

## Physics Engine



### What is our GOAL for this MODULE?

The goal for this module is to gain the knowledge of the physics engine and its usage.

### What did we ACHIEVE in the class TODAY?

- 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

### Which CONCEPTS/ CODING BLOCKS did we cover today?

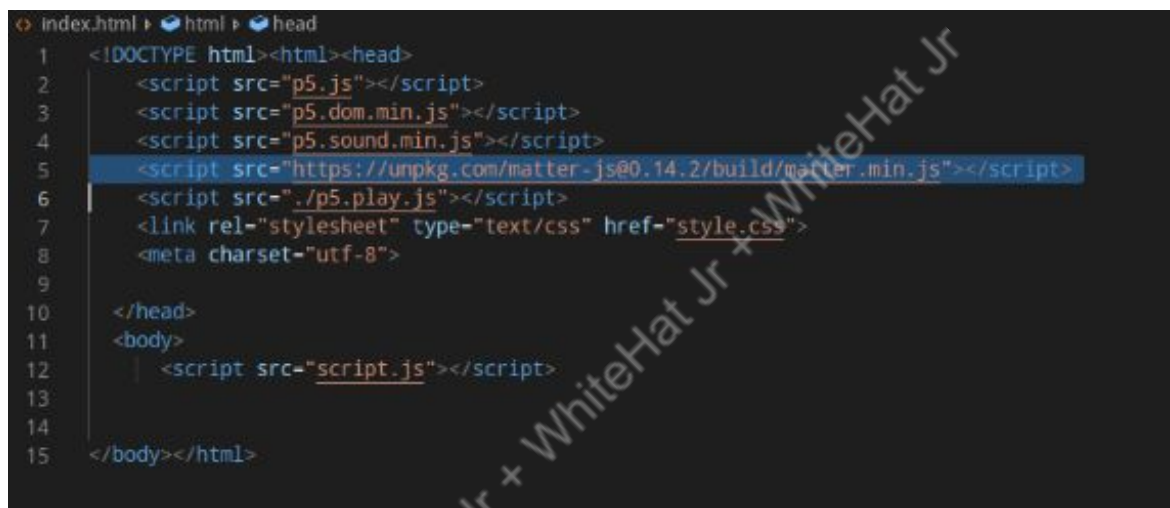
- The concept of creating a physics engine.
- Creating interactive objects which follow rules of physics.

### How did we DO the activities?

In our projects, we will be using matter.js as our physics engine.

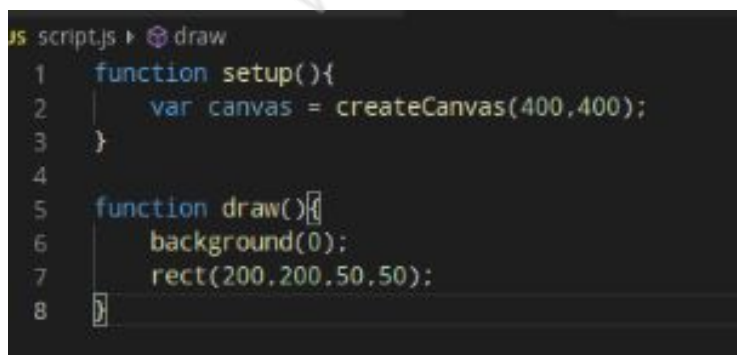
1. Get the boilerplate from the GitHub.
2. To use the matter.js library in our project, modify the index.html file to add the script tag for the library. Add the script tag with src as the link for the file:

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

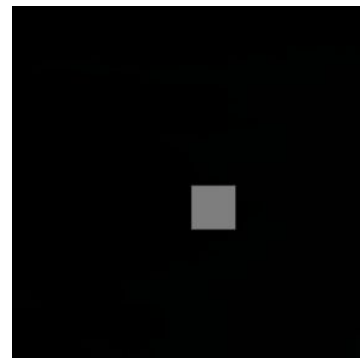


```
index.html ▶ html ▶ head
1 <!DOCTYPE html><html><head>
2   <script src="p5.js"></script>
3   <script src="p5.dom.min.js"></script>
4   <script src="p5.sound.min.js"></script>
5   <script src="https://unpkg.com/matter-js@0.14.2/build/matter.min.js"></script>
6   <script src="./p5.play.js"></script>
7   <link rel="stylesheet" type="text/css" href="style.css">
8   <meta charset="utf-8">
9
10  </head>
11  <body>
12    <script src="script.js"></script>
13
14
15 </body></html>
```

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

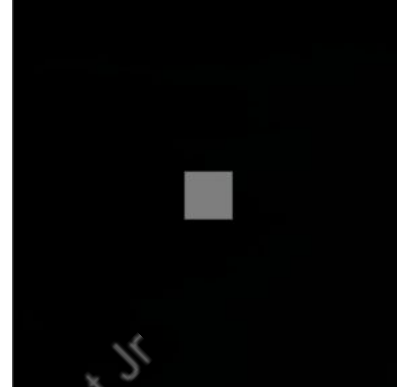


```
js script.js ▶ draw
1 function setup(){
2   var canvas = createCanvas(400,400);
3 }
4
5 function draw(){
6   background(0);
7   rect(200,200,50,50);
8 }
```



- 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  }
```



- 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 }
```

6. 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 }
```

7. 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 }
```

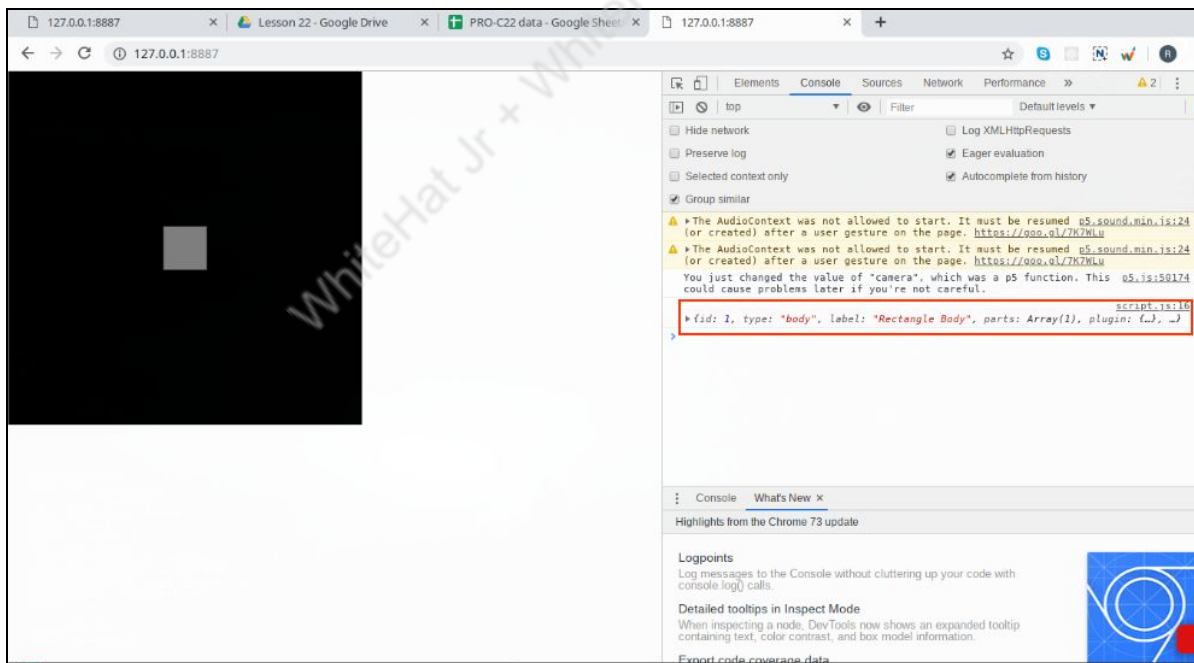
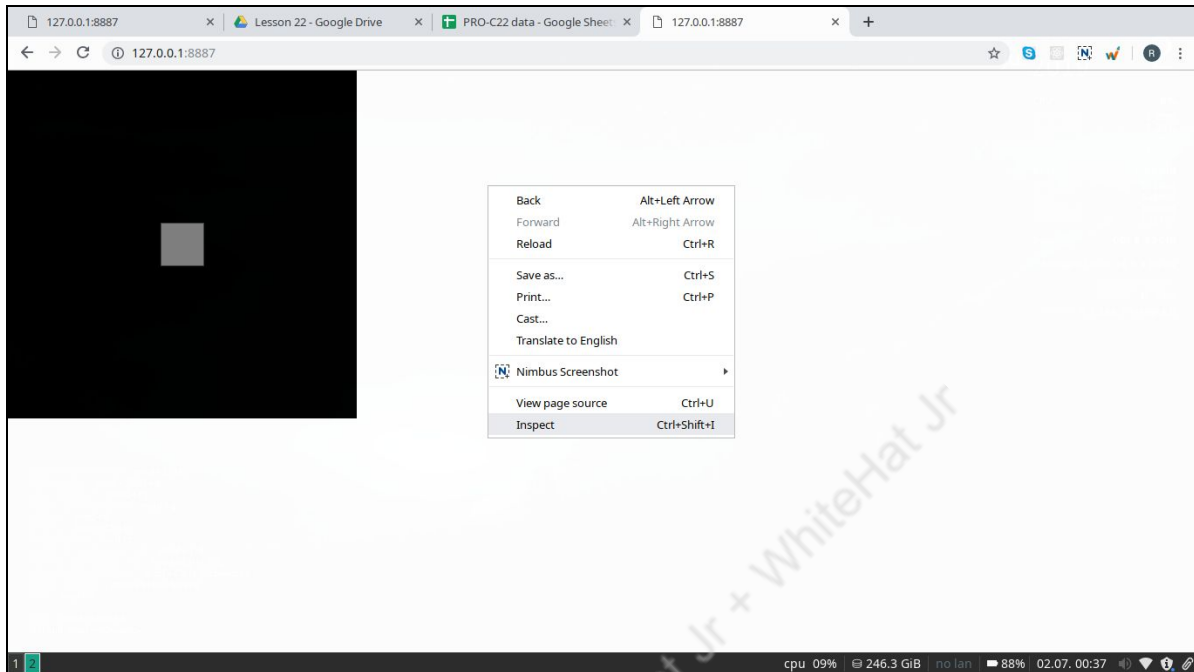
8. Write the code to add the body to the world.

```
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     World.add(world,object);
15 }
16
17 function draw(){
18     background(0);
19     rectMode(CENTER);
20     rect(200,200,50,50);
21 }
```

9. 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.

```
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     World.add(world,object);
15
16     console.log(object);
17 }
18
19 function draw(){
20     background(0);
21     rectMode(CENTER);
22     rect(200,200,50,50);
23 }
```

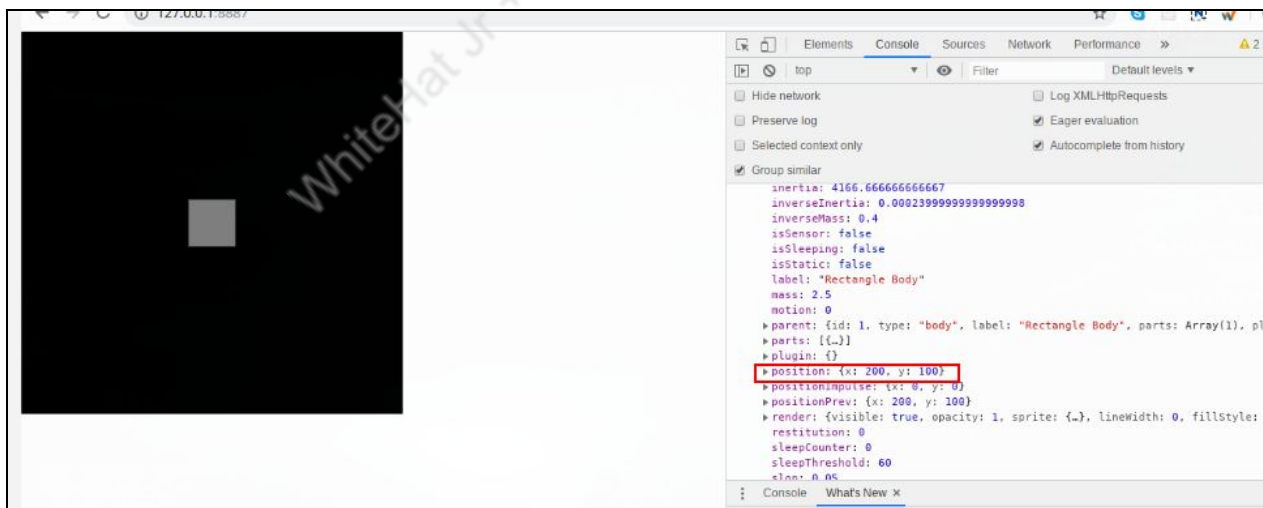
10. Right-click inside the browser and see the console output by pressing on inspect.



11. Print this object type on the console. You just need to write **console.log(object.type)**

```
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     World.add(world,object);
15
16     console.log(object);
17     console.log(object.type);
18 }
19
20 function draw(){
21     background(0);
22     rectMode(CENTER);
23     rect(200,200,50,50);
24 }
```

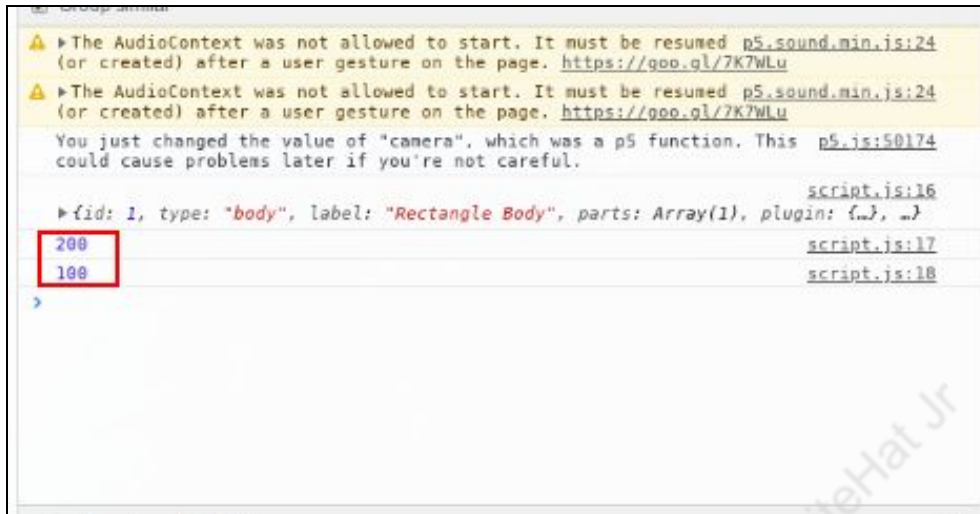
12. Click on the arrow to the left of the object we have created, you will see it has many attributes. It also has an attribute called position.





13. Print the x and y of this object:

`console.log(object.position.x)` and `console.log(object.position.y)`



```
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    World.add(world,object);
15
16    console.log(object);
17    console.log(object.position.x);
18    console.log(object.position.y);
19  }
20
21  function draw(){
22    background(0);
23    rectMode(CENTER);
24    rect(200,200,50,50);
25  }
```



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

```
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    World.add(world,object);
15
16    console.log(object);
17  }
18
19  function draw(){
20    background(0);
21    Engine.update(engine);
22    rectMode(CENTER);
23    rect(object.position.x,object.position.y,50,50);
24  }
```

15. For a static rectangle write this code:

```
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    var object_options = {
14      isStatic: true
15    }
16
17    object = Bodies.rectangle(200,100,50,50,object_options);
18    World.add(world,object);
19
20    console.log(object);
21  }
22
23  function draw(){
24    background(0);
25    Engine.update(engine);
26    rectMode(CENTER);
27    rect(object.position.x,object.position.y,50,50);
28  }
```

16. Create a ball which bounces on the ground like a tennis ball and then comes to rest.

```

1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4
5  var engine, world;
6  var ground;
7
8  function setup(){
9    var canvas = createCanvas(400,400);
10    engine = Engine.create();
11    world = engine.world;
12
13    var ground_options = {
14      isStatic: true
15    }
16
17    ground = Bodies.rectangle(200,390,200,20,ground_options);
18    World.add(world,ground);
19
20    console.log(ground);
21  }
22
23  function draw(){
24    background(0);
25    Engine.update(engine);
26    rectMode(CENTER);
27    rect(ground.position.x,ground.position.y,400,20);
28  }

```

17. Create a ball (ellipse) similar to the "object" created.

- Use **Bodies.circle** to create a circular game object.
- Use an ellipse to draw it.
- Change the **ellipseMode** to **RADIUS**

```

16
17    ground = Bodies.rectangle(200,390,200,20,ground_options);
18    World.add(world,ground);
19
20    ball = Bodies.circle(200,100,20);
21    World.add(world, ball);
22
23    console.log(ground);
24  }
25
26  function draw(){
27    background(0);
28    Engine.update(engine);
29    rectMode(CENTER);
30    rect(ground.position.x,ground.position.y,400,20);
31
32    ellipseMode(RADIUS);
33    ellipse(ball.position.x, ball.position.y, 20,20);
34  }

```

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

```
5  var engine, world;  
6  var ground, ball;  
7  
8  function setup(){  
9    var canvas = createCanvas(400,400);  
10   engine = Engine.create();  
11   world = engine.world;  
12  
13   var ground_options = {  
14     isStatic: true  
15   }  
16  
17   ground = Bodies.rectangle(200,390,200,20,ground_options);  
18   World.add(world,ground);  
19  
20   var ball_options = {  
21     restitution: 1.0  
22   }  
23  
24   ball = Bodies.circle(200,100,20, ball_options);  
25   World.add(world,ball);  
26  
27   console.log(ground);  
28 }  
29  
30 function draw(){  
31   background(0);  
32   Engine.update(engine);  
33   rectMode(CENTER);  
34   rect(ground.position.x,ground.position.y,400,20);  
35  
36   ellipseMode(RADIUS);  
37   ellipse(ball.position.x, ball.position.y, 20, 20);  
38 }
```



### What's next?

We will create our own Angry Birds game

**Extend your knowledge:**

- Learn about different kinds of forces that can be used in a game using physics engine

<https://p5js.org/examples/simulate-forces.html>

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr