Code your First Game

Timothy Clark

Adapted from code-your-first-game.com

–

tdhc.uk/pong

Chapter 1: The Basics

Step 1: First Code File

Extract from “Creating a Website”

The Internet, The Web and HTML

- The **Internet** is the global “network of networks” that use Internet protocol (IP) to link billions of devices worldwide.
- The **W**orld **W**ide **W**eb is an information space where web resources are identified by URLs, interlinked by hypertext links, and are accessed via the Internet
- A **web site** is an online location that maintains one or more web pages.
- A **web page** is an individual page, either online or offline.
- **H**yper **T**ext **M**arkup **L**anguage is a translatative language used to create content in a web page.
- The **W**orld **W**ide **W**eb **C**onsortium (W3C) is an organisation founded by Tim Berners-Lee, as an international standardisation organisation for the Web.

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8" />
  <title>My Game!</title>
</head>
```

First Code File

1. Create the file
2. Declare the DOCTYPE and Charset
3. Write some HTML (5!)
4. Add the <script> tags
5. Write some JavaScript!
6. Now create the canvas, 800 pixels × 600 pixels

```
</head>
<body>
  <strong>Hello World!</strong>
  <p>This is a paragraph</p>
</body>
```

First Code File

1. Create the file
2. Declare the DOCTYPE and Charset
3. Write some HTML (5!)
4. Add the <script> tags
5. Write some JavaScript!
6. Now create the canvas, 800 pixels × 600 pixels

```
<body>
    <script>
        console.log(“Hello World!”);
        window.alert ("Hello World!");
    </script>
</body>
```

First Code File

1. Create the file
2. Declare the DOCTYPE and Charset
3. Write some HTML (5!)
4. Add the <script> tags
5. Write some JavaScript!
6. Now create the canvas, 800 pixels × 600 pixels

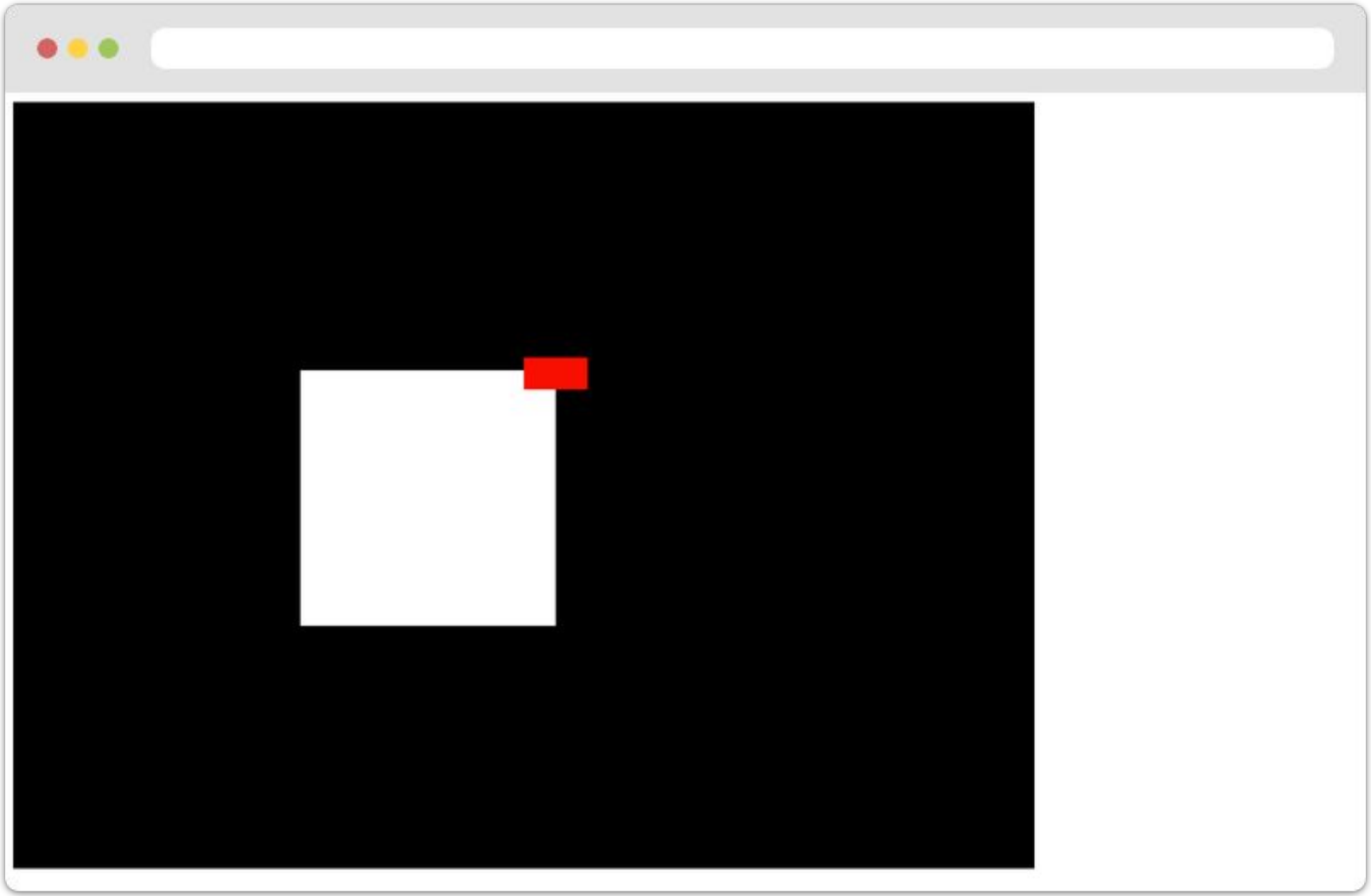
```
<canvas
  id="gameCanvas"
  width="800"
  height="600">
</canvas>
<script>
```

First Code File

1. Create the file
2. Declare the DOCTYPE and Charset
3. Write some HTML (5!)
4. Add the <script> tags
5. Write some JavaScript!
6. Now create the canvas, 800 pixels × 600 pixels

Chapter 1: The Basics

Step 2: Drawing and Position



```
<script>
  var canvas;
  var canvasContext;
</script>
```

Drawing and Position

1. Declare "canvas" and "canvasContext" Using "canvas",
2. document.getElementById("canvasID")
3. Get the canvasContext with "canvas.getContext('2d')"
4. Create the window.onload
5. Try out canvasContext.fillStyle then .fillRect
6. fillStyle looks for colour values
7. fillRect needs more values (X Coordinate,Y Coordinate, Width,Height)
8. Draw 3 rectangles

Drawing and Position

1. Declare "canvas" and "canvasContext" Using "canvas",
2. document.getElementById("canvasID")
3. Get the canvasContext with "canvas.getContext('2d')"
4. Create the window.onload
5. Try out canvasContext.fillStyle then .fillRect
6. fillStyle looks for colour values
7. fillRect needs more values (X Coordinate,Y Coordinate, Width,Height)
8. Draw 3 rectangles

```
canvas = document.getElementById('gameCanvas');
```

Drawing and Position

1. Declare "canvas" and "canvasContext" Using "canvas",
2. document.getElementById("canvasID")
3. Get the canvasContext with "canvas.getContext('2d')"
4. Create the window.onload
5. Try out canvasContext.fillStyle then .fillRect
6. fillStyle looks for colour values
7. fillRect needs more values (X Coordinate,Y Coordinate, Width,Height)
8. Draw 3 rectangles

```
canvas = document.getElementById('gameCanvas');
canvasContext = canvas.getContext('2d');
```

```
canvas = document.getElementBy
canvasContext = canvas.getConte

window.onload = function ( ) {

}
```

Drawing and Position

1. Declare "canvas" and "canvasContext" Using "canvas",
2. document.getElementById("canvasID")
3. Get the canvasContext with "canvas.getContext('2d')"
4. Create the window.onload
5. Try out canvasContext.fillStyle then .fillRect
6. fillStyle looks for colour values
7. fillRect needs more values (X Coordinate,Y Coordinate, Width,Height)
8. Draw 3 rectangles

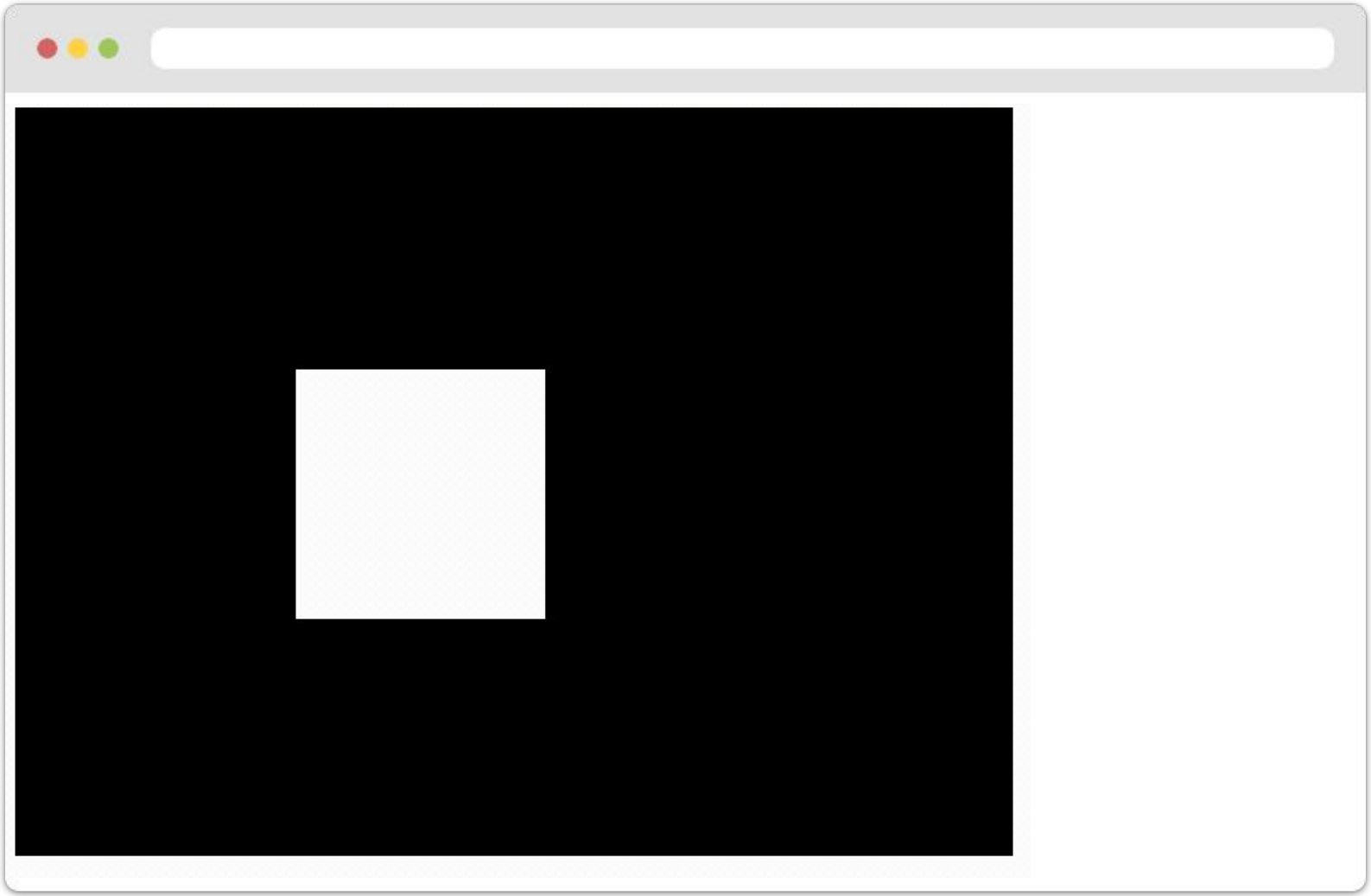
```
window.onload = function ( ) {
    canvasContext.fillStyle = ‘white’;
    canvasContext.fillRect (100,50, 250,300);
}
```

Drawing and Position

1. Declare "canvas" and "canvasContext" Using "canvas",
2. document.getElementById("canvasID")
3. Get the canvasContext with "canvas.getContext('2d')"
4. Create the window.onload
5. Try out canvasContext.fillStyle then .fillRect
6. fillStyle looks for colour values
7. fillRect needs more values (X Coordinate,Y Coordinate, Width,Height)
8. Draw 3 rectangles

Chapter 1: The Basics

Step 3: Movement and Time



Movement and Time

1. Draw the Ball
2. Package all the draw code into a function, drawAll
3. In window.onload, use setInterval to call drawAll at millisecond intervals
4. Declare a variable, ballX
5. Log its value each time drawAll is called
6. Increase its value each time drawAll is called - it now moves!

```
window.onload = function ( ) {
    canvasContext.fillStyle = 'white';
    canvasContext.fillRect (100,100, 50,50);
}
```

```
window.onload = function ( ) {

}

function drawAll ( ) {
    canvasContext.fillStyle = 'white';
    canvasContext.fillRect (100,100, 50,50);
}
```

Movement and Time

1. Draw the Ball
2. Package all the draw code into a function, drawAll
3. In window.onload, use setInterval to call drawAll at millisecond intervals
4. Declare a variable, ballX
5. Log its value each time drawAll is called
6. Increase its value each time drawAll is called - it now moves!

```
window.onload = function ( ) {
    setInterval(drawAll, 100);
}
```

Movement and Time

1. Draw the Ball
2. Package all the draw code into a function, drawAll
3. In window.onload, use setInterval to call drawAll at millisecond intervals
4. Declare a variable, ballX
5. Log its value each time drawAll is called
6. Increase its value each time drawAll is called - it now moves!

```
var ballX = 100;
```

Movement and Time

1. Draw the Ball
2. Package all the draw code into a function, drawAll
3. In window.onload, use setInterval to call drawAll at millisecond intervals
4. Declare a variable, ballX
5. Log its value each time drawAll is called
6. Increase its value each time drawAll is called - it now moves!

Movement and Time

1. Draw the Ball
2. Package all the draw code into a function, drawAll
3. In window.onload, use setInterval to call drawAll at millisecond intervals
4. Declare a variable, ballX
5. Log its value each time drawAll is called
6. Increase its value each time drawAll is called - it now moves!

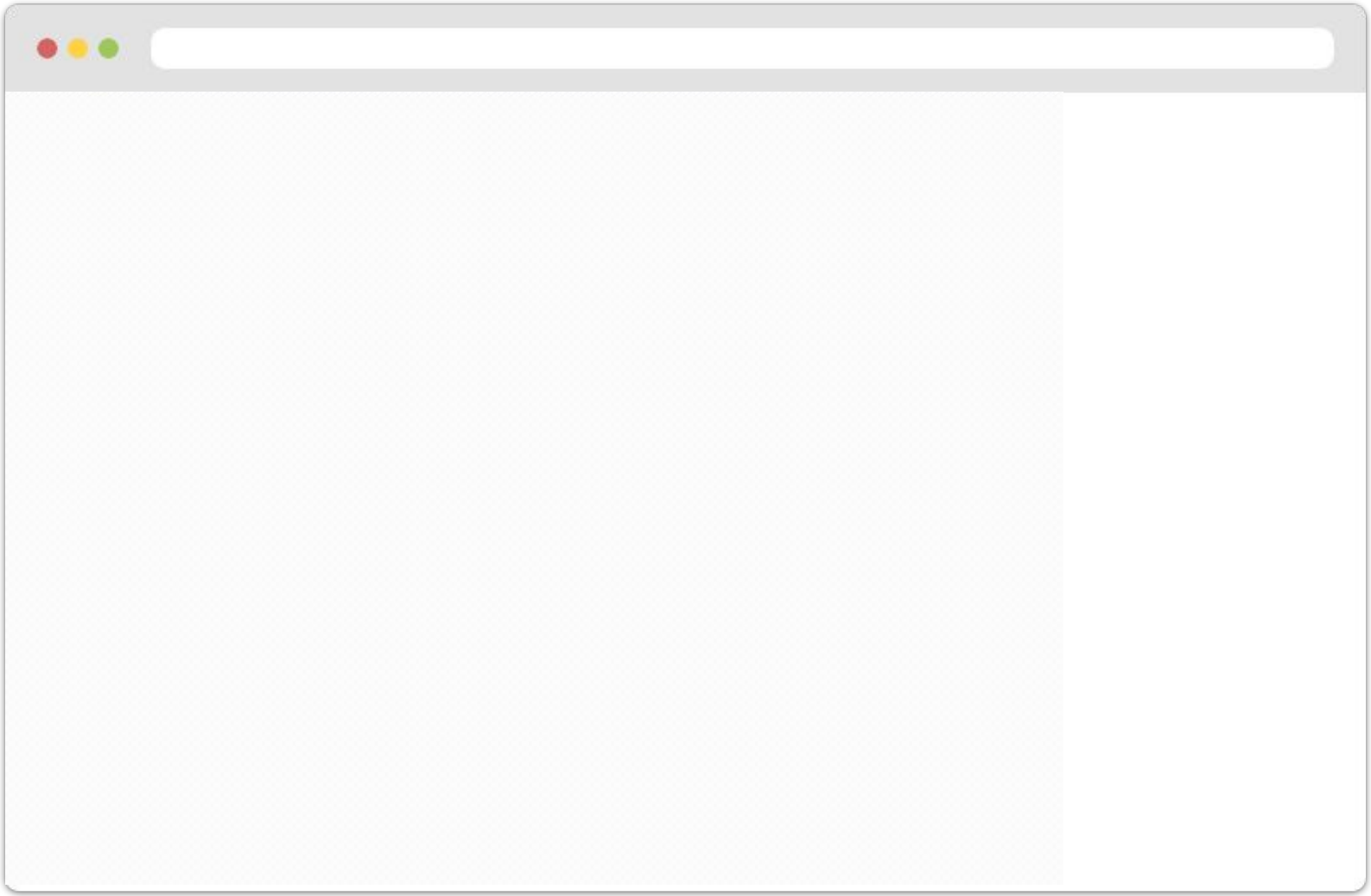
```
function drawAll ( ) {
    canvasContext.fillStyle = '#000';
    canvasContext.fillRect (0,0, canvas.width,canvas.height);
    canvasContext.fillStyle = 'white';
    canvasContext.fillRect (100,100, 50,50);

    console.log (ballX);
    ballX = ballX + 10;
}
```

Two ways to increment variables

```
ballX = ballX + 10;
ballX += 10;
```

Chapter 1: The Basics

Step 4: Cleaning Up



```
function drawAll ( ) {
    canvasContext.fillStyle = ‘black’;
    canvasContext.fillRect (0,0, canvas.width,canvas
    canvasContext.fillStyle = ‘white’;
    canvasContext.fillRect (0,210, 10,10);
    canvasContext.fillStyle = 'red';
    canvasContext.fillRect (ballX,100, 10,10);
}
function moveAll ( ) {
    console.log (ballX);
    ballX += 10;
}
```

Cleaning Up

1. Move the animation code to a new, separate function, moveAll
2. Declare a new variable for Frames Per Second, to be used by setInterval
3. Within setInterval, call an inline function which calls both moveAll and drawAll
4. Set the interval to 1 second (1000) divided by the FPS

```
window.onload = function ( ) {
    var FPS = 30;
    setInterval(drawAll, 100);
}
```

Cleaning Up

1. Move the animation code to a new, separate function, moveAll
2. Declare a new variable for Frames Per Second, to be used by setInterval
3. Within setInterval, call an inline function which calls both moveAll and drawAll
4. Set the interval to 1 second (1000) divided by the FPS

```
setInterval (function ( ) {
        moveAll ( );
        drawAll ( );
}   ,100);
```

Cleaning Up

1. Move the animation code to a new, separate function, moveAll
2. Declare a new variable for Frames Per Second, to be used by setInterval
3. Within setInterval, call an inline function which calls both moveAll and drawAll
4. Set the interval to 1 second (1000) divided by the FPS

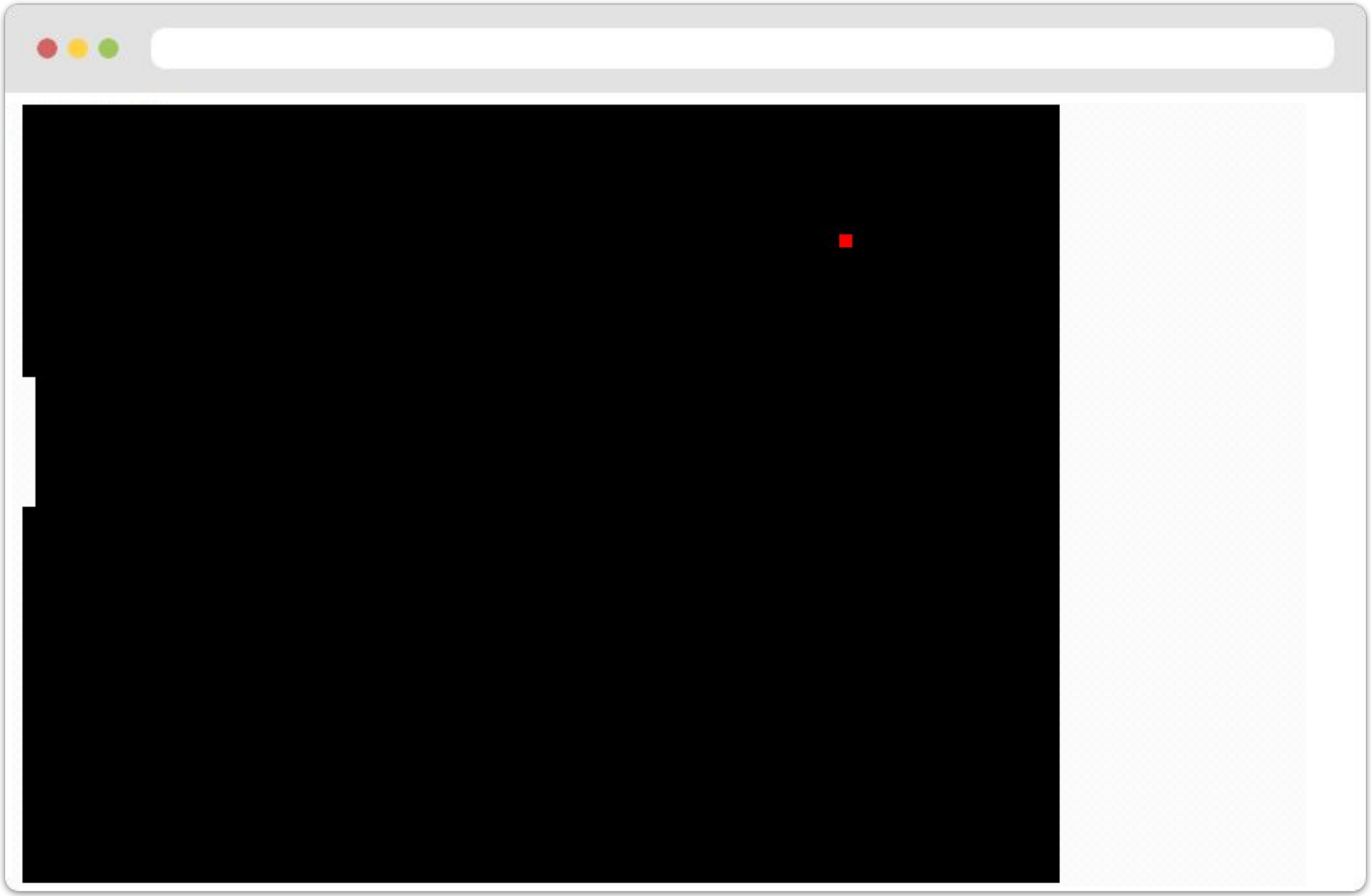
```
setInterval (function ( ) {
        moveAll ( );
        drawAll ( );
}   ,1000/FPS);
```

Cleaning Up

1. Move the animation code to a new, separate function, moveAll
2. Declare a new variable for Frames Per Second, to be used by setInterval
3. Within setInterval, call an inline function which calls both moveAll and drawAll
4. Set the interval to 1 second (1000) divided by the FPS

Chapter 2: Core Gameplay

Step 1: Bouncing the Ball



```
<script>
  var canvas;
  var canvasContext;
  var ballSpeedX = 5;
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

```
<script>
  var canvas;
  var canvasContext;
  var ballSpeedX = -5;
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

```
if (ballX > 800) {
    ballSpeedX = -ballSpeedX;
}
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

```
if (ballX > canvas.width) {
    ballSpeedX = -ballSpeedX;
}
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

```
if (ballX < 0) {
    ballSpeedX = -ballSpeedX;
}
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

```
function ( colour, X, Y, width, height) {
    canvasContext.fillStyle = colour;
    canvasContext.fillRect (X,Y, width,height);
}
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

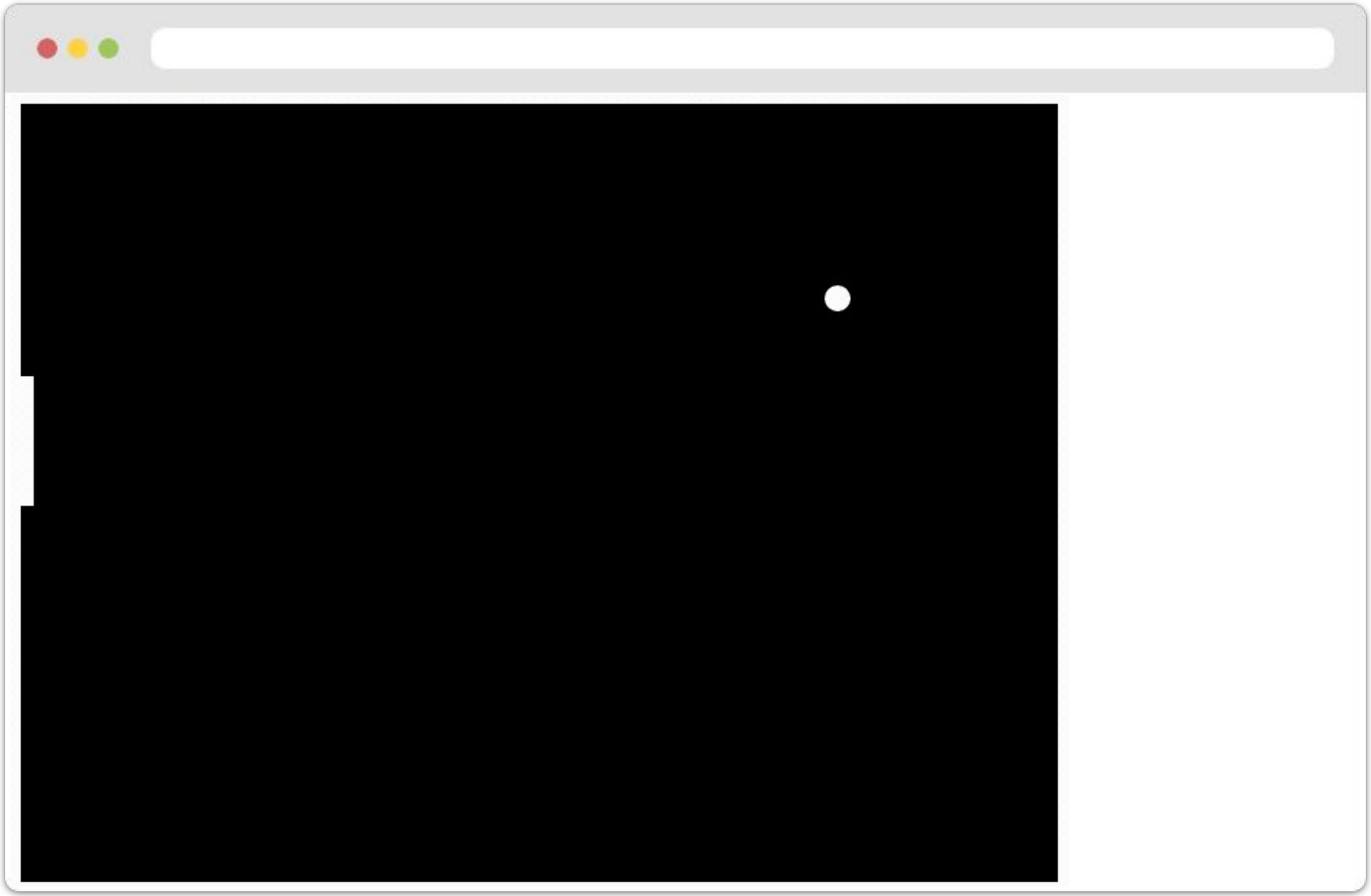
```
// comments are cool!
/*
   Multiple
   Lines
   Are
   Too!
*/
```

Bouncing the Ball

1. Declare a new variable, ballSpeedX to be used to move the ball
2. To change the ball direction, make the value negative
3. If ballX is greater than the canvas width, reverse its direction
4. Try to do this without hard coding!
5. Now try to apply this logic to the opposite side of the canvas
6. Create some functions to draw the rectangles, accepting position, dimensions and colour
7. Remember to comment up!

Chapter 2: Core Gameplay

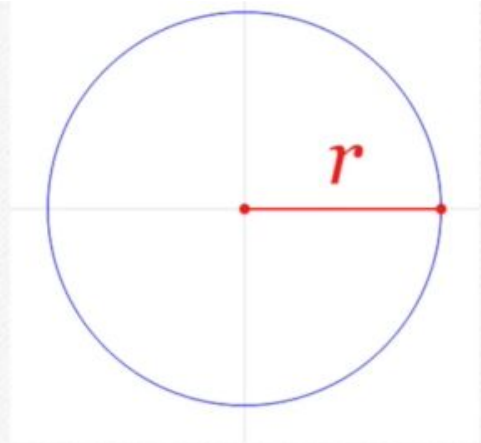
Step 2: Circle Draw Details



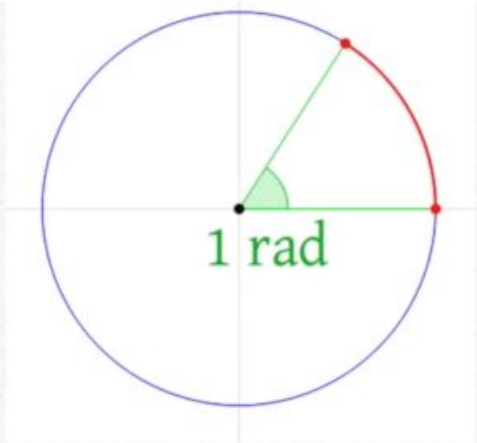
Circle Draw Details

1. Replace the ball draw code with a single fillStyle
2. Use canvasContext.beginPath() to define a shape to fill in
3. Use canvasContext.arc(ballX, 100, 10, 0, Math.PI*2, true)
4. Use canvasContext.fill()
5. Have a play with the .arc to see what the values represent

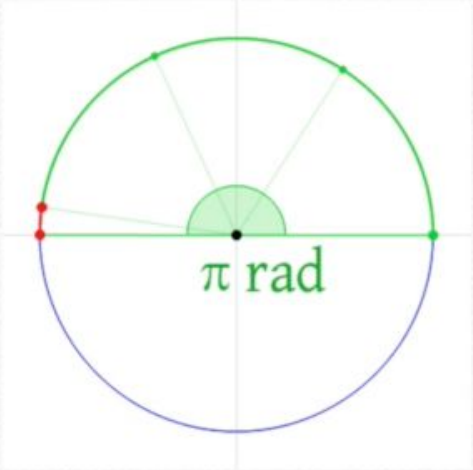
```
canvasContext.fillStyle = "white";
canvasContext.beginPath();
canvasContext.arc(ballX, 100, ballWidth/2, 0, Math.PI*2, true);
canvasContext.fill();
```



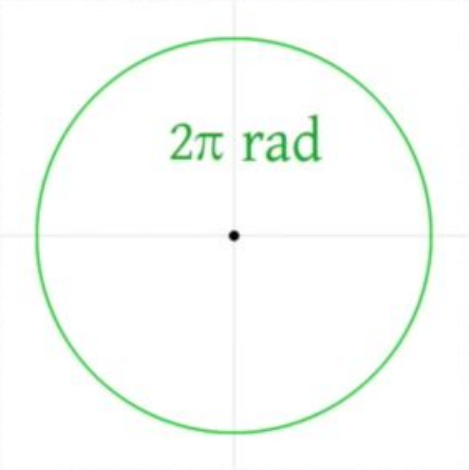
Circle showing its radius, r



Radius laid along edge, 1 radian



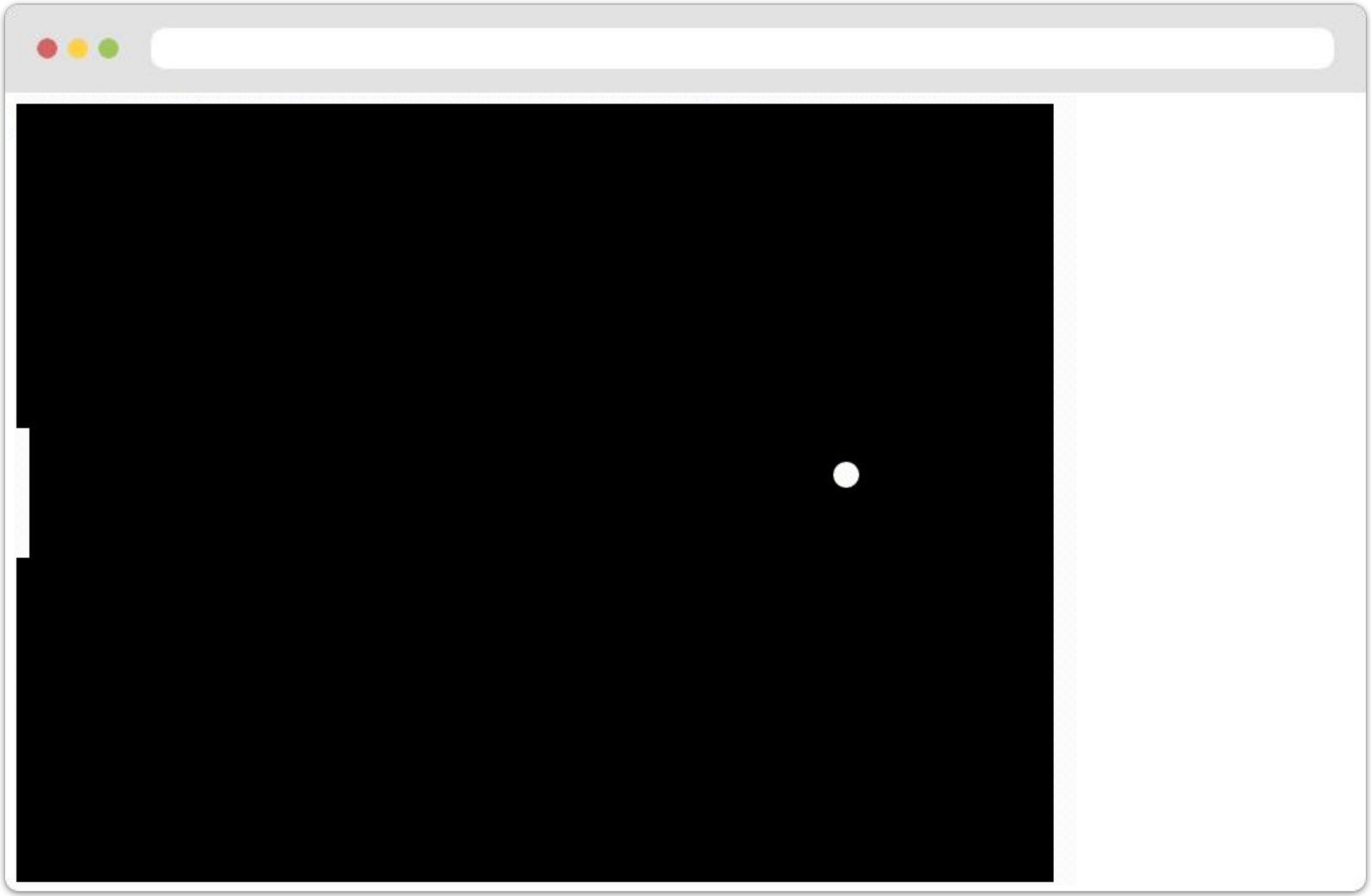
3.14159... (pi) radians is 180°



So 2*pi is one full spin, or 360°

Chapter 2: Core Gameplay

Step 3: Ball 2D Motion, Paddle



```
var ballY = 100;
var ballSpeedY = 5;
```

Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

```
ballY += ballSpeedY;

if (ballY >= canvas.height) {
      ballSpeedY = -ballSpeedY;
    } else if (ballY <= 0) {
      ballSpeedY = -ballSpeedY;
    };
```

Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

```
var ballWidth = 50;

const PADDLE_THICKNESS = 10;
const PADDLE_HEIGHT = 100;

var paddle1Y = 250;
```

Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

```
function calculateMousePos(evt) {
      var rect = canvas.getBoundingClientRect();
      var root = document.documentElement;
      var mouseX = evt.clientX - rect.left - root.scrollLeft;
      var mouseY = evt.clientY - rect.top - root.scrollTop;
      return {
        x: mouseX,
        y:mouseY
      }
    }

...

canvas.addEventListener('mousemove',  function (evt) {
       var mousePos = calculateMousePos(evt);
       paddle1Y = mousePos.y;
});
```

Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

```
colorRect (
    "white",
    0,
    paddle1Y,
    PADDLE_THICKNESS,
    PADDLE_HEIGHT);
```

Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

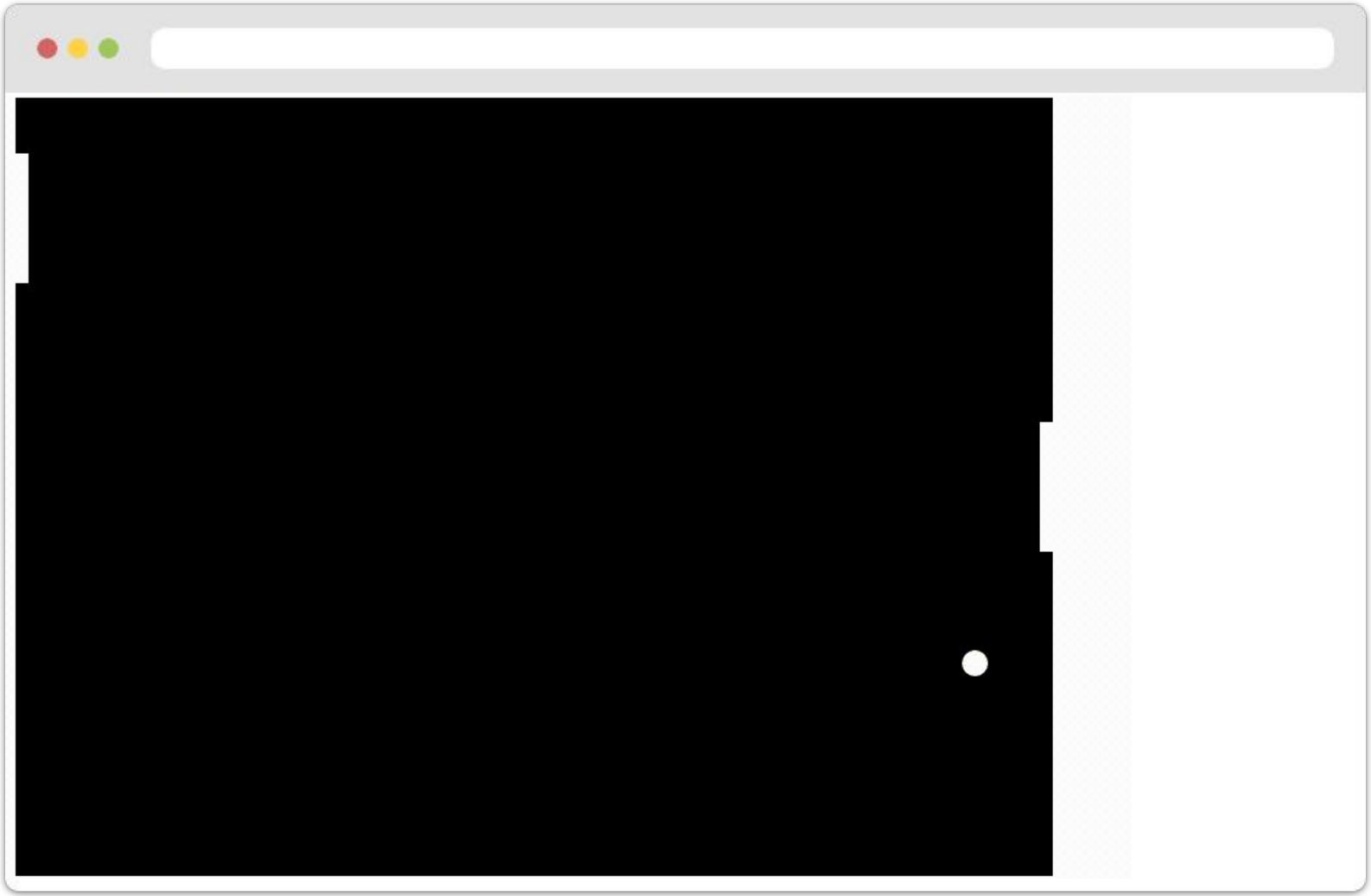
```
canvas.addEventListener('mousemove',  function (evt) {
        var mousePos = calculateMousePos(evt);
        paddle1Y = mousePos.y - (PADDLE_HEIGHT/2);
});
```

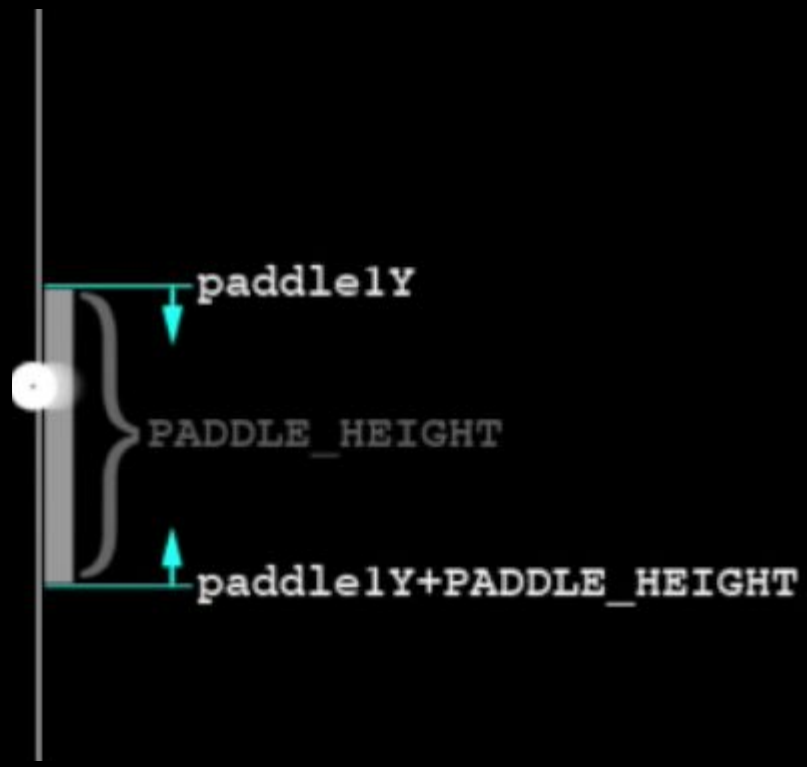
Ball 2D Motion, Paddle

1. Create two new variables for the Y speed and position
2. Under moveAll, set up ballY(You need to do 3 things)
3. Replace the hard coding in the drawAll function
4. Declare a new variable, paddle1Y for the position of the left paddle
5. Declare a new constant, PADDLE_HEIGHT Underneath, we'll set up a new function, calculateMousePos
6. We'll addEventListener to call the function when the mouse moves
7. Now update the paddle's draw code
8. Finally, we adjust the mousePos code to place the cursor in the centre

Chapter 2: Core Gameplay

Step 4: Ball Reset and Collision




paddle1Y
PADDLE_HEIGHT
paddle1Y+PADDLE_HEIGHT

```
function ballReset ( ) {
    ballX = canvas.width/2;
    ballY = canvas.height/2;
}
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

```
function ballReset ( ) {
    ballX = canvas.width/2;
    ballY = canvas.height/2;

    ballSpeedX = -ballSpeedX;
}


ballReset();
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

```
if (ballY < 0) {
      ballSpeedY = -ballSpeedY;
}

if (ballX < 0) {
      if (ballY > paddle1Y &&
          ballY < paddle1Y+PADDLE_HEIGHT) {
         ballSpeedX = -ballSpeedX;
      } else {
         ballReset();
      }
}
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

```
var paddle2Y = 250;
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

```
const PADDLE_THICKNESS = 10;
```

```
drawRect ("white",
canvas.width-PADDLE_THICKNESS,paddle2Y,
PADDLE_THICKNESS, PADDLE_HEIGHT);
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

```
canvas.addEventListener('mousemove',  function (evt)
        var mousePos = calculateMousePos(evt);
        paddle2Y = mousePos.y - (PADDLE_HEIGHT/2);
});
```

Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

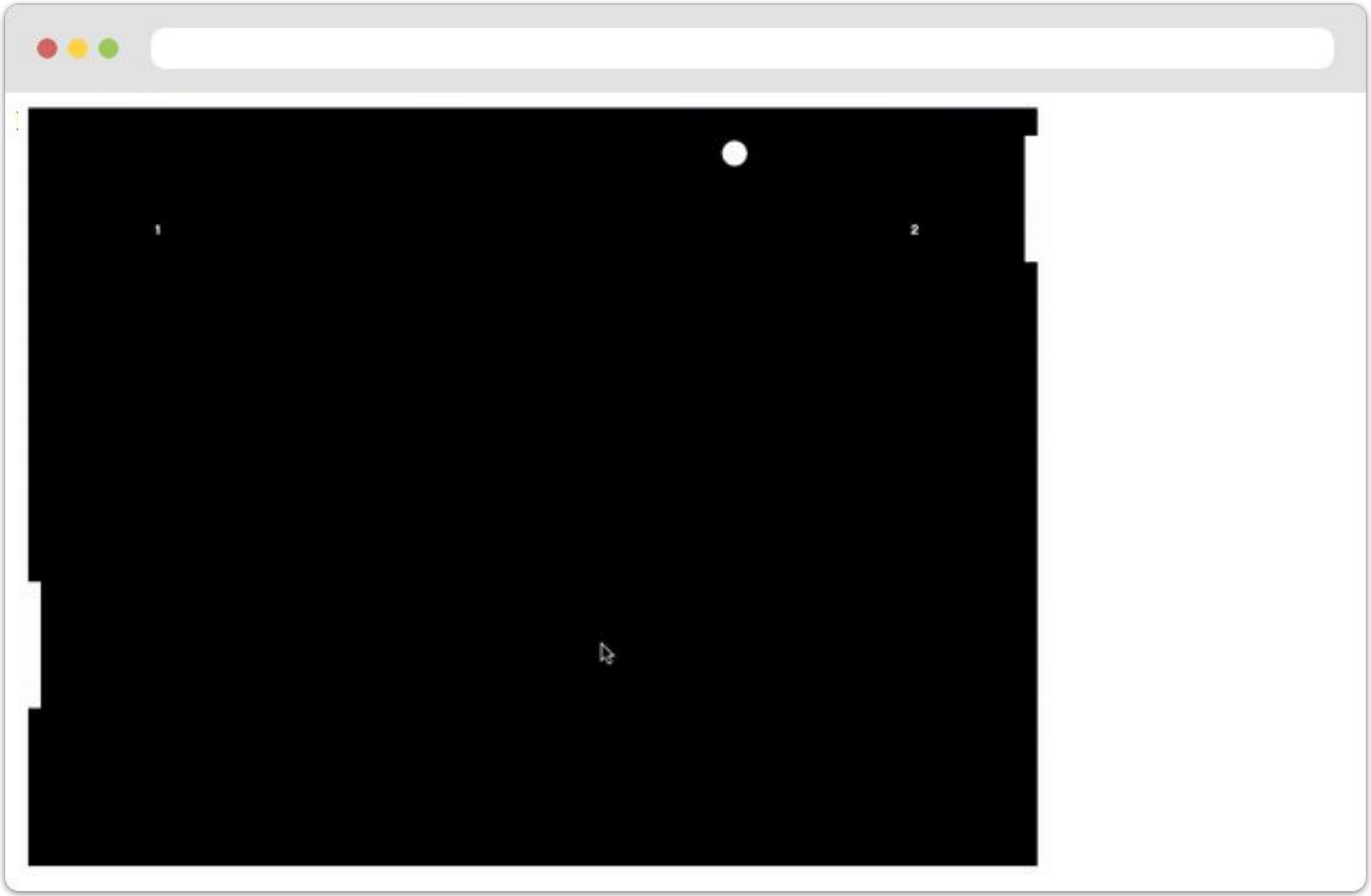
```
if (ballX > canvas.width) {
        if (ballY > paddle2Y &&
            ballY < paddle2Y+PADDLE_HEIGHT) {
           ballSpeedX = -ballSpeedX;
        } else {
           ballReset();
        }
}
```

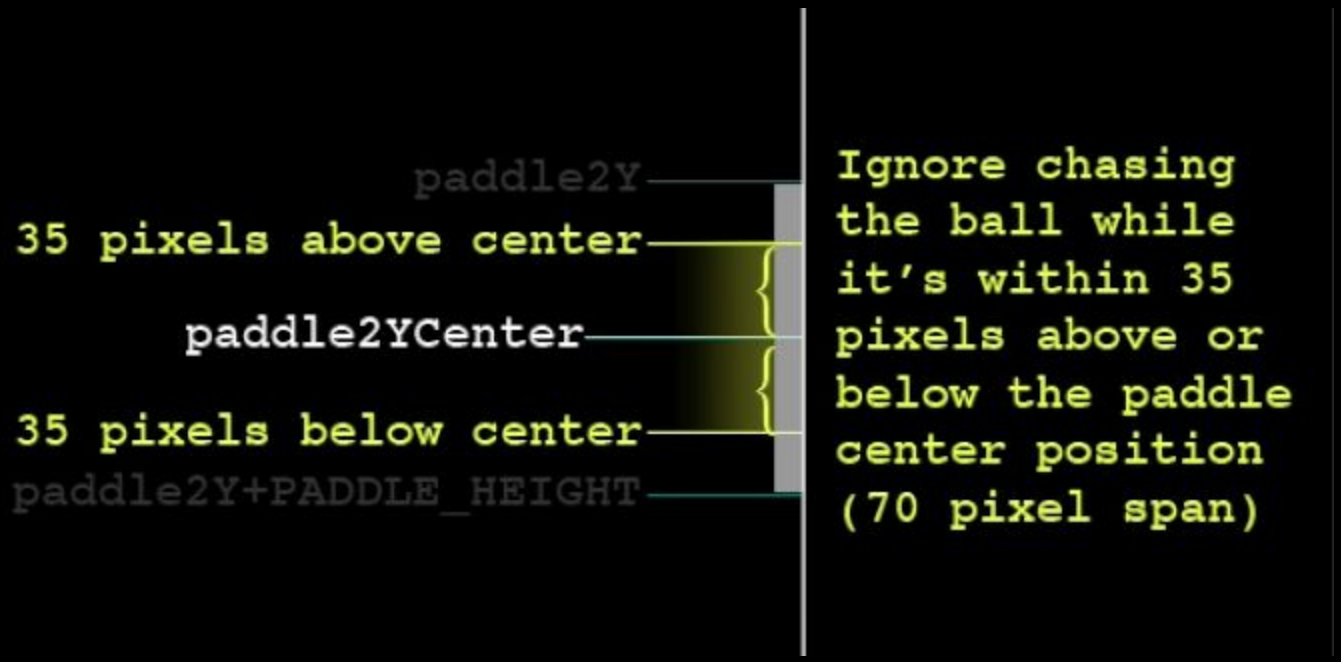
Ball Reset and Collision

1. Create a ballReset function, to place the ball in the centre
2. Under moveAll, comment out the code that flips the ball if it goes below 0
3. Call ballReset there
4. Move the commented out line into the ballReset function
5. Under moveAll, we need to add an if to deflect the ball if it hits the paddle, else ballReset
6. Test your code! Remember to check the edges
7. Now create the variables for a second paddle
8. In the draw code, duplicate the first paddle's code, and adjust for new variables and the paddle 2 position
9. Try to avoid hard coding, you'll need a new constant, PADDLE_THICKNESS!
10. Back at the addEventListener, change it to paddle2Y for testing
11. Under moveAll, copy the if(ballX < 0), and alter it to create if(ballX > canvas.width) for paddle 2

Chapter 2: Core Gameplay

Step 5: Paddle AI and Scoring


1
2


paddle2Y
35 pixels above center
paddle2YCenter
35 pixels below center
paddle2Y+PADDLE_HEIGHT
Ignore chasing the ball while it's within 35 pixels above or below the paddle center position (70 pixel span)

```
computerMovement ( );

function computerMovement ( ) {
    if (paddle2Y < ballY) {
        paddle2Y += 6;
    } else if (paddle2YCentre > ballY) {
        paddle2Y -= 6;
    }
}
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
var paddle2YCentre;

function computerMovement ( ) {
    var paddle2YCentre = paddle2Y + (PADDLE_HEIGHT/2);
    if (paddle2YCentre < ballY) {
        paddle2Y += 6;
    } else if (paddle2YCentre > ballY) {
        paddle2Y -= 6;
    }
}
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
function computerMovement ( ) {
    var paddle2YCentre = paddle2Y + (PADDLE_HEIGHT/2);
    if (paddle2YCentre < ballY - 35) {
        paddle2Y += 6;
    } else if (paddle2YCentre > ballY + 35) {
        paddle2Y -= 6;
    }
}
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
canvasContext.fillText("Some Text", 100, 100);
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
var player1Score = 0;
var player2Score = 0;
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
if (ballX >= canvas.width-(ballWidth/2)) {
    if (ballY > paddle2Y && ballY < paddle2Y+PADDLE_HEIGHT) {
      ballSpeedX = -ballSpeedX;
    } else {
      ballReset();
      player1Score++;
    }
  }

  if (ballX <= 0 + (ballWidth/2)) {
    if (ballY > paddle1Y && ballY < paddle1Y+PADDLE_HEIGHT) {
      ballSpeedX = -ballSpeedX;
    } else {
      ballReset();
      player2Score++;
    }
  }
```

Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

```
canvasContext.fillText (player1Score, 100, 100);
canvasContext.fillText (player2Score, canvas.width - 100, 100);
```

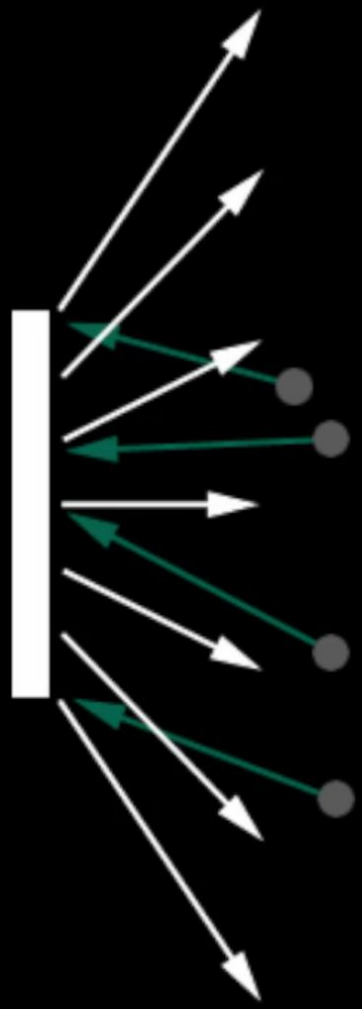
Paddle AI and Scoring

1. Create a new computerMovement function, called under moveAll
2. If paddle2Y is above the ball, move it down a little, else, move it up
3. Test the right paddle's movement, what two things do you spot?
4. Make a new variable for the paddle's centre, and adjust the if below
5. If the ball is 35 pixels above or below the centre, then move the paddle - this fixes the shaking motion
6. Use 'canvasContext.fillText' to add some text (under the existing draw code)
7. Declare a 'player1Score' and a 'player2Score' variable, both starting at 0
8. If it gets past player 1, player 2 should score a point and vice versa
9. Replace the text with code to display player1Score and player2Score

Chapter 3: Polishing Up

Step 1: Ball Control & Winning


click to continue



```
var  deltaY = ballY - (paddle1Y+PADDLE_HEIGHT/2);
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

ballSpeedY = deltaY * 0.35;

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
var deltaY = ballY - (paddle2Y+PADDLE_HEIGHT/2);
ballSpeedY = deltaY * 0.35;
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

const WINNING_SCORE = 3;

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
player2Score++;
ballReset();

player1Score++;
ballReset();
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
function ballReset () {
    if (player1Score >= WINNING_SCORE ||
        player2Score >= WINNING_SCORE) {
      player1Score = 0;
      player2Score = 0;
    }

    ballSpeedX = -ballSpeedX;
    ballX = canvas.width/2;
    ballY = canvas.height/2;
}
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

// computerMovement ();

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
if (player1Score >= WINNING_SCORE ||
    player2Score >= WINNING_SCORE) {
        player1Score = 0;
        player2Score = 0;

        showingWinScreen = true;
}
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
var showingWinScreen = false;


function moveAll () {
    if (showingWinScreen) {
        return;
    }
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

```
function drawAll () {
    drawRect ("black", 0,0, canvas.width, canvas.height);
    if (showingWinScreen) {
        return;
    }
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

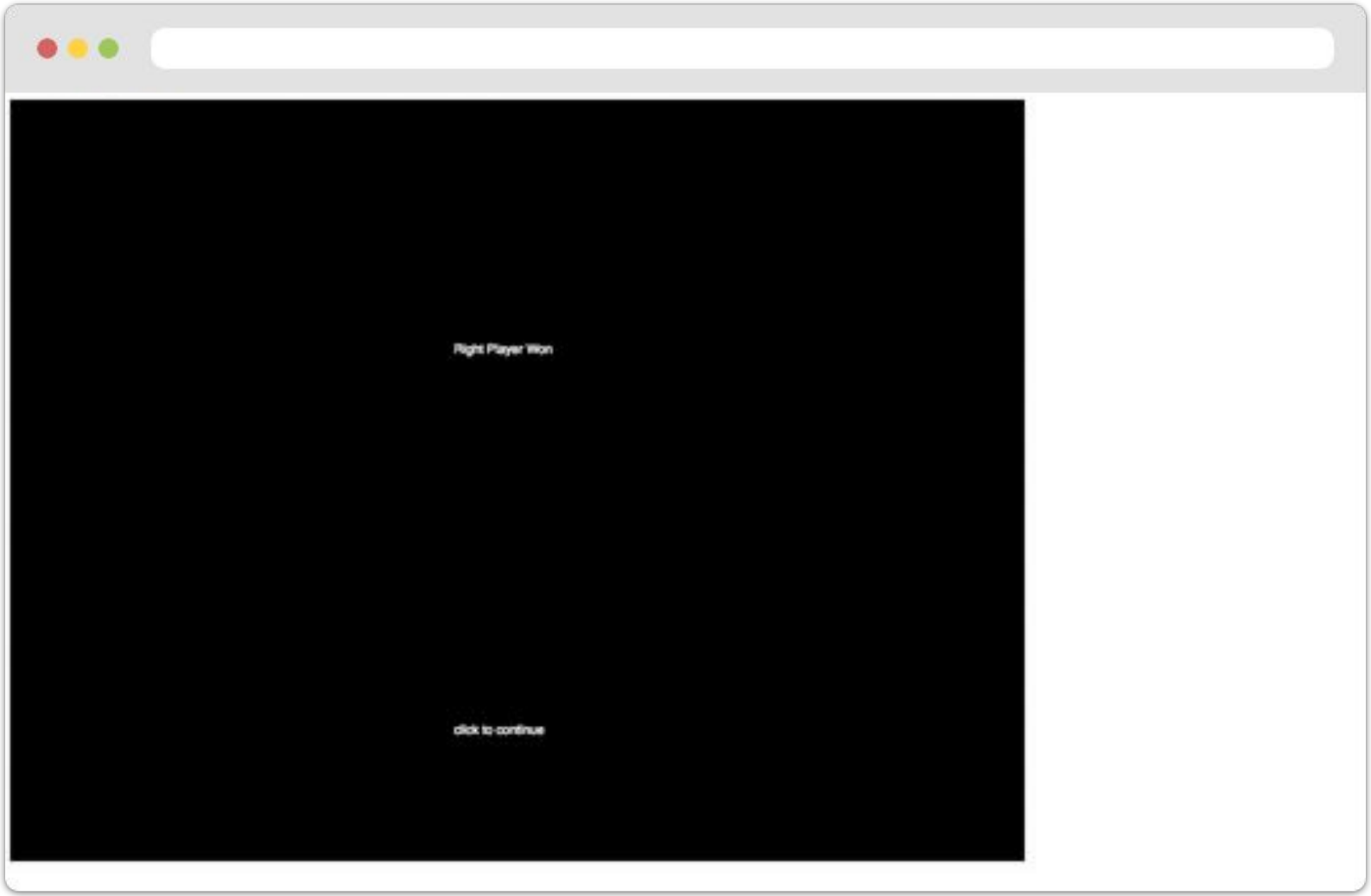
```
function drawAll () {
	drawRect ("black", 0,0, canvas.width, canvas.height);
	if (showingWinScreen) {
		canvasContext.fillStyle = 'white';
		canvasContext.fillText("Click to continue", 100, 100);
		return;
	}
```

Ball Control & Winning

1. If you leave the game running without input, you can see the balancing
2. Now, we need to introduce some ball control, but can you think why?
3. In the ball movement code, we'll declare a variable, deltaY=ballY-(paddle1Y+PADDLE_HEIGHT/2)
4. This will be the deviation from the centre of the paddle
5. We could directly set ballSpeedY from deltaY, but to prevent extreme speeds, we'll set it to about a third
6. Adapt and apply the code to work for the other paddle, be sure to test it!
7. Create a constant, WINNING_SCORE
8. Adjust the ball movement code so the ball is reset after the score is increased, then comment up!
9. When the ball resets, if one of the players has exactly/more than the winning score, reset the scores
10. Test this, then comment out the computerMovement call to test it on the other side
11. Declare a new boolean variable, showingWinScreen, and change it to true when someone wins
12. At the top of moveAll, add an if(showingWinScreen), and use 'return;' to bail out of excuting the function
13. Apply the same to the drawAll function, but keep the black background
14. You can also add some "Click to continue" text. Remember to test - can you see the problem?

Chapter 3: Polishing Up

Step 2: Mouse Click, Draw Net


Right Player Won
click to continue

```
canvas.addEventListener('mousedown', handleMouseClick);
```

Mouse Click, Draw Net

1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

```
function handleMouseClick(evt) {
	if (showingWinScreen) {
		player1Score = 0;
		player2Score = 0;
		showingWinScreen = false;
	}
}
```

Mouse Click, Draw Net

1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

```
function ballReset () {
        if (player1Score >= WINNING_SCORE ||
            player2Score >= WINNING_SCORE) {
                showingWinScreen = true;
        }

        ballSpeedX = -ballSpeedX;
        ballX = canvas.width/2;
        ballY = canvas.height/2;
}
```

Mouse Click, Draw Net

1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

```
if (player1Score >= WINNING_SCORE) {
        canvasContext.fillText("You Won!", 350, 200);
} else if(player2Score >= WINNING_SCORE) {
        canvasContext.fillText("You Lost.", 350, 200);
}
```

Mouse Click, Draw Net

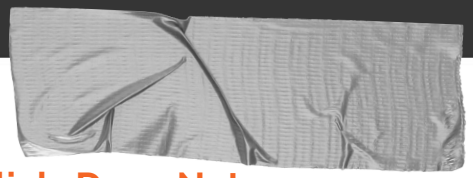
1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

```
function drawNet() {
	for (var i = 0; i < canvas.height; i += 40) {
		drawRect ('white', canvas.width/2-1,i, 2,20);
	}
}
```

Mouse Click, Draw Net

1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

drawNet();



Mouse Click, Draw Net

1. Above the previous addEventListener, we'll add a new one listening for 'mousedown'
2. We'll create a new handleMouseClick function above to set it up
3. If the win screen is showing, zero out the scores and turn off the win screen
4. Remember to remove the score reset from the ball movement code
5. Copy the if(player1Score>=WINNING_SCORE||player2Score>=WINNING_SCORE), to create an if in drawAll to display who won
6. We'll now add a drawNet function above drawAll, in which we're going to use a for loop to draw a net
7. The loop starts at zero, and goes up to canvas.height in intervals of 40 each time
8. Inside the loop, we use colorRect(canvas.width/2-1,i,2,20,'white') to draw repeating rectangles
9. Call the 'drawNet' function just above where the paddles are drawn

Chapter 3: Polishing Up

Optional: Publish Game

Extract from “Creating a Website”

Using Git

An introduction to Version Control, Git and GitHub

What is Git?

- **Version Control** refers to keeping track of changes made to a file or directory (folder), it can be found in word processors like Word and Google Docs
- **Git** is version control software created in 2005 by Linus Torvalds for the development of the Linux kernel
- **Repositories** are central locations in where data is stored and managed.
- **GitHub** was founded in 2008, built on top of git, it is used to host over 35 million Git repositories on its main site, GitHub.com
- We’re going to use GitHub.io, its **free** web hosting service for our website.

N.B. You can create the site without GitHub.io, we are only using it to host our site on the internet. Without hosting, no one can access our website!

Extract from “Creating a Website”

First, set up your account

1. Go to GitHub.com
2. Fill in the signup form, if you don’t want to use your personal email, use your school one!
3. You’ll probably need to confirm your email address

Extract from “Creating a Website”

Then, create the repository

1. Click the + in the top right corner
2. Select “New repository”
3. Name it
4. Tick “Initialize this repository with a README”
5. Click “Create repository”

Extract from “Creating a Website”

Finally, you can set up the website

1. Open the dropdown that says “Branch: master”
2. Type “gh-pages”, then click “Create branch :gh-pages”
3. Seeing as we are only going to use this branch, we can make it the default, by going to Settings → Branches
4. Back in the repository view, click on “Create new file”
5. Name it “index.html”
6. Type something, and commit the new file
7. Go to username.github.io/repositoryname