CART 253
Creative Computation 1

Email: l.wilkins@concordia.ca
Office Hours: Tuesday 12-1
Course Github: https://github.com/LeeRobot/CART253-F-22

What we'll be doing today

- Inspiration
- Time to work on project 2
- Learn about Loops
- Exercise about Loops (Submit on Moodle)

Inspiration

- Nature Of Code - Fractals https://natureofcode.com/book/chapter-8-fractals/
- Joshua Davis https://joshuadavis.com/

Upload your code to Github if you want to share!

- Add your link here and I'll share with the both classes!
- Github is a great way to store and share code
- There is version control so you can go back and see what you changed
- I recommend using the desktop client, here's a tutorial
- You can also just drag and drop your code into a repository

Second Project: Screens in Spaces, due week 8

- You may work in pairs (if you want)
- Create a **site-specific installation** using your computer screen and p5.js. Think about how people in these spaces interact and expect screens to behave in different contexts. Your piece does not have to be interactive, but it does have to evolve or change over time in some way. This can exist anywhere, but you should be able to show us the work either through video documentation and demo in class or, take us to the piece physically for critique. You can do this project in pairs or individually, but no bigger groups. You can use a makeymakey to simualte keybaord input.
- The piece must be interactive and/or change over time.
- Use at least 2 conditionals, 1 loop, and a function or array. We'l talk about those next week.
- Full details on Git and Moodle.

Second Project: Screens in Space, due week 8

- Some ideas:
  - Hallways
  - Classrooms
  - Lounge spaces
  - Outside
  - Lobby
  - Metro station
  - Restaurant/cafe
  - It can be somewhere far away or somewhere we can't visit (please record your installation / people interacting with it)
  - **How can your piece highlight, or contrast, or enhance the space its in? Can it blend in, can it make the viewer stop? How do people react to it? Can it take advantage of the architecture?**
  - **Think of projections, screens, tablets, laptops, screens etc.**

Second Project: Screens in Space, due week 8

- https://momentfactory.com/work/all/all/montreal-signe-ode-a-la-vie Projection mapping
- Putting p5.js on your phone https://www.youtube.com/watch?v=OyZNj7oMgek using screen cast or using your computer to host the code https://creative-coding.decontextualize.com/mobile/

```
let x = 250;
let y = 500;
...
function draw() {
  background(255, 0, 0);
  // Draw a circle
  fill(100);
  ellipse(x, y, 24, 24);
  y = y - 1;
  // Reset to the bottom if it is off the screen
  if (y < 0) {
    y = height;
  }
}
```

Animations, linear

```
let x= 250;
let y = 500;

function draw() {
  background(255, 0, 0);
  // Draw a circle
  fill(100);
  ellipse(x, y, 24, 24);
  // Jiggling randomly on the horizontal axis
  x = x + random(-10, 10);
  // Moving up at a constant speed
  y = y - 1;
  // Reset to the bottom if it is off the screen
  if (y < 0) {
    y = height;
  }
}
```

Animations, jiggly!

```
let x= 0;
let y = 0;
let dX = 0;
let dY = 0;

function draw(){

  x = lerp(x, dX, 0.05);
  y = lerp(y, dY, 0.05);

  ellipse(x, y, 50, 50);
}

function mousePressed(){
  dX = mouseX;
  dY = mouseY;
}
```

read lerp in the reference and lerp color . Lerp interplates the numbers between two values. For example, X, and dX. We can use it to move between two points, change between two colours, or smoothly make any transitions.

Animations, lerp!

If/else runs once per frame.

A while or a for loop can run multiple times before the frame resets.

```
while(this is true){
  Do this
}
```

This loop will repeat as long as the condition is true, so the code below it won’t run until the loop is finished.

As with any loop, it is important to ensure that the loop can 'exit', or that the test condition will eventually evaluate to false. This is to keep your loop from trying to run an infinite amount of times, which can crash your browser.

```
let myVar = 0;

while( myVar <= 5 ){
  print("my loop!");
}
```

Bad loop! This will run forever! Code after it will never be executed!

```
let myVar = 0;

while( myVar <= 5 ){
    print("my loop!");
    myVar++;
}
```

Good loop! It will run 5 times sequentially and then move on to the code below

```
let xPos = 0;
let yPos = 100;
let numOfCircles = 2;
let size = 100;

function draw() {
 let counter = 0;
 while(counter < numOfCircles){
  ellipse(xPos+(counter*size), yPos, size, size);
  counter++;
  }
}
```
Try: making it happen on mouse Pressed
Making random colours, random extra shapes

```
for(let i = 0; i < 5; i ++){
  print(i);
}
```

Create a variable that is used to count
Check if this statement is true before each loop
Do this at the end of the loop

A for loop allows us to keep an index number of which loop we are at

```
function draw() {
  print("start of draw");
  for(let i = 0;i < 5; i ++){
    print(i);
  }
  print("end of draw");
}
```

We can use this to see the structure of the code. The loop runs 5 times (starting at 0) per cycle of draw

```
for(let i = 0;i < 5; i ++){
  rect(i*50, 0, 50, 50);
}
```

Draw a line of rectangles. We can use the index “i” to multiply the size of each square we want for the x position

```
let size = 50;

 for(let i = 0;i < width/size; i ++){
    rect(i * size, 0, size, size);
  }
```

To simplify, lets make Size into a variable. That way we can easily change the size at any time.

We can also automatically determine the number of squares needed by dividing width by size.

```
  for(let i = 0;i < width/size; i ++){
    for(let j = 0; j < height/size; j++){
    rect(i*size, j*size, size, size);
    }
  }
```

Now, lets create a "nested" loop, meaning a loop inside another loop. We will use j as an index for this one.

"i" draws lines going accross
For each line going accross, "j" will draw them down

```
for(let i = 0;i < width/size; i ++){
print("this is the first loop");
  for(let j = 0; j < height/size; j++){
   print("this is the nested loop");
    rect(i*size, j*size, size, size);
  }
}
```

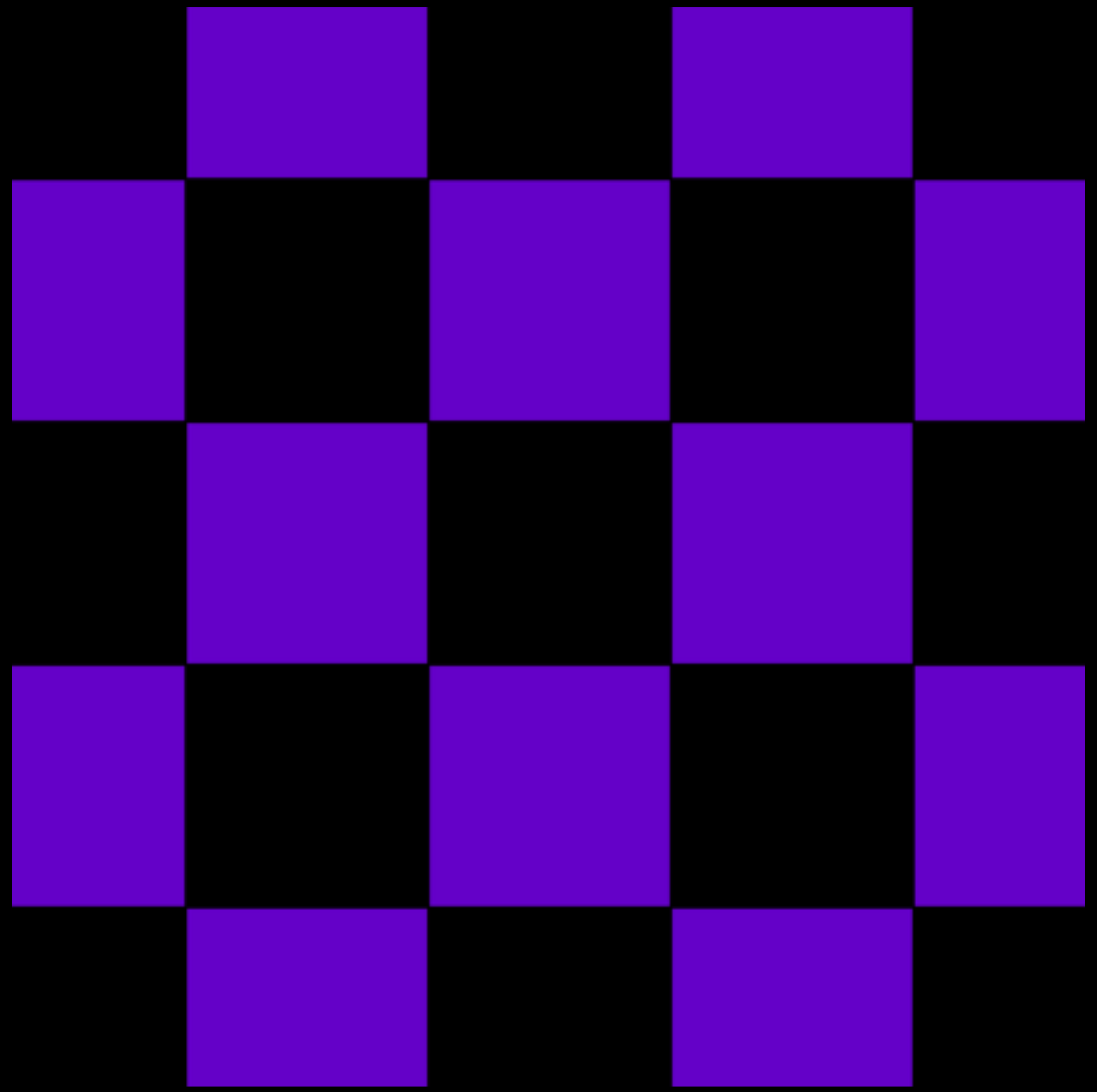
Lets use print to see the structure of this

```
  Loop to draw the rows
    Loop to draw the columns
    if it is an even row
      check if it is an even column, and set the fill one way
     otherwise
       Set fill the opposite way
    Draw the square
```

Create a program that:

- Draws a **checkerboard** (alternating square of color) every time the mouse is pressed with a **different** color scheme and **different** checker size each time.

Hints: Use a nested for loop to make the squares. Use Modulo to check for every second one. Use the mousePressed() function. Create variables for size.



In Class Exercise

Style & formatting

Full style guide here.

- Get rid of extra white space
- use TAB to indent functions, everything in side the function should be aligned.
- Curly brackets should be at the first line of the function, and alone at the end of the function.

```
function setup()    {
createCanvas(500,500);}
```

✗

```
function setup() {
  createCanvas(500,500);
}
```

✓