imc

js p5*

introduction to media computing
week 04

School of creative media, mw

Today’s topics (week 04)

js

- quick review
- blocks of code
- nested blocks of code
  - nested `if-else`
  - nested `while()`

Today’s topics (week 04)

js

- quick review
- blocks of code
- nested blocks of code
  - nested `if-else`
  - nested `while()`

p5*

- measuring distance with `dist()`
- keyboard interactivity with `KeyIsDown()`

Review: `if else`

```
if (x == 200) {
     //  Do something
}
else if (x < 200) {
     //  Do something
}
else {
     //  Do something else
}
```

Only <u>ONE</u> block of code will be executed.

Review: relational operators

js

```
if (x >= 200) {
    //  Do something
}
else {
    //  Do something else
}
```

operators	meaning
>	larger than
<	smaller than
>=	larger or equal to
<=	small or equal to
!=	not equal to
==	equal to

Review: Logical operators

operators	meaning
||	Logical OR
&&	Logical AND
!	Logical NOT

```
if (x == 0 || x == 200 ) {
    // Do something
}
```

This block will run only if
X equals to 0
OR
X equals to 200

Review: Modulo Operator '%'

- modulo operator '%' computes the remainder of an integer division. Example: `5 % 2` returns 1.
- This operator is particularly useful for some simple looping operation.

Review: the `while()` loop

```
let x = 5;
while (x > 0) {
  text(x, x * 12, 20);
  x--;
}
```

```
let y = 0;
while (y < 5) {
  text(y, 20, y * 12);
  y++;
}
```

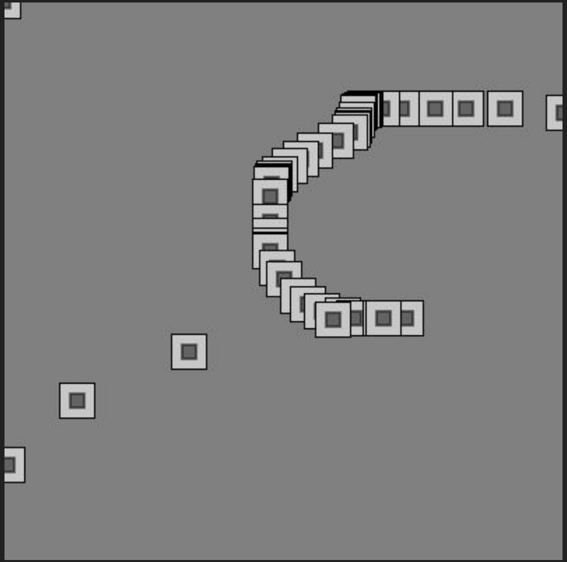
Blocks of Code

Blocks of Code

A simple concept to build, organize and reuse codes

```
// move-and-paint
rectMode(CENTER);
fill(200);
rect(mouseX, mouseY, 25, 25);
fill(100);
rect(mouseX, mouseY, 10, 10);
```

snippet #1: `move-and-paint`

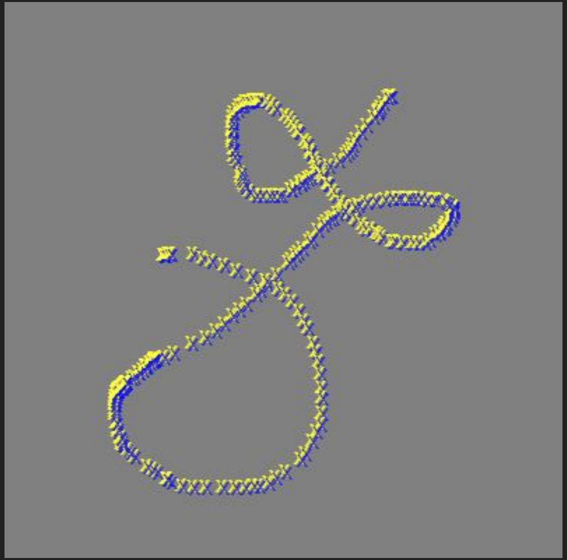


Blocks of Code

A simple concept to build, organize and reuse codes

```
// press-and-paint
if (mouseIsPressed) {
  fill(255,255,0);
  text("X", mouseX, mouseY);
  fill(0,0,255);
  text("X", mouseX+2, mouseY+2);
}
```

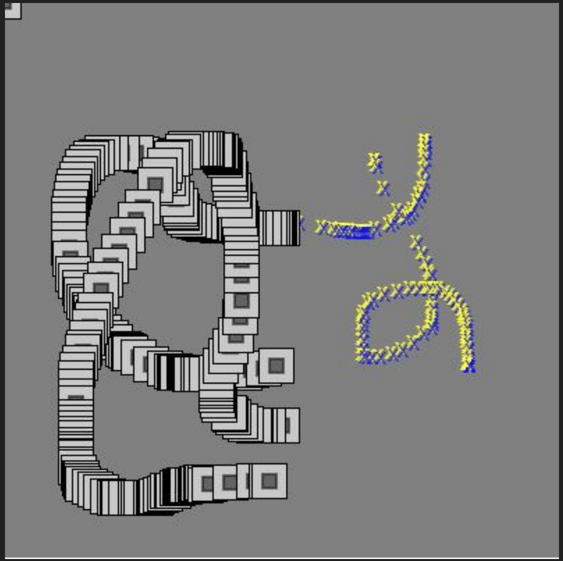
snippet #2: `press-and-paint`



Blocks of Code

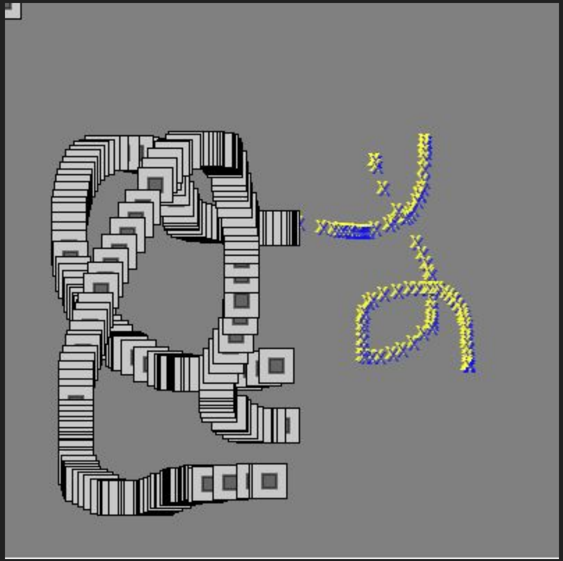
Let’s combine them with some condition !

```
if (mouseX > 200) {
  // press-and-paint
}
else {
  // move-and-paint
}
```



Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

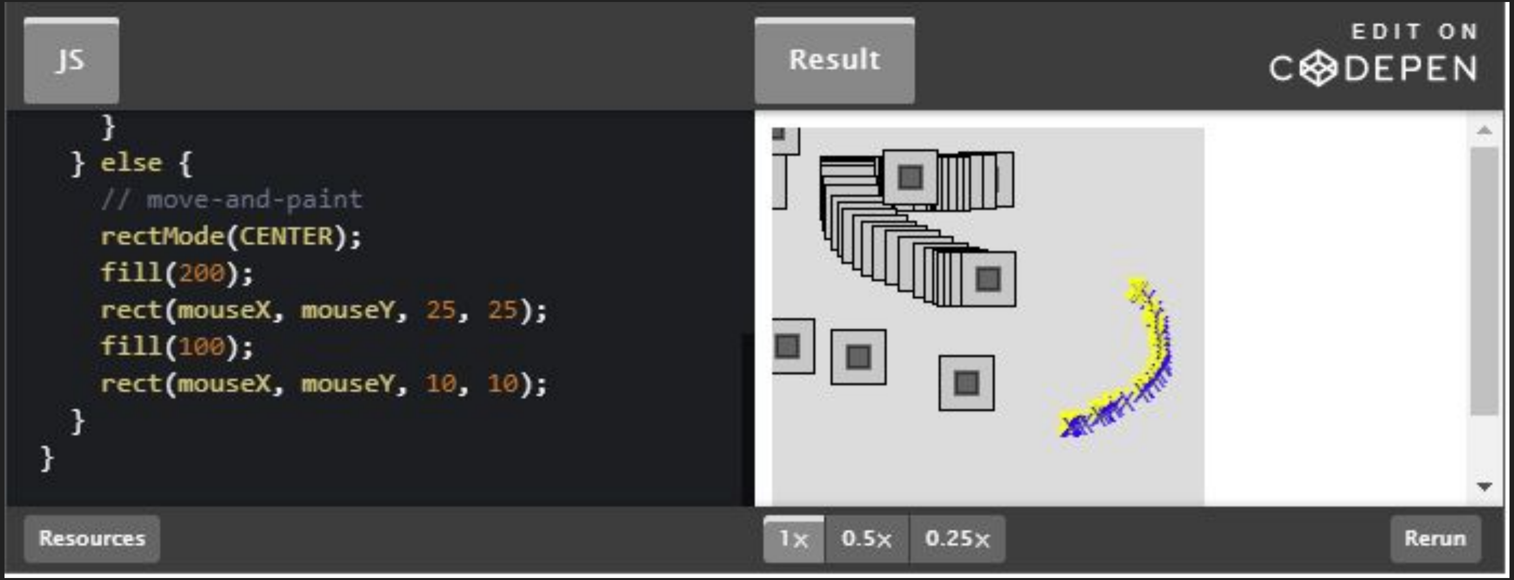


Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

In JavaScript, a **block of code** is always enclosed (identified) by a pair of parentheses { . . }.

Blocks of Code



```
    }
  } else {
    // move-and-paint
    rectMode(CENTER);
    fill(200);
    rect(mouseX, mouseY, 25, 25);
    fill(100);
    rect(mouseX, mouseY, 10, 10);
  }
}
```

Nested Blocks of Code

Nested Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

Nested Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

Outer block

Nested Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

inner block
inner block

Nested Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

inner block
inner block

Nested blocks

Nested `if-else {}`

js

Nested `if-else`

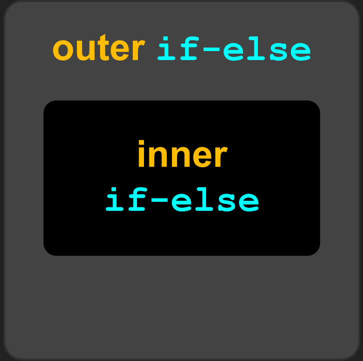
```
if ( <condA> ) {
   // some code here;
}
```

Nested if-else

```
if ( <condA> ) {
   // some Block here;
}
```

Nested `if-else`

```
if ( <condA> ) {
  if ( <condB> ) {
    // some code here;
  }
}
```

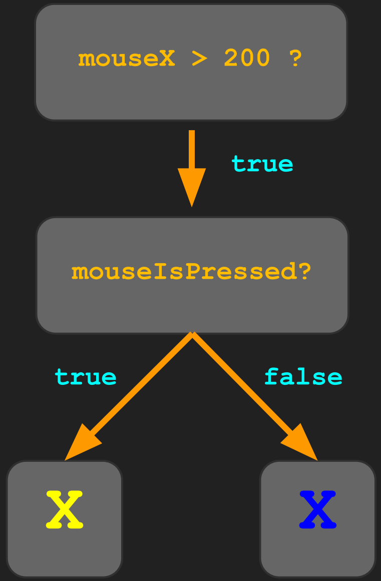


Nested if-else

js

Example 01

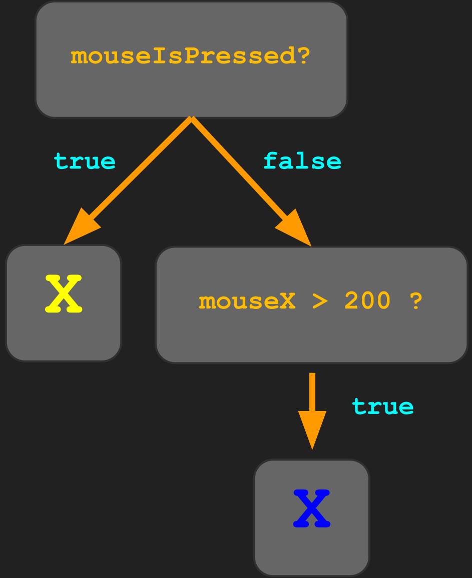
```
if (mouseX > 200) {

  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
  }
  else {
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }

}
```



Nested `if-else`

Example 02

```
if (mouseIsPressed) {
  fill(255,255,0);
  text("X", mouseX, mouseY);
}
else {

  if (mouseX > 200) {
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }


}
```



js

Nested `while()` loop

Nested `while()` loop

```
while ( <condA> ) {
   // some code here;
}
```

Nested `while()` loop

```
while ( <condA> ) {
   while ( <condB> ) {
   // some code here;
   }
}
```

Nested `while()` loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

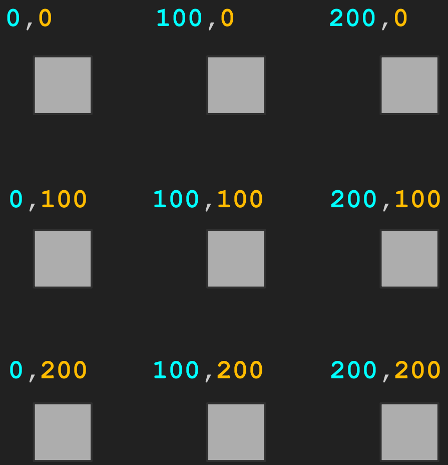
Nested loop is an essential construct for creating regular patterns

Nested while( ) loop

js

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

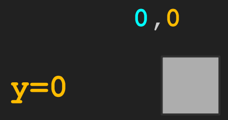


Nested while() loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step



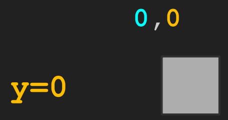
Nested while() loop

js

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step

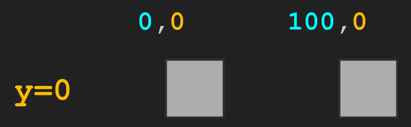


Nested while() loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step

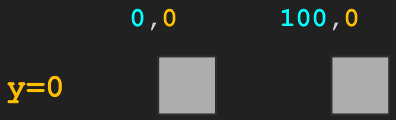


Nested `while()` loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step

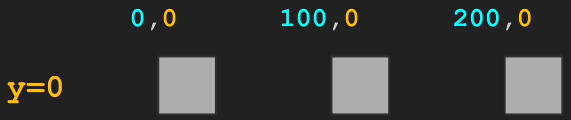


Nested while() loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step



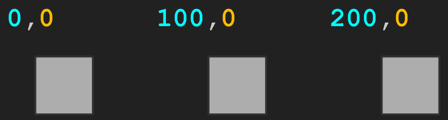
Nested while() loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

y=100

step-by-step



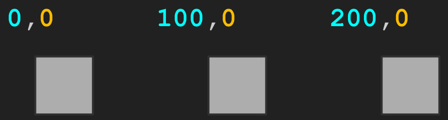
Nested while() loop

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

y=100

step-by-step



Nested while() loop

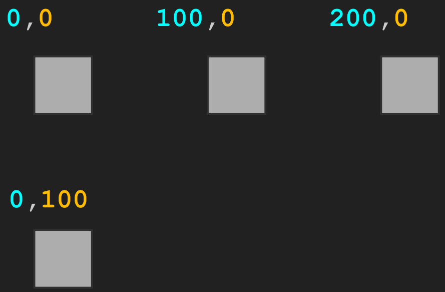
js

Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

y=100

step-by-step

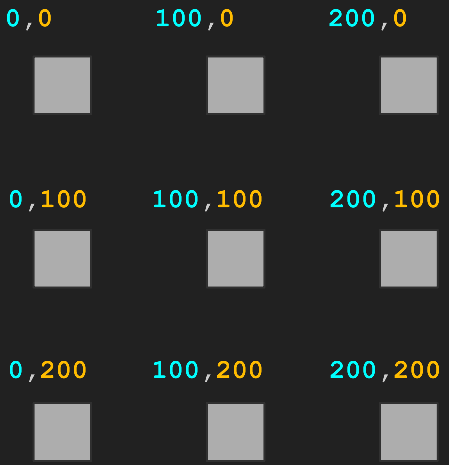


Nested `while()` loop

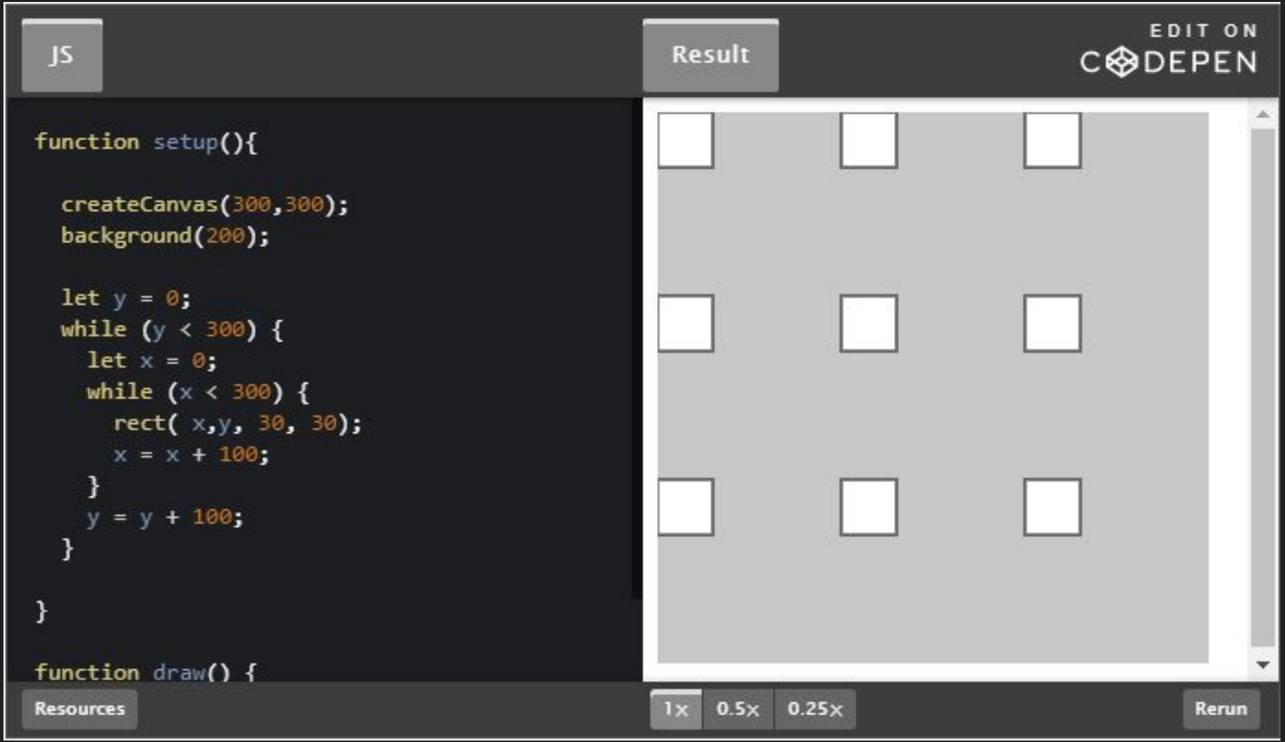
Example 01

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```

step-by-step



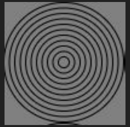
Nested `while()` loop



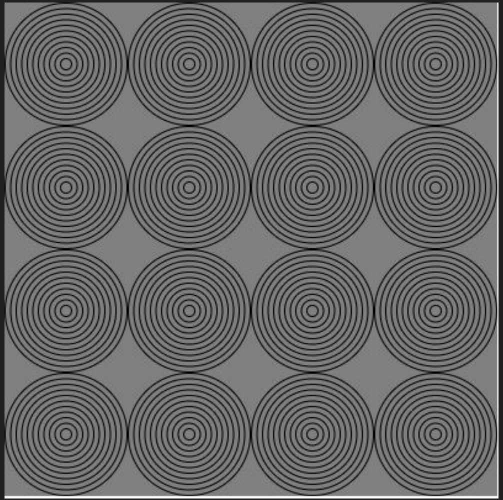
```
function setup(){

  createCanvas(300,300);
  background(200);

  let y = 0;
  while (y < 300) {
    let x = 0;
    while (x < 300) {
      rect( x,y, 30, 30);
      x = x + 100;
    }
    y = y + 100;
  }

}

function draw() {
```

exercise

In-class exercise 1



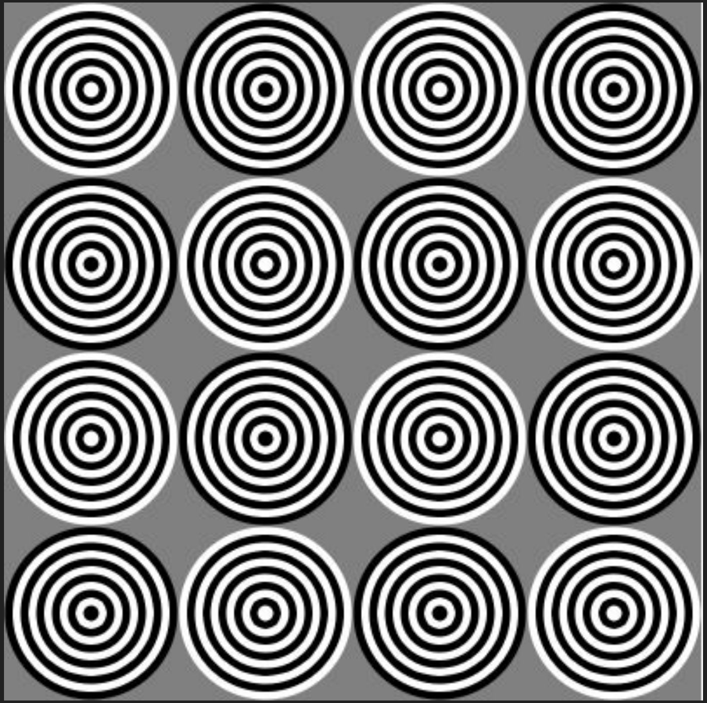
1. Use a `while()` loop to draw the shape shown on the left



2. Now, use the code in step 1 as a basic block for drawing the shape. Create a new <u>nested</u> `while()` loop to supply a sequence of `(x,y)` coordinates, and use the 'step 1' block to the code in step 1, such that fill the whole canvas with this shape.

exercise

In-class exercise 1 option*



p5*

Keyboard interactivity `keyIsDown()`
Measure distance with `dist()`

Keyboard interactivity `keyIsDown()`

- `keyIsDown(<KEYCODE>)` checks whether a key of `KEYCODE` on the keyboard is being pressed down.

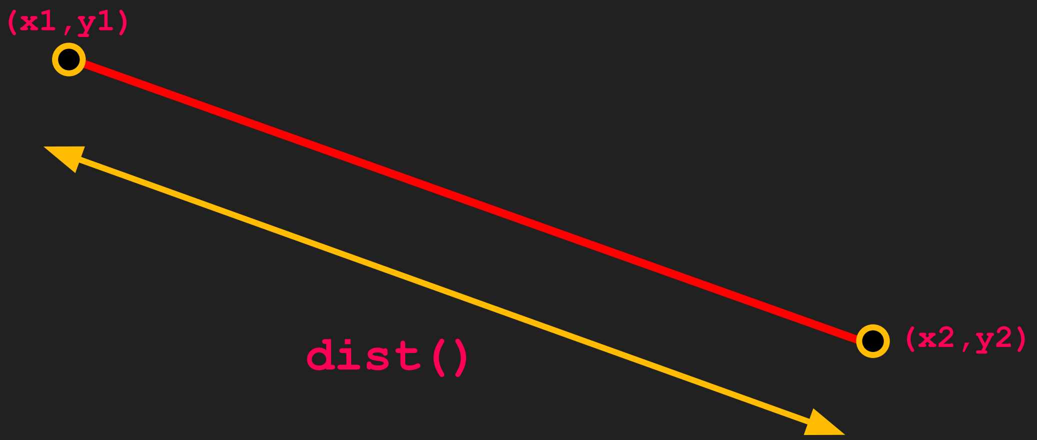
Most common `KEYCODE`:

`BACKSPACE, DELETE, ENTER, RETURN, TAB, ESCAPE, UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW`

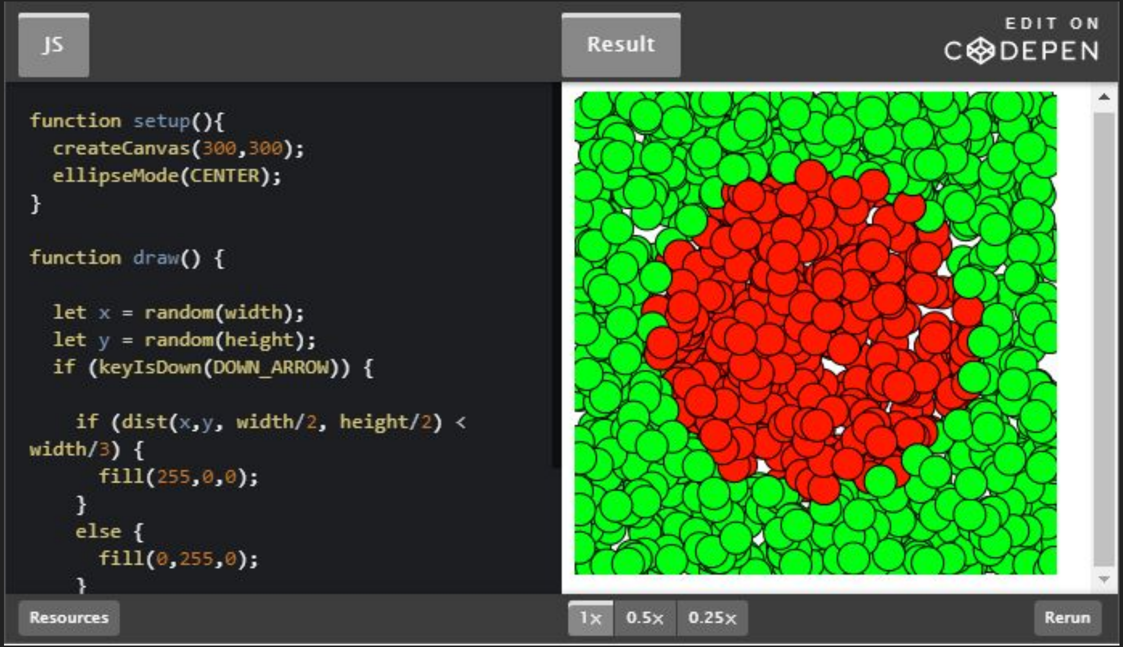
```
if (mousekeyIsDown(LEFT_ARROW)) {
    //  Draw or Do something
}
```

Measure distance with `dist()`

- `dist(x1,y1,x2,y2)` returns the distance between two given points `(x1,y1)` and `(x2,y2)`.



Measure distance with `dist()`



```
function setup(){
  createCanvas(300,300);
  ellipseMode(CENTER);
}

function draw() {

  let x = random(width);
  let y = random(height);
  if (keyIsDown(DOWN_ARROW)) {

    if (dist(x,y, width/2, height/2) <
width/3) {
      fill(255,0,0);
    }
    else {
      fill(0,255,0);
    }
```

This sketch draws a small circle at random places whenever the `DOWN_ARROW` key is pressed.

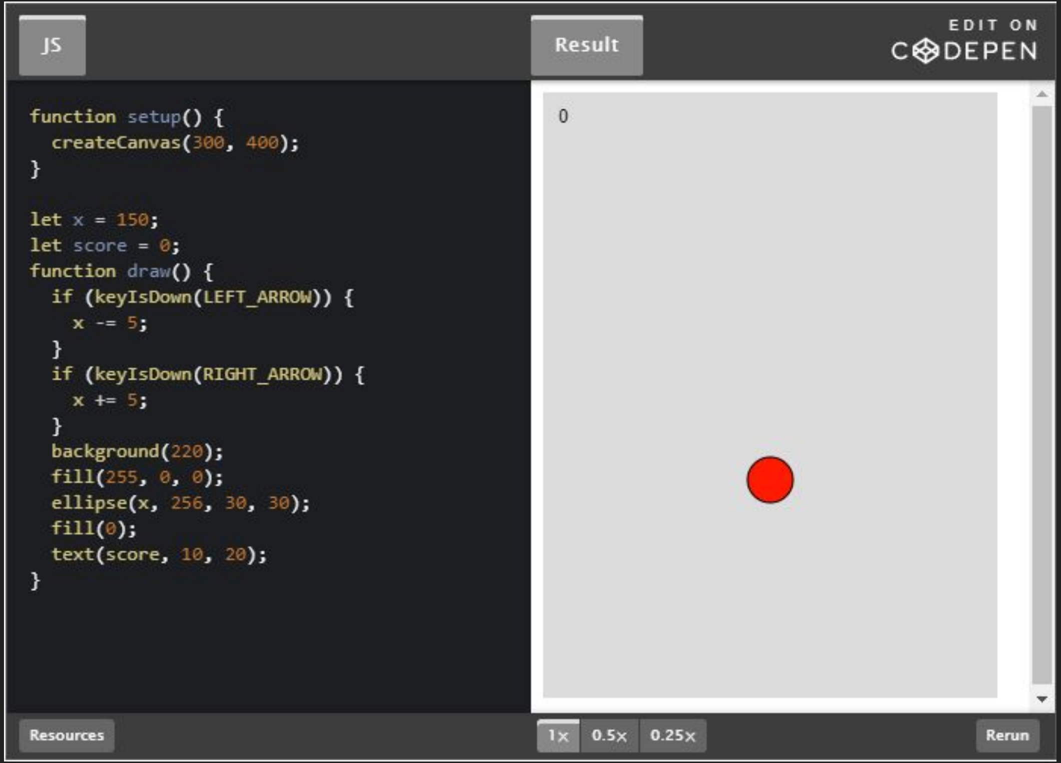
The random circle's color depends on its distance from the canvas's center (measured by `dist()`)

imc week 04

School of creative media, mw

exercise

In-class exercise 2



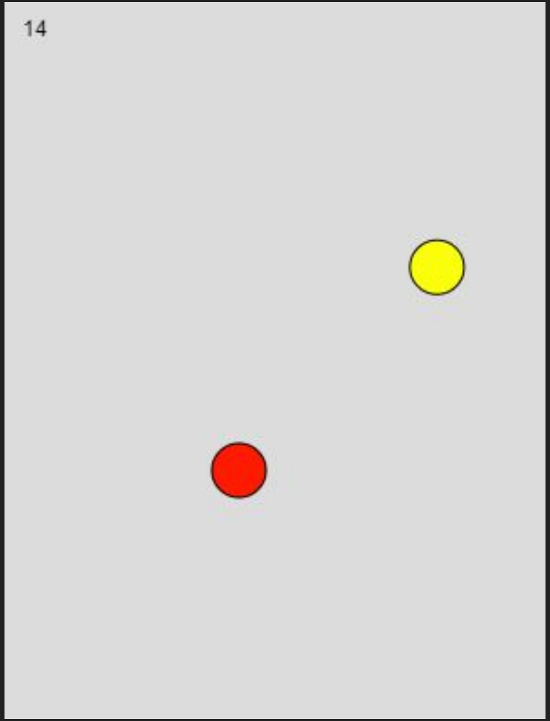
```
function setup() {
  createCanvas(300, 400);
}

let x = 150;
let score = 0;
function draw() {
  if (keyIsDown(LEFT_ARROW)) {
    x -= 5;
  }
  if (keyIsDown(RIGHT_ARROW)) {
    x += 5;
  }
  background(220);
  fill(255, 0, 0);
  ellipse(x, 256, 30, 30);
  fill(0);
  text(score, 10, 20);
}
```

Copy the 'in-class exercise 2 Skeleton' code from our Canvas Week 04 page.

Get familiar with the code structure, and the use of `keyIsDown()` for keyboard interactivity.

exercise

In-class exercise 2



Our 1st Nano GAME (a playable example is available on our Canvas Week 04 page)

1. Add both 'up' and 'down' keys to allow your red ball to move around in the whole canvas.

2. Keep your red ball remain in the canvas when it meets the boundary.

3. Create a random “Yellow” target, and the goal is to use your red ball to hit this target, every successful hit will score 1-point.

4. Re-create a new random “Yellow Target” when the last one is hit.