Support Items Tutorial

This tutorial will be specific. The mechanics will be similar to those from the franchise games called Mega Man. The mechanics I'm refiring are the rush coil and the rush jet. The first lets you jump higher while the second is used to hover in the air. In addition, I will be using an already made first person player script which should not be an issue as it should be able to work with most player movement scripts. Also, the mechanics will be mostly used for a 3D game and not a 2D game.

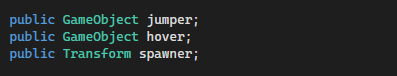
If you want to use this tutorial, then at least you should know a bit of the basics with unity and how to write scripts.

The first thing that needs to be done is to create two game objectives in different colour materials. Each will be used for each mechanic. In addition, create an empty game object and parent it to the player but have it away from the player as it will be used to for creating the first two objects, and finally duplicate the object which will be used for the hover mechanic and attach it to the player and have it be at the bottom of the player model or box collider.



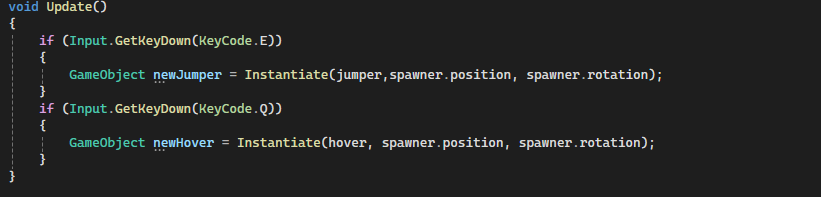
Once all game objects are made create four different c# scripts. The first script will be used to spawn the game objects, the second will be used for the jump support mechanic, the third will be used to activate the fourth script and some lines of code which will be added to the player script, and the fourth will be used to turn on or off some components and other variables in other scripts Aswell.

On the first script add these variables which will be used to connect the game objects to the script and have the location of where to create them.



After that in the update function add these two if statements and the lines of code as seen in the image below. The if statements have code which checks if the correct key buttons are being pressed down. While the two lines are used to create the two game objects in front of the player by using the empty game object position and rotation.

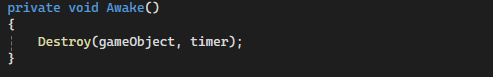
Once the code is written you can attach the script to the player.



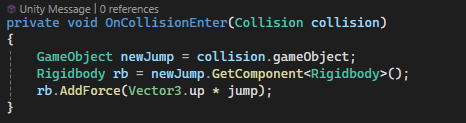
Now go to the second c# script and add before entering visual studio attach it to the jump support game object. Once that is done in the image below add these two float variables. The first will be used for how high the player will jump, and the second will be used as the timer on how long the object will be in the game after being created.



Then create a private awake function which will have this line of code. The code will simply destroy the game object once the time has passed.



After that create another private function but this time it's for when the game object has another game object enter its box collision which can be seen in the image below.



Then inside of the function as seen in the image above add these lines of code. The first and second are used to first connect to the game object which has collided with the jump game object and then get the rigidbody componet which will be used on the last line of code. And then in the last line of code in the function it will move the game object into the air by pushing it using force.

Once the jump script is done open the next script and first add a variable which will be used to connect to another script.



Once written add one more variable which will be used for the same thing as the jump script.



Then after writing all the variables for the script make an awake function and write two lines of code. The first as seen in the image below is used to find the object with the exact name and then get its script. While the second line which I don't need an image as its the exact same as in the jump script so you can duplicate it.

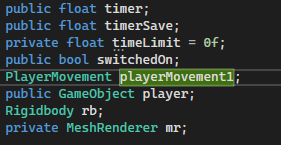


Then like with the jump script make a collision function but instead add these two lines of code as scene in the two images below. The first uses a bool variable in the last script file so once there I will explain this line. while the second is just to destroy the game object.

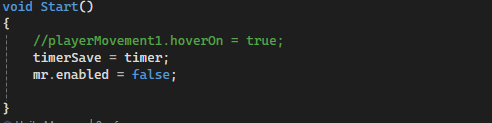




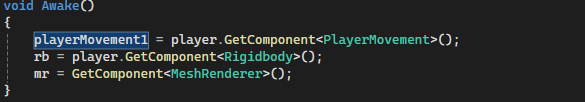
Now if you get an error about the second to last line of code it's fine since we are now going to work on the last script which will fix it. First make sure the last script is attached to the duplicate game object. Once all of that is checked create all of the variables seen in the image below. The first three variables are all connected to how long the player can be in a hover mode. The next variable used as a switch which the previous script has it be made true when the player collides with the hover game object. The next variable is used to connect to the player movement script while the other is used to connect to the player object. The last two are used to allow access to the two components which are the rigidbody of the player and the mesh renderer of the duplicate game object.



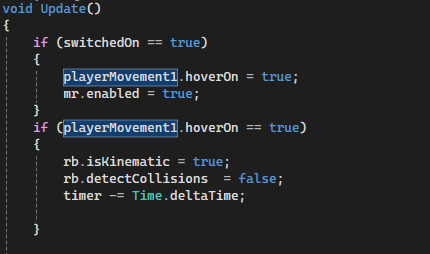
In the start function add these two lines of code which will copy the time you put for the timer variable in the inspector while the other will turn off the mesh of the duplicate game object.



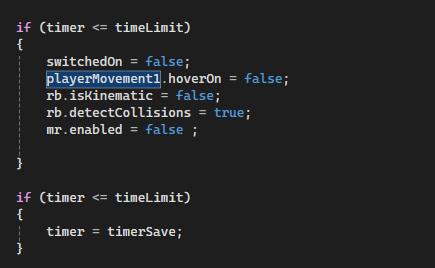
Then make an awake function and add these lines. They are all used for similar reasons which I already explained when telling the used for the variables.



In the update function add these two if statements. The first is used to turn on a bool which will later be added to the player script and the second is to turn on the mesh of the duplicate object. The second is used to turn on and off some parts of the player rigidbody and to start decreasing the timer every second.



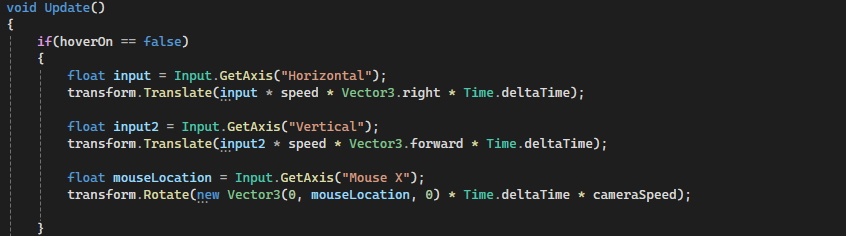
After adding those two add two more if statements. These check if the timer is less than the time limit which is zero. The code has to be like this as any other way will not work.



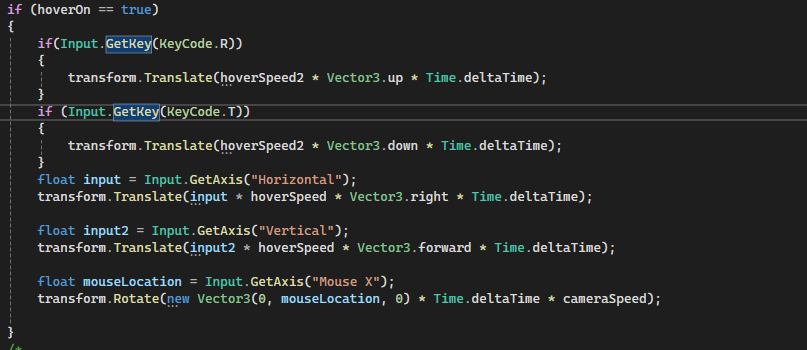
Inside the first if statement as seen in the image above add all those lines of code which are reverting everything back to how it was before. Then in the last if statement add that one line of code which simply reverts the timer to how it was at the start.

Now once you are finished there is one last thing that is needed. On the player script add the one bool which I mentioned before. Now in your movement code copy all of it and add them inside two if statements. In the first statement it checks if the bool is false the code stays the same. While in the second will have a different set of code if the bool variable is true.





In the second statement have these lines of code. These lines of code are used to move the player up or down depending on the speed set. In the image I use a different float to my player normal speed but if you want to also do that then make sure that you also replace player speed variable with a different speed variable in the in the player movement code in the second if statement as seen in the image below.



References:

There are no references in this tutorial as everything I did was with the knowledge from the other tutorials.