Welcome!

Instructor: Nina Freeman

openFrameworks and C++
Week 2

Quick review!

What is openFrameworks?

What's a variable?

How do you declare and initialize a variable?

What are some of the C++ data types?

Where can you see a variable's data type?

What are some of the C++ arithmetic operators?

Relational and Equality Operators

==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

== Equal to

5 == 5

True

1 == 2

False

Returns true if both expressions are equivalent. Otherwise, returns false.

!= Not equal to

5 != 4

True

1 != 1

False

Returns true if both expressions are NOT equivalent. Otherwise, returns false.

> Greater than

5 > 2

True

8 > 10

False

Returns true if the expression on the left is greater than the right. Otherwise, returns false.

< Less than

2 < 5

True

8 < 1

False

Returns true if the expression on the left is less than the right. Otherwise, returns false.

>= Greater than or equal to

2 >= 2

True

5 >= 10

False

Returns true if the expression on the left is greater than or equal to the right. Otherwise, returns false.

<= Less than or equal to

2 <= 5

True

8 <= 1

False

Returns true if the expression on the left is less than or equal to the right. Otherwise, returns false.

Logical Operators

! Not

&& And

|| Or

! Not

!(5 == 5) evaluates to false because the expression at its right (5 == 5) is true.

!(6 <= 4) evaluates to true because (6 <= 4) would be false.

!true evaluates to false
!false evaluates to true.
It returns the opposite boolean value of the resulting expression!

&& And

5 > 1 && 7 > 1
True because both expressions are true.

7 <= 5 && 8 > 3
False because 7 is not less than or equal to 5.

Checks to see if two expressions both evaluate to true. If one or both are false, then it returns false.

|| Or

5 == 5 || 7 == 1
True because 5 == 5 is true.

6 > 7 || 8 == 0
False because both expressions are false.

If at least one expression is true, then it returns true. If both are false, it returns false.

Review!

Tell me if the following expressions are TRUE or FALSE.

1) 5 > 1

2) 80 < 3

3) 5 == 6

4) 3 >= 8 && 1==1

5) !(8 == 0)

6) 5 < 6 || 6 <= 7

What're all these logic things and operands useful for?

If statements!

Do I like ice cream?

If True

↓

eats ice cream

If False

↓

does not eat ice cream

If statements check to see if a condition is true, and if it is, executes an action.

If not the program would skip the action and do whatever comes next.

```
if(condition){
  //some statement
}

//the program continues
```

Try writing an if statement in draw{} that makes a circle if the expression is true.

Use those operands we learned earlier to make an expression!

Here's the outline again:

```
if(condition){
  //draw a circle
}
```

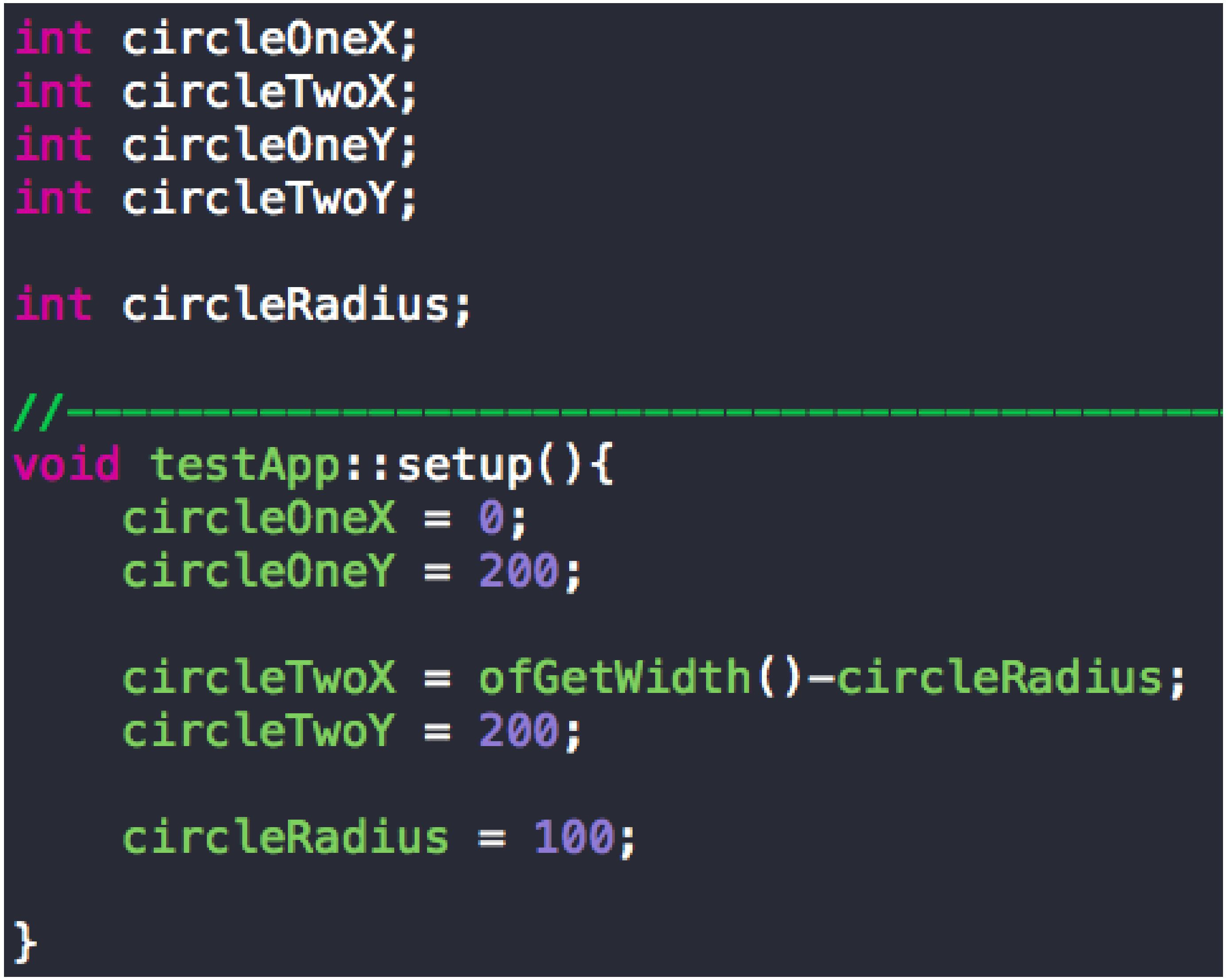
You can also specify a statement to execute if the condition is false by using an **if else statement**.

```
if(condition){
  //statement 1
}else{
  //statement 2
}

//program continues
```

Now we'll draw some moving circles that get bigger when they overlap.

Our circles will also move forever, and not go off the screen.

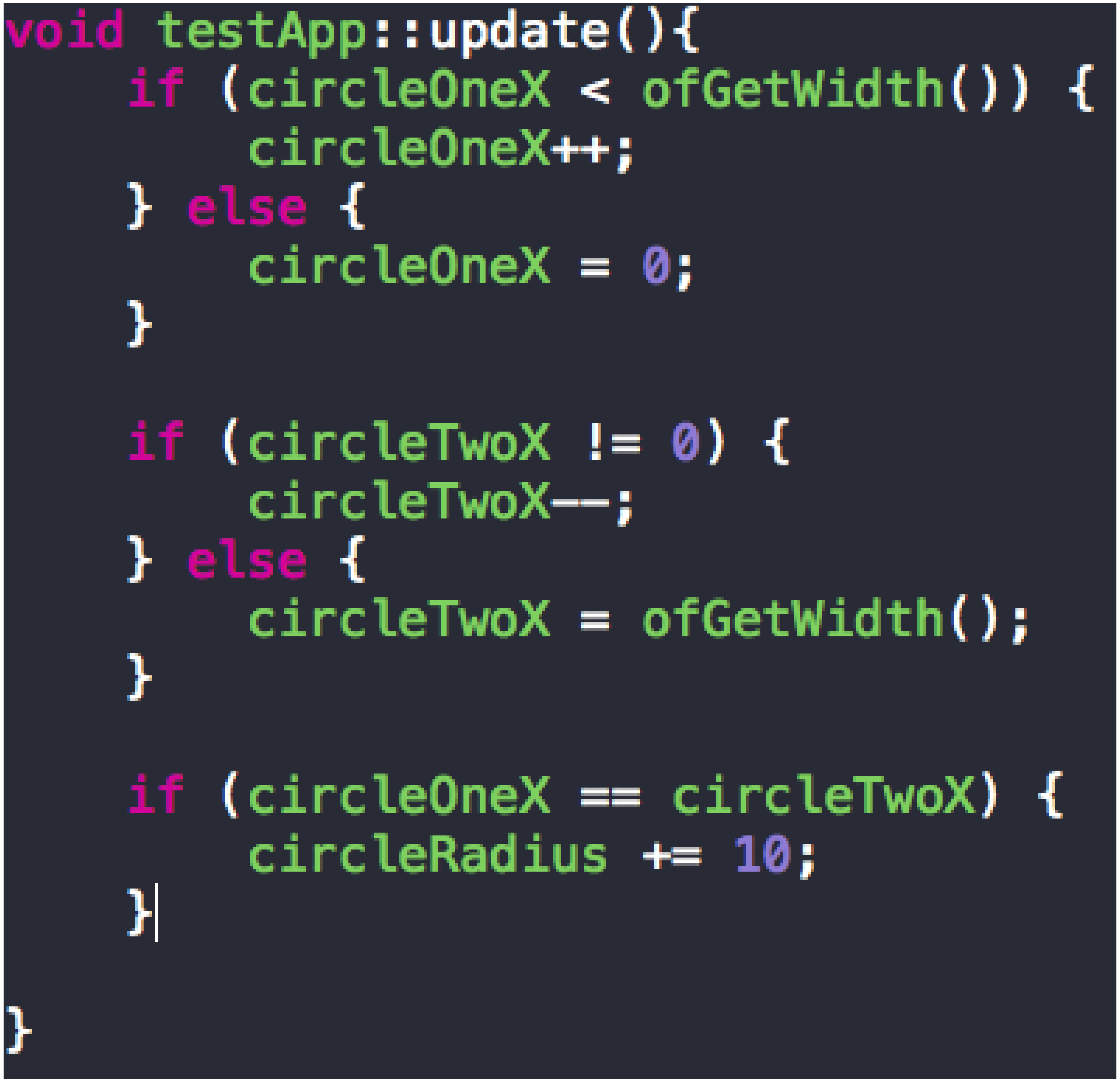
Declare and initialize your variables.



```
int circleOneX;
int circleTwoX;
int circleOneY;
int circleTwoY;

int circleRadius;

//-----------------------------------------
void testApp::setup(){
    circleOneX = 0;
    circleOneY = 200;

    circleTwoX = ofGetWidth()-circleRadius;
    circleTwoY = 200;

    circleRadius = 100;

}
```

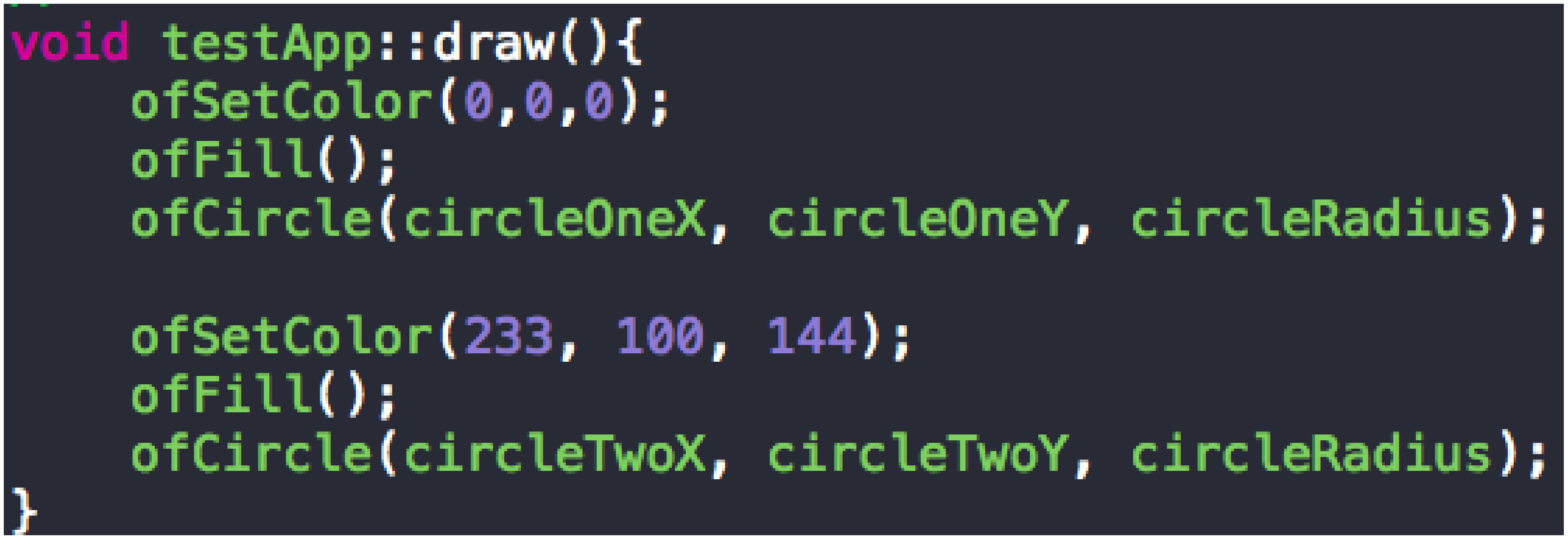
Our circles should move and go back to their original positions when they go off screen.

The radius should get bigger when they overlap.



```
void testApp::update(){
    if (circleOneX < ofGetWidth()) {
        circleOneX++;
    } else {
        circleOneX = 0;
    }

    if (circleTwoX != 0) {
        circleTwoX--;
    } else {
        circleTwoX = ofGetWidth();
    }

    if (circleOneX == circleTwoX) {
        circleRadius += 10;
    }

}
```

Finally, let's draw those circles!



```
void testApp::draw(){
    ofSetColor(0,0,0);
    ofFill();
    ofCircle(circleOneX, circleOneY, circleRadius);

    ofSetColor(233, 100, 144);
    ofFill();
    ofCircle(circleTwoX, circleTwoY, circleRadius);
}
```

Review!

What's an if statement?

Any ideas as to how an if statement might be used in a game?

Give me an example of one relational, one equality and one logical operator.

You can have many else clauses in an if else statement.

```
if(condition1){
  //statement 1
}else if(condition2){
  //statement 2
}else if(condition3){
  //statement 3
}else {
  //statement 4
}
```

If statements can be nested.

If condition1 is true: statement 1 will run and then condition2 will be checked.

If condition2 is also true then the nested action will run.

```
if(condition1){
  //statement 1
  if(condition2){
    //nested stmnt
  }
}else{
  //statement 2
}
```

If statements can be nested.

What do you think happens if condition1 or 2 are false?

If condition1 is false, then the program will skip to the else, and the nested if statement is skipped.

If condition1 is true and 2 is false, then statement 1 will run, followed by the else statement.

```
if(condition1){
  //statement 1
  if(condition2){
    //nested stmnt
  }
}else{
  //statement 2
}
```

If statements are useful for keeping track of game states. Here's an example!

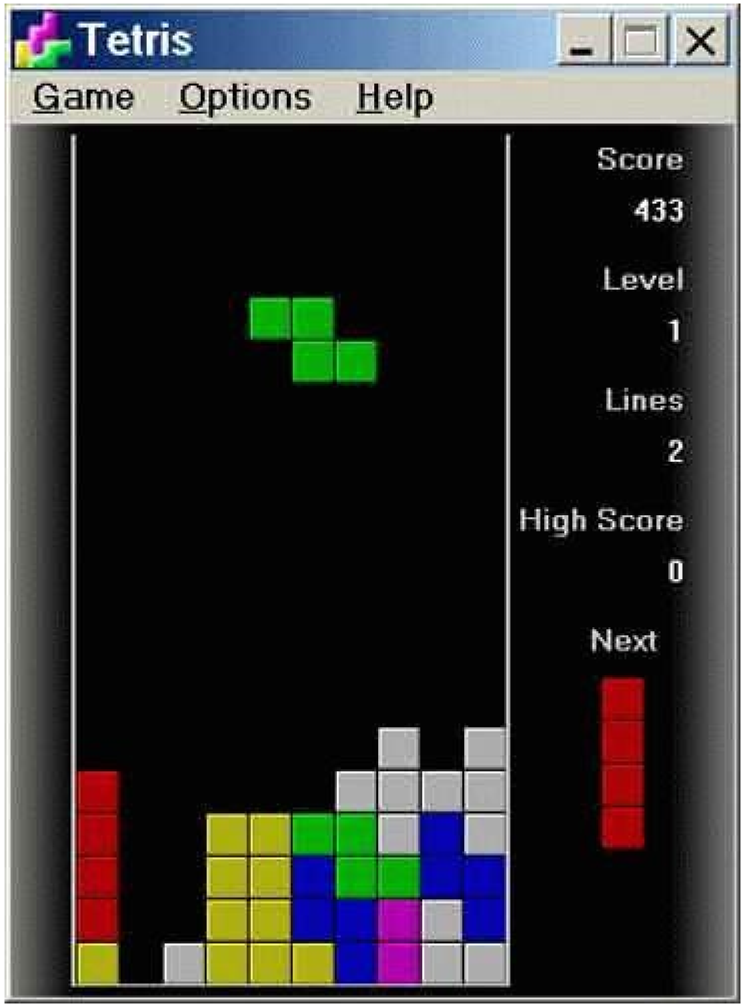
A game that runs until a player reaches 10 points:

Have a “playing” boolean variable set to true.

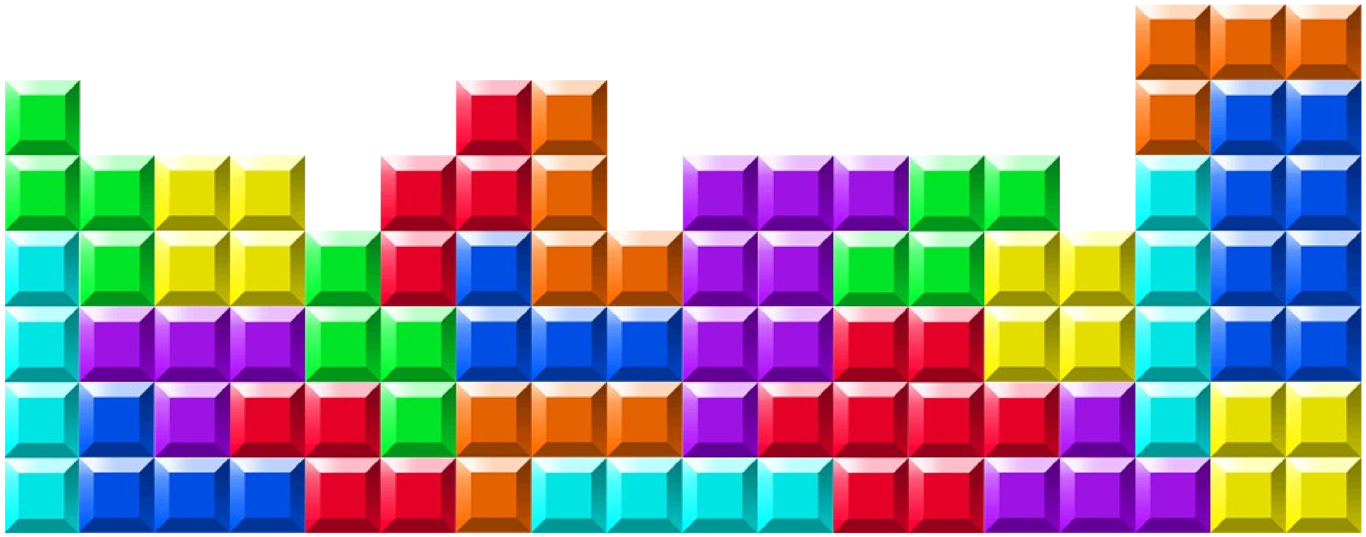
In update, check if the player has hit 10 points. If yes, then set “playing” to false.

Have another if statement that checks if “playing” is false, and if so, switch to the game over screen.

How does Tetris know when it's game over?



While the blocks are not hitting the top of the screen, keep dropping new blocks. Otherwise, the game ends.



Review

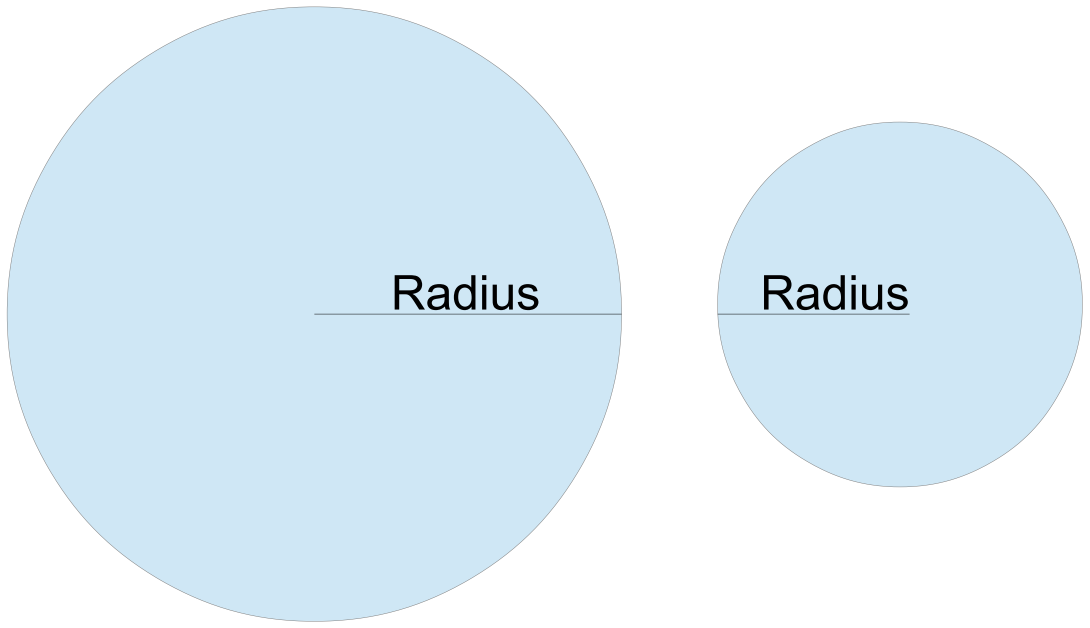
What's the syntax for an if statement?

What's a nested if statement?

When does the code in an else clause execute?

Collision Detection

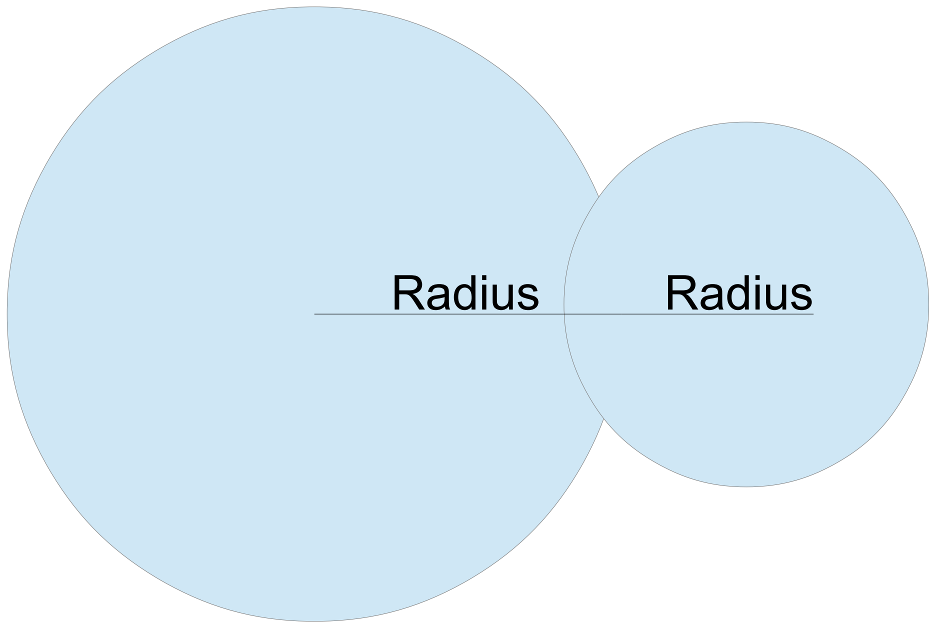
How can I tell when two circles are touching?



Not colliding!

Collision Detection

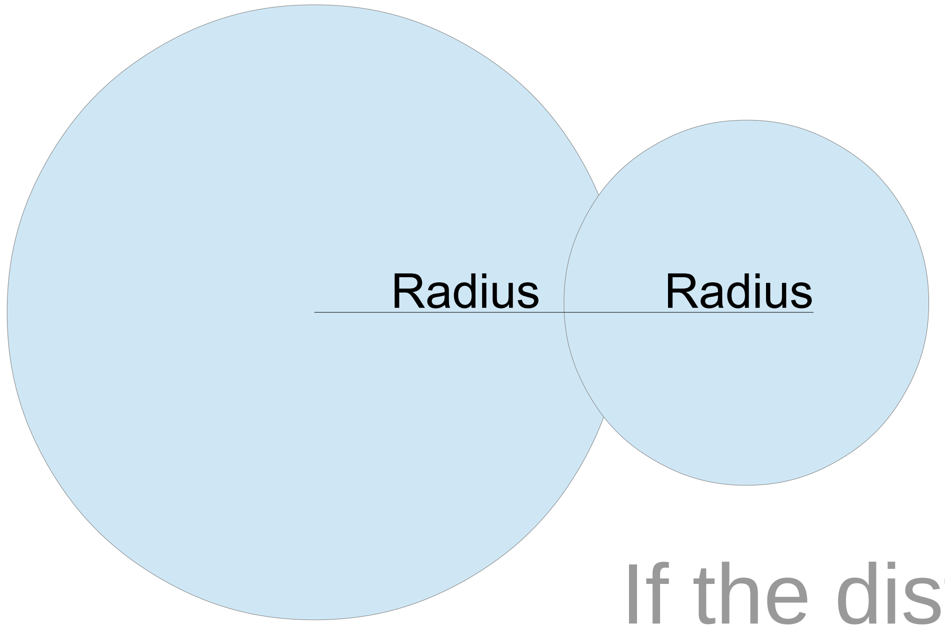
How can I tell when two circles are touching?



Colliding!
But how... in code?

Collision Detection

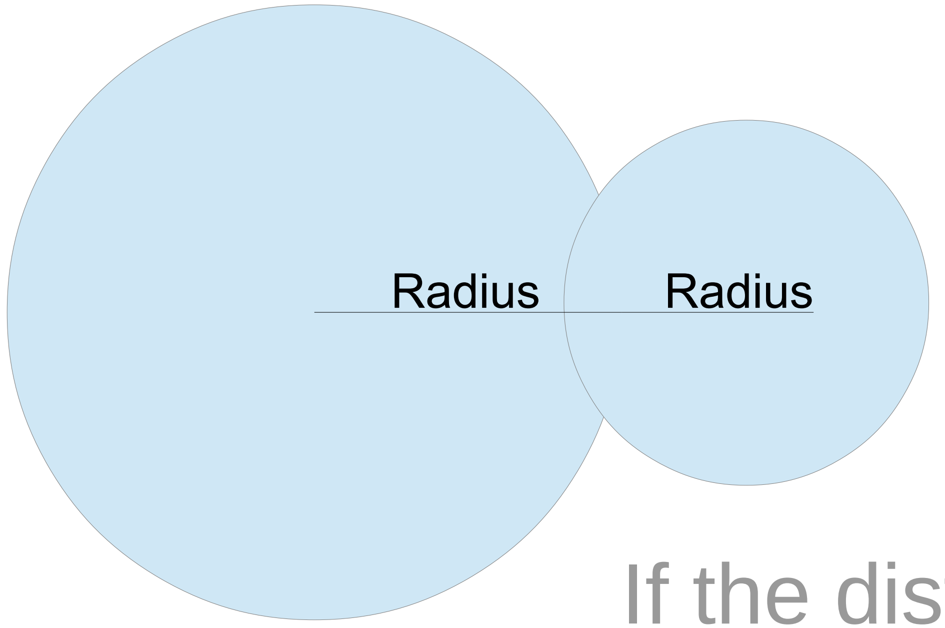
How can I tell when two circles are touching?



If the distance between the center-points of the circles is less than the sum of their radii, they have collided!

Collision Detection

What variables would we need to write code for this?



If the distance between the center-points of the circles is less than the sum of their radii, they have collided!

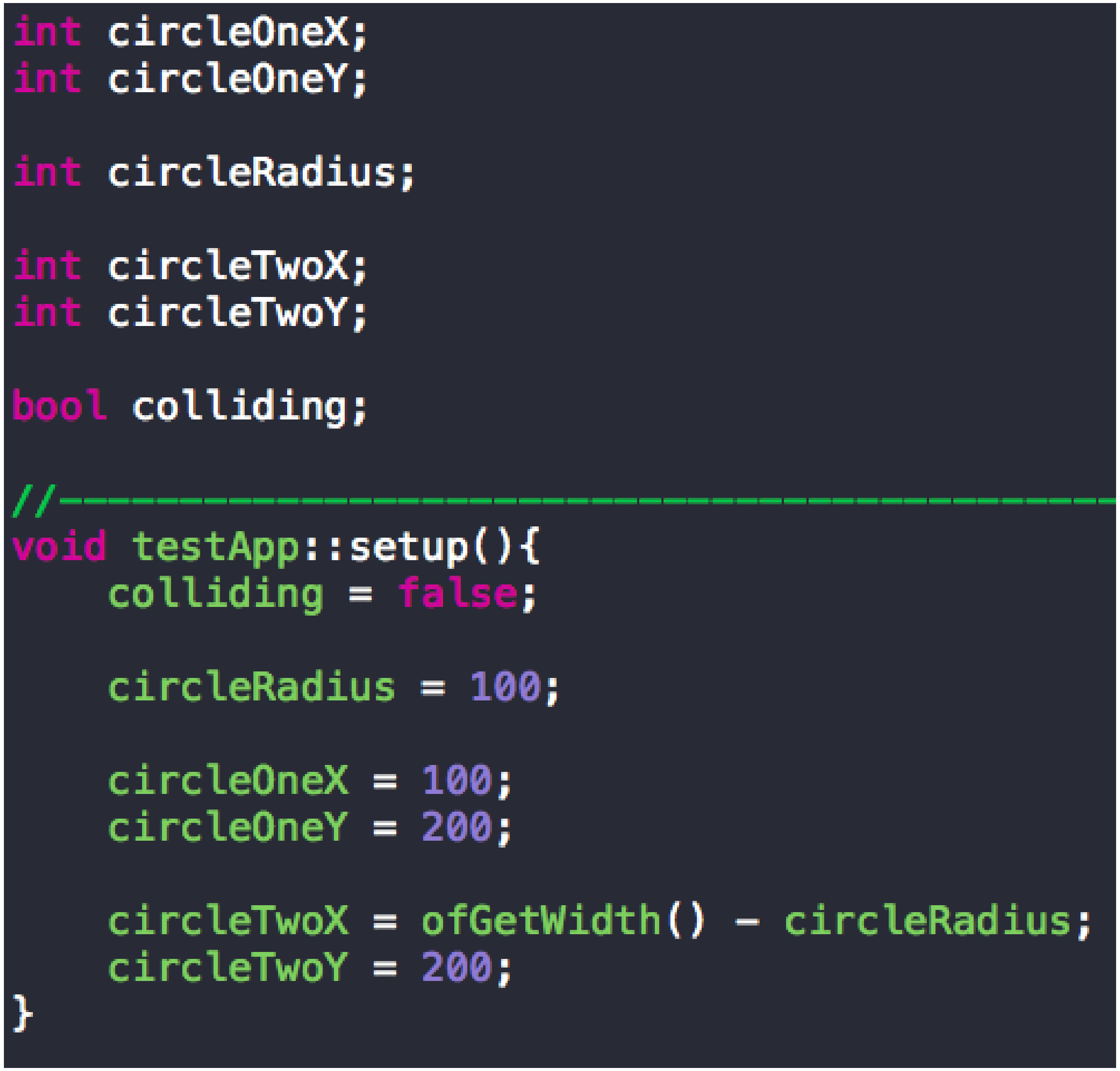
Declare and initialize the following variables:

Data type: int

```
circleOneX = 100;
circleOneY = 200;

circleTwoX = ofGetWidth() - circleRadius;
circleTwoY = 200;

circleRadius = 100;
```

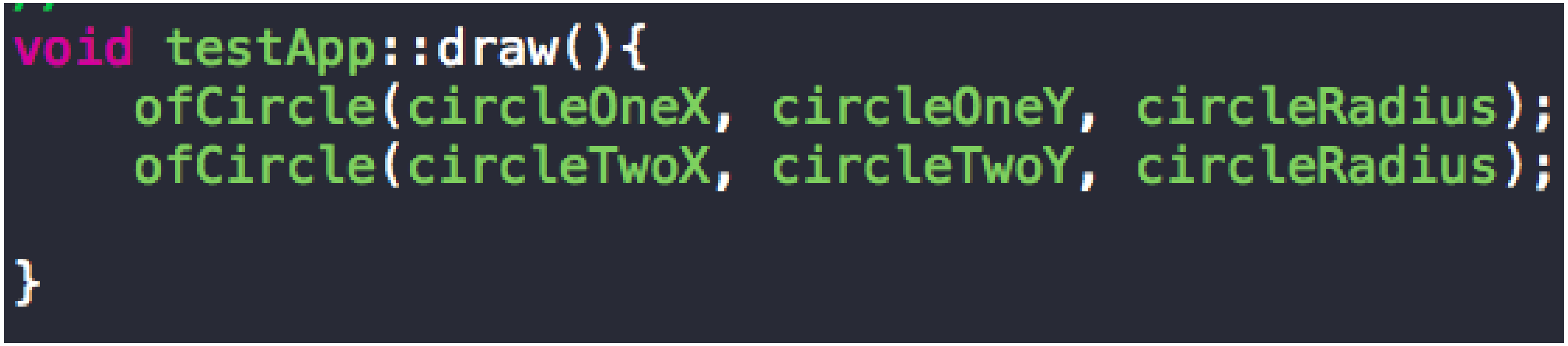
Data type: bool

```
colliding = false;
```



```
int circleOneX;
int circleOneY;

int circleRadius;

int circleTwoX;
int circleTwoY;

bool colliding;

//-------------------------------------------
void testApp::setup(){
    colliding = false;

    circleRadius = 100;

    circleOneX = 100;
    circleOneY = 200;

    circleTwoX = ofGetWidth() - circleRadius;
    circleTwoY = 200;
}
```

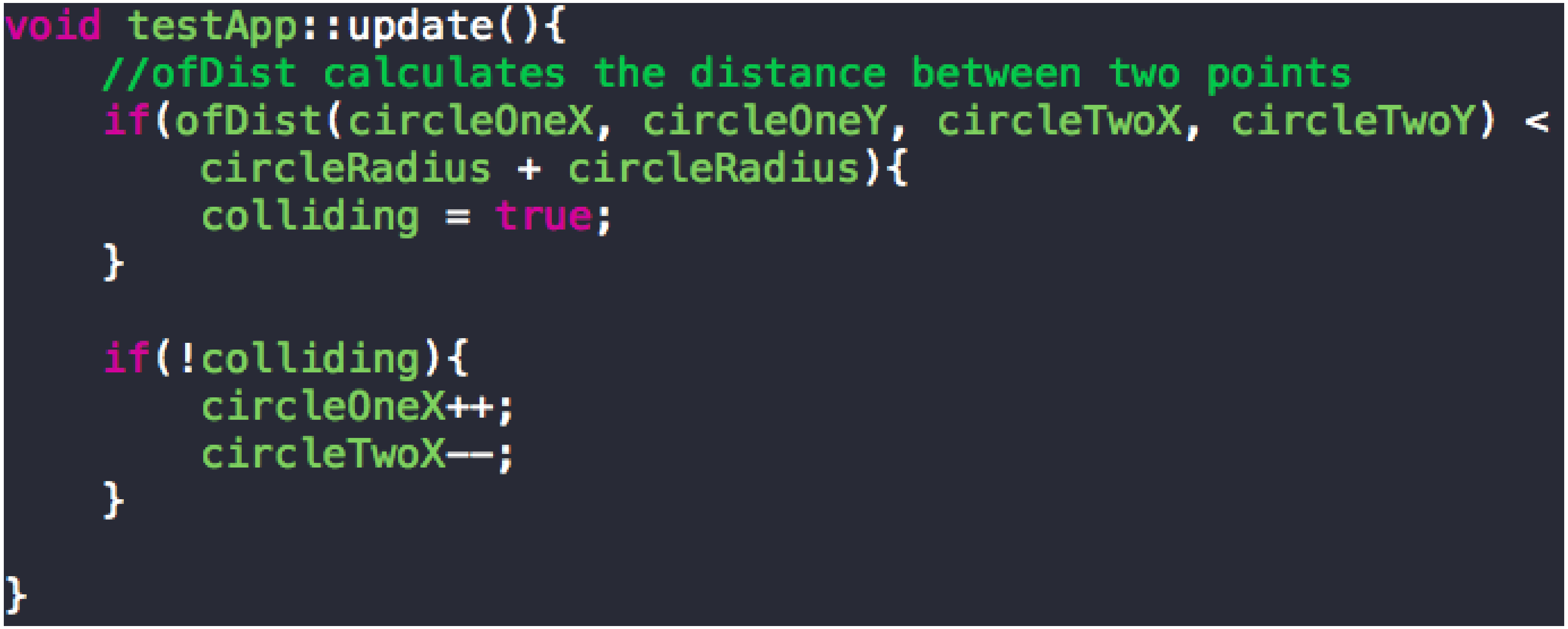
Using your variables and ofCircle(), draw two circles to the screen.



```
void testApp::draw(){
    ofCircle(circleOneX, circleOneY, circleRadius);
    ofCircle(circleTwoX, circleTwoY, circleRadius);

}
```

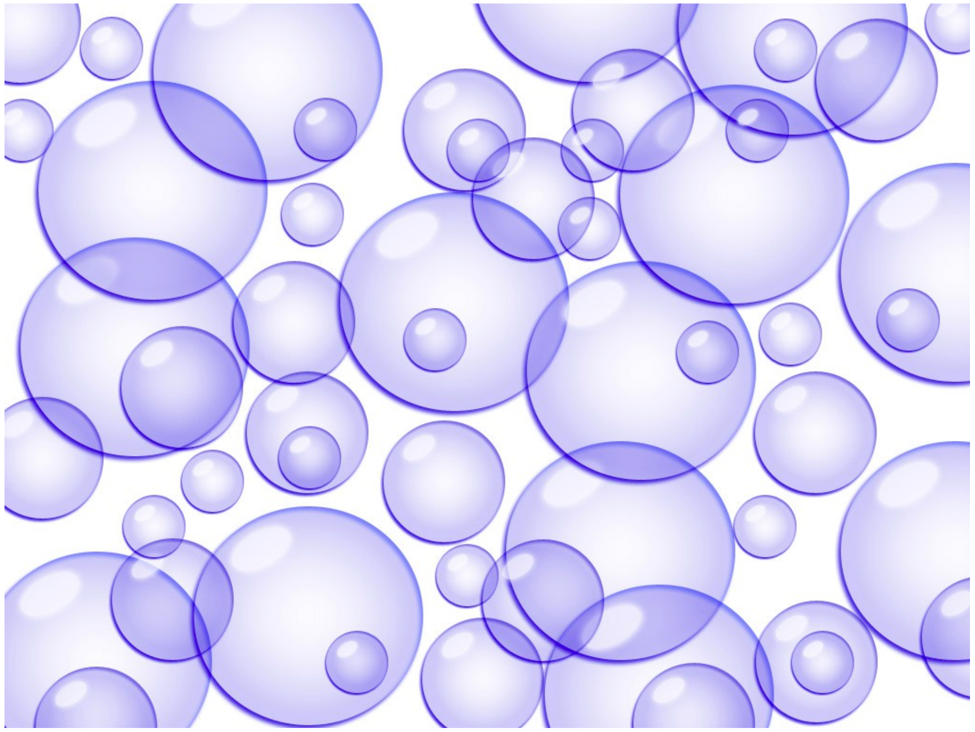
Now let's make them move towards each other, and when they touch, stop them.

How might we do this? Can you tell me the logic in plain english?



```
void testApp::update(){
    //ofDist calculates the distance between two points
    if(ofDist(circleOneX, circleOneY, circleTwoX, circleTwoY) <
        circleRadius + circleRadius){
        colliding = true;
    }

    if(!colliding){
        circleOneX++;
        circleTwoX--;
    }

}
```

Cool, so now we know how to make variables, and draw circles that can move, grow and collide... what if we want to draw 10 circles? Or 20?



For Loops

For Loops repeat a statement while a condition remains true.

Used to perform a repetitive action based on a counter which is initialized and increased on each iteration.

It’s good for code that you want to repeat a specific number of times.

```
for(initialization; condition; increase){
   //statement
}
```

For Loops

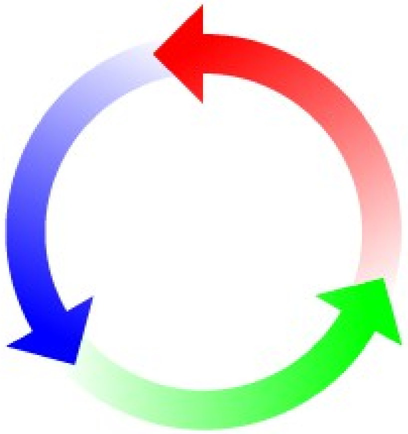
Syntax

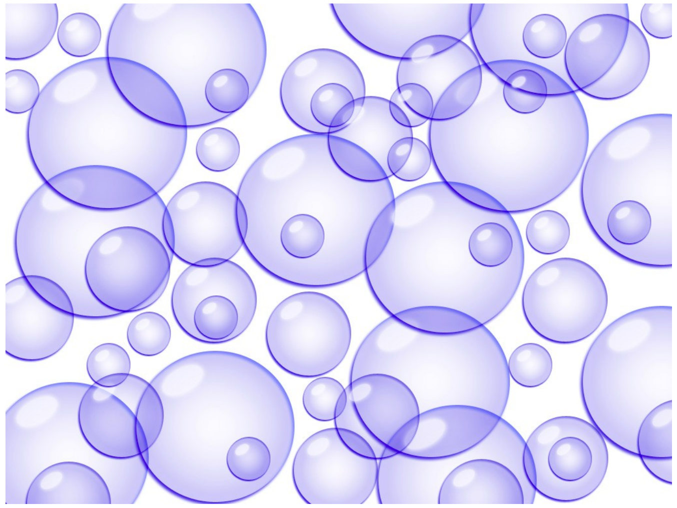
```
for(initialization; condition; increase){
   //statement
}
```

Real Code

```
for(int i = 0; i < 5; i++){
  //this statement executes
  //5 times
}
```

For loops are great for repetitive actions. They're especially good at iterating code over multiple objects...

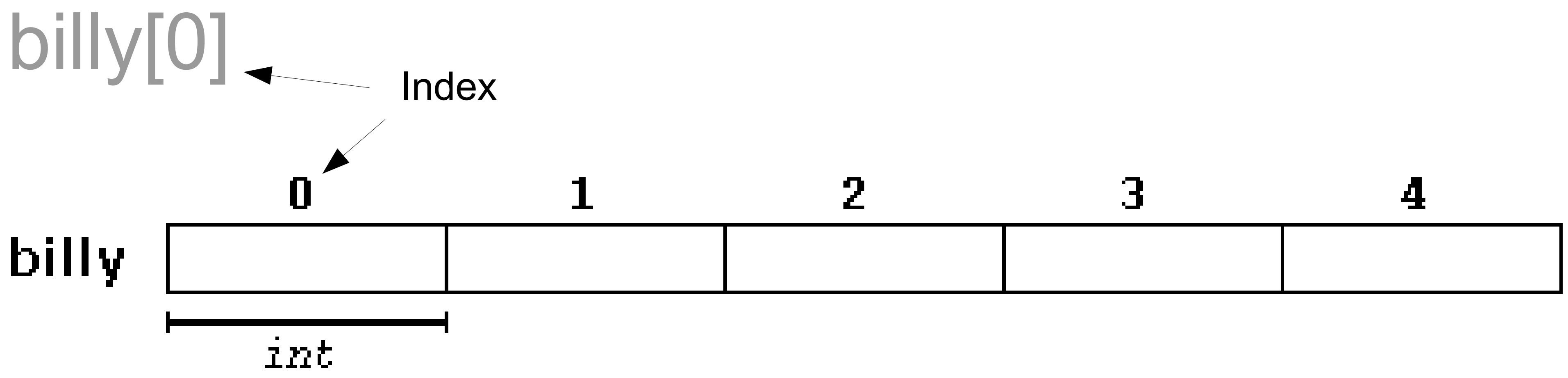




Array

An array is a container that holds a series of elements of the same type.

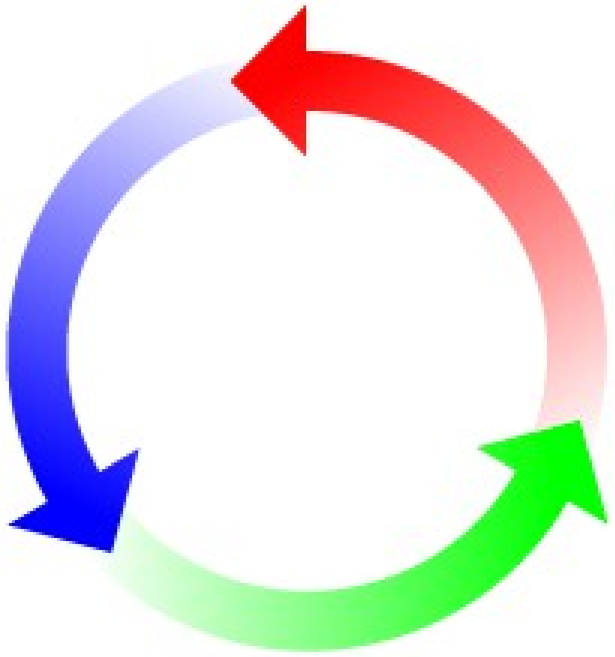
Each element can be individually referenced by their index.

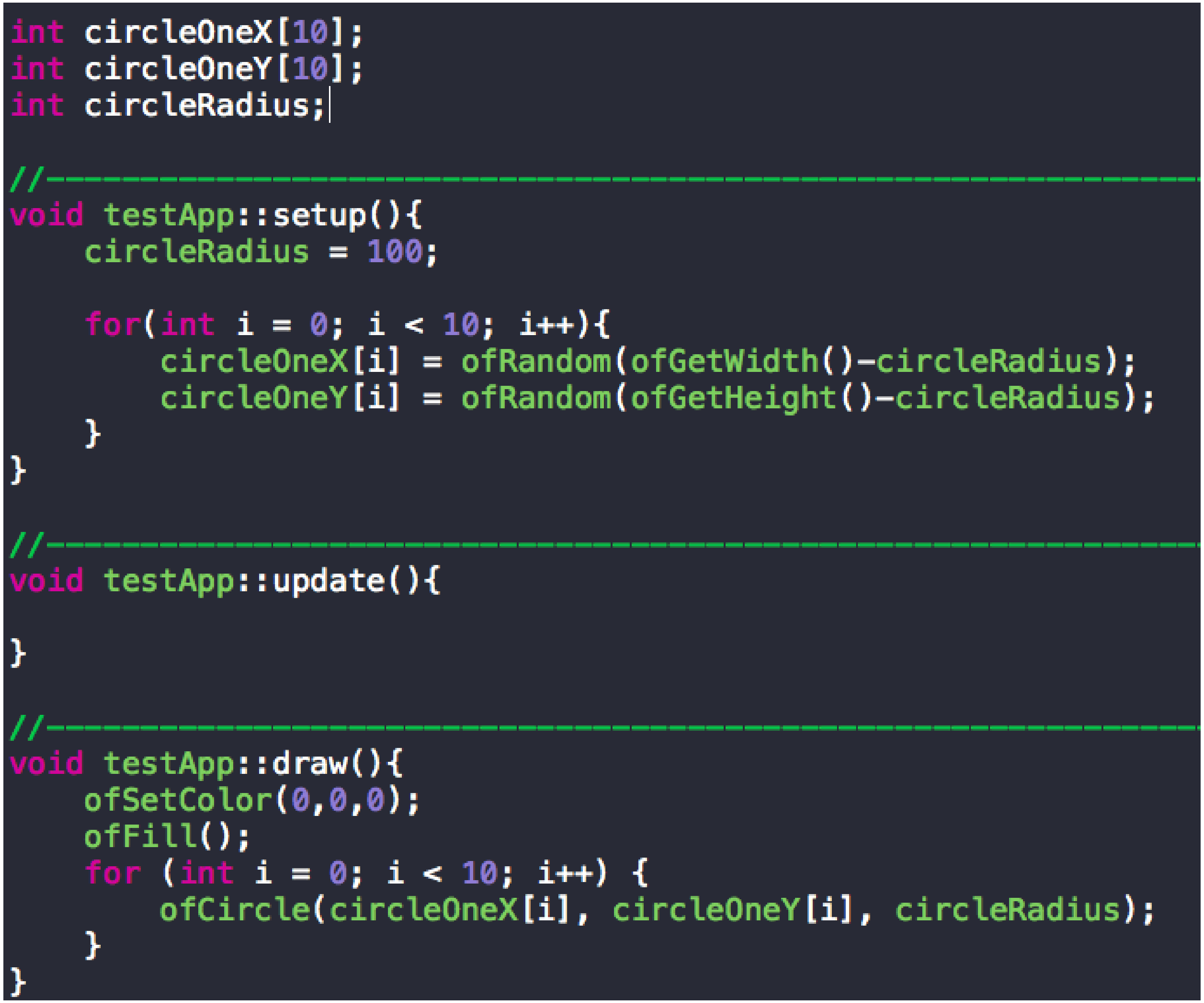


Array

Good for objects that need to be grouped together.

Let's try using a for loop to execute code on each element in an array.





```
int circleOneX[10];
int circleOneY[10];
int circleRadius;

//--------------------------------------------------------------------
void testApp::setup(){
    circleRadius = 100;

    for(int i = 0; i < 10; i++){
        circleOneX[i] = ofRandom(ofGetWidth()-circleRadius);
        circleOneY[i] = ofRandom(ofGetHeight()-circleRadius);
    }
}

//--------------------------------------------------------------------
void testApp::update(){

}

//--------------------------------------------------------------------
void testApp::draw(){
    ofSetColor(0,0,0);
    ofFill();
    for (int i = 0; i < 10; i++) {
        ofCircle(circleOneX[i], circleOneY[i], circleRadius);
    }
}
```

Review!

What's an if statement?

What's a variable and give me an example of a data type.

Give me an example of one relational, one equality and one logical operator.

What are for loops good for? How about arrays?