PONG

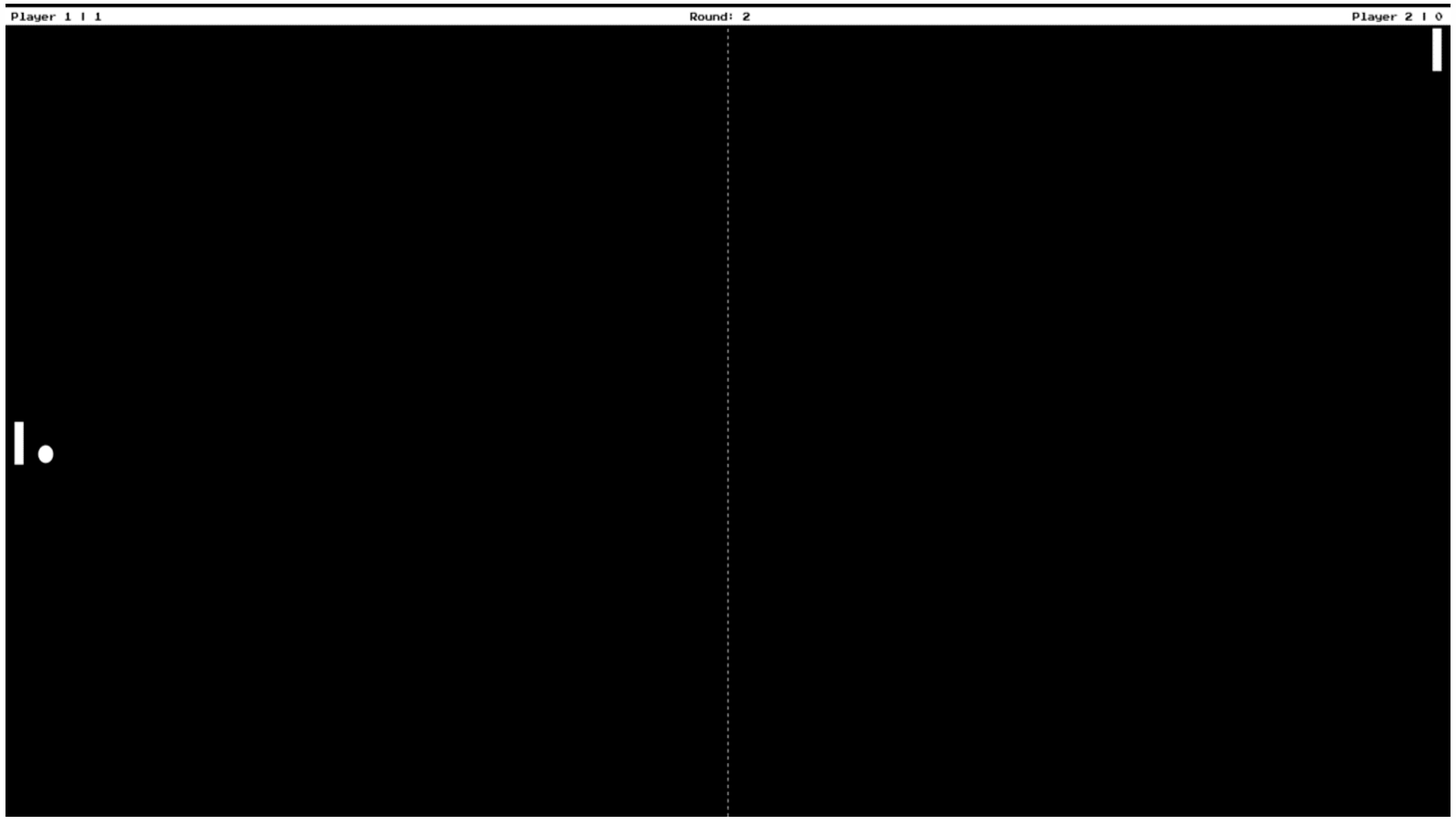
SESSION 1

PROJECT INFO

- PONG IS A TABLE TENNIS STYLE ARCADE GAME.
- IT CAN HAVE 1-2 PLAYERS (WE WILL DO 1 + AI)

WHAT OBJECTS CAN YOU SEE?

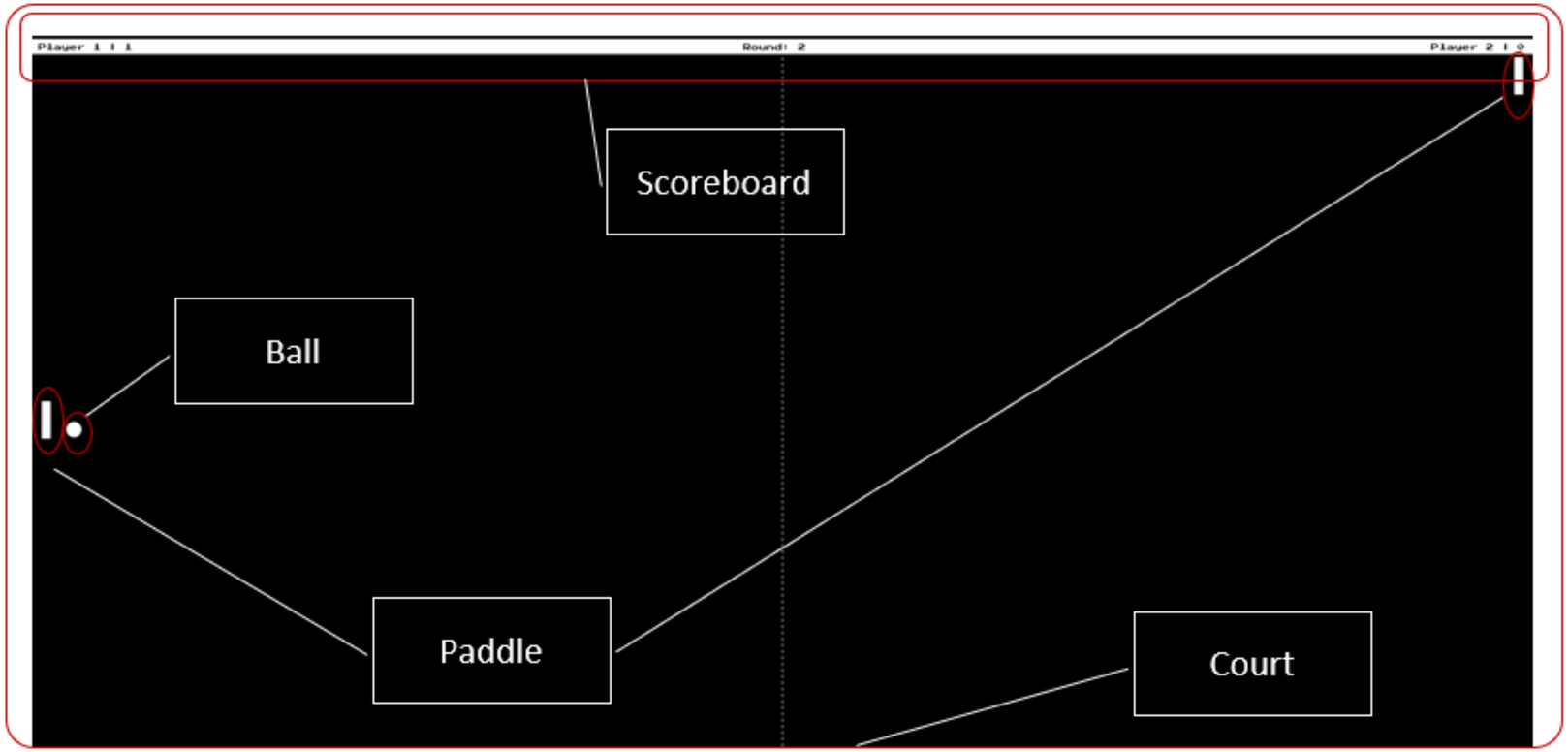
(HINT: FOUR ARE VISIBLE HERE)



WHAT OBJECTS CAN YOU SEE?

(HINT: FOUR ARE VISIBLE HERE)

These will be some of the classes within the code



SESSION MATERIALS

FORK OR DOWNLOAD THE GITHUB REPO:
HTTPS://GITHUB.COM/INFPALS/IP2023-BIG-PROJECT-1-UPDATED-TEMPLATE

If you came to the last session, you may want to copy the new SETTINGS constants added to the template, this will make work easier later.

FUNCTIONS, VARIABLES AND CLASSES

Variables can be defined different ways, here is an example, using the let keyword:

```
Let variable = 1
```

Functions can be defined different ways, here is an example using the function keyword:

```
function exampleFunction(x, y, z) {
    return x + y + z
}
```

FUNCTIONS, VARIABLES AND CLASSES

Classes can be defined using the class keyword, an example of a class with a constructor, variables and a method are shown below for reference.

```
class ExampleClass {
    constructor(var1, var2) {
        this.var1 = var1;
        this.var2 = var2;
    }

    exampleMethod() {
        let string = "This is an example string";
        return string
    }
}
```

When you want to instantiate a new object (e.g. a new ExampleClass) you can use the new keyword, e.g. **let exampleObject = new ExampleClass("exampleString", "exampleString2")**

AN IMPORTANT FUNCTION

The JS function *setInterval(f, rate)* takes a function ‘f’ and calls the function at specified intervals (in milliseconds).

Easy way to write this is to embed function with ‘function’ keyword, for example:

```
setInterval(function() {
    //do something
}, 1000);
```

ATTENDANCE FORM

Please fill in to let us know you came and that we should keep planning similar events.

LET'S GET STARTED!

- Within a constructor for Game we want to pass the canvas and store it in a field.

We also need a start method that runs a game loop

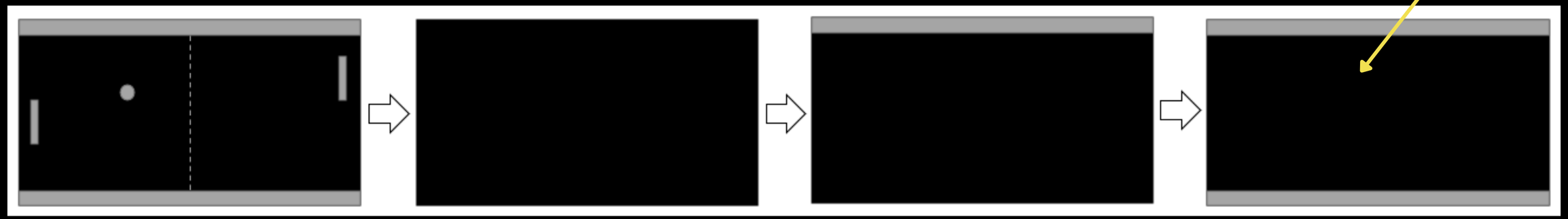
- In SETTINGS add an FPS variable and set it to 60
- In the start method, add a setInterval function with the rate equal to 1/SETTINGS.FPS* 1000 (to get the length in ms) - see previous slide for more info on setInterval.
- Within the setInterval function, add a comment //update, we will come back to this later.

GAME LOOP (CONT'D)

- To ensure the game is running at constant time, you must find the change in time (dT) between frames.
  - Before setInterval, create a variable called previousTime with the value of Date.now()
  - At the top of setInterval, create a variable called now, also with the value of Date.now()
  - To find dT, find the difference in time and then divide by 1000.0 to get in seconds.
- After the //update comment, make sure to update previousTime to equal now

CREATE THE COURT

- As before, create the constructor passing the canvas and then store the canvas in the class.
- Then create a draw method*, passing the canvas as a parameter.
  - First, set your context variable let ctx = canvas.getContext('2d’)
  - Paint over the background with the background colour and then paint each border (example shown below)
  - You may wish to set constants for margins, colours, etc. in the SETTINGS
  - Relevant functions include:
    - *ctx.fillStyle = COLOUR and ctx.fillRect(x, y, width, height)*



**Remember a method is just a function within a class*

CREATE THE COURT (CONT'D)

- You should now have a bordered court (the method won’t do anything until called).
- To add the centre line
  - https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/setLineDash
    - Try and work together and ask the leaders for help if you get stuck!
- In the Game constructor, you can now instantiate a Court.
- Then, create a draw method in Game which clears the space and draws the court (using the draw method from Court)
  - Remember to get the 2d context
  - A useful function is *clearRect(x, y, width, height)*

LET'S TEST THE CODE

- At the top of *Game.start()* write *let that = this*
  - This adds a reference to the outer Game class, which will lose the ‘this’ reference when entering the inner function of setInterval
    - If this doesn't make too much sense, follow through the logic keeping track of what 'this' refers to, when it enters a new inner funnction, it changes.
  - Under *//update*, write *that.draw()* to draw the game.
    - We will remove this line later when implementing the game update loop.

CREATE THE PADDLE

This will be a similar idea to the court, so less details are given but refer back if needed.

- Create a constructor to initialise and store the variables of coordinates x and y, width and height, the player number, and a court.
- Then create a draw method, the same way as before.
  - Hint: the paddle is a rectangle with coordinates, width and height stored in the class, this gives you everything you need to draw it.
- In the Court constructor, instantiate 2 paddles, one for each player.
  - To reference the court from within the court, you can use the *this* keyword.
- In the court draw method, draw the two paddles using the *paddle.draw(canvas)* methods.

CREATE THE BALL

- Create a constructor to initialise and store the variables of coordinates x and y, radius, and a court.
- Then create a draw method, the same way as before.
  - Hint: Circles are created using the *ctx.arc(x, y, radius, startAngle, endAngle)* method, where the angles are in **radians**. You will first need to begin the arc path using ctx.beginPath()
  - The ball can be filled with *ctx.fill()* but don't forget to set a fillStyle as before!
- Instantiate the ball in the Court constructor, placing it in the centre.
- Then, draw the ball, as you did with the paddles.

NEXT TiME...

You should now have a basic UI.

Next time we will cover:

- Animating the canvas
- Implementing collision detection
- Game logic & end game scenarios