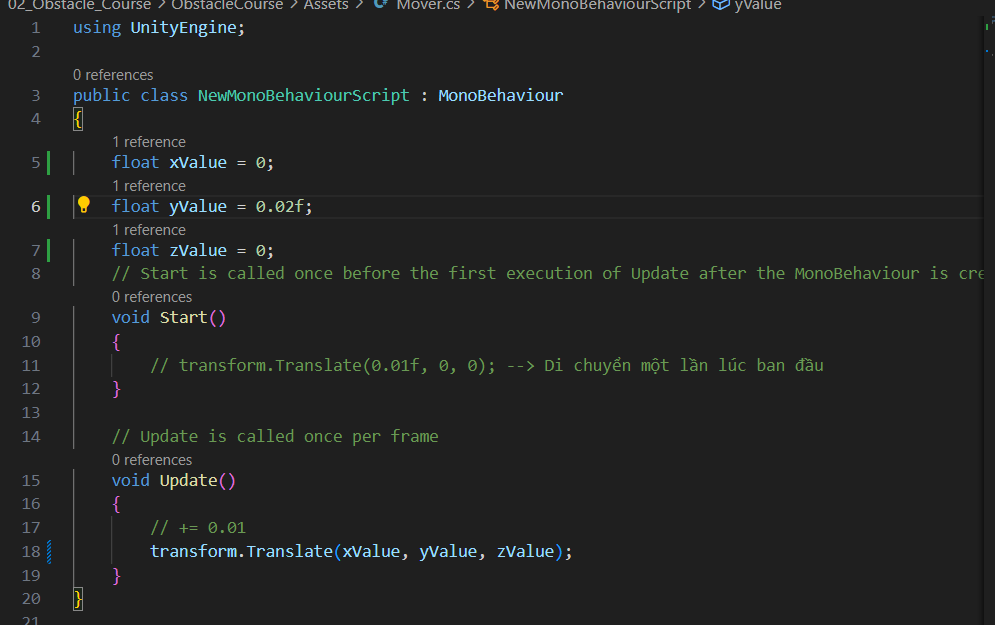
* Obstacle Course
  + Player experience
  + careful Clever Nah
  + Nimble / agile
* Core mechanic
  + Move & Dodge Obstacles
* Game loop
  + get from A to B
* A quick, Silly Challenge
  + Give your Player a name
  + mine is going to be "Dodgy"
* ***Get Started***
  + Create a new Unity 3D project.
  + Add a ground Plane
  + Create your “player”
  + Rename your player
* ***Variables Are Like Boxes***
  + Variable help us store(Lưu trữ), manipulate(Thao Tác) and refer(tham chiếu) to information
  + Each variable has a NAME
  + Each variable contains DATA
  + Each variable is of a particular(Cụ thể) TYPE

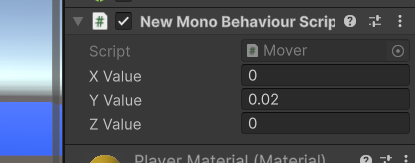
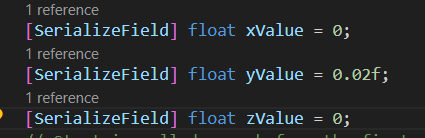
Exercise: create two more variables

Create a variable for the y value and for the z value.

Change the values so your player flies straight up in the air.



* ***Serialize Your Other Variables***
  + Make all three of your variables serialized and therefore accessible in the inspector
  + While in play mode, make your player run away from the camera and then run back to the camera

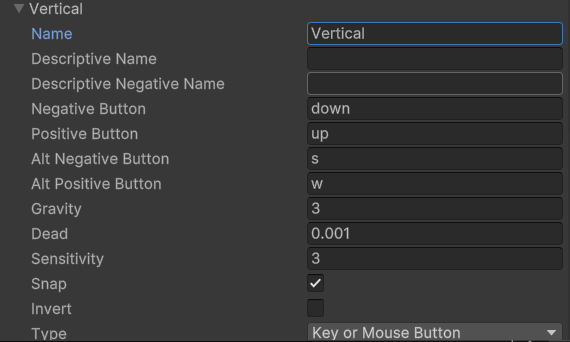
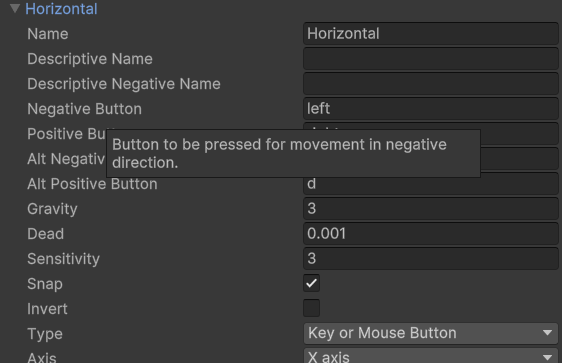
 → Điều chỉnh được

***Edit*** → Project Setting → Input Manager → Horizontal: Alt Negative Button: a

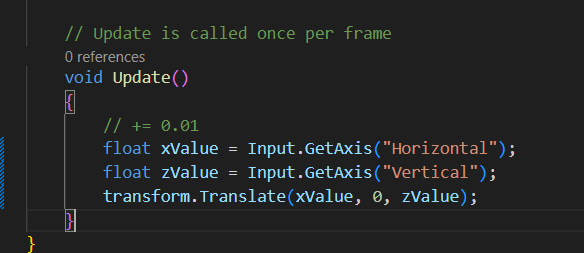
Alt Positive Button: d

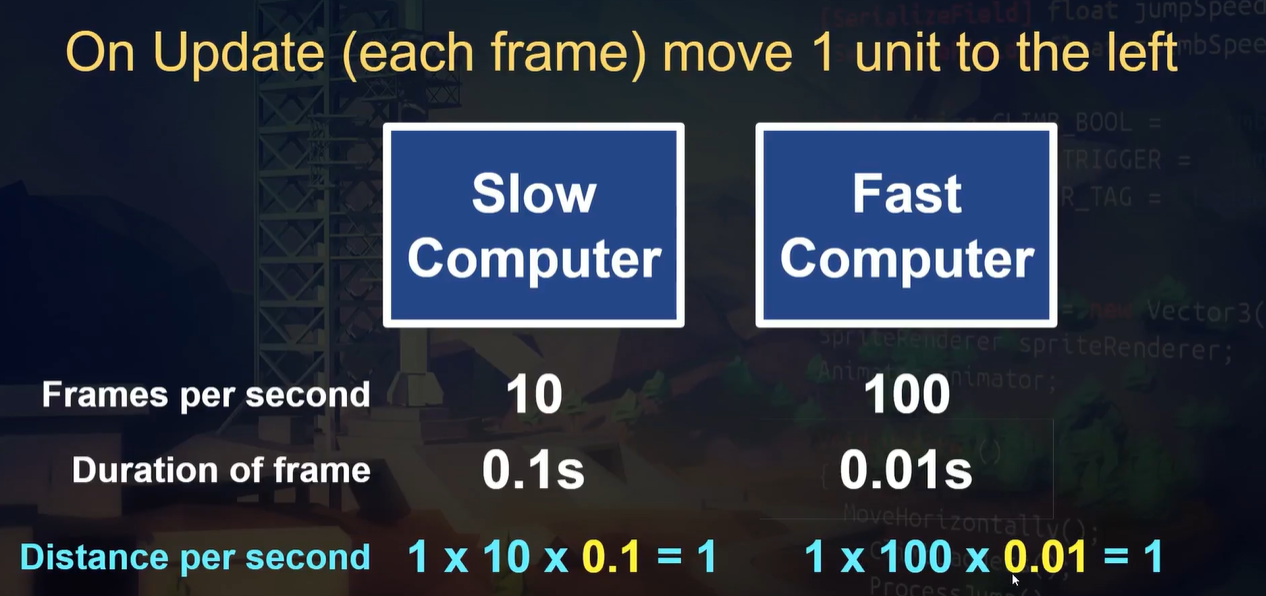
→ Vertical: Alt Negative Button: s

Alt Positive Button: w

* ***Add Vertical Axis:***

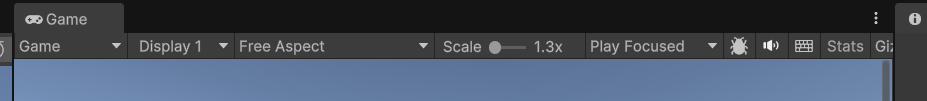
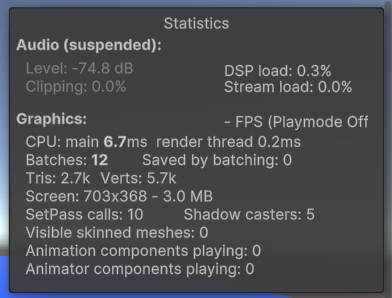
Update one of our variables so we are moving our player forwards and backwards (along the ground plane, not flying in the air)

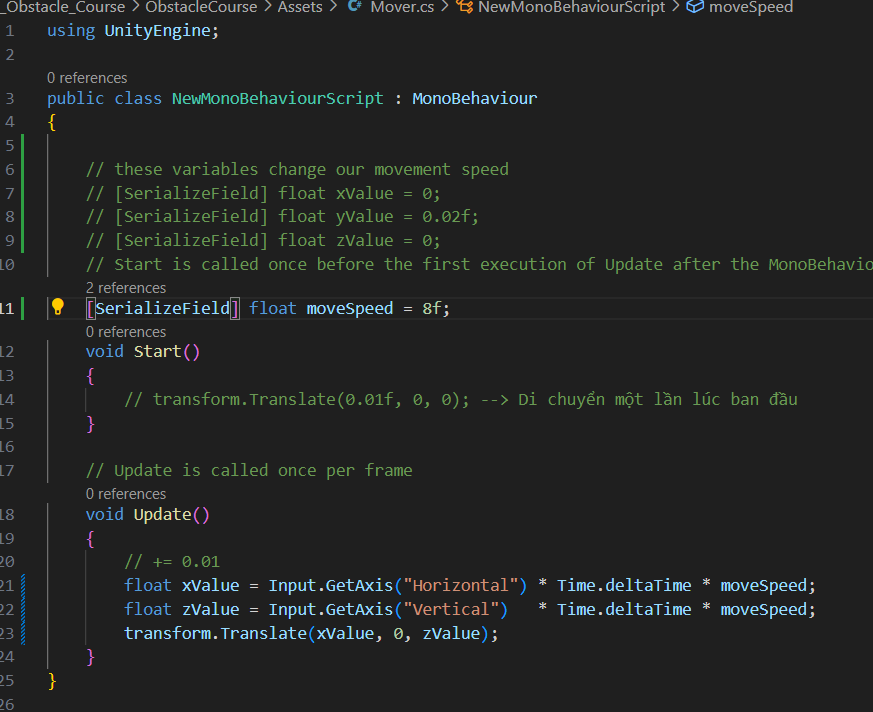
* ***Using Time.deltaTime:*** 
  + Using Time.deltaTime Unity can tell us how long each frame took to execute
  + when we multiply something by Time.deltaTime it makes our game "Frame rate independent"
  + Ie. The game behaves the same on fast and slow computers
  + Note:

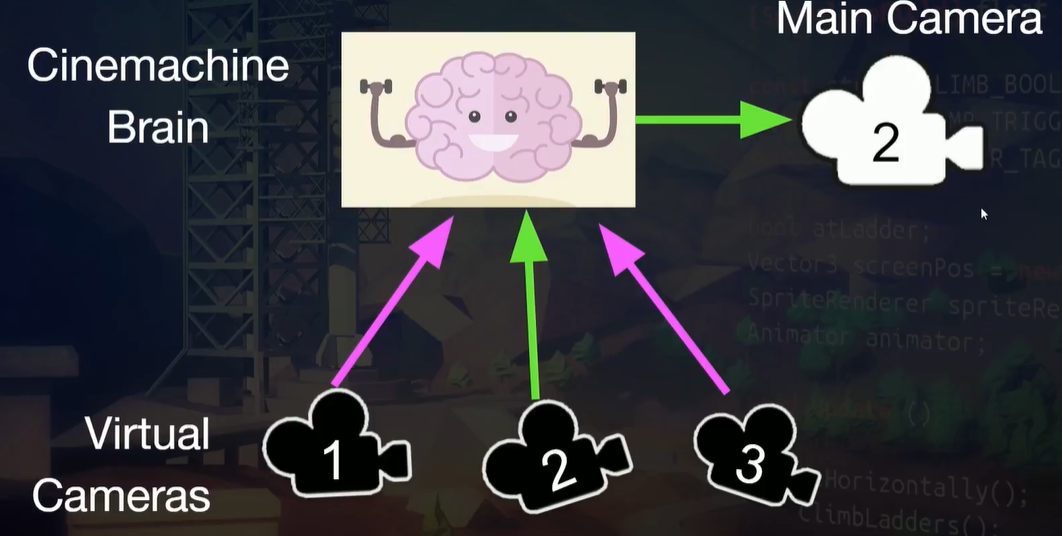
Frames per second: Số khung hình trên giây

Duration of frame: thời lượng khung hình

Distance per second: Khoảng cách mỗi giây

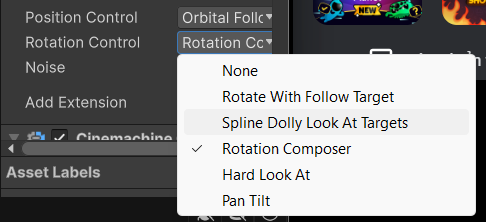
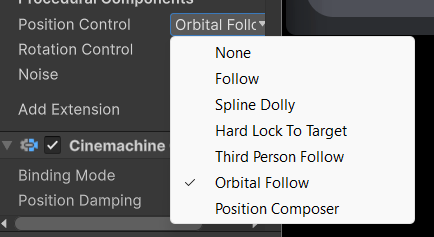
*  → Chọn Start
*  → Bảng thông tin
* ***Multiply By A Speed Variable:***
  + Create a new variable called moveSpeed. The value of moveSpeed does not need to update each frame.
  + Make it available in the inspector.
  + Multiply your xValue and zValue by moveSpeed.
  + Tune your player movement (as best you can for now).

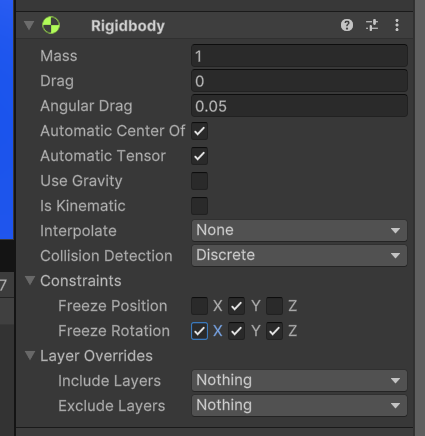


* ***What Is Cinemachine?***
  + Cinemachine is a powerful package (Gói mạnh mẽ) that lets up:
    - Manage(quản lý) multiple cameras in our scene.
    - Easily create rules for our cameras.

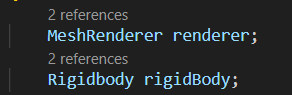
\*\*\*Window → Package Manager → Unity Registry → Cinemachine → Install\*\*\*

* ***Add A Follow Camera*** 
  + Install Cinemachine from Package Manager
  + Add Cinemachine Brain component to camera
  + Rename Virtual Camera
  + Point the Virtual Camera to follow the player
  + Tune distance

 → Freelock Cam (+Mouse)

* ***Make some walls***
  + Add some walls around the outside of your play are
  + If you like, make them a different colour
* ***Add Component Rigidbody → Player:*** 
* ***What are methods*** 
  + Methods (also called Functions) execute blocks of the that makes our game do things.
  + To achieve this we must:
  + 1. DECLARE and define our methods
  + 2. CALL out methods when we want it is execute
* ***But What About Start() & Update()***
  + We’ve been defining Start() & Update() but not calling them
  + Where are they called?
  + Unity’s internal logic is taking care of calling them for us the right time
  + These are referred to as “callbacks”
* ***Get Set Up***
  + Create a new C# script called ObjectHit
  + Attach that script to all 4 walls
* ***Using get Component***
  + GetComponent<MeshRenderer>().material.color = Color.red;

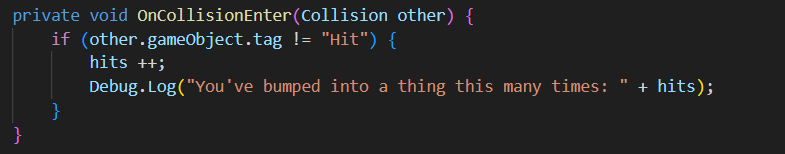
MeshRenderer: Thành phần dùng để hiển thị đối tượng 3d trong Scene

* ***Prepare For Score Feedback*** 
  + Create a new C# script called Scorer
  + Create a new OnCollisionEnter() method
  + When we hit something, print to the console, "You've bumped into a thing this many times: "
* ***Problem & Solution*** 
  + **Problem to solve:** 
    - Make an object fall after 3 seconds has passed
  + **Solution** 
    - A Timer - Time.time
    - A mechanism(cơ chế) to "do a thing if 2 seconds has elapsed(trôi qua)" - if statement
    - A way(Cách) to start the object falling after 3 seconds - disable / enable gravity
* ***Print Out Time Elapsed***
* On every frame, print out to the console how much time has elapsed since the game started
* Hint: Use Time.time within debug.log
* Add your script to a game object(?)
* ***Simple definition***
  + Caching is a technique of storing frequently used data/information in memory, so that it can easily be accessed when needed-

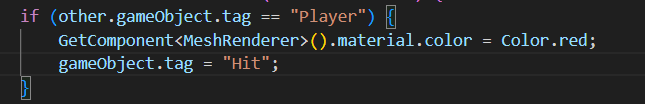
-

***Make it Fall***

* Add 2 lines to our code that make our object become visible and fall after it has waited for the right amount of time

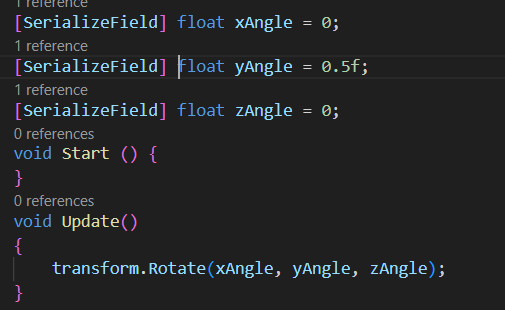


Scorer



ObjectHit

* ***Rotate An Object:***
  + SerializeField 3 variables and pass those in to our Rotate method so that our object spins.



* **Build your Obstacle Course** 
  + Create a fun layout that makes the player go from A to B
  + Use your dropper, rollers and spinners to create interesting moments for the player

