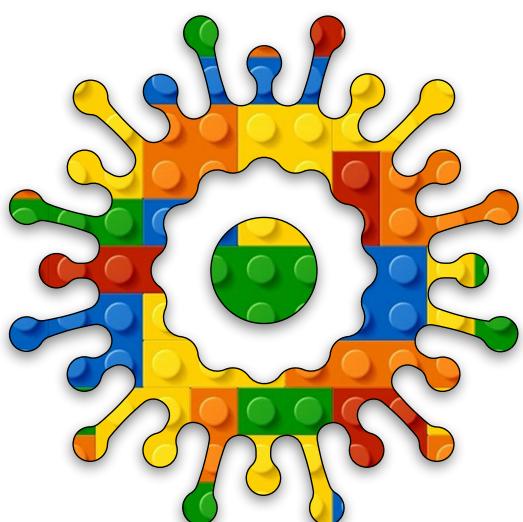


Creative Coding Module A Unit #6

ellipses and triangles





Module A Unit #6 ellipses and triangles

- Sketch A6.1 drawing an ellipse
- Sketch A6.2 mouseX and mouseY
- Sketch A6.3 stretchy time
- Sketch A6.4 triangle
- Sketch A6.5 width and height
- Sketch A6.6 rebuilding the triangle
- Sketch A6.7 a length
- Sketch A6.8 a bit more pointy
- Sketch A6.9 a hundred of them
- Sketch A6.10 push pop stop
- Sketch A6.11 rotate the triangle
- Sketch A6.12 split the triangles up
- Sketch A6.13 random angles
- Sketch A6.14 random length
- Sketch A6.15 random colouring



Introduction to ellipses and triangles

As well as introducing two new shapes, we will also explore how to interact with shapes with the mouse. We will be able to move and stretch these shapes, offering more opportunities to create interesting designs that are more than static images.

Key elements

- 中 ellipse()
- 中 triangle()
- 中 mouseX
- 中 mouseY
- 中 move
- 中 stretch



Sketch A6.1 drawing an ellipse

We start with a new sketch with an ellipse. The `ellipse()` function has four arguments: the x and y coordinates, and the width and height of the ellipse.

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

function draw()
{
    background(220)
    ellipse(200, 200, 300, 100)
}
```

Notes

If you use only three dimensions, you will draw a circle.

Challenge

Try other dimensions for the height and width.

Code Explanation

<code>ellipse(200, 200, 300, 100)</code>	We have an ellipse at the centre of the canvas (200, 200) and a width of 300 and height of 100
--	--

Figure A6.1

The screenshot shows the p5.js IDE interface. At the top, there's a menu bar with 'File', 'Edit', 'Sketch', 'Help', and 'English'. On the right, it says 'Hello, TheHappyCoder!' with a gear icon. The main area has tabs for 'sketch.js' (selected), 'Preview', and 'Console'. The 'Preview' tab shows a gray canvas with a single black ellipse centered at (200, 200) with dimensions 300x100. The 'sketch.js' code is as follows:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background(220)
9     ellipse(200, 200, 300, 100)
10}
```

The 'Console' tab is empty.



Sketch A6.2 mouseX and mouseY

We can use some built-in variables; one set of variables is called **mouseX** and **mouseY**. They return the coordinates of the mouse cursor position on the canvas. We can illustrate it with our ellipse.

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

function draw()
{
    background(220)
    ellipse(mouseX, mouseY, 300, 100)
}
```

Notes

As you move your mouse cursor around the canvas, the ellipse follows it.

Code Explanation

ellipse(mouseX, mouseY, 300, 100)	mouseX returns the x coordinate and mouseY returns the y coordinate of the cursor.
-----------------------------------	--

Figure A6.2

The screenshot shows the p5.js IDE interface. At the top, there's a red header bar with the 'p5.' logo, followed by 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', 'English ▾', and a user greeting 'Hello, TheHappyCoder! ▾'. Below the header is a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a gear for settings. The main workspace is titled 'sketch.js' and shows the following code:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8   background(220)
9   ellipse(mouseX, mouseY, 300, 100)
10 }
```

The preview window on the right shows a gray canvas with a single black ellipse centered at the mouse position. The bottom left corner of the workspace has a 'Console' tab and a 'Clear ▾' button.



Sketch A6.3 stretchy time

If we move `mouseX` and `mouseY` to the dimensions of the ellipse, we can see how we can change the shape of the ellipse through moving our cursor.

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

function draw()
{
    background(220)
    ellipse(200, 200, mouseX, mouseY)
}
```

Notes

You can now stretch the ellipse by moving your mouse.

Challenge

Try it with the rectangle.

Code Explanation

<code>ellipse(200, 200, mouseX, mouseY)</code>	The <code>mouseX</code> and <code>mouseY</code> are the width and height respectfully.
--	--

Figure A6.3

The screenshot shows the p5.js code editor interface. At the top, there are navigation buttons (File, Edit, Sketch, Help, English) and a user profile "Hello, TheHappyCoder!". Below the header is a toolbar with icons for play/pause, stop, auto-refresh (checked), and a gear for settings. The main area has tabs for "sketch.js" and "Preview". The code editor contains the following JavaScript code:

```
1 function setup()
2 {
3   createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8   background(220)
9   ellipse(200, 200, mouseX, mouseY)
10 }
```

The preview window shows a gray canvas with a small dark gray dot at the center (200, 200) and a larger white ellipse centered at the same coordinates.



Sketch A6.4 triangle

Instead of an ellipse, let's make a triangle. A triangle has coordinates for all three corners, so the function `triangle()` has six arguments. There are a lot of numbers to get your head around; often, I sketch it out on a piece of paper first so I get the right coordinates for the right corner.

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

function draw()
{
    background(220)
    triangle(100, 100, 100, 300, 300, 300)
}
```

Notes

We have three sets of coordinates: (100, 100), (100, 300) and (300, 300).

Challenge

Draw more triangles, make a simple pattern.

Code Explanation

<code>triangle(100, 100, 100, 300, 300, 300)</code>	A triangle with three sets of coordinates for each corner (vertex)
---	---

Figure A6.4

The screenshot shows the p5.js IDE interface. At the top, there's a toolbar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right of the toolbar is a user profile 'Hello, TheHappyCoder! ▾' and a gear icon. Below the toolbar, the file name 'sketch.js' is displayed along with a note 'Saved: just now'. On the left, the code editor contains the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background(220)
9     triangle(100, 100, 100, 300, 300, 300)
10 }
```

The preview window on the right shows a light gray square canvas with a single black triangle drawn on it, starting from the bottom-left corner (100, 100) and extending to the top-right corner (300, 300).



Sketch A6.5 width and height

We can make use of another built-in variable, `width` and `height`. These variables return the width and height of the canvas. So in our case, the value of `width` is **400** and the value of `height` is also **400**. If we change the dimensions of the canvas, these values also change. If we divide the `width` and `height` by 2, we always get the centre of the canvas, whatever its dimensions.

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

function draw()
{
    background(220)
    translate(width/2, height/2)
    triangle(100, 100, 100, 300, 300, 300)
}
```

Notes

We have translated the origin of the canvas to its centre rather than the top left-hand corner. This has pushed the triangle down and to the left by **200 pixels**.

Challenge

Change the dimensions of the canvas.

Code Explanation

<code>translate(width/2, height/2)</code>	The <code>width/2</code> is 200 and the <code>height/2</code> is also 200
---	---

Figure A6.5

The screenshot shows the p5.js IDE interface. At the top, there's a menu bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right of the menu is a greeting 'Hello, TheHappyCoder! ▾' and a gear icon for settings. Below the menu is a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and a preview button. The main workspace is titled 'sketch.js' and contains the following code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background(220)
9     translate(width/2, height/2)
10    triangle(100, 100, 100, 300, 300, 300)
11 }
```

The preview window on the right shows a light gray canvas with a white triangle centered at the bottom-right corner, rotated 45 degrees.



Sketch A6.6 rebuilding the triangle

This is where a physical drawing often helps. We can redraw the triangle with new coordinates. Remember that the origin is now in the centre of the canvas, and we measure the **x** and **y** coordinates from the centre; hence, **-100** is to the left of the centre for **x** and upwards for the **y** coordinates, and vice versa for **100**.

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

function draw()
{
    background(220)
    translate(width/2, height/2)
    triangle(-100, -100, -100, 100, 100, 100)
}
```

Notes

We have the same triangle as previously.

Challenge

Change the coordinates to draw different types of triangles.

Figure A6.6

The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right of the toolbar, it says "Hello, TheHappyCoder! ▾". Below the toolbar, the code editor has tabs for "sketch.js" and "Preview". The "sketch.js" tab contains the following JavaScript code:

```
1 function setup()
2 {
3     createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8     background(220)
9     translate(width/2, height/2)
10    triangle(-100, -100, -100, 100, 100, 100)
11 }
```

The "Preview" tab shows a gray canvas with a single black triangle drawn on it. The triangle is oriented with its base at the bottom-left and its apex at the top-right, centered at the origin (width/2, height/2).



Sketch A6.7 a length

Let's give the dimension a variable called **len** (for length).

```
let len = 100

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

function draw()
{
    background(220)
    translate(width/2, height/2)
    triangle(-len, -len, -len, len, len, len)
}
```

Notes

This can be handy if we change the dimension and therefore only need to change it once.

Figure A6.7

The screenshot shows the p5.js code editor interface. At the top, there's a red header bar with the 'p5' logo. Below it is a toolbar with icons for play, stop, auto-refresh (which is checked), and user information ('Hello, TheHappyCoder!'). The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following JavaScript code:

```
1 let len = 100
2
3 function setup()
4 {
5   createCanvas(400, 400)
6 }
7
8 function draw()
9 {
10  background(220)
11  translate(width/2, height/2)
12  triangle(-len, -len, -len, len, len, len)
13 }
```

The 'Preview' window shows a light gray canvas with a single black right-angled triangle drawn in the center. The triangle has its hypotenuse pointing towards the bottom-right corner of the canvas. The vertices are located at (-len, -len), (-len, len), and (len, len). The sketch was saved 'just now'.



Sketch A6.8 a bit more pointy

Make it a bit smaller and more pointy so that it isn't an equilateral triangle.

```
let len = 50

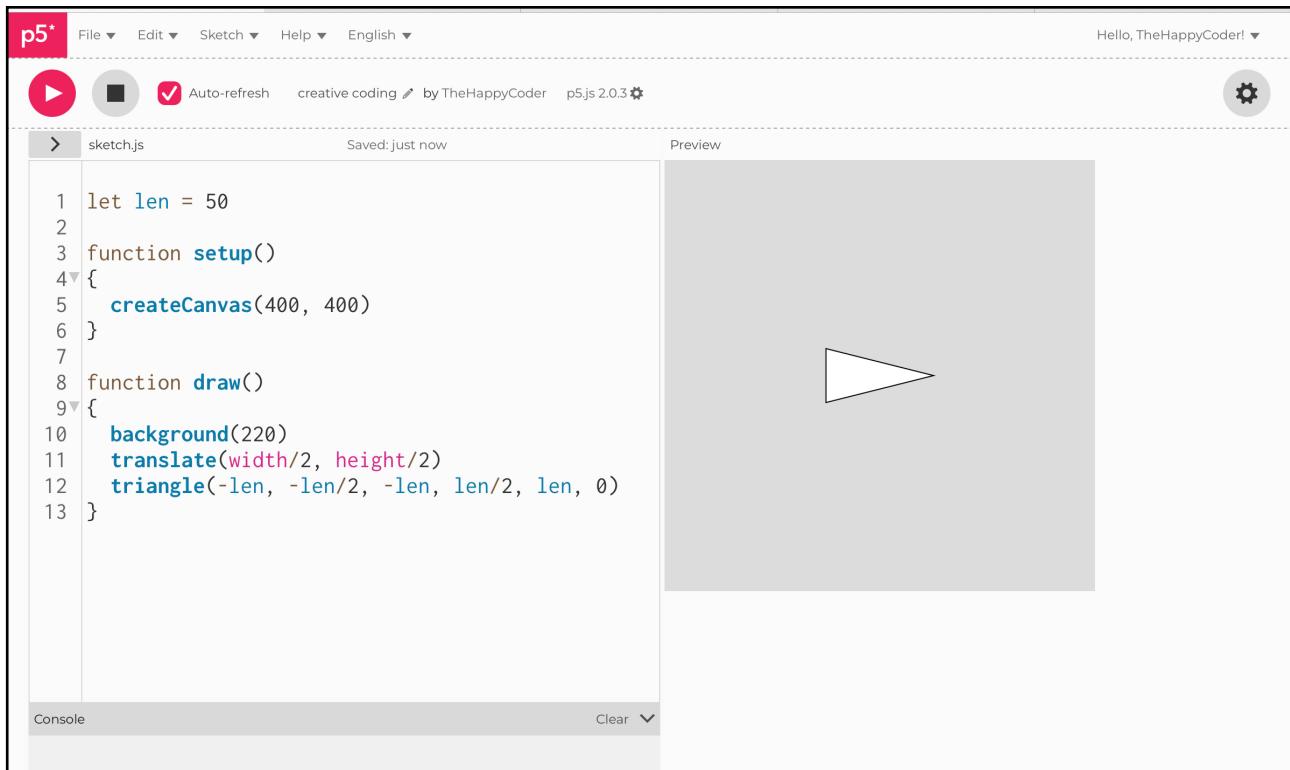
function setup()
{
    createCanvas(400, 400)
}

function draw()
{
    background(220)
    translate(width/2, height/2)
    triangle(-len, -len/2, -len, len/2, len, 0)
}
```

Notes

We can see what direction it is pointing in.

Figure A6.8



The screenshot shows the p5.js code editor interface. At the top, there's a red header bar with the 'p5' logo. Below it is a toolbar with icons for play, stop, auto-refresh (which is checked), and help. The main area has tabs for 'sketch.js' and 'Preview'. The code in 'sketch.js' is:

```
let len = 50
function setup()
{
  createCanvas(400, 400)
}
function draw()
{
  background(220)
  translate(width/2, height/2)
  triangle(-len, -len/2, -len, len/2, len, 0)
}
```

The 'Preview' window shows a white triangle centered on a gray background. The triangle is oriented with its apex pointing upwards. In the bottom left corner of the editor, there's a 'Console' tab and a 'Clear' dropdown menu.



Sketch A6.9 a hundred of them

Now we create a `for()` loop to draw **100** triangles. We put those two lines of code inside the `for()` loop.

```
let len = 50

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

function draw()
{
    background(220)

    for (let i = 0; i < 100; i++)
    {
        translate(width/2, height/2)
        triangle(-len, -len/2, -len, len/2, len, 0)
    }
}
```

Notes

The problem is that the `translate()` function is accumulative, which means it shifts the origin by that amount each iteration in the `for()` loop. We need to use `push()` and `pop()`, and also we need to stop repeating the `for()` in `draw()` with `noLoop()`.

Code Explanation

<code>for (let i = 0; i < 100; i++)</code>	A <code>for()</code> loop where the variable <code>i</code> which is defined and initialised to 0, then incremented by 1 until it reaches 100
---	---

Figure A6.9

The screenshot shows the p5.js web-based code editor interface. At the top, there's a navigation bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right of the bar, it says 'Hello, TheHappyCoder! ▾' and has a gear icon. Below the bar, there are icons for play, stop, and refresh, followed by the text 'Auto-refresh' with a checked checkbox, 'creative coding' with a link icon, 'by TheHappyCoder', and 'p5.js 2.0.3' with a link icon.

The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. The 'sketch.js' section contains the following code:

```
1 let len = 50
2
3 function setup()
4 {
5   createCanvas(400, 400)
6 }
7
8 function draw()
9 {
10  background(220)
11  for (let i = 0; i < 100; i++)
12  {
13    translate(width/2, height/2)
14    triangle(-len, -len/2, -len, len/2, len, 0)
15  }
16 }
```

The 'Preview' section shows a light gray canvas. In the center, there is a white equilateral triangle pointing downwards. In the bottom right corner of the canvas, there is a small white L-shaped notch or cutout.



Sketch A6.10 push pop stop

We add `push()` and `pop()` to stop the `translate()` from adding itself 100 times, and also the `noLoop()` to stop the `draw()` function once it has drawn the 100 triangles.

```
let len = 50

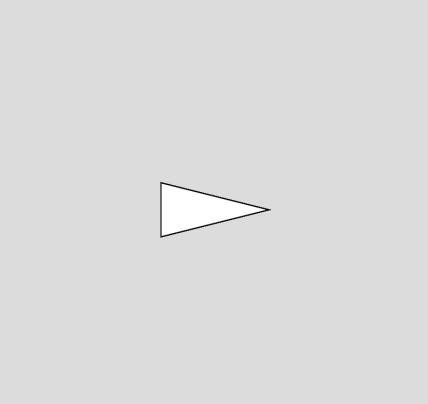
function setup()
{
    createCanvas(400, 400)
}

function draw()
{
    background(220)
    for (let i = 0; i < 100; i++)
    {
        push()
        translate(width/2, height/2)
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

We have **100** triangles, but they are all in one place. We could also put all of this code in the `setup()` function and therefore no need to use `noLoop()`.

Figure A6.10



The screenshot shows the p5.js IDE interface. The top bar includes the p5 logo, file navigation, and a user profile. The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following sketch.js code:

```
let len = 50
function setup()
{
  createCanvas(400, 400)
}
function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(width/2, height/2)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

The preview window shows a white equilateral triangle centered on a gray background.



Sketch A6.11 rotate the triangle

We are rotating all the triangles by 45° degrees.

```
let len = 50
let angle = 45

function setup()
{
    createCanvas(400, 400)
    angleMode(DEGREES)
}

function draw()
{
    background(220)
    for (let i = 0; i < 100; i++)
    {
        push()
        translate(width/2, height/2)
        rotate(angle)
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

We have 100 triangles, all rotated by 45°.

Challenge

Try other angles.

Figure A6.11

The screenshot shows the p5.js web editor interface. At the top, there are navigation menus: File, Edit, Sketch, Help, and English. On the right, it says "Hello, TheHappyCoder! ▾". Below the menu bar, there are icons for play, stop, and refresh, followed by "Auto-refresh" with a checkmark, "creative coding" by TheHappyCoder, and "p5.js 2.0.3". To the right of these is a gear icon.

The main area has tabs for "sketch.js" and "Preview". The "sketch.js" tab shows the following code:

```
let angle = 45
function setup()
{
  createCanvas(400, 400)
  angleMode(DEGREES)
}
function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(width/2, height/2)
    rotate(angle)
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

The "Preview" tab shows a gray canvas with a single white triangle centered at the bottom-left. The triangle has vertices at (-len, -len/2), (-len, len/2), and (len, 0). The angle between the two sides is 45 degrees.



Sketch A6.12 split the triangles up

But we want to see all the triangles; to do that, we can translate them randomly across the whole canvas (**width** and **height**).

```
let len = 50
let angle = 45

function setup()
{
    createCanvas(400, 400)
    angleMode(DEGREES)
}

function draw()
{
    background(220)
    for (let i = 0; i < 100; i++)
    {
        push()
        translate(random(width), random(height))
        rotate(angle)
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

Now each triangle drawn has its own random position.



Challenges

1. Try pushing them all into a square in the middle.
2. Make the triangles smaller or random in size.
3. Can you work out how to rotate each one separately?

Figure A6.12

The screenshot shows the p5.js code editor interface. At the top, there's a menu bar with 'File ▾', 'Edit ▾', 'Sketch ▾', 'Help ▾', and 'English ▾'. To the right, it says 'Hello, TheHappyCoder! ▾' and has a gear icon. Below the menu, there are icons for play, stop, and refresh, followed by the text 'Auto-refresh creative coding by TheHappyCoder p5.js 2.0.3'.

The main area is divided into two sections: 'sketch.js' on the left and 'Preview' on the right. In 'sketch.js', the code is as follows:

```
1 let angle = 45
2
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   angleMode(DEGREES)
8 }
9
10 function draw()
11 {
12   background(220)
13   for (let i = 0; i < 100; i++)
14   {
15     push()
16     translate(random(width), random(height))
17     rotate(angle)
18     triangle(-len, -len/2, -len, len/2, len, 0)
19     pop()
20   }
21   noLoop()
22 }
```

In the 'Preview' section, the output of the code is shown: a collection of white triangles on a light gray background. The triangles are rotated at an angle of 45 degrees and have varying widths and heights, creating a dynamic, overlapping pattern.



Sketch A6.13 random angles

This is how you can randomly set the angles. Change the angle to a full **360°** and rotate randomly within that angle.

```
let len = 50
let angle = 360

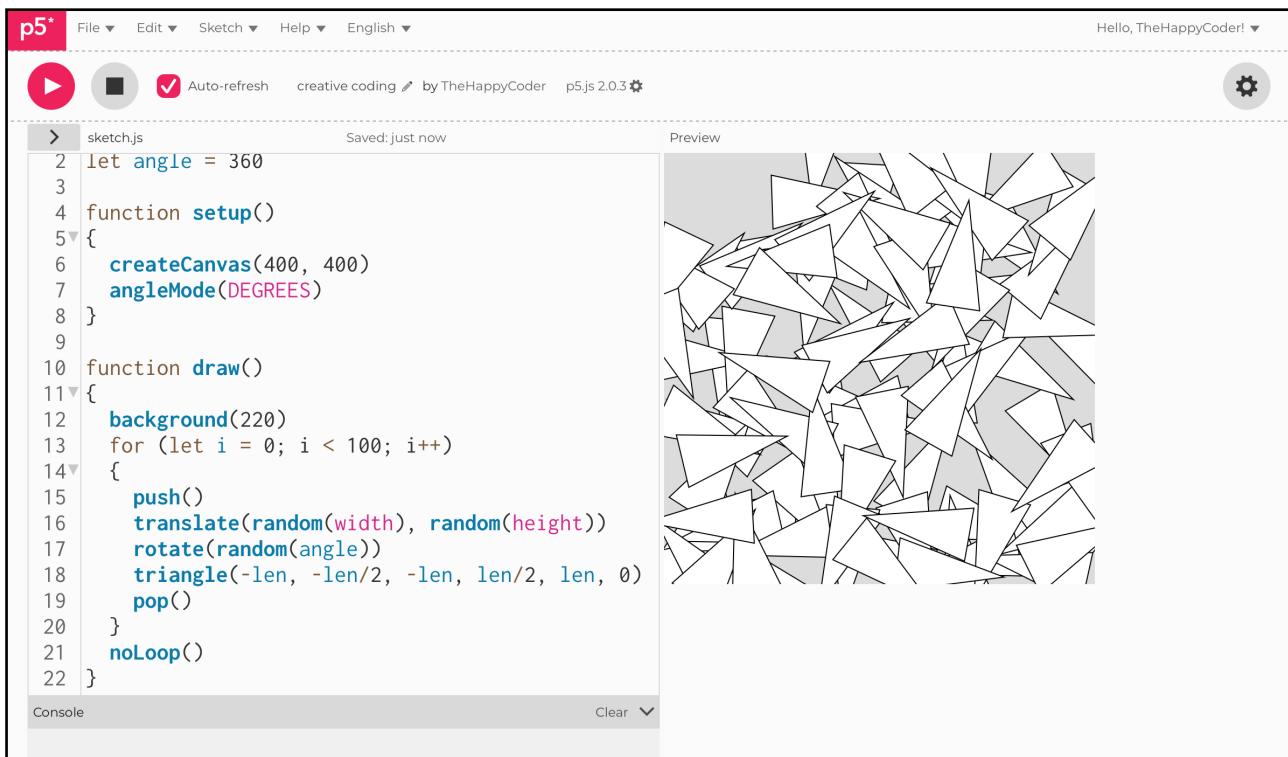
function setup()
{
    createCanvas(400, 400)
    angleMode(DEGREES)
}

function draw()
{
    background(220)
    for (let i = 0; i < 100; i++)
    {
        push()
        translate(random(width), random(height))
        rotate(random(angle))
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

Rotating randomly from **0°** to **360°** degrees

Figure A6.13



The screenshot shows the p5.js code editor interface. At the top, there are navigation menus: File, Edit, Sketch, Help, and English. On the right side, it says "Hello, TheHappyCoder! ▾". Below the menu bar, there are icons for play, stop, and refresh, followed by the text "Auto-refresh" with a checked checkbox. To the right of that is "creative coding" with a small person icon, "by TheHappyCoder", and "p5.js 2.0.3". On the far right is a gear icon.

The main area has a title "sketch.js" and a status message "Saved: just now". To the right is a "Preview" window showing a complex geometric pattern of many triangles of varying sizes and orientations, rendered in shades of gray on a white background.

The code editor itself contains the following JavaScript code:

```
sketch.js
let angle = 360
function setup()
{
  createCanvas(400, 400)
  angleMode(DEGREES)
}
function draw()
{
  background(220)
  for (let i = 0; i < 100; i++)
  {
    push()
    translate(random(width), random(height))
    rotate(random(angle))
    triangle(-len, -len/2, -len, len/2, len, 0)
    pop()
  }
  noLoop()
}
```

At the bottom left is a "Console" button, and at the bottom right is a "Clear" dropdown menu.



Sketch A6.14 random length

Now for the size of the triangle. This will keep the proportions the same. We will have triangles between 5 and 50 long.

```
let len = 50
let angle = 360

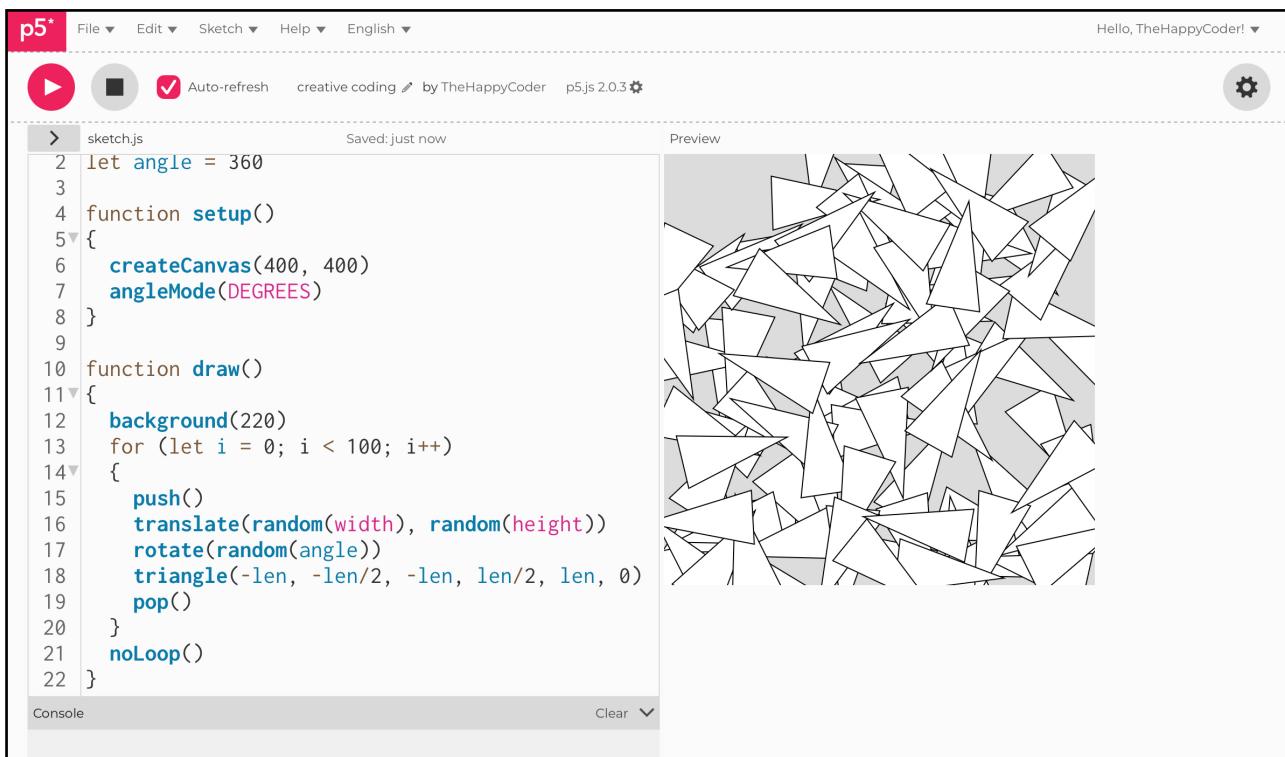
function setup()
{
    createCanvas(400, 400)
    angleMode(DEGREES)
}

function draw()
{
    background(220)
    for (let i = 0; i < 100; i++)
    {
        push()
        translate(random(width), random(height))
        rotate(random(angle))
        len = random(5, 50)
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

There is a lot to play with here.

Figure A6.14



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by menu items: File, Edit, Sketch, Help, and English. To the right of the toolbar, it says "Hello, TheHappyCoder! ▾". Below the toolbar, the code editor has a file list on the left showing "sketch.js" and a preview window on the right displaying a complex geometric pattern of many triangles.

```
> sketch.js
1 let angle = 360
2
3
4 function setup()
5 {
6   createCanvas(400, 400)
7   angleMode(DEGREES)
8 }
9
10 function draw()
11 {
12   background(220)
13   for (let i = 0; i < 100; i++)
14   {
15     push()
16     translate(random(width), random(height))
17     rotate(random(angle))
18     triangle(-len, -len/2, -len, len/2, len, 0)
19     pop()
20   }
21   noLoop()
22 }
```

Below the code editor, there are "Console" and "Clear" buttons.



Sketch A6.15 random colouring

Make the background white and give each triangle a random colour (and alpha).

```
let len = 50
let angle = 360

function setup()
{
    createCanvas(400, 400)
    angleMode(DEGREES)
}

function draw()
{
    background(255)

    for (let i = 0; i < 100; i++)
    {
        push()
        translate(random(width), random(height))
        rotate(random(angle))
        len = random(5, 50)
        fill(random(255), random(255), random(255), random(255))
        triangle(-len, -len/2, -len, len/2, len, 0)
        pop()
    }
    noLoop()
}
```

Notes

Starting to look interesting.

 Challenges

1. What else could you develop alongside these ideas?
2. Different shapes, colours, etc.

Figure A6.15

The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, and refresh, followed by the text "Hello, TheHappyCoder! ▾". Below the toolbar, the file name "sketch.js" is displayed along with the message "Saved: just now". The code editor area contains the following JavaScript code:

```
5  createCanvas(400, 400)
6  angleMode(DEGREES)
7
8
9
10 function draw()
11 {
12   background(255)
13   for (let i = 0; i < 100; i++)
14   {
15     push()
16     translate(random(width), random(height))
17     rotate(random(angle))
18     len = random(5, 50)
19     fill(random(255), random(255), random(255),
random(255))
20     triangle(-len, -len/2, -len, len/2, len, 0)
21     pop()
22   }
23   noLoop()
24 }
```

The preview window on the right shows a dynamic geometric pattern of numerous triangles of various colors (purple, green, blue, yellow, pink) and sizes, scattered across a white background.



Next . . .

In the next unit (#7), we will go very small and look at pixels; they are the dots on the canvas that make up your screen.