# 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
- Talk about project 2
- Learn about conditionals
- Exercise about conditionals (Submit on Moodle)

# Inspiration

- Exporting graphics from p5.js to draw on a plotter by Roni Cantor
- One Last Time By Shereen Cheong
- https://refikanadol.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/

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

```
if(this is true) {
    Do this
} else {
    Do Something Else
}
```

```
if(this is true) {
    Do this
} else if (this is true) {
    If not, check if this is true.
}
```

```
if(this is true) {
    Do this if the condition is met
} else if (this is true) {
    If it isn't, but the second condition is met, do this
} else {
    If none are true, do this
}
```

The if/else chain only runs 1 of the sets of commands, checking them in order.

```
if(mouse is pressed) {
    Make screen blue always if the mouse is pressed,
and always increase the counter.
  if(check if counter is hit) {
      But Only draw a circle if the counter is hit
  }
}
```

> greater than

< less than

>= greater than or equal to

<= less than or equal to

== equal to

!= not equal to

=== equality with strict type checking, safer

!== inequality with strict type checking, safer

*** = is to assign a value, == or === is to compare a value ***

# Operators: Relational

mouseIsPressed is a boolean variable, meaning it is either TRUE or FALSE. There is also keyIsPressed!

```
if(mouseIsPressed == true) {
    Fill(random(255);
    ellipse(random(width), random(height), 100, 100);
}
```

How can we make it follow the mouse?

+ addition

- subtraction

* multiplication

/ division

% modulo

++ add one shorthand

- - subtract one shorthand

# Operators: Mathematical

Start myVar at 0, draw circles while its less than 100, increase the myVar by 1 every time.

```
let myVar = 0;

if(myVar < 100 ) {
    fill(random(255));
    ellipse(random(255), random(255), 100, 100);
    myVar++;
}
```

Start myVar at 0, draw circles while its less than 100, increase the myVar by 1 every time. If the condition is not met, draw rectangles

```
let myVar = 0;

if(myVar < 100 ) {
    fill(random(255));
    ellipse(random(255), random(255), 100, 100);
    myVar++;
} else {
    fill(0);
    rect(random(255), random(255), 100, 100);
}
```

Start myVar at 0, draw circles while its less than 100, increase the myVar by 1 every time. If the condition is not met, draw rectangles

```
let myVar = 0;

if(myVar < 100 ) {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
} else if(myVar < 200 ) {
    fill(0);
    rect(random(width), random(height), 100, 100);
}
myVar++;
```

```
let myVar = 0;

if(myVar < 100 ) {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
} else if(myVar < 200 ) {
    fill(0);
    rect(random(width), random(height), 100, 100);
} else {
    fill(255);
    rect(random(width), random(height),200, 200);
}
    myVar++;
```

```
let myVar = 0;

if(myVar < 100 ) {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
   print('first if statement is running');
} else if(myVar < 200 ) {
    fill(0);
    rect(random(width), random(height), 100, 100);
    print('second if statement is running');
} else {
    fill(255);
    rect(random(width), random(height),200, 200);
    print('else statement is happening'
}
myVar++;
```

You can use print to see which statements are running. Sometimes your if statement is never true, so the code never gets executed.

% (modulo) is an operator that looks at the remainder. You can use this to do something every X number of times, for example to slow things down. In this example,

```
let myVar = 0;

if(myVar%2 == 0 ) {
    fill(random(255));
    rect(0, 0, width, height);
}
myVar++;
```

Remember the myVar++ has to be outside the if statement to make sure it runs everry time, regardless of whether the background is changing.

|| logical OR // if ( feelingTired || reallySleepy ) { lieDown(); }

&& logical AND // if ( haveWallet && (money > 10) ) { buyCake(); }

! logical NOT // showDebugInfo = ! inProductionMode;

# Operators: Logical

Mouse is pressed AND key is pressed

```
if(mouseIsPressed == true && keyIsPressed == true )  {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
}
```

Mouse is pressed OR key is pressed

```
if(mouseIsPressed == true || keyIsPressed == true )  {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
}
```

- pwinMouseY
- mouseButton
- mouseIsPressed
- mouseMoved()
- mouseDragged()
- mousePressed()
- mouseReleased()
- mouseClicked()
- doubleClicked()
- mouseWheel()
- requestPointerLock()

keyIsPressed

- keyPressed()
- keyReleased()
- keyTyped()
- keyIsDown()
- movedX
- movedY
- mouseX
- mouseY
- pmouseX

Lets try and make a click counter using mouse pressed. I want to be able to click 100 times.

```
let clickCounter = 0;

if(mouseIsPressed == true && clickCounter < 100 )  {
    fill(random(255));
    ellipse(random(width), random(height), 100, 100);
    clickCounter++;
    print(clickCounter);
}
```

What did we learn about how the variable is increasing?

The mousePressed function is called automatically every time the mouse button is pressed, and only called once. We can use this to keep track of the action of mouse being clicked, instead of the state of the mouse.

```
function draw(){

}

function mousePressed(){

}
```

mouseIsPressed / mouseIsPressed (the variable)
You can use this to check if the mouse is pressed, and make conditions that happen throughout any amount of time that the mouse is pressed.

mousePressed() / keyIsPressed() (the function)
You can call this function once when the mouse is pressed. It only runs when that action is performed. Theres also released/ mouse wheel, click, drag, etc.

# Functions / Variables

The mousePressed function is called automatically every time the mouse button is pressed, and only called once.

```
function draw(){

}

function mousePressed(){

  if(clickCounter < 100)  {
      fill(random(255));
      ellipse(random(width), random(height), 100, 100);
  }
  clickCounter++;
  print(clickCounter);
}
```

Every second click using %

```
function draw(){

}

function mousePressed(){

  if(clickCounter%2==0)  {
      fill(random(255));
      ellipse(random(width), random(height), 100, 100);
  }
  clickCounter++;
  print(clickCounter);
}
```

Start myVar at 0, draw circles while its less than 100, increase the myVar by 1 every time. If the condition is not met, draw rectangles

```
let myVar = 0;

function draw(){
  if(myVar < 100 ) {
      fill(random(255));
      ellipse(random(width), random(height), 100, 100);
  } else if(myVar < 200 ) {
      fill(0);
      rect(random(width), random(height), 100, 100);
  }
}

function mousePressed(){
      myVar++;
}
```

# Nesting IF Statements

```
let myVar = 0;

function draw(){
  if(mouseIsPressed ) {
        fill(0);
        rect(random(width), random(height), 100, 100);
     if(myVar < 20 ) {
        background(255);
     }
  }
}

function mousePressed(){
     myVar++;
}
```

```
if(condition){
    Is this condition ever being met?
}
```

# Debug using print

# 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);
}
```

Create a program that
- draws a red circle in the middle of the page ONLY if the mouse is pressed
- draws a green circle in the left of the page ONLY if a key is pressed
- if none are pressed, there are no shapes on the screen
- After 3 mouse presses, the background should turn blue.

**Submit in Moodle as a zipped project**

# In Class Exercise