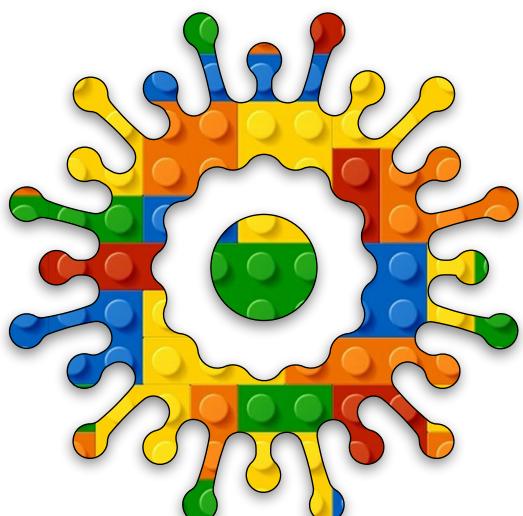


Creative Coding

Module C

Unit #5

graphics texture





Module C Unit #5 graphics texture

- Sketch C5.1 starting sketch
- Sketch C5.2 applying texture
- Sketch C5.3 separate canvas
- Sketch C5.4 adding graphics
- Sketch C5.5 drawing on the sides
- Sketch C5.6 a bit of fun tweaking
- Sketch C5.7 creating text
- Sketch C5.8 words on a plane
- Sketch C5.9 text on a cylinder



Introduction to graphics texture

This is a powerful function that is also extremely fun (in my opinion). It gives you the opportunity to put images, photos, videos and even draw on the faces of a shape while it is moving. For this unit, we are just keeping it simple to add shapes, text and colour for now.



Sketch C5.1 starting sketch

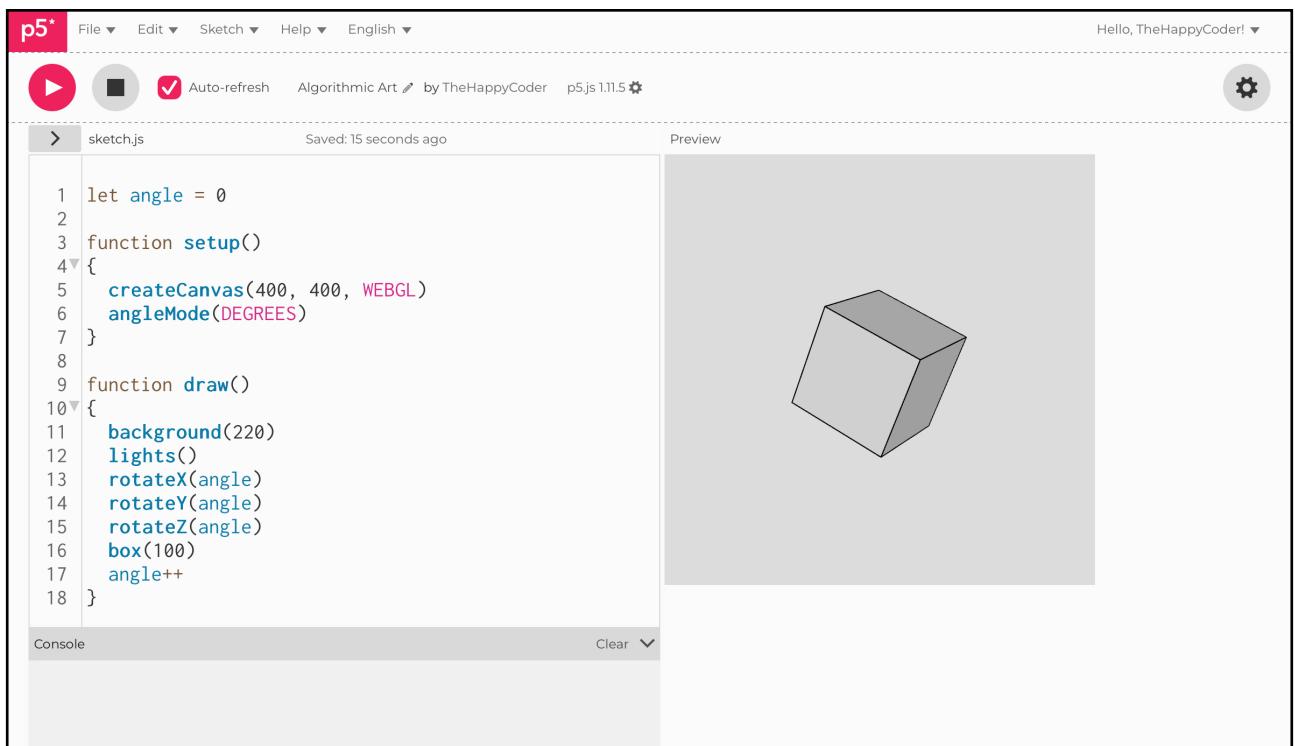
Our starting sketch gives us a nice rotating cube.

```
let angle = 0

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

function draw()
{
    background(220)
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    box(100)
    angle++
}
```

Figure C5.1



A screenshot of the p5.js IDE interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), and user information (Hello, TheHappyCoder!). The main area shows a sketch titled "sketch.js" saved 15 seconds ago. The code uses p5.js functions like createCanvas, angleMode, background, lights, rotateX, rotateY, rotateZ, and box to create a 3D cube. The preview window on the right displays a perspective view of the cube.

```
let angle = 0
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
}
function draw()
{
  background(220)
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  box(100)
  angle++
}
```



Sketch C5.2 applying texture

We create a square object **400** by **400** and apply it as a texture to the box. The **texture()** function wraps an image onto a regular, primitive shape. Effectively, we have created and added another canvas onto the sides of the cube.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
}

function draw()
{
    background(220)
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```



Notes

We haven't done anything with the texture as such, so it is a blank canvas. It is difficult to try to explain in words what is happening; that is why it is better to do it and see what it can produce. It makes it more intuitive than academic.



Code Explanation

let graphics	Variable to hold the graphics object
graphics = createGraphics(400, 400)	Creating the graphics object and giving it a size of 400, 400
texture(graphics)	Putting the texture onto the cube with the graphics object

Figure C5.2

The screenshot shows the p5.js code editor interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), a user profile (Hello, TheHappyCoder!), and a preview button. The code editor window has a left panel for code and a right panel for preview. The code in the left panel is:

```
sketch.js
let graphics
function setup()
{
  createCanvas(400, 400, WEBGL)
  angleMode(DEGREES)
  graphics = createGraphics(400, 400)
}
function draw()
{
  background(220)
  lights()
  rotateX(angle)
  rotateY(angle)
  rotateZ(angle)
  texture(graphics)
  box(100)
  angle++
}
```

The preview window on the right shows a 3D perspective drawing of a cube. The cube is oriented diagonally, with its front face rotated upwards and to the right. The edges of the cube are thin black lines.



Sketch C5.3 separate canvas

Now we can manipulate the texture as a separate canvas. Starting simple, we give it a yellow background.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
}

function draw()
{
    background(220)
    graphics.background(200, 200, 0)
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```

Notes

We now have a yellow cube; we haven't filled it with a colour but treated each side as a canvas and given it a background colour. We call the new **background()** function onto the **graphics** object.

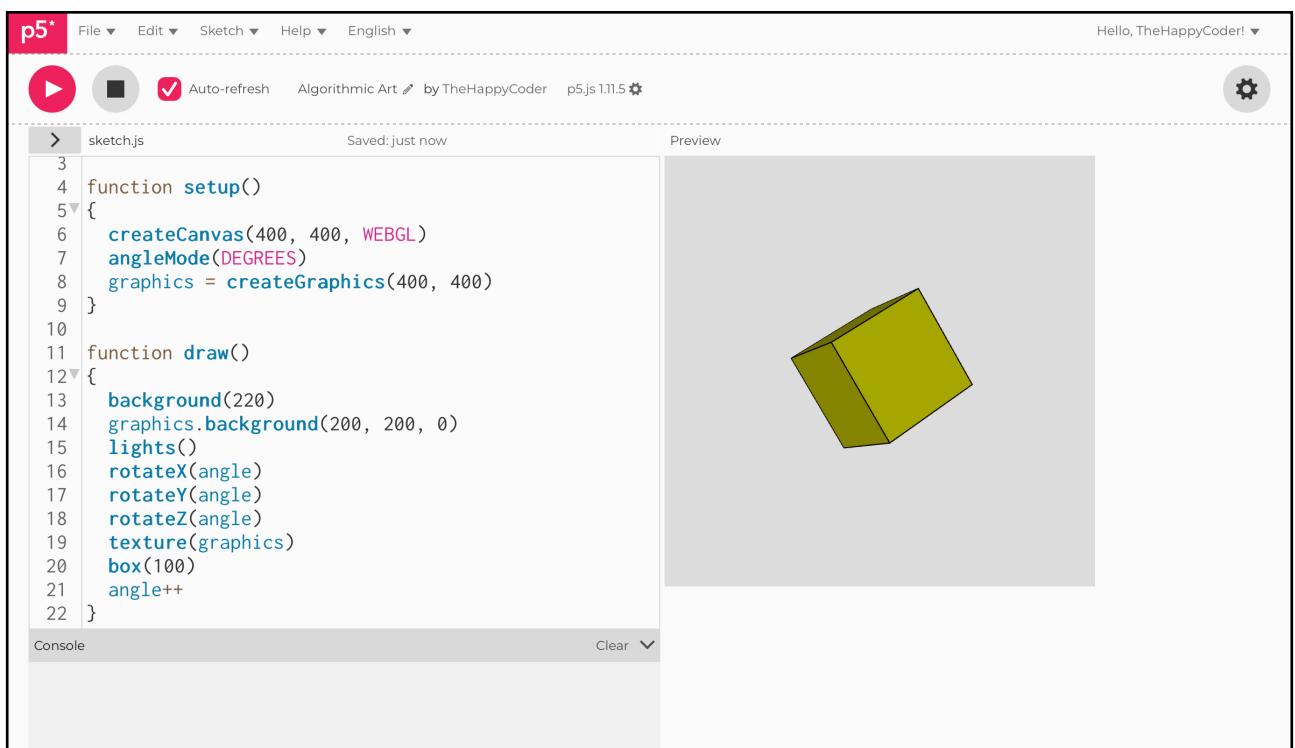
Challenges

1. Change the colour.
2. Try `graphics.fill()` to see if that does anything.

Code Explanation

<code>graphics.background(200, 200, 0)</code>	Gives the graphics canvas a background
-----------------------------------------------	----------------------------------------

Figure C5.3



The screenshot shows the p5.js IDE interface. The top bar includes the p5 logo, file navigation, and a user profile. The code editor on the left contains the following JavaScript code:

```
sketch.js
3
4 function setup()
5{
6  createCanvas(400, 400, WEBGL)
7  angleMode(DEGREES)
8  graphics = createGraphics(400, 400)
9 }
10
11 function draw()
12{
13  background(220)
14  graphics.background(200, 200, 0)
15  lights()
16  rotateX(angle)
17  rotateY(angle)
18  rotateZ(angle)
19  texture(graphics)
20  box(100)
21  angle++
22 }
```

The preview window on the right displays a 3D perspective view of a yellow rectangular box rotating in space.



Sketch C5.4 adding graphics

We can draw shapes on the side of the cube; in this case, we can draw a circle which appears on all the surfaces of the box.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
}

function draw()
{
    background(220)
    graphics.background(200, 200, 0)
    graphics.circle(100, 100, 150)
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```

Notes

Notice that the circle really does appear on each side of the cube; this shows that the texture is applied separately to each side of the cube. The size of the canvas (and circle) is scaled to fit.



Challenges

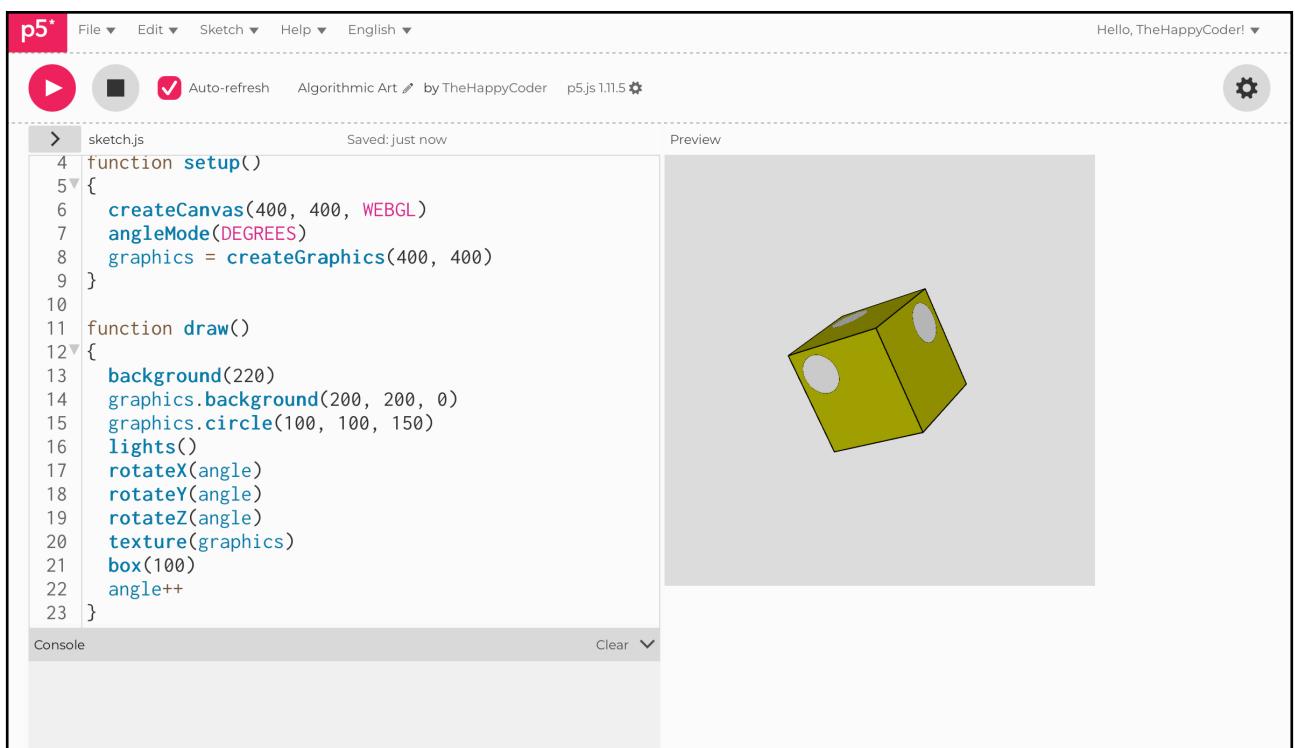
1. Change the size of the graphics to (100, 400).
2. Try other 2D shapes.



Code Explanation

graphics.circle(100, 100, 150)	Creating a circle on the graphics canvas, 100 in, 100 down and radius 150
--------------------------------	------------------------------------------------------------------------------

Figure C5.4



The screenshot shows the p5.js code editor interface. At the top, there's a toolbar with icons for play, stop, auto-refresh (which is checked), and settings. The title bar says "sketch.js" and "Saved: just now". The right side of the interface is a "Preview" window showing a 3D perspective of a yellow cube. The cube has two circular holes on its faces. The left side of the interface is the code editor with the following sketch.js code:

```
4 function setup()
5 {
6   createCanvas(400, 400, WEBGL)
7   angleMode(DEGREES)
8   graphics = createGraphics(400, 400)
9 }
10
11 function draw()
12 {
13   background(220)
14   graphics.background(200, 200, 0)
15   graphics.circle(100, 100, 150)
16   lights()
17   rotateX(angle)
18   rotateY(angle)
19   rotateZ(angle)
20   texture(graphics)
21   box(100)
22   angle++
23 }
```

Below the code editor is a "Console" section which is currently empty.



Sketch C5.5 drawing on the sides

We can fill the circle with a colour (red in this case), and we can also have the circle move across the sides of the box as you move your mouse across the main canvas.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
}

function draw()
{
    background(220)
    graphics.background(200, 200, 0)
    graphics.fill(255, 0, 0)
    graphics.circle(mouseX, mouseY, 150)
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```



Notes

The circle follows the mouse across the main canvas.



Challenge

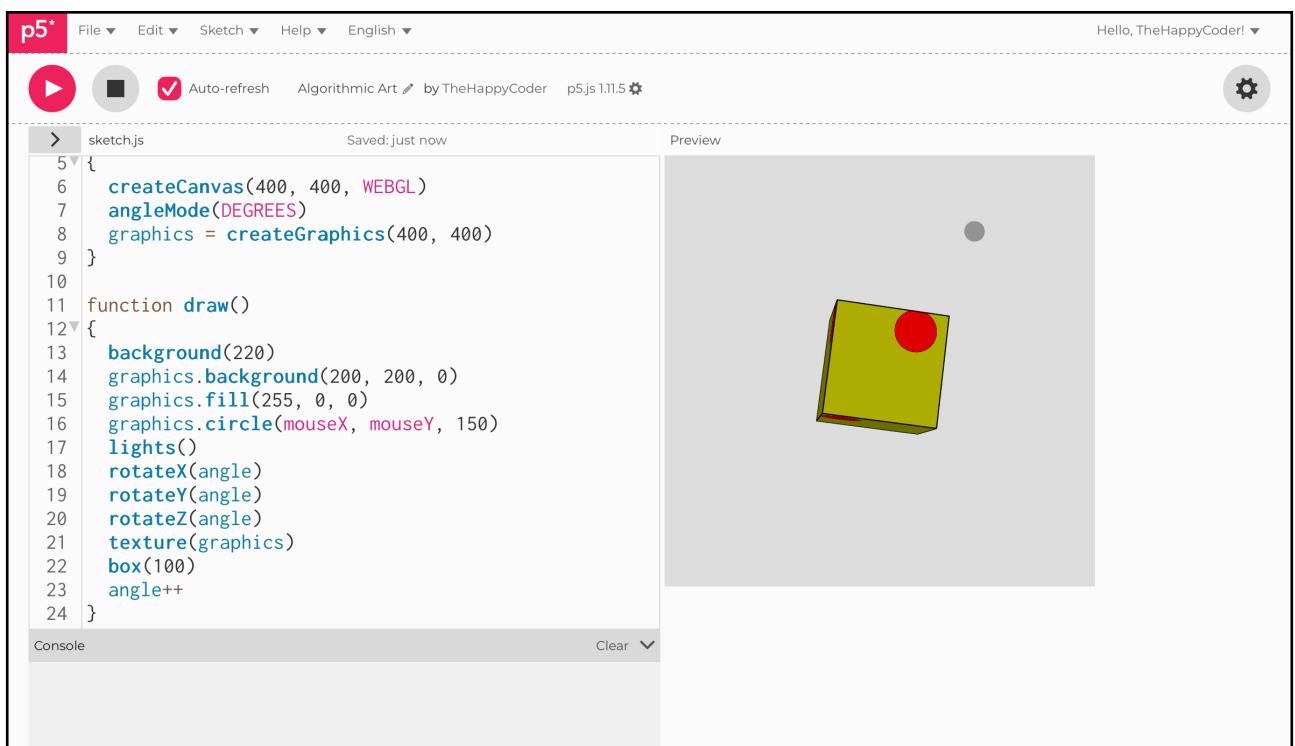
Could you get the circle to bounce off the edges?



Code Explanation

<code>graphics.fill(255, 0, 0)</code>	Fills the circle with red colour
<code>graphics.circle(mouseX, mouseY, 150)</code>	Gives the x and y coordinates of the mouse

Figure C5.5



The screenshot shows the p5.js IDE interface. The code editor on the left contains the following JavaScript code:

```
sketch.js
Saved: just now
Preview

5  {
6    createCanvas(400, 400, WEBGL)
7    angleMode(DEGREES)
8    graphics = createGraphics(400, 400)
9  }
10
11 function draw()
12 {
13   background(220)
14   graphics.background(200, 200, 0)
15   graphics.fill(255, 0, 0)
16   graphics.circle(mouseX, mouseY, 150)
17   lights()
18   rotateX(angle)
19   rotateY(angle)
20   rotateZ(angle)
21   texture(graphics)
22   box(100)
23   angle++
24 }
```

The preview window on the right displays a 3D perspective view of a yellow cube. A single red circle is positioned on the top face of the cube. The rest of the cube's faces are rendered in a darker shade of yellow.



Sketch C5.6 a bit of fun tweaking

Watch what happens when we comment out the graphics background and add in `noStroke()`.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
}

function draw()
{
    background(220)
    // graphics.background(200, 200, 0)
    graphics.fill(255, 0, 0)
    graphics.circle(mouseX, mouseY, 150)
    noStroke()
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```



Notes

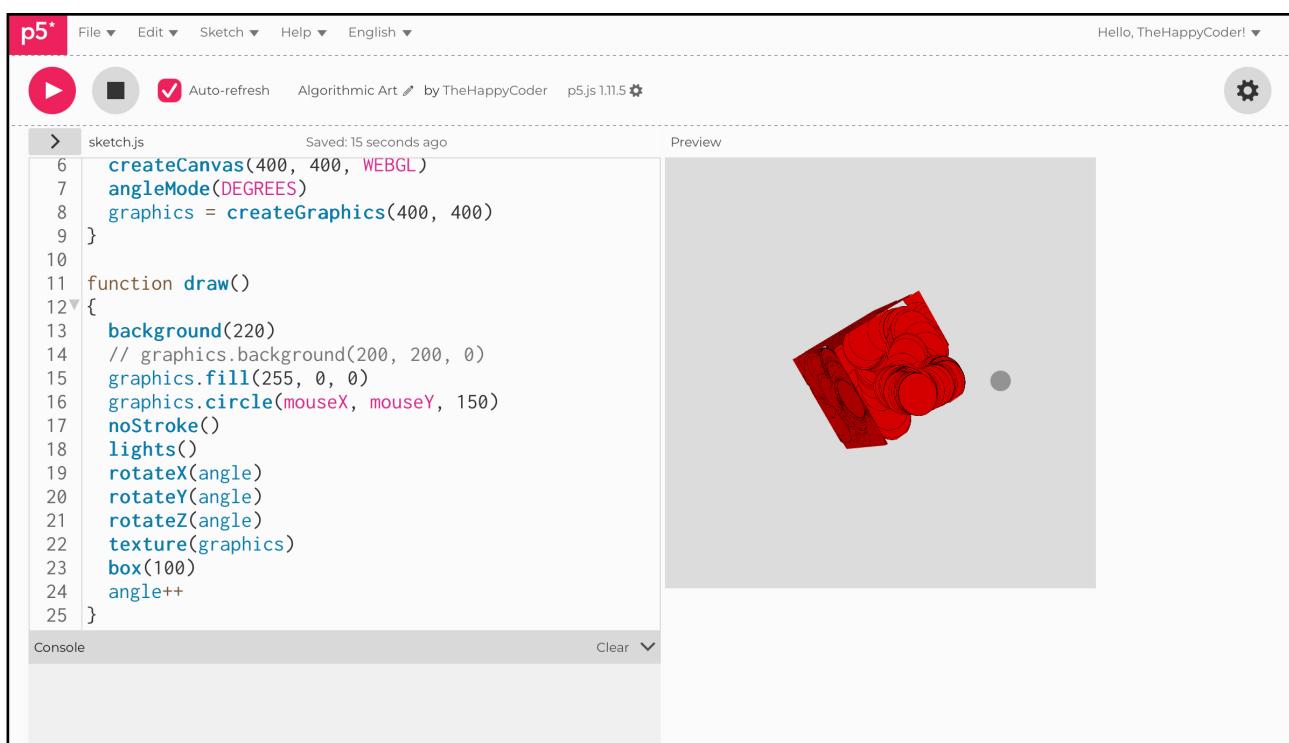
We have an invisible cube to paint on. The `noStroke()` only works on the cube, not the circle.



Challenge

How would you use `noStroke()` on the circle?

Figure C5.6



The screenshot shows the p5.js web editor interface. The top bar includes the p5 logo, file navigation (File, Edit, Sketch, Help, English), a user greeting (Hello, TheHappyCoder!), and a settings gear icon. The main area has tabs for 'sketch.js' and 'Preview'. The code editor contains the following JavaScript code:

```
sketch.js
6 createCanvas(400, 400, WEBGL)
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 }
10
11 function draw()
12 {
13   background(220)
14   // graphics.background(200, 200, 0)
15   graphics.fill(255, 0, 0)
16   graphics.circle(mouseX, mouseY, 150)
17   noStroke()
18   lights()
19   rotateX(angle)
20   rotateY(angle)
21   rotateZ(angle)
22   texture(graphics)
23   box(100)
24   angle++
25 }
```

The preview window shows a 3D perspective view of a red cube. The cube is composed of numerous small red circles, creating a textured appearance. A single small gray circle is visible in the upper right corner of the preview area.



Sketch C5.7 creating text

We can also create text on the cube.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
    graphics.textAlign(CENTER, CENTER)

}

function draw()
{
    background(220)
    // graphics.background(200, 200, 0)
    graphics textSize(75)
    graphics.text('HAPPY', 200, 200)

    noStroke()
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    box(100)
    angle++
}
```



Notes

Notice that we had to prefix the `textAlign()` with the `graphics` reference.



Challenge

Could you rotate the text at the same time?

Figure C5.7

The screenshot shows the p5.js IDE 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, it says "Hello, TheHappyCoder!" with a dropdown arrow. The main area has tabs for "sketch.js" and "Preview". The code in "sketch.js" is as follows:

```
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 graphics.textAlign(CENTER, CENTER)
10 }
11
12 function draw()
13 {
14   background(220)
15   // graphics.background(200, 200, 0)
16   graphics textSize(75)
17   graphics.text('HAPPY', 200, 200)
18   noStroke()
19   lights()
20   rotateX(angle)
21   rotateY(angle)
22   rotateZ(angle)
23   texture(graphics)
24   box(100)
25   angle++
26 }
```

The "Preview" window shows a 400x400 pixel canvas with a light gray background. In the center, the word "HAPPY" is written in a bold, black, sans-serif font, rotated diagonally upwards from left to right. Below the text, there are some small, dark, hand-drawn style marks or scratches.



Sketch C5.8 words on a plane

If we put it on a plane...

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
    graphics.textAlign(CENTER, CENTER)
}

function draw()
{
    background(220)
    // graphics.background(200, 200, 0)
    graphics textSize(75)
    graphics.text('HAPPY', 200, 200)
    noStroke()
    lights()
    rotateX(angle)
    rotateY(angle)
    rotateZ(angle)
    texture(graphics)
    plane(100)
    angle++
}
```



Notes

What is interesting is that the text always ends up the right way (not upside down or reversed) when you use all three axes.



Challenge

See what happens when you only have the **x** or **y** axis of rotation.

Figure C5.8

The screenshot shows the p5.js IDE 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, it says "Hello, TheHappyCoder! ▾". Below the toolbar, the title "sketch.js" is shown along with the message "Saved: just now". The code editor contains the following P5.js code:

```
sketch.js
7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 graphics.textAlign(CENTER, CENTER)
10 }
11
12 function draw()
13 {
14   background(220)
15   // graphics.background(200, 200, 0)
16   graphics.textSize(75)
17   graphics.text('HAPPY', 200, 200)
18   noStroke()
19   lights()
20   rotateX(angle)
21   rotateY(angle)
22   rotateZ(angle)
23   texture(graphics)
24   plane(100)
25   angle++
26 }
```

To the right of the code editor is a preview window showing a gray square containing the word "HAPPY" in black, rotated 45 degrees counter-clockwise. Below the code editor is a "Console" section with a "Clear" button.



Sketch C5.9 text on a cylinder

What about round the side of a cylinder? We need to make some adaptations to get this to work reasonably. You can play with everything later.

```
let angle = 0
let graphics

function setup()
{
    createCanvas(400, 400, WEBGL)
    angleMode(DEGREES)
    graphics = createGraphics(400, 400)
    graphics.textAlign(CENTER, CENTER)
}

function draw()
{
    background(220)
    graphics.background(220)
    graphics textSize(50)
    graphics.text('HAPPY CODER', 200, 200)
    noStroke()
    // lights()
    // rotateX(angle)
    rotateY(-angle)
    // rotateZ(angle)
    texture(graphics)
    cylinder(50, 200)
    angle++
}
```



Notes

Quite a bit of refactoring. I won't go into all of it, but I will allow you to experiment with what works or doesn't work.



Challenges

1. Try other shapes.
2. Add some colour.
3. Could you get the text to move up and down?

Figure C5.9

The screenshot shows the p5.js web editor interface. At the top, there's a toolbar with icons for play/pause, stop, auto-refresh (which is checked), and user information ('Hello, TheHappyCoder!'). Below the toolbar, the file name 'sketch.js' is displayed along with a 'Saved: just now' message. To the right of the file name is a 'Preview' button.

The code editor area contains the following JavaScript code:

```
sketch.js
Saved: just now
Preview

7 angleMode(DEGREES)
8 graphics = createGraphics(400, 400)
9 graphics.textAlign(CENTER, CENTER)
10 }
11
12 function draw()
13 {
14   background(220)
15   graphics.background(220)
16   graphics textSize(50)
17   graphics.text('HAPPY CODER', 200, 200)
18   noStroke()
19   // lights()
20   // rotateX(angle)
21   rotateY(-angle)
22   // rotateZ(angle)
23   texture(graphics)
24   cylinder(50, 200)
25   angle++
26 }
```

Below the code editor is a 'Console' section with a 'Clear' button. On the right side of the editor, there's a preview window showing a gray square with the text 'HAPPY CODER' centered in white.