blue {
shift
}

3 - Interaction in Processing

Overview: We are going to learn to control shapes in Processing using the computer mouse and keyboard.

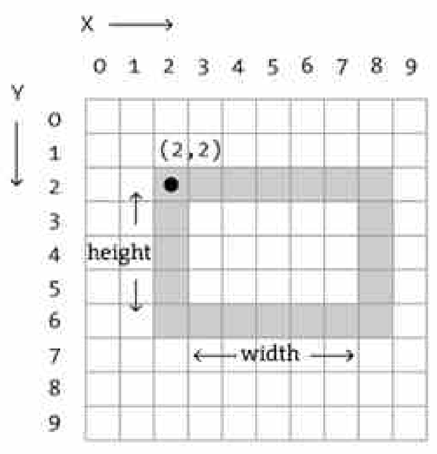
STEP 1: RECAP SESSION 2 - Drawing Shapes

Drawing shapes have multiple values needed, a point only needs two, the x and y. For rectangles, we have 4 values:

```
rect(x, y, width, height);
```

Example:

```
rect(2, 2, 7, 5);
```



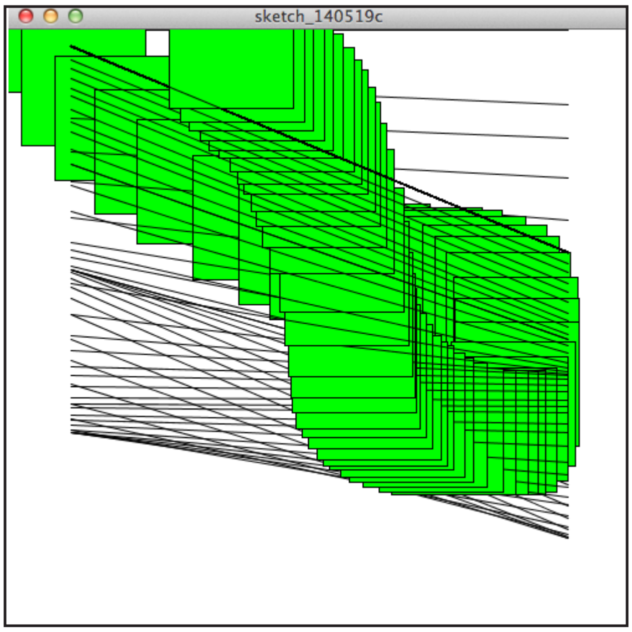
Processing was designed to create visual images with a simple line of code.

You can draw a range of different shapes to the processing window.

2D Primitives

point()
line()
rect()
ellipse()
arc()
quad()
triangle()

Processing Reference:
http://www.processing.org/reference/

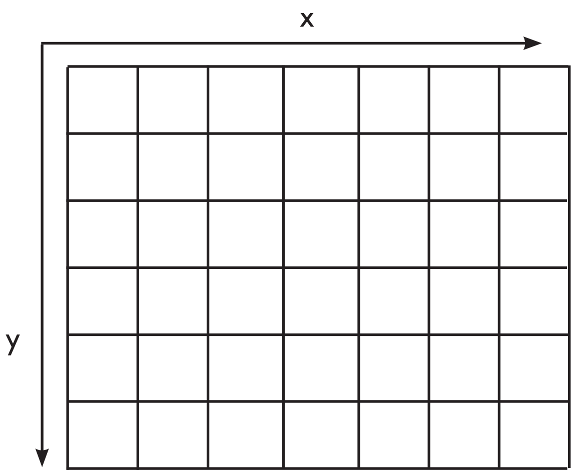


STEP 2: USING THE MOUSE TO MOVE THINGS

You can use the computer inputs to control objects on the screen.

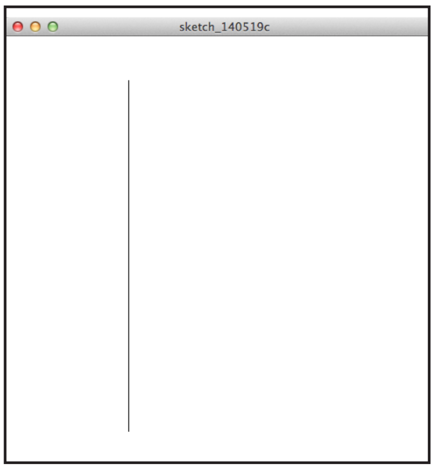
First we will use the mouse to move a shape on the screen.

mouseX - moves along the x axis
mouseY - will move along the y axis



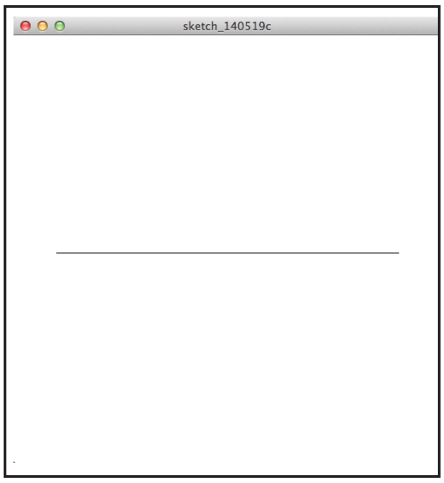
Lets start by using only the x axis. Type in the code below:

```
void setup(){
  size(500,500);

}


void draw()
{
  background(255);
  line(mouseX, 50, mouseY, 450);
}
```



We will now change the axis from the x axis to the y axis using the mouseY input.

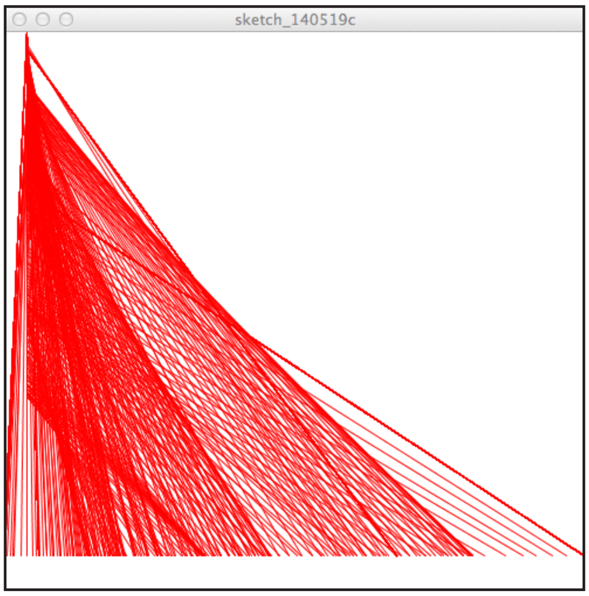
```
line(50, mouseY, 450, mouseY);
```



Experiment changing the length of your line, also using the stroke()change the colour of your line.

Now lets experiment with moving your background () from the draw() function to the setup()function.

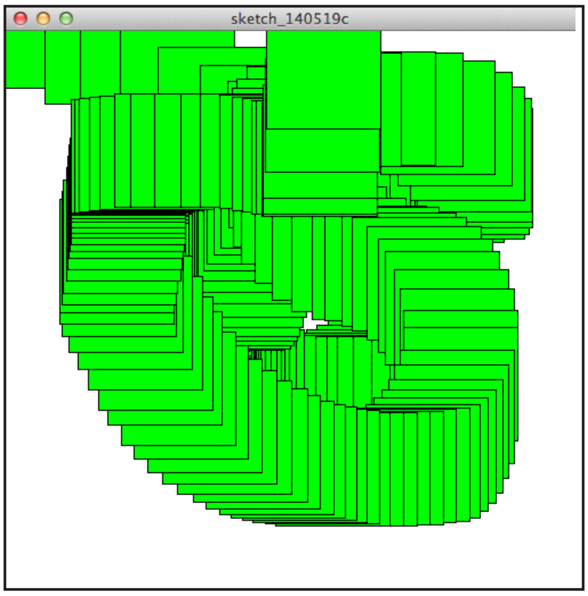
What do you see on the screen?



Now lets experiment with changing the values in your line() code, mix up using both the mouseX and mouseY.

Lets try adding in a rectangle to our code. To be able to move the rectangle using the mouse we need to tell the program where to place our mouse, by default the rectangle is place on the top left corner of the shape. Using the rectMode() we can control this. We will use the rectMode(CENTER) in our code:

```
void setup() {
  size(500, 500);
  background(255);
  rectMode(CENTER);
}

void draw()
{

  fill(0, 255, 0);
  rect(mouseX, mouseY, 100, 100);
}
```



note Now add more rectangles and ellipse to your code, make each one is a different colour.

STEP 3: KEYPRESS

The next stage is understanding how we can use the keys on our computer. To understand this we will first need to take a look at using if and else statements in our code.

The if statement:

When you hit the key = do something (in this case draw a red square).

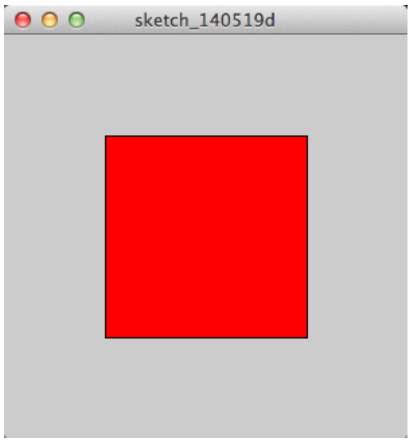
The else statement:

If you are not pressing down the key = do something else (make the square change to green).

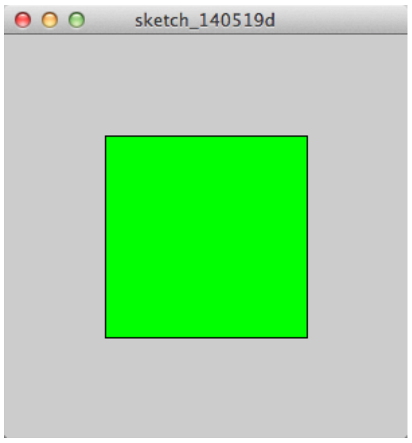
Lets add a square to the window and see what happens when you press a key on your keyboard:

```
void setup() {
 size(300, 300);
}
void draw() {
 if (keyPressed) {
  fill(0, 255, 0);
 }
 else {
  fill(255, 0, 0);
 }
 rect(75, 75, 150, 150);
}
```

Before a key is pressed



After a key is pressed



Now we will be more specific with controlling our shape colour. The above code tells us that we can draw the square whenever any key is press. Now we want to be able to control many shapes when many different keypresses.

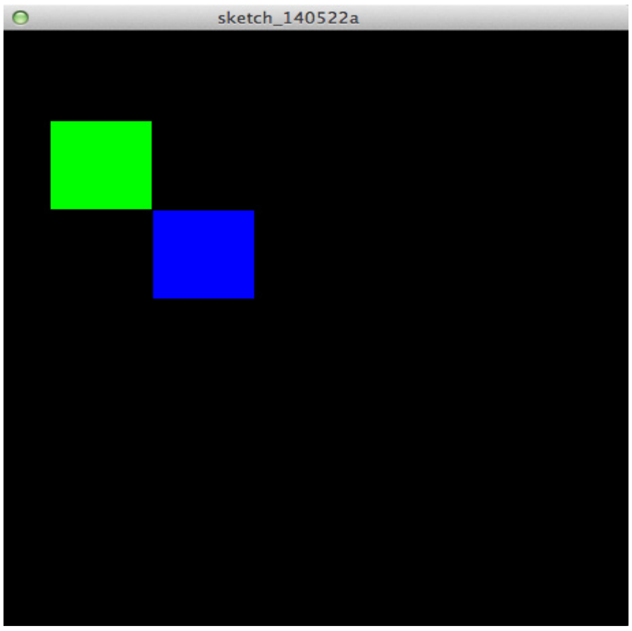
```
if (keyPressed) {
  if (key == 'b')
```

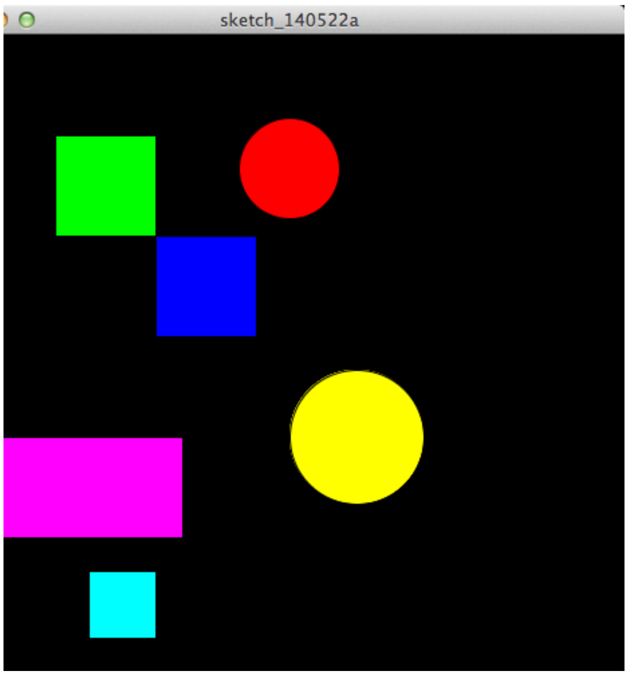
Can you work out how to control two different shapes using two different keys?

Now we will add lots of different shapes onto our screen. We have used squares here but you can add any shapes you wish.

For this code you will need to add a black backgound, remebering the colour code is (0, 0, 0) for black.

```
void setup() {
  size(500, 500);
  background(0, 0, 0);
}
void draw() {
  if (keyPressed) {
    if (key == 'b') {
      fill(0, 255, 0);
      rect(75, 75, 75, 75);
    }
    if (key == 'c') {
      fill(0, 0, 255);
      rect(150, 150, 75, 75);
    }
  }
```





At the end of your code we will add in the else statement, when the 'p' key is pressed then the screen will clear.

```
else {
  if (key == 'p') {
    clear();
  }
 }
}
```

Now its your chance to make your own, using at least 6 letters on your keyboards, control 6 different shapes on your screen. Don't forget to add a clear option in your code!

STEP 4: RANDOM FUNCTION

You can easily generate random numbers using the random() function. You can use this to make the shapes colour change every time it is redrawn. To do this the random() function requires two values , which determine the range.

Whenever the random() function is used an unexpected number will be called.

For Example:

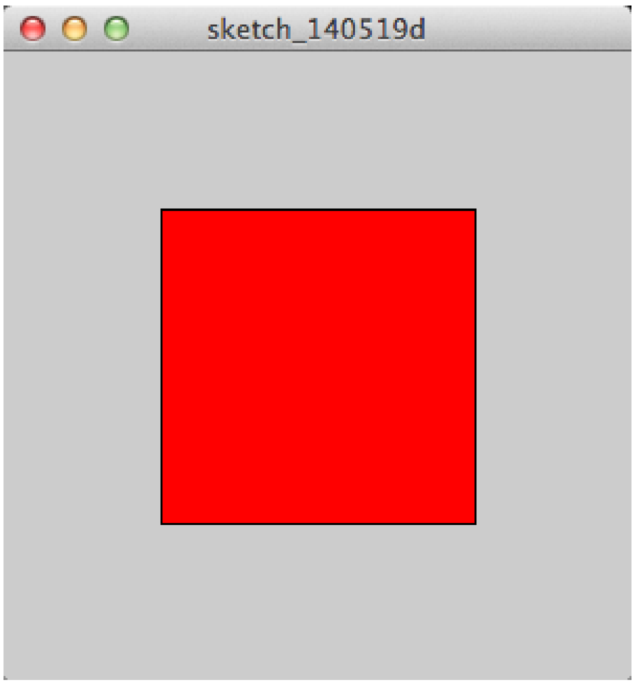
```
random(10)
```

Will return a number from 0 and upto, but not including 10. So anywhere between 0 and 9.

```
random(5, 10)
```

Will return a number from 5 and upto, but not including 10. So anywhere between 5 and 9.

To use this for calling unexpected colours in our shapes, we can add a random() function to the shapes fill(R,G,B) code.



```
fill(random(0, 255), random(0,255), random(0,255));
  rect(75, 75, 75, 75);
```

note Try adding the random() function into all your shapes, what do you see happen every time you press the keys?

1. Can you alter the code from our faces we made in the last session?
2. Can you add the random() function to anything else in your code?

DON'T FORGET TO SAVE YOUR WORK FROM THIS SESSION!