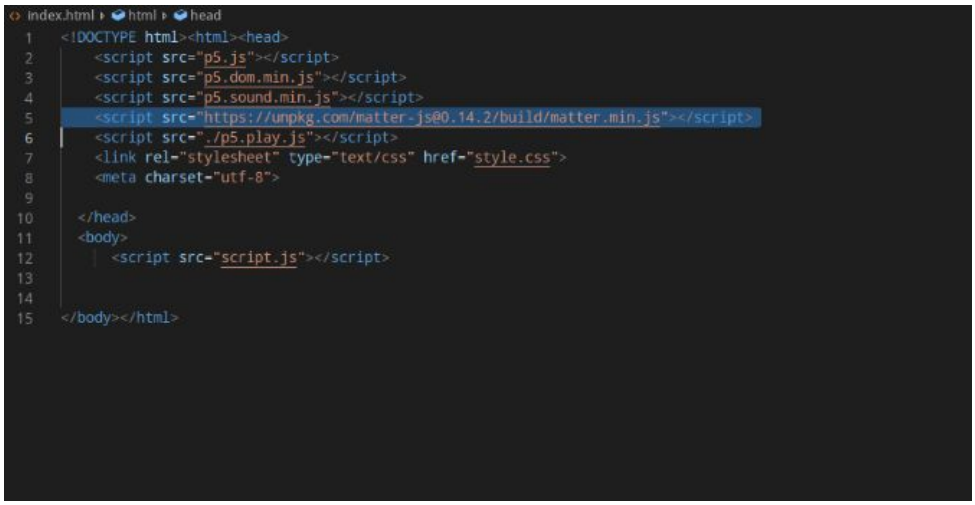
Physics Engine

What we did:

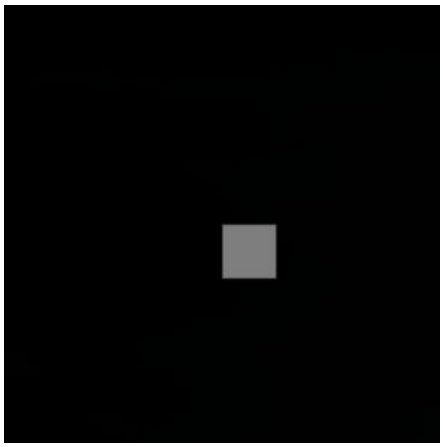
- Use a physics engine to create a world and the objects in them
- Integrate the physics engine with the p5 code to create interactive objects following the rules of physics in this world.
- Tune the physics engine to change the behaviour of the objects in this world

How we did it: In our projects, we will be using matter.js as our physics engine.
Step 1: Get the boilerplate from the GitHub.
To use the matter.js library in our project, modify the index.html file to add the script tag for the library. Add the script tag with src as the link for the file:

```
<script src="https://unpkg.com/matter-js@0.14.2/build/matter.min.js"></script>
```

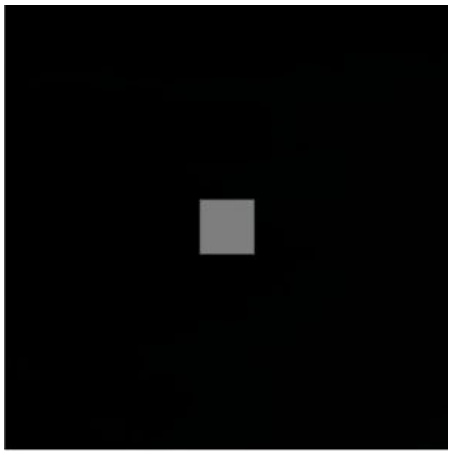


Step 2: Open the script.js file and create a canvas; draw a rectangle at the centre in the canvas



Step 3: Instruct the computer to take the x and y coordinates to be at the centre of the rectangle. Tell the computer: rectMode(CENTER)

```
JS script.js ▸ draw
1  function setup(){
2      var canvas = createCanvas(400,400);
3  }
4
5  function draw(){
6      background(0);
7      rectMode(CENTER);
8      rect(200,200,50,50);
9  }
```



Step 4: Create a Ground using the physics engine. Write code to namespace Matter.World, Matter.Engine and Matter.Bodies.

```
JS script.js ▸ draw
1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4
5
6  function setup(){
7      var canvas = createCanvas(400,400);
8  }
9
10 function draw(){
11     background(0);
12     rectMode(CENTER);
13     rect(200,200,50,50);
14 }
```

Step 5: Create a physics engine

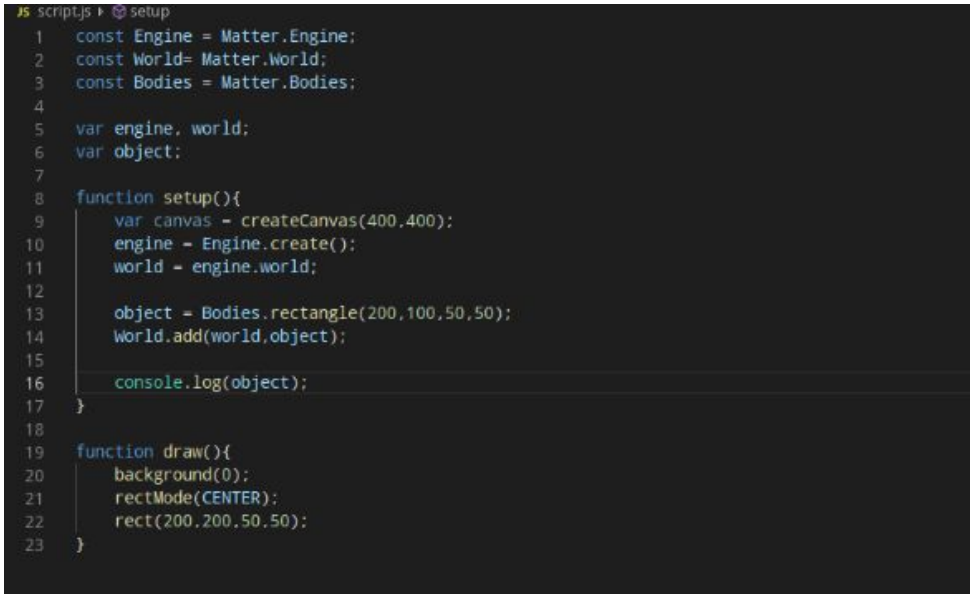
```
JS script.js ▸ setup
1   const Engine = Matter.Engine;
2   const World= Matter.World;
3   const Bodies = Matter.Bodies;
4
5   var engine, world;
6
7   function setup(){
8       var canvas = createCanvas(400,400);
9       engine = Engine.create();
10      world = engine.world;
11  }
12
13  function draw(){
14      background(0);
15      rectMode(CENTER);
16      rect(200,200,50,50);
17  }
```

Step 6: Make an object in this world. Use Bodies to create a body in this world— create a rectangular body in the world just above the previous rectangle.

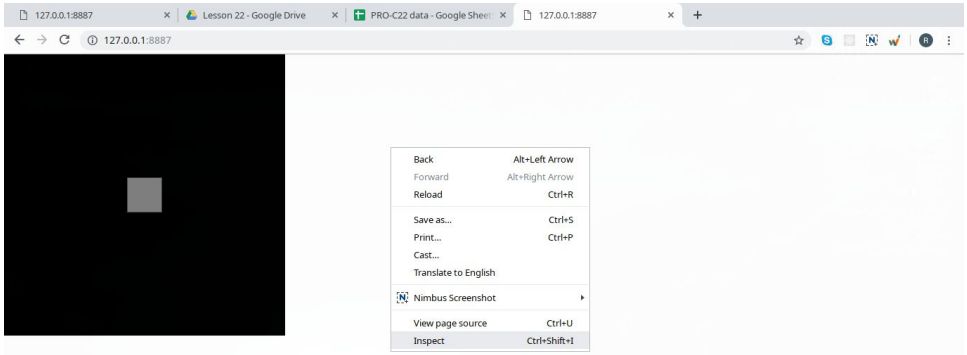
```
JS script.js ▸ setup
1   const Engine = Matter.Engine;
2   const World= Matter.World;
3   const Bodies = Matter.Bodies;
4
5   var engine, world;
6   var object;
7
8   function setup(){
9       var canvas = createCanvas(400,400);
10      engine = Engine.create();
11      world = engine.world;
12
13      object = Bodies.rectangle(200,100,50,50);
14
15  }
16
17  function draw(){
18      background(0);
19      rectMode(CENTER);
20      rect(200,200,50,50);
21  }
```

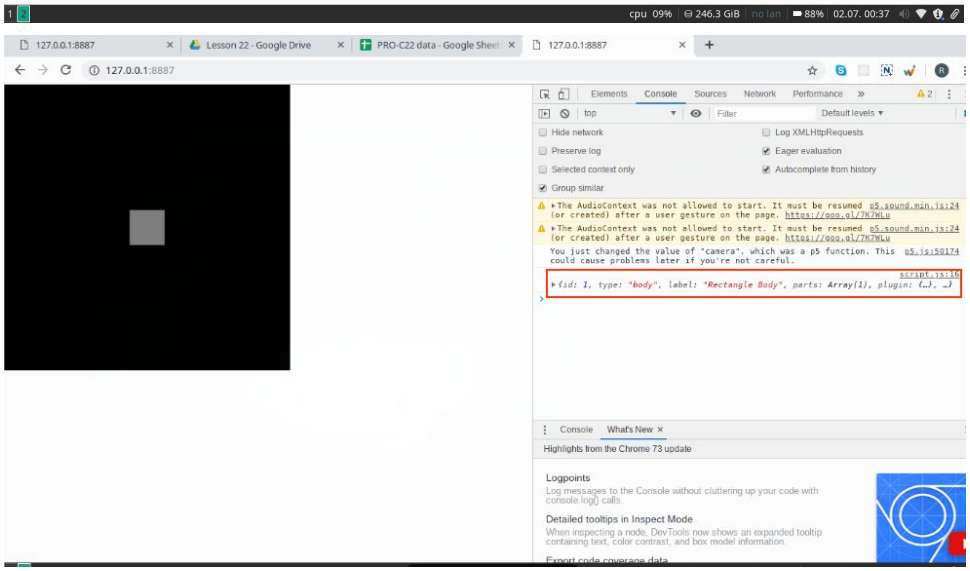
Step 7: Write the code to add the body to the world.

Step 8: We see another body other than the rectangle we had drawn. See the value inside the object by printing it on the console by console logging the object.



Step 9: Right-click inside the browser and see the console output by pressing on inspect.

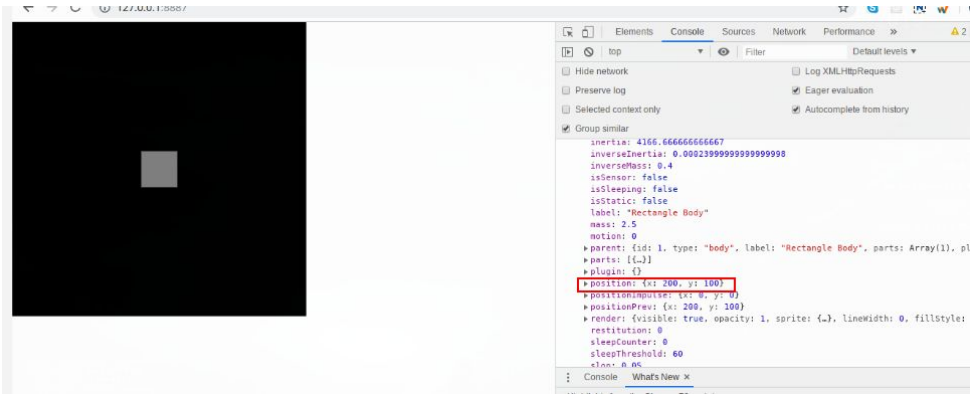




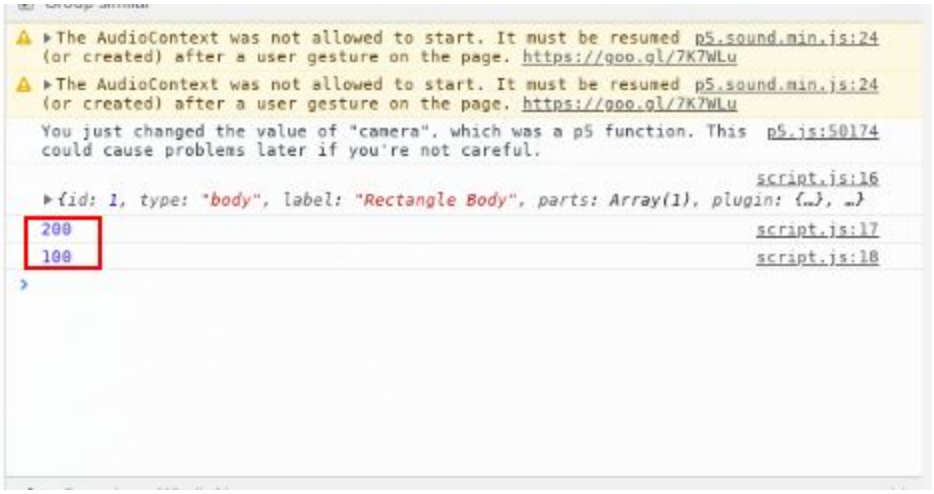
Step 10: Print this object type on the console. You just need to write console.log(object.type)

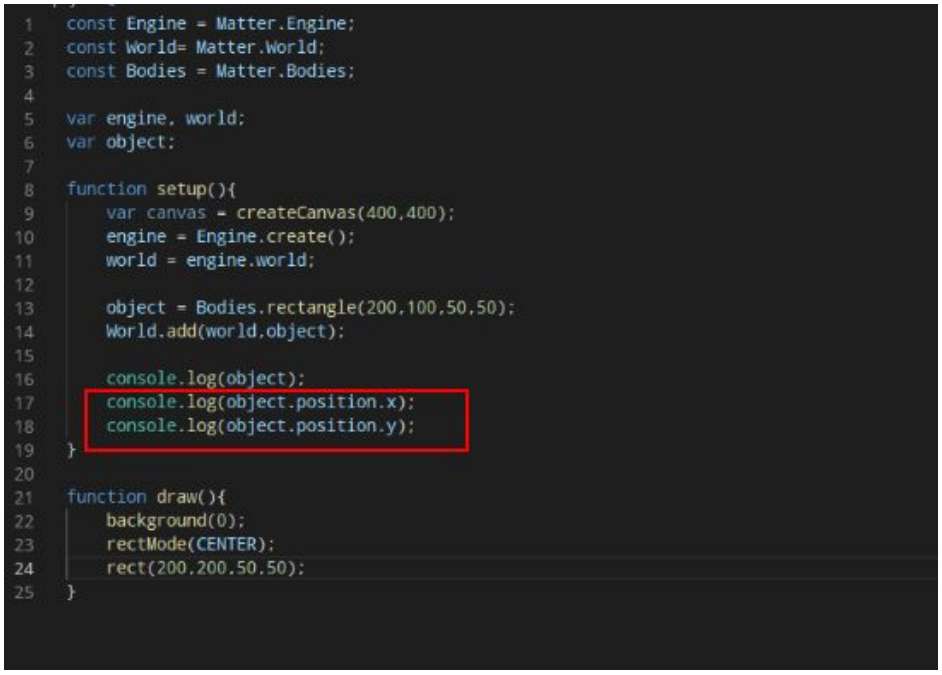
```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var object;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    object = Bodies.rectangle(200,100,50,50);
    World.add(world,object);

    console.log(object);
    console.log(object.type);
}

function draw(){
    background(0);
    rectMode(CENTER);
    rect(200,200,50,50);
}
```

Step 11: click on the arrow to the left of the object we have created, you will see it has many attributes. It also has an attribute called position.



Print the x and y of this object: console.log(object.position.x) console.log(object.position.y)





```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var object;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    object = Bodies.rectangle(200,100,50,50);
    World.add(world,object);

    console.log(object);
    console.log(object.position.x);
    console.log(object.position.y);
}

function draw(){
    background(0);
    rectMode(CENTER);
    rect(200,200,50,50);
}
```

Step 12: In the draw function - instead of drawing a rectangle at any position, draw it at the position of our object.

```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var object;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    object = Bodies.rectangle(200,100,50,50);
    World.add(world,object);

    console.log(object);
}

function draw(){
    background(0);
    Engine.update(engine);
    rectMode(CENTER);
    rect(object.position.x,object.position.y,50,50);
}
```

Step 13: For a static rectangle write this code

```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var object;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    var object_options ={
        isStatic: true
    }

    object = Bodies.rectangle(200,100,50,50,object_options);
    World.add(world,object);

    console.log(object);
}

function draw(){
    background(0);
    Engine.update(engine);
    rectMode(CENTER);
    rect(object.position.x,object.position.y,50,50);
}
```

Step 14: Create a ball which bounces on the ground like a tennis ball and then comes to rest

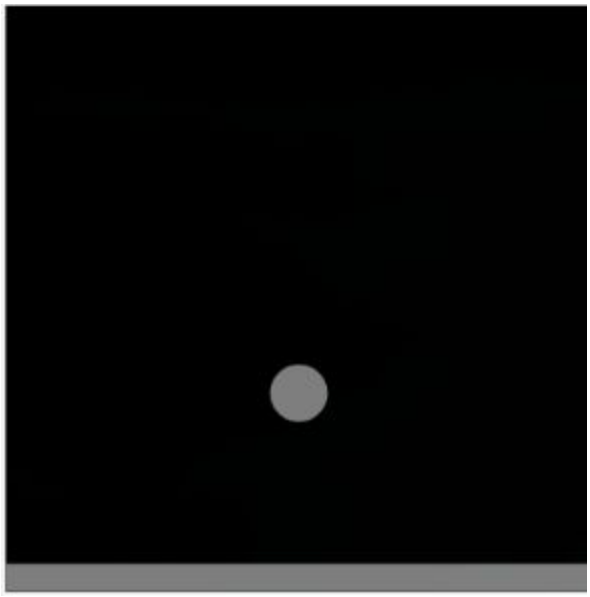
```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var ground;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    var ground_options ={
        isStatic: true
    }

    ground = Bodies.rectangle(200,390,200,20,ground_options);
    World.add(world,ground);

    console.log(ground);
}

function draw(){
    background(0);
    Engine.update(engine);
    rectMode(CENTER);
    rect(ground.position.x,ground.position.y,400,20);
}
```

Create a ball (ellipse) similar to the "object" created.
Use Bodies.circle to create a circular game object. Use ellipse to draw it. Change the ellipseMode to RADIUS

```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var ground,ball;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    var ground_options ={
        isStatic: true
    }

    ground = Bodies.rectangle(200,390,200,20,ground_options);
    World.add(world,ground);

    ball = Bodies.circle(200,100,20);
    World.add(world, ball);

    console.log(ground);
}

function draw(){
    background(0);
    Engine.update(engine);
    rectMode(CENTER);
    rect(ground.position.x,ground.position.y,400,20);

    ellipseMode(RADIUS);
    ellipse(ball.position.x, ball.position.y, 20,20);
}
```

Step 15: Add restitution and pass it while the ball object is created, to make it bounce like a tennis ball.

```
var ground, ball;

function setup(){
    var canvas = createCanvas(400,400);
    engine = Engine.create();
    world = engine.world;

    var ground_options ={
        isStatic: true
    }

    ground = Bodies.rectangle(200,390,200,20,ground_options);
    World.add(world,ground);

    var ball_options ={
        restitution: 1.0
    }

    ball = Bodies.circle(200,100,20, ball_options);
    World.add(world,ball);

    console.log(ground);
}

function draw(){
    background(0);
    Engine.update(engine);
    rectMode(CENTER);
    rect(ground.position.x,ground.position.y,400,20);

    ellipseMode(RADIUS);
    ellipse(ball.position.x, ball.position.y, 20, 20);
}
```



What's next?: Create our own Angry Birds Game