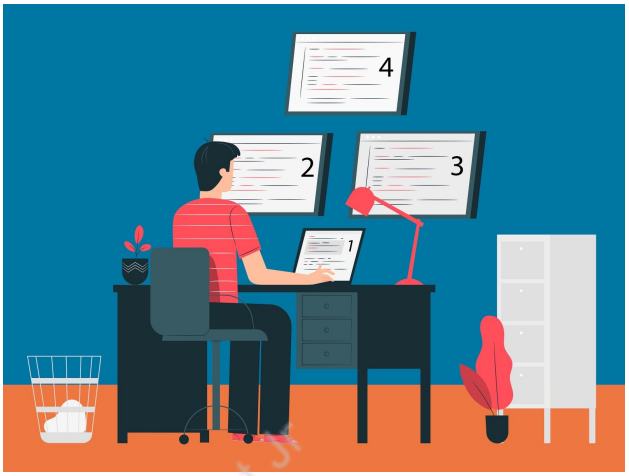
PRO-C25

WhiteHat Jr
Live Online Coding for Kids

Inheritance



What is our GOAL for this MODULE?

The goal of this module is to learn the concept of inheritance to create objects with properties of parent class.

What did we ACHIEVE in the class TODAY?

- Added an image property to one of the classes and used it to make one of the game objects animated.
- Introduced the concept of inheritance and how a subclass can inherit the properties and functions of a parent class.
- Wrote a subclass which extends the properties and functions of a parent class.
- We revised the following as well:
    - A class is a blueprint of an object.
    - A class contains defined properties(like width, height) and functions (like display()).
    - A class is used to create one or more objects having the same properties and functions as defined in the class.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- The concept of inheritance

How did we DO the activities?

1. The Bird class blueprint had properties like body, width, and height. We added an additional property to it called image and loaded the bird image.

```
class Bird {
    constructor(x, y) {
      var options = {
        'density':1.5,
        'friction': 1.0,
        'restitution':0.5
      };
      this.body = Bodies.rectangle(x, y, 50, 50, options);
      this.width = 50;
      this.height = 50;
      this.image = loadImage("sprites/bird.png");
      World.add(world, this.body);
    };
    display(){
      var pos = this.body.position;
      pos.x = mouseX;
      pos.y = mouseY;
      var angle = this.body.angle;

      push();
      translate(pos.x, pos.y);
      rotate(angle);
      strokeWeight(3);
      stroke('blue')
      fill('red')
      rectMode(CENTER)
      rect(0, 0, this.width, this.height);
      pop();
    };
  };
```

2. We wanted an image instead of a rectangle. We used the image() instruction instead of the rect() instruction.
    - The first argument was for the image.
    - The second and third arguments were for the position. We changed the position to where we wanted and used 0, 0.
    - The fourth and fifth were for the width and height and we used it from the

```
Bird.js ▸ Bird ▸ display
class Bird {
    constructor(x, y) {
      var options = {
        'density':1.5,
        'friction': 1.0,
        'restitution':0.5
      };
      this.body = Bodies.rectangle(x, y, 50, 50, options);
      this.width = 50;
      this.height = 50;
      this.image = loadImage("sprites/bird.png");
      World.add(world, this.body);
    };
    display(){
      var pos = this.body.position;
      pos.x = mouseX;
      pos.y = mouseY;
      var angle = this.body.angle;

      push();
      translate(pos.x, pos.y);
      rotate(angle);
      imageMode(CENTER)
      image(this.image, 0, 0, this.width, this.height);
      pop();
    };
  };
```

property of the class (defined in the constructor).

In programming language, we have a concept of a Parent / Base class and Children / Sub classes. Children/Sub classes that are created using Parent / Base class inherit all the properties and functions from the parent class.

3. We wrote the code to create a BaseClass. Our Base object can have all the properties and functions which we had in the Bird class.

```
BaseClass.js ▸ BaseClass ▸ constructor
class BaseClass{
    constructor(x, y, width, height, angle) {
        var options = {
            'restitution':0.8,
            'friction':1.0,
            'density':1.0
        }
        this.body = Bodies.rectangle(x, y, width, height, options);
        this.width = width;
        this.height = height;
        this.image = loadImage("sprites/base.png");
        World.add(world, this.body);
    }
    display(){
        var angle = this.body.angle;
        push();
        translate(this.body.position.x, this.body.position.y);
        rotate(angle);
        imageMode(CENTER);
        image(this.image, 0, 0, this.width, this.height);
        pop();
    }
}
```

4. Included the src of the BaseClass in the index.html file.

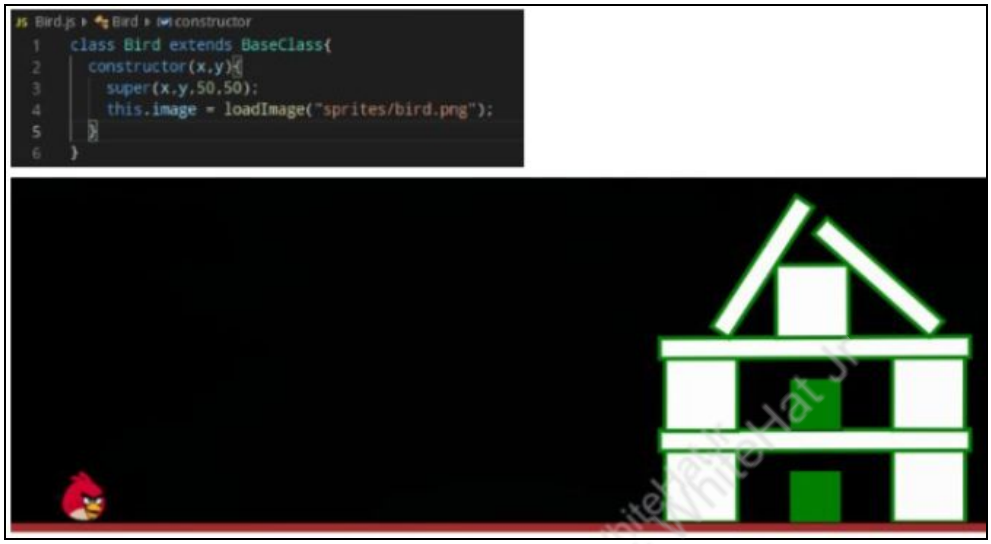
```
index.html > html > head > script
<!DOCTYPE html>
<html>
<head>
    <script src="p5.min.js"></script>
    <script src="p5.dom.min.js"></script>
    <script src="p5.sound.min.js"></script>
    <script src="matter.js"></script>
    <script src="BaseClass.js"></script>
    <script src="Ground.js"></script>
    <script src="Box.js"></script>
    <script src="Pig.js"></script>
    <script src="Log.js"></script>
    <script src="Bird.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">
</head>
<body>
  <script src="sketch.js"></script>
</body>
</html>
```

Box and the Pig classes were very similar to the BaseClass. These classes became the child class for this parent BaseClass and inherited all the properties and functions. All the properties and functions of a parent class were present in the child class.

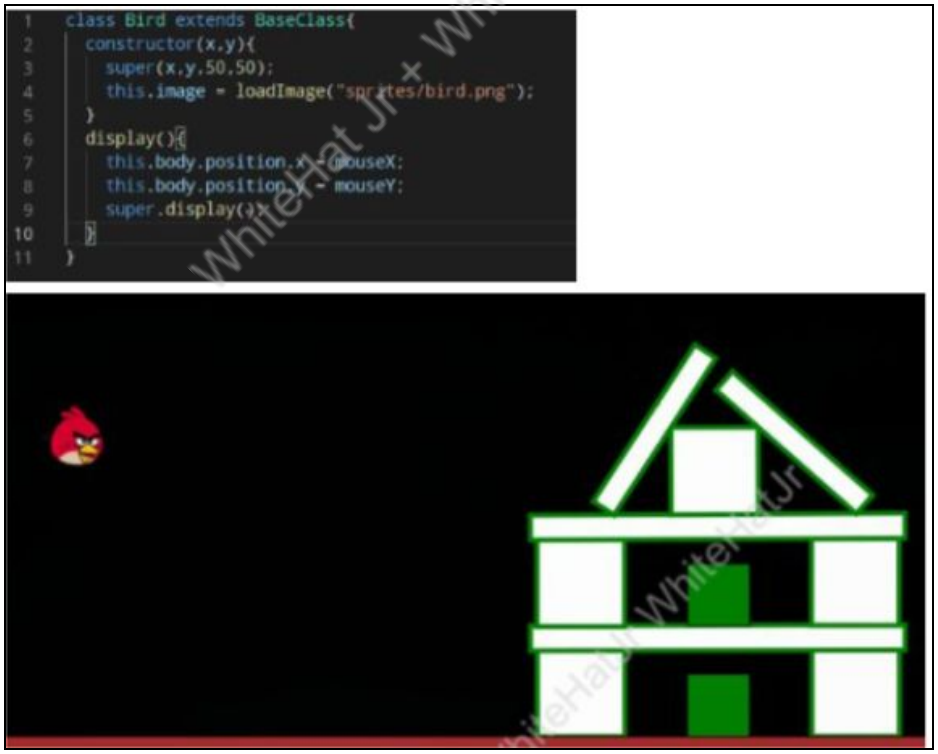
5. We created a child Bird Class which inherited all the properties and functions of our BaseClass.

```
Bird.js > Bird > constructor
class Bird extends BaseClass{
  constructor(x,y){
    super(x,y,50,50);
    this.image = loadImage("sprites/bird.png");
  }
}
```

6. Finally, we added the bird image to the bird class constructor as well. You could do it inside the constructor and overwrite any of the properties of the parent class inside the child class and change it.



7. To override the display function of the base class by writing code for it, we used super.display() to refer to the parent class display function.



8. You added images to all the other objects in the game by modifying their class blueprint.

```
class Box extends BaseClass {
  constructor(x, y, width, height){
    super(x,y,width,height);
    this.image = loadImage("sprites/wood1.png");
  }

};
```

```
class Pig extends BaseClass {
  constructor(x, y){
    super(x,y,50,50);
    this.image = loadImage("sprites/enemy.jpg");
  }

};
```

```
class Log extends BaseClass{
  constructor(x,y,height,angle){
    super(x,y,20,height,angle);
    this.image = loadImage("sprites/wood2.png");
    Matter.Body.setAngle(this.body, angle);
  }
}
```

9. You added the background image in the sketch file.

sketch.js ▸ preload

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

var engine, world;
var box1, pig1;
var backgroundImg;

function preload(){
    backgroundImg = loadImage("sprites/bg.png");
}

function setup(){
    var canvas = createCanvas(1200,400);
    engine = Engine.create();
    world = engine.world;


    ground = new Ground(600,height,1200,20)

    box1 = new Box(700,320,70,70);
    box2 = new Box(920,320,70,70);
    pig1 = new Pig(810, 350);
    log1 = new Log(810,260,300, PI/2);

    box3 = new Box(700,240,70,70);
    box4 = new Box(920,240,70,70);
    pig3 = new Pig(810, 220);

    log3 =  new Log(810,180,300, PI/2);

    box5 = new Box(810,160,70,70);
    log4 = new Log(760,120,150, PI/7);
    log5 = new Log(870,120,150, -PI/7);
```

sketch.js ▸ draw

```
    box5 = new Box(810,160,70,70);
    log4 = new Log(760,120,150, PI/7);
    log5 = new Log(870,120,150, -PI/7);

    bird = new Bird(100,100);

}

function draw(){
    background(backgroundImg);
    Engine.update(engine);
    console.log(box2.body.position.x);
    console.log(box2.body.position.y);
    console.log(box2.body.angle);
    box1.display();
    box2.display();
    ground.display();
    pig1.display();
    log1.display();

    box3.display();
    box4.display();
    pig3.display();
    log3.display();

    box5.display();
    log4.display();
    log5.display();

    bird.display();
}
```

What's NEXT?

In the next class, you will learn about Git and GitHub.

EXTEND YOUR KNOWLEDGE:

Explore more examples of inheritance here:
https://p5js.org/examples/objects-inheritance.html