Taken from tutorial series: https://youtu.be/MNF6QVARlng

In the viewport add a cable. Set the length to 0 and enable collision and set the cable width to 2. Set the collision to custom and set everything except pawn and physical body to block. Uncheck visible.

Extend the event begin play and drag in a reference to cable. Extend it and add a set visibility node. Set the new visibility to false and hook the input exec pin to the output exec pin of the begin play node. Extend the cable reference node and add a set cable length node and set the value to 0. Hook its input exec pin to the set visibility node’s output exec pin. Extend the cable node again and add a set cable width node and set its value to 2. Hook its input exec pin to the set cable length node’s output exec pin. Extend the cable reference and add a set end location node and set the location to the start point of the cable when shooting it.

Say the shooting is triggered by the K key. Add the K key node and extend the pressed pin and add a line trace for objects node. Add a reference to the follow camera and extend it and add a get world location node. Extend it again and add a get forward vector node. Extend the return value pin of the forward vector node and add a vector \* float pin and set the float value to 5000 extend the return value and add a vector+vector node and to the other input pin add the get world location node. Hook the return value to the end pin and hook the value of the get world location node to the start pin of the line trace for objects node. Extend the objects type pin and add a make array node and set the 0th element to World Static and add an element and set the 1st element to World Dynamic. For testing you may set the draw debug type to persistant.

Now during the simulation whenever the grapple is triggered a line trace is seen at the point where the camera looks.

Extend the out pin and add a break hit result node. Extend the output exec pin and add a branch node and hook the condition pin with the hit result pin. Extend the location pin and promote it to a variable called hit location. Extend the true pin of the branch node and add a set actor rotation node. The target must be the character as we want it to face the direction in which the grapple is thrown. Make a Boolean variable called Hooked. Extend the output exec pin and add a set hooked pin and set it to true. Extend the output exec pin of the set hooked pin and attach it to the input exec pin of the set hook location node.

We have just added a set actor rotation node without setting a value to it. To set the value add a get actor location node. Extend the return value and add a find look at rotation node and set the target pin to the location pin of the break hit result node. Split the output struct pin and extend the yaw pin and add a make rotator node – make sure the yaw pin is connected to the yaw of the make rotator node - and hook the return value to the new rotation pin of the set actor rotation node.

But this case works only if the we are not using controller yaw. To make sure the system works even when we are using controller yaw. Extend the true pin of the branch node and add another branch node. Set the condition to use controller yaw and hook the false pin to the set actor rotation node. Copy the nodes that are in the set actor rotation region until the set hooked pin. And paste it above and attach the true pin to the input exec pin of the set actor rotation node. Of course the code now is the same as the false condition but when using controller yaw even the character tilts and that’s what we are trying to avoid. Extend the output exec pin of the set actor rotation pin and add a delay pin and set the delay to 0.7 extend the completed pin and add a timeline node and hook it to the play from start pin of the timeline node. Extend the update pin and add a set actor rotation pin. Add a get actor rotation node and split the struct pin and extend the roll and add a make rotator node and also hook the yaw pin with the make rotator node’s yaw pin. Add a lerp rotator node and hook the B pin with the return value of the make rotator node. Add a get actor rotation node and hook the return value to the A pin. Extend the return value and hook to the set actor rotation node. Hook the alpha pin of the lerp node to the time pin of the timeline node.

Double click the timeline node to go to the timeline editor. We made this to create a simple timeline so as to make the rotation transition smooth instead of snapping between positions. Click on the f button and an external curve is created. Set the time to 0.5. Add a key and set the time to 0 and value 0. Add a new key and set the time to 0.5 and value 1. Right-click on the first key and set the value to auto so as to get a sine-like transition instead of a linear transition. Finally make sure the exec pin of the set actor rotation node is hooked to the set hooked node below.

Now on simulation, the character turns in the direction of the grapple press and if yaw is enabled, it also tilts in the direction but transitions back to the erect position.

Extend the event tick node and add a branch node hook the Hooked variable to the condition pin of the branch node. Extend the true pin and add another branch node. Create a new variable called hook move finished and hook it to the condition pin of the branch node.

Make a function called Move Cable. Open it and add a reference to the cable and extend it and add a set visibility node. Hook its input exec pin to the exec pin of the Move Cable. Extend its output exec pin and add a branch node. Extend the cable node and add a get world location node extend the return value and add a vector - vector node. To the second pin hook the Hook Location variable. Extend the return value of the get world location node and add a VInterp to node. Set the target pin to the hook variable. Extend the delta time pin and add a get world delta seconds node. Set the interp speed to say 10. Extend the return value of the vector – vector node and add a vector length node. Extend the return value and add a float<=float node. Set the 2nd value to 100. Extend the return value and hook it to the condition pin of the branch node. Extend the false pin and add a set world location node and set the new location to the return value of the VInterp to node. Set the target pin to the cable node. Extend the output exec pin of the set actor location node and add a return node. Make an output for the return node of Boolean type called Reached Location. Make sure that in the return node the Reached Location value is false. Extend the true pin and add a return node. Here set the value of Reached Location to true.

Go back to the event graph and extend the false pin and add a move cable node. Extend the reached location pin and add a set hook move finished node.

Now on simulation the rope is created at the point the person looks but it moves with the person too. To prevent this we need to update the location with each move and turn of the player.

Extend the true pin of the branch node and add a sequence node. Extend the then 0 pin and add a set world location and rotation node and set the target as the cable node. Extend the new location and add a VInterp to node. Extend the cable node and add a get world location node and hook its return value to the current pin of the VInterp to node. Set the target pin to the Hook Location node, the delta time pin to the get world delta seconds node and the interp speed pin to 250. Extend the new rotation pin and add a find look at rotation node. Extend the start pin and add a get actor location node and set the target to the player character. Extend the target pin and add the Hook Location node.

The above paragraph moves the start location of the cable according to character movement. Now on simulation, even on movement the cable remains attached to the surface.

Create a new event or function for stop grapple, extend the exec pin and add a set hooked node and set the value to false. Extend the output exec pin and add a set hook move finished node and set the value to false. Add a reference to the cable node. Extend it and add a set visibility node and set the visibility to false. Hook its input exec pin to the hook move finished node’s output exec pin. Extend the cable node and add s set world location node and hook its input exec pin to the set visibility node’s output exec pin. Extend the new location pin and add a get actor location node and set the target to self. This is to cancel the grapple. You might have to manually set a grapple location instead of the get actor location if the desired result isn’t obtained. Extend the cable node and add a set end location node and set the node to the initial default value whatever it was.

Go to the branch node connected to the return value of the look at object node. Extend the false pin and add the stop grapple node. This is to call the stop grapple node if the tracer doesn’t hit anything.

Now on simulation on activating cancel cable the cable disappears and on firing appears at the new location again.

As of now, the cable is always tense when fired. To make it appear loose and to have a few extra lengths after fire you could follow these instructions.

Extend the set world rotation and location node and add a do once node. Connect the reset pin to the end of the stop grapple definition i.e., to the set world location node of the stop grapple definition. Add a reference to the cable node and extend it and add a set cable node. Extend the completed pin of the do once node and hook it to the input exec pin of the set cable length node. add a get actor location node, extend the return value and add a vector – vector node. to the second pin add a hook location node. Extend the return value and add a vector length node, extend it and add a float + float node and set the second value to say 50. Hook the return value to the cable length value. Extend the output return value and promote it to a variable called new cable length. Extend the cable reference node and add a set end location pin and set the value to the default end location whatever it is. Hook its input exec pin to the output exec pin of the set new cable length node.

Now on simulation the cable is no longer tense. It might look like a line breaking instead of getting curved when the character moves but that’s because of the number of segments used. The greater the number of segments, more is the curvy motion.

If you want you can break the rope if it becomes to tense.

Extend the Then 1 pin of the sequence node and add a branch node. Add a get actor location node and extend the return value and add a vector – vector node. To the second pin add a hook location node. Extend the return value and add a vector length node, extend its return value and add a float >= float node and to the second pin extend it and add a float + float node and to one pin hook the new cable length node and to the other set the value to say 400. Hook the return value of the float >= float node to the condition pin of the branch node. Extend the true pin and add a Stop grapple node.

Now at each iteration it checks whether the cable becomes too tense and then breaks it. But this works only if the cable is loose at initial fire as it gives the users the idea that the cable breaks when it’s tense. But if the loose cable functionality isn’t implemented due to the possibility of varying cable fires the result will look glitchy because on one iteration the cable could go for a long distance but in the next iteration if you fire the cable at a short distance and move back it could snap thereby making the logic hard to understand.

On simulation the cable breaks when it becomes too tense.

Finally, to implement a grapple we need to launch the player at the press of a specific button.

Create a function/event for the move player when pressed functionality. Extend it and add a branch node. Let’s say I use the G key to trigger it. Make a Boolean variable called G is pressed. Add a G node and extend the pressed pin and add a set G is pressed pin and set the value to true. Extend the released pin and add a set G is pressed node and set the value to false. Set the condition pin of the branch node to G is pressed variable. Add a reference to the cable node. Extend it and add a set cable length node. Hook its input exec pin to the true pin of the branch node. Add a get actor location (player) node extend the return value and add a vector – vector node. Set the second pin to the Hook Location variable. Extend the return value of the vector – vector node and add a vector length node. Hook the return value to the cable length node.

Create a function called move player extend it and add a launch character node and set the target to the player. Add a get player pawn node, extend the return value and add a get actor location node. Add a hook location reference node, extend it and add a vector – vector node. To the second pin hook the return value of the get actor location node. Extend the return value of the vector – vector node and add a vector \* float node. Extend the float value and add a float \* float node. Set the first pin as the get world delta seconds node and the second pin value as say 250. Hook the return value of the vector \* float node to the launch velocity pin. Enable the X-Y override.

Go back to the move player when pressed definition and between the true pin and set cable length node add the move player node. In the event tick region after the first branch node to which hooked variable is set as condition, extend the true pin and add the move player when press G node.

Taken from tutorial series: https://youtu.be/s3KUmwc7zwQ

One way to keep the character in the orbit of the hook is to add a force at every tick.

Add a character movement node, extend it and add an add force node and hook its input exec pin to the output exec pin of the event tick node. Add a hook position variable node, extend the return value and add a vector – vector node(it must be attached to the 2nd pin). Extend the other pin and add a get actor location node (player). Extend the return value of the vector – vector node and add a dot product node. To the other pin hook the get velocity (player) node. Extend the return value of the vector – vector node and add a normalize node. Extend its return value and add a vector \* float node and to the second pin hook the return value of the dot product node. Extend the return value and add a float \* vector node and set the value to say -2.0 and hook the result to the force pin of the add force node.

Now on simulation the player is in orbit of the hook.