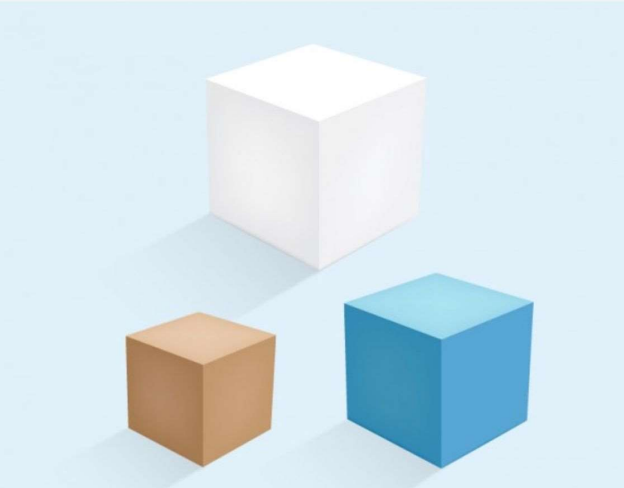
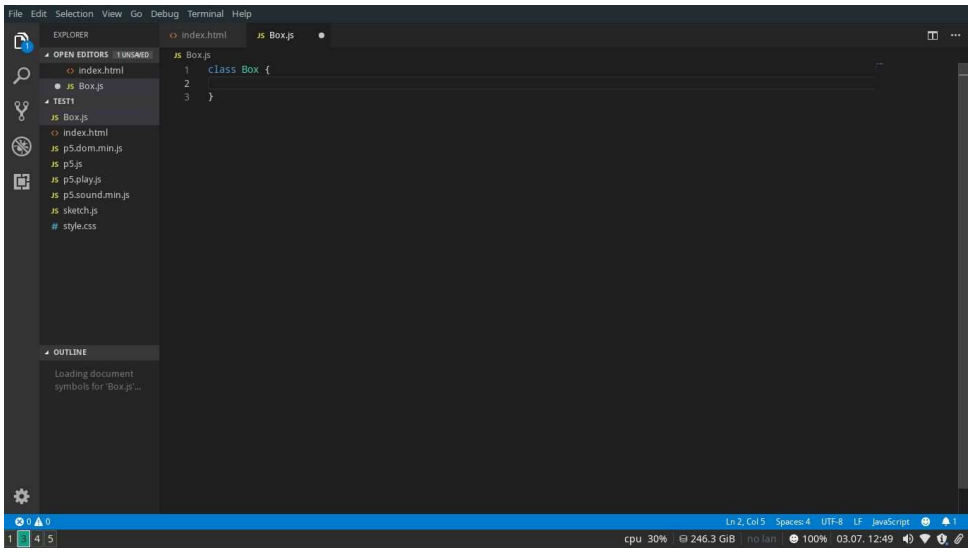
Class



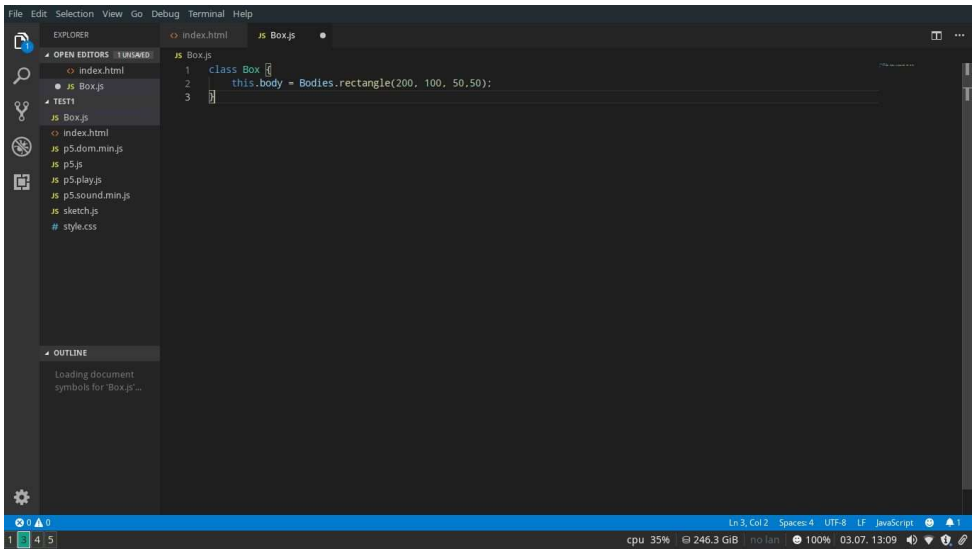
What we did:

- Create a Box class which creates a template for new objects to be made using the physics engine.
- Create two box objects using the Box class template.
- Tune the physics engine for properties like density, friction etc. for these objects so that they topple over each other
- Display the rectangle so that it can draw with its orientation.

How we did it: We need many boxes as obstacles in which our enemy Pigs will be hiding.
Step 1: Create a blueprint for our box. A blueprint for an object is called a Class.
Create a new file in the same folder called Box.js.



Step 2: Put the x,y, width and height values here as 200,100, 50,50
**Note: Put all these inside constructor () { }



Step 3: Add an option here, which will finetune the physics engine for the object

Add this object to the world

```
class Box {
    var options = {
        restitution: 1;
    }
    this.body = Bodies.rectangle(200, 100, 50,50, options);
    World.add(world, this.body);
}
```

Display this object— write a display() function to do that.

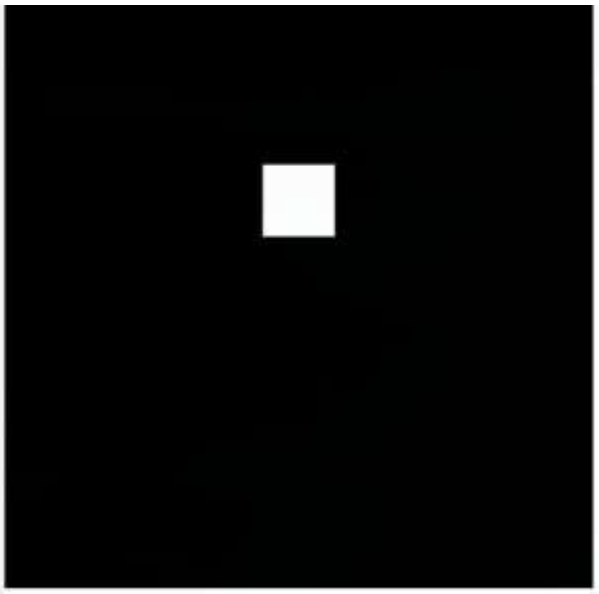
```
class Box {
    var options = {
        restitution: 1;
    }
    this.body = Bodies.rectangle(200, 100, 50,50, options);
    World.add(world, this.body);

    display(){
    var pos =this.body.position;
    rectMode(CENTER);
    fill(255);
    rect(pos.x, pos.y, this.width, this.height);
}
```

Step 4: In the sketch.js file, remove all the statements associated with creating the bodies.

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

var engine, world;
var box1;

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


}

function draw(){
    background(0);
    Engine.update(engine);

}
```

Step 5: Create a new object and display it with just two statements.

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

var engine, world;
var box1;

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

    box1 = new Box();
}

function draw(){
    background(0);
    Engine.update(engine);

    box1.display();
}
```



Step 6: Write code in the Box class to show how a constructor of a class can take arguments: Tell the computer where to draw the rectangle and of what dimensions, by passing the x,y, width and height to the constructor.

```
class Box {
    constructor(x,y,width,height) {
      var options = {
          restitution:0.8
      }
      this.body = Bodies.rectangle(x,y,width,height);
      World.add(world, this.body);
    }
    display(){
      var pos =this.body.position;
      rectMode(CENTER);
      fill(255);
      rect(pos.x, pos.y, this.width, this.height);
    }
};
```

Step 7: Create the second box object using the Box class.

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

var engine, world;
var box1;

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

    box1 = new Box(200,100,50,50);
    box2 = new Box(100,50,50,100);
}

function draw(){
    background(0);
    Engine.update(engine);

    box1.display();
    box2.display();
}
```

Create a Ground class blueprint and then create a ground object using it

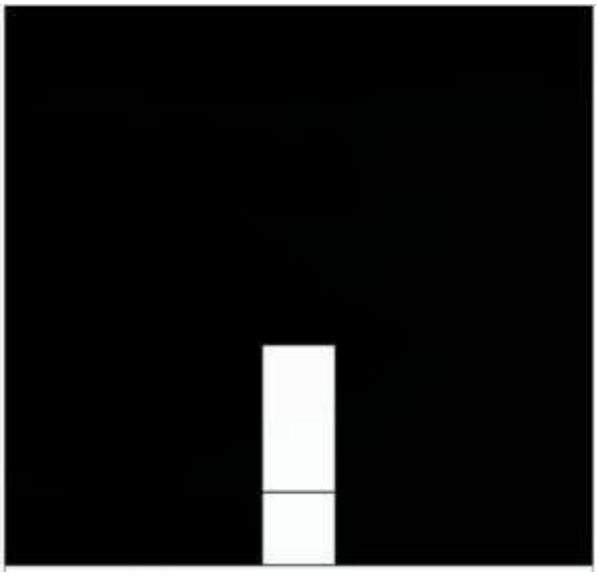
```
class Ground {
    constructor(x,y,width,height) {
      var options = {
          isStatic: true
      }
      this.body = Bodies.rectangle(x,y,width,height,options);
      this.width = width;
      this.height = height;
      World.add(world, this.body);
    }
    display(){
      var pos =this.body.position;
      rectMode(CENTER);
      fill(255);
      rect(pos.x, pos.y, this.width, this.height);
    }
  };
```

```
<!DOCTYPE html><html><head>
    <script src="p5.js"></script>
    <script src="p5.dom.min.js"></script>
    <script src="p5.sound.min.js"></script>
    <script src="https://unpkg.com/matter-js@0.14.2/build/matter.min.js"></script>
    <script src="p5.play.js"></script>
    <script src="Box.js"></script>
    <script src="Ground.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">

  </head>
  <body>
      <script src="sketch.js"></script>


</body></html>
```

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

var engine, world;
var box1;

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

    box1 = new Box(200,300,50,50);
    box2 = new Box(200,100,50,100);
    ground = new Ground(200,390,400,20)
}

function draw(){
    background(0);
    Engine.update(engine);

    box1.display();
    box2.display();
    ground.display();
}
```

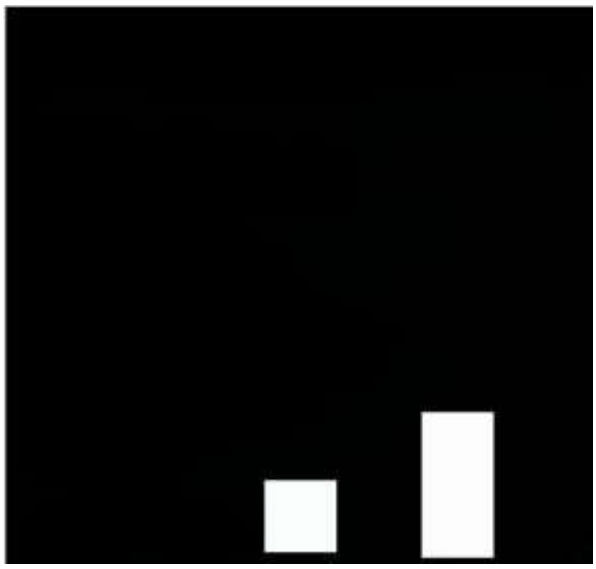
Step 8: Stack the two box objects on top of each other: Allow the box2 to fall on the top of box 1.

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

var engine, world;
var box1;

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

    box1 = new Box(200,300,50,50);
    box2 = new Box(200,100,50,100);
    ground = new Ground(200,390,400,20)
}

function draw(){
    background(0);
    Engine.update(engine);

    box1.display();
    box2.display();
    ground.display();
}
```



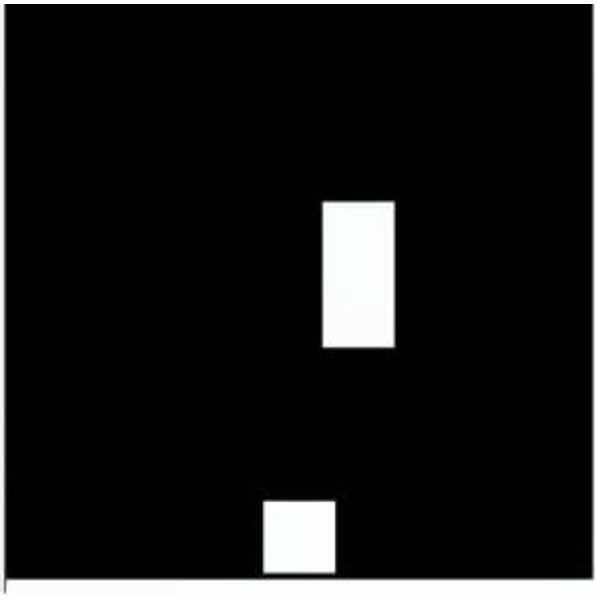
Shift box 2 slightly to the right

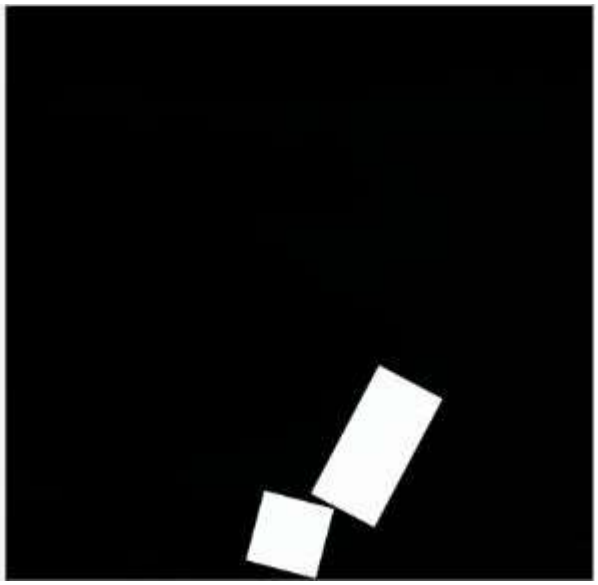
```
const Engine = Matter.Engine;
const World= Matter.World;
const Bodies = Matter.Bodies;

var engine, world;
var box1;

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

    box1 = new Box(200,300,50,50);
    box2 = new Box(240,100,50,100);
    ground = new Ground(200,390,400,20)
}

function draw(){
    background(0);
    Engine.update(engine);

    box1.display();
    box2.display();
    ground.display();
}
```



Step 9: Store the new translation and rotation setting and then revert back to the old setting when the object is drawn.
This is done using push() and pop().
push() -> captures the new setting
pop() -> reverts back to the old setting
translate() -> to change the 0 of the axis to a given x and y position.

```
class Box {
  constructor(x, y, width, height) {
    var options = {
        'restitution':0.8
    }
    this.body = Bodies.rectangle(x, y, width, height, options);
    this.width = width;
    this.height = height;

    world.add(world, this.body);
  }
  display(){
    var pos =this.body.position;
    var angle = this.body.angle;
    push();
    translate(pos.x, pos.y);
    rotate(angle);
    rectMode(CENTER);
    fill(255);
    rect(0, 0, this.width, this.height);
    pop();
  }
};
```





You can play around with more properties of objects like restitution, density, friction etc.

What's next?: Using what we learned today to create the stack of obstacles for the pig in the Angry Birds Game.