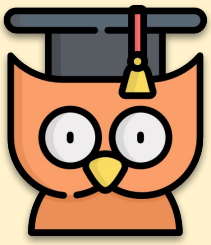
[the academy_of_code]

Grade 4
Unit 4

If Statements, Collision Detection for Making Games

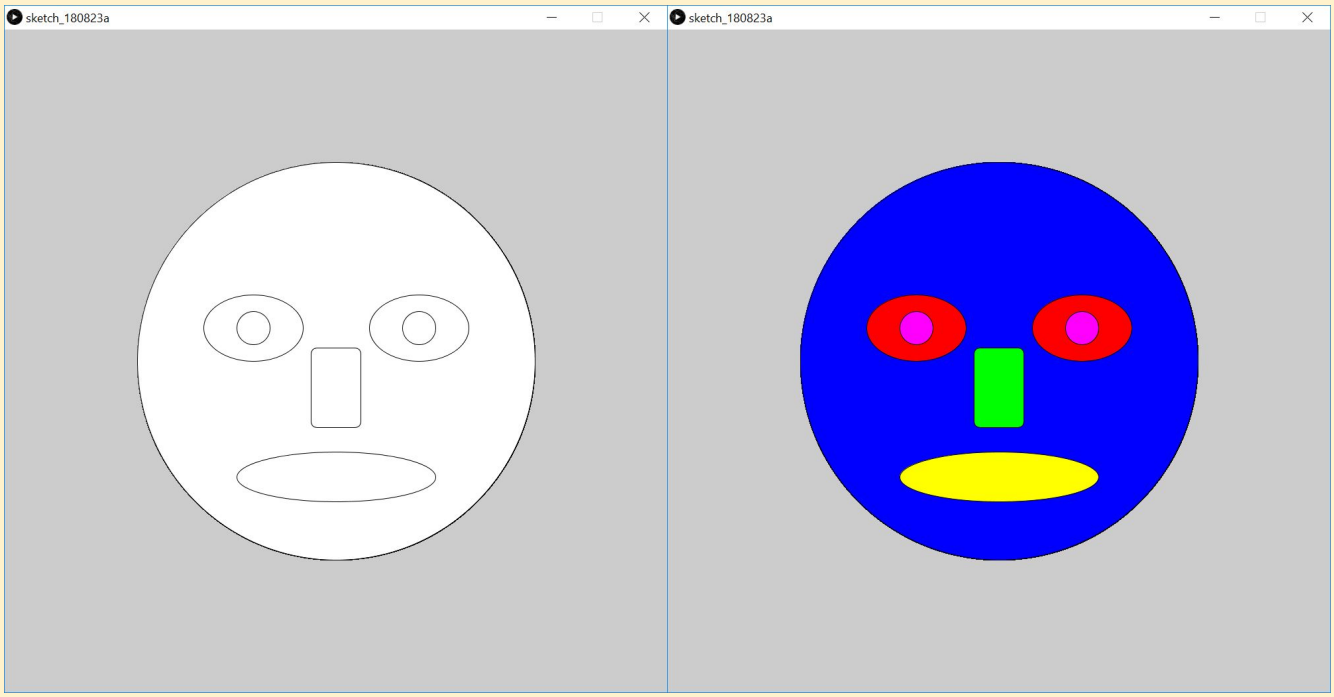
www.theacademyofcode.com/handouts

Lesson 1 - If Statements I



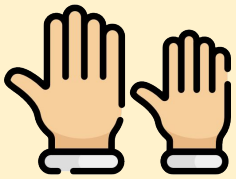
Learning Outcomes:

- We will learn about **if statements**; what they are and how to use them
- Use 2 new **variables**: **mousePressed** and **keyPressed**
- We will build our first **interactive** program, where things **change** depending on the **inputs** of the user

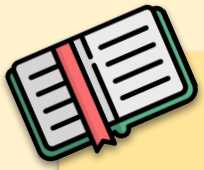


Make sure you know all these topics before you start:

- ✓ Using void **setup()** and void **draw()**
- ✓ **Variables**; what they are and what they do
- ✓ Drawing shapes and colouring them in with **fill()**

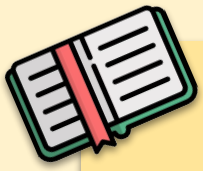


REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



What are if statements?

When we're coding, we often want to test to see **if** certain **conditions** are **true**. For example, in a computer game, the player's gun will only shoot **if** the player presses the button that is associated with "fire", **otherwise** nothing will happen. To do this, we need to use **if statements**.



Anatomy of an if statement

Let's take a look at this example:

```
if (mousePressed == true) { // this line is called the condition
  background(255, 0, 0); // this line is only run if the condition is true
}
```

What does this code do?

Write the above code into a code which includes a *void setup* and *void draw.*

Be Careful!

if statements **DO NOT** use semicolons **(;)**

Instead, after an **if** statement, you must create a new **{code block}** , just like **void setup()** and **void draw()**

Also, you use **2 equal signs (==)** inside an **if** statement, as this is the computers' equal sign, also known as **Logical Equal**.

Write the following code on a new Processing sketch and run it.

When you press the mouse, a circle should appear on the screen.

```
void setup() {
  size(600, 600);
}

void draw() {
  background(150);
  if (mousePressed == true) {
    ellipse(width/2, height/2, 200, 200);
  }
}
```

Task 1

Change your code so that instead of drawing **circle** when you press the mouse, it draws a **rectangle**.

Task 2

Add a **circle** that is **always** shown on the screen (does **not** depend on whether or not you're pressing the mouse).

Task 3

Use a **variable** so that when the mouse is clicked the size gets bigger/ the position of the ellipse changes.

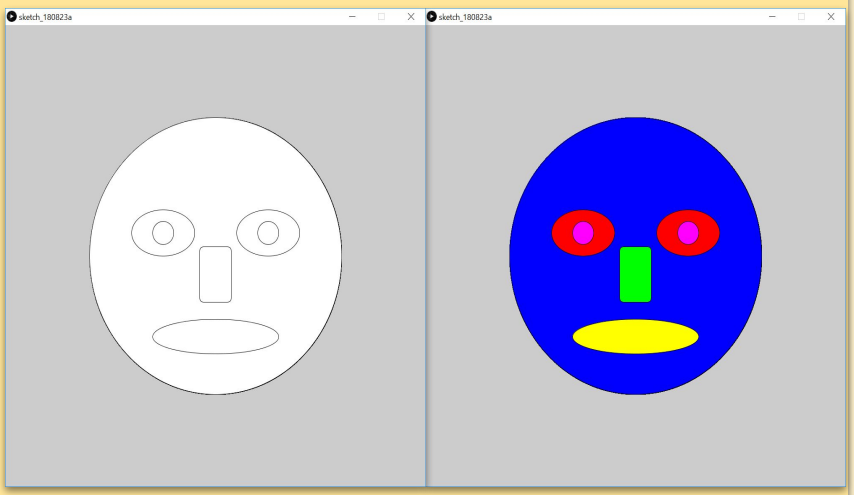
Extra Challenge

Extend your code so that **if** you press any **key**, you change the **colour** of the background.

Hint: Use **keyPressed** this time instead of **mousePressed**

Challenge

Make a program that draws a very basic face (use 3-6 shapes), and have different parts of the face change colour **if** the mouse is clicked.



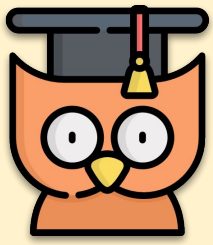
More Advanced Challenge

Make a program that draws a very basic house (use 3-6 shapes.. we went overboard in our example!), but make the **rooms** change colour if you press the **mouse** or **key**. We've given a starting code for a simple house below.

Hint: you'll have to group the windows that will change colour together and put the **fill()**/rect commands and if statements before them. **Also,** use 'tweak mode' if possible.

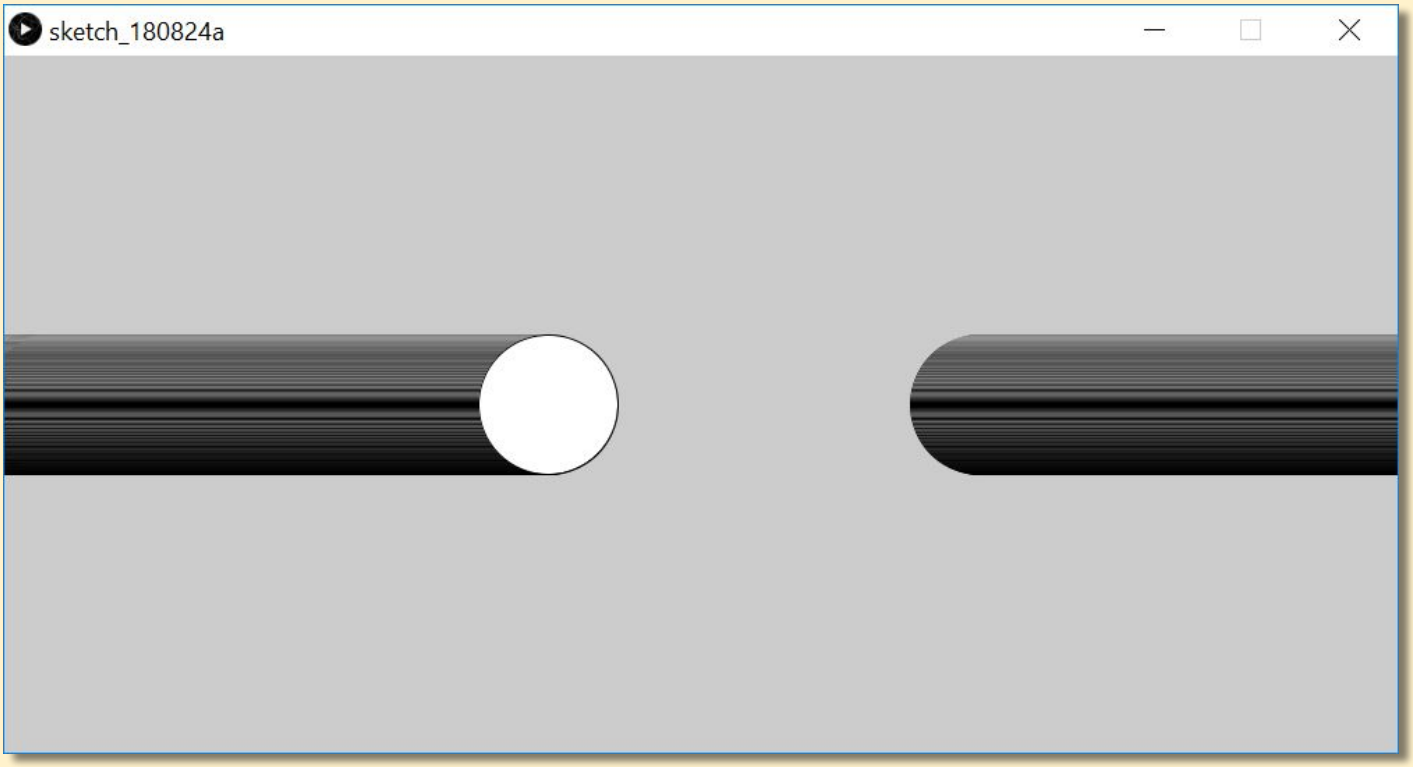
```
void setup() {
  size(700, 600);
}

void draw() {
  background(0);
  rectMode(CENTER);
  fill(9, 80, 6); //green
  rect(300, 570, 855, 130); //grass

  fill(118, 31, 31); //house colour
  rect(360, 400, 400, 300); //house

  fill(88);
  triangle(160, 250, 561, 250, 339, 100); //roof
  fill(118, 31, 31);
  rect(237, 172, 57, 112); //chimney

  fill(33);
  rect(357, 540, 62, 22); //step
  fill(183, 24, 129);
  rect(357, 474, 54, 120);//door
}
```

Lesson 2 - If Statements II



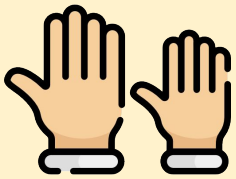
Learning Outcomes:

- We will learn more about **if statements**
- Use **else statements** for the first time
- Learn about **end conditions**
- Learn new **logic symbols**

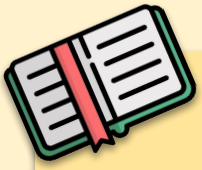


Make sure you know all these topics before you start:

- ✓ Using void **setup()** and void **draw()**
- ✓ Using variables
- ✓ Using **if** statements



REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



End Conditions

Last time we learnt about **if** statements and we talked about programs when they're used, like in animations and games. To make more advanced programs, we need to use **end conditions**. They let us **loop** shapes across the screen or **stop** a shape from moving if we need to. But before we get started on those, let's learn some **new logic symbols**!

Symbol	Meaning
==	Equal to (Logical Equal)
>	
<	
>=	Greater than or equal to
<=	

Discussion Time

It looks like someone forgot to write all the meanings in...

Can you think of what the blank symbols could mean?

Expert Tip

Try to get **very** familiar with these, you will be using them **all** the time!

Have a look at the following task and without writing about it, have a think about why everything is there and what you think will happen?

1. Write the following code on a new Processing sketch and run it.

```
float ballX;

void setup() {
  size(600, 600);
  ballX = 50;
}

void draw() {
  ellipse(ballX, height/2, 100, 100);
  if (ballX < 200) {
    ballX += 5;
  }
}
```

2. Now **fix** the code, so that the ball stops when it gets to the middle of the screen, not before. Save the file.

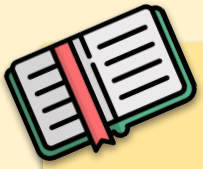
3. Now make it stop when it gets to the right edge of the screen. But **only** when the **right side** of the ball hits the edge, not the middle. Save the file.

4. Now, try and have the ball reappear on the left side of the screen after it leaves the right. Save the file as a new file.

Now would be a great time to save your sketch if you haven't already

Congratulations

You have written your first end condition!



else statements

else statements can also be used **along with if** statements. They are **always** used **after** an **if** statement and will **only** run the code in their code block **if** the condition in the **previous if** statement was **false**.

Anatomy of an else statement

```
if (mousePressed == true) {
    background(0);
  }
else { // otherwise
    background(255);
  }
```

If the **condition** inside the **if** statement is **true**, run the code inside the **if {code block}**

Otherwise, run the code inside the **else {code block}**

Let's Get Coding

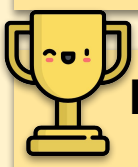
Can you get your ball from the previous task to **loop**? When the ball goes off the **right** hand side of the screen, it should **loop** back to the **left** side of the screen.

Hint: You'll need to put in an **else** statement **after** your existing **if** statement, like so:

```
if (ballX < 200) {
    ballX++;
  }
else {
    // move your ball back to the
    // left side of the screen
  }
```

Expert Tip

'**ballX++**' is the same as saying '**ballX = ballX +1**' Less typing is always better!



Extra Challenge

Make the animation **smooth**! Only change the position of your ball when the **left side** of the ball **leaves** the **right edge** of the screen, and then **only** the **right side** of the ball should appear on the left.

Let’s Get Coding

Using **else if** statements create a rollover programme that functions like the example on the right.

We’ve given you the starting code below. Your task is to finish it so that all areas light up.

```
void setup() {
  size(480, 270);
}

void draw() {
  background(255);
  stroke(0);
  line(240, 0, 240, 360); //Cuts the window Vertically
  line(0, 135, 640, 135); //Cuts the window horizontally

  noStroke();
  fill(random(255), random(100, 200), random(100, 200));

  if (mouseX < 240 && mouseY < 135) {
    rect(0, 0, 240, 135);
  } else if (mouseX > 240.......
```

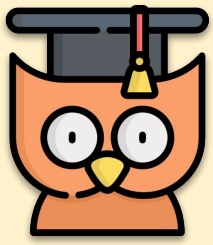
Extra Challenge

Change your code so it **only** uses **if** statements, no **else** statements allowed!

Extra Challenge

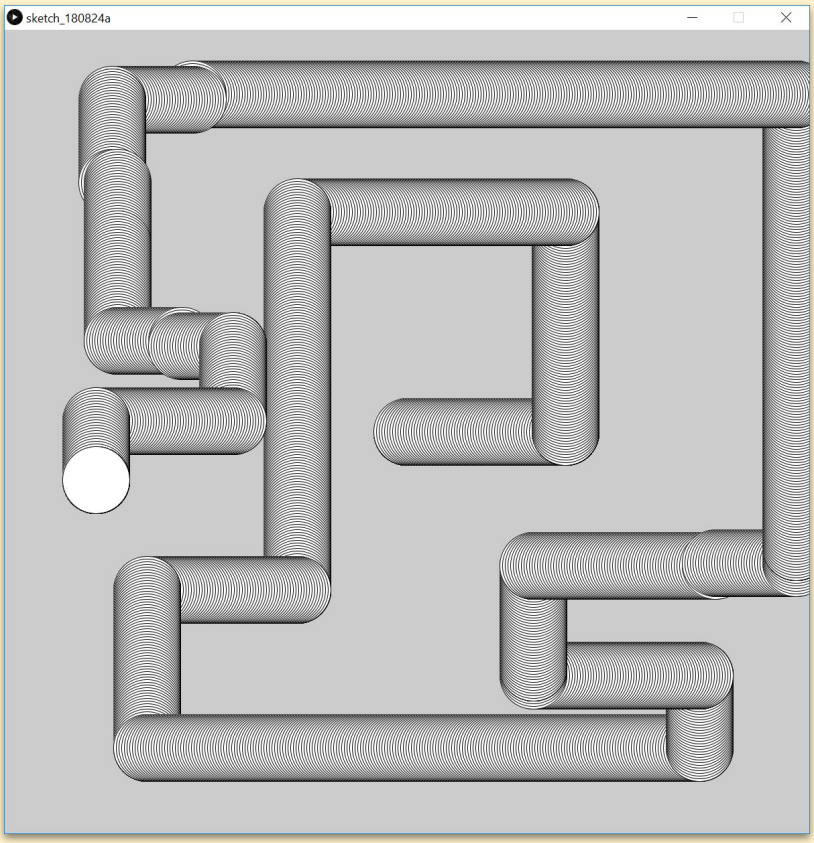
Customise your programme so that it does something different. Perhaps made a phone keypad (with text) that changes a little when numbers are hovered over.

Lesson 3 - IF Statements III



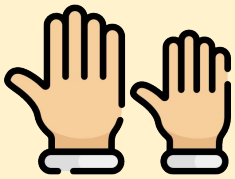
Learning Outcomes:

- Have an understanding of advanced **if** statement concepts, e.g. nested **if** statements.
- Change the location and colour of shapes using the keyboard.
- Use different mouse buttons to run seperate pieces of code.

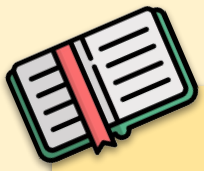


Revision topics before starting make sure you know:

- ✓ Declaring and initializing variables.
- ✓ Using **void setup()** & **void draw()**.
- ✓ Using **if** statements.
- ✓ Using **keyPressed**.



REMEMBER: Put up your hand. We love to help!

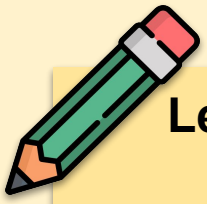


What are nested if statements?

Nested if statements are basically if statements **within** if statements. Let's take a simple case:

```
if (keyPressed) { // test all keys on keyboard
  if (key == 'd') { // test the 'd' key
    ballXPos += 1; // increase variable by 1
  }
}
```

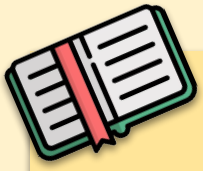
The code inside the if statements "ballXPos += 1;" will only execute if the conditions in **both** if statements are true. In this example, if **any** key on the keyboard has been pressed, the second if statement will then be tested. If the 'd' key has been pressed, then the code will execute.



Let's get coding!

Create a new Processing sketch and write the following code:

```
float ballXPos = 600;

void setup() {
  size(800, 800);
}

void draw() {
  background(255);

  ellipse(ballXPos, height/2, 100, 100);

  if (keyPressed) {
    if (key == 'd') {
      ballXPos += 4;
    }
  }
}
```



else if Statements

In the previous unit we were introduced to **else** statements that can be used **along with if** statements. We are now going to go one step further and introduce **else if** statements that can be used along with **if** & **else** statements.

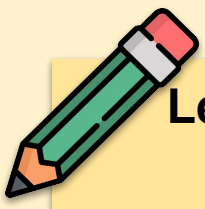
else if statements are used when more conditions are needed. They are always used **after** an if statement and their condition will only be tested if the condition in the previous **if**/**else if** statement was **not true**.

Anatomy of an else if statement

If the condition inside the **if** statement is **true**, run the code inside the **if {code block}**
otherwise, move onto the next **else if** statement and see if the condition is **true**.

Or **else**, if none of the conditions are **true**, run the code inside the **else {code block}**

```
if (keyPressed) {
  if (key == 'd') {
    fill(255, 255, 0); // yellow
  }
  else if (key == 'a') {
    fill(0, 255, 0); // green
  }
  else if (key == 'w') {
    fill(255, 0, 0); // red
  }
  else {
    fill(255); // white
  }
}
```



Let's get coding!

1. **Add** code to make the circle move to the **left** when the '**a**' key is pressed.

2. **Add** code to make the circle move to **up** when the '**w**' key is pressed and **down** when the '**s**' key is pressed.

Challenge Task

Add a few more keys to change the colour of the circle and the colour of the background.

Challenge Task

Make it so you see only one circle at a time (hide the trail!).

Let's get coding!

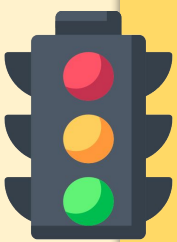
1. **Add** the following code to the **start** of your void **draw**():

```
if (mousePressed) {
  if (mouseButton == LEFT) {
    background(0, 0, 255);
  }
}
```

2. **Add** code to make the background change to a different colour when the **right** mouse button is pressed.

Challenge Task

Change your code so that the size of the circle will increase when the right mouse button is pressed and decrease when the left mouse button is pressed.



At this point, you should know how to:

- Nest **if** statements inside one another.
- Use curly brackets **{}** to separate code blocks.
- Use different keys on the keyboard and different buttons on the mouse to execute code blocks.
- Use **else if** statements where appropriate.

Let's get coding!

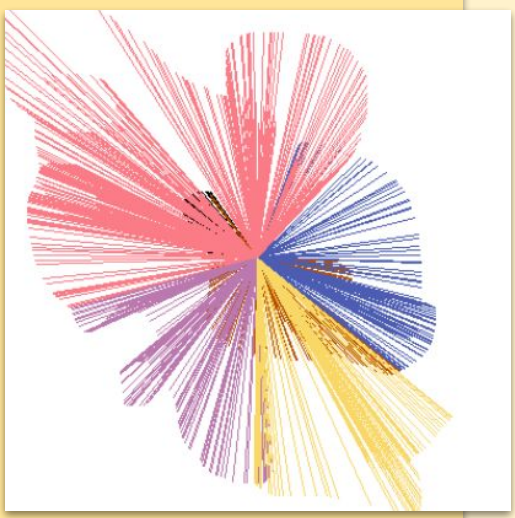
Using the code below and **else if** statements, write a code that uses mouse movement and keyboard input to create something pretty. Perhaps when you press the 'r' key, the colour will change to red etc. Make it so each time you click the mouse the color changes maybe. Experiment!

NOTE: to colour a line you need to use **stroke** instead of **fill**.

Starting code:

```
void setup(){
 size(400,400) ;
 frameRate(24) ;
}

void draw(){
 line(100,200, mouseX, mouseY) ;
}
```

Finished example:



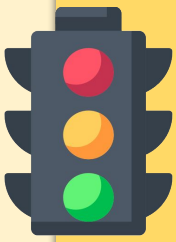
Challenge Task

```
void mousePressed(){
 ellipse(mouseX,mouseY,30,30) ;
}
```

Add the code on the right and customise it so that your drawing prints circles when the mouse is clicked.

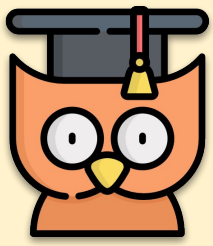
void mousePressed() is a function and should be used outside of **void draw/setup.**

At this point, you should know how to:



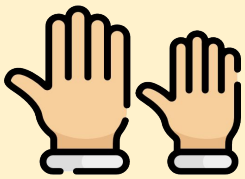
- Nest **if** statements inside one another.
- Use curly brackets **{}** to separate code blocks.
- Use different keys on the keyboard and different buttons on the mouse to execute code blocks.
- Use **else if** statements where appropriate.

Lesson 4 & 5 - Collision Detection

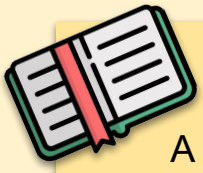


Learning Outcomes:

- Writing a code with edge detection
- Making a DVD bounce screen
- Writing code with circle/point collision
- Writing code with circle/circle collision



REMEMBER: Put up your hand. We love to help!



Collision Detection

A common geometric task, especially in games, is determining whether two object **intersect**; a bullet and a player, a player and an enemy, a wall and a car etc. Often in games a bounding box is fitted around a sprite and two sprites are said to **collide** if their bounding boxes intersect. This problem is also referred to as **collision detection**.

If you can determine that two shapes touch, you can trigger some action- think of detecting when the user has clicked a button, or when a game character touches the floor or an enemy. **Collision detection** can be accomplished using code that ranges from simple **if statements** to complicated algorithms handling thousands of objects at once. We'll start with something simple and work our way up from there.

Let's get coding - Edge Detection

We're going to write a code that creates variables that store the **x and y positions** and **x and y speeds** of the ball. It will uses those variables to draw the ball, and then it adjusts the position of the ball by its speed.

```
float xPos;
float yPos;
float xSpeed;
float ySpeed;
```

In **void setup** give the canvas a **size** of **(800,600)** and initialise the variables; *xPos* at 150, *yPos* at 100, *xSpeed* at 5 and *ySpeed* at 4.

In **void draw**, give it a **background** and create an ellipse with a position of **xPos** and **yPos** with a height and width of **50.** We will also need to check if the ball has gone off the left or right sides of the window, and if so it reverses the **xSpeed** of the ball. The code on the right will do this.

```
xPos = xPos + xSpeed;

if (xPos < 25 || xPos > width-25) {
  xSpeed *= -1;
}
```

(Remember: Multiplying something by -1 changes its sign +/-).

Add the '*yPos*' version of the above code.

DVD Challenge

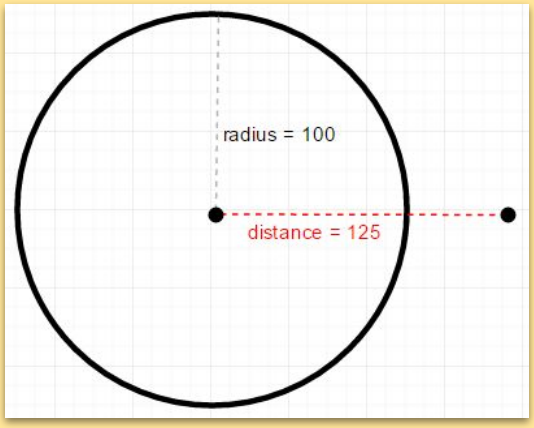
Do a google image search for the DVD logo. Using the snipping tool, capture a copy of this and save it to your project folder. Import the image and edit your code so that the DVD logo bounces instead of an ellipse.

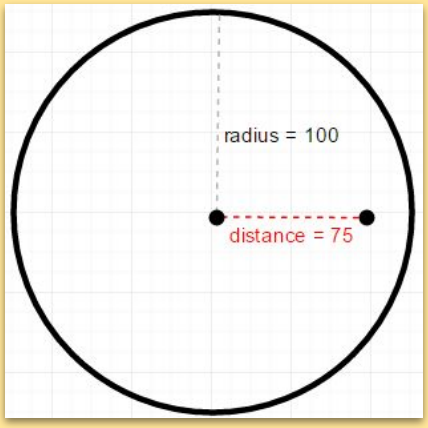
ULTIMATE CHALLENGE - Edit the code to get the code to hit the corner exactly. This is what everybody wants in life - https://youtu.be/m8NAlDOCG6g

Let's get coding - Circle Point Collision

Bouncing the ball is a basic version of edge detection, but every other form of collision detection will follow those basic steps. The only thing that changes is the particulars of how those steps are implemented.

We're going to make a code where the colour of a circle changes **if** the mouse (a point) goes inside the circle. For this, think about the relationship between the **radius** of a circle and the distance between the center of the circle and a point. If the distance between the centre of the circle and the point is less than the radius of the circle (diagram on the right), then that point is inside the circle!





Variables needed (initial value)

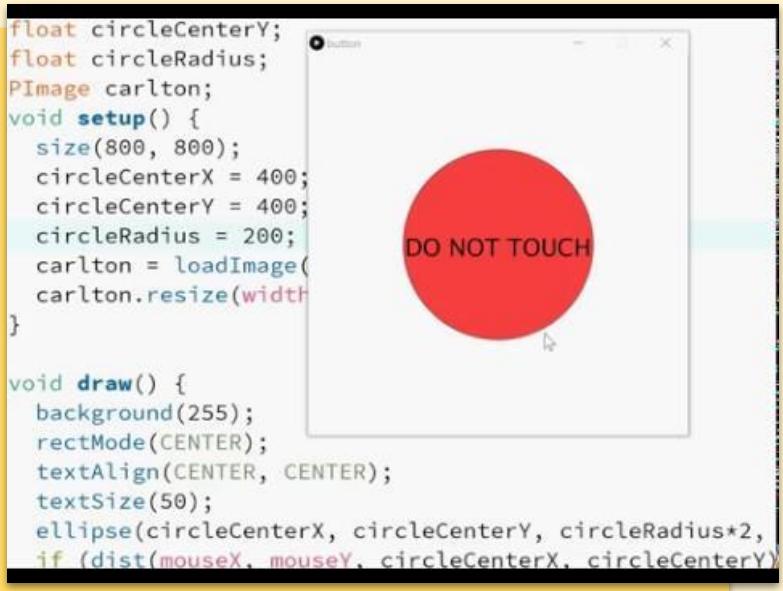
- circleCenterX; (400)
- float circleCenterY; (400)
- float circleRadius; (200)

Make a size of (800,800) and in **void draw** insert the code below. It states that if the distance between the point (the mouse) and the centre of the circle is less than the circle radius, a circle will be **red** else it will be green.

```
if (dist(mouseX, mouseY, circleCenterX, circleCenterY) < circleRadius) {
  fill(255, 0, 0);
} else {
  fill(0, 255, 0);
}

ellipse(circleCenterX, circleCenterY, circleRadius*2, circleRadius*2);
```

Random Challenge

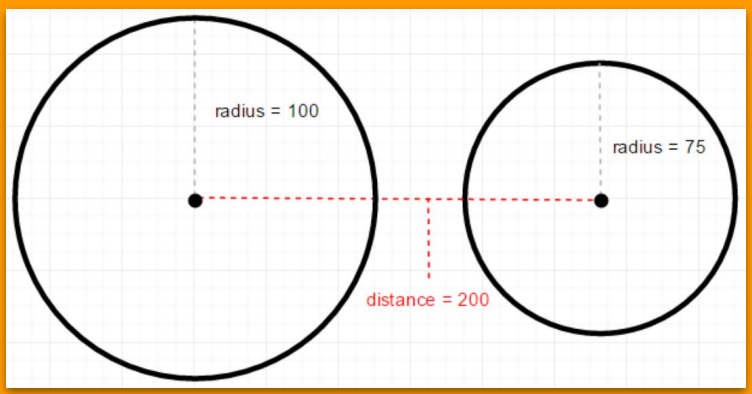
Edit the previous code so that the circle is a ‘button’ which makes something random appear like in the example to the right.

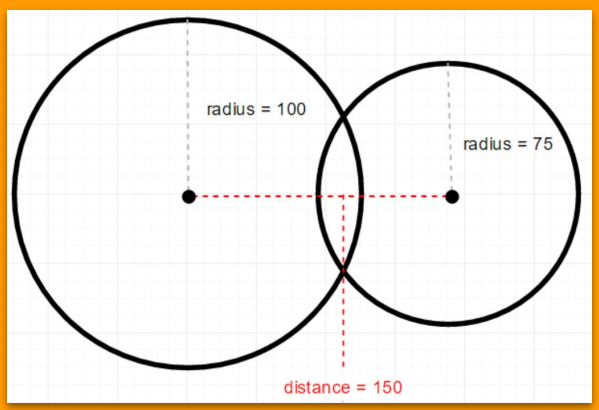


Circle/Circle Collision

Now that we know how to detect whether a point is colliding with a circle, we can expand that to detect when a circle collides with another circle. The idea is the same: we still check the distance between the points (the centres of the circles). But instead of checking against the radius of only one circle, **we check against the sum of the radiuses of both circles**. If the distance between the centers of the circle is less than the sum of the two radiuses, then the circles are colliding!

Let’s Have a Look

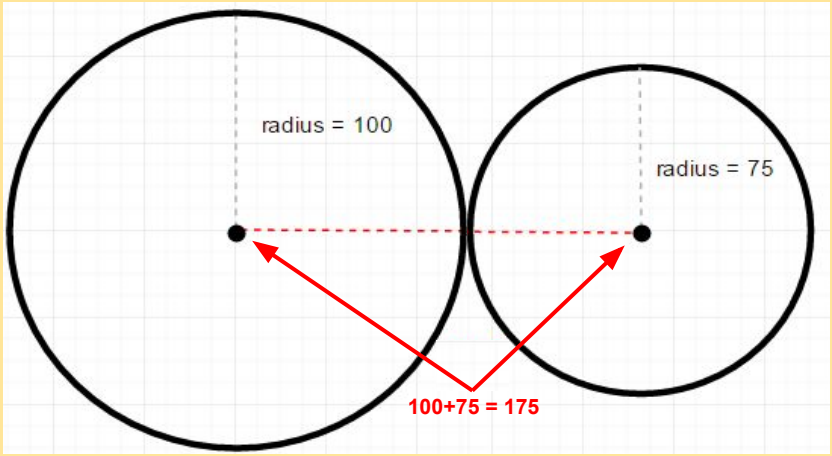




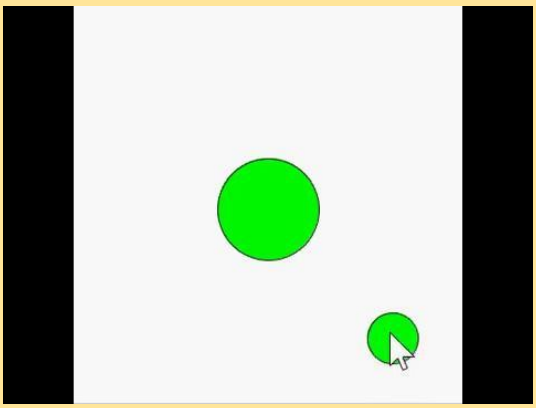
In the example on the above, notice that the distance between their centres (200) is greater than the their two radiuses added together, so we know these circles **are not** colliding.

On the other hand in this case, the distance between their centers (150) is less than the sum of the two radiuses (175), so we know these circles **are** colliding.

Let's get coding - Circle Circle Collision



Putting this into code, we still use the `dist()` function to get the distance between the points, but now we compare it to the **sum** of the radiuses.



Variables needed (initial value)

- bigCircleRadius (25)
- smallCircleRadius (25)

Create a canvas with a size of (300,300).

In void draw, give a background colour and write the following code for the big ellipse. `ellipse(width/2, height/2, bigCircleRadius*2, bigCircleRadius*2);`

Why are we dividing the width by two but multiplying the radius by two?

In the code below checks whether the distance between centres of the circles (which are located at the **cursor position** (mouseX, mouseY) and the centre of the window (width/2, height/2) is **less than the sum** of the radiuses of the circle. If this is true, then we've detected a collision between the two circles.

```
if (dist(mouseX, mouseY, width/2, height/2) < bigCircleRadius + smallCircleRadius){
  fill(255, 0, 0);
} else {
  fill(0, 255, 0);
}
```

Random Challenge

Similar to the challenge on page 18, edit the previous code so that the circles do something random. Consider:

- Adding text
- Adding an image
- Adding randomness

Over the next few lessons, we're going to be making a ball dodging game. Have your collision detection include another **if statement** or a different kind of collision detection. This will all be good practice as the ball dodging game will involve using everything we've studied so far.