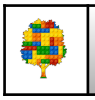
Section B
Unit #7
Keyboard
Mouse



Content

Unit #7 Keyboard & Mouse

Unit #7 Keyboard & 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 mouse 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

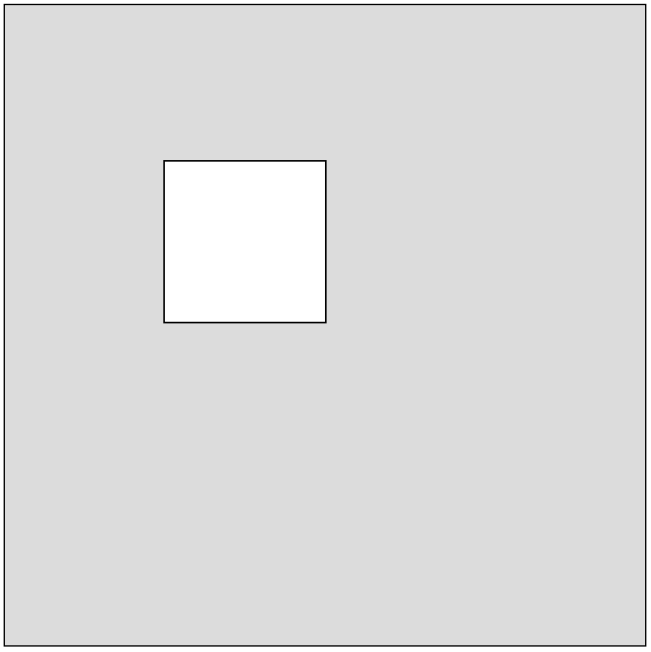
This can also be used to move objects around and interact with function. This is a comprehensive list in the sketches below but you will surprised when you might need it.

Sketch B7.1 KeyIsPressed

First you will need to click on the canvas and then press any key, it changes anything when any key is pressed

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

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

Click on the canvas and then press any key, it should toggle between black and white

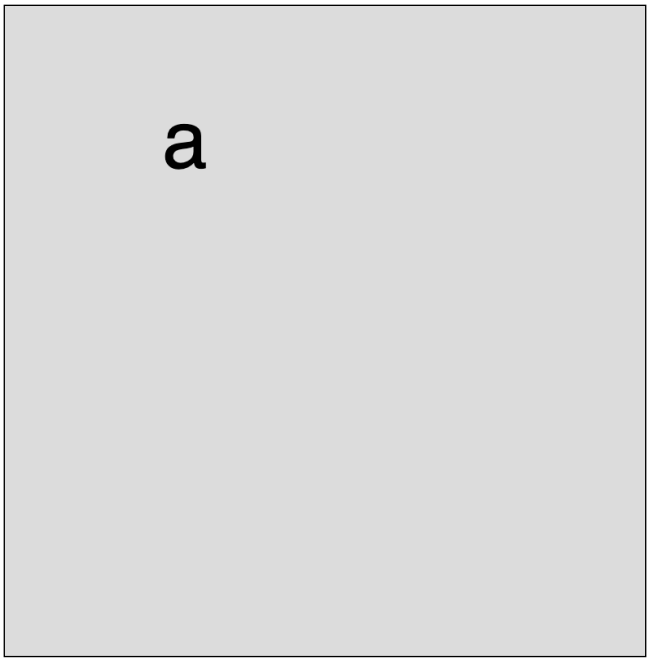


Sketch B7.2 Key

This prints any key that is pressed

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

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

I pressed the letter A on the keyboard
(remember to click the canvas first)



Sketch B7.3 keyPressed()

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
  }
}
```

Sketch B7.4 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
  }
}
```

Sketch B7.5 KeyCode

Here we check for a specific key to be pressed, in this case the up arrow changes it to white and the down arrow 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 === UP_ARROW)
  {
    value = 255
  }
  else if(keyCode === DOWN_ARROW)
  {
    value = 0
  }
}
```

Sketch B7.6 keyTyped()

Press key **A** on the keyboard for white square and **B** for 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
  }
}
```

Sketch B7.7 KeyIsDown()

This creates an event only while that specific key is held down, suggest you start a new sketch. 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)
}
```

Sketch B7.8 movedX

The variable **movedX** contains the horizontal movement of the mouse since the last frame - **movedY** is obvious. With this example, move you mouse slightly left or right and then stop, you should see it gently slide back

```
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

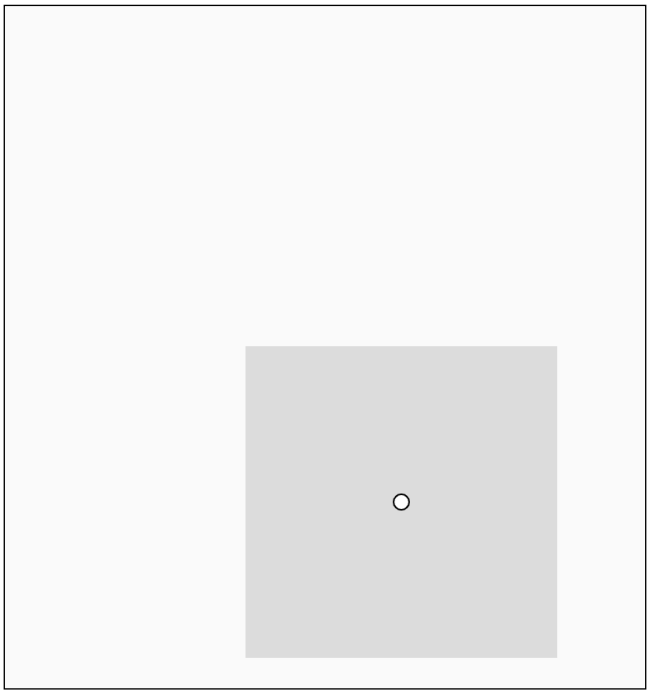
This will also apply to movedY

Sketch B7.9 winMouseX and pwinMouseX

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)
}
```

As you move the mouse the canvas moves too also the circle responds to the speed you move it

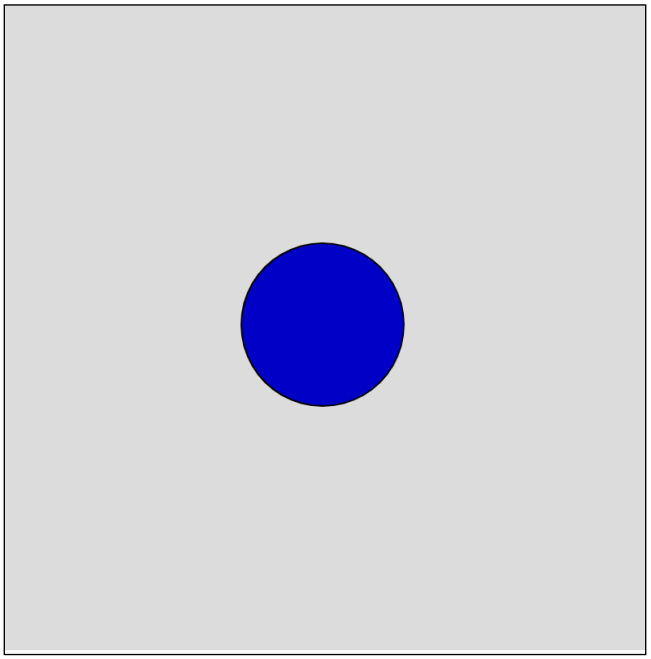


Sketch B7.10 mouseIsPressed()

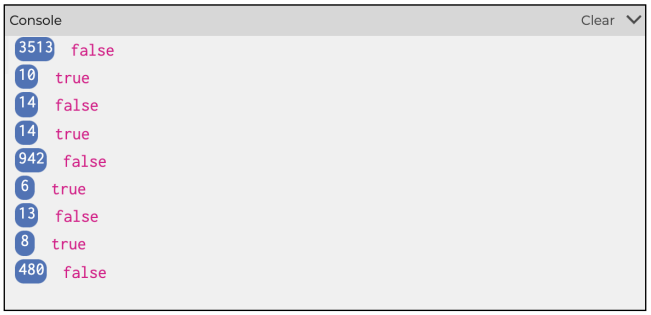
Simple responds when any mouse button is pressed

```
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)
}
```

When you click on one of the mouse buttons the circle turns red



The console records with a true or false response to the mouse button

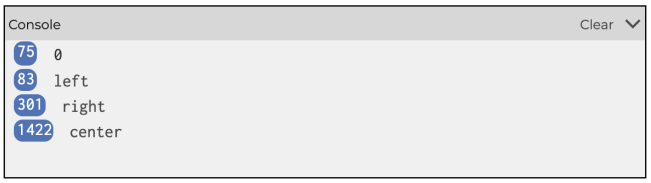


Sketch B7.11 mouseButton()

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

```
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)
    }
  }
  console.log(mouseButton)
}
```

Useful if you need to identify which button is pressed



Sketch B7.12 mouseMoved()

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
  }
}
```

Sketch B7.13 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
  }
}
```

Sketch B7.14 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
  }
}
```

Sketch B7.15 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
  }
}
```

Sketch B7.16 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
  }
}
```

Sketch B7.17 doubleClicked()

As it says on the tin

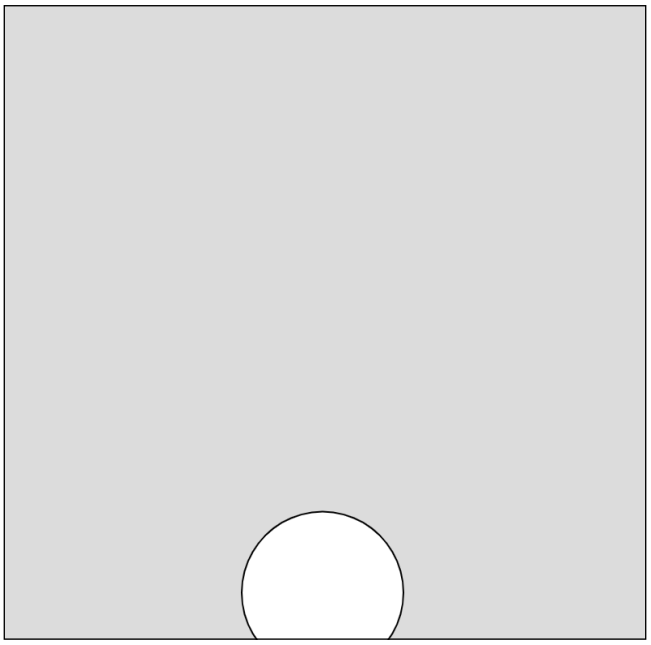
```
let value = 0

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
  }
}
```

Sketch B7.18 mouseWheel()

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
}
```

The circle moves up and down with the wheel of the mouse



Notes

The event.delta value can be altered to suite your mouse, I have divided by 4 just for a slightly slower response.

Sketch B7.19 easing

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