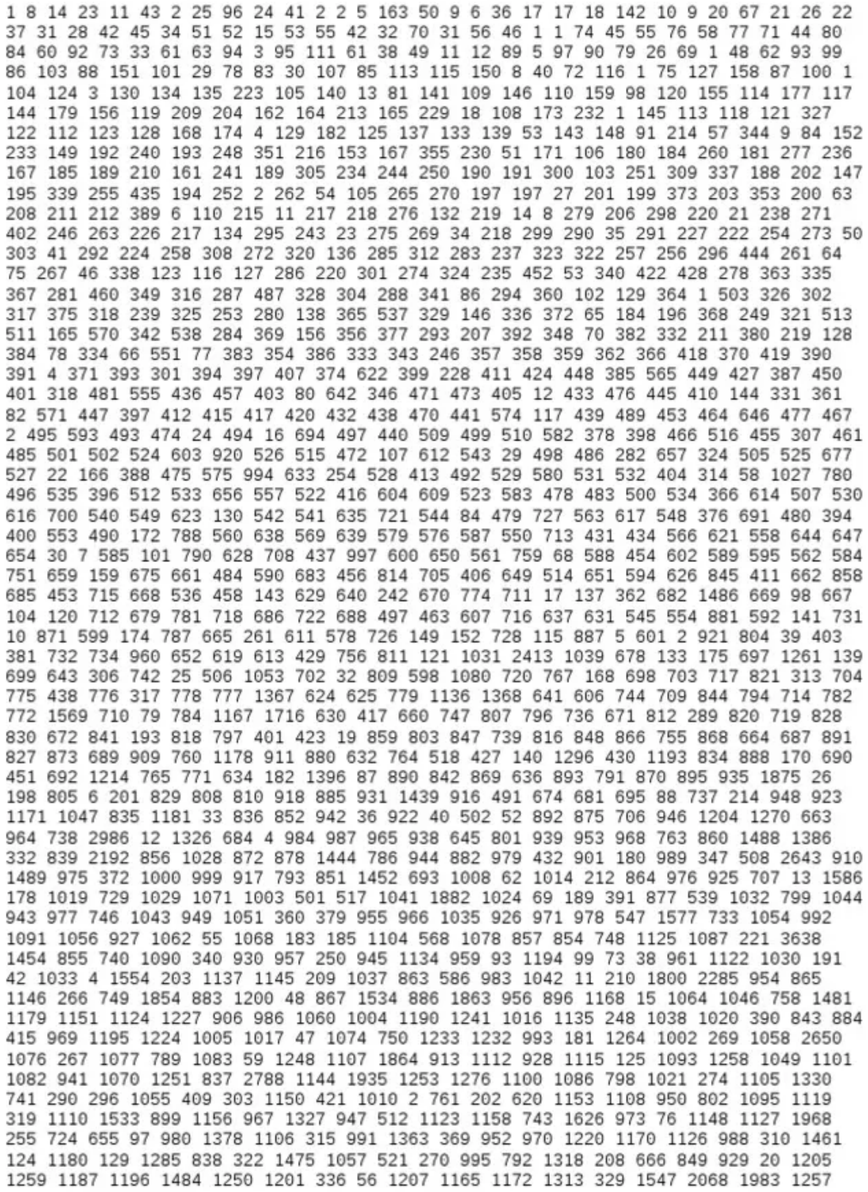


Creative Coding & Creative Computing Frameworks

Week4: Refresher...with a little bit of randomness

Hour 1:
Introduce the cheat sheet and review past resources and content.

Hours 2 & 3 (lecturers swap rooms between hours):
Room One: Refresher of theories and histories
Room Two: Exercise of using randomness in code:
a. In a p5.js sketch

Hour four:
Develop your 'random' exercise AND/OR go over past concepts and exercises - lecturers will be around to help.

Catchup: The Coding Environment and Core Concepts

The Creative Coding Cheat Sheet
We've produced a PDF that covers resources, concepts and techniques that were introduced in the previous weeks, including:

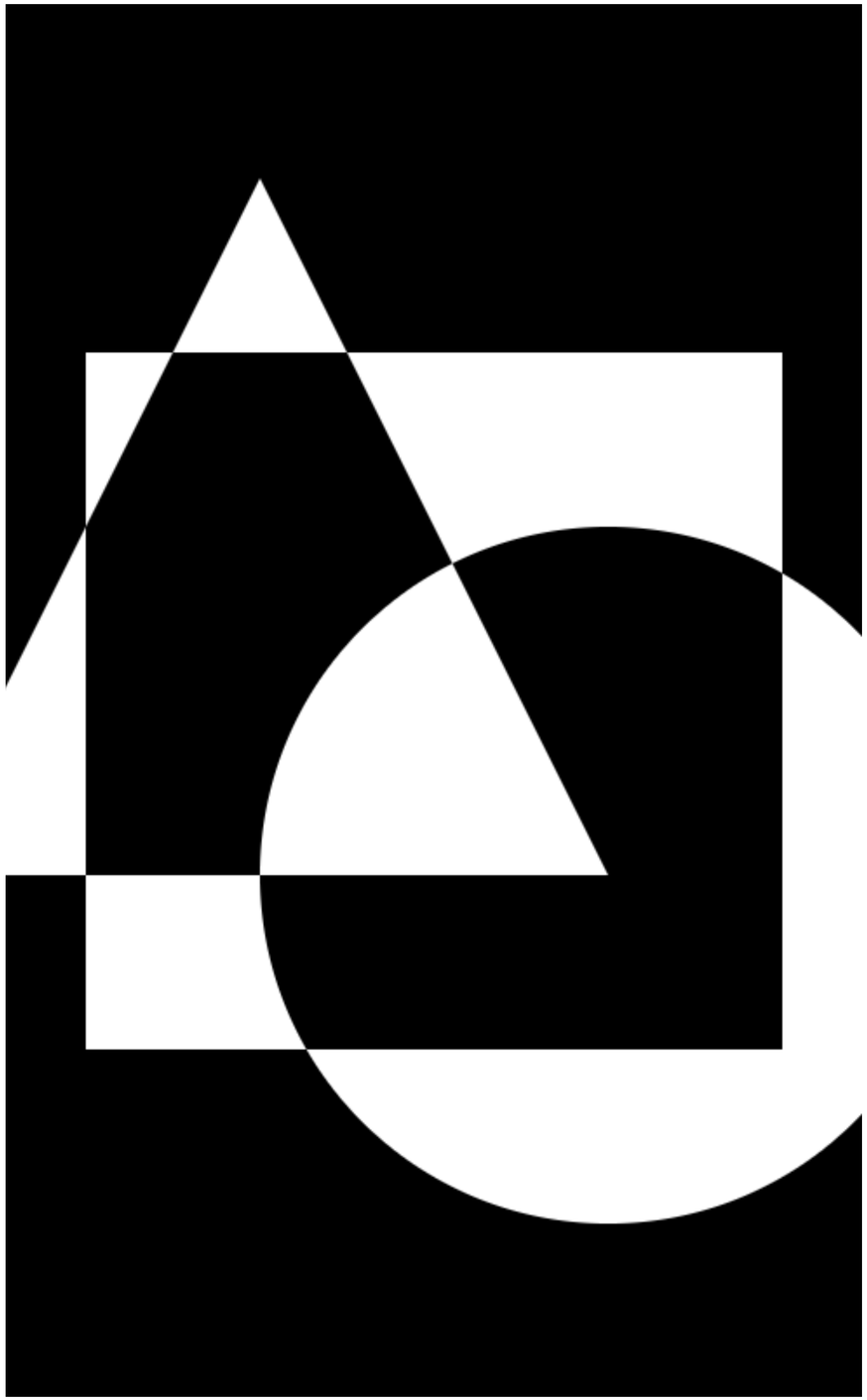
- Key technical resources
- Setting up your coding environment
- HTML+CSS+JavaScript
- Core programming concepts
- JavaScript Frameworks

Let's take a look at the Cheat Sheet and have a quick refresher.

Download from:
https://github.com/IrtizaNasar/CCI-Diploma22-CreativeCoding/raw/main/Creative_Coding_Cheat_Sheet.pdf

Note: we will periodically update this doc.

Also see: The GitHub repository for this unit:
https://github.com/IrtizaNasar/CCI-Diploma22-CreativeCoding



Catchup: p5.js

For this catchup session we are going to code along with each other to create this drawing.

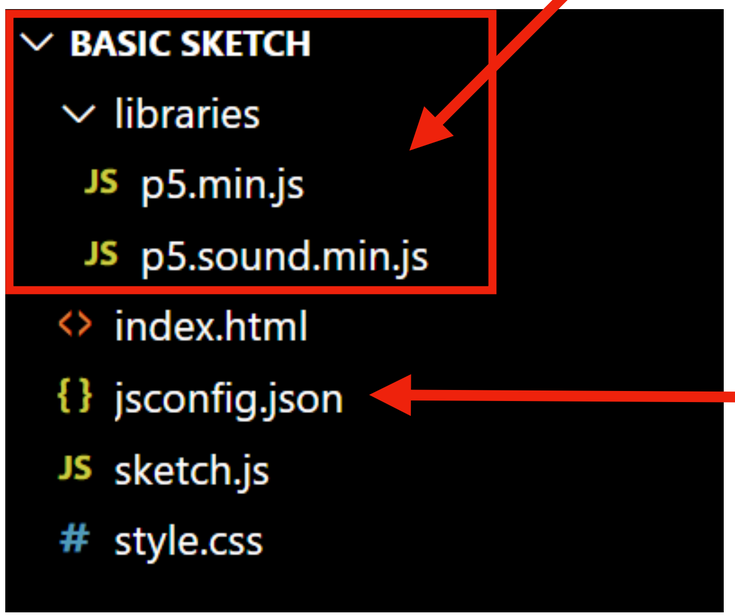
Open VSCode and create a new p5 project. Make sure you put it in a folder on your desktop.

Catchup: p5.js

p5.js

Setting up your canvas in p5
When you create a new p5 project you will find a folder containing the files you need to run your code.

The libraries folder contains your p5 library. This is so you can use all the p5 specific functions such as draw() or rect(). You don't need to do anything with this.

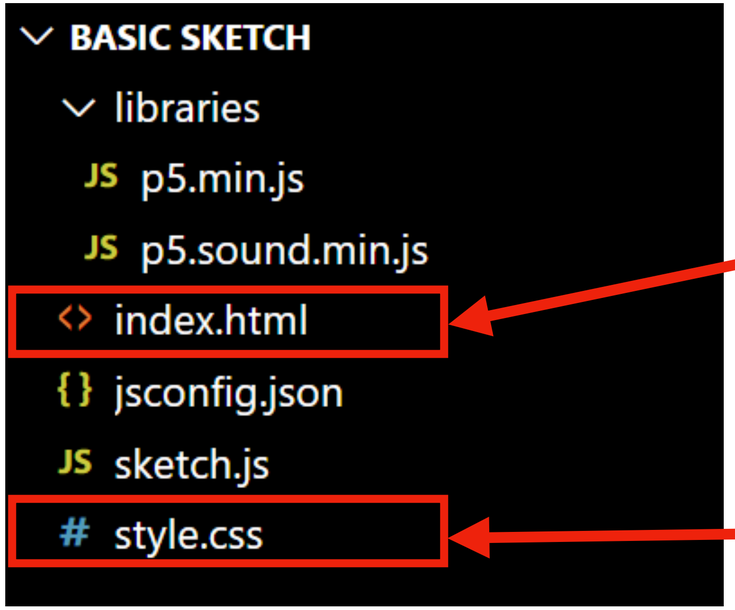


This file is related to linking your p5 library to your sketch.
You don't need to do anything with this.

Catchup: p5.js

p5.js

Setting up your canvas in p5
When you create a new p5 project you will find a folder containing the files you need to run your code.



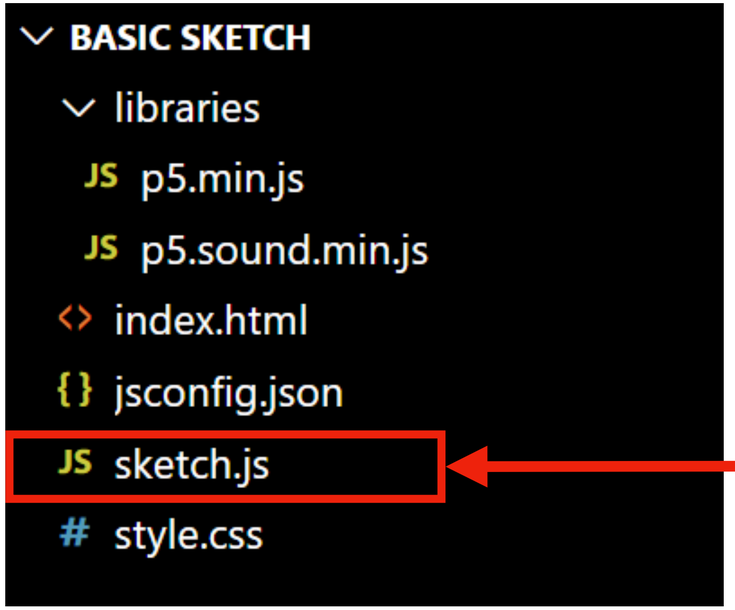
This is your HTML file. You open this in your browser (Chrome). It contains HTML code to control all the formatting on how your webpage looks in a browser..

This is your CSS file. It controls all the styling of each element your webpage.

Catchup: p5.js

p5.js

Setting up your canvas in p5
When you create a new p5 project you will find a folder containing the files you need to run your code.



This is your JavaScript file. This is the <script> element of your page and where your p5 code resides. Open and edit this file with VSCode.

Catchup: p5.js

Setting up your canvas in p5
Your `setup()` function is specific to p5 and is set to run only when you first run your page. If you refresh your page in your browser window, it will rerun from the start and the setup code will run again at the start.

The `createCanvas()` function sets the size of your canvas.

`createCanvas(400,400);`
- will set the size of the canvas to 400 by 400 pixels.

`windowWidth | windowHeight`
If you want to size of your canvas to take up the whole page you can set the size to `windowWidth` and `windowHeight`. These are values that p5 knows and will automatically resize to how large your browser is. Notice that they are written in cameCase.

```
1   function setup() {
2     createCanvas(windowWidth, windowHeight);
3   }
```

Catchup: p5.js

Setting up your canvas in p5
Your `setup()` function is specific to p5 and is set to run only when you first run your page. If you refresh your page in your browser window, it will rerun from the start and the setup code will run again at the start.

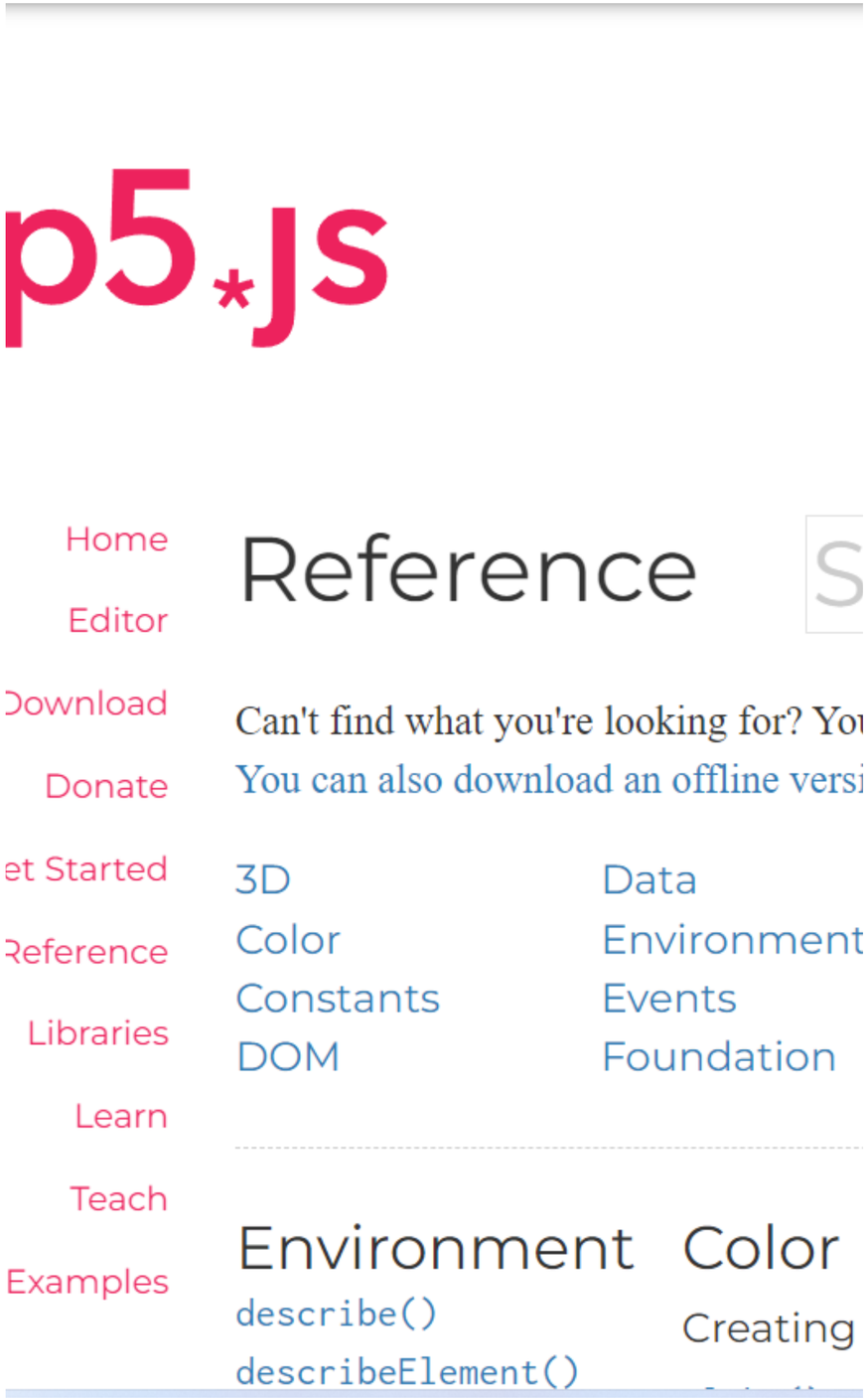
p5s `createCanvas()` function sets the size of your canvas.

`createCanvas(400,400);`
- will set the size of the canvas to 400 by 400 pixels.

`windowWidth | windowHeight`
If you want to size of your canvas to take up the whole page you can set the size to `windowWidth` and `windowHeight`. These are values that p5 knows and will automatically resize to how large your browser is. Notice that they are written in camelCase, the first letter of the second (and any consecutive words) are capitalised.

```
1   function setup() {
2     createCanvas(windowWidth, windowHeight);
3   }
```

Catchup: p5.js



Using the p5 reference page

The p5 library of functions are fully documented on: **https://p5js.org/reference/**

For each function there is some example code, a description and the syntax.

The function `rect()`, which we've seen before and draws a rectangle, has values x, y and w between the brackets. The x position, y position and the width (or size) of the rectangle.

Syntax

```
rect(x, y, w, [h], [tl], [tr], [br], [bl])
```

```
rect(x, y, w, h, [detailX], [detailY])
```

Parameters

x	Number: x-coordinate of the rectangle.
y	Number: y-coordinate of the rectangle.
w	Number: width of the rectangle.

Catchup: p5.js

Write your code in the order you want them to be run.

The order you write you code is important. In this sketch the background is drawn first. Then the rectangle is drawn. This means the rectangle appears on top of the background.

If you were to draw the rectangle first, then the background your rectangle would be hidden under the background colour and not be visible.

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

  background(153,50,204);

  // draw a rectangle + set the fill colou
  fill(255);
  rect(0, 0, 100);
}
```

p5.js

Home
Editor
Download
Donate
Get Started
Reference
Libraries
Learn
Teach
Examples
Contribute
Books

Reference

triangle()

Examples

```
triangle(30, 75,
describe('white t
right of canvas')
```

Description

Draws a triangle to the canvas. A
three points. The first two argume
two arguments specify the second

Catchup: p5.js

Drawing a triangle
In p5 there is a triangle function that lets you set the coordinates of each point. Between the brackets are 6 values, which is 3 pairs of coordinates.

You might find it easier to read the code if they are written on 3 lines:

```
triangle(
     x1, x1,
     x2, y2,
     x3, y3
);
```

If you write it like this pay attention to the commas, there isn't one after the final value.

Syntax

```
triangle(x1, y1, x2, y2, x3, y3)
```

Parameters

x1	Number: x-coordinate of the first point
y1	Number: y-coordinate of the first point
x2	Number: x-coordinate of the second point
y2	Number: y-coordinate of the second point
x3	Number: x-coordinate of the third point
y3	Number: y-coordinate of the third point

Catchup: p5.js

To draw a triangle on top of your rectangle write the code to draw it within the setup function but below the code for a rectangle.

```
10    // draw a triangle + set the colour
11    fill(0,255,0);
12    triangle(
13      0, 100,
14      100, 100,
15      50, 0
16    );
17  }
```

Catchup: p5.js

Sometimes it's easier to draw the triangle in one space so you can easily set the size, then use the `translate()` function to move it to the position you want.

To use the translate function you must place the triangle drawing code within `push()` and `pop()`.

Here the translate moves the triangle to be drawn in the centre of the canvas, the minus value is to take into account the size of the triangle:

```
11  push();
12    // draw the triangle at this position
13    translate((width/2) - 50, (height/2) - 50);
14    // draw a triangle + set the colour
15    fill(0,255,0);
16    triangle(
17      0, 100,
18      100, 100,
19      50, 0
20    );
21  pop();
```

Catchup: p5.js

```
1   function setup() {
2     createCanvas(400, 400);
3
4     background(153,50,204);
5
6     // draw a rectangle + set the fill colour
7     fill(255);
8     rect(0, 0, 100);
9
10    push();
11      // draw the triangle at this position
12      translate((width/2) - 50, (height/2) - 50);
13      fill(0,255,0);
14      // call the function to draw the triangle
15      drawTriangle();
16    pop();
17
18  }
19
```

Creating a custom function
That's quite a lot of code to look at in the setup() function. It may be easier to read if we created our own function to draw the triangle.

We can create function **below the setup() and draw() functions.**

function drawTriangle() {
        ..... triangle drawing code
}

To use the function, write it's name and brackets in the setup() function. Like the code on the right.

```
21  function drawTriangle() {
22    // draw a triangle + set the colour
23    triangle(
24      0, 100,
25      100, 100,
26      50, 0
27    );
28  }
```

Catchup: p5.js

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

  background(153,50,204);

  // set the shapes to be draw with no stroke (border)
  noStroke();

  // draw a rectangle + set the fill colour
  fill(255);
  rect(width/2 - 50, height/2 - 50, 100);

  push();
    // draw the triangle at this position
    translate((width/2) - 50, (height/2) + 50);
    fill(0,255,0);
    // call the function to draw the triangle
    drawTriangle();
  pop();

}

```

stroke() | noStroke()
To set our shape to have no border. We can set this using `noStroke()` function before we draw the shapes.

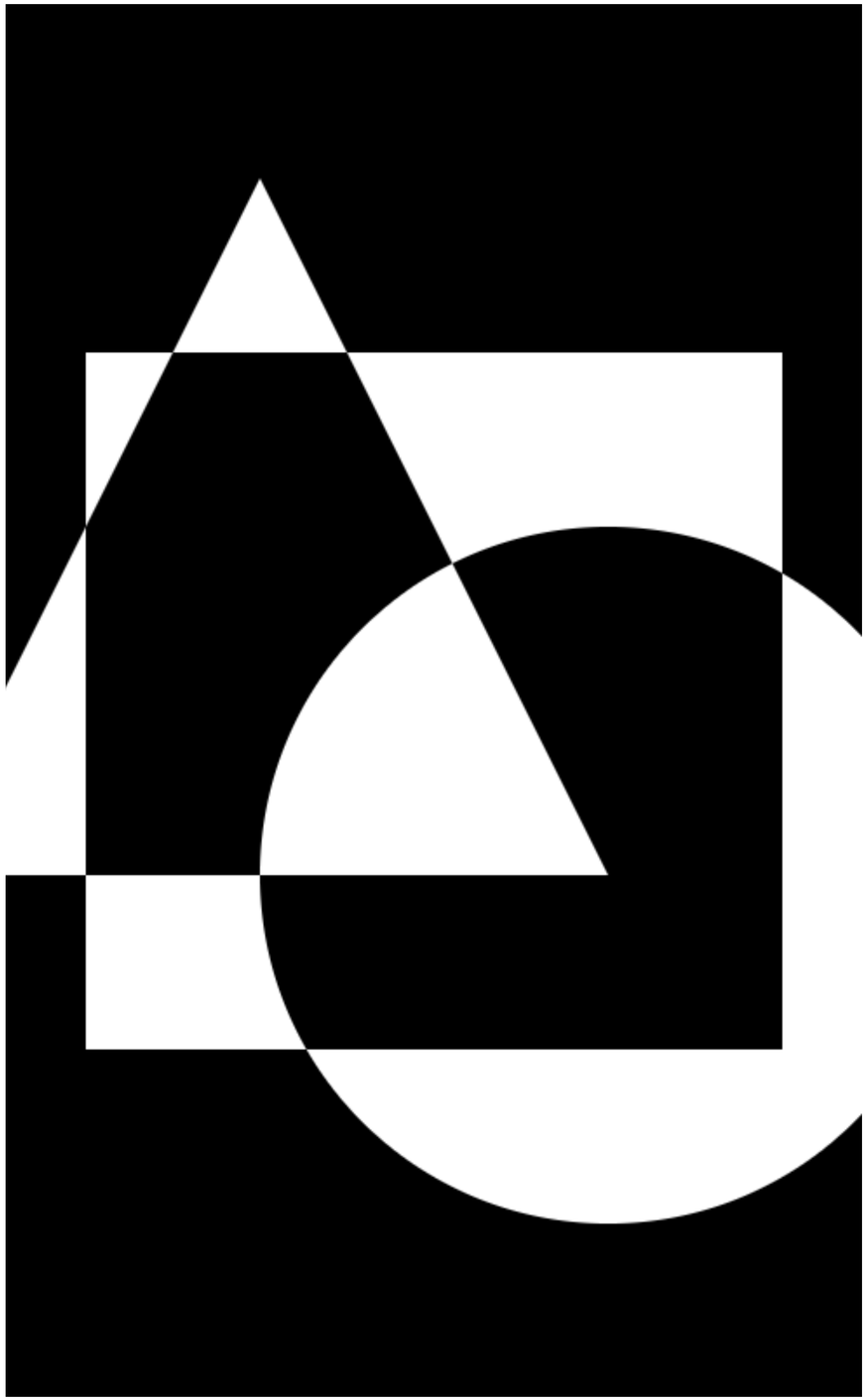
If you want to add the border, use the stroke() function. In the brackets put the rgb values you want. The following line would set the colour of the border to red.

```
stroke(255,0,0);
```

If you want to change the thickness of the border, use the function `strokeWeight()`.

To set the thickness of the border around the shapes to 10 pixels across:

```
strokeWeight(10);
```



Catchup: p5.js

blendMode()
To create effect on the right we can use the blendMode() function.

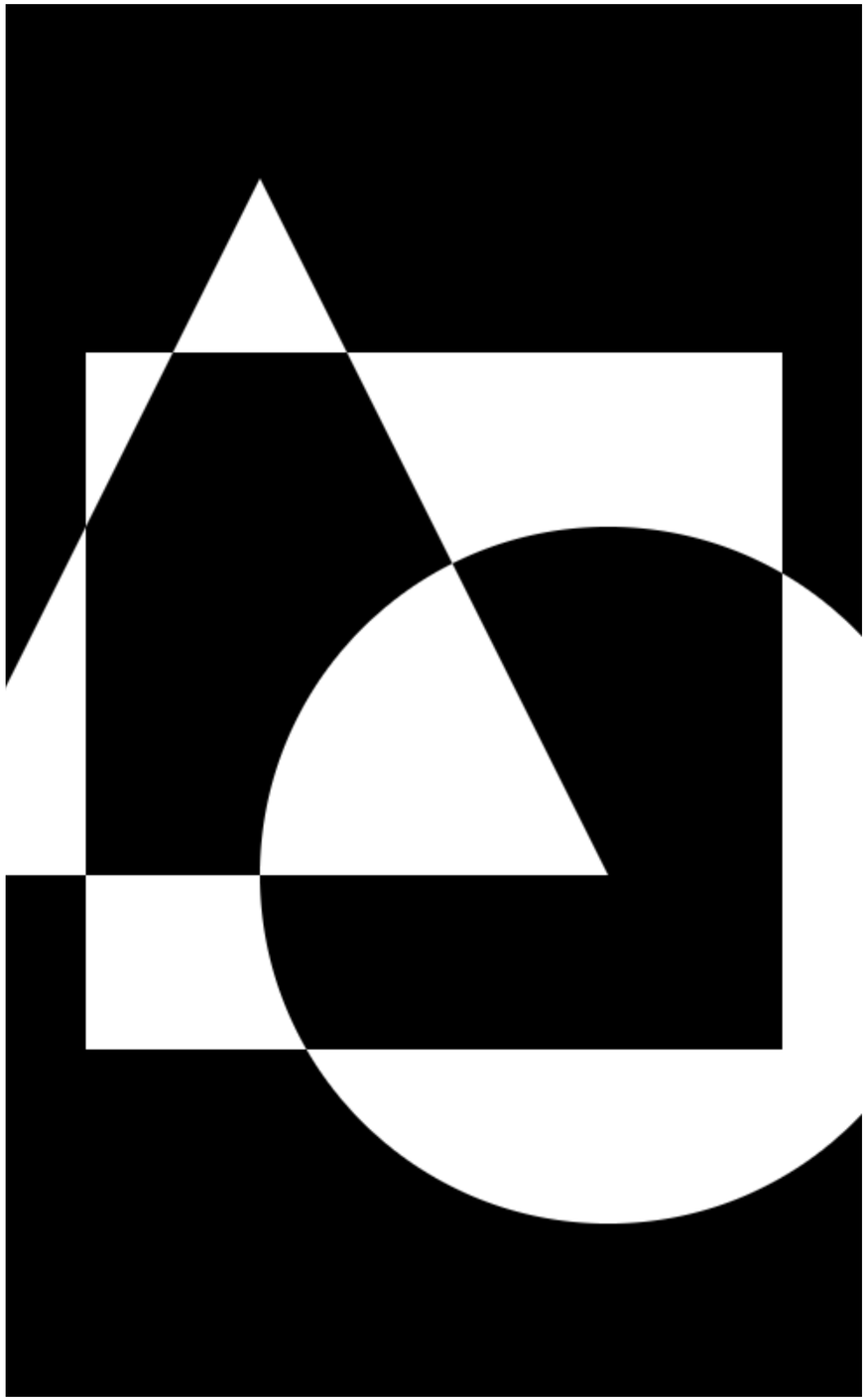
Step 1
Before drawing the background set the blend mode to BLEND:

```
blendMode(BLEND);
```

Step 2
Before drawing the shapes, but after drawing the background. Set the blend mode to DIFFERENCE.

```
blendMode(DIFFERENCE);
```

```
1   function setup() {
2     createCanvas(400, 400);
3     blendMode(BLEND);
4     background(0);
5
6     // set the shapes to be draw with no stroke (border)
7     noStroke();
8
9     blendMode(DIFFERENCE);
10    // draw a rectangle + set the fill colour
11    fill(255);
12    rect(width/2 - 50, height/2 - 50, 100);
13
14    push();
15      // draw the triangle at this position
16      translate((width/2) - 50, (height/2) + 50);
17      // call the function to draw the triangle
18      drawTriangle();
19    pop();
20
21  }
22
23
```



Catchup: p5.js

Coding Challenge

- Draw an ellipse after your triangle and rectangle.

- Tweak the colours to create a different effect.

    This sketch set the background to black:
    `background(0);`

    And the fill is set to white. This is set before the first shape is drawn:
    `fill(255);`

- Tweak the position values of your shapes so they sit nicely on top of each other with an offset.

```
<script>

let stayPut = true;
let time = 1;
let checkTime = "";

while(stayPut){
    checkTime = "hour "+time;
    if(checkTime=="hour 1"){
        console.log(checkTime);
    }else if(checkTime=="hour 2"){
        console.log(checkTime);
    }else if(checkTime=="hour 3"){
        console.log(checkTime+" -> Swap Rooms!!");
        stayPut = false;
    }
    time++;
}

</script>
```

Hours 2 & 3
(lecturers swap rooms between hours)

Room One:
Refresher of theories and histories

Room Two:
Make sure everyone can get going with the technical exercise and play with code.

Creative Coding & Creative Computing Frameworks

Refresher of theories and histories

Week 1: Destabilising the now

While its roots lie in technical, academic and military histories that go back to the mid-twentieth century, platform capitalism as we live it now has arisen in a remarkably short period of time: for the significant companies, in less than a couple of decades (Facebook, b. 2004).

How does this process extract/hoard planetary resources – of labour, precious metals, energy?

How does it shape our cities and ecosystems? Who is its default subject?

There are many superficial examples in our direct neighbourhoods where this form of 'tech' development has visible effects (Getir/Gorillaz, cashless cafés, the turn to subscription bikes and scooters)?

But these changes betray an underlying violence – systems restructure social composition, expel the poor, and frame 'safety' and 'security' only for the rich.

Critical histories destabilise this 'now'

Lecture Title: Week 4 - Refresher of theories and histories

Top Image: Contextual absences at the time Mute magazine started publishing (1994)
Bottom Image: The default subject of WIRED magazine, the Silicon Valley house mag

Creative Coding & Creative Computing Frameworks

Refresher of theories and histories

Capitalism is crisis

There are remarkable symmetries in the peaks and troughs of capitalist development and the popular imaginaries of software and the internet.

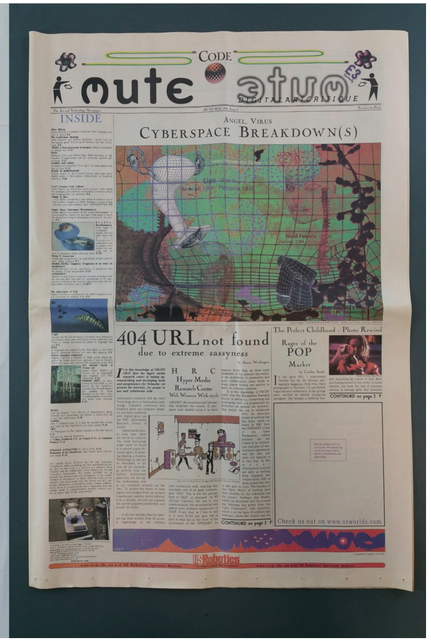
In the context of the Russian-Ukranian war Palantir CEO, Alex Karp, speaks of "software + heroism slaying the giant", and the world always underestimating "kinetic + software + heroism". Karp's questions are instructive about a general operational mindset:

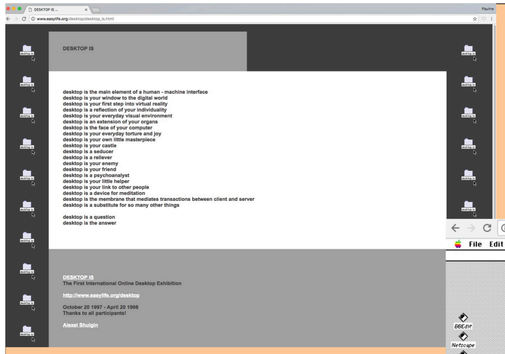
"When you use information technology, who's moving, what's moving, under what conditions? Can you identify people from space, can you unmask people who are trying to mask themselves? What is the action of the battlefield, where do you put your assets, where are their biggest assets?" (CNBC, 22 September 2022)

As capitalist crises recur more frequently, how do we see digital forms adapt? Who is developing them, and to what end? What platforms are most powerful, who do they strengthen, and how?

Lecture Title: Week 4 - Refresher of theories and histories

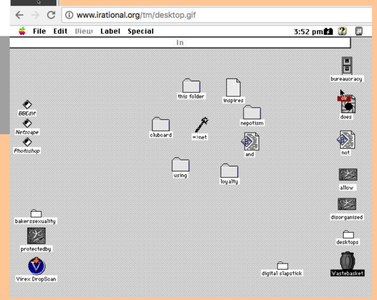
Top Image: National newspapers on 16 September 1992, when the pound was forced to exit the European Exchange Rate Mechanism (ERM)
Bottom Image: early representations of 'cyberspace', in which human beings are telematically enhanced, or given god-like powers





Alexei Shulgin, *Desktop is…* (1998)

De-naturalise the GUI



Rachel Baker (above)
& Garnet Hertz (left)

Heath Bunting
Kings X phone-in
(1994)

@ kings x

phone in

RELEASE

During the day of Friday 5th August 1994
the telephone booth area behind the destination board
at kings X British Rail station will be borrowed
and used for a temporary cybercafe.

It would be good to concentrate activity around 18:00 GMT,
but play as you will.

TELEPHONE Nos.

```
0171 278 2207  ..........................  0171 387 1736
0171 278 2208  ..........................  0171 387 1756
0171 837 6028  ..........................  0171 387 1823
0171 837 5193  ..........................  0171 278 2179
0171 837 6417  ..........................  0171 278 2163
0171 278 4290  ..........................  0171 278 2083
0171 837 1034  ..........................  0171 387 1362
0171 837 7959  ..........................  0171 278 2017
0171 837 1644  ..........................  0171 387 1569
0171 837 7234  ..........................  0171 387 1526
0171 837 1481  ..........................  0171 387 1587
0171 837 0867  ..........................  0171 837 0298
0171 278 7259  ..........................  0171 837 0399
0171 278 2502  ..........................  0171 837 1768
0171 278 2501  ..........................  0171 387 1398
0171 278 2275  ..........................  0171 837 3758
0171 278 2217  ..........................  0171 837 0933
0171 278 2260  ..........................  0171 837 0499
```

Please do any combination of the following:

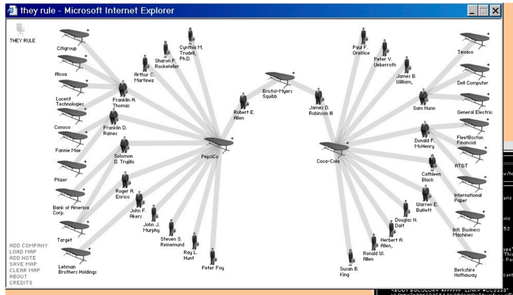
(1) call no./nos. and let the phone ring a short while and then hang up
(2) call these nos. in some kind of pattern
(the nos. are listed as a floor plan of the booth)
(3) call and have a chat with an expectant or unexpectant person
(4) go to Kings X station watch public reaction/answer the phones and chat
(5) do something different

This event will be publicised worldwide

I will write a report stating that:
(1) no body rang
(2) a massive techno crowd assembled and danced to the sound of ringing telephones
(3) something unexpected happened

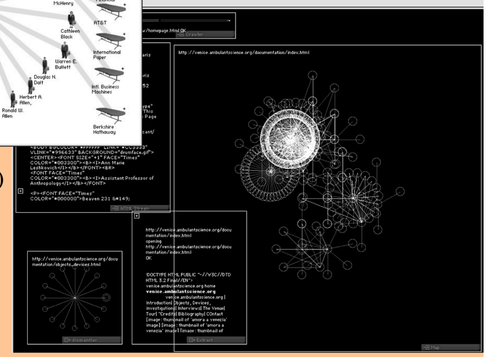
No refreshments will be provided/please bring pack lunch

heath@cybercafe.org



Josh On, They Rule (2001, 2004, 2011)
http://www.theyrule.net

Picture invisible structures



I/O/D (Simon Pope, Matthew Fuller, Colin Green),
I/O/D 4: The Webstalker (1997)
http://bak.spc.org/iod

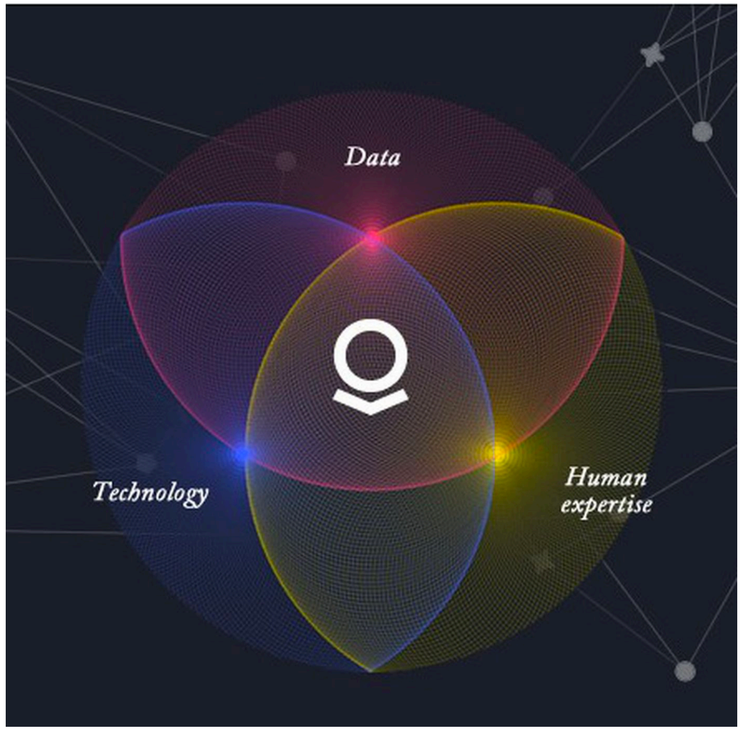
Week 1: artists

- **Mute magazine (1994)**
  Point back to history via format-appropriation (newspapers: *The Daily Courant* (18thc), *The Financial Times*)

- **Heath Bunting (1994)**
  Appropriates for play ready-made & 'found' systems and infrastructures of the network society (public phones terminals in 'kings x phone in')

- **Net.art's 'heroic phase' ('90s)**
  Upsets GUI users' habitual expectations of the 'desktop' and 'browser' / 'navigator' (both of whose names demonstrate the bureaucratic and colonial histories of computing)

- **Picture invisible structures**
  Immediate sceptical approaches to the way the browser functions to obscure deeper power dynamics (and to map these)



Creative Coding & Creative Computing Frameworks

Refresher of theories and histories

Week 2: Code and reality

As a form of writing, code has unique qualities and capacities due to its status as a form of dynamic instruction: it sets in motion chains of action and consequence, and through these can have an impact on material reality.

This places software – built on code – in a complicated arena with regard to the critical tools we might use to understand and engage with it.

Software studies argues it can't be anything other than interdisciplinary.

Software is an amalgam and hybrid, drawing on the representational and mobilising powers of visual and textual symbolic languages, as well as the information-processing speed and physical architectures of hardware and computer chips. For human usage, these are translated, disguised, softened and made to mirror everyday objects and activities so that we more easily and smoothly incorporate them into life.

Lecture Title: Week 4 - Refresher of theories and histories

Top Image: René Magritte, *The Treachery of Images* (1929)
Bottom Image: Palantir diagram modelling software's workings

Creative Coding & Creative Computing Frameworks

Refresher of theories and histories

The possible and the probable

But what is the vision of reality that is imposed on the world through these languages, processes, and tools?

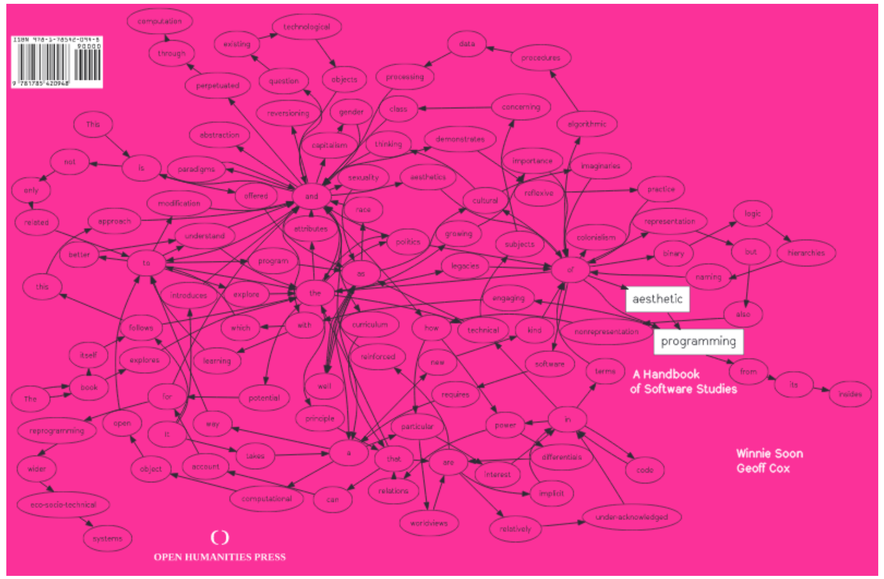
The deep relationship software has to phenomena of prediction and control makes its mediation of social and economic life in an age of political and climatological turbulence entirely logical.

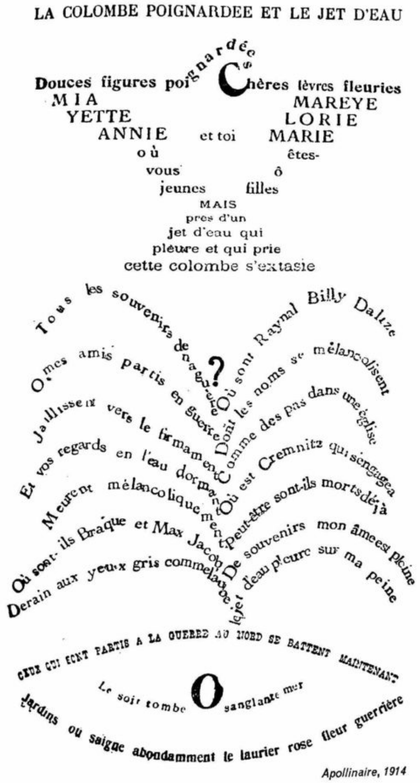
Annette says:

It’s no longer about governing what is, about judging, punishing, and controlling past behaviours, but about governing uncertainty. The mass processing of data is about taming uncertainty. Algorithmic governmentality has a much larger target in its sights: the excess of the possible over the probable. It aims to reduce the possible to the probable by affecting behaviours through warning rather than through prohibition or obligation.

Lecture Title: Week 4 - Refresher of theories and histories

Top Image: Wizard Saruman and the Palantir all-seeing globe (*The Lord Of The Rings: The Return of the Kings*, Dir. Peter Jackson, 2003)
Middle Image: Tom Cruise in *Minority Report* (2002), Dir. Steven Spielberg – based on the Philip K. Dick novel, *The Minority Report* (1956)
Bottom Image: Anti-government protests in Valparaiso, Chile, 2019



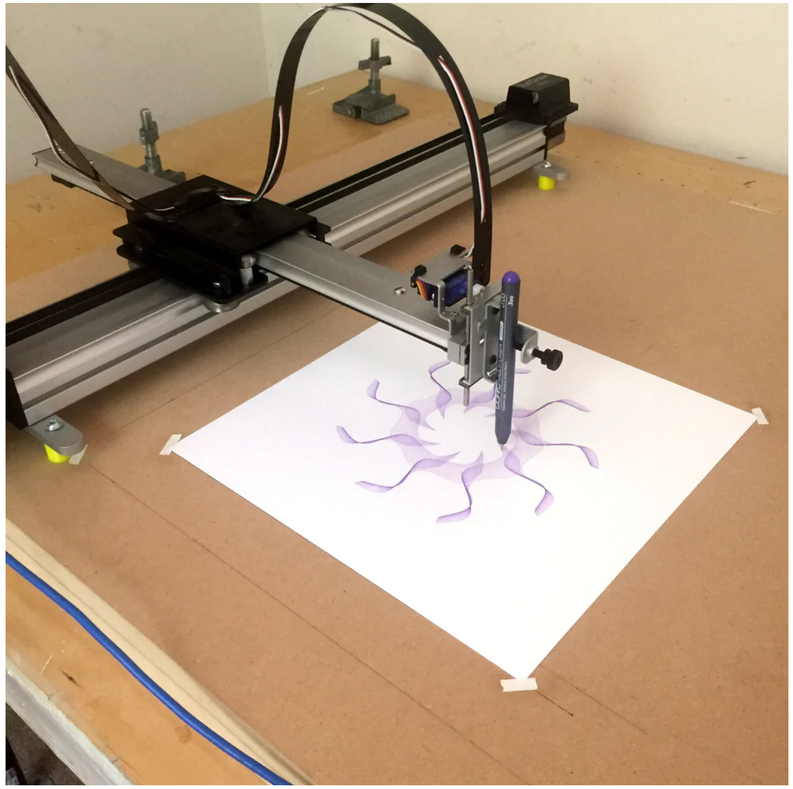


Week 2: critical sources

- ***Aesthetic Programming* (2021)**
  Winnie Soon's and Geoff Cox's 'instruction manual' proudly asserts that a new curriculum is necessary to develop a critical literacy of software and forms of computational thought. *Aesthetic Programming* acts out in its very form the connection of theory and practice

- ***Programmed Visions* (2011)**
  One of many essential books by Wendy Chun, this emphasiss the degree to which software institutes ignorance as much as illumination and knowledge; and the unique place it occupies with regards to metaphor and symbolism, utility and obstacle

- **Poetry, performance, collage**
  We looked to art forms whose ambitions are – also – to 'mobilise' energy, people, resources, using language, splice, instruction (Guillaume Apollinaire, Carolee Schneemann, Susan Howe, Wangechi Mutu)



Creative Coding & Creative Computing Frameworks

Refresher of theories and histories

Week 3: Drawing 1 – arrays, conditionals, loops

Array: in the history of the West, classification was an integral part of the enlightenment project, forming one of the primary planks of colonial thought

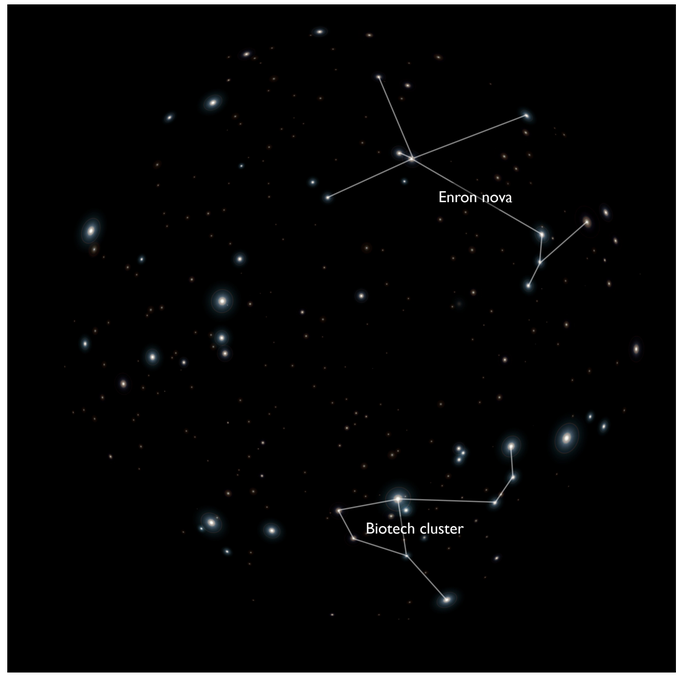
As we also learn on Thursdays in ML, the relationship between exterior and interior, appearance and essence, was taken to be knowable, consistent, representable. Standards of normality and deviance were devised in biology, culture, society

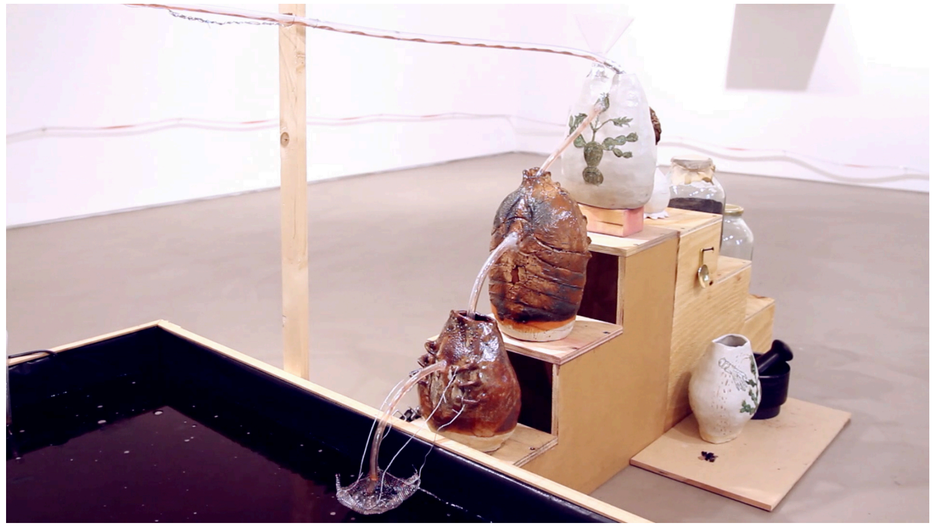
Conditionals: a state acting as the condition for an action or event (if / then)

Loop: continually repeated instructions aiming for a certain condition or state. They are repeated, their effects checked, and there is either an exit route, a prescribed duration after which things end (or go back to the beginning), or an infinite loop

Lecture Title: Week 4 - Refresher of theories and histories

Top Image: Michael Landy, Break Down (2001)
Bottom Image: pen plotter looping





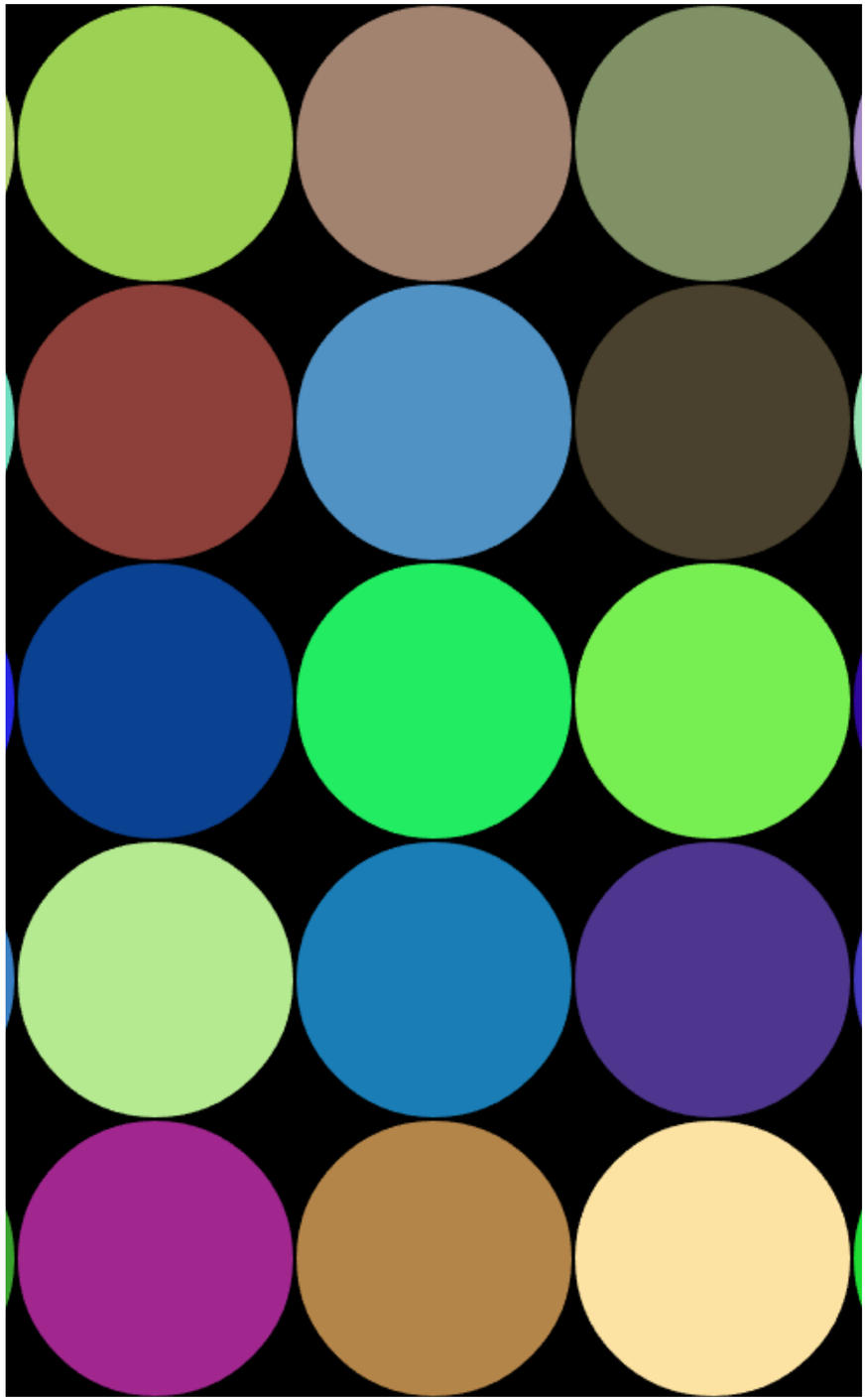
Week 3: artists

- **Black Shoals Stock Market Planetarium (2000-2016)**
  Lisa Autogena and Joshua Portway use live financial data to catalyse self-learning behaviours and show how we blur nature, culture, biology

- **A Body Reduced to Brilliant Colour (2016)**
  Candice Lin stages the corrosion of 'civilisation' by small, seemingly insignificant natural and bodily processes

- **The Way Things Go (1986-87)**
  In the climate of TINA (Thatcher's There Is No Alternative), Fischli & Weiss' event chains picture the tenuousness of cause and effect

- **Dead The Ends (2014)**
  Seymour's bargain-basement rendition of *La Jetée* made using animated GIFs of other time-travel films, tropes, characters



Technical Exercise: p5 Loops Refresher and a bit of Random

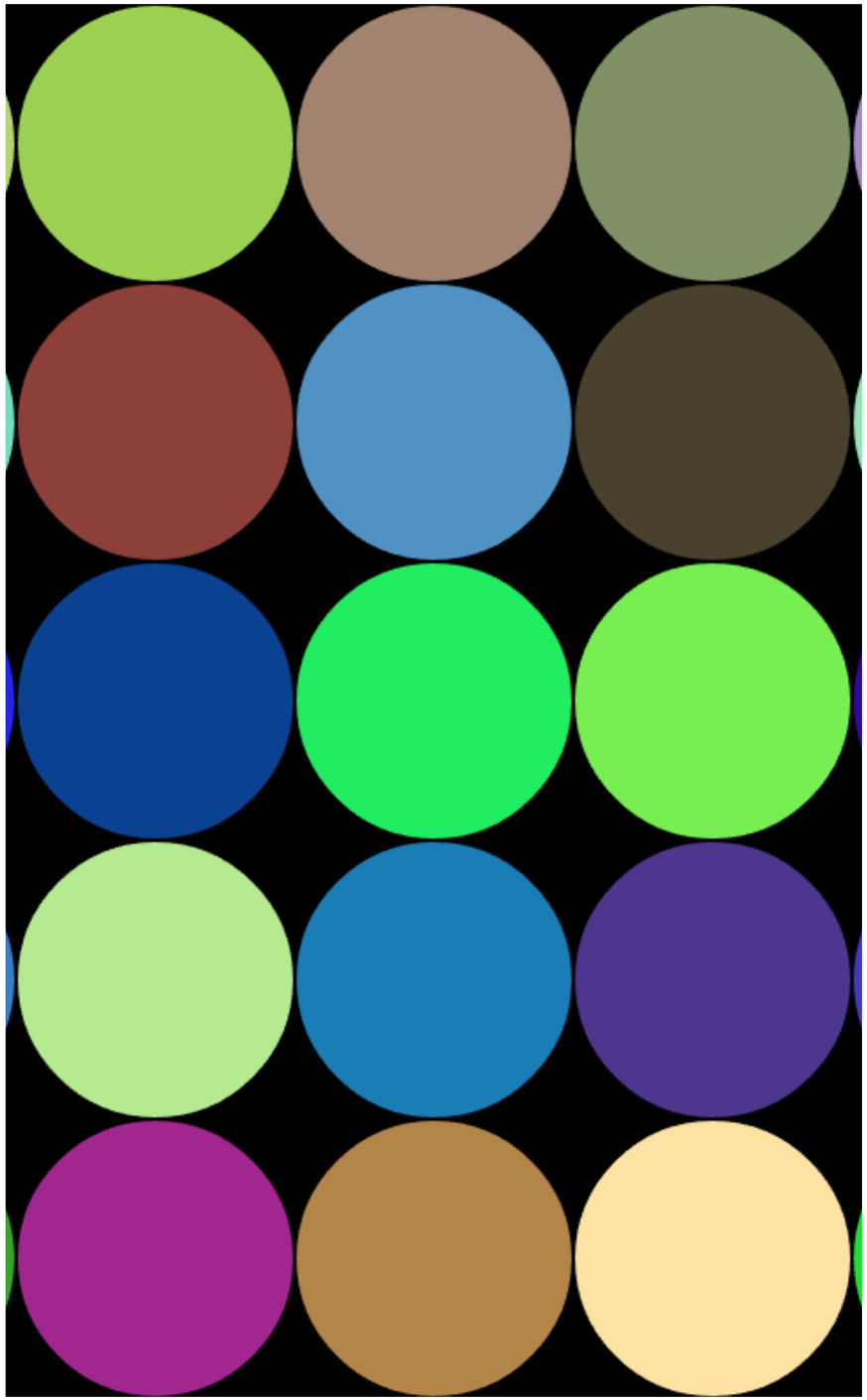
In this exercise we are returning to our grid of circles. We will write the code out and explore the random function in p5.

Let's recap!

The first step is to create 4 variables:
-A counter to track which circle is being drawn
-step determines the size of the circle.
-x and y are the circles position.

Declare variables at the top of the sketch. use the `let` keyword, then the variables name. Then set the initial value after the equals sign. Then a semi colon at the end of the line.

```
1   let counter = 0; // variable to track which circle is being drawn
2   let step = 80; // variable to store the size of the circle
3   let x = step/2; // variable to store the x position of the circle
4   let y = step/2; // variable to store the y position of the circle
```

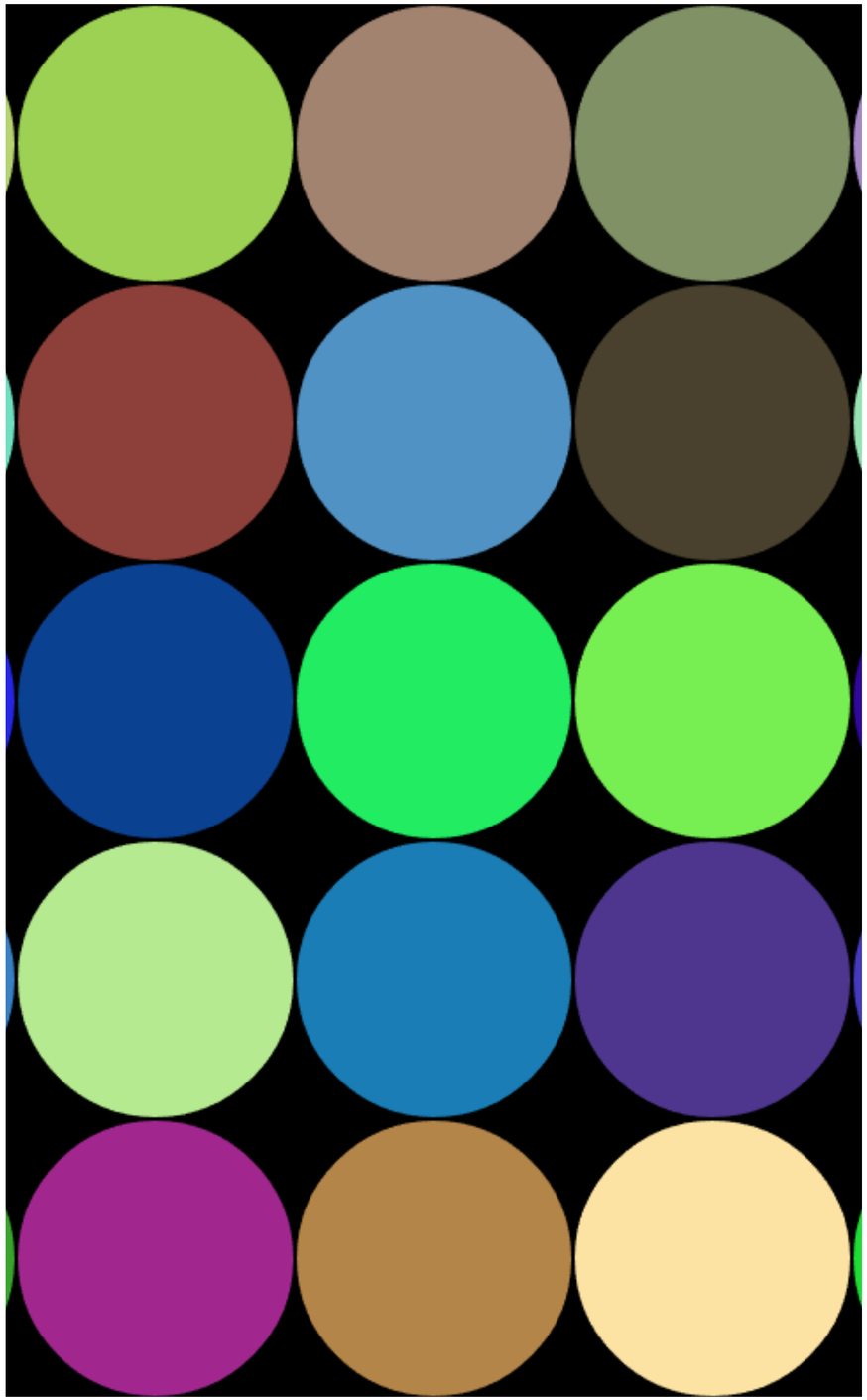


Technical Exercise: p5 Loops Refresher and a bit of Random

Set up the canvas and the background in the `setup()` function. Notice that this is all we will do in the setup function so this is set at the beginning.

```
6   /* The code in the setup function runs only once when the sketch starts */
7   function setup(){
8
9     createCanvas(400,400); // p5 function to create a canvas of the set size
10    background(0); // draws the background a set colour
11
12  }
```

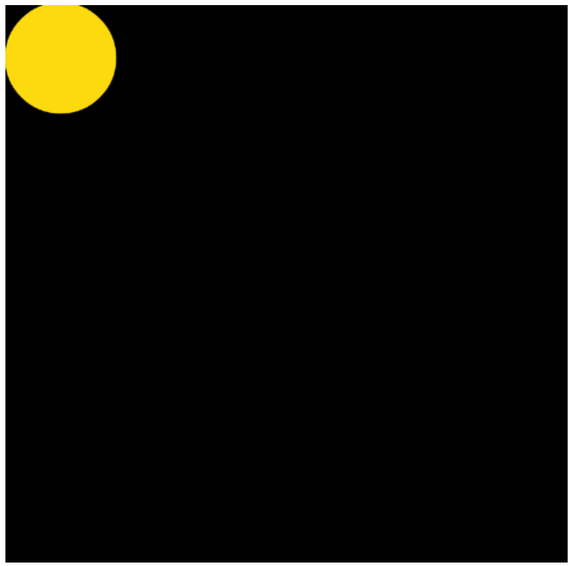
We will use the `draw()` function to draw the circles as these will not be static, but changing over time. Which means they need to be drawn in the `draw()` function as this is run on every frame.

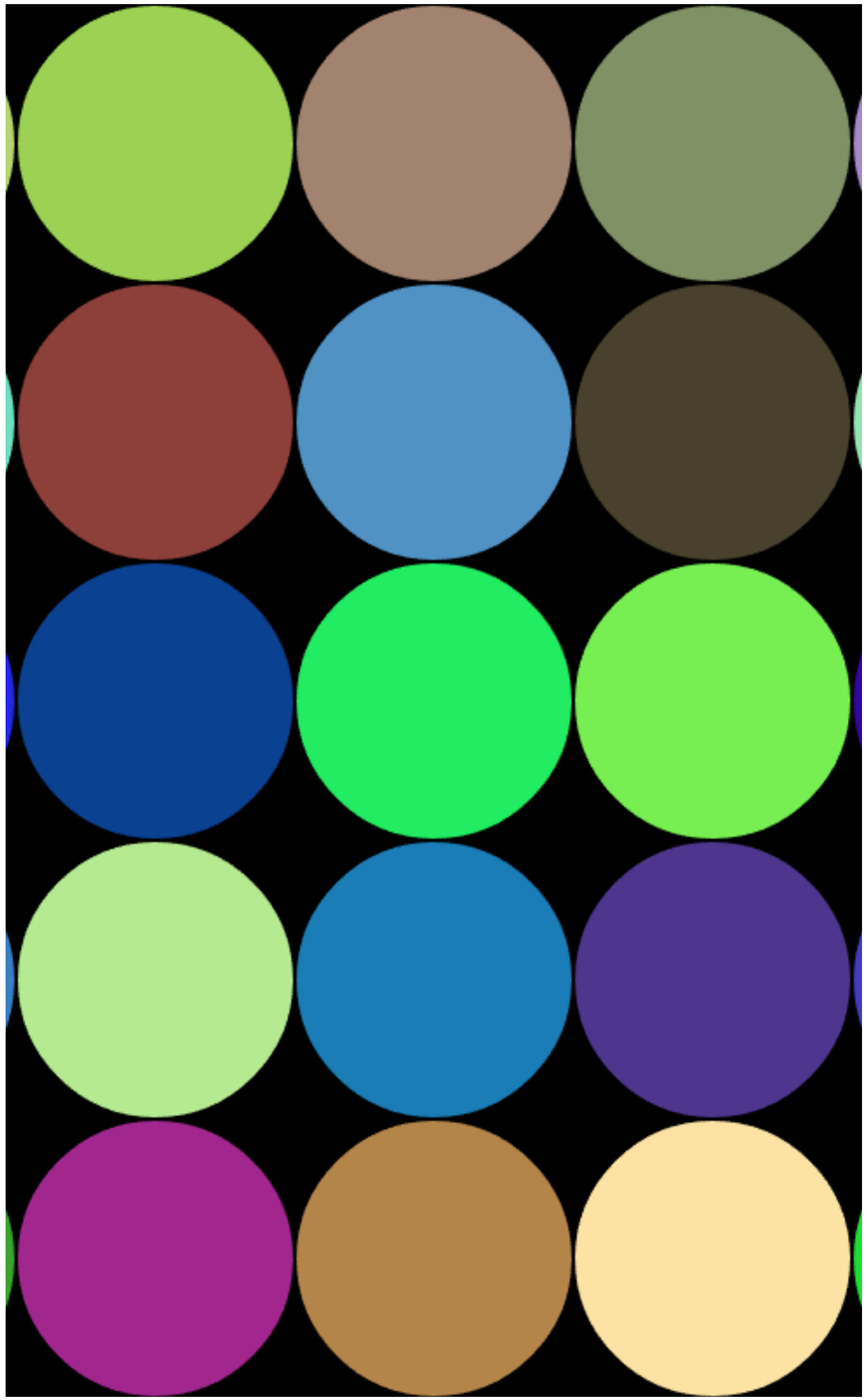


Technical Exercise: p5 Loops Refresher and a bit of Random

Next we will start to write the draw function. This will be below the setup function. Inside we will draw the first circle and set it's colour using the fill function: The circles position is set to the variables x and y we created at the top of the sketch and the size is set the variable step.

```
function draw() {
     fill(253, 218, 13);
     ellipse(x,y,step,step);
}
```





Technical Exercise: p5 Loops Refresher and a bit of Random

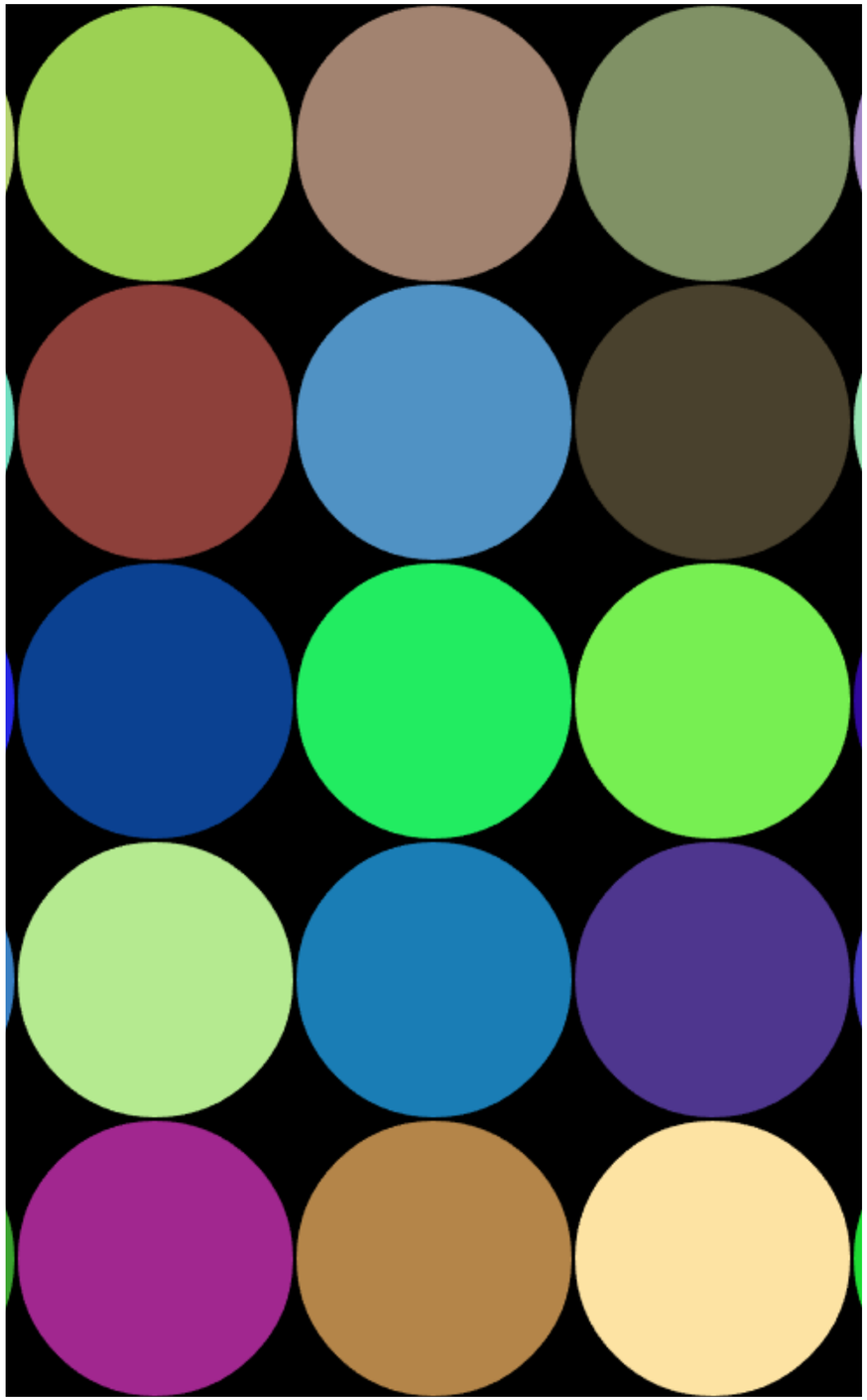
Next we will write our while loop so that all the circles are drawn. Here is the top line:

```
while (counter < (width/step)*(height/step)) {
```

The loop will stop when the condition in the bracket is false. We have set the counter to start at 0, will will add one to the value on every loop. Will will add `counter++` to the bottom of the loop. After the loop we will reset counter to 0 to start the process again.

Although we are drawing more circles using the loop we cannot see them because they are on top of each other.

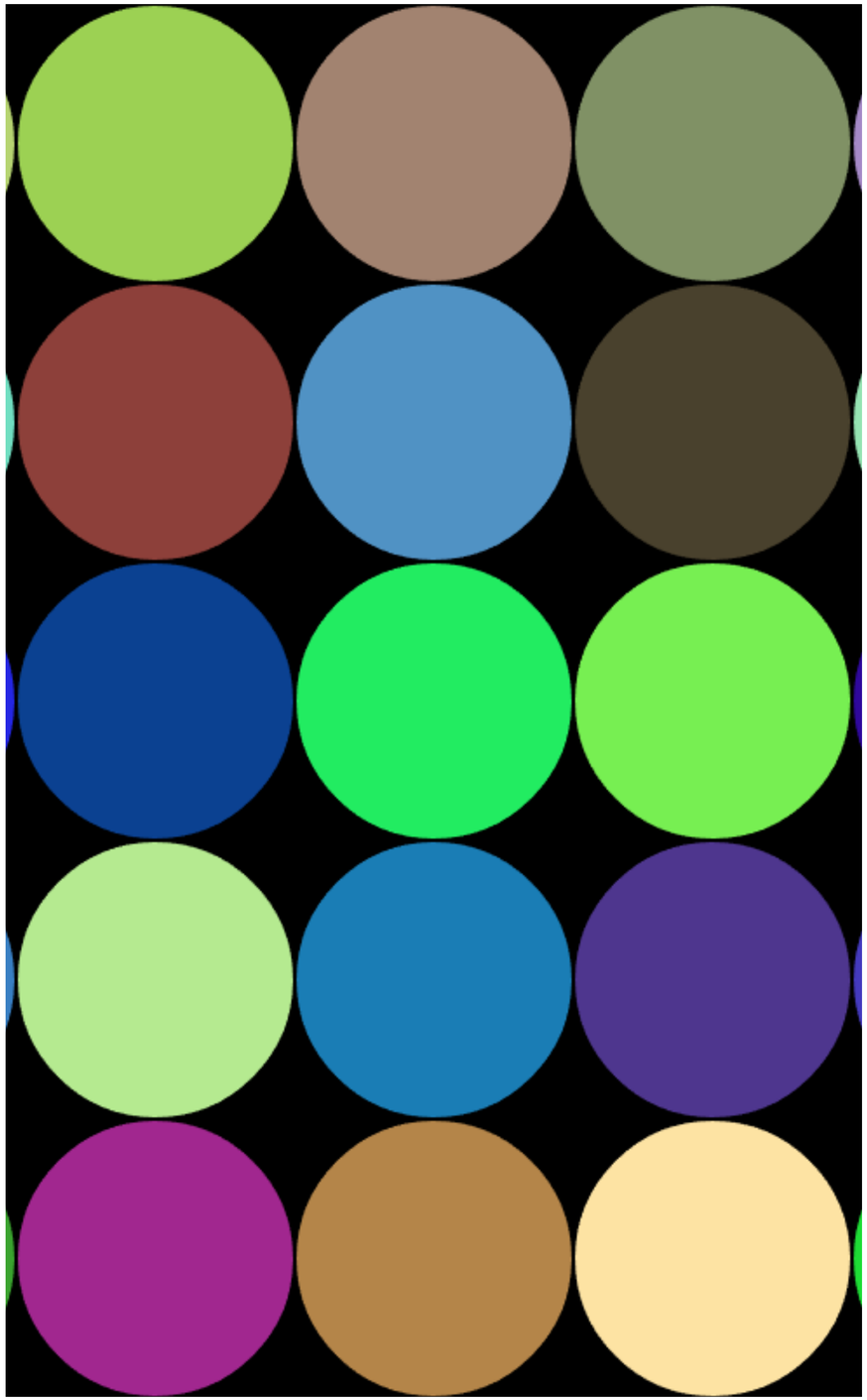
```
13  /* The code in the draw function is run on every frame until closed */
14  function draw() {
15
16    // this code loops until all the circles are drawn
17    while (counter < (width / step) * (height / step)) {
18      fill(253, 218, 13);
19      ellipse(x, y, step, step);
20      // increment the counter by one every loop
21      counter++;
22    }
23    // after the loop reset the counter
24    counter = 0;
25  }
26
```



Technical Exercise: p5 Loops Refresher and a bit of Random

The next step is to update the position of each circle. First we can update the x position. On every loop we want to add step to the x position. We can write this after we have drawn the circle on line 21.
x += step;

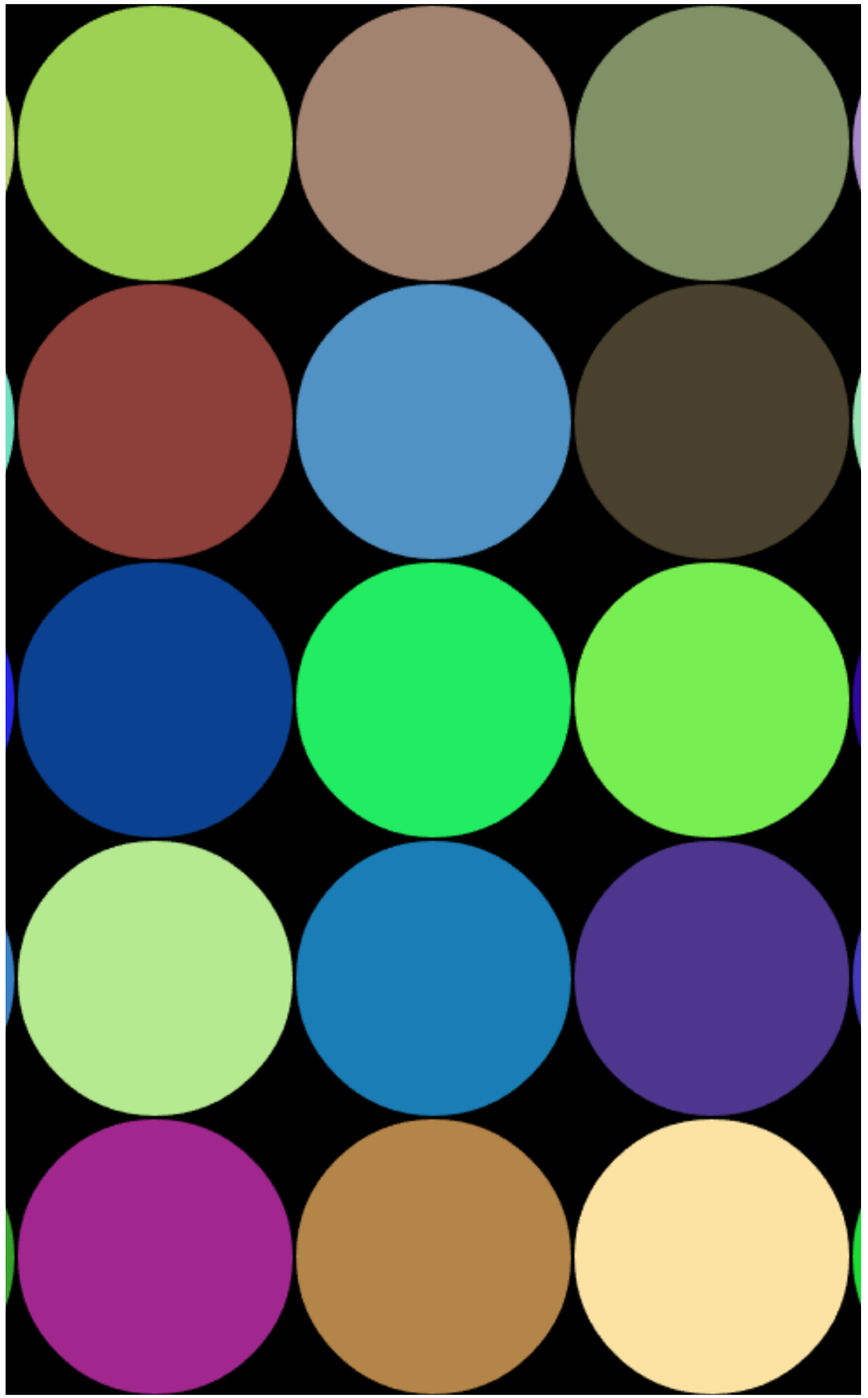
```
13  /* The code in the draw function is run on every frame until closed */
14  function draw() {
15
16    // this code loops until all the circles are drawn
17    while (counter < (width / step) * (height / step)) {
18      fill(253, 218, 13);
19      ellipse(x, y, step, step);
20
21      x += step; // updates the x position once the circle is drawn
22
23      // increment the counter by one every loop
24      counter++;
25    }
26    // after the loop reset the counter
27    counter = 0;
28  }
```



Technical Exercise: p5 Loops Refresher and a bit of Random

We only need to update the y position at the end of the line. We can check by using a conditional statement to check if the x variable is more than the width of the canvas: When it is higher we update the x position variable and add step to the y position

```
23    // the code in this conditional statement is only run once the x position moves
24    if(x >= width){
25      x = step/2; // resets the x position to the left of the canvas to start the n
26      y += step; // updates the y position to move onto the next row of circles
27    }
```



Technical Exercise: p5 Loops Refresher and a bit of Random

Coding Challenge Part #1: Random function
The random function allows you to either pick a random number between two numbers or choice a random item in an array. For this challenge we will pick a random number between 2 values.

The code `random(0,255)` will choose a random number between 0 and 255.

Challenge: Use it in your code to set a random colour for each circle. Use the reference as a guide.

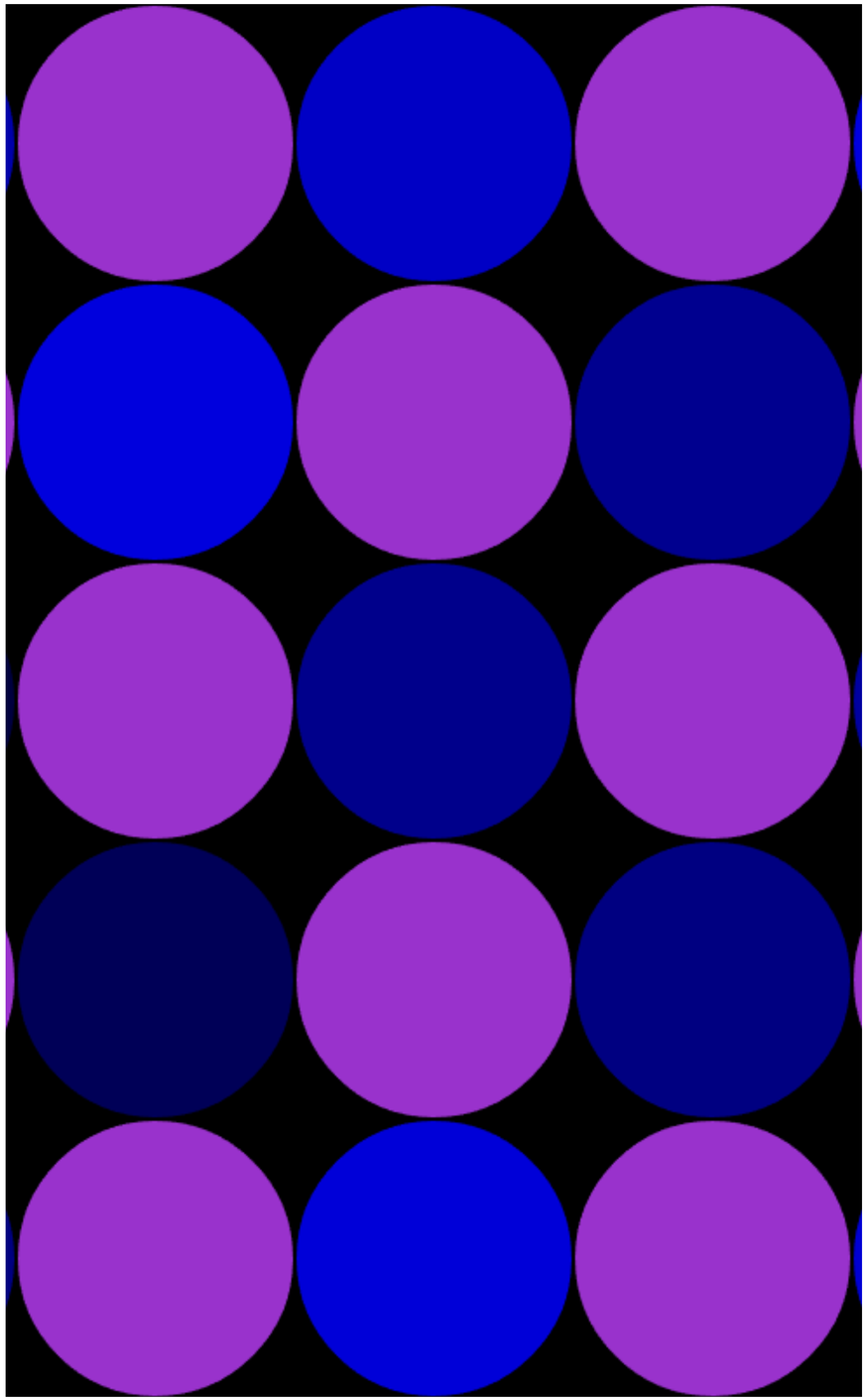
Syntax

```
random([min], [max])
```

```
random(choices)
```

Parameters

min	Number: the lower bound (inclusive) (Optional)
max	Number: the upper bound (exclusive) (Optional)
choices	Array: the array to choose from



Technical Exercise: p5 Loops Refresher and a bit of Random

Coding Challenge Part #2: Modulo

Draw a checkerboard grid of shapes and colours of your choice using a while loops and conditional statements.

Hint: Use a conditional statement with a modulo (%) to determine whether your counter is odd or even.

```
if (number%2 === 0) {
     Console.log('even');
}
```

```
<script>

let stayPut = true;
let time = 1;
let checkTime = "";

while(stayPut){
    checkTime = "hour "+time;
    if(checkTime=="hour 1"){
        console.log(checkTime);
    }else if(checkTime=="hour 2"){
        console.log(checkTime);
    }else if(checkTime=="hour 3"){
        console.log(checkTime+" -> Swap Rooms!!");
        stayPut = false;
    }
    time++;
}

</script>
```

Hour 4

Develop your 'random' exercise AND/OR go over past concepts and exercises - lecturers will be around to help.