Creative Computation for Visual Communication Design

Coding Workshop 2.2.

Assignment I

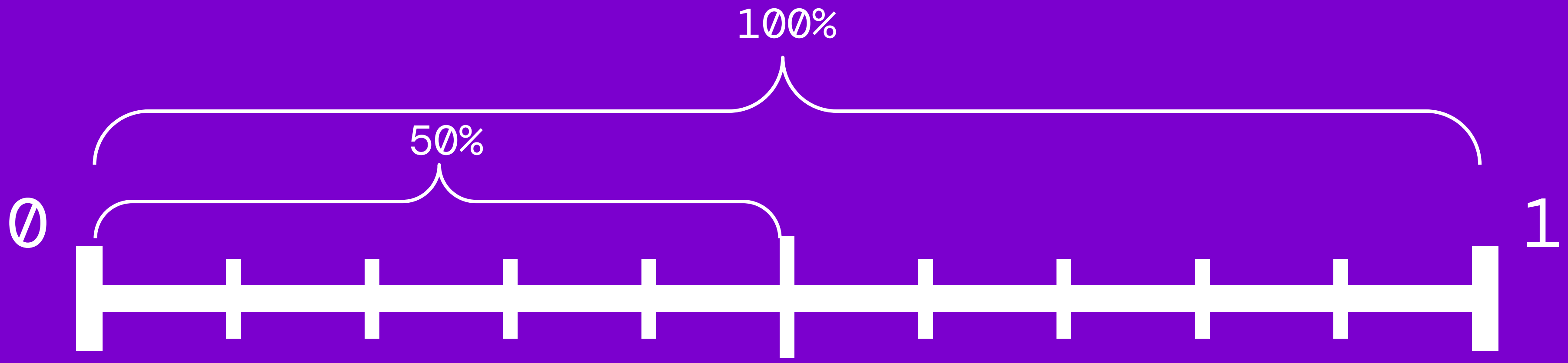
- Using width, height and colour() before setup
  - The sketch doesn't know the values for canvas width and height before the canvas has been created!
  - Also colour variables can't be created with colour() before the canvas is created!
- Width vs. windowWidth
  - Width of the canvas vs. width of the browser window
  - If the canvas is set to be the size of the window, then these two values are equal!

Randomness & Probability

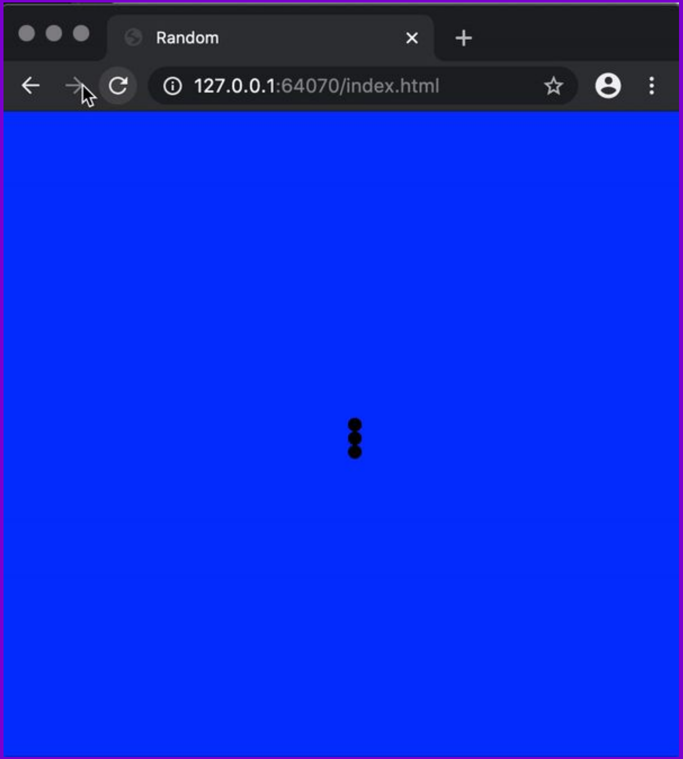
HOW CAN WE MAKE DIFFERENT THINGS HAPPEN WITH DIFFERENT PROBABILITY?

Random & Probability

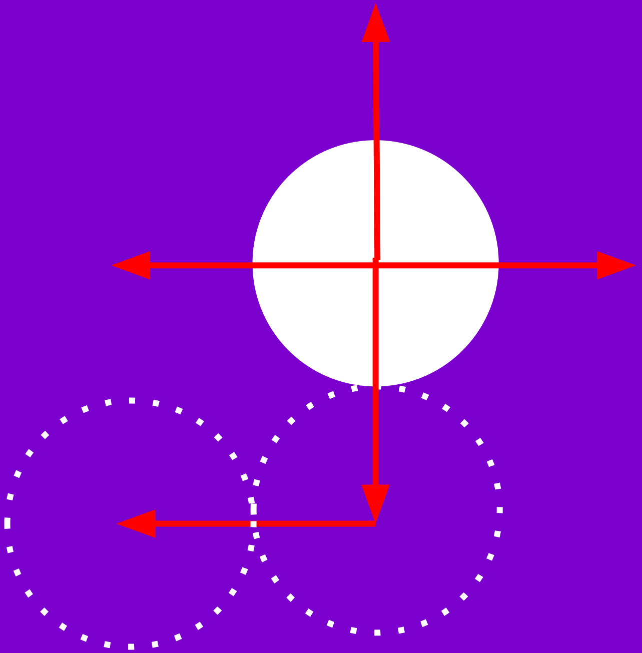
- Random numbers are uniformly distributed
  - `random()` produces all numbers between 0 and 1 with same probability
- We can use `random()` to create probability distributions
  - Doing different things with different likelihood



Exercise 1: Random walker



Exercise 1: Random walker

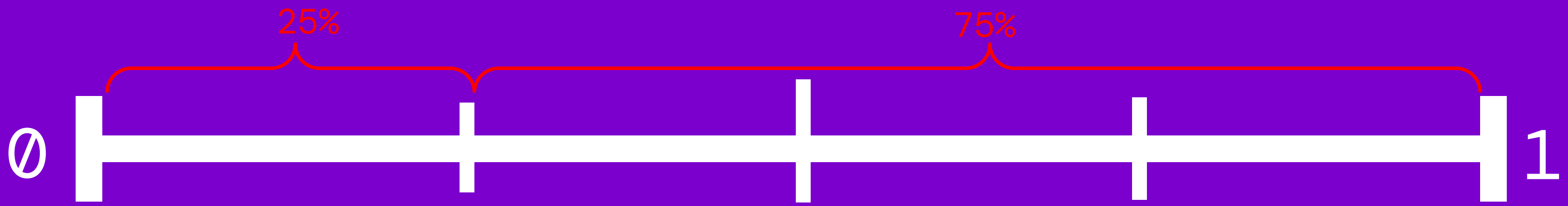


1. Start from somewhere on the canvas
2. Randomly choose a direction to move
   a. RIGHT
   b. LEFT
   c. UP
   d. DOWN
3. Move to the new location
4. Repeat steps 2-4

Random & Probability

- We can use a **conditional statement** to perform different events with different probabilities

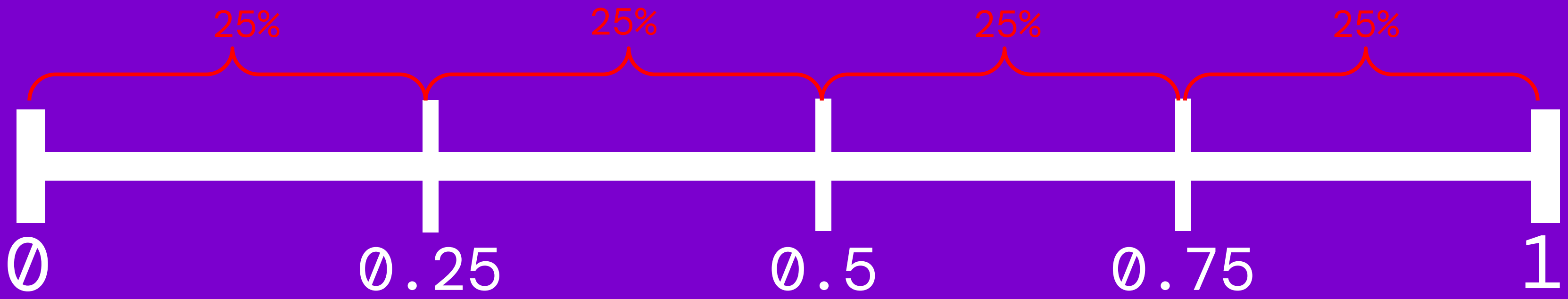
```
var ran = random(); //random number between 0 and 1
if(ran < 0.25) {//do something with 25% chance}
else {//do something with 75% chance}
```



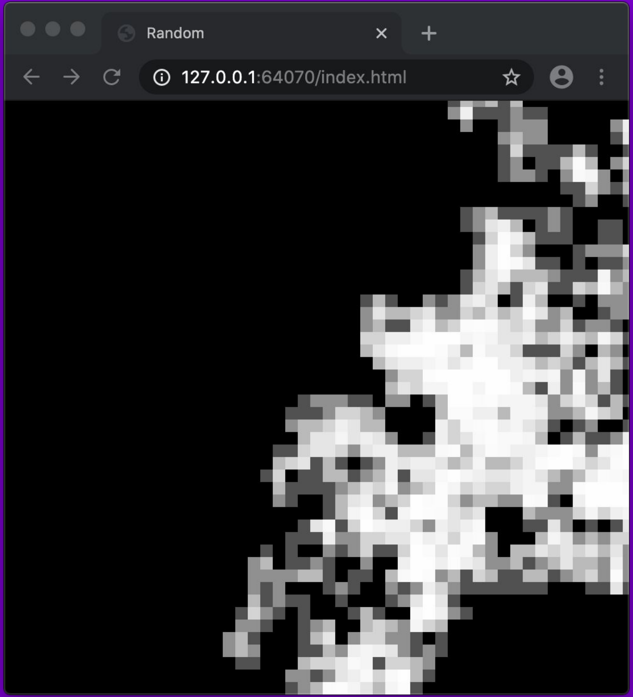
Random & Probability

- Multiple different events with different probabilities can be stacked using the else-if structure

```
var ran = random(); // number between 0 and 1
if(ran < 0.25) { // 25% chance }
else if(ran < 0.5) { // 25% chance  }
else if(ran < 0.75) { // 25% chance  }
else {// 25% chance }
```

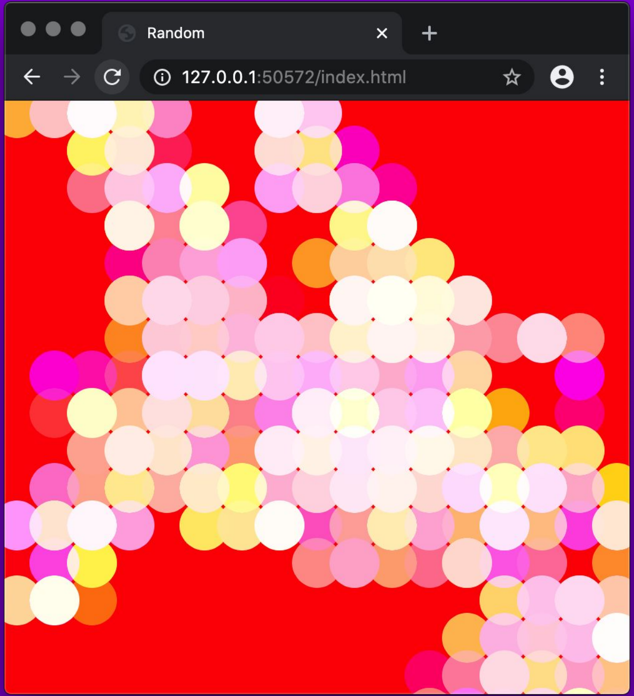


Variations



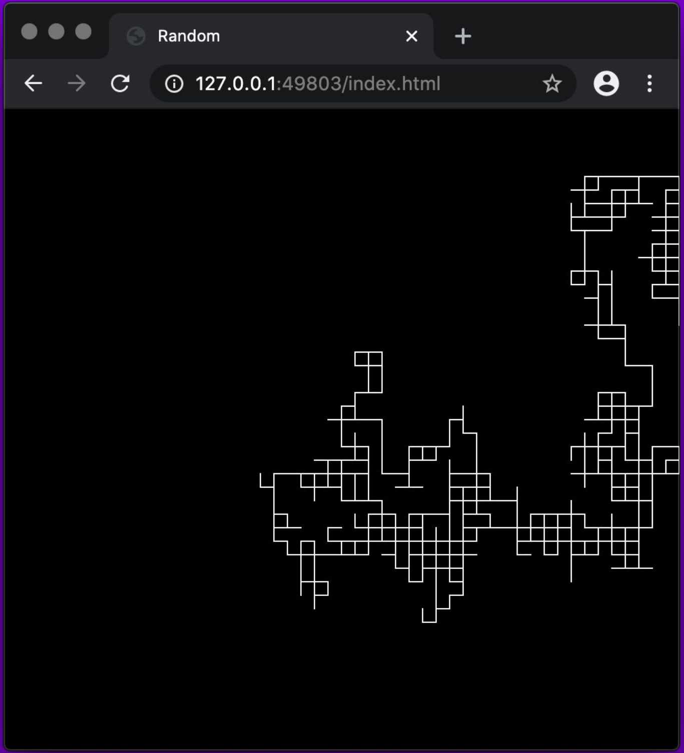
- Make a biased walker: Try changing the probability distribution so the walker prefers one direction
- Draw different shapes for the random walker
- Play with size, colour and opacity

Variations



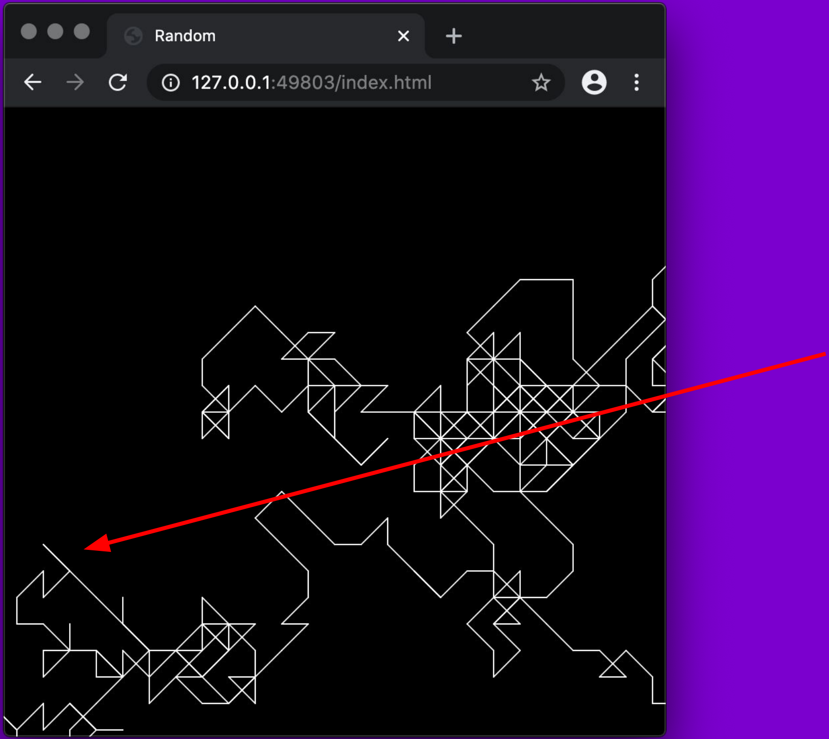
- Randomize the colours!
- Try different blend modes: check the blendMode function
- Vary the distance between steps and the size of the walker

Variations: Slightly advanced



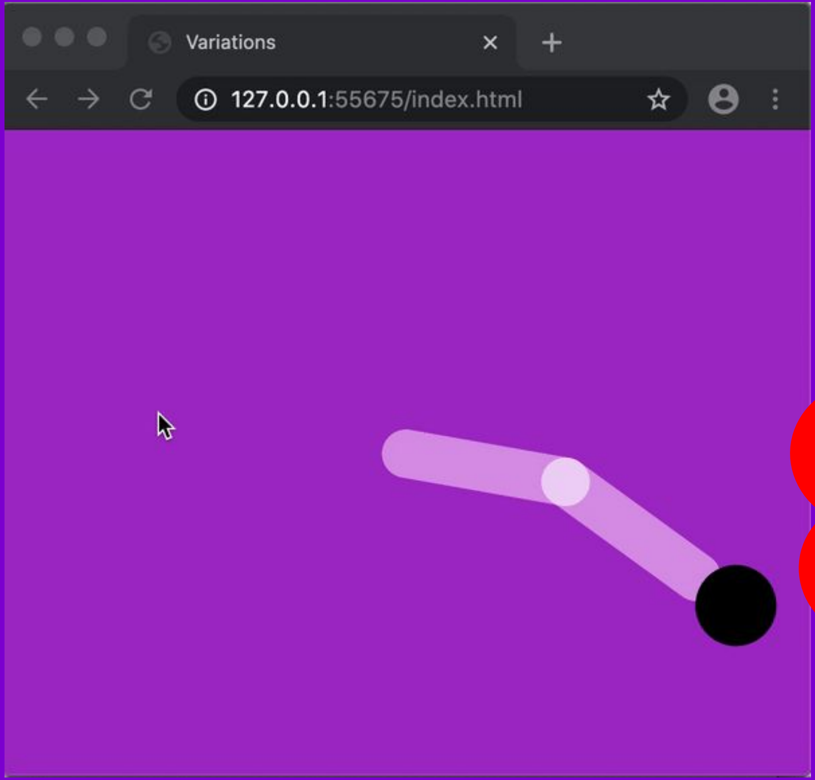
- Draw lines instead of shapes!
    - Pay attention how you call the line() function and update the coordinate variables

Variations: ADVANCED



- Make the walker move also diagonally!
- Make the walker avoid going out of bounds
- How to avoid the walker from going back where it came from?

Transformations



PROBLEM:
How to rotate shapes?

Transformations

- In drawing software like Illustrator, moving, rotating and scaling objects is easy
  - Transformations affect individual shapes
- With code you are drawing the entire frame at once
  - Transformations affect all the following shapes
  - Transformations are reset when frame is refreshed

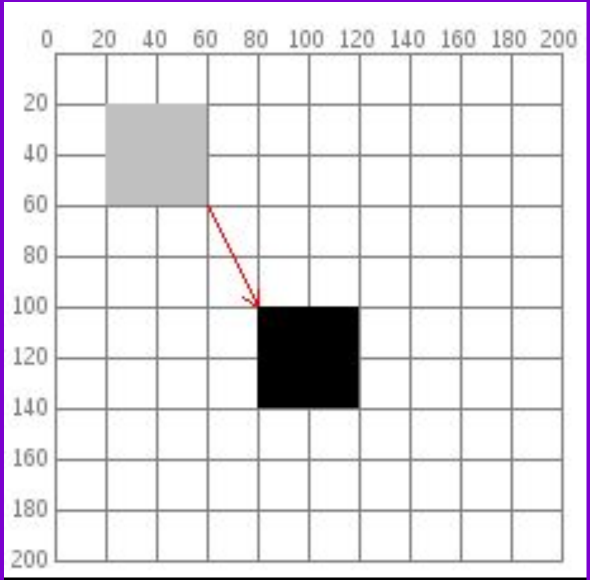
```
translate(x,y);
rotate(rad); //default is radians
scale(x,y); //decimal percentage
```

Transformations affect the entire coordinate system!

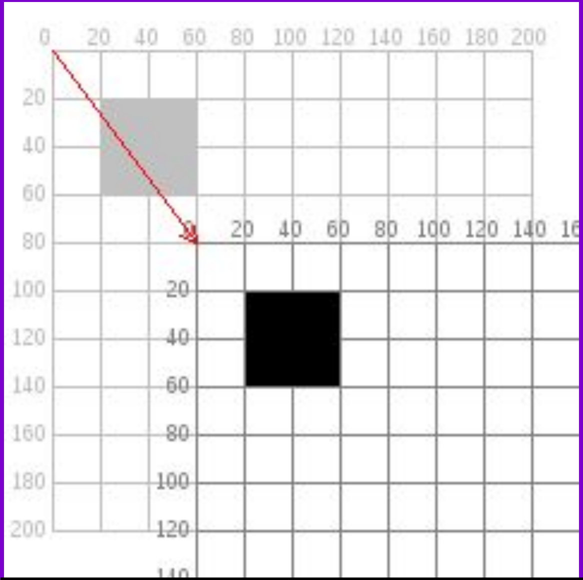
Transformations: Translate

- Moves the point of origin

```
translate(x,y);
```



```
rect(20,20,40);
rect(80,100,40);
```

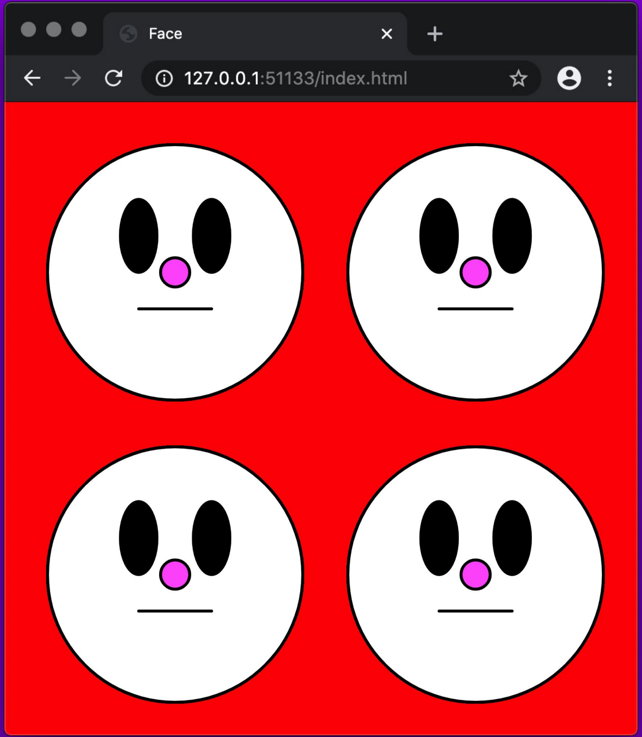


```
rect(20,20,40);
translate(60,80);
rect(20,20,40);
```

Transformations accumulate!

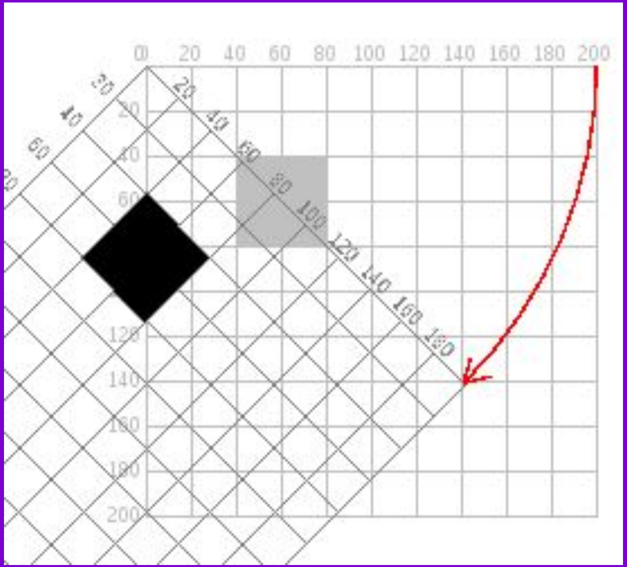
Transformations: Translate

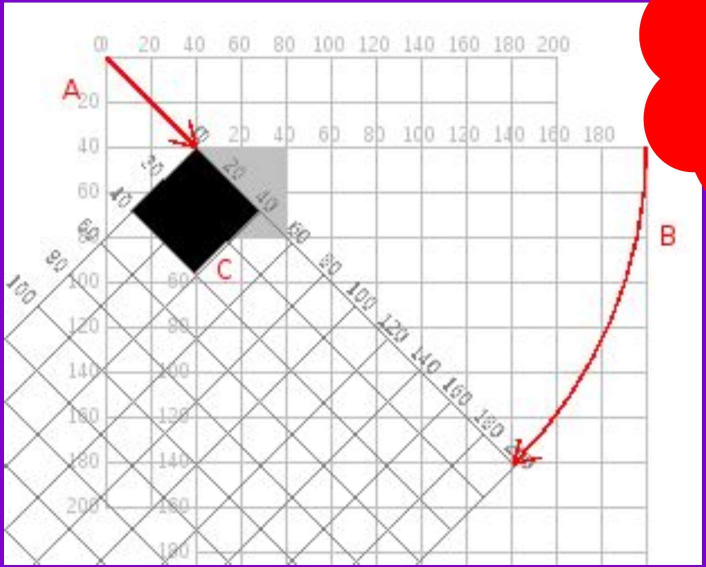
- Translating is useful when drawing the same complicated shape in different locations
  - Define the coordinates in relation to the origin, then move the origin and repeat drawing
  - "Grouping shapes"
- Translating is also necessary when rotating shapes!



Transformations: Rotate

- Rotates the coordinate system **around the point of origin**
- Default unit is radians
  - Use angleMode(DEGREES)



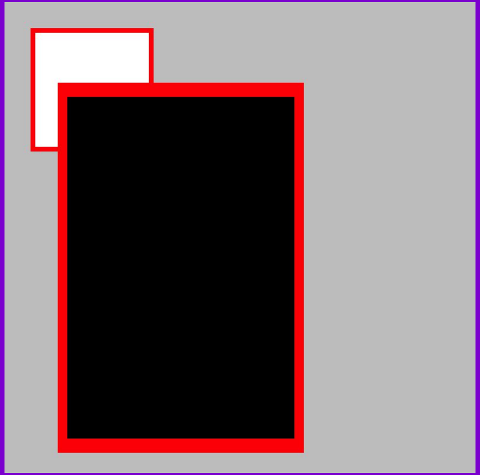


First translate, then rotate!

Transformations: Scale

- Scales the coordinate system in **relation to the origin**
  - X and Y axis can be scaled individually
- Unit is decimal percentage
  - scale(2) increases the size 200%

```
function draw(){
    strokeWeight(4);
    stroke(255,0,0);
    fill(255);
    rect(25,25,100,100);
    scale(2,3);
    fill(0);
    rect(25,25,100,100);
}
```



Affects also strokeWeight!

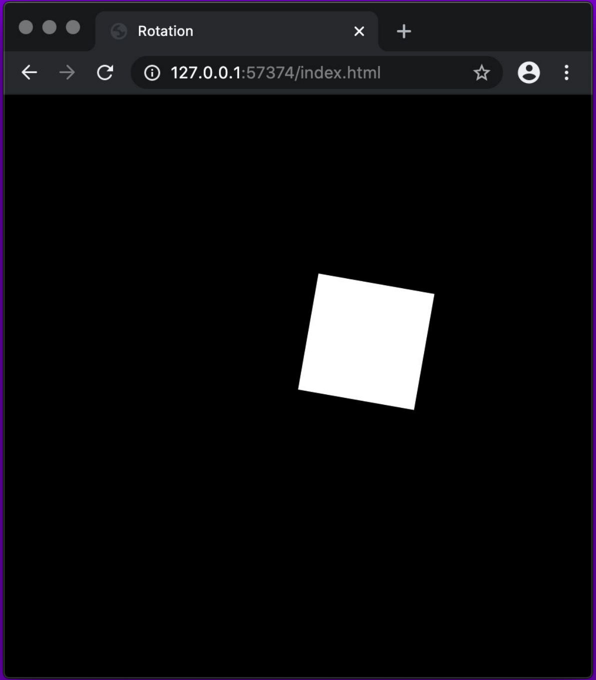
Transformations: push() & pop()

- Transformations are cumulative and affect all the following drawing commands
  - Drawing styles eg. fill() also affect all following drawing commands
- We can save and restore transformations and styles with push() and pop() functions
- Push() and pop() can be nested for more complex effects
  - Indenting your code makes it more legible!

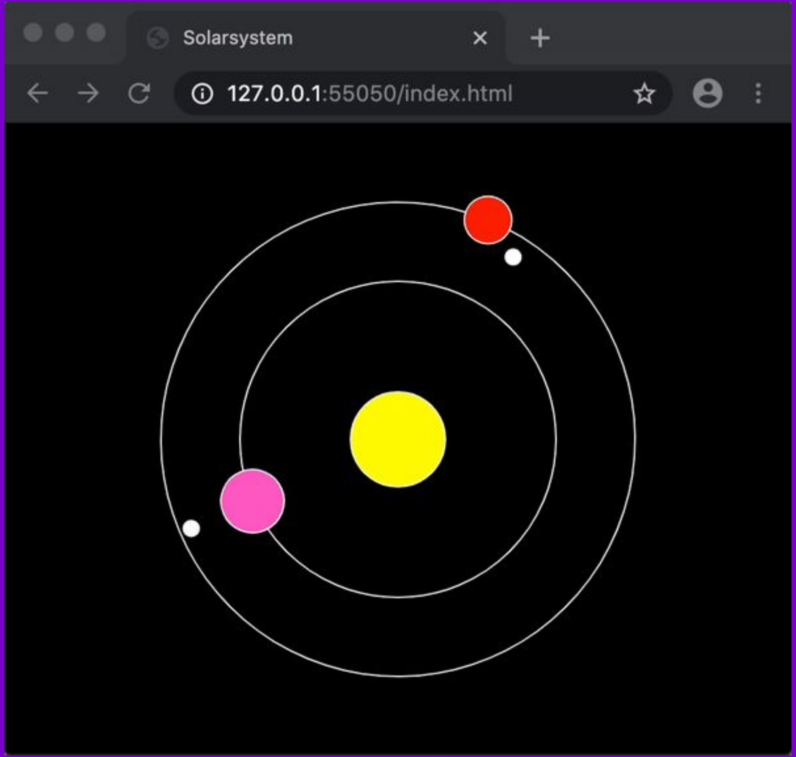
```
push(); //start new drawing state
    fill(0); //change fill to black
    translate(100,100); //move origin
    rect(0,0,50,50); //draw rectangle at new origin
pop(); //restore the original drawing state and style
```

Always use push and pop together!

Exercise 2: Simple rotation

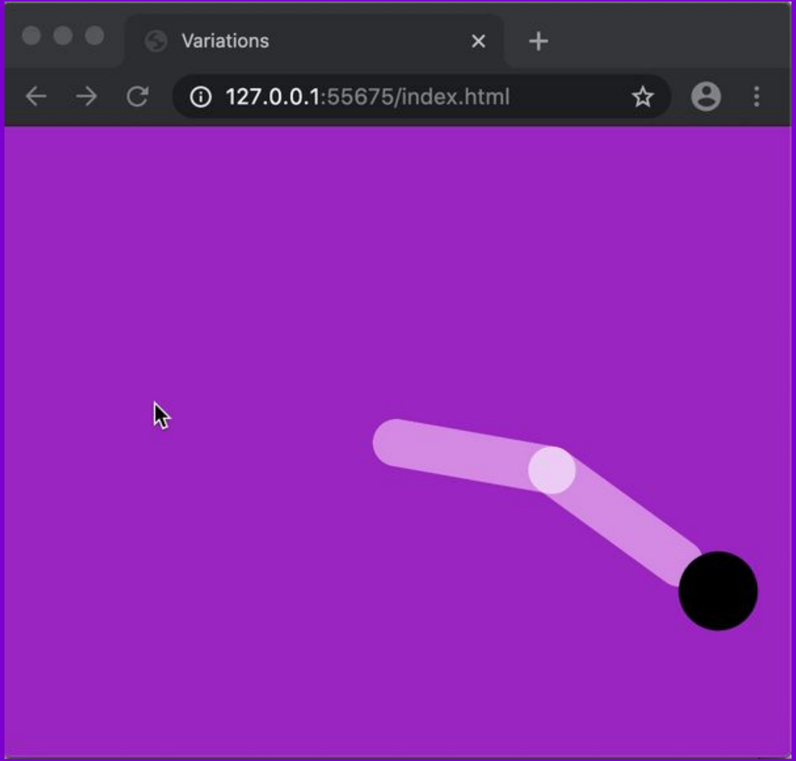


Exercise 3: Solarsystem



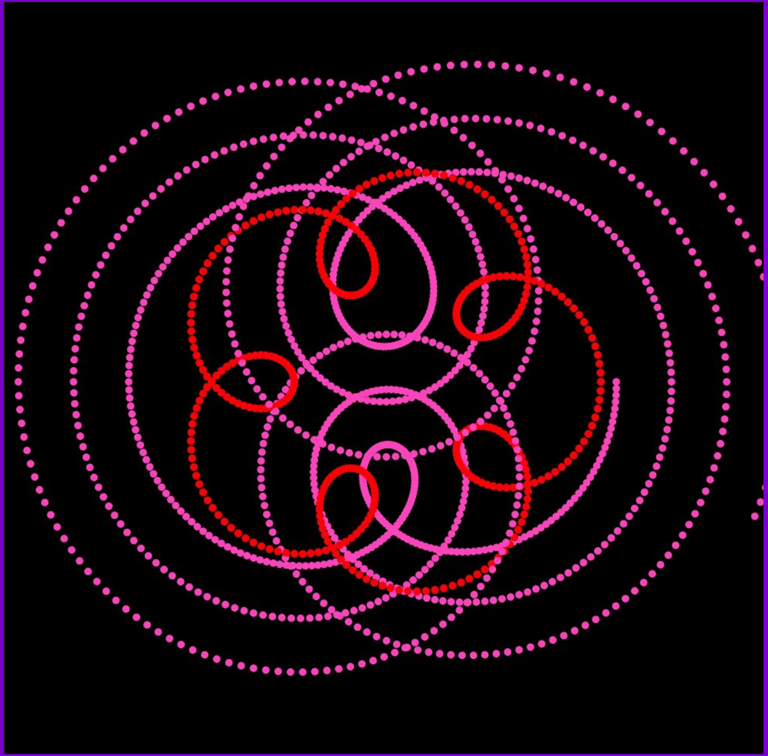
- VARIATION: Add more planets, stars and moons.

Variations: Arm



- Make it interactive!
  - Use mouseX and mouseY to rotate shapes
- Add even more joints
- Hint: you can use the map() function to scale and restrict angle values

Variation: Spirograph



- Start from the solar system example, but don't update the background in draw()!
- Play with changing values of rotation and translation to get cool patterns!

TO DO THIS WEEK

1. Attempt one of the variations or make something else creative with this week's exercises
2. Post screenshots of the outcomes to the Showcase forum

Recap

```
//create probability distributions with random()
if(random() < 0.4) { /* execute with 40% chance */ }
else { /* execute with 60% chance */ }

//transformations
translate(x,y); //move point of origin
rotate(rad); //rotate around origin, default in radians
scale(p); //scale coordinate system in decimal percents
push(); //save previous transformations and drawing styles
pop(); //reset to previous transformations and styles

angleMode(MODE); // set angle unit to DEGREES or RADIANS
```