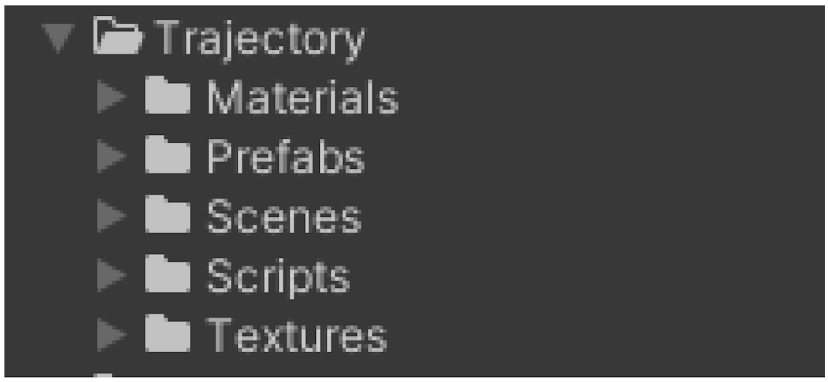
Unity Trajectory

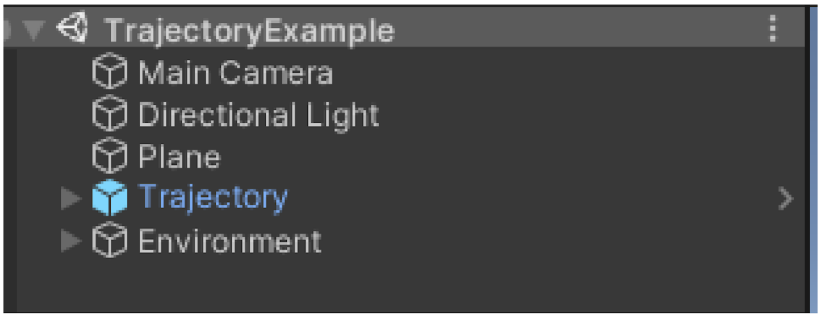
This package does not have any license, you can use it as you want or make modifications.

I will give information about how to use the Trajectory package I have prepared in this document.This code is an example in Unity that simulates drawing a trajectory using a LineRenderer.
When you download the project from Github, the file layout that will appear before you will be like this.

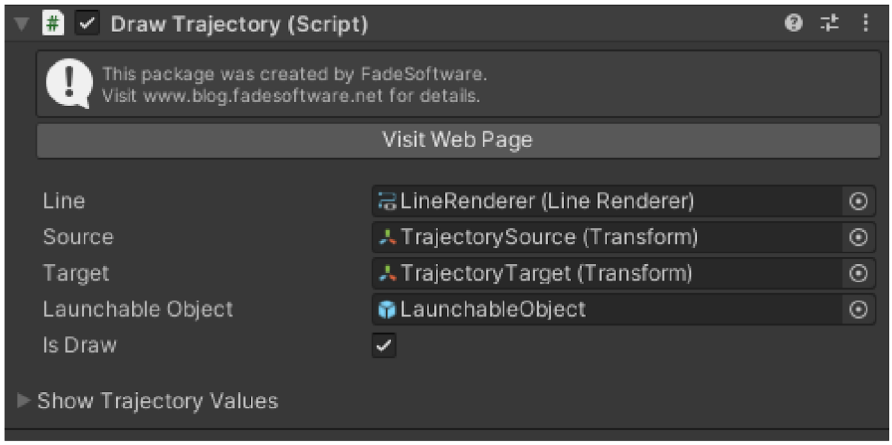


There is one editor class in the package, one static class where calculations are performed, and one monobehaviour class where trajectory operations are performed. The part that interests you is the DrawTrajectory.a script called cs. You will not be dealing with other scripts.

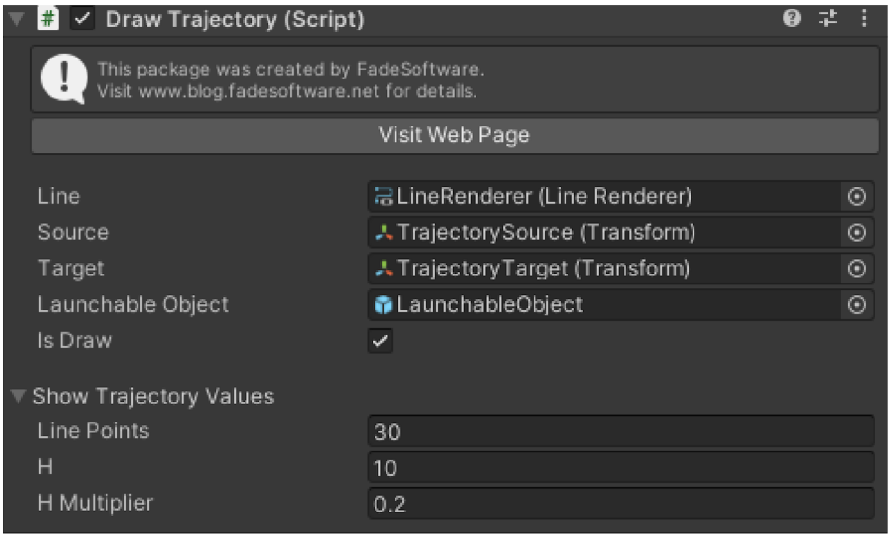
A scene has been prepared in the Scenes folder to be an example for you in a scene called TrajectoryExample.



If you click on the "Trajectory" object in the TrajectoryExample scene, you will see the script called DrawTrajectory.cs, this script does our launching and drawing operations.



DrawTrajectory.cs is written with a special editor for ease of use. If you click on "Show Trajectory Values", you will see the height, height multiplier and number of points required to draw the trajectory.



Now let's make the code explanation

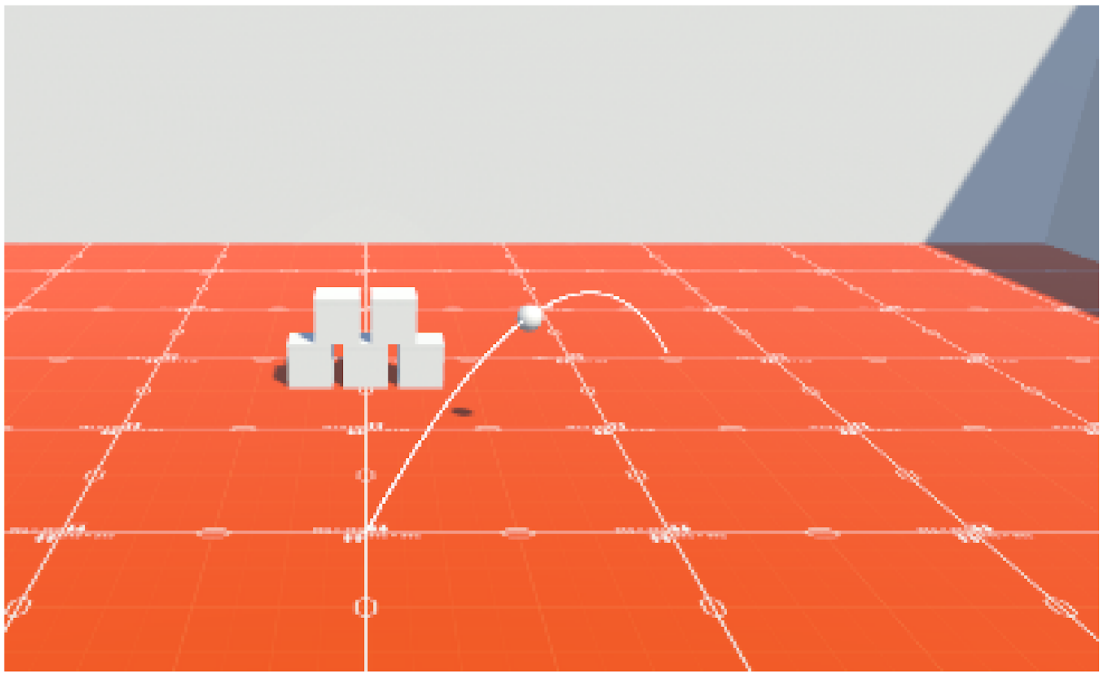
- **The code includes a "DrawTrajectory" class in Unity, which has various variables such as LineRenderer, source position, target position, number of line points, line resolution, height (h), gravity (g), and others.**
- **The Update() function is continuously called and listens for mouse clicks.**
- **If the left mouse button is pressed, the LineRenderer is enabled, and the target position is adjusted based on the mouse position using raycasting.**
- **Some formulas are used to calculate the trajectory height (h).**
- **The TrajectoryCalculations.Draw3DLine() function returns a Vector3 list based on the source and target positions, the number of line points, time interval, height, and gravity.**
- **The position count and positions of the LineRenderer are set to the calculated line points.**
- **If the left mouse button is released, the LineRenderer is disabled, and an object is instantiated.**
- **The initial velocity is calculated using the CalculateInitialVelocity() function and applied to the object using AddForce().**

This way, you can simulate a trajectory drawn by the LineRenderer based on mouse clicks and make an object move along that trajectory.

```
Unity İletisi | 0 başvuru
private void Update()
{
    if (isDraw)
    {
        if (Input.GetMouseButton(0))
        {
            //LineRenderer component enabled equal to true
            line.enabled = true;
            //The target position is moved by raycast.
            target.transform.position = rayUtility.GetMouseWorldPosition(Input.mousePosition, transform.position);
            //Some calculations for the height of the trajectory
            h = (target.position - source.position).magnitude * hMultiplier;
            //We assign values of type Vector3 returned from the Draw3DLine function to a list and give this list to the linerenderer
            List<Vector3> LinePoints = TrajectoryCalculations.Draw3DLine(source.position, target.position, linePoints, timeBetweenPoints, h, g);
            line.positionCount = LinePoints.Count;
            line.SetPositions(LinePoints.ToArray());
        }

        if (Input.GetMouseButtonUp(0))
        {
            //LineRenderer component enabled equal to false
            line.enabled = false;
            //When we lift our hand from the mouse, we create the object
            GameObject obj = Instantiate(launchableObject, source.position, Quaternion.identity);
            //Calculate the velocity with CalculateInitialVelocity and give it to the object as an add force.
            obj.GetComponent<Rigidbody>().AddForce(TrajectoryCalculations.CalculateInitialVelocity(source.position, target.position, h, g), ForceMode.Impulse);
        }
    }
}
```

To use it, all you need to do is throw the DrawTrajectory script on the stage and give the corresponding variables and components.



You can access the codes from this link: https://github.com/FadeSoft/Unity-Library/tree/main/Assets/Trajectory

#Unity Trajectory

You can use on your own project

Video: https://www.youtube.com/watch?v=uyNNa8HsLgQ

Website: https://www.blog.fadesoftware.net/

LinkedIn: https://www.linkedin.com/in/samedbatman/

Mail: softwarefade@gmail.com