Speed and Direction

14 min

In order to create smooth animations, we need to create a variable for the speed and then incrementally change that value with each frame.

In the example below, we are animating an ellipse that is falling at a set speed. To achieve this, we first define a variable named `speed`. We then set a variable named `yPos` for the ellipse's y position. Afterward, we increment `yPos` with the `speed` variable.

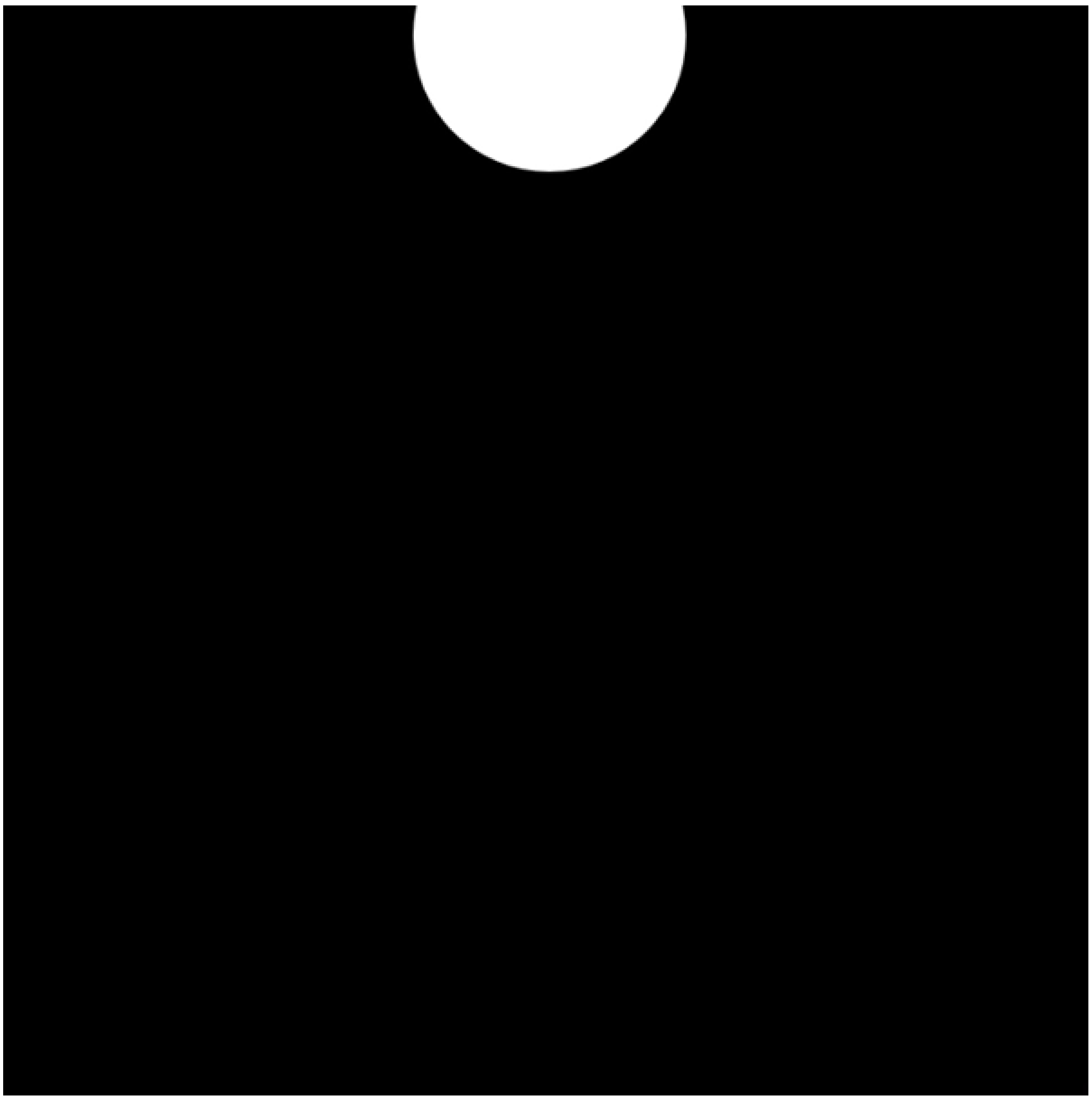
```
let speed = 1; // Determines the speed
let yPos = 0; // Starting position on y-axis

function draw(){
   ellipse(width / 2, yPos, 20, 20);
   yPos += speed; // Increments the value of the y position
}
```

To reset the ellipse to its initial location, we first write an `if` statement to check whether the ball has reached the bottom of the canvas. You can tell if it has reached the bottom when the y position of the ball is greater than the height (`yPos > height`). At this point, reassign the y position of the ball back to 0, which is the top of the canvas.

```
if(yPos > height) {
  yPos = 0; // When yPos becomes greater than height, reset to 0
}
```

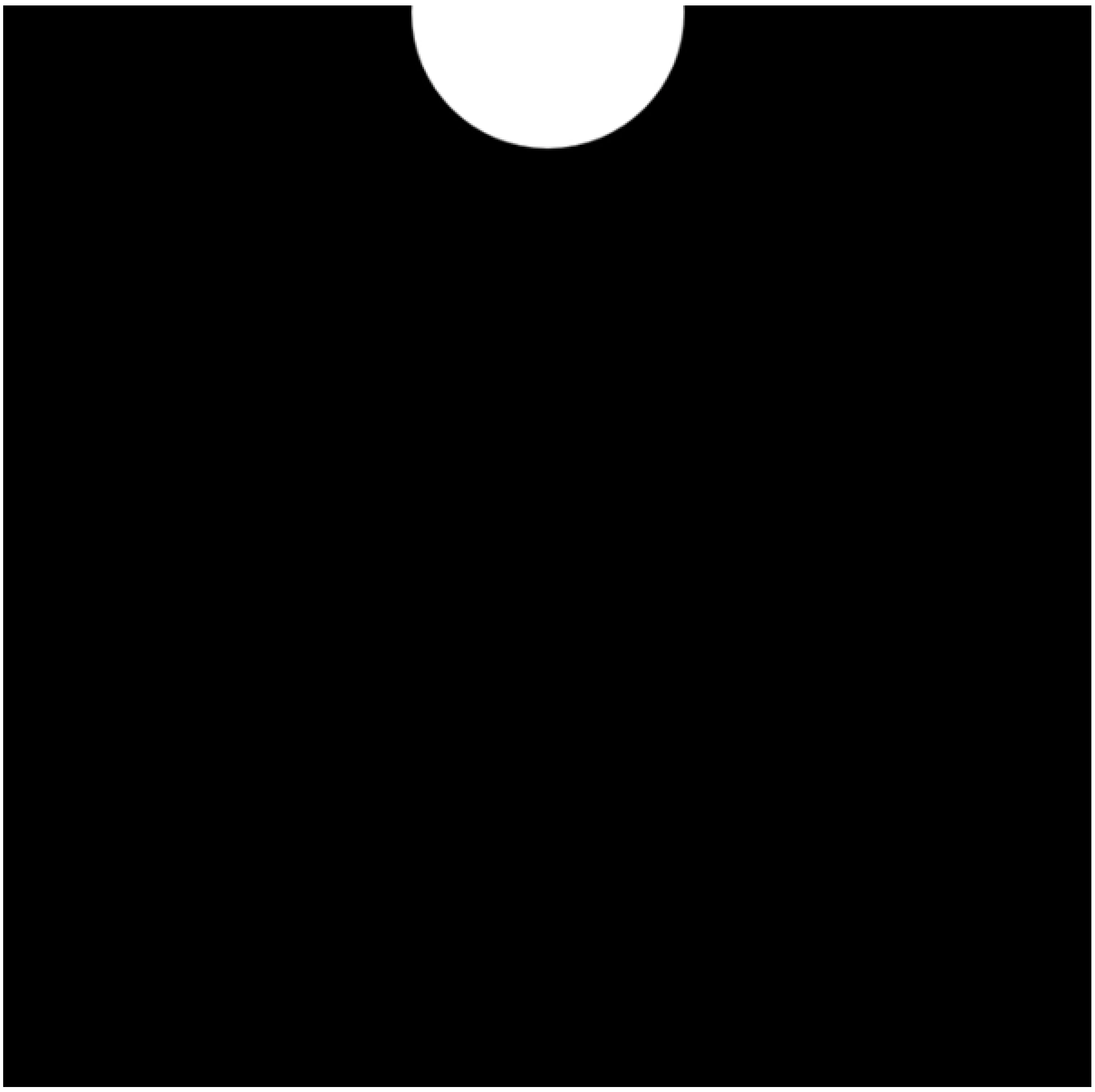
With the code above you can create this infinitely looping animation of a dropping ball.



Instead of resetting the ball's position to the top, you can have it bounce up when it reaches the bottom of the sketch. To move the ball in the opposite direction, inverse the `speed` variable by multiplying the value by -1.

```
if(yPos > height){
  speed = speed * -1; // Negative value changes speed to opposite
direction
}
```

By inverting the direction of the speed, we have created an animation of a ball that bounces when it reaches the bottom.



Let’s now try to have the ball bounce off all four sides of the canvas!

To achieve this, we first need to differentiate the speed for the x-axis and the y-axis movement. Let’s do this by creating variables called `xSpeed` and `ySpeed` to represent the movements along both axes.

```
// Initialize speed along x and y-axis to 1
let xSpeed = 1;
let ySpeed = 1;
```

Next, let’s write an `if` statement to check when the ellipse’s x position is greater than the canvas `width` or less than 0. We use an OR operator (`||`) to check if either condition is true.

```
if(xPos < 0 || xPos > width) {
  // If xPos is less than 0 or greater than width, run this code
}
```

We need to write another if statement to check when the ellipse’s y position is greater than the canvas `height` or less than 0.

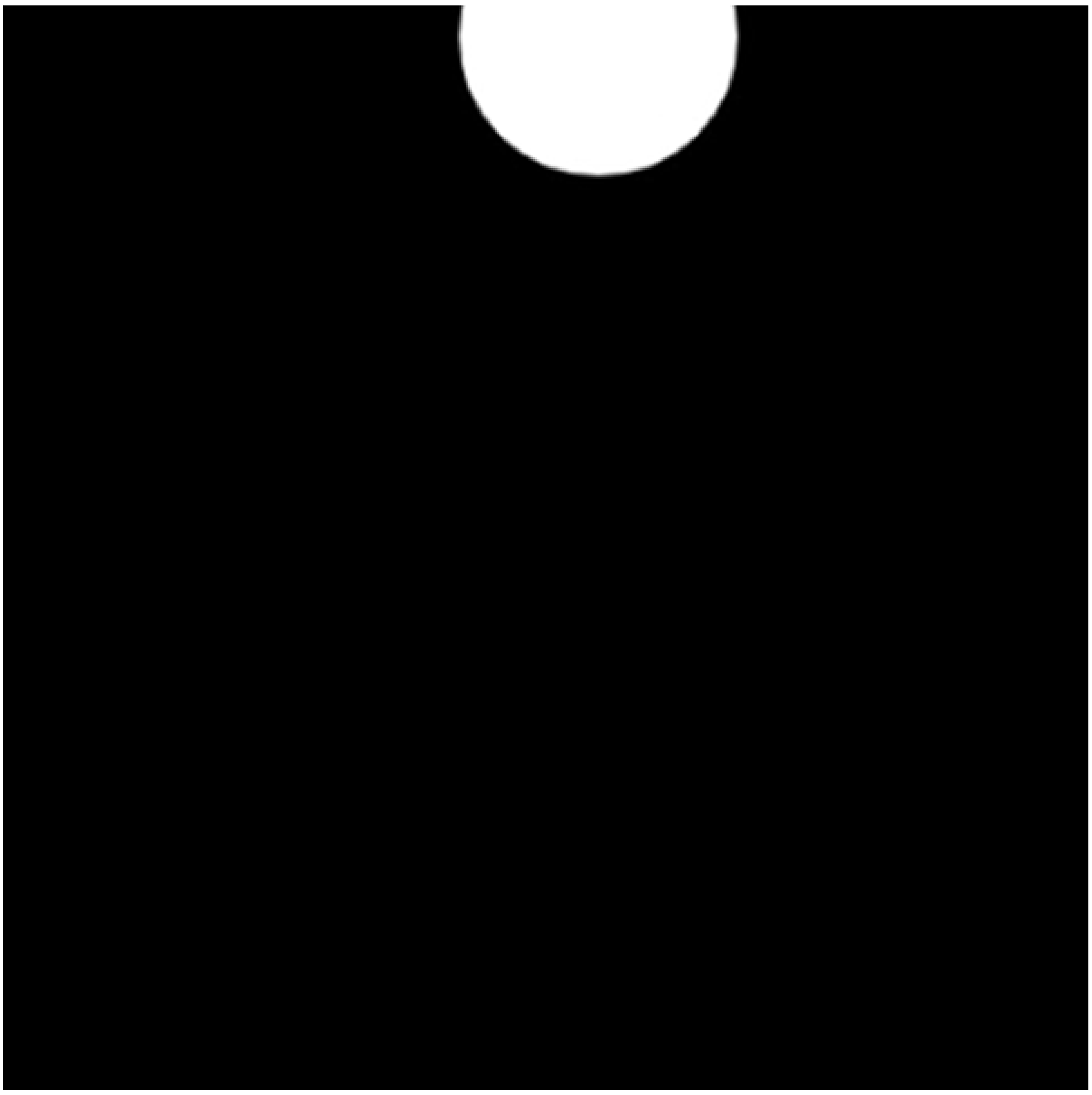
```
if(yPos < 0 || yPos > height) {
  // If yPos is less than 0 or greater than height, run this code
}
```

When the ellipse touches any side of the canvas, we move it in the opposite direction by multiplying -1 to the variable representing speed. We write the code to change the direction within the `if` statements.

```
if(xPos < 0 || xPos > width) {
  xSpeed = xSpeed * -1; // Invert xSpeed when xPos is less than 0 or
greater than width
}

if(yPos < 0 || yPos > height) {
  ySpeed = ySpeed * -1; // Invert ySpeed when yPos is less than 0 or
greater than height
}
```

When we piece all the steps together it will create an animation of a ball bouncing around the canvas.



Instructions

Checkpoint 1 Passed

1.

Let’s practice controlling the speed and direction by creating a bouncing ball around the canvas. Note that the global variables are already defined for you at the top of the sketch.

Underneath the `ellipse()` function, increment the `circleX` variable, the x position of the ball, by a value of `xSpeed`. This will move your ball along the x-axis.

Hint

Remember that you can increment a value by writing:

```
x += amount;
```

This takes the current value of the variable and then adds `amount` to it.

2.

Below the line you just added, increment the `circleY` variable, the y position of the ball, by a value of `ySpeed`. Your ball should now be gradually dropping diagonally.
Hint
Remember that you can increment a value by writing:

```
x += amount;
```

This takes the current value of the variable and then adds `amount` to it.

3.

Now, let's make the ball move in the opposite direction anytime it reaches the left and right sides of the canvas.

Below the increment statements you added, write an `if` statement that evaluates whether the `circleX` variable has become less than 0 or greater than the canvas `width`. If the ball has reached either end, invert the direction of the `xSpeed`.
Hint
Remember that you can use the OR operator (`||`) inside the `if` statement to check if one of the multiple statements is true.

You can invert the direction by multiplying the speed value by -1.

4.

Lastly, write another `if` statement that evaluates whether the `circleY` variable has has become less than 0 or greater than the canvas `height`. If the ball has reached either side, invert the direction of the `ySpeed`.

Your ball should now bounce anytime it has reached any sides of the canvas.
Hint
Remember that you can use the OR operator (`||`) inside the `if` statement to check if one of the multiple statements is true.

You can invert the direction by multiplying the speed value by -1.

sketch.js

```
// Global variables
let circleX = 300;
let circleY = 0;
let xSpeed = 1;
let ySpeed = 1;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(0);
  // Bouncing ball
  ellipse(circleX, circleY, 120);

  // TODO: Increment the x position of the ball
  circleX += xSpeed;

  // TODO: Increment the y position of the ball
  circleY += ySpeed;

  // TODO: If statment to inverse direction when ball hits left or right edge
  if (circleX < 0 || circleX > width) {
    xSpeed *= -1;
  }

  // TODO: If statment to inverse direction when ball hits top or bottom edge
  if (circleY < 0 || circleY > height) {
    ySpeed *= -1;
  }
}
```