```
public class PlayerScript2D : MonoBehaviour
{

    public GameObject trajectoryPointPrefeb;
    //public GameObject ballPrefab;
    public float gravityScale = 1f;
    public float mass = 1f;
    public int numOfTrajectoryPoints = 30;
    public float power = 25;
    public float minRange = -48;
    public float maxRange = 48;
    public bool aboutToJump = false;
    private int defaultLayer = 9;
    private float originOffSet = 0.15f;
    Rigidbody2D rigidbody2d;
    public float xVelocity = 100f;
    public float yVelocity = -100f;
    bool movingRight = false;
    public bool jumped = false;
    //private GameObject ball;
    private bool isPressed, isBallThrown;
    private List<GameObject> trajectoryPoints;
    //private Vector3 currentPosition;
    //private GameObject[] ground;

    //private Rigidbody rigid;

    // Use this for initialization
    void Start()
    {
        rigidbody2d = GetComponent<Rigidbody2D>();
        GetComponent<Rigidbody2D>().mass = mass;
        //ground = GameObject.FindGameObjectsWithTag("Ground");
        trajectoryPoints = new List<GameObject>();
        isPressed = isBallThrown = false;
        //currentPosition = transform.position;
        //rigid = GetComponent<Rigidbody>();
        // TrajectoryPoints are instatiated and added to the list
        for (int i = 0; i < numOfTrajectoryPoints; i++)
        {
            // optimization needed for
GetComponent<SpriteRenderer>
            GameObject dot =
(GameObject)Instantiate(trajectoryPointPrefeb);
            dot.GetComponent<SpriteRenderer>().enabled = false;
            trajectoryPoints.Add(dot);
        }
    }

    // Update is called once per frame
    void Update()
    {
```

```
        var horz = Input.GetAxis("Horizontal");
        int layerMask = ~(1 << defaultLayer); //Exclude layer 9
        //RaycastHit2D hitInfo =
Physics2D.Raycast(transform.position, -Vector2.up);
        Vector2 startingPoint = new Vector2(transform.position.x,
transform.position.y + originOffSet);

        // Raycast For left collision
        RaycastHit2D hitInfoLeft =
Physics2D.Raycast(startingPoint, Vector2.left, 0.5f, layerMask);
        Debug.DrawRay(startingPoint, (Vector2.left.normalized *
0.5f), Color.red);

        // Raycast For right collision
        RaycastHit2D hitInfoRight =
Physics2D.Raycast(startingPoint, Vector2.right, 0.5f, layerMask);
        Debug.DrawRay(startingPoint, (Vector2.right.normalized *
0.5f), Color.red);

        RaycastHit2D hitInfoDown =
Physics2D.Raycast(startingPoint, Vector2.down, 0.7f, layerMask);
        Debug.DrawRay(startingPoint, (Vector2.down.normalized *
0.7f), Color.red);

        if (hitInfoDown.collider != null && !isPressed)
        {
            aboutToJump = false;
        }

        if (Input.GetMouseButtonDown(0) && hitInfoDown.collider !=
null)
        {
            isPressed = true;
            aboutToJump = true;

            jumped = false;
            //transform.position = currentPosition;

            //if (isBallThrown)
            //{
            //Destroy(GetComponent<Rigidbody2D>());
            GetComponent<Rigidbody2D>().velocity = new Vector2(0,
0);
            //Debug.Log(" Second code velocity: " +
GetComponent<Rigidbody2D>().velocity);
            GetComponent<Rigidbody2D>().angularVelocity = 0.05f;
            //}


            // Remove this if you want to keep the line(Trajectory
Points) shown
            if (trajectoryPoints.Count == 0)
```

```
            {
                for (int i = 0; i < numOfTrajectoryPoints; i++)
                {
                    // obtimization needed for
GetComponent<SpriteRenderer>
                    GameObject dot =
(GameObject)Instantiate(trajectoryPointPrefeb);
                    dot.GetComponent<SpriteRenderer>().enabled =
false;
                    trajectoryPoints.Add(dot);
                }
            }

        }
        else if (Input.GetMouseButtonUp(0) && hitInfoDown.collider
!= null)
        {
            isPressed = false;
            jumped = true;
            ThrowBall();

            foreach (GameObject point in trajectoryPoints)
            {
                Destroy(point);
            }

            trajectoryPoints.Clear();
        }

        // when mouse button is pressed, cannon is rotated as per
mouse movement and projectile trajectory path is displayed.
        if (isPressed)
        {
            //Vector3 vel = GetForceFrom(ball.transform.position,
Camera.main.ScreenToWorldPoint(Input.mousePosition));
            Vector3 vel = GetForceFrom(transform.position,
Camera.main.ScreenToWorldPoint(Input.mousePosition));
            //float angle = Mathf.Atan2(vel.y, vel.x) *
Mathf.Rad2Deg;
            /*vel.x = Mathf.Clamp(vel.x, -20, 20);
            vel.y = Mathf.Clamp(vel.y, -6, 18);*/
            //Debug.Log(vel);
            // Rotation code
            //transform.eulerAngles = new Vector3(0,0,angle);

            if (GetComponent<Rigidbody2D>() == null)
            {
                gameObject.AddComponent<Rigidbody2D>();
            }

            /*tComponent<Rigidbody2D>().mass = mass;
            GetComponent<Rigidbody2D>().gravityScale =
```

```
gravityScale;*/
            GetComponent<Rigidbody2D>().mass = mass;
            GetComponent<Rigidbody2D>().gravityScale = 0f;
            setTrajectoryPoints(transform.position, vel /
GetComponent<Rigidbody2D>().mass);
        }

        //Debug.Log(playerScript.aboutToJump);

        if (horz > 0 && !isPressed)
        {
            if (!movingRight)
            {
                Vector2 v = rigidbody2d.velocity;
                v.x = 0;
                rigidbody2d.velocity = v;
                rigidbody2d.angularVelocity = 0.05f;
            }
            //Vector2 movement = new Vector2(horz, 0);
            //rigidbody2d.velocity = new Vector2(horz,
rigidbody2d.velocity.y);
            if (hitInfoRight.collider != null)
            {
                //Debug.Log(hitInfoRight.collider.name);
                Vector2 v = rigidbody2d.velocity;
                v.x = 0;
                rigidbody2d.velocity = v;
                rigidbody2d.angularVelocity = 0.05f;
            }
            else if (hitInfoRight.collider == null)
            {
                //rigidbody2d.AddForce(movement);
                rigidbody2d.AddForce(new Vector2(xVelocity,
yVelocity));
            }

            movingRight = true;
        }
        else if (horz < 0 && !isPressed)
        {
            if (movingRight)
            {
                Vector2 v = rigidbody2d.velocity;
                v.x = 0;
                rigidbody2d.velocity = v;
                rigidbody2d.angularVelocity = 0.05f;
            }

            //Vector2 movement = new Vector2(horz, 0);
            //rigidbody2d.velocity = new Vector2(horz,
rigidbody2d.velocity.y);
            if (hitInfoLeft.collider != null)
```

```
            {
                //Debug.Log(hitInfoLeft.collider.name);
                Vector2 v = rigidbody2d.velocity;
                v.x = 0;
                rigidbody2d.velocity = v;
                rigidbody2d.angularVelocity = 0.05f;
            }
            else if (hitInfoLeft.collider == null)
            {
                //rigidbody2d.AddForce(movement);
                rigidbody2d.AddForce(new Vector2(-xVelocity,
yVelocity));
            }

            movingRight = false;
        }
        else if (horz == 0 || horz != 0)
        {
            Vector2 v = rigidbody2d.velocity;

            if (hitInfoDown.collider != null && !jumped)
            {
                //Debug.Log("Stopped");
                v.x = 0;
            }

            rigidbody2d.velocity = v;
            rigidbody2d.angularVelocity = 0.05f;
        }
    }

    // Create new ball
    /*private void createBall()
    {
        ball = (GameObject)Instantiate(ballPrefab);
        Vector3 pos = transform.position;
        pos.z = 1;
        ball.transform.position = pos;
        ball.SetActive(false);
    }*/

    // Throw the ball
    private void ThrowBall()
    {
        //ball.SetActive(true);
        if (GetComponent<Rigidbody2D>() == null)
        {
            gameObject.AddComponent<Rigidbody2D>();
        }

        isBallThrown = true;
```

```
        // Added for testing remove later
        //tComponent<Rigidbody2D>().mass = mass;
        GetComponent<Rigidbody2D>().mass = mass;
        GetComponent<Rigidbody2D>().gravityScale = gravityScale;

GetComponent<Rigidbody2D>().AddForce(GetForceFrom(transform.positi
on, Camera.main.ScreenToWorldPoint(Input.mousePosition)),
ForceMode2D.Impulse);
    }

    // Following method gives force to the ball
    private Vector2 GetForceFrom(Vector3 fromPos, Vector3 toPos)
    {
        Vector2 force = (new Vector2(toPos.x, toPos.y) - new
Vector2(fromPos.x, fromPos.y)) * power;
        force.x = Mathf.Clamp(force.x, minRange, maxRange);
        force.y = Mathf.Clamp(force.y, minRange, maxRange);
        //Debug.Log(force);
        return force;
    }

    // Display projectile trajectory path. It takes two arguments,
start position of object(ball) and initial velocity of
object(ball).
    void setTrajectoryPoints(Vector3 pStartPosition, Vector3
pVelocity)
    {
        float velocity = Mathf.Sqrt((pVelocity.x * pVelocity.x) +
(pVelocity.y * pVelocity.y));
        float angle = Mathf.Rad2Deg * (Mathf.Atan2(pVelocity.y,
pVelocity.x));
        float fTime = 0;

        fTime += 0.1f;
        for (int i = 0; i < numOfTrajectoryPoints; i++)
        {
            float dx = velocity * fTime * Mathf.Cos(angle *
Mathf.Deg2Rad);
            float dy = velocity * fTime * Mathf.Sin(angle *
Mathf.Deg2Rad) - ((Physics2D.gravity.magnitude * gravityScale) *
fTime * fTime / 2.0f);
            Vector3 pos = new Vector3(pStartPosition.x + dx,
pStartPosition.y + dy, 2);
            trajectoryPoints[i].transform.position = pos;
            trajectoryPoints[i].GetComponent<Renderer>().enabled =
true;
            trajectoryPoints[i].transform.eulerAngles = new
Vector3(0, 0, Mathf.Atan2(pVelocity.y - (Physics.gravity.magnitude
* gravityScale) * fTime, pVelocity.x) * Mathf.Rad2Deg);
            fTime += 0.1f;
        }
```

```
        /*foreach (GameObject point in trajectoryPoints)
        {
            foreach (GameObject groundObject in ground)
            {
                if
(groundObject.GetComponent<BoxCollider2D>().bounds.Contains(ground
Object.transform.position))
                {
                    point.GetComponent<SpriteRenderer>().enabled =
false;
                }
            }
        }*/
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        // Enable this if you want the player to not move in the
air
        if (collision.transform.name.Contains("Ground"))
        {
            //aboutToJump = false;
            jumped = false;
        }

        Transform parent = collision.transform.parent;
        if (parent != null)
        {
            if (parent.name == "Saw")
            {
                var explodable = GetComponent<Explodable>();
                explodable.explode();
                ExplosionForce ef =
GameObject.FindObjectOfType<ExplosionForce>();
                ef.doExplosion(transform.position);
            }
        }
    }
}
```