


Topic	FORCE APPLICATION ON A BODY	
Class Description	The student will learn to apply force to a body. They will also use OOP concepts to create a class for the ground body to create multiple objects out of the ground class.	
Class	C21	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Create the Ground class. • Create functions to apply force on the body. • Attach a function to the button on the canvas. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	Warm-Up - Slide show option Teacher-Led Activity Student-Led Activity Wrap-Up - Slide show option	5 Mins 15 Mins 20 Mins 5 Mins
WARM-UP SESSION - 5 mins		
<div>  </div> <p>Teacher starts slideshow from slides 1 to 14 Refer to speaker notes and follow the instructions on each slide.</p>		
Activity details		Solution/Guidelines

*Hey <student's name>. How are you? It's great to see you!
Are you excited to learn something new today?*

Run the presentation from slide 1 to slide 3.

The following are the warm-up session deliverables:

- Greet the student.
- Revision of previous class activity.
- Quizzes

ESR: Hi, thanks, yes I am excited about it!

Click on the slide show tab and present the slides

QnA Session

Question

Select the correct block to set the static property of the rotator to be true.

- A. isStatic=true
- B. isStatic true
- C. isStatic:false
- D. isStatic:true


Answer

D

Select the correct block of code to create the rotator1 body.



D

A.	<pre>// rotator1 = rectangle(250,200,150,20, // rotator_options); // World.add(world,rotator1);</pre>	
B.	<pre>// rotator1 = Bodies.rectangle(); // World.add(world,rotator1);</pre>	
C.	<pre>// rotator1 = Bodies.circle(250,200,150,20); // World.add(world,rotator1);</pre>	
D.	<pre>// rotator1 = Bodies.rectangle(250,200,150,20, // rotator_options); // World.add(world,rotator1);</pre>	
<div>Continue the warm-up session</div>		
<div>Activity details</div>		<div>Solution/Guidelines</div>
<p>Run the presentation from slide 4 to slide 13 to set the problem statement.</p> <p>The following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> Appreciate the student on his performance in the quizzes. Explain how the application of Force changes a scenario. 		<p>Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in students.</p>
<div>Teacher ends slideshow</div> 		
<div>TEACHER-LED ACTIVITY - 15 mins</div>		
<div>Teacher Initiates Screen Share</div>		
<div> CHALLENGE <ul style="list-style-type: none"> Create the Ground class. Create functions to apply force on the body. </div>		

Teacher Action	Student Action
<p>In the last class, we learned to create physics bodies, and we have also changed the properties to see how they affect a certain body.</p>	<p><i>The teacher downloads the template code from Teacher Activity 1 and runs it in the VS Code Editor.</i></p>
<p>IN CASE THE STUDENT HAS MOVED FROM V2 MAKE SURE TO REVISE THESE CONCEPTS: (Covered in VA too)</p> <p>Remember in the first class, we had studied about classes and objects? In the real world, before any object - like a pen - is manufactured, first a design/blueprint for the pen is made. Using this design/blueprint we make new pens.</p> <p>Programmers do the same when programming. They design the blueprint for an object - they call it class. Using this class, programmers can make any number of new objects. We will be doing this in today's class.</p>	
<p>In this class, we are going to create a class for the ground body to create multiple objects like ground, ball, wall, etc.</p> <p>First, create a new file named ground.js and add this to our index.html as shown below:</p> <pre data-bbox="162 1176 958 1785"><!DOCTYPE html><html><head> <script src="p5.min.js"></script> <script src="p5.dom.min.js"></script> <script src="p5.sound.min.js"></script> <link rel="stylesheet" type="text/css" href="style.css"> <meta charset="utf-8"> </head> <body> <script src="matter.min.js"></script> <script src="ground.js"></script> <script src="sketch.js"></script> </body></html></pre> <p><i>The teacher opens the ground.js file</i></p>	

In this file, we will create the **Ground** class using OOP concepts.

In the last class, we created a rectangle body in **sketch.js** to display the ground. Now, imagine we need four similar objects on our canvas. Creating each object in **sketch.js** will make our code lengthy and repetitive. We will therefore create objects using OOP.

We have learned how to create a class in Lesson 16.

Note: In case if the student has moved from V2 it was covered in Lesson 1; brief them a little about the class as covered in VA.

In the **Ground** class, we will add all the parameters which are needed to create a physics body, we will also create functions or methods for this **Ground** class.

Create the class by using the **class** keyword and name of the class, as **Ground**. Remember, the first letter of the class name is always in uppercase.

In this class first, we need to create a **constructor()**. This function is executed when we create the object of the class. In the constructor function, we will pass the arguments which are to be given in **sketch.js**. These are **x** and **y positions** and the **width** and **height** of the body.

When we will create the object of the **Ground** class, we can choose the position and the dimensions of the body.

In the **constructor()** function we need to specify the options/properties for the physics body.

We want the ground to remain stationary, do you recall which property we can use?

ESR: We can add property **isStatic: true**

```
class Ground
{
  constructor(x, y, w, h)
  {
    let options = {
      isStatic:true
    };

    this.body = Bodies.rectangle(x, y, w, h, options);
    this.w = w;
    this.h = h;
    World.add(world, this.body);
  }
}
```

Here we created a body for the ground using the rectangle shape and assign this to the **this.body**.
 All the variables in the class are referred using the **this** keyword.

After this, we will add this body to the world.

Now we will create a function to display the ground in the canvas.

In the **Ground** class create a **show()** function, functions inside the class are called **methods**, and we don't need to add the function keyword in front of them.

In the **show()** function, we create a variable **pos**, in this we will store the position of the ground body.

If we want to give certain properties for one shape, such as **color**, **stroke**, and **strokeWeight** we can encapsulate these in-between **push()** and **pop()** statements

The properties added here do not affect other shapes on the canvas.

The shape of the ground is a rectangle, so we will create a rectangle using the **rect()** function and pass the position and dimension of the physics body to it.

```
show() {  
  var pos = this.body.position;  
  push();  
  rectMode(CENTER);  
  stroke(255);  
  fill(127);  
  rect(pos.x, pos.y, this.w, this.h);  
  pop();  
}
```

Our class code is now complete.

Now head to the **sketch.js** file and create objects for the **Ground** class.

We will make 4 objects because we need 4 walls.

First declare 4 variables as:

```
var ground;  
var right;  
var left;  
var top_wall;
```

In the **setup** function, we create the objects from the

Ground class using the **new** keyword and assign them to the respective variable.

We want to keep the ground near the bottom of the canvas so its y position can be set at **390** and similarly for all the other walls we have to set the position.

But we won't be able to see any of the walls now. We need to call the **show** function for each and every **wall** object, then only it will be visible.

We will do that in the **draw()** function.

```
var ground;
var left;
var right;
var top_wall;

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

  ground =new Ground(200,390,400,20);
  right = new Ground(390,200,20,400);
  left = new Ground(10,200,20,400);
  top_wall = new Ground(200,10,400,20);

  rectMode(CENTER);
  ellipseMode(RADIUS);
}
```

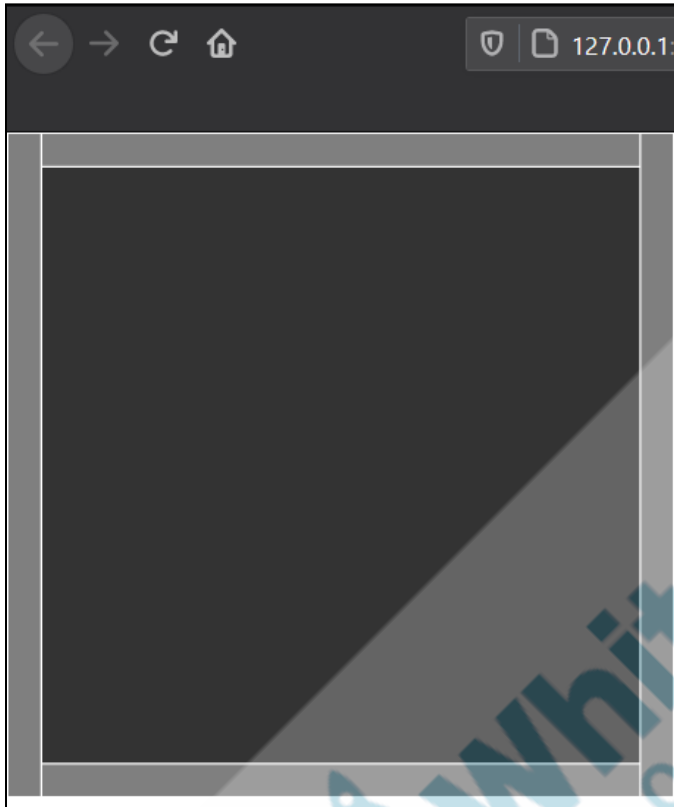
Let's now call the **show()** inside **draw()**.


```
function draw()  
{  
  background(51);  
  ground.show();  
  top_wall.show();  
  left.show();  
  right.show();  
  Engine.update(engine);  
}
```

Now we can run the code to see the output.

There will be 4 walls on the canvas.

Output:



We have four walls on the canvas.

In the next step let's create the body for the ball and apply force on it to bounce it in different directions.

Now it's your turn, please share your screen with me.




Teacher starts slideshow :Slide 15 & Slide 16

Run the presentation slide to set the student activity context.

- Function to apply force on the ball.

- Create a button to allow the player to have control over the ball.

<div>  Teacher ends slideshow </div>	
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY - 20 mins	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 	
<p style="text-align: center;"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Apply force on the body. • Create buttons on the canvas and attach functions to it. 	
Teacher Action	Student Action
<p>Just like in the real world, if you want to move a body (any object) you have to apply some kind of force to it.</p> <p>Either you push the object or you pull it.</p> <p>In the same way, if we want to move any object in the physics engine we can apply force on the object.</p> <p>We can apply force in a very specific direction such as x-direction then the body will only move in the horizontal direction.</p> <p>Similarly, a force applied in the Y direction will move the body in the vertical direction.</p> <p>We can join the 2 forces as well which will make the object move in both directions. It will be similar to how we throw a ball and make it have a parabolic trajectory.</p>	<p><i>The student downloads the code for Student Activity 1 and opens it in the VS Code editor.</i></p>

First, let's create the ball body on the canvas. Then we can apply the force to it.

We have created the ball body in the last class as well

Can you tell me how can we do that?
Great!

First, let's make a variable as **var ball** in **sketch.js**;

After which, in the **setup()** create a **circle()** body using the physics engine and assign it to **var ball**. The **restitution** property to set at **0.5** in **var ball_options**.

Remember to add the **ball** object in the **World** as shown in the following code snippet:

ESR: Using **body.circle()** function

```
var ball;

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

  ground = new Ground(200,390,400,20);
  right = new Ground(390,200,20,400);
  left = new Ground(10,200,20,400);
  top_wall = new Ground(200,10,400,20);

  var ball_options = {
    restitution: 0.95
  }

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

  rectMode(CENTER);
  ellipseMode(RADIUS);
}
```

In the **draw()** function we can display the ball by passing its position in the **ellipse()** function.

```
function draw()
{
  background(51);

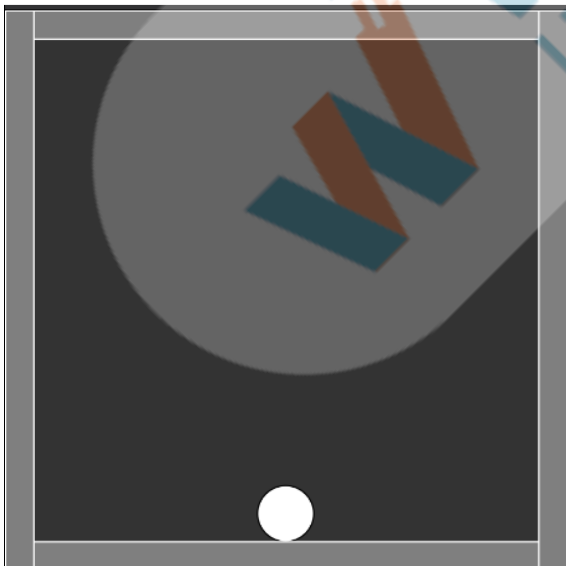
  ellipse(ball.position.x,ball.position.y,20);

  ground.show();
  top_wall.show();
  left.show();
  right.show();
  Engine.update(engine);
}
```

We can now run the code to see the output.

The student runs the code and observes the output.

Output:



As you can see, we now have a ball in our output that

obeys the laws of physics we have set it with.

Now we move onto the next step which is to apply force on the ball.

For this we will create a function and add this function to a button, when the user clicks the button, force will be applied to the ball.

Let's create a function named **hForce()** as this function will apply force in the horizontal direction.

```
function hForce()
{

}
```

Matter.js offers us the function to apply the force on the body, where we can choose the body and points on which we want to apply the force on, direction, and the value of force.

This function is called **Matter.Body.applyForce(ball,{x:0,y:0},{x:0,y:0});**

Here the first argument is the **body**, the second is the **x & y** positions which is the **initial amount of force applied on the body** and the third is the **amount and direction of the force** in **x** and **y**-direction.

If we apply **+ve** force in the **x**-direction, that will move the ball in the **right** direction and similarly, **-ve** force in the **left** direction.

Similarly, **+ve** value of **y** will move the ball in the **downward** direction and **-ve** value will move in the

upward direction.

```
function hForce()
{
  Matter.Body.applyForce(ball,{x:0,y:0},{x:0.05,y:0});
}
```

We will also create the function to apply force in the vertical direction.

```
function vForce()
{
  Matter.Body.applyForce(ball,{x:0,y:0},{x:0,y:-0.05});
}
```

Now that we have added the component of force to the ball, we will now create buttons on the canvas and attach these functions to the buttons.

The buttons we are creating are called **image** buttons, so arrow images we saw in the visual will be displayed on the canvas, and they will work like buttons.

First, you create two variables as **var btn1, btn2;**

Next, in the **setup()** function, we will create buttons. Button is created using the **createImg()** function, in the arguments. Then we will set the position and the size of the button.

After that, we will attach the function with a button using the **mouseClicked()** function. We will pass the function name as an argument in this function.

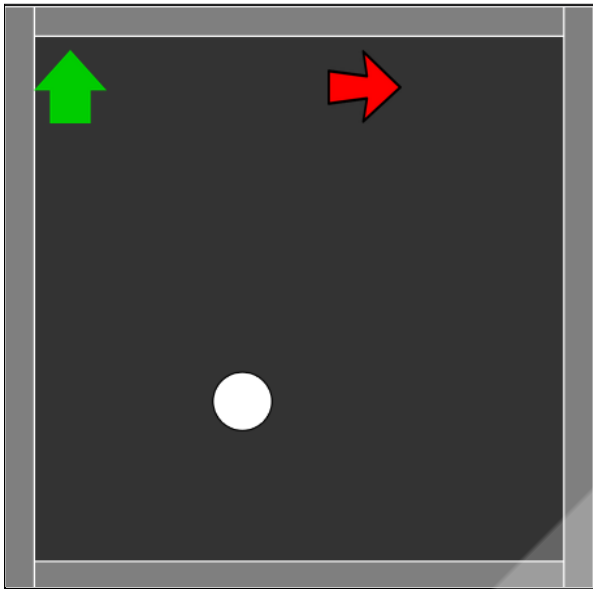
```
var btn1;  
var btn2;  
  
function setup() {  
  createCanvas(400,400);  
  engine = Engine.create();  
  world = engine.world;  
  
  btn1 = createImg('right.png');  
  btn1.position(220,30);  
  btn1.size(50,50);  
  btn1.mouseClicked(hForce);  
  
  btn2 = createImg('up.png');  
  btn2.position(20,30);  
  btn2.size(50,50);  
  btn2.mouseClicked(vForce);  
}
```

Now we can run the program to see that our buttons are working.

When we press the right arrow button, it will push the ball in the right direction. And when we press the up arrow button it will push the ball in the vertical direction.

The student runs the code to observe the output.

Output:



We can play with it by moving the ball.

We also change the restitution property and change the behavior of the ball.
 Similarly, we can increase the amount of force and direction to observe the effects.

ESR:

The student changes different parameters and observes their effect on ball motion.

Now we have come to the end of this class.

We have learned a lot of new concepts such as creating a physics body and then multiple objects.

We learned that we can change the position of a body by applying force to it.

In the next class, we will be starting on our quest to create the Pirate Invasion Game, and we will learn about how to use the concept of blueprints in the game.

Teacher Guides Student to Stop Screen Share




Wrap-Up SESSION - 5 Mins



Teacher starts slideshow

from slide 17 to slide 27

Activity details	Solution/Guidelines
<p>Run the presentation from slide 14 to slide 24.</p> <p>The following are the wrap-up session deliverables:</p> <ul style="list-style-type: none"> ● Explain the facts and trivias ● Next class challenge ● Project for the day ● Additional Activity 	<p>Guide the student to develop the project and share it with us.</p>
Quiz time - Click on in-class quiz	
Question	Answer
<p>Consider the following snippet of code</p> <pre>function hForce() { Matter.Body.applyForce(ball,{x:0,y:0},{x:0.05,y:0}); }</pre> <p>The applyForce() function is _____</p> <p>A. applying force in the horizontal direction. B. applying force in the vertical direction. C. applying force in both directions. D. applying force in none of the directions.</p>	<p>A</p>
<p>What does the following snippet of code do?</p> <pre>ellipse(ball.position.x,ball.position.y,20);</pre> <p>A. Displaying the ball by passing its position in the ellipse function.</p>	<p>A</p>

<p>B. Creating an ellipse on the screen.</p> <p>C. Creating the sprite in place of the ball.</p> <p>D. None of the above.</p>	
<p>Which property is used to bounce the body off the ground?</p> <p>A. restitution</p> <p>B. isStatic</p> <p>C. rotate()</p> <p>D. applyForce()</p>	<p>A</p>
<p>End the quiz panel</p>	
<p>FEEDBACK</p> <ul style="list-style-type: none"> • Encourage the student to make reflection notes in Markdown format. • Complement the student for her/his effort in the class. • Review the content of the lesson. 	
<p>Step 4: Warp-Up (5 mins)</p>	<p>You get Hats off for your excellent work!</p> <p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div data-bbox="1055 1203 1347 1302"> <p>Creatively Solved Activities  +10</p> </div> <div data-bbox="1055 1312 1347 1411"> <p>Great Question  +10</p> </div> <div data-bbox="1055 1421 1347 1520"> <p>Strong Concentration  +10</p> </div>
<p>Project Overview</p> <p>CRUMPLED PAPER BALLS</p> <p>Goal of the Project:</p> <p>In Class 21, you learned how to apply force on the body. Also, how to create a class for the ground body to create multiple objects. In this project, you will have to apply what</p>	<p><i>Students engage with the teacher over the project.</i></p>

you have learned in the class and design the elements of a simple game of throwing crumpled paper balls in a waste paper basket.

Story:

You want to inculcate the habit of throwing the waste in the trash bin in young individuals and help keep your city clean. So you have decided to create a simple game of throwing crumpled paper balls in a waste paper basket.

I am very excited to see your project solution and I know you will do really well.

Bye Bye!

Teacher ends slideshow



Teacher Clicks

✕ End Class

ADDITIONAL ACTIVITIES

Additional Activities

Encourage the student to write reflection notes in their reflection journal using Markdown.

Use these as guiding questions:

- What happened today?
 - Describe what happened.
 - The code I wrote.
- How did I feel after the class?
- What have I learned about programming and developing games?
- What aspects of the class helped me? What did I find difficult?

The student uses the Markdown editor to write their reflections in a reflection journal.

Activity Name	Description	Link
Teacher Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/C21_matterjs-Boilerplate
Teacher Activity 2	Complete Reference code	https://github.com/pro-whitehatjr/c21_complete
Student Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/Pro_c21_SA
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Assessment/PRO_VD/BJFC-PRO-V3-C21-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/3ff333a2-8178-4bca-b607-4146b3db5dc5.pdf