C++ and openFrameworks

Week 4

Instructor: Nina Freeman

Review!

What is a function and what is it good for?

What are some examples of functions we have used?

What are function parameters? What about arguments?

What are return values?

What does it mean to “pass by value”?

What is abstraction? What about encapsulation?

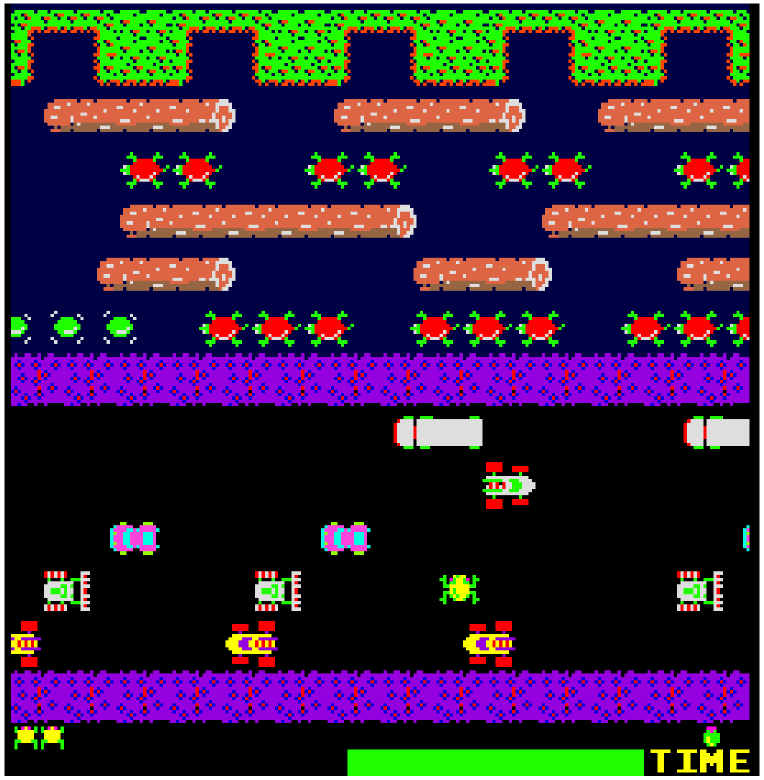
Final Presentations!

Design and present a working prototype using openFrameworks.

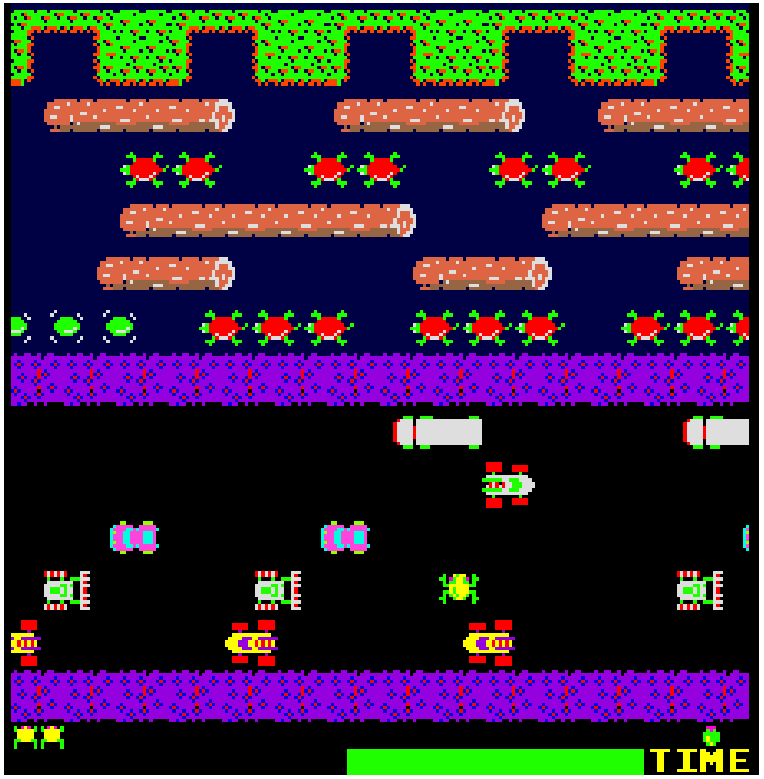
Focus on the game mechanic and use simple circles to build your idea.

But first...

Let's build a simple prototype inspired by Frogger to get ourselves ready!



Create a fresh (and clean) openFrameworks project. We're going to build a game!



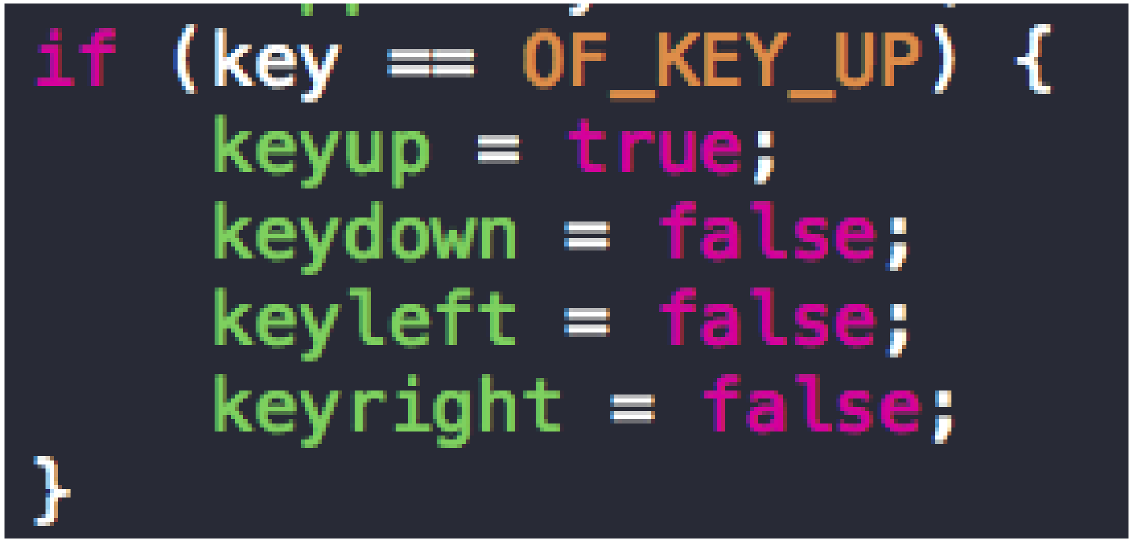
Declare and define these variables:

Name	Type	Value
playerX	float	ofGetWidth()/2
playerY	float	10
playerSpeed	float	3
playerRadius	int	20
keyup, keydown, keyleft, keyright (4 separate vars)	bool	false

Draw the player to the screen using playerX, playerY and playerRadius in Draw.

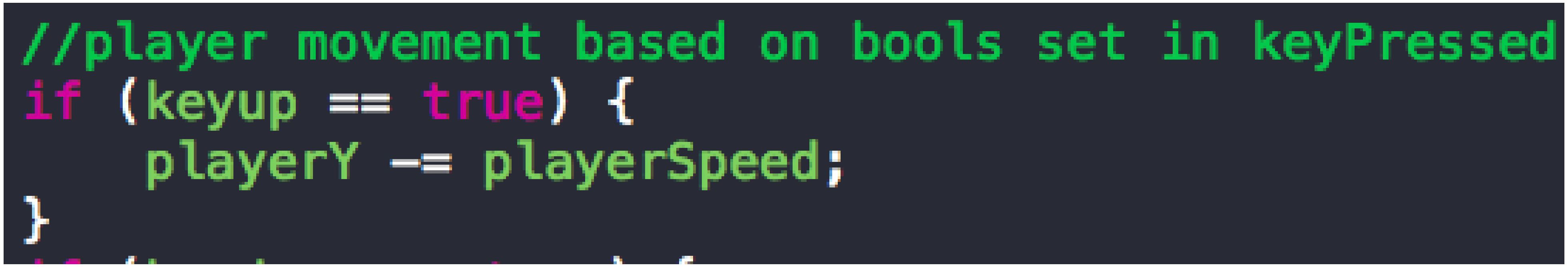
Hint: Use the oF function for drawing circles!

Given the following if statement, write the corresponding if statements for DOWN, LEFT, and RIGHT in keyPressed.



```
if (key == OF_KEY_UP) {
    keyup = true;
    keydown = false;
    keyleft = false;
    keyright = false;
}
```

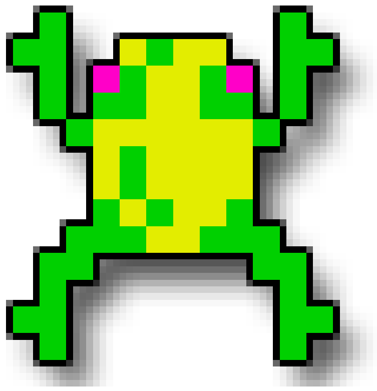
Given the following if statement, write the corresponding if statements for DOWN, LEFT, and RIGHT in Update.



```
//player movement based on bools set in keyPressed
if (keyup == true) {
    playerY -= playerSpeed;
}
```

You should have a moving player!

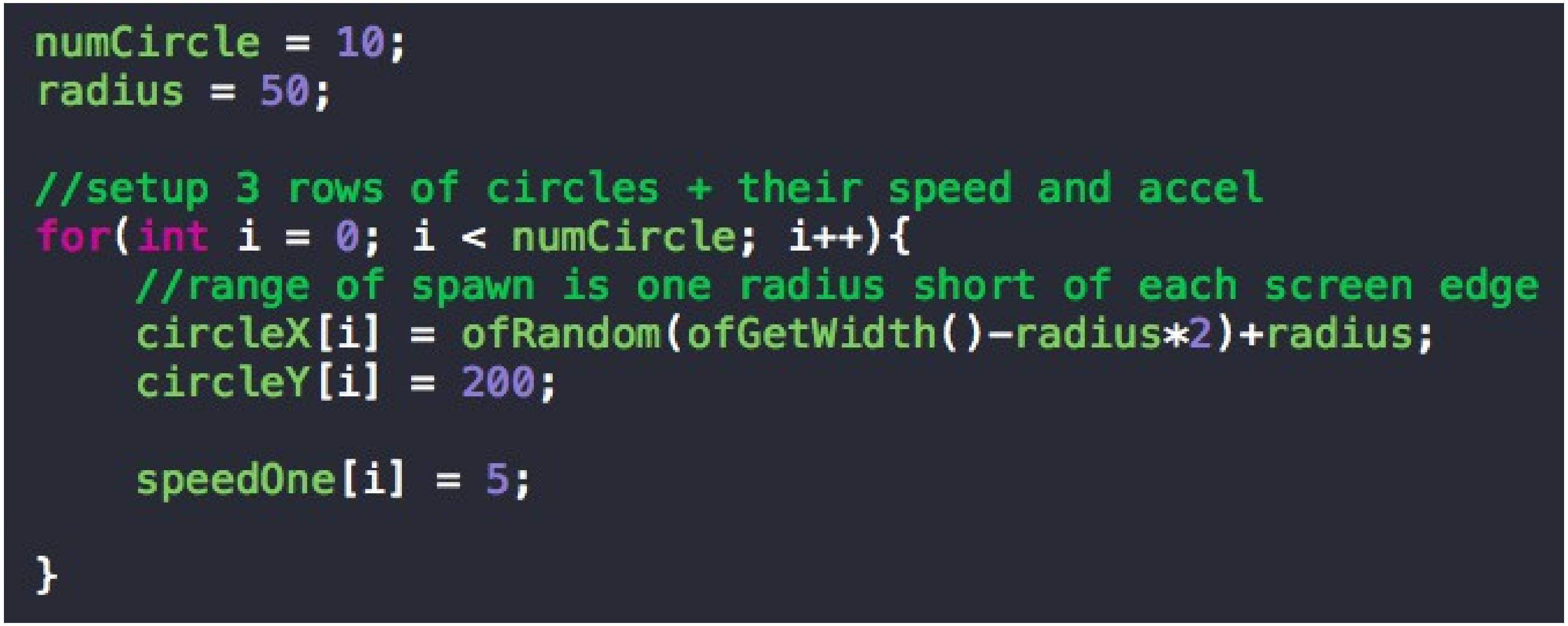
Yay!



Declare and define these variables:

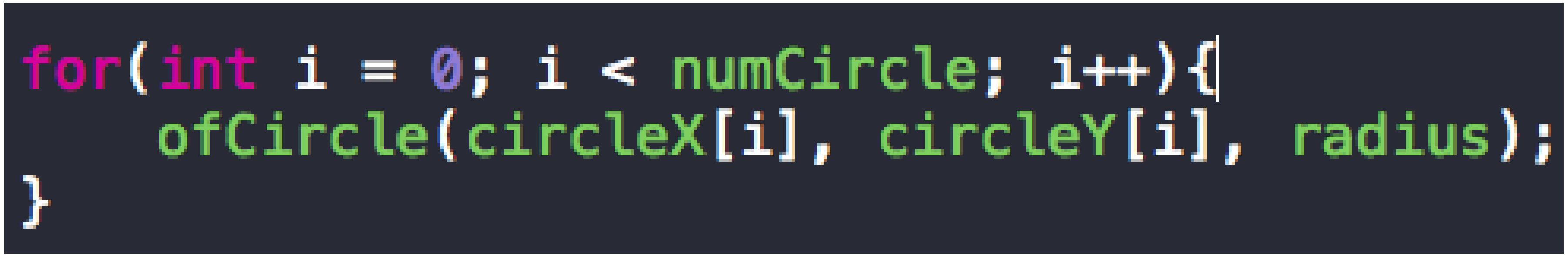
Name	Type	Value
circleX	float	Array, size 10 (we'll declare together)
circleY	float	Array, size 10 (we'll declare together)
speedOne	float	Array, size 10 (we'll declare together)
radius	int	50
numCircle	int	10

Let's use a for loop to build our first row of circles in Setup!



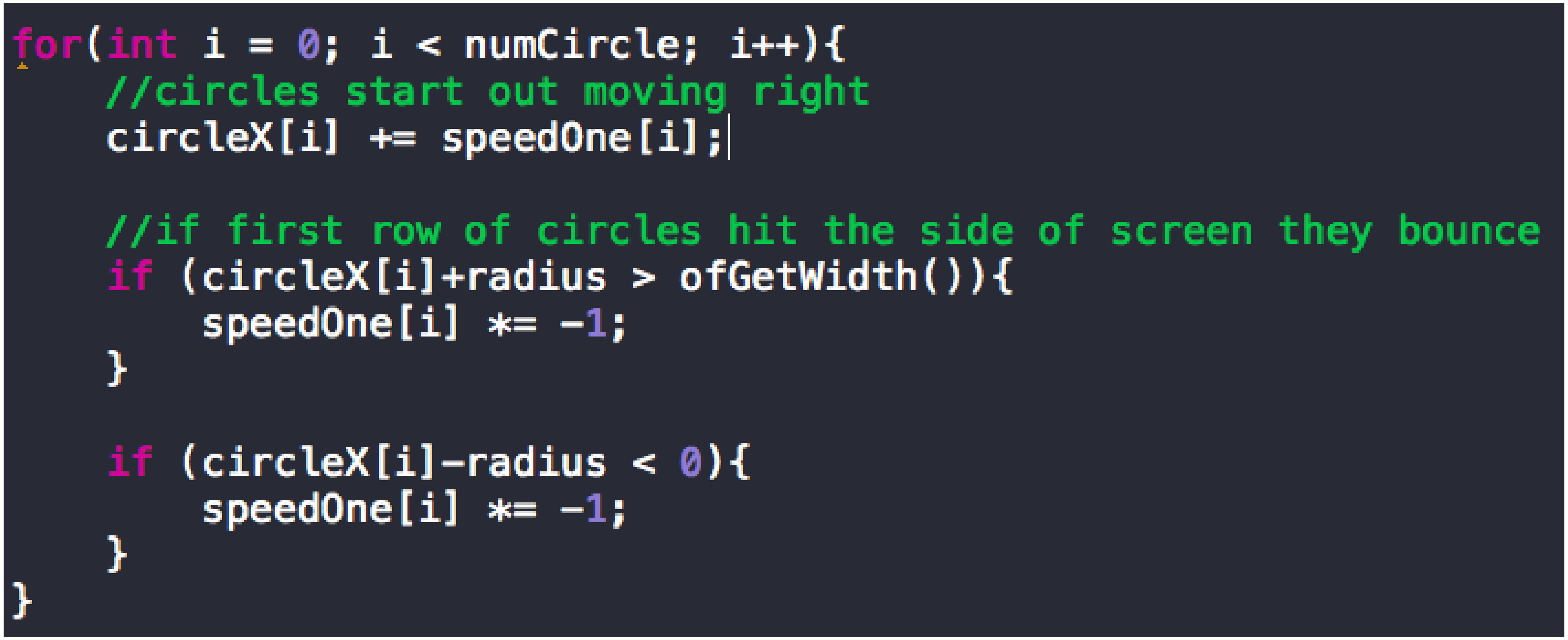
```
numCircle = 10;
radius = 50;

//setup 3 rows of circles + their speed and accel
for(int i = 0; i < numCircle; i++){
    //range of spawn is one radius short of each screen edge
    circleX[i] = ofRandom(ofGetWidth()-radius*2)+radius;
    circleY[i] = 200;

    speedOne[i] = 5;

}
```

Let's use a for loop to draw our first row of circles in Draw!



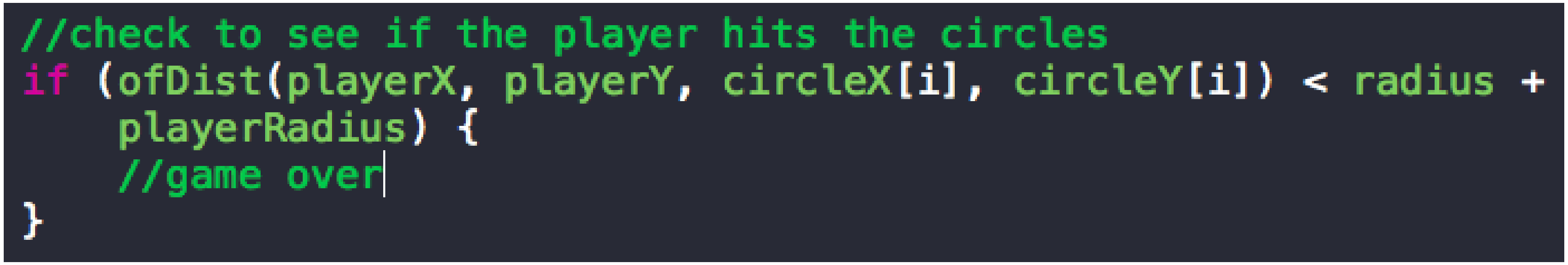
```
for(int i = 0; i < numCircle; i++){
    ofCircle(circleX[i], circleY[i], radius);
}
```

Make the first row of circles move in Update.



```
for(int i = 0; i < numCircle; i++){
    //circles start out moving right
    circleX[i] += speedOne[i];

    //if first row of circles hit the side of screen they bounce
    if (circleX[i]+radius > ofGetWidth()){
        speedOne[i] *= -1;
    }

    if (circleX[i]-radius < 0){
        speedOne[i] *= -1;
    }
}
```

Collision detection in Update.



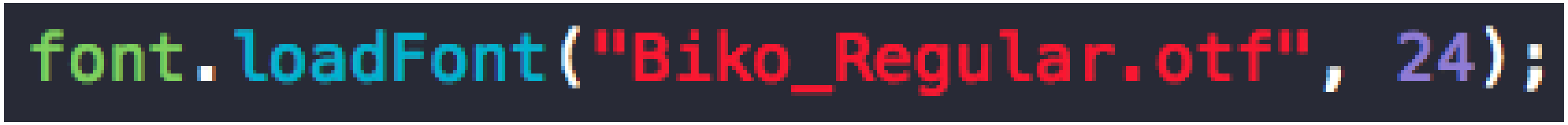
```
//check to see if the player hits the circles
if (ofDist(playerX, playerY, circleX[i], circleY[i]) < radius +
    playerRadius) {
    //game over
}
```

But wait... what's the win condition? What kind of code should we write to implement a win and lose state?

Game State time!

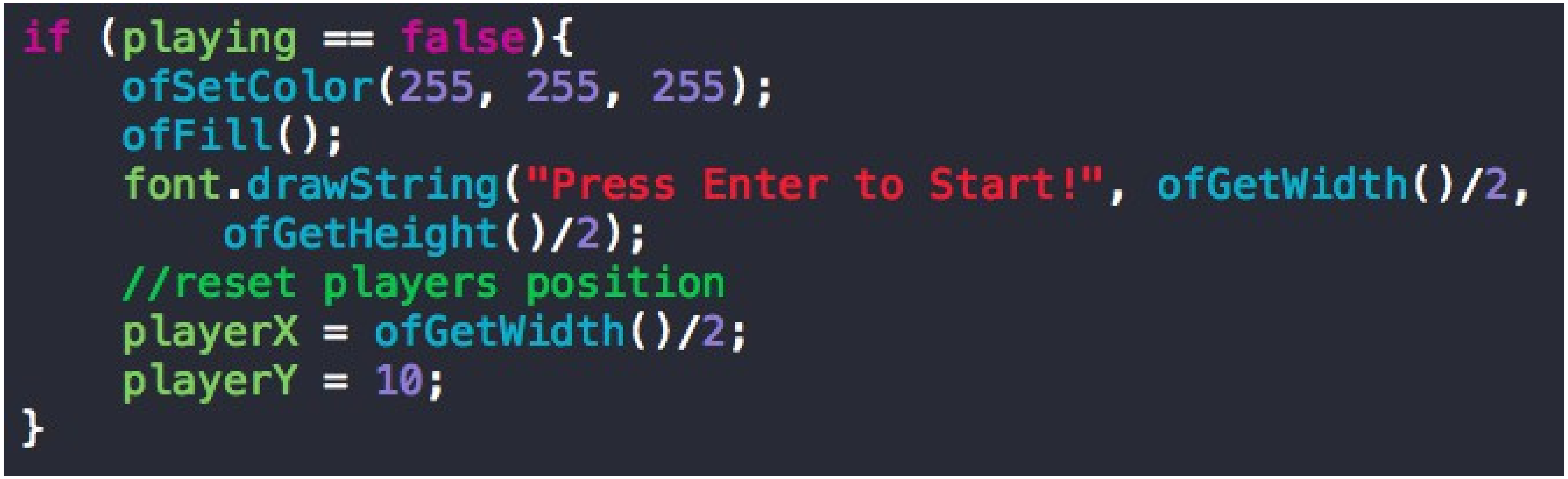
Declare two boolean variables called playing and winning. Initialize them in Setup to false.

Then, declare a variable called font with the data type ofTrueTypeFont. Initialize it like this in Setup (make sure there's a font in your data/lib folder!):



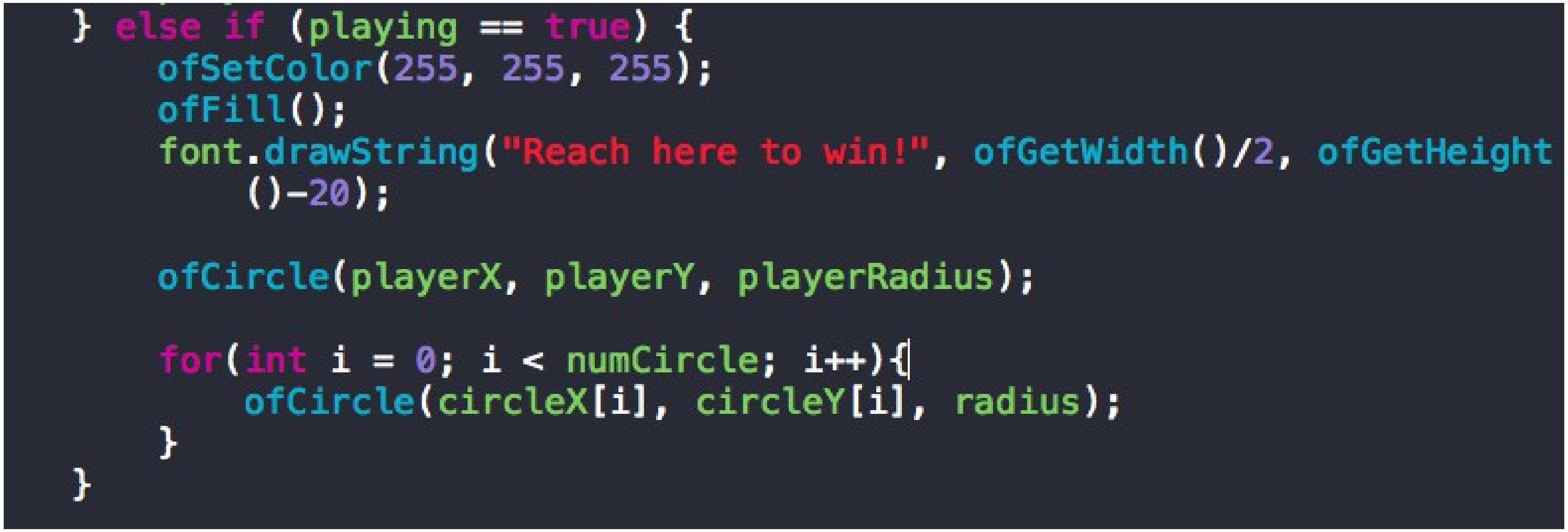
```
font.loadFont("Biko_Regular.otf", 24);
```

Next, we will tell our program what to do when the player is NOT playing in Draw.



```
if (playing == false){
    ofSetColor(255, 255, 255);
    ofFill();
    font.drawString("Press Enter to Start!", ofGetWidth()/2,
        ofGetHeight()/2);
    //reset players position
    playerX = ofGetWidth()/2;
    playerY = 10;
}
```

Next, we will tell our program what to do when the player IS playing by following up our last if statement with an else if in Draw.



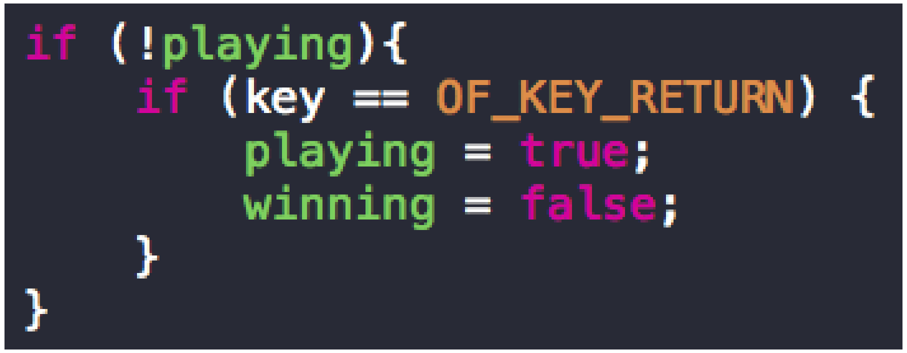
```
} else if (playing == true) {
    ofSetColor(255, 255, 255);
    ofFill();
    font.drawString("Reach here to win!", ofGetWidth()/2, ofGetHeight
        ()-20);

    ofCircle(playerX, playerY, playerRadius);

    for(int i = 0; i < numCircle; i++){
        ofCircle(circleX[i], circleY[i], radius);
    }
}
```

We need a way to switch between the two game states we just set up.

Any ideas where and how we might write that code?

We need a way to switch between the two game states we just set up.

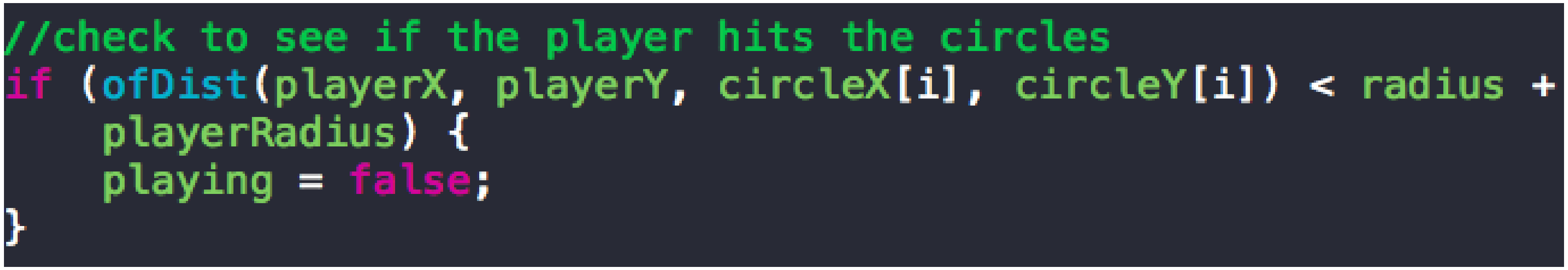
Switch from start screen to game play in keyPressed.



```
if (!playing){
    if (key == OF_KEY_RETURN) {
        playing = true;
        winning = false;
    }
}
```

We need a way to switch between the two game states we just set up.

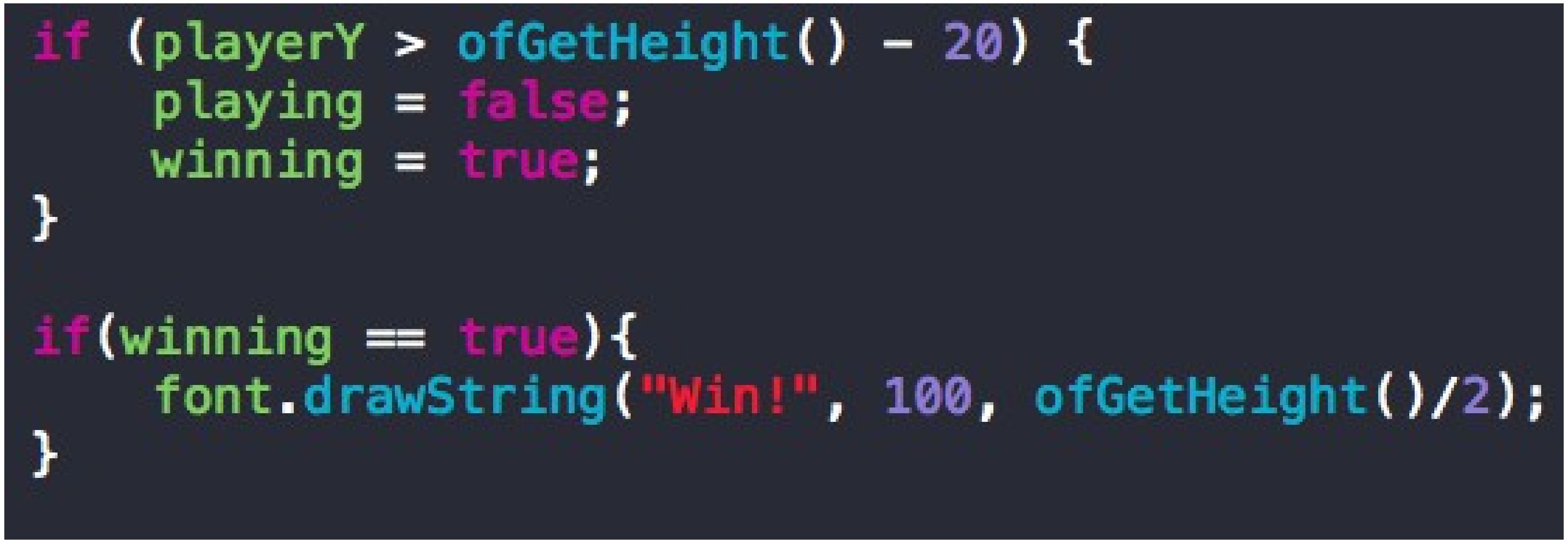
Lose condition in Update.



```
//check to see if the player hits the circles
if (ofDist(playerX, playerY, circleX[i], circleY[i]) < radius +
    playerRadius) {
    playing = false;
}
```

What about a win state?

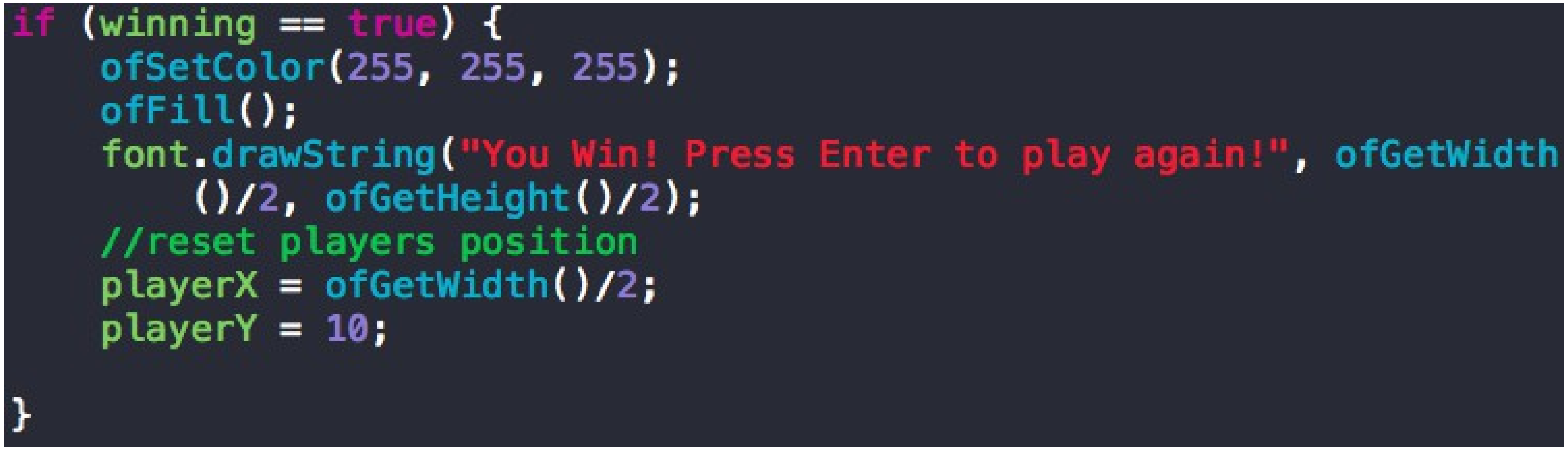
Try this in Update:



```
if (playerY > ofGetHeight() - 20) {
    playing = false;
    winning = true;
}

if(winning == true){
    font.drawString("Win!", 100, ofGetHeight()/2);
}
```

What about a win state?

Try this in Draw:

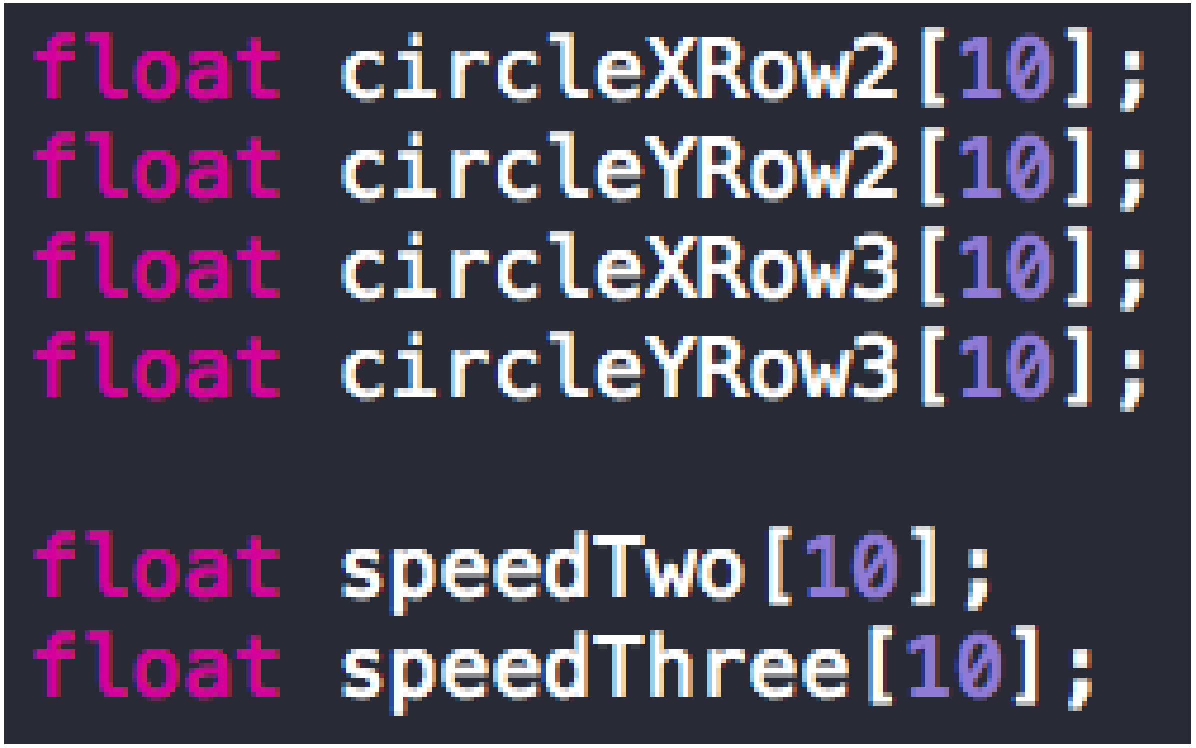


```
if (winning == true) {
    ofSetColor(255, 255, 255);
    ofFill();
    font.drawString("You Win! Press Enter to play again!", ofGetWidth
        ()/2, ofGetHeight()/2);
    //reset players position
    playerX = ofGetWidth()/2;
    playerY = 10;

}
```

Voila! A Frogger clone... but with only one row of cars. Let's test our game and then try adding 2 more rows of cars...

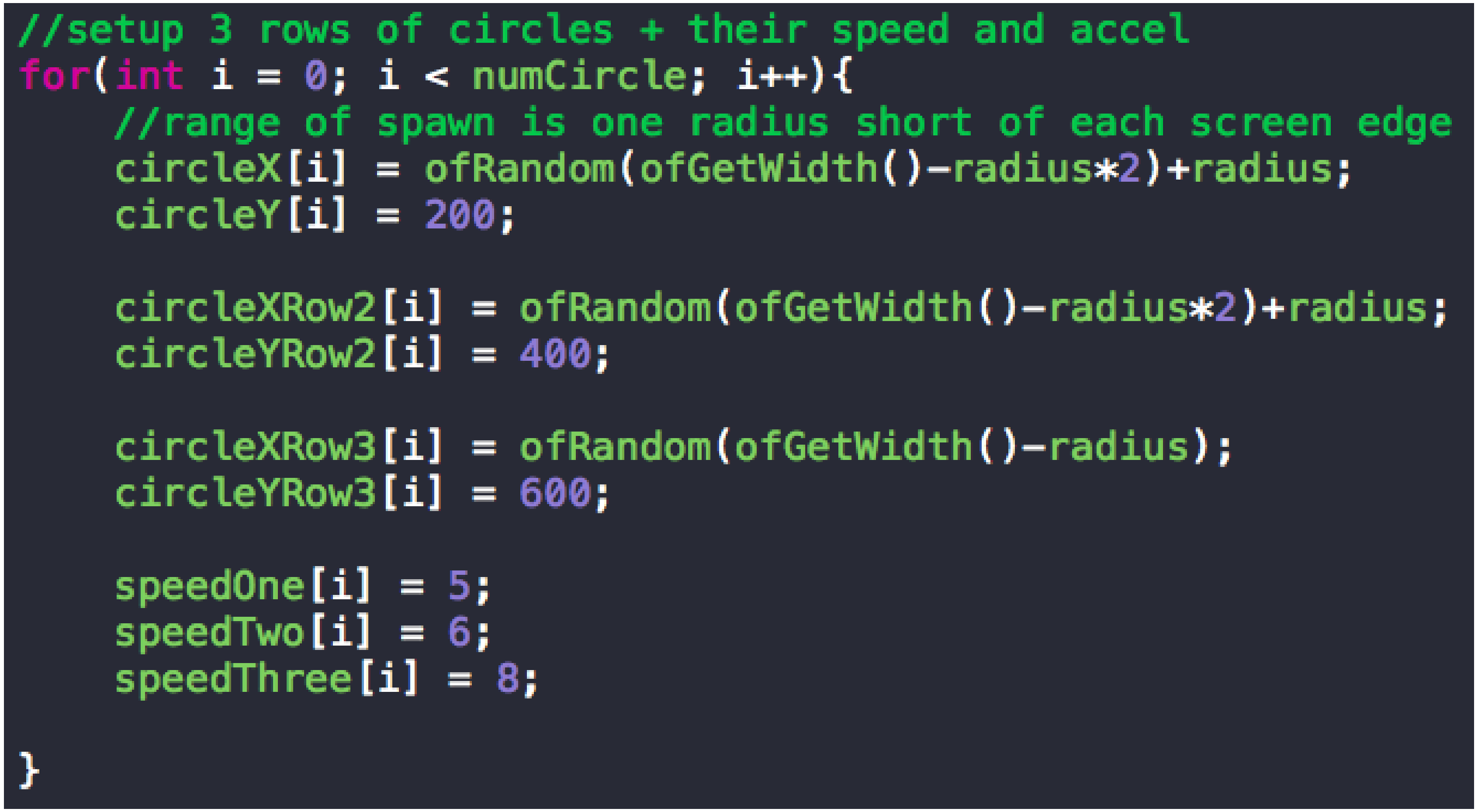
What kind of code do we need to write to add two additional rows of moving circles?

Declare your variables.



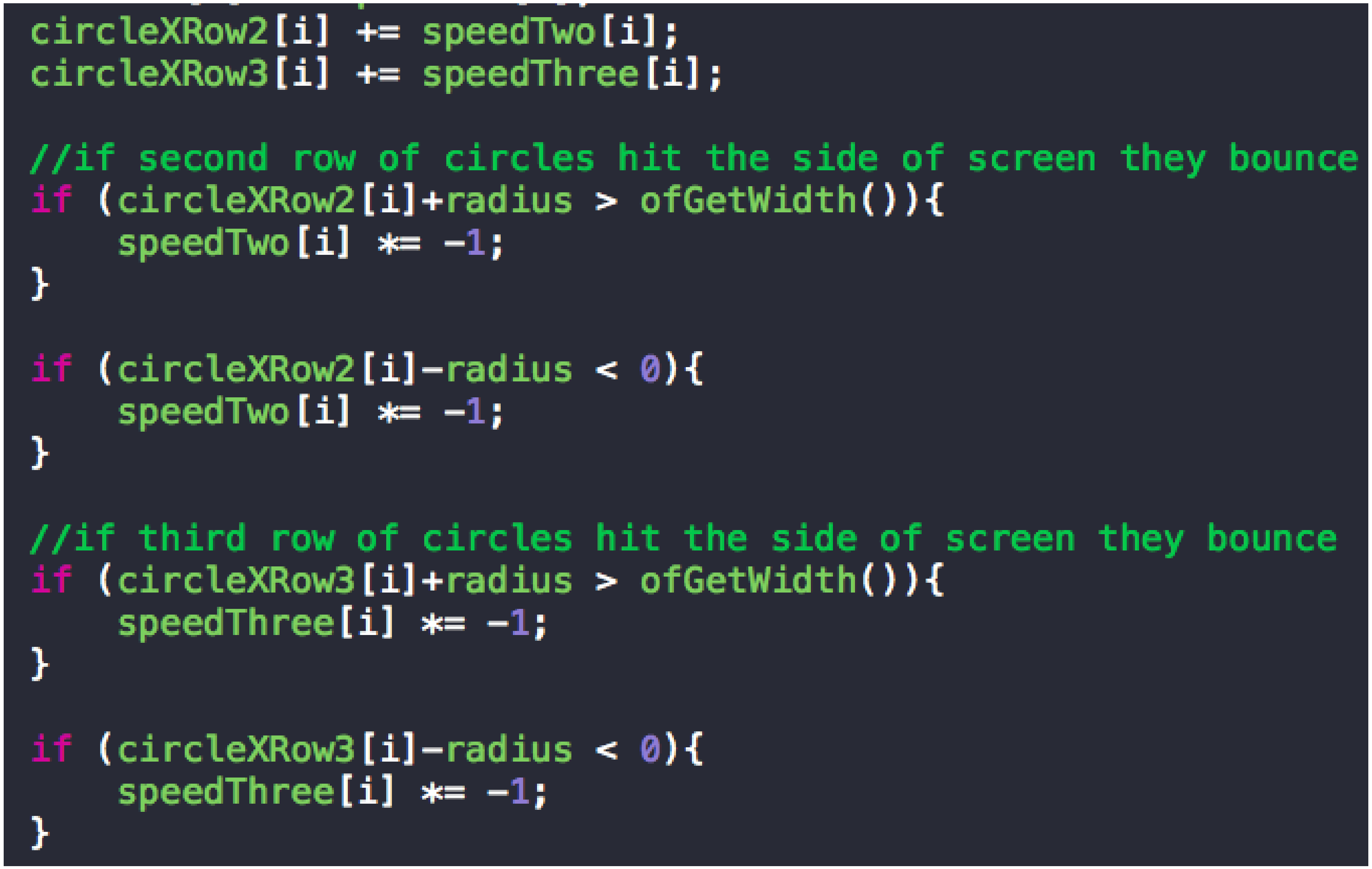
```
float circleXRow2[10];
float circleYRow2[10];
float circleXRow3[10];
float circleYRow3[10];

float speedTwo[10];
float speedThree[10];
```

Add this to your for loop in Setup.



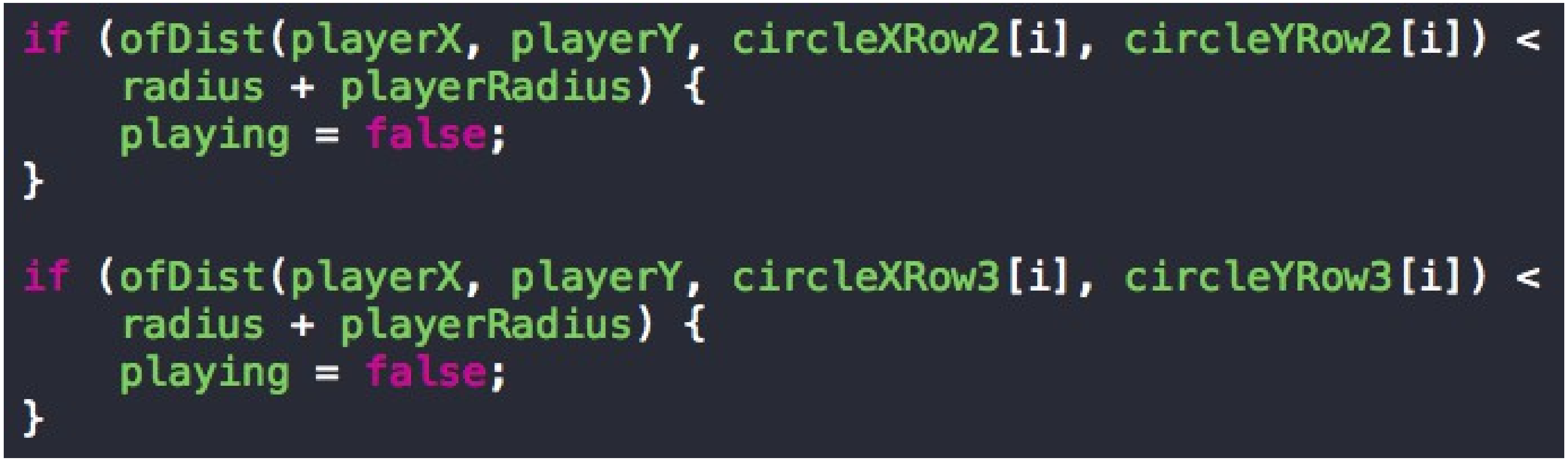
```
//setup 3 rows of circles + their speed and accel
for(int i = 0; i < numCircle; i++){
    //range of spawn is one radius short of each screen edge
    circleX[i] = ofRandom(ofGetWidth()-radius*2)+radius;
    circleY[i] = 200;

    circleXRow2[i] = ofRandom(ofGetWidth()-radius*2)+radius;
    circleYRow2[i] = 400;

    circleXRow3[i] = ofRandom(ofGetWidth()-radius);
    circleYRow3[i] = 600;

    speedOne[i] = 5;
    speedTwo[i] = 6;
    speedThree[i] = 8;

}
```

Add this to your for loop in Update.



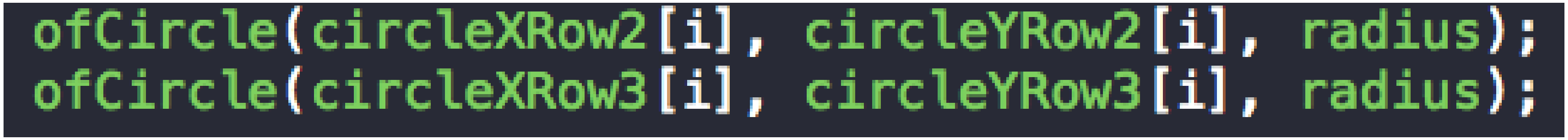
```
circleXRow2[i] += speedTwo[i];
circleXRow3[i] += speedThree[i];

//if second row of circles hit the side of screen they bounce
if (circleXRow2[i]+radius > ofGetWidth()){
    speedTwo[i] *= -1;
}

if (circleXRow2[i]-radius < 0){
    speedTwo[i] *= -1;
}

//if third row of circles hit the side of screen they bounce
if (circleXRow3[i]+radius > ofGetWidth()){
    speedThree[i] *= -1;
}

if (circleXRow3[i]-radius < 0){
    speedThree[i] *= -1;
}
```

Add this to your for loop in Update.



```
if (ofDist(playerX, playerY, circleXRow2[i], circleYRow2[i]) <
    radius + playerRadius) {
    playing = false;
}

if (ofDist(playerX, playerY, circleXRow3[i], circleYRow3[i]) <
    radius + playerRadius) {
    playing = false;
}
```

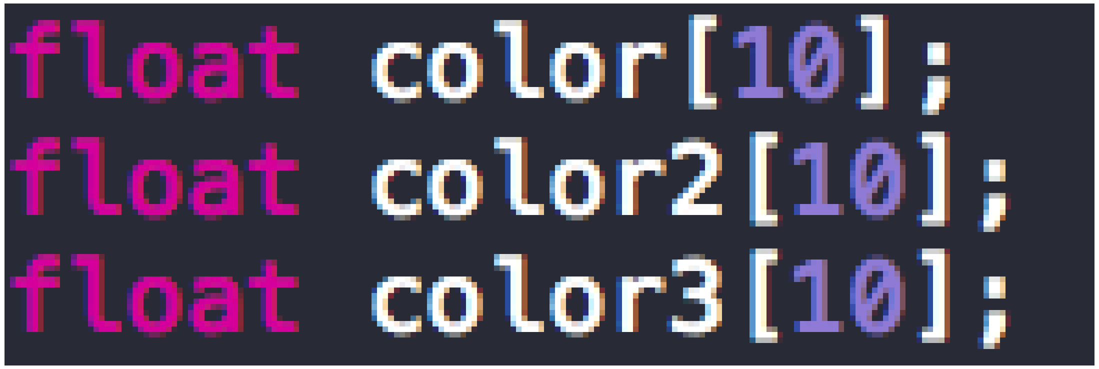
Finally, add this to your for loop in Draw.



```
ofCircle(circleXRow2[i], circleYRow2[i], radius);
ofCircle(circleXRow3[i], circleYRow3[i], radius);
```

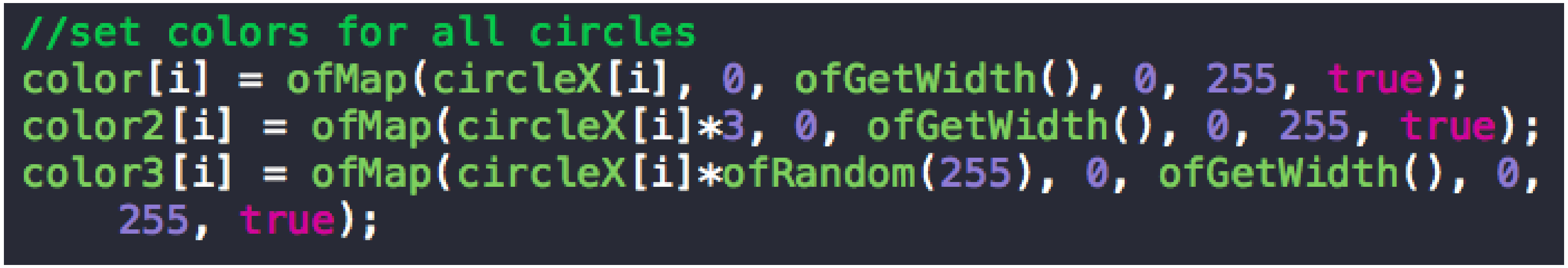
Bonus! If you run your game and all is well, let's add some crazy colors.

Declare these arrays.



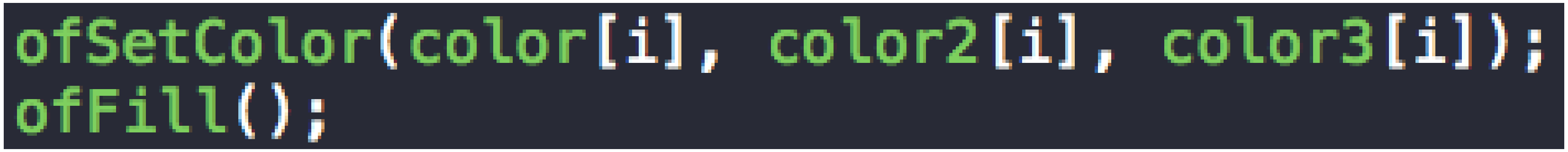
```
float color[10];
float color2[10];
float color3[10];
```

Put this in your for loop in Update.



```
//set colors for all circles
color[i] = ofMap(circleX[i], 0, ofGetWidth(), 0, 255, true);
color2[i] = ofMap(circleX[i]*3, 0, ofGetWidth(), 0, 255, true);
color3[i] = ofMap(circleX[i]*ofRandom(255), 0, ofGetWidth(), 0,
    255, true);
```

Put this in your for loop in Draw.



```
ofSetColor(color[i], color2[i], color3[i]);
ofFill();
```