Grappling Hook Rope Wrapper 2D guidance documentation

Index:

1-) Summary of the Asset:

The main feature of this asset is a physics 2D extension which provides a line renderer the ability to wrap around other 2D objects.

The other features such as shooting the rope and movement of player is merely there to support or help demonstrate the main feature.

Scripts that provide each feature is separated so that there is no confusion while implementing.

You can find a demonstration of the game here:
https://simmer.io/@Anchorson/fatbottomninjas

2-) Mechanics of the Components:

a-) PlayerMovementController component: This is a simple script which, when attached to a gameobject with rgbdy2d component, moves the component with preassigned velocity. It uses keyboard input W,A,S,D to move.

b-) RopeShootController component: Shoots, releases and anchors the rope. It should be attached to the Rope object.

Shoots the rope starting from the position of the Player and towards the positon of the mouse at the moment of shooting. Uses preassigned velocity to move the rope object and a rgbdy2D component should be attached to rope object.

Player can control using mouse input. Use Left mouse click for shooting, right mouse button for release.

When the rope object coincides an object that the rope can anchor, the movement stops and the anchor position as *Vector3* is sent to the RopeWrapController.cs component as input.

The anchoring mechanism uses OnTriggerEnter2D so keep that in mind.

In order to avoid bugs with raycast2D in wrapping mode, the anchor point is made sure to be out of the collider boundaries of anchored object.

You can check `PushAnchorOutOfColliderBoundaries()` method for details of this mechanism.

c-) RopeReelController component: Sole feature of this component is to apply a preassigned magnitude of force to the Player object towards the closest anchor point (or the pivot point) of the rope. In effect, reeling the player.

d-) RopeWrapController component: This is the mvp of all components and the main point of this asset.

After getting the anchor input from the RopeShootController component, this component draws the rope on screen using the linerenderer.

Using raycast2D, it continiously checks for obstacles on screen (colliders) that obstructs the line of sight from anchor point to the player.

If the line of sight is cut by a collider, it adds a bending point (pivot point) and records the swinging direction (clockwise or counter) of the player. Which in effect means wrapping the rope around the object that obstructs the line of sight.

When a pivot point is added, from there on this component checks the line of sight from this pivot point to the Player. If there is an obstruction while the player swings, it repeats the above mentioned process, again wrapping the rope.

For unwrapping, the script continously looks for a line of sight from Player to the *2nd* closest pivot point. If this line of sight is clear, than it removes the closest pivot point from the list, making the 2nd closest pivot point the closest. In other words, it unwraps the rope.

3-) How to Implement in Your Project

RopeWrapController, RopeShootController and RopeReelController all should be a component of a “rope object”. This object should also have a linerenderer, rgbdy2D and collider2d (for anchoring).

Then you can drag and drop your Player object to player variables in these components. For moving the player you can use the PlayerMovementController component. And you are good to go.

4-) Possible Modifications

Obviously you can use your own controls for moving the player. PlayerMovementController is merely a simple make do for movement.

You can modify the RopeShootController or write you own shooting component. What really matters is the anchor input that this component feeds to RopeWrapController.

If you want to modify the RopeWrapController you should keep the notes given in following and last part of this document.

5-) Important Notes

These notes are pertenent for implementation and modification of the asset.

- RopeWrapController only works on “`RopeWrapLayer`“, which essentially means this component needs and works on a dedicated layer only. And the obstacles that will be wrapped should be on this layer.
- Obstacles that will be wrapped should have Polygon2D colliders. Other collider types will not work and you will get an error.
- Obstacles should have kinematic Rigidbody2D components. You will receive an error otherwise.,
- Both RopeShootController&RopeWrapController scripts make sure that anchor and pivot(bend) points are outside the boundaries of the obstacle that will be wrapped or anchored. If you try to modify these scripts, keep that feature or you will have trouble with Raycast2D. This feature is what is lacking in most of the similar assets. I strongly suggest not changing it for robust functioning.