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

- Sign up for project times
- Learn about functions
- In class exercise about functions

Sign up for a presentation slot!

- https://docs.google.com/spreadsheets/d/1lcvVL19xLVkVVgk5Dw01eTYGCNxmEaGdVfBiFTCXjsg/edit#gid=0

  Presentations are next week!

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/

Functions are a way of organizing code. We already use functions all the time! We can create custom functions that run at specific times, and can be modified to create variations on a drawing, effect, or equation.

Functions

```
function drawSquare(){

}
```

drawSquare is the name of the function.
The word function tells us we are making a function!
Round brackets are used to hold parameters, which we'll get to later.
Like other functions, they need curly brackets.

Custom functions should exist outside the draw or setup loops.

What is a function?

```
function setup(){

}
function draw(){

}

function drawSquare(){

}
```

Where does it go?

```
function draw(){

}

function drawSquare(){
  rect(random(width), random(height), 100, 100);
}
```

We can put anything in a function! As much code as you want, and you can reference other functions too.

Writing a function

```
function draw(){
   drawSquare();
}

function drawSquare(){
  rect(random(width), random(height), 100, 100);
}
```

Our function won’t run unless it is called. Calling a function means referencing the function name. You can call a function anywhere you want, in draw, setup, mousePressed, or keyPressed.

Calling a function

We can pass a parameter through a function, which allows us to run the same code with different variations.

Parameters

```
drawSquare(100);
```

```
function drawSquare(Fcolor){
  fill(Fcolor);
  rect(mouseX, mouseY, 100, 100);
}
```

We can put a parameter inside the first round brackets, and anywhere you use that parameter name, the value will be repeated.

Passing a parameter

```
let counter = 0;
function draw(){
  drawSquare(counter);
  counter++;
}

function drawSquare(Fcolor){
  fill(Fcolor);
  rect(mouseX, mouseY, 100, 100);
}
```

```
drawSquare(100, 30);


function drawSquare(Fcolor, size){
  fill(Fcolor);
  rect(mouseX, mouseY, size, size);
}
```

Inside the function, we refer to the parameter by its name. Anywhere the name is, it will be replaced by the parameter you pass through the function.

Multiple Parameters

```
Let sqSize = [100, 30, 50, 10];

function draw(){
  drawSquare(sqSize[0]);
}

function drawSquare(size){
  rect(mouseX, mouseY, size, size);
}
```

Use values from an array

```
function draw(){
 for(let i = 0; i < sqSize.length; I++){
   drawSquare(sqSize[i]);
 }
}

function drawSquare(size){
  rect(mouseX, mouseY, size, size);
}
```

Cycle through an array

A function can return a value by using the return command. This means that it can give you a result, or pass along a number.

```
if(check()){
// do a task
}

function check(){
  return true;
}
```

Returns

A function can return a value by using the return command. This means that it can give you a result, or pass along a number.

```
if(check()){
// do a task
}

function checks){
  if(mouseX < 100 && mouseY < 100){
  return true;
 }
}
```

Returns

A function can return a value by using the return command. This means that it can give you a result, or pass along a number.

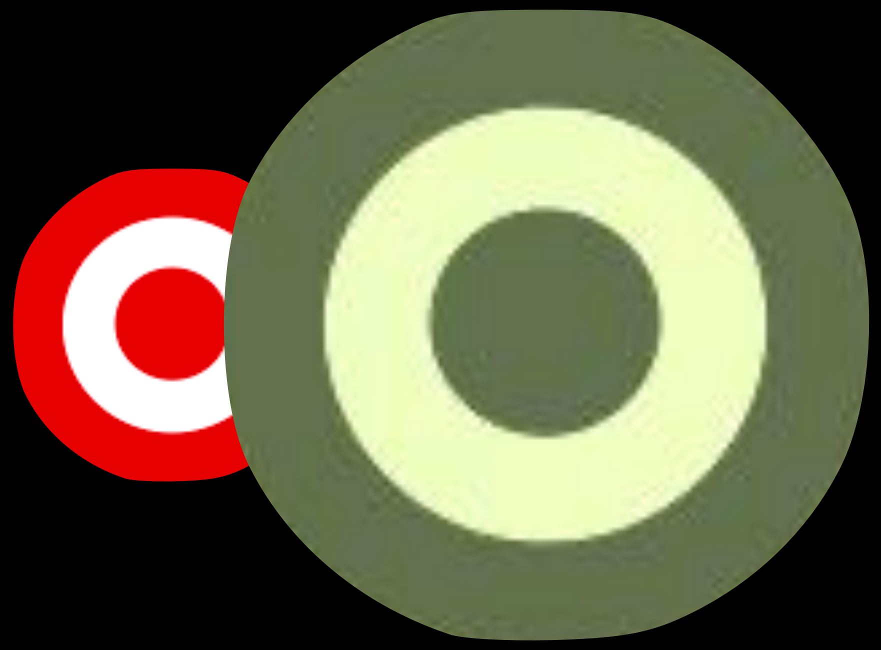
```
if(checkMousePosition()){
 drawSquare();
}

function checkMousePosition(){
  if(mouseX < 100 && mouseY < 100){
  return true;
 }
}

function drawSquare(){
  rect(mouseX, mouseY, 100, 100);
}
```

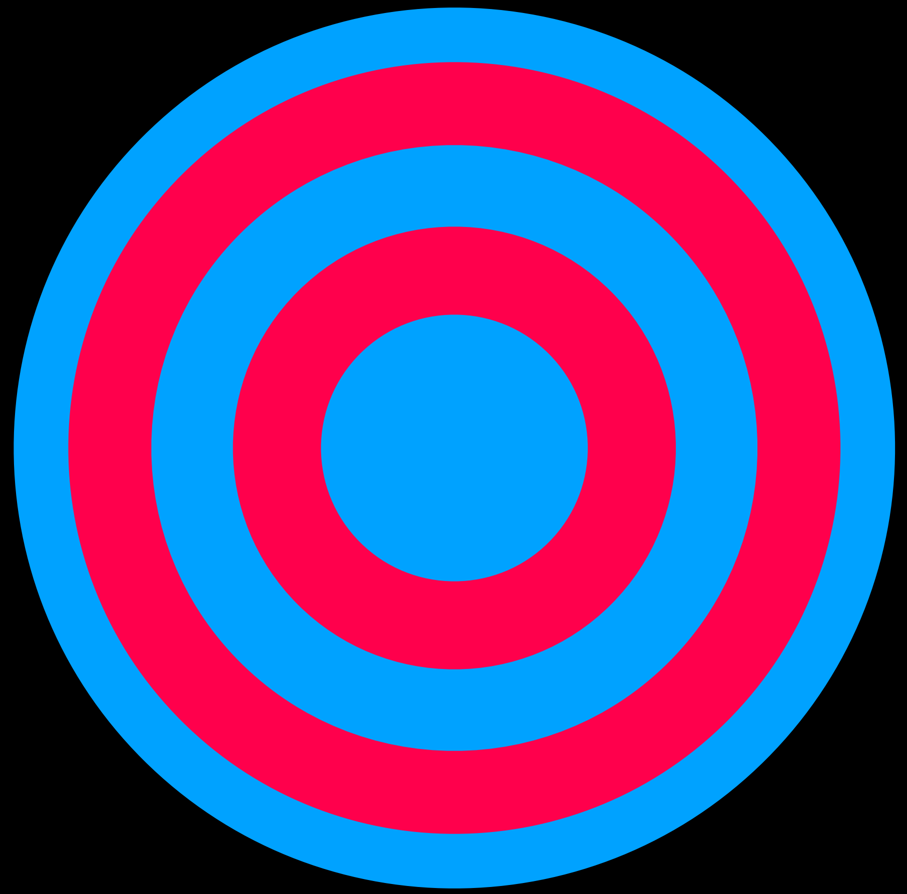
This is a simple check function, but you can get very complex with them and it can simplify your code and make it much more readable.

Check & Run

Create a piece of code that uses a function to draw targets on the screen using a for loop when the mouse is clicked at the mouse position. The targets should alternate between two colours that are randomly selected with each new target.

Use a different function to return a value to check the mouse position. If the mouse position in in the upper right or lower left, add a circle to the next target drawn. If the position is in the lower right or upper left, remove a circle in the next target drawn





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);
}
```

✓