Topic	CANNON AND CANNONBALL	
Class Description	**Students will design the cannonBall class. Students will also create the moving cannon to adjust the shooting angle and shoot the cannonballs.**	
Class	**C23**	
Class time	**45 mins**	
Goal	• Code to set different angles to cannon. • Create cannonballs class. • Fire cannonballs from the cannon at the set angle.	

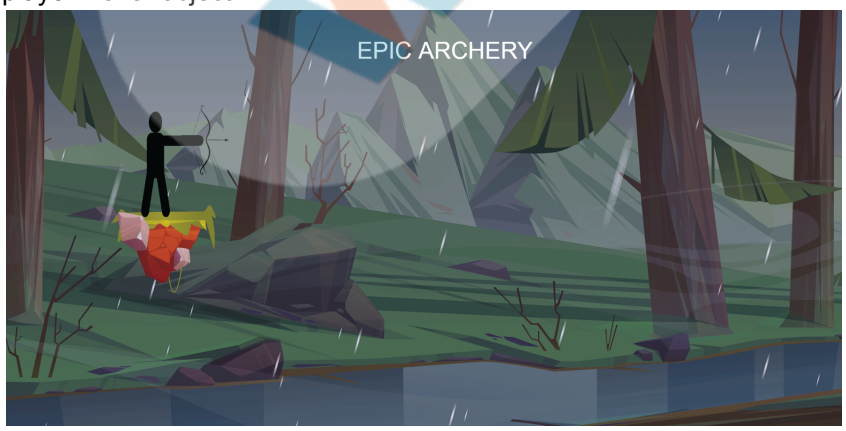
Resources Required	• Teacher Resources:
	  ○ Laptop with internet connectivity
	  ○ Visual Studio Code
	  ○ Earphones with mic
	  ○ Notebook and pen
	• Student Resources:
	  ○ Laptop with internet connectivity
	  ○ Visual Studio Code
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**WARM-UP** **Teacher-led Activity** **Student-led Activity** **WRAP-UP**	**5 mins** **10 mins** **25 mins** **5 mins**

WARM-UP SESSION - 5 mins

Teacher starts slideshow  **from slides 1 to 13**

Refer to speaker notes and follow the instructions on each slide.

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 3** **Following are the WARM-UP session deliverables:** • **Greet the student.** • **Revision of previous class activity.** • **Warm-up Quiz**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	**Answer**
Which of the following is the correct line of code to make a rectangular shaped body? A. this.body = Matter.Bodies(x, y, width, height, options); B. this.body = Matter.Bodies.rect(x, y, width, height); C. this.body = Matter.rectangle(x, y, width, height, options); D. this.body = Matter.Bodies.rectangle(x, y, width, height, options);	**D**
Which of the following option is the correct block to create a playerArcher object? 	**B**

A. playerArcher = new (340, playerBase.position.y - 112, 120, 120); B. playerArcher = new PlayerArcher(340, playerBase.position.y - 112, 120, 120); C. playerArcher = PlayerArcher(340, playerBase.position.y - 112, 120, 120); D. playerArcher = new PlayerArcher( );	
Continue the WARM-UP session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 4 to 13 to set the problem statement. **Following are the WARM-UP session deliverables:** ● Appreciate the student on his performance in the quizzes.	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
CHALLENGE ● **Create class blueprints for the different objects in the game.**	
Teacher Action	**Student Action**
Let us quickly review the code we wrote from the last class. *Teacher downloads the template code from Teacher Activity 1 and runs in the VS Code Editor* *Teacher reviews the code for the **Tower, Ground and***	*ESR: Student reads through the code and explains what each block of code does.*

***Cannon class** and how to create the **tower and ground using Matter.bodies and cannon** objects using the class.*	
In the previous class we had created our cannon. Now we need the cannonball to shoot from it.	
	
The above image is for Teacher ref; Teacher and student can open Teacher Activity 2 / student Activity 2 to show the output.	
So now let's start by creating the **CannonBall.js** file in the js folder and add it to the **index.html file**. Teacher creates a **cannonBall.js** file and adds it to the **index.html** file.	

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Pirates Invasion</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.10.2/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.10.2/addons/p5.sound.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.8.0/addons/p5.dom.min.js"></script>
    <script src="./lib/p5.gif.min.js"></script>
    <script src="./lib/matter.js"></script>

    <link rel="stylesheet" type="text/css" href="style.css" />

    <script src="./js/Cannon.js"></script>
    <script src="./js/CannonBall.js"></script>
  </head>
  <body>
    <script src="sketch.js"></script>
  </body>
</html>
```

In **CannonBall.js**, we'll create the **cannonball** class. Inside the class, we'll create the **constructor()** function. This function will take 2 parameters x and y that will be the positions of the cannonball.	
What is the shape of a cannonball?	**ESR:** The shape of the cannonball is circular.
The cannonball we are going to create will be circular in shape and the circles have a radius. So the first property that we would have in the constructor is the radius. Using **this.radius** we'll set the radius to 30.	
Now we need to create the circular body for the cannonball, can you tell me how we can create a circular body?	**ESR:** We can create the circular body using the **Bodies.circle()**
Yes, so using **Bodies.circle()** we'll create the circular body. And then add the body to the world. The cannonball should not move by itself, right?	**ESR:** **yes**

You must have seen in the movies that when a person ignites the wire then only the cannon fires.
So to keep the cannonball stationary we are going to set **isStatic** as true in the options.

We'll also load the cannonball image so that we can add it to the cannonball in the display.

```
class CannonBall {
  constructor(x, y) {
    var options = {
      isStatic: true
    };
    this.r = 30;
    this.body = Bodies.circle(x, y, this.r, options);
    this.image = loadImage("./assets/cannonball.png");
    World.add(world, this.body);
  }
```

Now our constructor is ready, we need to display this cannonball.
In the cannonball class, we'll create the **display()** function.

In this function, we'll first declare a variable as var **pos = this.body.position,** this variable will store the x,y position of the cannonball body.
When we need to access the position value we can simply write pos.x and pos.y, instead of writing this.body.position.x.

Now we need to display the image of the cannon ball. Ideally the position of the cannon ball will be the same as the position of the cannon.

But when we create the image in p5.js, it draws the image from the top left corner.

We want our image to be drawn from the center.
In order to do that we are going to use the **imageMode(CENTER)** function.
This function tells our code to draw the images from the center which means when we pass the x and y position of the image, these points will be at the center of the image not at the top left corner.

But the problem is, this setting will be applied to all the images on the canvas including the background image. Due to this the center of the background image will be at the origin of the canvas.

Following image shows the effect of the imageMode(CENTER) when not enclosed in the push() and pop()

To prevent these issues we can use **push()** and **pop()** functions.

These functions are used when we want to apply new settings only for certain shapes or images.
In our case we will enclose the **imageMode(CENTER)** and **image()** between the **push()** and **pop()**

```
display()
{
  var pos = this.body.position;
  push();
  imageMode(CENTER);
  image(this.image, pos.x, pos.y, this.r, this.r);
  pop();
}
```

Now the class is ready. So let's create the cannonball object in the **sketch.js** file.
Inside the **setup()** function we'll create a new cannonball using the cannonball class that we created.

We'll be passing the position of the cannon to the cannonball as we want the cannonball to be at the position from which we can shoot it.

*The teacher codes to create the new cannonball class in the **setup()** function.*

in the **setup()** function in **sketch.js** file

```
cannon = new Cannon(180, 110, 130, 100, angle);
cannonBall = new CannonBall(cannon.x, cannon.y);
```

In the **draw()** function

```
cannon.display();
cannonBall.display();
```

OUTPUT:

Awesome, now we can see the cannonball on the cannon.	

But our cannon and the cannonball both are static now. How should cannon and cannonball move according to you? Great! Cannon should move up and down and the ball can be shot from the cannon.	**ESR:** We want the cannon to move left to right in the angle. And we want to shoot the cannonball from the cannon.
Do you want to add that functionality? Alright! Let's get you started.	**ESR:** Yes.

Teacher Stops Screen Share
STUDENT-LED ACTIVITY - 25 mins
• **Ask Student to press ESC key to come back to panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**
ACTIVITY • **Add the angle to the cannon.** • **Shoot the cannonball from the cannon.**
Teacher starts slideshow  **:Slide 14 to slide 17**

Run the presentation slide to set the student activity context. • Add the angle to the cannon. • Shoot the cannonball from the cannon.	

Teacher ends slideshow 	
Teacher Action	**Student Action**
Guide the student to download the code. First, we'll start by changing the angle of the cannon. In the game how was the cannon moving in the game?	*Student downloads the code for Student Activity 1 and opens in the VS Code editor.* **ESR:** The cannon was moving in the upward direction when the left arrow key was pressed and in a downward direction when the right arrow key was pressed.
Yes! So let's write the code to change the angle of the cannon when the left and right arrow keys are pressed. As we'll be changing the angle property we'll define a variable to store the angle in the **constructor()** for the cannon in **cannon.js**. We'll be using an if condition to check which arrow key has been pressed by the user. If the left arrow key is pressed then the cannon would move in the upward direction.	

```
class Cannon {
  constructor(x, y, width, height, angle) {
    this.x = x;
    this.y = y;
    this.width = width;
    this.height = height;
    this.angle = angle;
    this.cannon_image = loadImage("assets/canon.png");
    this.cannon_base = loadImage("assets/cannonBase.png");
  }
```

But we write the code to change the angle of the cannon; we need to understand how angles work in the p5.js library.

Angles have 2 units-radians and degrees.
Just as we have different units for measuring length such as meter and feet.

Radians and degrees both are useful. But degrees is easy to work with because we have learned about the degrees in our math and drawing class as well.

But the problem is that in p5.js the unit of the angle is by radians by default.
So we have to change that to degrees.

For that purpose we have a function called **angleMode()**.
In this function we will write the name of the unit we want to use for angle.

In our case we will write **angleMode(DEGREES).**

This code we will write in the **setup()** function.

We also need to set the initial value for the angle. Let's keep it 15. When our code will start, the cannon will be at this angle and then the user can increase or decrease the angle value.

```
function setup() {
  canvas = createCanvas(1200, 600);
  engine = Engine.create();
  world = engine.world;

  angleMode(DEGREES);
  angle = 15;

  ground = Bodies.rectangle(0, height - 1, width * 2, 1, { isStatic: true });
  World.add(world, ground);

  tower = Bodies.rectangle(160, 350, 160, 310, { isStatic: true });
  World.add(world, tower);

  cannon = new Cannon(180, 110, 130, 100, angle);
  cannonBall = new CannonBall(cannon.x, cannon.y);
}
```

Our angle mode is now set for degrees; now we can write the if condition to move the cannon up and down.

For example, to move the cannon downwards we will add a value to our angle.
In p5.js angle increases in the clockwise direction.

To move the cannon upwards we will be subtracting a value from the angle.

The value to be added or reduced from the angle can be chosen based on how much movement we want on each key press. If we choose a very high value, then the cannon will move alot by each key press and vice versa.

<The teacher guides the student to add the condition to change the cannon angle in upward direction when the left arrow key is pressed and in downward direction when the right arrow key is pressed.>	*<The student codes to add the condition to change the cannon angle in upward direction when the left arrow key is pressed and in downward direction when the right arrow key is pressed.>*

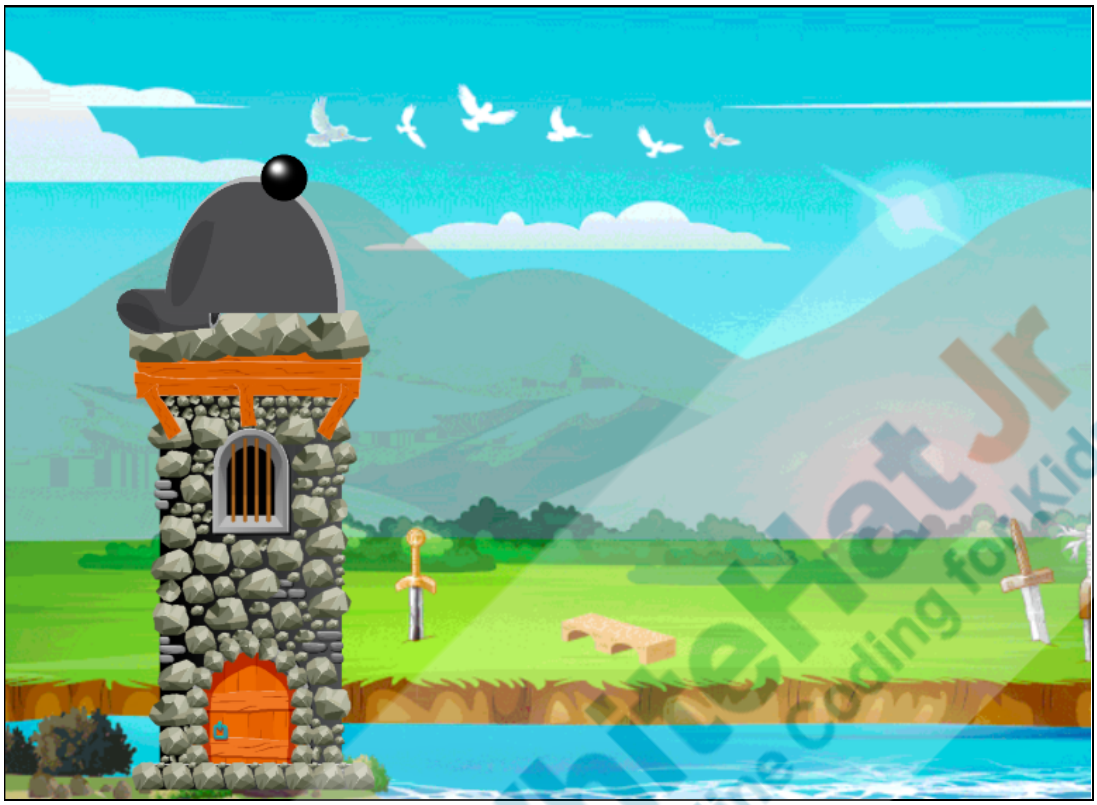
In the **cannon.js** file,

```
display() {
    console.log(this.angle)
    if (keyIsDown(RIGHT_ARROW)) {
      this.angle += 1;
    }

    if (keyIsDown(LEFT_ARROW)) {
      this.angle -= 1;
    }

    push();
    rotate(this.angle);
    imageMode(CENTER);
    image(this.cannon_image, this.x, this.y, this.width, this.height);
    pop();
    image(this.cannon_base, 70, 20, 200, 200);
    noFill();
  }
}
```

Now our cannon is moving but there is still an issue. Can you tell me what the issue is? If you look carefully, the cannon is rotating around the origin point which is at the top left corner.	**ESR:** varied



Why do you think this is happening?	**ESR:** Varied

When we rotate any object in p5.js by default it rotates around the origin. But in our case, we want to rotate the cannon at its origin point right? To solve this problem we need to shift our origin at the position of the cannon. For that we can use a function named **translate().** This function translates (shift) the origin to the points given inside the brackets. In our case, we want to shift the origin to the position of the cannon; so we need to pass this.x and this.y in the **translate()** function. But this should be enclosed between the **push()** and **pop()** function otherwise we will shift the origin for all the bodies and we do not want that to happen. There is one more important concept which we have to be aware of since we are shifting the origin to the position of the cannon then to create the cannon we will only give x and y as 0,0. Because the origin itself came at the position of the cannon and we want to create the cannon there only. If we write some other value then the cannon will be shifted to that amount in the x and y-direction.	**ESR:** Yes.

```
  push();
  translate(this.x, this.y);
  rotate(this.angle);
  imageMode(CENTER);
  image(this.cannon_image,0,0, this.width, this.height);
  pop();
  image(this.cannon_base, 70, 20, 200, 200);
  noFill();
}
```

Now we can see that cannon is moving at its own origin point.

But there is still one problem.
Can you tell me what is the issue here?

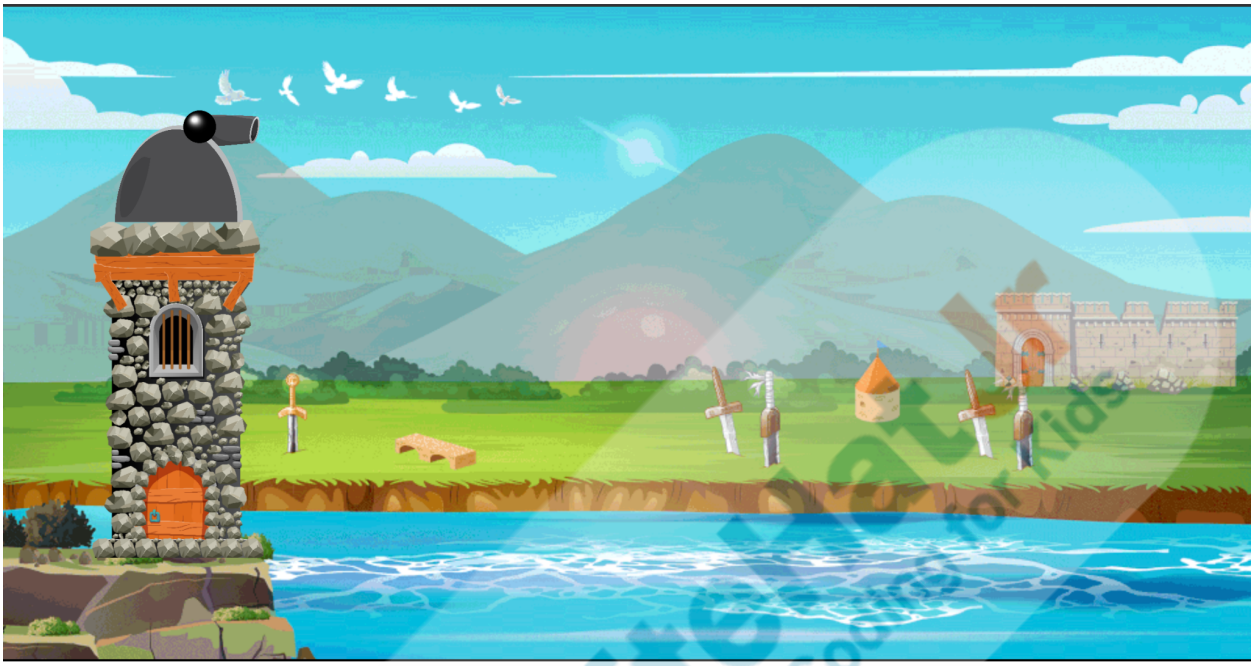
ESR:
The cannon is rotating at 360 degrees.

Yes, we do not want that and in reality cannons do not rotate like this; they have a certain range of angle in which they move up and down. To restrict the motion of the cannon we can use the if conditions. To add the multiple conditions we are going to use AND(&&) operator. In our case we are restricting the movement of the cannon between 70 and -30 degrees. You can change these values if you want more movement in the cannon. *<The teacher guides the student to add the condition to restrict the cannon movement.>* *The teacher can have the student try the different values for **this.angle** to explain the impact.*	*Student codes to add the condition to restrict the cannon movement.*

```
display() {
  console.log(this.angle)
  if (keyIsDown(RIGHT_ARROW) && this.angle<70  ) {
    this.angle += 1;
  }

  if (keyIsDown(LEFT_ARROW) && this.angle>-30 ) {
    this.angle -= 1;
  }
```

The teacher asks the student to run the code to check the output. Now we have the cannon moving as we want.	*Student runs the code to check the output*

The next part will be to create the **cannonball** object in the **sketch.js** file and shoot it from the cannon.	
	
How do we want the **cannonball** object to be moving? To do so let's create a function called **shoot()** in the **cannonBall.js** file where we'll **set the velocity** to the cannonball. When we created the cannonball we set its isStatic to true, this will keep the ball stationary. In order to shoot the ball first we need to set this flag to false. But we only want this when the user presses the button to shoot. For this we will use a function from the matter.js library this is called as **Matter.Body.setStatic(this.body,false)**	**ESR:** We need the cannonball to be launched in a certain direction and velocity. *<The student codes to create a **shoot()** function. Then sets the velocity to the ball in the shoot function and then calls the function when the key is released.>*

This will enable us to move our ball.

Now we are going to set the velocity(speed) of the cannonball.

Matter.min.js library has a function that will help us to set some velocity to the cannonball. The function is called **setVelocity()**.

This function takes the body to which we want to give velocity.
And the x & y velocity value as the parameters.

x value is 30 which will make the ball move in the right direction and the y value is -20 which means ball will move downwards. When both of these are values applied together like we are doing here then the ball will move in right direction and then keep on falling down.

```
Matter.Body.setVelocity(this.body, { x: 30, y: -20 });
```

Our **shoot()** function is defined now, we'll call this function on key release.

In sketch.js file

```
cannonBall = new CannonBall(cannon.x, cannon.y);
```

In cannonBall.js file

```
shoot() {
  Matter.Body.setStatic(this.body, false);
  Matter.Body.setVelocity(this.body, { x: 30, y: -20 });
}
```

In sketch.js

```
function keyReleased() {
  if (keyCode === DOWN_ARROW) {
    cannonBall.shoot();
  }
}
```

OUTPUT:

Now what do we see? And how do we want the cannonball to be shot from the cannon?	**ESR:** The ball is getting shot from the cannon only in one direction. ***ESR:*** We want the cannonball to be shot from the ball at the angle that the cannon is pointing.

To do so **p5 library** has a predefined function **p5.Vector.fromAngle()**. this function by default accepts the angle in **radians** but The angle we are providing is on degrees as our angle mode is in degrees, in order to pass the angle value to this function we need to **convert angle to radians.** That is done by multiplying the angle value with (pi/180) which is (3.14/180). so that our code will work in proper manner This function creates and returns a 2D vector. Vector is just like an array. This vector contains x and y values. A vector is a quantity that has a magnitude and the direction as well. Like we want to give velocity to our cannon ball. For that we need the amount of the velocity, that will be our magnitude and we also gave the direction in which our cannonball should move. From this vector, we can get the x and y values and pass them to our cannonball. We are already setting the angle to the cannonball by moving the cannon so we just need to get the velocity from the given cannon angle. When we get the velocity from the function we'll multiply the force using the multiply function as we want the cannonball to be launched with a great speed. in the **setVelocity()** function the angle is in radians, but we need to convert that into degrees by multiplying with (180/pi) which is (180/3.14)	*Student opens Student Activity 3* *<The student codes to use the **p5.Vector.fromAngle()** function.>*

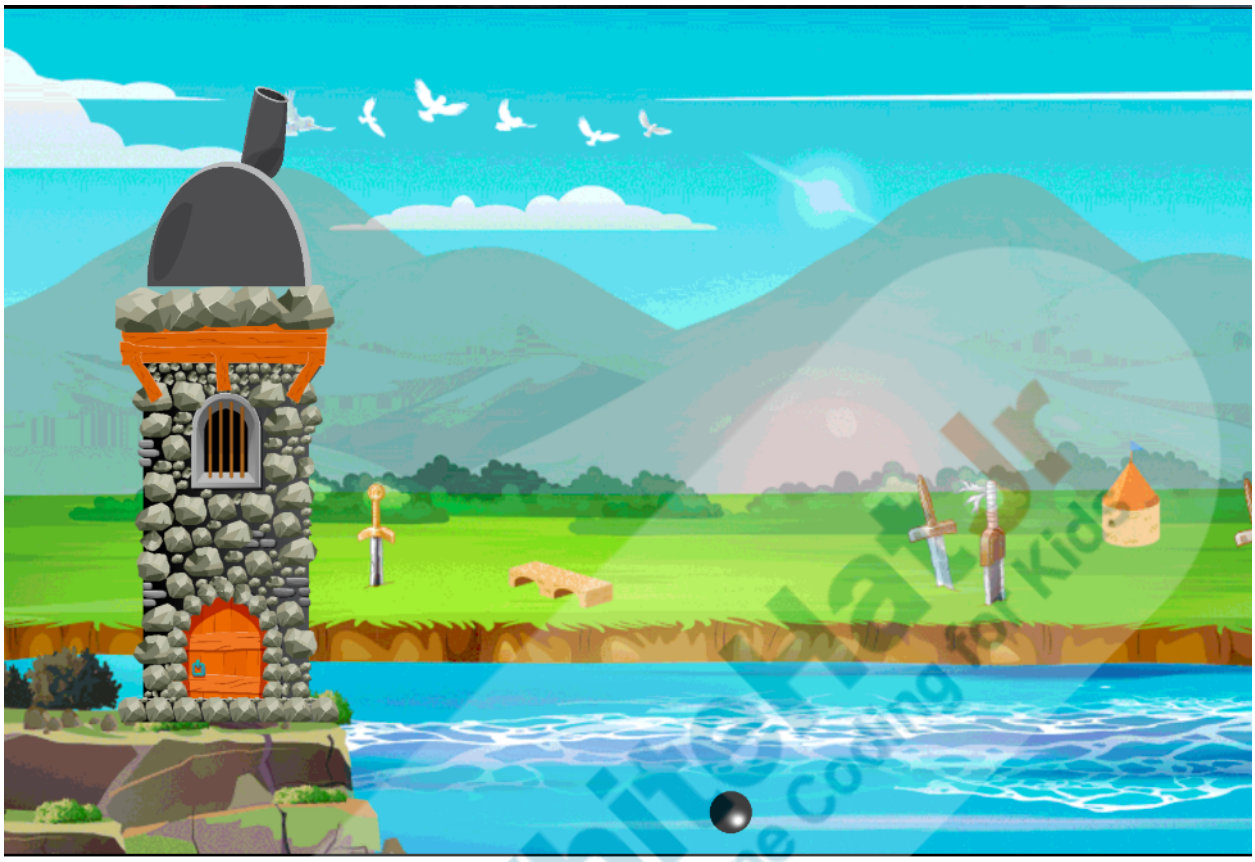
Shoot() function in the **cannonBall.js** file.

```
shoot() {
  var newAngle = cannon.angle - 28;
  newAngle = newAngle*(3.14/180)
  var velocity = p5.Vector.fromAngle(newAngle);
  velocity.mult(0.5);
  Matter.Body.setStatic(this.body, false);
  Matter.Body.setVelocity(this.body, {
    x: velocity.x * (180 / 3.14),
    y: velocity.y * (180 / 3.14) });
}
```

- *Modify the code in **cannonBall.js** for **shoot().***
- *Store the angle of canon in a variable **velocity.***
- *Multiply that value with 0.5 using mult().*
- *Change the **isStatic** property of cannonBall to false, so that It can fall.*
- *Set velocity using **setVelocity()** and **velocity** variables.*

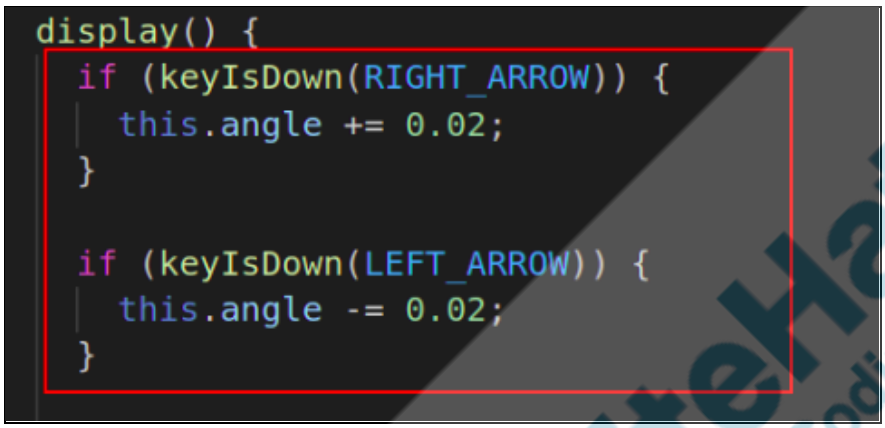
mult() function used here will multiply the values stored in var **velocity** with 0.5 as the initial values that we are getting from the vector are low. So by multiplying the value we are increasing the velocity to give good speed to cannonballs.

Teacher will ask the student to run the code and check, and help the student to identify and fix any errors.	*Student runs the code to check output.*



Awesome! Now our cannonball is being shot as we wanted.	

Teacher Guides Student to **Stop Screen Share**	
WRAP-UP SESSION - 5 Mins	
Teacher starts **slideshow**  **from slide 18 to 28**	
Activity details	**Solution/Guidelines**
Run the presentation from slide 18 to 28 **Following are the WARM-UP session deliverables:** • **Appreciate the student.** • **Revise the current class activities.**	Discuss with the student the current class

• **Discuss the quizzes.**	activities and Student will ask doubts related to the activities.
Quiz time - Click on in-class quiz	
Question	**Answer**
What does the following piece of code do?  A.Changes the angle of the cannon if the user presses the right or left arrow key. B.Changes the position of the cannon if the user presses the right or left arrow key. C.Changes the angle of the cannon if the user clicks on the screen. D. Changes the cannonball angle.	**D**
What is the syntax of setVelocity() function? A.Matter.Body.SetVelocity(velocity,body) B.Matter.SetVelocity(velocity,body) C.Matter.Body.setVelocity(body, velocity) D.matter.body.SetVelocity(velocity,body)	**C**
To make the cannonball to be shot from the ball in the angle that the cannon is pointing, what do we use?	**B**

A. p5.Vector.shoot() B. p5.Vector.fromAngle() C. p5.Vector.shootFromAngle() D. Vector.fromAngle()	

End the quiz panel

FEEDBACK

- **Appreciate the student for their efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Step 4: Wrap-Up (5 mins)	You get Hats off for your excellent work! So far we are just shooting one ball from the cannon as of now. In the next class, we'll learn how to create multiple balls and shoot them from the cannon.	*Make sure you have given at least 2 Hats Off during the class for:* 
*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.** **Project Overview** **EPIC ARCHERY STAGE 2** **Goal of the Project:** In Class 23, you have learned how to create a cannonball and set angles for cannonballs and cannon. In this project, you will create bows for the player and change the player		**Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.**

bow's angle with arrow keys. Plus, shoot the arrow on pressing a space key. *** This is a continuation of Project-22; make sure you have completed and submitted that before attempting this one.** **Story:** Archery is one of the oldest arts which is still practiced. After reading the information about Archery in a book, your friend Georgie wants to play Archery. To give him a virtual experience, you want to use your coding expertise and physics engine concepts to create an Archery game for him. Can you allow him to set the angle of the bow using arrow keys? Also, create an arrow and shoot towards the target. I am very excited to see our project solution and I know you will do really well. Bye Bye!	*Students engage with the teacher over the project.*
Teacher ends slideshow 	
Teacher Clicks 	
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?	*The student uses the Markdown editor to write their reflections in a reflection journal.*

○ 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?	

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C22-reference-link
Teacher Activity 2	Pirates Invasion Game	https://whitehatjr.github.io/PiratesInvasion/
Teacher Activity 3	Teacher Reference	https://github.com/pro-whitehatjr/PRO-C23-Reference_code
Student Activity 1	Boilerplate code.	https://github.com/pro-whitehatjr/PRO-C23-SA-boilerplate
Student Activity 2	Game Output	https://whitehatjr.github.io/PiratesInvasion/
Student Activity 3	Vector from Angle documentation	https://p5js.org/reference/#/p5.Vector/fromAngle
Teacher Reference visual aid link	Visual aid link	https://s3-whjr-curriculum-uploads.whjr.online/fcdc0353-1be7-43f3-b49c-872d650fc6dc.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/e2a1418e-106b-4d86-81dd-ccf50a289f96.pdf
Project Solution	Epic Archery Stage 2	https://github.com/pro-whitehatjr/pro-c23-project-sol