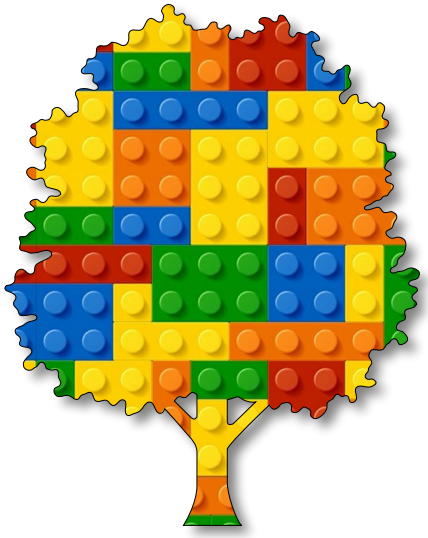
Algorithmic Art

Module F

Unit #1

Keyboard and mouse



Content

Module F Unit #1 Keyboard & Mouse

Introduction to the keyboard and mouse

This unit shows how you can interact with the mouse and keyboard. There are several useful functions built into p5. It can determine where the (x, y) position of the mouse is on the canvas, and know whether the mouse has been clicked or pressed, etc.

The Keyboard

You can use the keyboard to interact with objects and text. There are some special keys that are available in p5.js. Some of the special keys in p5.js are... (notice that they are uppercase)

The up arrow is UP_ARROW
The down arrow is DOWN_ARROW
The left arrow is LEFT_ARROW
The right arrow is RIGHT_ARROW
Return is RETURN

The Mouse

This can also be used to move objects around and interact with functions. This is a comprehensive list in the sketches below, but you will be surprised when you might need it.

Sketch F1.1 KeyIsPressed

❗ Start a new sketch

First, you will need to click on the canvas and then press any key on your keyboard.

```
function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  if (keyIsPressed === true)
  {
    fill(0)
  }
  else
  {
    fill(255)
  }
  square(100, 100, 100)
}
```

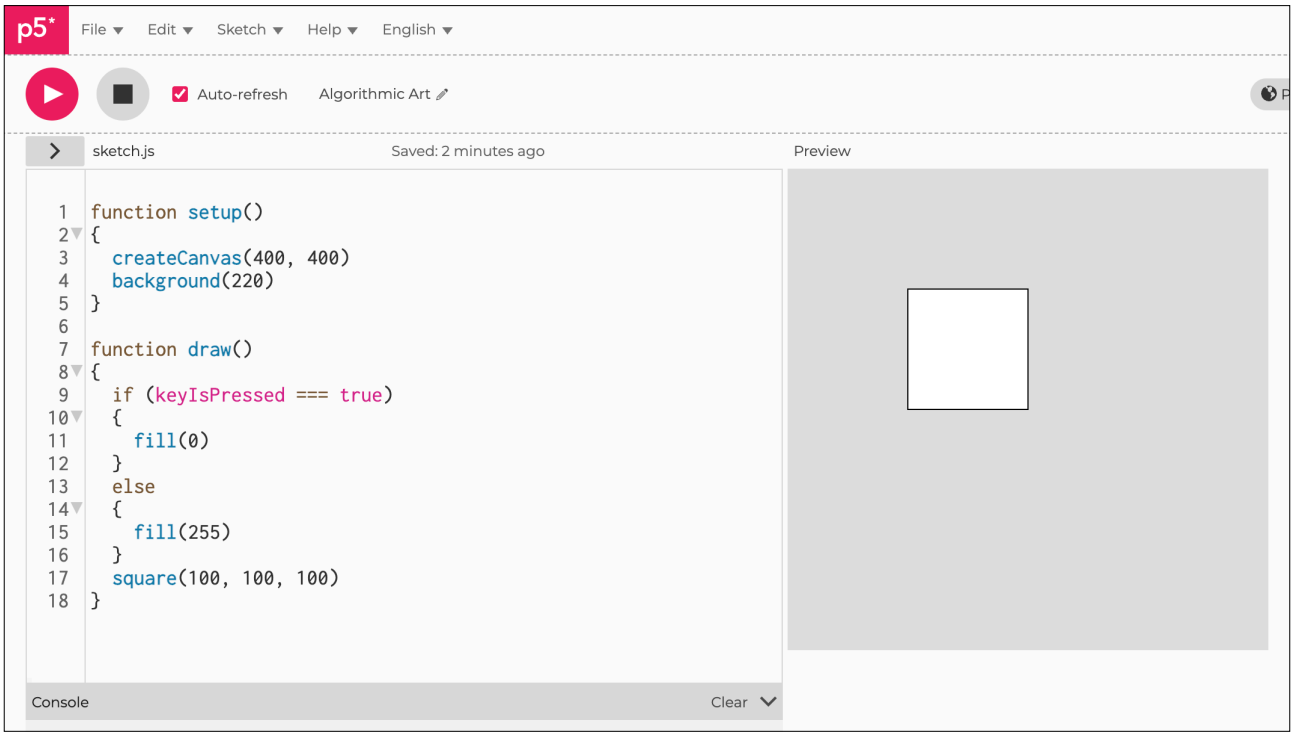
🗒️ Notes

The square will change from white to black when you press a key.

🛠️ Code Explanation

`if (keyIsPressed === true)`	Check to see if any key on the keyboard has been pressed

Figure F1.1


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 2 minutes ago
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 background(220)
5 }
6
7 function draw()
8 {
9 if (keyIsPressed === true)
10 {
11 fill(0)
12 }
13 else
14 {
15 fill(255)
16 }
17 square(100, 100, 100)
18 }
Console
Clear

Sketch F1.2 Key

❗ Start a new sketch.
This prints any key that is pressed on the canvas.

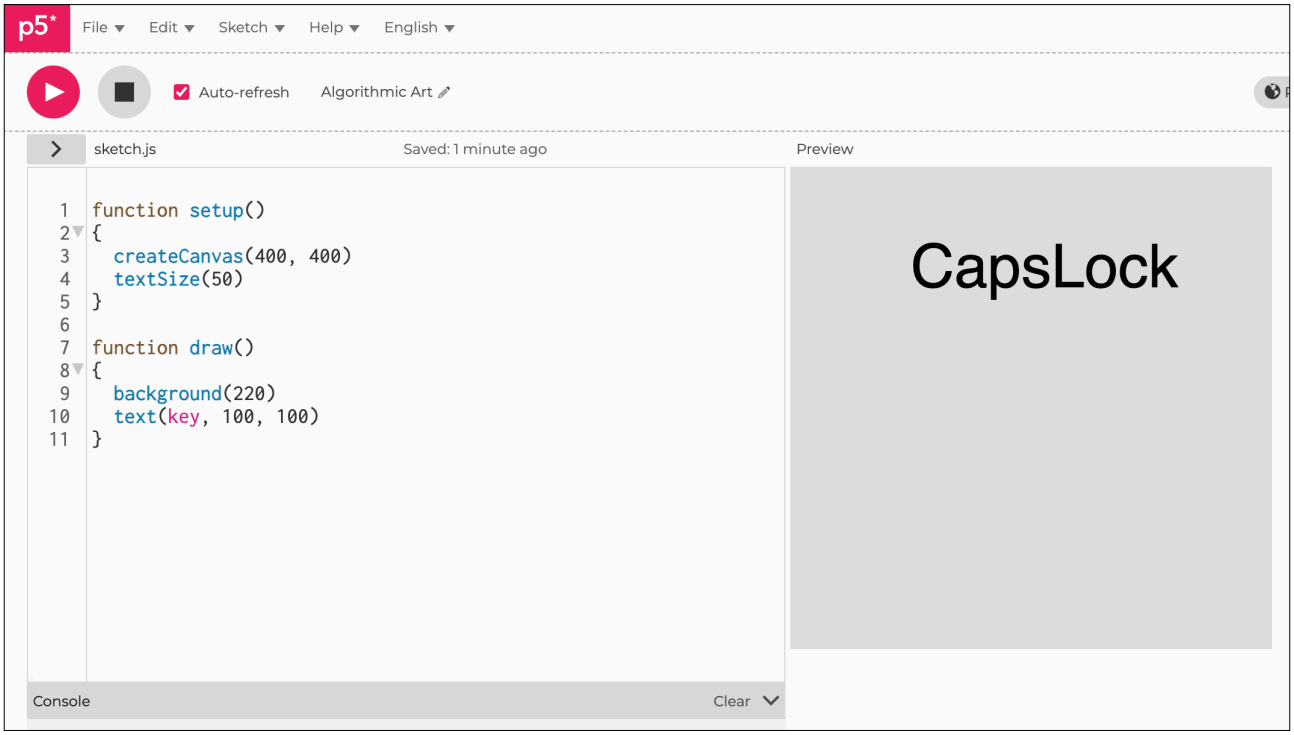
```
function setup()
{
  createCanvas(400, 400)
  textSize(50)
}

function draw()
{
  background(220)
  text(key, 100, 100)
}
```

🗒️ Notes
Showed the pressing of the Caps Lock key.

🛠️ Code Explanation

`text(key, 100, 100)`	Returns the key pressed

Figure F1.2


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 1 minute ago
Preview
function setup()
{
createCanvas(400, 400)
textSize(50)
}
function draw()
{
background(220)
text(key, 100, 100)
}
CapsLock
Console
Clear

Sketch F1.3 keyPressed()

❗ Start a new sketch

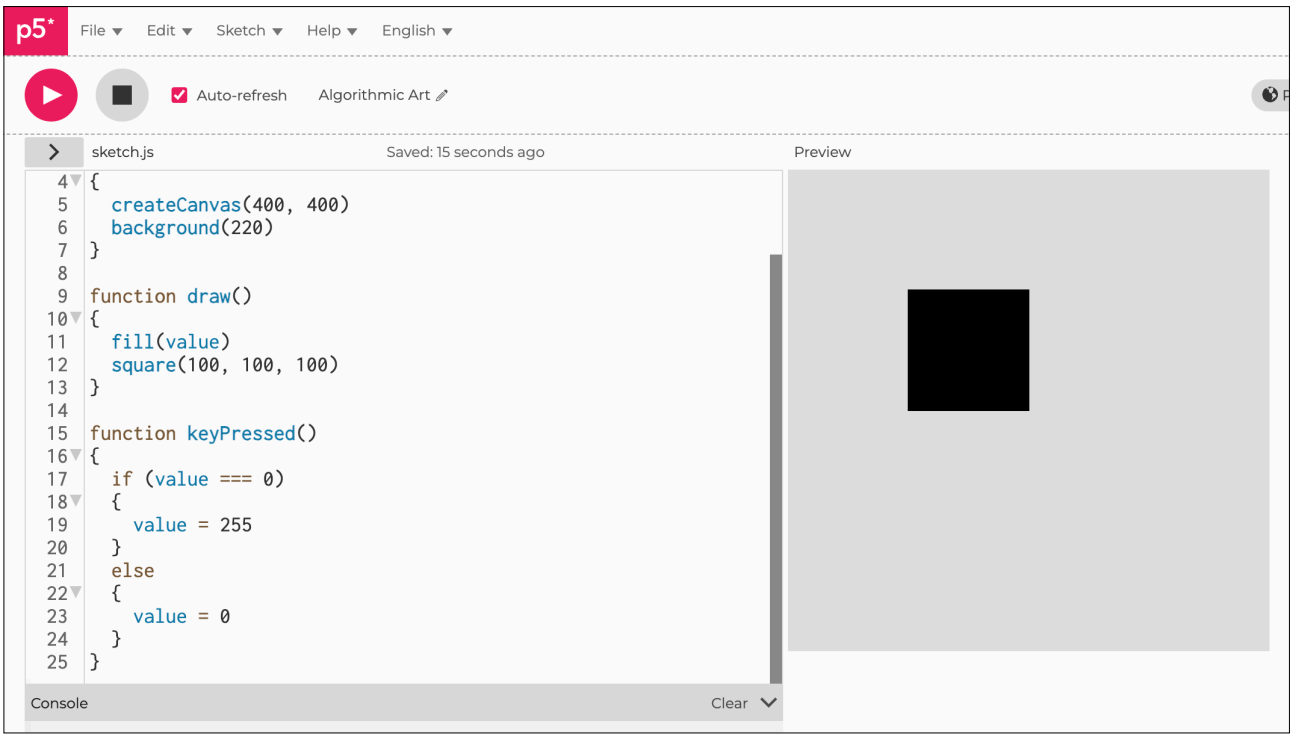
This toggles the square fill colour every time a key is pressed.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function keyPressed()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

🗒️ Notes

Toggles between being filled white and black every time you press any key.

Figure F1.3


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
{
createCanvas(400, 400)
background(220)
}
function draw()
{
fill(value)
square(100, 100, 100)
}
function keyPressed()
{
if (value === 0)
{
value = 255
}
else
{
value = 0
}
}
Console
Clear

Sketch F1.4 keyReleased()

This simply acts when the key is released

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function keyReleased()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

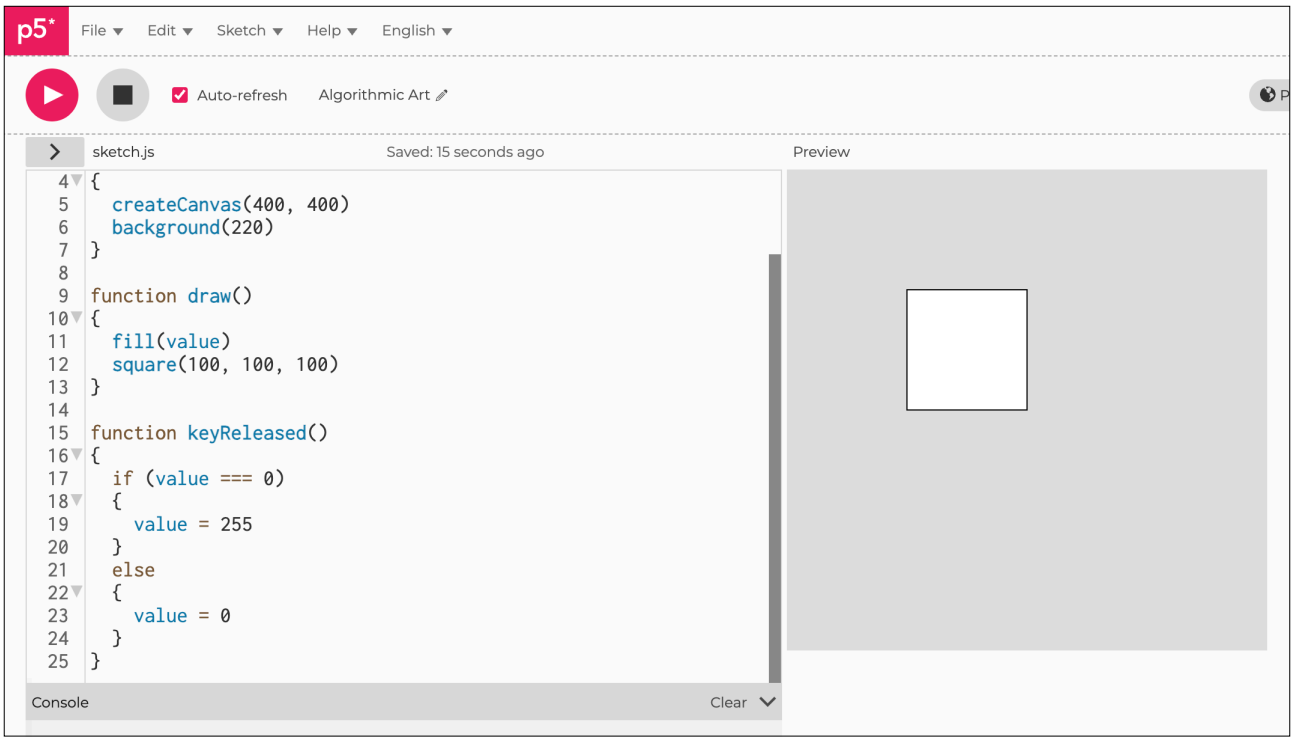
🗒️ Notes

Only changes at the point of release.

🛠️ Code Explanation

`function keyReleased()`	As you let go of the key

Figure F1.4


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
{
createCanvas(400, 400)
background(220)
}
function draw()
{
fill(value)
square(100, 100, 100)
}
function keyReleased()
{
if (value === 0)
{
value = 255
}
else
{
value = 0
}
}
Console
Clear

Sketch F1.5 KeyCode

Here we check for a specific key to be pressed; in this case, the up arrow (38) changes it to white, and the down arrow (40) toggles it to black.

```
let value = 126

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function keyPressed()
{
  if (keyCode === 38)
  {
    value = 255
  }
  else if(keyCode === 40)
  {
    value = 0
  }
}
```

🗒️ Notes

All keys have a numeric value. The effect is to change the colour of the square, black or white, depending on whether you press the up arrow or down arrow on your keyboard.

Sketch F1.6 keyTyped()

Press key A on the keyboard for a white square and B for a black square.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function keyTyped()
{
  if (key === 'a')
  {
    value = 255
  }
  else if (key === 'b')
  {
    value = 0
  }
}
```

Notes

We toggle black to white and vice versa.

Sketch F1.7 KeyIsDown()

❗ Suggest starting a new sketch.
This creates an event only while that specific key is held down. Use the arrow keys: up, down, left, and right to move the red circle.

```
let x = 100
let y = 100

function setup()
{
  createCanvas(400, 400)
  fill(200, 0, 0)
}

function draw()
{
  background(220)
  if (keyIsDown(LEFT_ARROW))
  {
    x -= 5
  }
  if (keyIsDown(RIGHT_ARROW))
  {
    x += 5
  }
  if (keyIsDown(UP_ARROW))
  {
    y -= 5
  }
  if (keyIsDown(DOWN_ARROW))
  {
    y += 5
  }
  circle(x, y, 50)
}
```

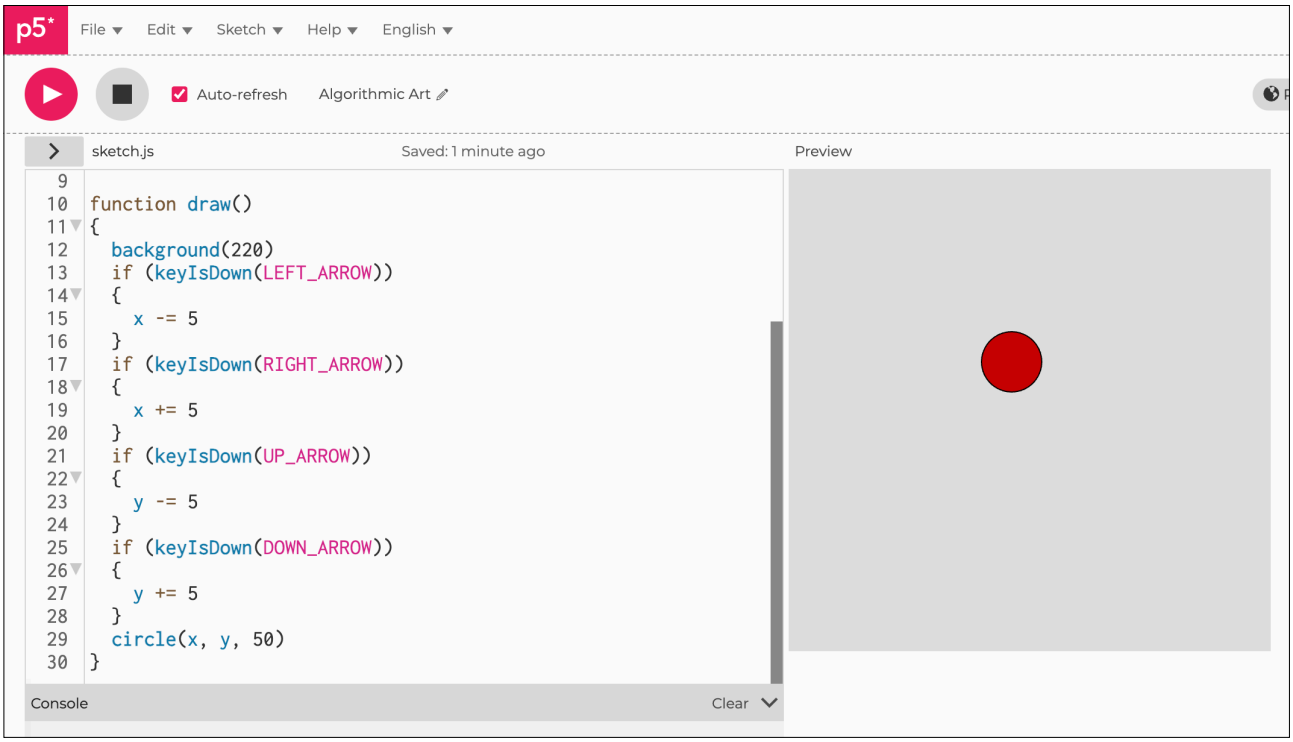
🗒️ Notes

The red circle should move easily around the canvas.

🛠️ Code Explanation

`if (keyIsDown(LEFT_ARROW))`	An example of calling a specific arrow key

Figure F1.7



Sketch F1.8 movedX

❗ Start a new sketch.
The variable movedX contains the horizontal movement of the mouse since the last frame.

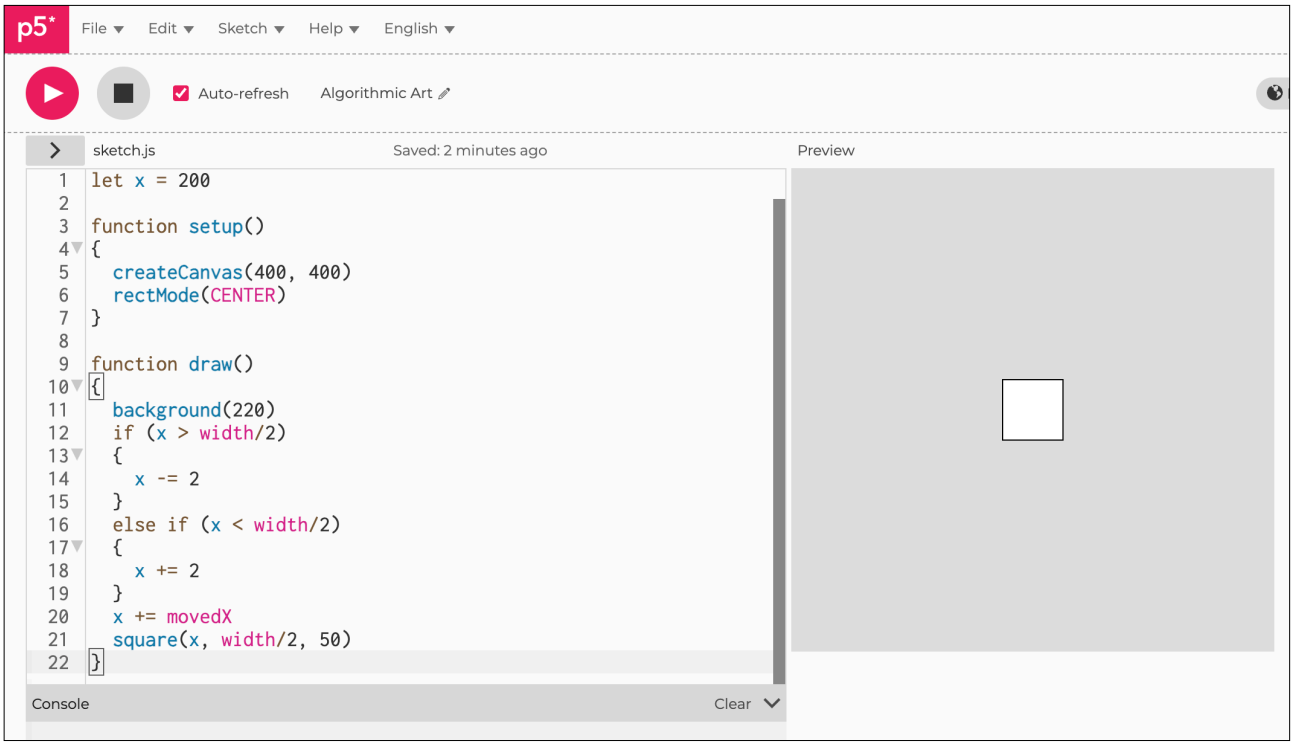
```
let x = 200

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

function draw()
{
  background(220)
  if (x > width/2)
  {
    x -= 2
  }
  else if (x < width/2)
  {
    x += 2
  }
  x += movedX
  square(x, width/2, 50)
}
```

🗒️ Notes
With this example, move your mouse to the left or right and then stop, you should see the square gently slide back This will also apply to movedY.

🛠️ Code Explanation

x += movedX	It is a variable based on the distance each frame from its last position.

Figure F1.8


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 2 minutes ago
Preview
let x = 200
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
}
function draw()
{
background(220)
if (x > width/2)
{
x -= 2
}
else if (x < width/2)
{
x += 2
}
x += movedX
square(x, width/2, 50)
}
Console
Clear

Sketch F1.9 winMouseX and pwinMouseX

❗ Start a new sketch.

This one is an interesting one with many things happening at once, where the pointer moves the canvas around, and the speed of the canvas determines the size of the circle.

```
let myCanvas
let speed

function setup()
{
  myCanvas = createCanvas(200, 200)
}

function draw()
{
  background(220)
  speed = winMouseX - pwinMouseX
  circle(width/2, height/2, 10 + speed * 5)
  myCanvas.position(winMouseX, winMouseY)
}
```

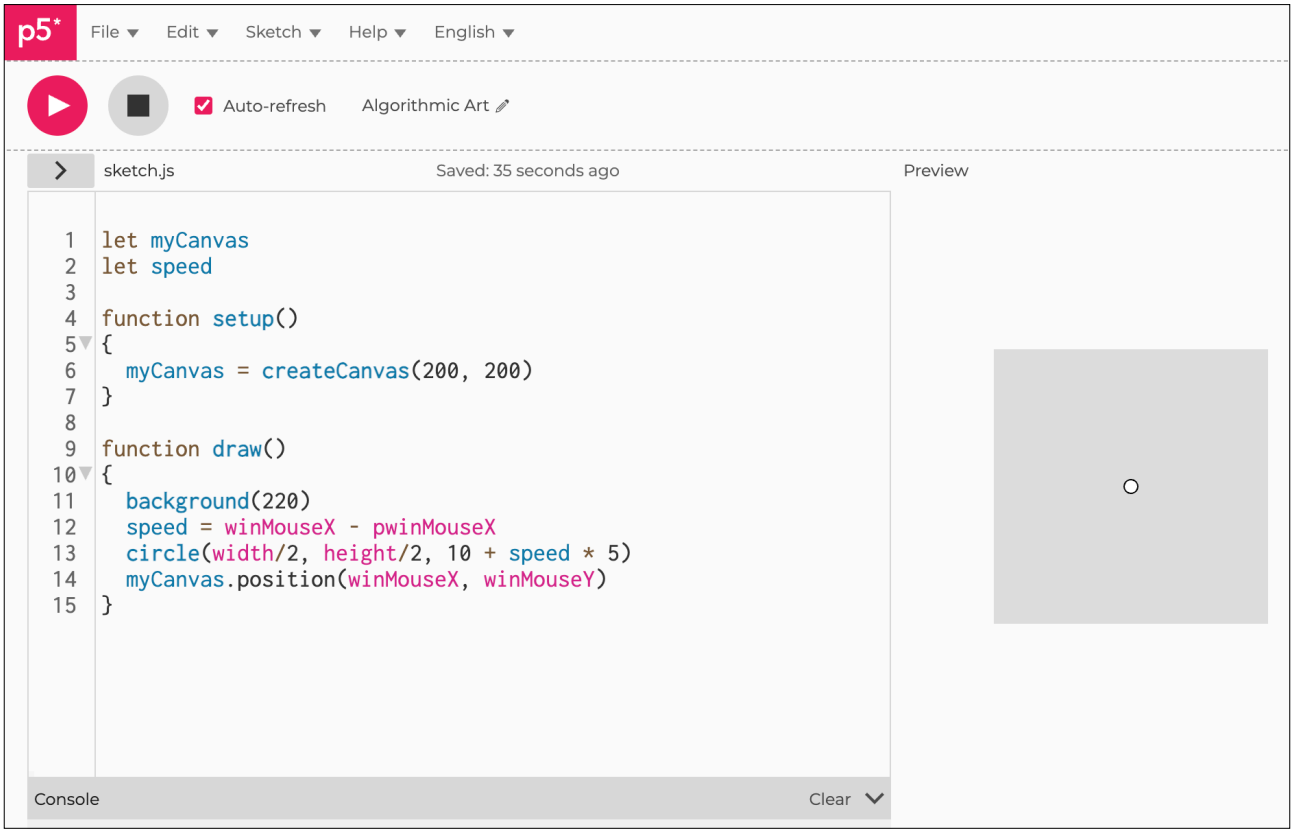
🗒️ Notes

The speed variable is the distance between the current position of the mouse and the previous position (at any one time), so if you move it fast, then that distance is larger momentarily until the previous one catches up to the current.

🛠️ Code Explanation

`myCanvas = createCanvas(200, 200)`	Create a new canvas object

Figure F1.9


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 35 seconds ago
Preview
1 let myCanvas
2 let speed
3
4 function setup()
5 {
6 myCanvas = createCanvas(200, 200)
7 }
8
9 function draw()
10 {
11 background(220)
12 speed = winMouseX - pwinMouseX
13 circle(width/2, height/2, 10 + speed * 5)
14 myCanvas.position(winMouseX, winMouseY)
15 }
Console
Clear

Sketch F1.10 mouseIsPressed()

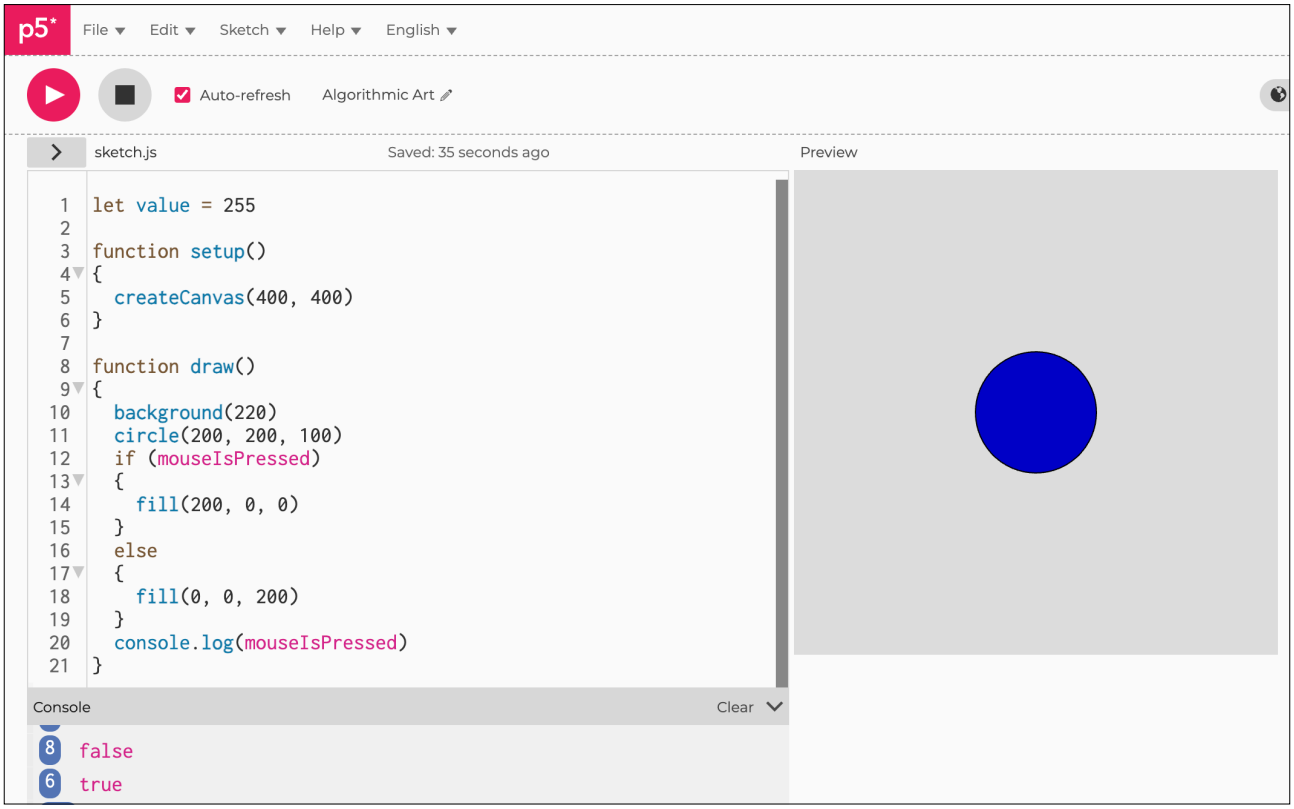
❗ Start a new sketch.
Simple responds when any mouse button is pressed.

```
let value = 255

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

function draw()
{
  background(220)
  circle(200, 200, 100)
  if (mouseIsPressed)
  {
    fill(200, 0, 0)
  }
  else
  {
    fill(0, 0, 200)
  }
  console.log(mouseIsPressed)
}
```

Notes

When you click on one of the mouse buttons, the circle turns red. The console records a true or false response to the mouse button.

Figure F1.10


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 35 seconds ago
Preview
let value = 255
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
circle(200, 200, 100)
if (mouseIsPressed)
{
fill(200, 0, 0)
}
else
{
fill(0, 0, 200)
}
console.log(mouseIsPressed)
}
Console
Clear
false
true

Sketch F1.11 mouseButton()

This provides feedback depending on which button you press on your mouse (including the wheel).

❗ Remove console.log()

```
let value = 255

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

function draw()
{
  background(220)
  circle(200, 200, 100)
  if (mouseIsPressed)
  {
    if (mouseButton.left)
    {
      fill(200, 0, 0)
    }
    if (mouseButton.right)
    {
      fill(0, 200, 0)
    }
    if (mouseButton.center)
    {
      fill(0, 0, 200)
    }
  }
}
```

🗒️ Notes

Depending on which button you click, it will change the colour of the circle accordingly. Useful if you need to identify which button is pressed.

Sketch F1.12 mouseMoved()

❗ A new sketch.
Move your mouse cursor across the canvas.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function mouseMoved()
{
  value += 10
  if (value > 255)
  {
    value = 0
  }
}
```

🗒️ Notes

As you move the mouse, you get the effect of changing the greyscale colour of the square in degrees of 10.

Sketch F1.13 mouseDragged()

This time it only changes when the mouse button is held down as it is moved.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function mouseDragged()
{
  value += 10
  if (value > 255)
  {
    value = 0
  }
}
```

Notes

The same effect. This is a useful mouse function.

Sketch F1.14 mousePressed()

This is a simple way to toggle using a mouse button.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function mousePressed()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

Notes

Just toggles between a black square and a white square.

Sketch F1.15 mouseReleased()

Does the same but only acts when the button is released.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function mouseReleased()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

Notes

Almost the same effect.

Sketch F1.16 mouseClicked()

Very similar to mousePressed.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function mouseClicked()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

Notes

There is a very subtle difference. Useful in certain conditions.

Sketch F1.17 doubleClicked()

As it says on the tin.

```
let value = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  fill(value)
  square(100, 100, 100)
}

function doubleClicked()
{
  if (value === 0)
  {
    value = 255
  }
  else
  {
    value = 0
  }
}
```

🗒️ Notes

Just another option, especially if you want one action for one click and something different for a double-click.

Sketch F1.18 mouseWheel()

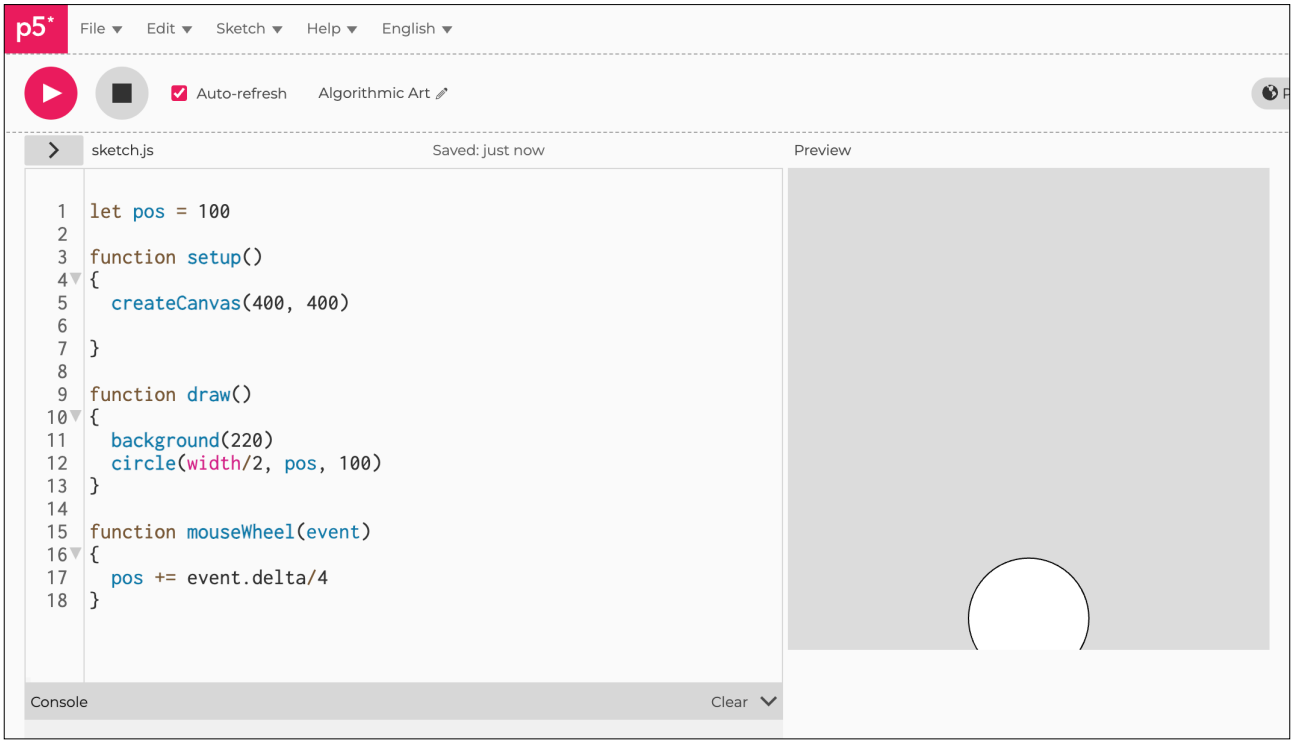
❗ A new sketch.
The event.delta property returns the amount the mouse wheel has scrolled.

```
let pos = 100

function setup()
{
  createCanvas(400, 400)

}

function draw()
{
  background(220)
  circle(width/2, pos, 100)
}

function mouseWheel(event)
{
  pos += event.delta/4
}
```

🗒️ Notes

The circle moves up and down with the wheel of the mouse. The event.delta value can be altered to suit your mouse. I have divided by 4 just for a slightly slower response. You can change the direction using -= instead.

Figure F1.18


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
let pos = 100
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
circle(width/2, pos, 100)
}
function mouseWheel(event)
{
pos += event.delta/4
}
Console
Clear

Sketch F1.19 easing

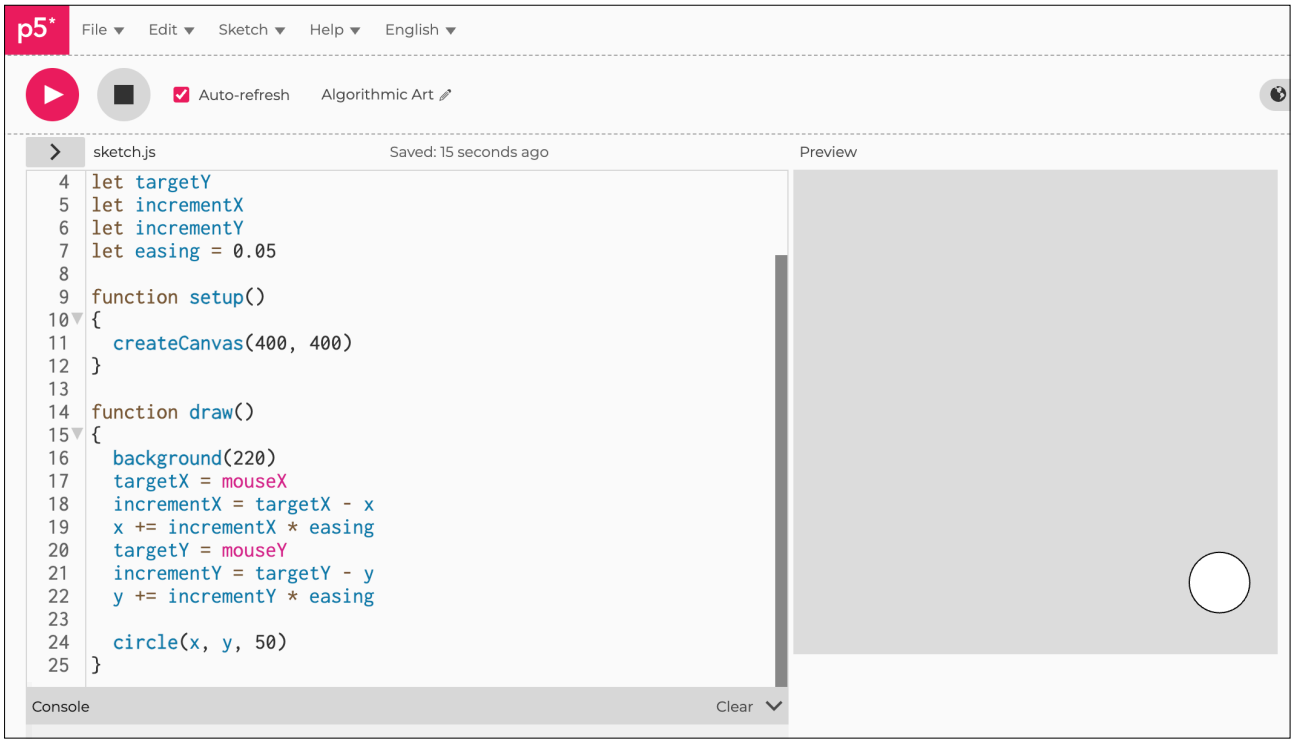
❗ A new sketch.
Smoothing out the motion.

```
let x = 0
let y = 0
let targetX
let targetY
let incrementX
let incrementY
let easing = 0.05

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

function draw()
{
  background(220)
  targetX = mouseX
  incrementX = targetX - x
  x += incrementX * easing
  targetY = mouseY
  incrementY = targetY - y
  y += incrementY * easing

  circle(x, y, 50)
}
```

🗒️ Notes
This is just a bit of fun to show how to make an object move smoothly. It could be applied to many things. The variable `easing` is not a function but just a variable name, but easing is the description of smoothing.

🌻 Challenge
Try making the value of easing = 1 and see what happens.

Figure F1.19


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
4 let targetY
5 let incrementX
6 let incrementY
7 let easing = 0.05
8
9 function setup()
10 {
11 createCanvas(400, 400)
12 }
13
14 function draw()
15 {
16 background(220)
17 targetX = mouseX
18 incrementX = targetX - x
19 x += incrementX * easing
20 targetY = mouseY
21 incrementY = targetY - y
22 y += incrementY * easing
23
24 circle(x, y, 50)
25 }
Console
Clear