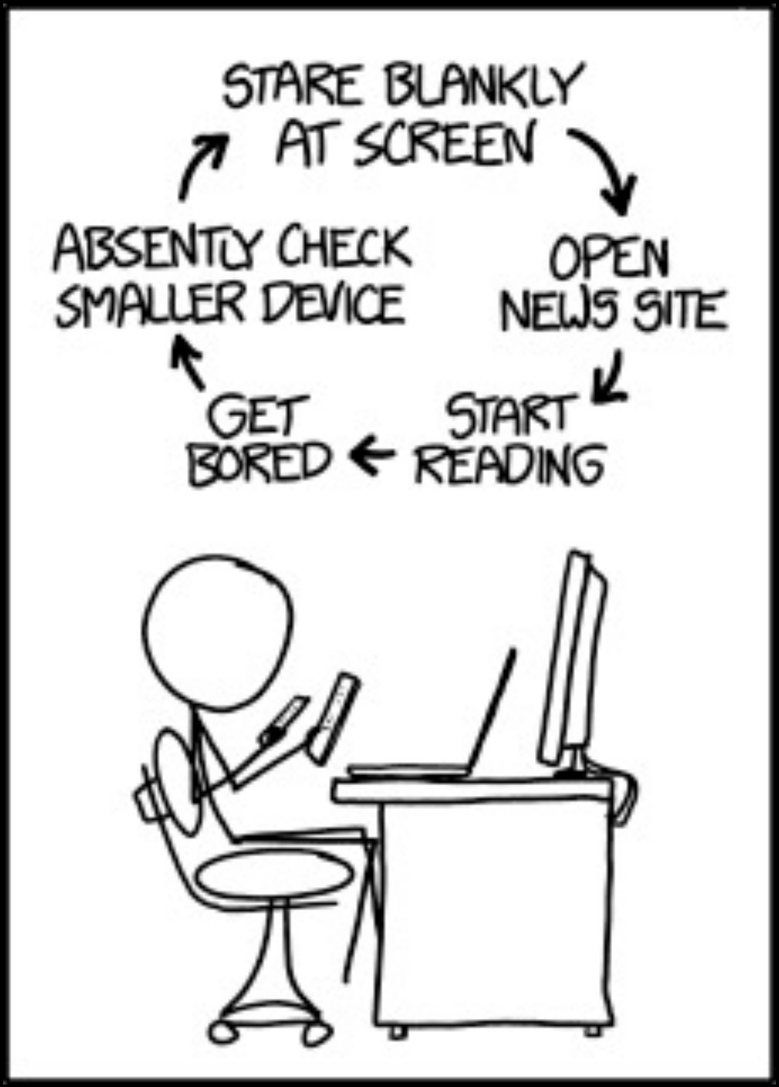
Computer Creativity

Loops

UBC Okanagan

Slides courtesy of Dr. Abdallah Mohamed.



Source:
XKCD 1411

Alt-text: Ugh, today's kids are forgetting the old-fashioned art of absentmindedly reading the same half-page of a book over and over and then letting your attention wander and picking up another book."

Reference Material
Loops

Objectives

- After reading, you should be able to:
  - define: loop, iteration
  - explain the difference between the `while` and `for` loops.
  - explain what ++ and -- operators do.
  - be able to use a loop in your sketches.
  - be aware and avoid common loop problems.
  - use nested iterations

- *Like conditionals, today is also mostly a revision of topics previously discussed in COSC111 and COSC122. we also discuss how they can be used in Processing.*

Iteration and Looping: Overview

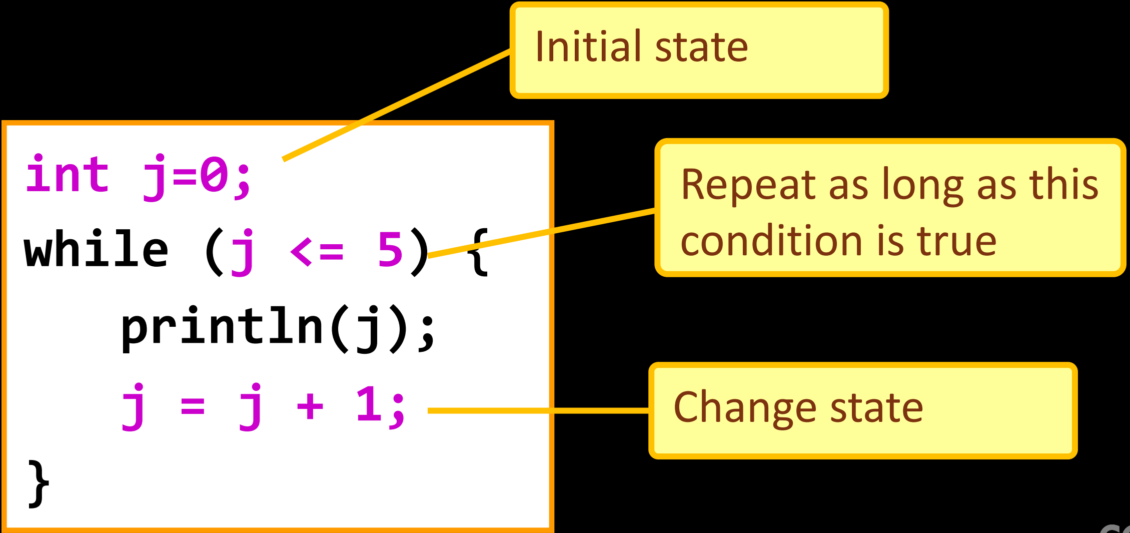
- A computer does simple operations extremely quickly.
- If all programs consisted of simple statements and decisions as we have seen so far, then we would never be able to write enough code to use a computer effectively.
- To make a computer do a set of statements multiple times we program ***looping structures***.
- A ***loop*** repeats a set of statements multiple times until some condition is satisfied.
  - Each time a loop is executed is called an ***iteration***.

The While Loop

- The most basic looping structure is the `while` loop.
- A while loop continually executes a set of statements while a condition is true.
- Syntax:

```
while (<condition>){
      <statements>
}
```

- Example:

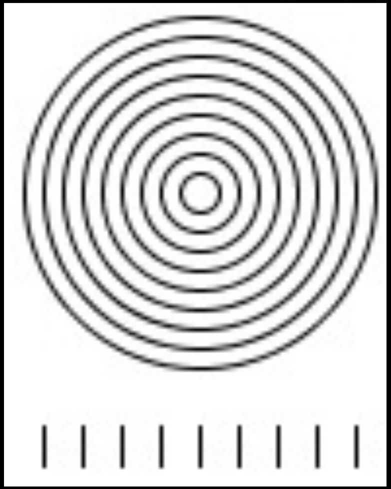


```
int j=0;
while (j <= 5) {
    println(j);
    j = j + 1;
}
```

Example

Repetition in Sketches

- Lets say we want to draw the sketch below
- ***One way*** to do this is to use the code on the right.
- As you see, there is a lot of repetition when drawing the circles or lines



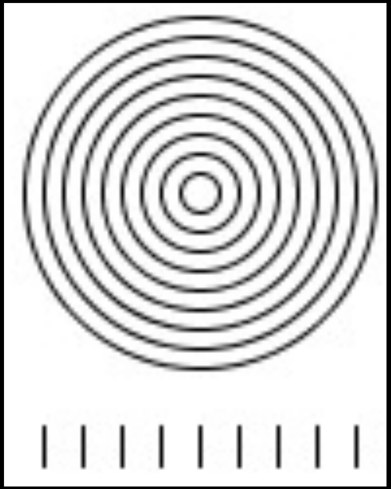
```
size(100,130); background(255);
noFill(); stroke(0);
// draw circles
ellipse(50, 50, 10, 10);
ellipse(50, 50, 20, 20);
ellipse(50, 50, 30, 30);
ellipse(50, 50, 40, 40);
ellipse(50, 50, 50, 50);
ellipse(50, 50, 60, 60);
ellipse(50, 50, 70, 70);
ellipse(50, 50, 80, 80);
ellipse(50, 50, 90, 90);
// draw lines
line(10, 110, 10, 120);
line(20, 110, 20, 120);
line(30, 110, 30, 120);
line(40, 110, 40, 120);
line(50, 110, 50, 120);
line(60, 110, 60, 120);
line(70, 110, 70, 120);
line(80, 110, 80, 120);
line(90, 110, 90, 120);
```

Example

Repetition in Sketches (2)

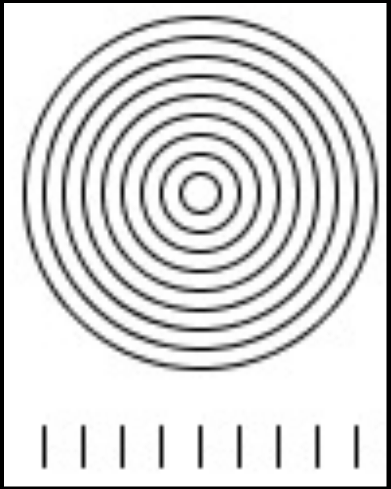
- Let’s use ***variables*** to replace hard coded values.
- Next, we will try to eliminate repetition (next slide)

```
size(100,130);background(255);
noFill(); stroke(0);
int spacing = 10, x, y, d, len;
// draw circles
x=50; y=50; d=10;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
ellipse(x, y, d, d); d += spacing;
// draw lines
x=10; y=110; len=10;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
line(x, y, x, y+len); x += len;
```



Repetition in Sketches (2)

- We can use a **while loop** to have the same output with much less code.



```
size(100,130);background(255);
noFill(); stroke(0);
int spacing = 10, x, y, d, len;

// draw circles
x=50; y=50; d=10;
while(d < 100){
  ellipse(x, y, d, d); d += spacing;
}

// draw lines
x=10; y=110; len=10;
while(x < 100){
 line(x, y, x, y+len); x += len;
}
```

The ++ and -- Operators

- It is very common to subtract 1 or add 1 from the current value of an integer variable.
- There are two operators which abbreviate these operations:

++	add one to the current integer variable
--	subtract one from the current integer variable

- Example:

```
int j=0;
j++;      // j = 1;  Equivalent to j = j + 1;
j--;      // j = 0;  Equivalent to j = j - 1;
```

The For Loop

- The most common type of loop is the for loop. Syntax:

```
for (<initialization>;<condition>;<next iteration>){
   <statement list>
}
```

- Explanation:
  1) *<initialization>* section - is executed once at the start of the loop
  2) *<condition>* section - is evaluated *before* every loop iteration to check for loop termination
  3) *<next iteration>* section - is evaluated *after* every loop iteration to update the loop counter

The For Loop

- Although Java will allow almost any code in the three sections, there is a typical usage:

```
for (i = start; i < end; i++){
    statement
}
```

- Example:

```
int i;
for (i = 0; i < 5; i++) {
    println(i);         // Prints 0 to 4
}
```

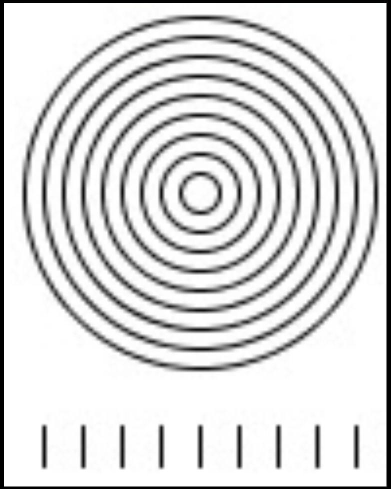
Examples of For Loops

```
for (int i = 0; i < 10; i++)       // start at 0 and count up to 9
for (int i = 0; i < 11; i++)       // start at 0 and count up to 10
for (int i = 0; i <= 10; i++)      // start at 0 and count up to 10
for (int i = 0; i < 10; i += 2)    // start at 0 and count up to 9 by 2

for (int i = 10; i > 0; i--)       // start at 10 and count down to 1
for (int i = 10; i >= 0; i--)      // start at 10 and count down to 0
for (int i = 10; i > 0; i -= 2)    // start at 10 and count down to 1 by 2
```

Repetition in Sketches (3)

- This is the same code as before but using a **for loop**.



```
size(100,130);background(255);
noFill(); stroke(0);
int spacing = 10, x, y, d, len;

// draw circles
x=50; y=50; d=10;
for(d = 10; d < 100; d += spacing){
  ellipse(x,y,d,d);
}

// draw lines
y=110; len=10;
for(x=10; x < 100; x += len){
 line(x,y,x,y+len);
}
```

Rules for Loops

- The iteration variable is a normal variable that must be declared, but it has the special role of controlling the iteration.
    - i, j, and k are the most common choices due to convention and because they are short.
- The starting point of the iteration can begin anywhere, including negative numbers.
- The continuation/termination test must be an expression that results in a Boolean value. It should involve the iteration variable to avoid an ***infinite loop***.
- The next iteration can have any statements, although usually only use the step size to change iteration variable.
    - The step size can be positive or negative and does not always have to be 1.

Common Problems – Infinite Loops

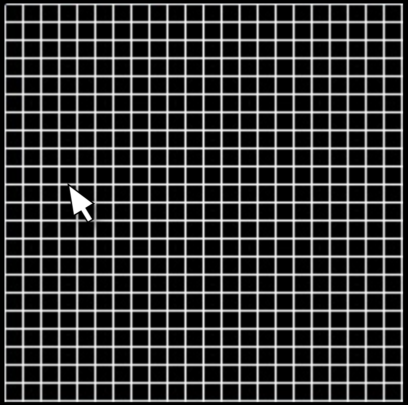
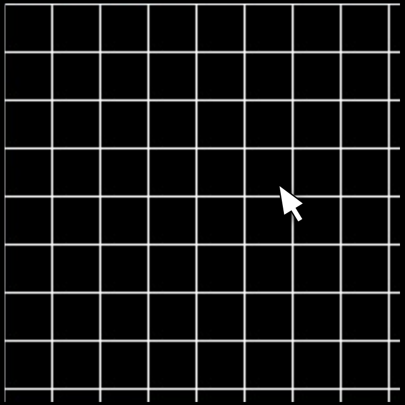
- ***Infinite loops*** are caused by an incorrect loop condition or not updating values within the loop so that the loop condition will eventually be false.
- Examples:

```
int i;
for (i=0; i < 10; i--){   // Should be i++
   println(i);  // Infinite loop: 0,-1,-2,..
}

i = 0;
while (i < 10) {
   println(i);  // Infinite loop: 0,0,0,..
}               // should change i inside the loop
```

Common Problems – Infinite Loops (2)

- Be careful when using certain system variables to change the state of the loop condition.
- For example, this code shows a grid of which **size** changes as `mouseX` changes.

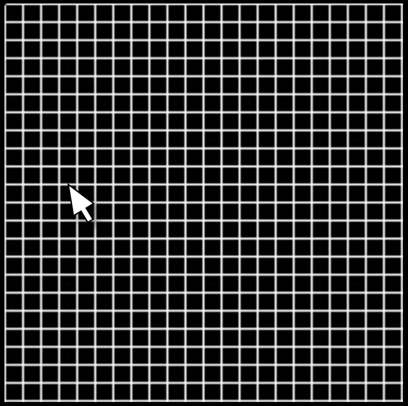
- However, there will be an infinite loop whenever `mouseX` is 0
  - and hence `x` doesn't change inside the loop, leading to the condition to be always true.

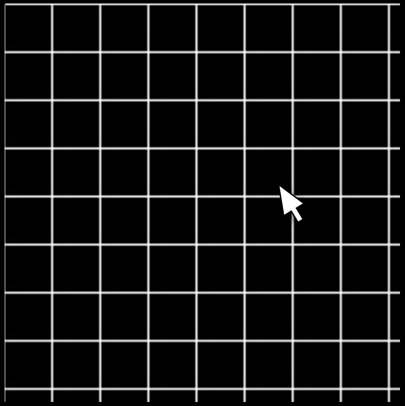
```
int x = 0, spacing = 10;
void setup() {
  size(200, 200);
}
void draw() {
  background(0);
  stroke(255);
  x = 0;
  spacing = mouseX / 4;
  while (x < 200) {
    line(x, 1, x, 199);
    line(1, x, 199, x);
    x = x + spacing;
  }
}
```

BAD code: results in infinite loop

Common Problems – Infinite Loops (3)

- To fix this problem , use the `constrain()` built-in function to force the value of `spacing` stay within a certain range.





```
int x = 0, spacing = 10;
void setup() {
  size(200, 200);
}
void draw() {
  background(0);
  stroke(255);
  x = 0;
  spacing = constrain(mouseX/4,1,50);
  while (x < 200) {
    line(x, 1, x, 199);
    line(1, x, 199, x);
    x = x + spacing;
  }
}
```

Problem solved!

Common Problems – Brackets, ;

- A ***one statement loop*** ***does NOT need brackets***
  - yet, try to ***always use brackets*** to avoid problems. For example:

```
int i=0;
while (i <= 10)
    println(i);        // prints 0 (infinite loop)
    i++;               // i is not incremented inside the loop
```

- Do ***NOT*** put a ***semi-colon at the end*** of the loop:

```
int i;
for (i=0; i <= 10; i++);   // Causes empty loop
{   println(i);            // Prints 11
}
```

Common Problems – Off-By-One

- The most common error is to be ***off-by-one***. This occurs when you stop the loop one iteration too early or too late.

- Example:
  - This loop was supposed to print 0 to 10, but it does not.

```
for (i=0; i < 10; i++)
    print(i);          // Prints 0..9 not 0..10
```

- To fix this code to print 0 to 10, use <= instead of <

Common Problems – Loop Variables

- ***Scope Issues***: It is possible to declare a variable in a for loop but that variable goes out of scope (disappears) after the loop is completed.

```
int i;
for (i=0; i <= 10; i++) {
    println(i);                     // Prints 0..10
}
println(i);                         // Prints 11
```

- The other approach:

```
for (int i=0; i <= 10; i++){  // Declare i in for loop
    println(i);                     // Prints 0..10
}
println(i);   // error - i doesn’t exist outside loop
```

The do..while Loop

- The last looping structure called a **do..while** loop. The do..while loop is similar to the while loop except that the loop condition is tested at the end of the loop body.
  - This structure is useful when you know a loop must be executed at least once.
- Syntax:

```
do {
      statement
} while (condition);
```

- Example:

```
do{
   num = num / 2;
} while (num >= 0);
```

Loop Nesting

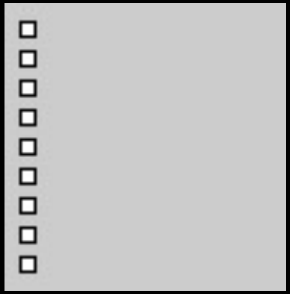
- Similar to decision statements (e.g. `if` and `switch`), it is possible to nest the `for`, `while`, and `do..while` loops.

- Be very careful to include correct brackets when nesting loops.
  - It is a good idea to always include brackets in your code to make your code more readable and prevent mistakes.

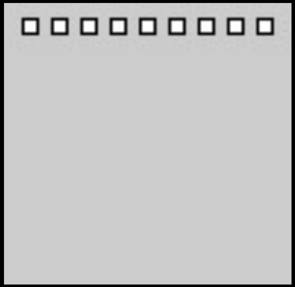
Example

Nested Loops (1)

```
rectMode(CENTER);
for (int y = 10; y < 100; y += 10)
  rect(10, y, 5, 5);
```

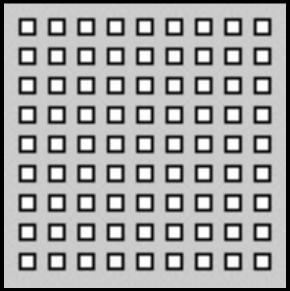


```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  rect(x, 10, 5, 5);
```



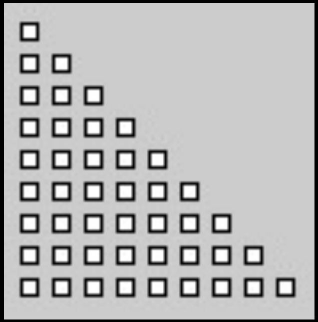
2 nested loops to fill the sketch with rectangles

```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  for (int y = 10; y < 100; y += 10)
      rect(x, y, 5, 5);
```



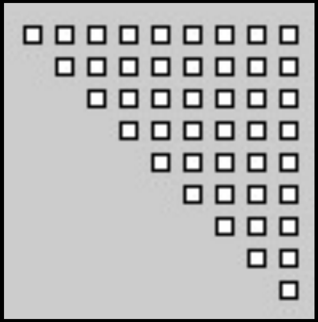
Nested Loops (2)

```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  for (int y = x; y < 100; y += 10)
      rect(x, y, 5, 5);
```



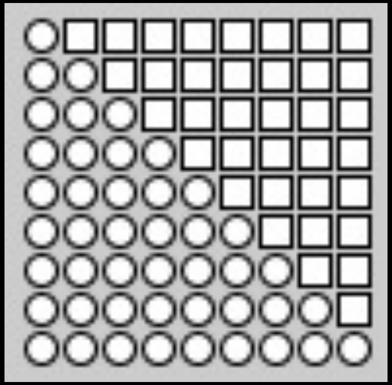
Note that we changed the limits of the inner loop

```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  for (int y = 10; y <= x; y += 10)
      rect(x, y, 5, 5);
```

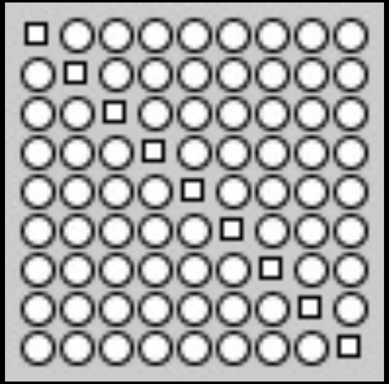


Nested Loops (3): with Selection

```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  for (int y = 10; y < 100; y += 10)
      if(x>y)
        rect(x, y, 8, 8);
      else
        ellipse(x, y, 8, 8);
```



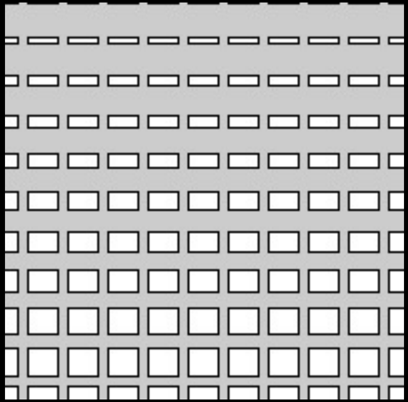
```
rectMode(CENTER);
for (int x = 10; x < 100; x += 10)
  for (int y = 10; y < 100; y += 10)
      if(x==y)
        rect(x, y, 5, 5);
      else
        ellipse(x, y, 8, 8);
```



Nested Loops (4)

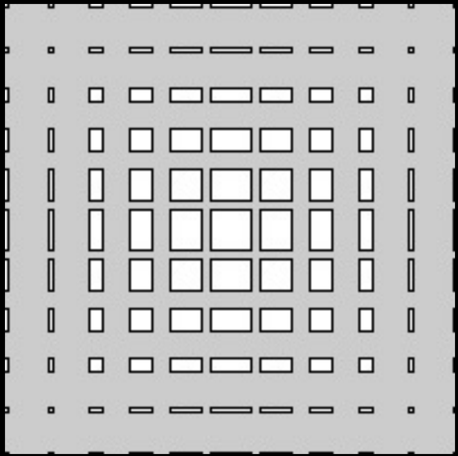
```
size(200,200);
rectMode(CENTER);
for (int x = 0; x <= 200; x += 20)
   for (int y = 0; y <= 200; y += 20)
      rect(x, y, 15 , 2+y/15.0);
```

rectangle height is proportional to it's y



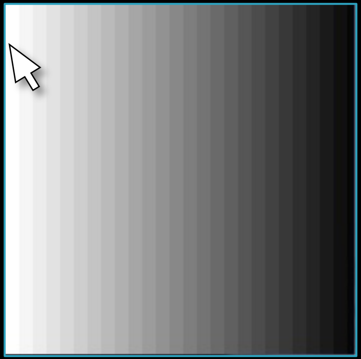
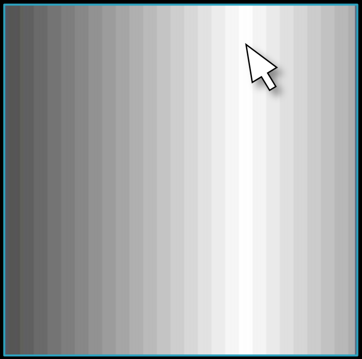
```
size(200,200);
rectMode(CENTER);
translate(100,100);
for (int x = -100; x <= 100; x += 20)
   for (int y = -100; y <= 100; y += 20)
     rect(x, y, 18-abs(x/5.0), 18-abs(y/5.0));
```

rectangle's height and width are proportional to its x & y



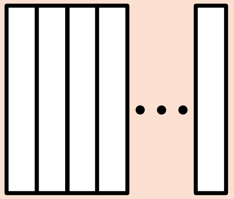
Loops and User Interactivity

- ***In a previous example***, we used `mouseX` to determine the spacing between lines drawn using a `for` loop.
- ***Here***, we display many rectangles (i = 0 to width), each filled with a gray shade computed based on the distance between the rectangle and mouseX.
  - Closer rectangle are brighter

```
void setup() {
  size(255, 255);
  noStroke();
}
void draw() {
  background(0);
  int x = 0;
  while (x < width){
    float shade = abs(mouseX - x);
    fill(255-shade);
    rect(x,0,10,height);
    x += 10;
  }
}
```

Draw many rectangles side-by-side



The color of each rectangle is based on the horizontal distance between the mouse and the rectangle

Loops review

Looping Review

- A loop structure makes the computer repeat a set of statements multiple times.
  - for loop is used when you know exactly how many iterations to perform
  - while loop is used when you keep repeating the loop until a condition is no longer true
  - a do..while loop is used when a loop has to be performed at least once
- When constructing your loop structure make sure that:
  - you have the correct brackets to group your statements
  - you do not add additional semi-colons that are unneeded
  - make sure your loop terminates (no infinite loop)
- Remember the operators ++ and -- as short-hand notation.

For Loops

What is the output of this code?

```
int i;
for (i=0; i <= 10; i++);
    print(i);
```

A. nothing

B. error

C. `11`

D. The numbers 0, 1, 2, …, 10

For Loops

What is the output of this code?

```
int i;
for (i=0; i < 10; i++)
   print(i);
```

A. nothing

B. error

C. The numbers 0, 1, 2, …, 9

D. The numbers 0, 1, 2, …, 10

For Loops

What is the output of this code?

```
int i;
for (i=2; i < 10; i--)
    print(i);
```

A. nothing

B. infinite loop

C. The numbers 2, 3, 4, …, 9

D. The numbers 2, 3, 4, …, 10

Scope Issues

What is the output of the following code?

```
void draw(){
  int shade = 0;
  background(shade);
  shade += 1;
}
```

A. The background color will gradually change from black to white

B. The background color will gradually change from white to black

C. The background color will never change

D. This code has an error and will never compile

Scope Issues

What is the output of the following code?

```
void setup(){
  int shade = 0;
}
void draw(){
  background(shade);
  shade += 1;
}
```

A. The background color will gradually change from black to white

B. The background color will gradually change from white to black

C. The background color will never change

D. This code has an error and will never compile

Scope Issues

What is the output of the following code?

```
int shade = 0;
void draw(){
  background(shade);
  shade += 1;
}
```

A. The background color will gradually change from black to white

B. The background color will gradually change from white to black

C. The background color will never change

D. This code has an error and will never compile

Loops vs. draw()

Which code is better? We want the ball to **gradually** move from left to right.

(1)

```
int x = 0;
void draw(){
  background(0);
  ellipse(x,50,20,20);
  x++;
}
```

(2)

```
for(int x = 0; x < 100; x++){
  background(0);
  ellipse(x,50,20,20);
}
```

A. (1) is better than (2)

B. (2) is better than (1)

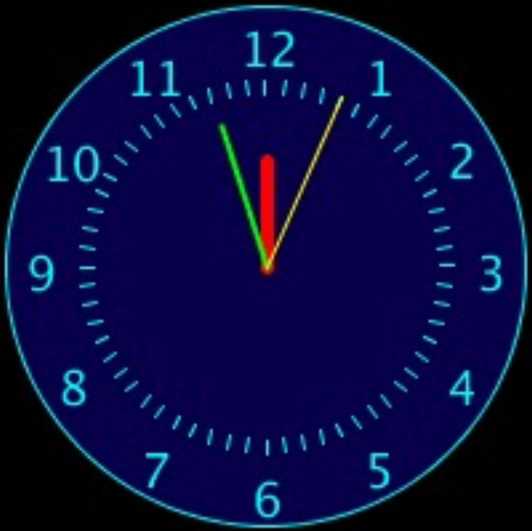
C. They are both the same

D. I don't understand what you are talking about.

Analogue Clock *(version 4 (?))*

- Update your version 2 of the analogue clock code to look similar to the given figure.
- Download this <u>**starter code**</u> then use ***loops*** to
  - Put all the numbers from 1 to 12
  - Put all the little minute ticks (dashes) around the clock.

Follow the instructions in the starter code to finish your animation.

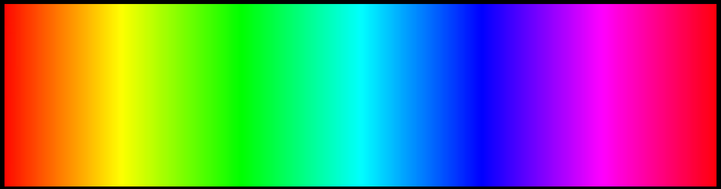


Fast replay

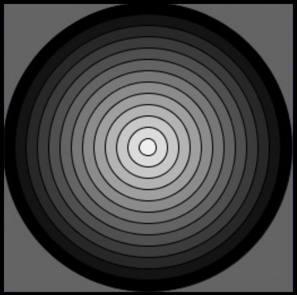
`while` Loops

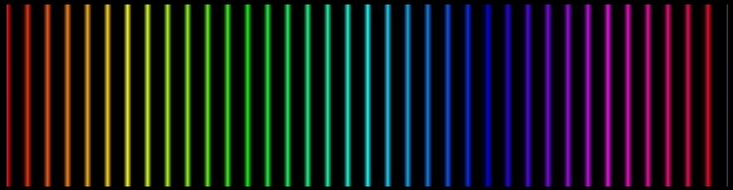
Write code that uses `while` loops to generate the following sketches.

(A)

(B) 

***Hint**: above sketches can be produced by drawing many adjacent circles (for A) or lines (for B) while slightly changing the color. For example, the illustrations below are the same as the above except that the **spacing** between the circles/lines is larger.*





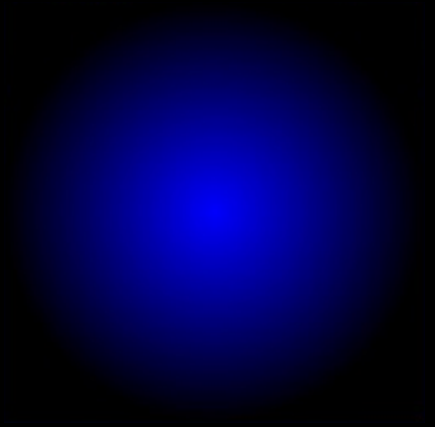
(*You can* re-use some code from the pre-class readings)

Lecture Activity Task

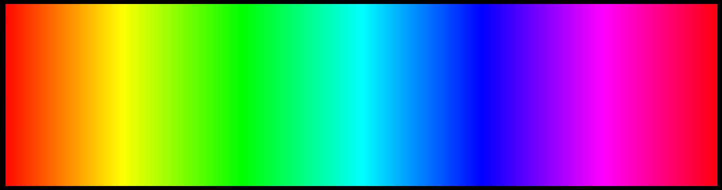
for Loops

Repeat this exercise again using for loops.

(A)

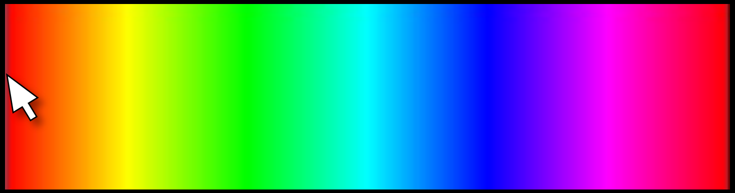


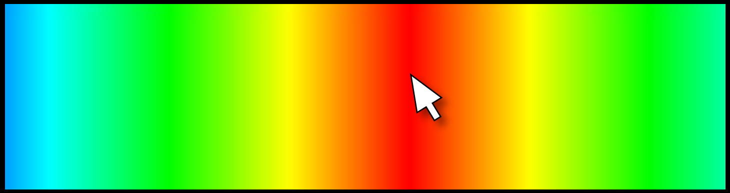
(B)



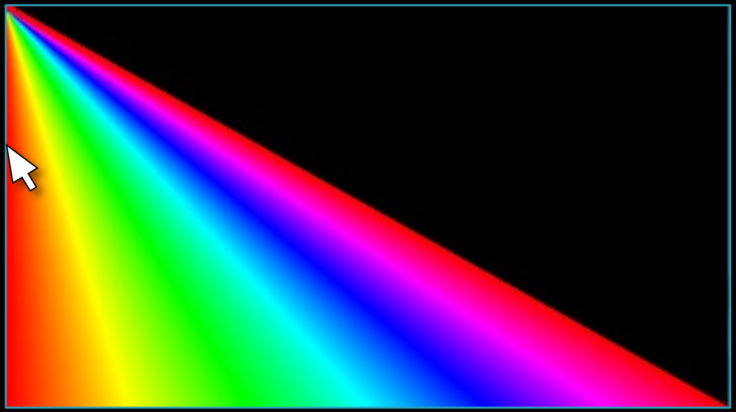
Loops and User Interactivity

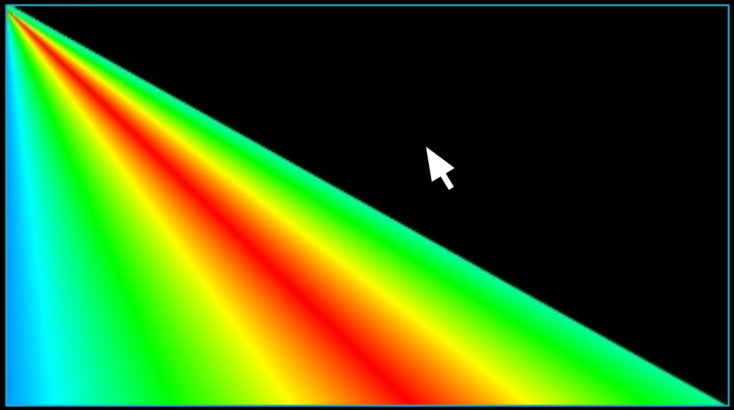
A) Based on the idea from previous slide, write code to show all colors as in the example below – note that the red color follows the mouse (hint: use HSB, refer to last program in pre_class notes)


Size: 360 x 100


Size: 360 x 100

B) Change your above code so that the output is similar to below

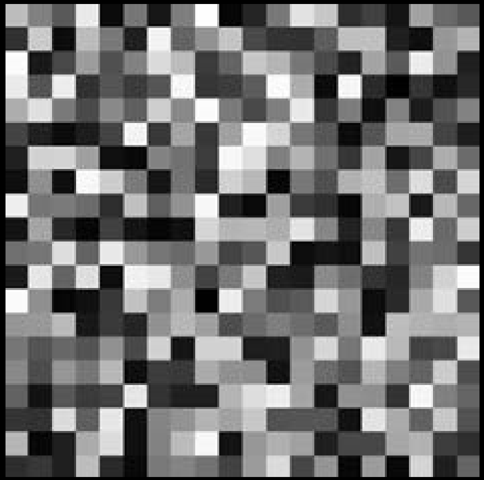

Size: 360 x 200


Size: 360 x 200

Nested `for` Loops

A) Shiffman Exercise 6-8:

- 1) Create a grid of squares, each colored randomly, using a `for` loop.
- 2) Recode the same pattern using a `while` loop instead of `for`.



B) Generate the grid of squares in the lower figure. Note that

- each square is colored in a way that the window has a gradient shading.
- each square has an outline that is slightly lighter in color than the fill color.