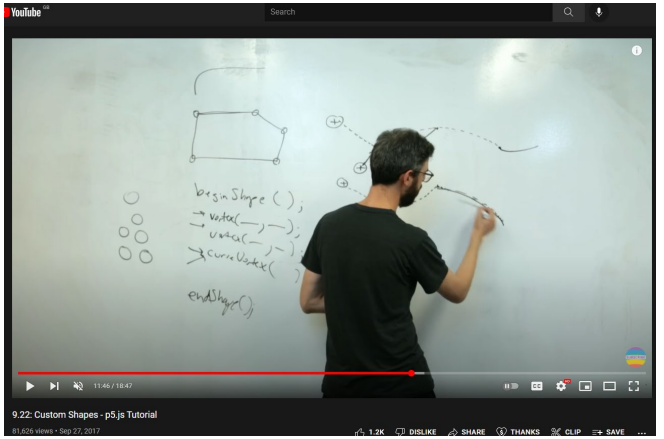


Figure 6. Video by 'The Coding Train' about custom shapes

I found it was a lot of coordinates, to make it simpler it was suggested to first start with point() – this draws a point on the canvas. This allowed for a basic outline and layout for where the lines would join once you began inputting coordinates within beginShape() – allows for more complex shapes by recording the vertices. The image would end once you inout endShape() – this stops vertices from being recorded. This has two separate MODES – you can either leave it with empty brackets or CLOSE. This joins up the final coordinate with the first coordinate. This MODE joins the shape to make it one.

Translate and Rotation

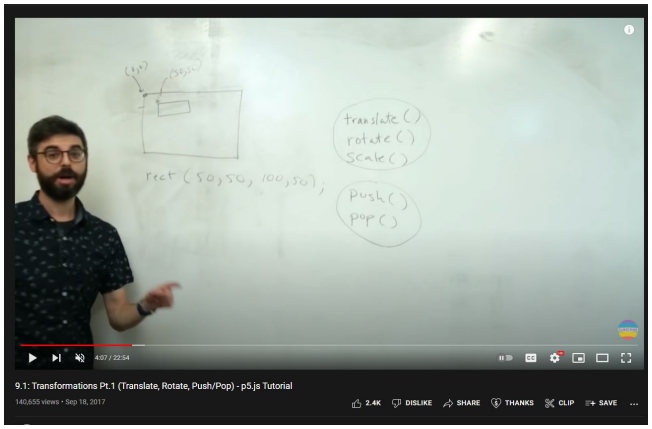


Figure 7. Video by 'The Coding Train' about transformations

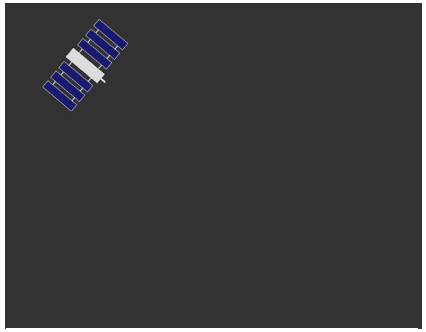


Figure 8. Image of a satellite showcasing translate and rotate

The translate() function enables the user to change the starting coordinate for their shape. The original starting coordinates is the top left of the page at 0px on the x axis and 0px on the y axis.

P5 works in radians, this can be changed if you change the angleMode() – this allows the user to change the preset angle mode from radians, in the setup() function – the initial code. I changed my mode to DEGREES as this was easier for me to understand. This now makes the shape rotate around the origin of your canvas, this can be changed with translate().

So this doesn't affect the rest of the work you must utilise the push() and pop() functions. Push() saves the code beforehand and pop() restores it. This means any changes made within the push() and pop() functions does not impact the rest of the work. For example, if a translate() value is changed within, it will only affect the coordinates of the shapes/objects within that push() and pop() function.

```
function draw(){
  background(50);
    //Satellite
    push(); //Save
    strokeWeight(1);//Setting the thickness of the stroke
    fill("#DCDCDC");//Colour is Gainsboro
    translate(240,180); // Setting the point for the shapes to be drawn around
    rotate(130); //Rotates the rectangle 130 degrees
    rect(0,0,30,100); //x and y axis was specified in 'translate'
    stroke(255); //Changed the stroke colour to white
    strokeWeight(4); //Increased the stroke weight to increase lines thickness
    line(15,-1,15,-11); //Placement of line from x1 - x2 and y1 - y2
    pop(); //Restore
```

Figure 9. Partial part of the code for the satellite shown in Figure 8

Particles

Following my theme, I wanted to add stars as one of my images. I took inspiration from the work achieved in workshop week 8. This involved classes – a template for objects, arrays – a variable that can store more than one value, and loops.

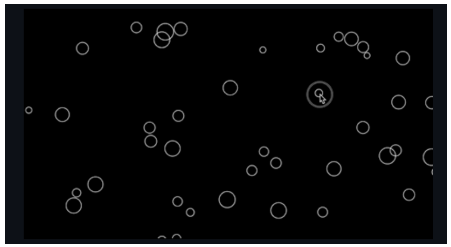


Figure 10. Example of the jitter from workshop week 8

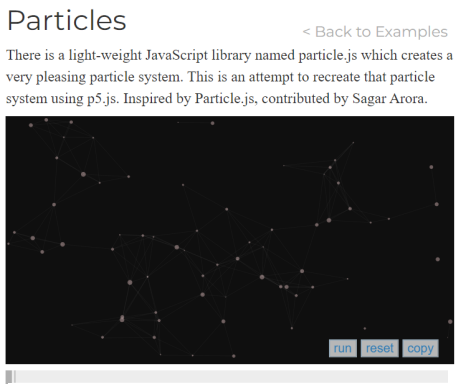


Figure 11. P5js examples page on particles

However, I had also seen an example of particles on the p5.examples page that I also wanted to incorporate into my work alongside the jitter I made in the workshop. This involved combining the code which meant the shapes would bounce around the screen but also appear when the mouse was pressed.

```
move() { //function move
  if(this.x < 0 || this.x > width) //If this.x is less than 0 or greater than width its movement changes - causing it to bounce around the screen
    this.xSpeed*=-1;
  if(this.y < 0 || this.y > height) //If this.y is less than 0 or greater than height its movement changes - causing it to bounce around the screen
    this.ySpeed*=-1;
  this.x+=this.xSpeed; //Causes the ellipse to bounce around the screen as it the values state where the object is
  this.y+=this.ySpeed; //Causes the ellipse to bounce around the screen as it the values state where the object is
}
```

Figure 12. The code for the movement speed of the particles

Evaluation

Earth

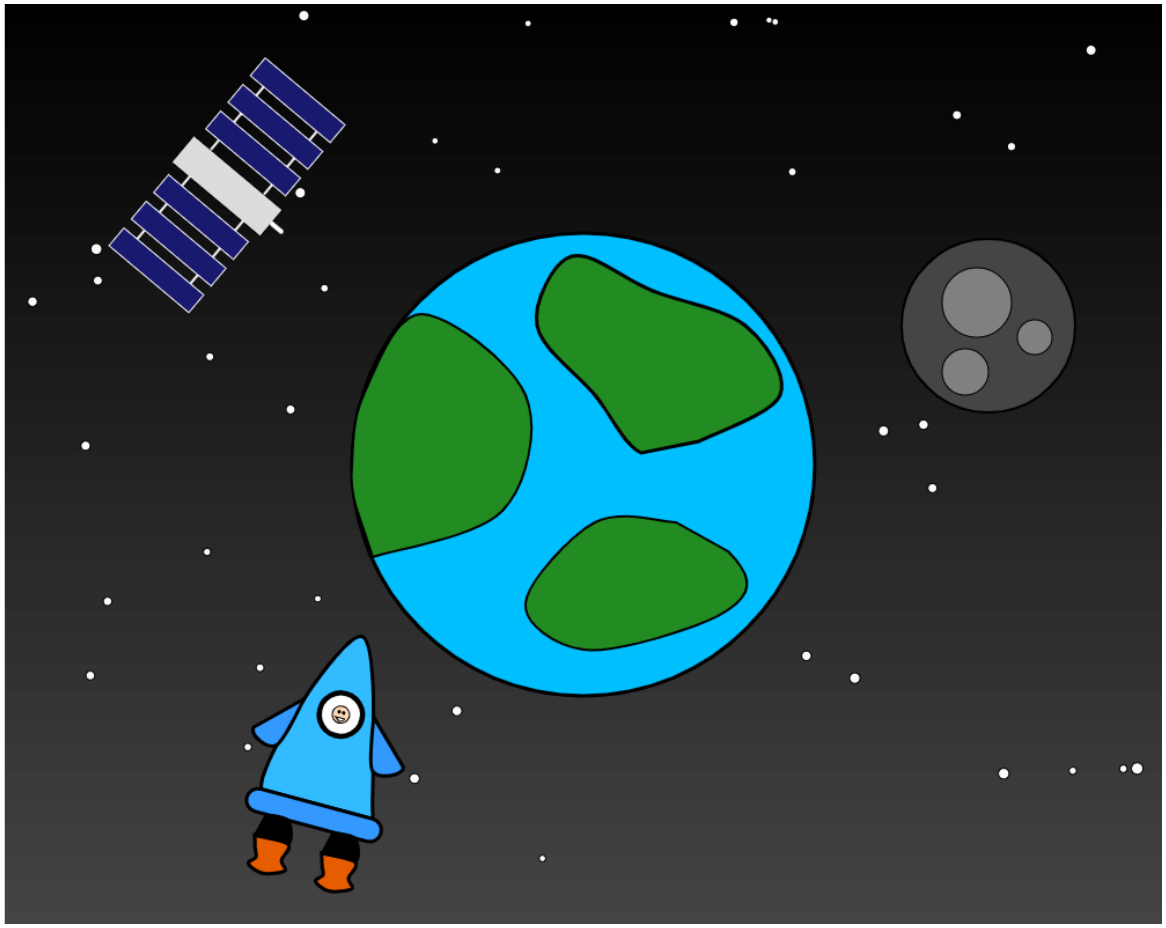


Figure 13. A p5.js sketch - EARTH

This image titled ‘EARTH’ contains a lot of my research findings. It includes the jitter function, however it has no movement, although the random position aspect is kept, and it randomly positions the ellipse() in the background every time the page is refreshed. The satellite in the top left utilises translate() and rotate() as well as multiple strokes and strokeWeights() – the outline colour and outline thickness of each shape. The background is a gradient, and it allows for things to stand out more and adds a very ‘nightly’ atmosphere. The Earth and the spaceship makes use of the beginShape() function. In addition to this, the spaceships also makes use of translate() with the fire.

If this image was to be improved, it would definitely be changes to the irregular shapes. These have room for improvement and could possibly add more detail. The Earth, instead of being an ellipse() could possibly be changed into a sphere(), this could then be changed to an animated image with the Earth rotating. In addition to this, the moon could do with more detail and an overall improvement, this could also be changed to a sphere().

Solar System

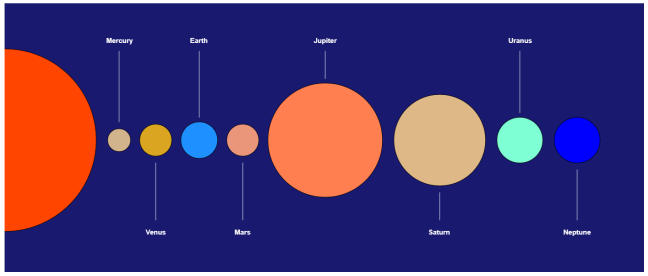


Figure 14. A p5.js sketch – SOLAR SYSTEM

This image titled ‘SOLAR SYSTEM’ takes inspiration from figure 2. This image makes use of the drawshape_’x’() function (x is a name placement here), all the planets were made in their own drawshape function and then placed in the function draw() once finished.

```
//Sun
function drawshape_sun(){ //New function to store all information related to 'sun'
  push(); //Save
  fill("#FF4500"); //fill the colour of the ellipse, colour is OrangeRed
  strokeWeight(1); //Small stroke outline
  stroke(0); //Stroke is black
  ellipse(0,300,400); //ellipse is half on the screen as the middle point is directly on the x axis
  pop(); //Restore
}
```

Figure 15. Code to showcase the function drawshape()

This kept the workflow easy to follow. I also made use of the push() and pop() functions in this.

Overall, this image could be improved in condensing the code. There is most likely a better way of laying out the code in comparison to what I did in the end, especially when it came to labelling each planet. In addition to this, details could have been added to each planet, for example Saturn and Uranus are missing their rings. If this image was to be animated it would most likely be the planets orbiting the sun, once again this image could take advantage of the sphere() shape instead of ellipse(). Furthermore, detail could also be added to the background, maybe the inclusion of the asteroid belt with particles could be added for more depth to the image.

Stars

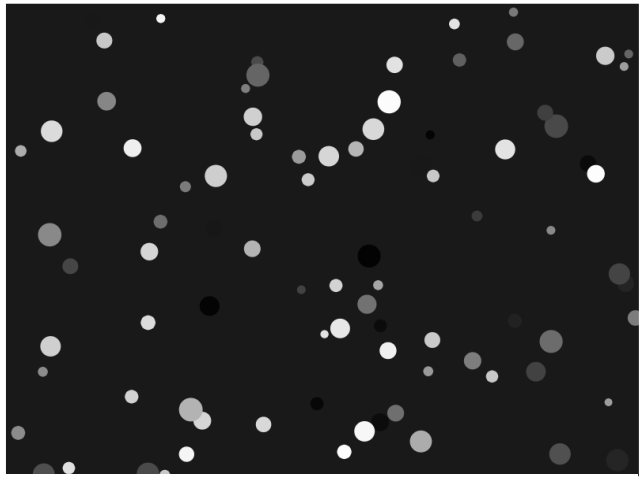


Figure 16. A p5.js sketch - STARS

This image titled 'STARS' makes use of the week 8 workshop as well as the p5.examples particles. At first the image starts with an empty background but once the screen is tapped ellipse() start to spawn in where the mouse is clicked. These ellipse() flicker as they continuously change colour. They never leave the screen as they have been to set to bounce around the screen, this was inspired by the particles page.

If this image was to be improved, it would be a change in shape. Instead of the main shape being an ellipse() it would instead be a star. This is available in p5.examples, however I could not get it to work in my code. In addition to this, a more detailed background could add more to the feel and reinforce the theme of space for this image. Perhaps stationary stars to add a depth of field. When I was working on this code, I wanted to implement a comet that would follow the mouse around, I was unsuccessful in doing so as I couldn't successfully change the shape from a polygon to an ellipse, I increased the sides of the polygon to have it appear like a circle however, once it started moving it was clear that it wasn't.

Conclusion

My research has further developed my understanding of coding in JavaScript and by using it for images it has taught me new possibilities as well as ways of improving my code from here onwards. This could also be said for anyone who views my work. If my work is used as a basis, anyone who followed my research, and my suggested improvements could expand upon my work and take it to the next level.

References

[1] Meckin, D. GitHub *Week 08.* Available from: https://github.com/davemeckin/Intro_to_Creative_Programming/tree/master/Week_08 [Accessed 02 December 2021]

[2] p5.Js *Particles.* Available from: https://p5js.org/examples/simulate-particles.html [Accessed 19 January 2022]

[3] p5.Js *Reference*. Available from: https://p5js.org/reference/#/p5/beginShape [Accessed 20 January 2022]

[4] Shiffman, D. The Coding Train *9.1: Transformations Pt.1 (Translate, Rotate, Push/Pop) – p5.Js Tutorial.* Available from: https://www.youtube.com/watch?v=o9sgjuh-CBM [Accessed 17 January 2022]

[5] Shiffman, D. The Coding Train *9.22: Custom Shapes – p5.Js tutorial.* Available from:

https://www.youtube.com/watch?v=76fiD5DvzeQ&t=706s [Accessed 18 January 2022]

[6] w3schools *HTML Color Names.* Available from: https://www.w3schools.com/colors/colors_names.asp [Accessed 17 January 2022]

[7] w3schools *JavaScript Variables*. Available from: https://www.w3schools.com/js/js_variables.asp [Accessed 21 January 2022]

[8] Workman, K. Kevin Workman *Vertical Gradient – p5.Js Let's Code - #genuary 24.* Available from: https://www.youtube.com/watch?v=DJgDW3F68Xc [Accessed 18 January 2022]