Player Controller Plan

Aim :

* To build an all round controller for third person games which can traverse over complicated terrains and be extendable enough to be used for custom purposes such as ledge climbing.
* Clean and efficient code that can be reused when needed and variables can be edited so the motor is what will be moved while the controller and other scripts can control the movement.
* The ability to make this an all round controller for the audience who will want to learn about how this type of mechanisms work.

Expected Base Features :

* Simple Movement
* Slope Handling
* State Machine
* Collision + Multi Edge Detection
* Artificial Damping *based on surface normal*
* Climbing Stairs
* Terrain Movement and Proper Lerping
  + Sticking to ground while gliding over fractures
* Smooth Frame Independent Movement
* Tight Space Prevention
* Collision saving
  + Building custom collision system that checks for the local objects in a radius and only checks collision within those bounds, extremely optimised version.

Expected Complex Features :

* Slope Slipping
  + Realistic (Not Instant Slip)
* Jump Holding power
* Crouching and Sprinting
* Sliding and Collider Size Changing for going under logs etc…
* Strafing
* Ledge Climbing (Proper Prediction System based on Input)

There are two aspects of the controller :

* Player Controller
* Player Motor

**The controller** is what will actually receive input and determines how that input is used, the controller is what will have the actually state machine inside it always calculating the current state of the player based on the environment and input. Then the inputs and movement will be conducted differently.

**Player Motor** will be the system that takes the input and just does what the input tells it to do, it needs to be very flexible with many exposed variables and functions for any use case. So that even non-player controller scripts can change the variables if need be.

Different States:

These are the base movement states :

* Crouching
* Strafing
* Jumping
* Landing
* Sprinting
* Walking
* Still
* Falling

These are the base control states :

* Aiming
* Shooting
* Attacking
* Climbing
* Sliding

Motor:

* Needs to be frame independent so that some devices don’t run the player faster or slower than others.
* *Needs a controlling script at that point, because a ladder can have it’s own script that controls the player when it is on that ladder. So the motor will always have a controlling script that can take control from the player controller at any point – when this happens any input given from player controller script will be ignored until that controlling script is removed…*

Different Controllers –

**Base Movement Controller**

**Ladder Movement Controller**

**Ledge Controller**

How will the controller be built from scratch?

* First the motor and the base controller will be built keeping in mind the future use of controllers so that the one controller doesn’t take up all the control for the motor.
* The motor must have all of it’s controlling variables exposed and methods must be exposed as well.
* Everything from the collision to slopes must be called through the controller at hand, because for example while climbing a ladder there is no use of collisions and when climbing up the side of a mountain we don’t want collisions. We need everything to be procedural.

Will there be use of any assets?

* Not for the mechanics of the player unless there are some good collision too
* **I will also probably buy FINAL IK kit because we want the most optimised script, there is no shame in buying assets.**

**Collision Detection –**

The rigid body system in Unity is actually very sophisticated if used correctly, I will be using the continuous option for the player in Unity as well as the arrows the player will shoot, along side with a on collision enter system to do further checks if the collision actually had a reaction or not.

Integration of animations?

* Once the model is built I will be using the inbuilt mechanism state machine controller which is the best for blending animations (blend trees)…
* **I will also probably buy FINAL IK kit because we want the most optimised script, there is no shame in buying assets.**
* You will need robust IK for ledge climbing etc…

How will ledge climbing be done?

* Each ledge wall will essentially be a network of nodes, there are the base nodes which are the nodes that the player can connect to when they are close to it, the neighbour nodes which each node has and the player can travel to each one by moving in that direction. Then the end nodes which represent him moving off the ledge.
* Jumping off the ledge will be a procedural movement controlled by an arc if it is a large distance apart or if it is a small distance apart player will just move there, but arms and legs will be controlled by IK during this period to give a realistic look.

**PHYSICS**

* Collision will be based on dynamic Rigidbodies while reporting collision flags back to the main script, which will do an overlap, check every frame.
* Climbing up slopes is done through future interpolation and getting a point in the forward direction and moving towards it, but the problem with interpolation using Vector3.Lerp is that it slows down near the end which results in unrealistic movement, so you must find the difference in the positions and find a direction and move towards that direction at a constant speed.