Productive Bludgers Melee Fighting Game Action System Overview

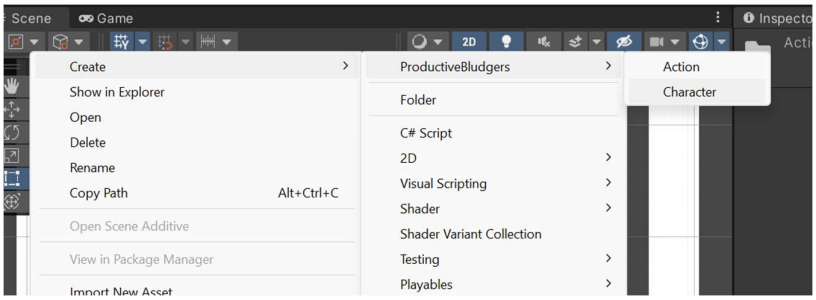
I have created the action system for this game with the intention that team members can create their own character move sets without coding knowledge. It still consists of typing in a bunch of numbers, so is not the most intuitive system in the world, but my hope is that it allows for some experimentation with game mechanics without the need for programming knowledge.

Character and Action objects can be created in the unity editor. The character object has several “Input" variables which store a list of actions, multiple actions are added to one input either to create complex moves or to trigger different moves based on the player state (Currently this is just either on the ground or in the air, but more options could be added later).

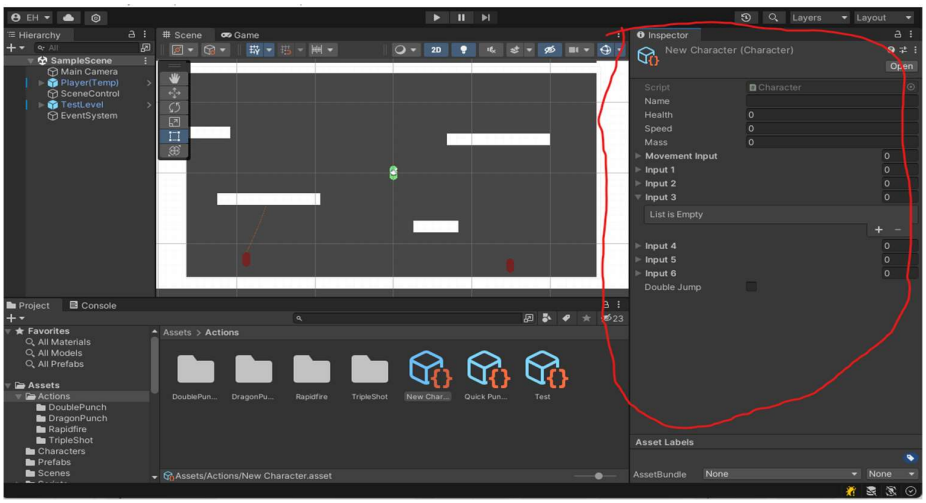
Right now pretty much all values are determined by Unity defaults so are obviously subject to change in order to fit the needs of the project

Creating a Character

To create a new character right click in the project window inside unity then select Create->ProductiveBludgers->Character from the menu that appears.



Name the character something appropriate, the take a look at its values in the Inspector window on the right (assuming default Unity editor layout).



Character Properties

Name:

The name of the character. Currently this is not implemented and does nothing.

Health, Speed and Mass:

These are also currently not implemented, but will eventually determine the amount of damage a character can take, the movement speed and the amount of knockback they sustain when taking hits (along with the knockback value on the attack itself).

Movement Input:

Not implemented right now. Currently the “Movement Action” Input will cause the character to do a dash, but the intention is for this to be customisable in the same way that other Inputs are. This field exists for that purpose.

Input 1 – 6:

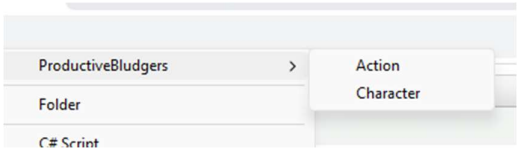
These store a list of the actions that will take place when the associated button is pressed on the controller/Keyboard.

Double Jump

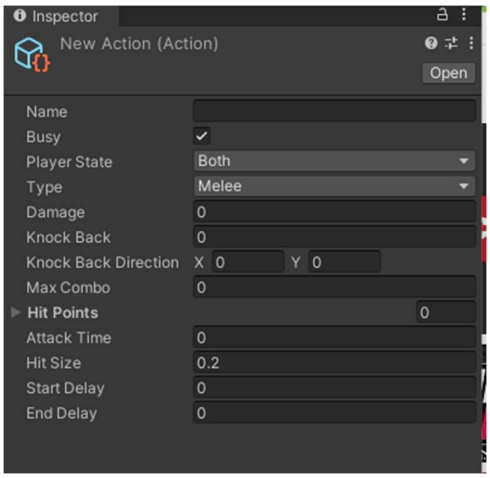
When this is enabled the character can perform one additional jump when not on the ground.

Creating an Action:

More or less the same a s creating a character, right click in the project window and select Create->ProductiveBludgers->Action.



Now name the action and observer the inspector window again, this time the window will appear different depending on the options selected.



Name:

The name of the action, this does nothing and I'm actually not sure it will find a use so it may be removed in future.

Busy:

When the is enabled the player is placed in a "Busy" state for the duration of the action and they will not accept any further inputs, it should be enabled in most cases.

Player State:

This specifies a state the character must be in for an action to take place, current options are "Airborne", "Grounded", or "Both". If "Grounded" or "Airborne" are selected and the player character is not in the corresponding state when the action is triggered, there will be no effect.

Type:

The Type of action, Option are "Melee", "Projectile", and "Movement". The Option selected will change the other available variables.

Start and End Delay:

Start and end delay fields are present for all action types. If the busy field is enabled the character will not accept any input during these times (as well as for the duration of the attack itself). In many fighting games when an attack is triggered, an animation may play for several frames before a hit can land (perhaps a character moves their arm above their head before swinging downwards – the actual hitbox for the attack is not present for the first part where the arm is moved upwards). This allows for this space before and after the attack part of the animation, or if the action is being combined

with other actions, it allows for it to start at a certain time withing that sequence of actions. End delay is also important from a gameplay perspective to provide an opponent with a window to attack in the case that a move misses.

Attack Time:

This field is present for Melee actions, as well as for the "Dodge" movement action. This is the duration of time the actual effect of the action will be active.

Damage:

This field is present for Melee and Projectile actions, this is the amount of damage that is applied by the attack for a single hit.

Max Combo:

This field is also present for Melee and Projectile actions. This is the number of times an attack can possibly hit. Any value 0 or below is treated the same as a value of 1. When an attack lands the hitbox is disabled, but in the case that "Max Combo" is greater than 1 it will be enabled again after a short time* until the specified number of hits have landed (or the Attack Time is completed).

** The time between hits is currently hard coded. I may add this to the action object so that it can be modified at a later date, however I feel if this number is not consistent across attacks gameplay may feel odd, so adding it may just be introducing additional complexity unnecessarily.*

Knock Back:

The amount of force that is applied to push the target of an attack.

Knock Back Direction:

The direction that a target is pushed by the Knock Back force. For this, and all other directional(Vector) values, positive X is the direction the character is facing at the time of the attack, and positive Y is up. In most cases this would be 1,0 (away from the character on the X axis).

Projectile Prefab:

Used for projectile actions. This is the actual projectile that is spawned (it requires a prefab object containing the projectile.cs script).

Projectile Start:

Used for projectile actions. This is the position relative to the character (+X forward, +Y up) that a projectile is spawned in.

Projectile speed:

Used for projectile actions. The initial speed of the projectile.

Direction:

Used for projectile actions. This is the direction the projectile will travel once spawned, options are:

- **Facing Direction** - The projectile will travel "forward" according to the direction the player is facing
- **Input Direction** - The projectile will travel in a direction dictated by player movement input
- **Custom Direction** - In the case of custom direction a "Custom Direction" field will appear allowing for a specified direction.

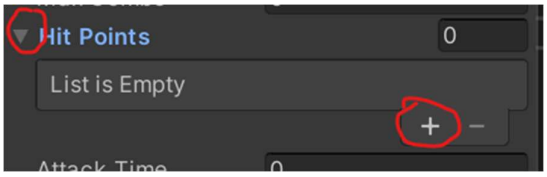
Movement:

This field applies to Movement actions. This selects the type of movement, currently the options are “Add Force”, “Move To Point”, and “Dodge”.

- **Add Force** – this applies a force to the character, the force is expressed as a single Vector with an X and Y value for left/right and up/down. So for example to push the character up you would place a positive value in the y field, the larger the value the more force applied in that direction. There is no time scale for this, the force is applied all at once immediately on completion of the “Start Delay”.
- **Move To Point** – This will move the character to a point in space relative to their current position. This takes a destination and a speed input at the moment and the relative field will appear, however the current implementation is problematic and this should change. One of the changes I intent is to remove speed and instead use “Attack Time” to specify the total time for the movement to take rather than the speed for it to occur at.
- **Dodge** – This will place the player in an “invulnerable” state where they will not collide with attacks or other players for the duration of “Attack Time”. Please note the code will place the player in this state at the start of the timer and the take them out at the end, this means if two dodge actions were to overlap the first one to finish would effectively cancel out the remaining time on the other.

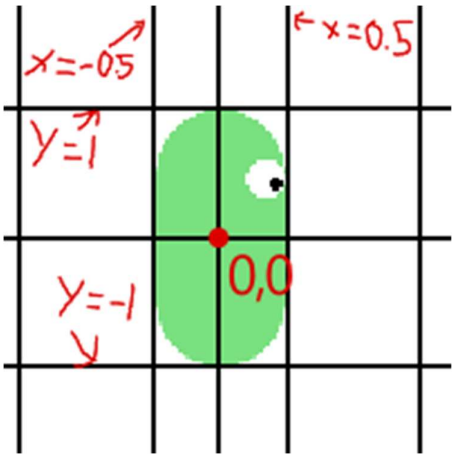
Hit Points:

This applies to Melee actions, and there is a reason I left it until (almost) last. This dictates the position of the hit box for the attack, and will take a list of positions, requiring at least one in order to work. To add a value, first expand the list by clicking the arrow to the left of “Hit Points”, then click on the “+” icon to create a new value.



With more than one vale entered the hitbox will move along a path starting from the first value and finishing at the last, this will happen over the time specified in “attack time”.

As with the “Projectile Start” value, the positions are relative to the character, with 0,0 being the centre of the character model and positive X being the direction the character is facing. As it is setup now the current placeholder character is 2 units tall and 1 wide. This means the top of the character is y=1 and bottom is y=-1, and the sides are at x = +/- 0.5.



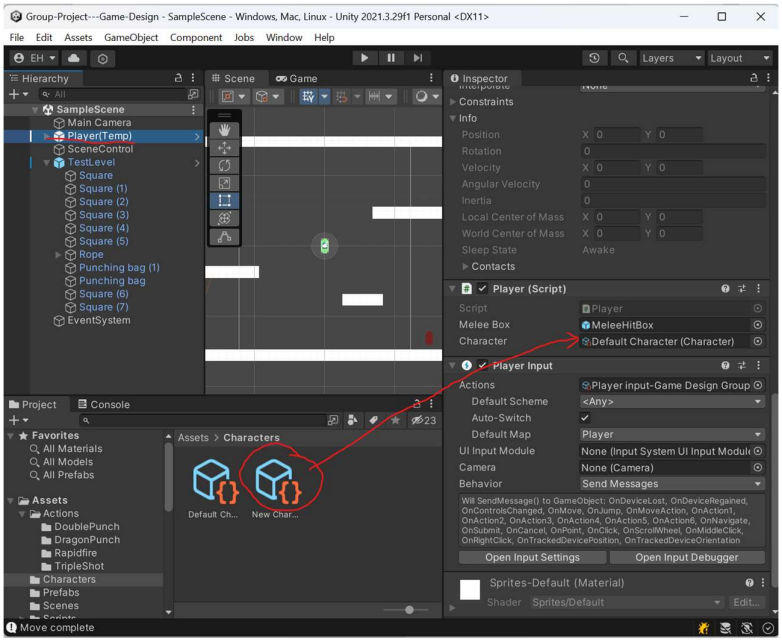
So for example, a melee attack to match a short punch animation straight in front may start just before the front edge of the character at (0.4,0), then extend further along the x axis, say to (0.9,0).

Hit Size:

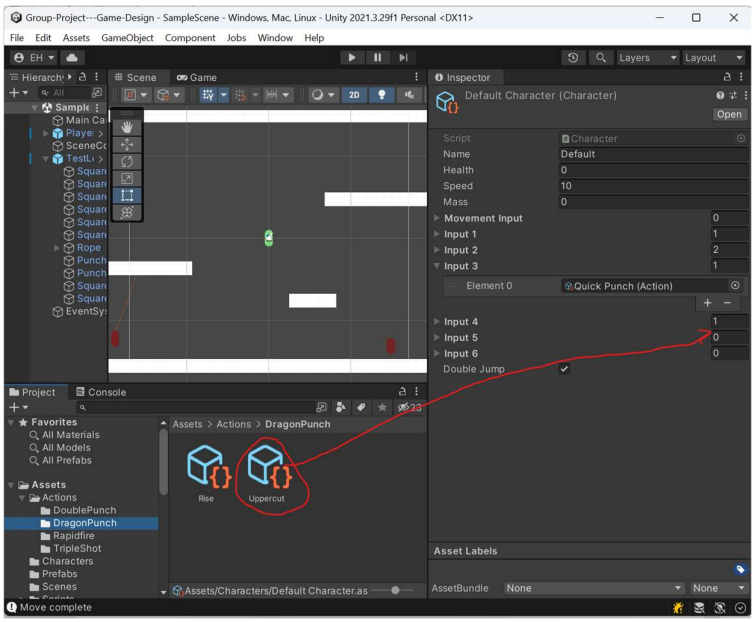
This is the radius of the hit box (hit circle really), the distance from the hit point specified as above that a hit will register.

Setting up the Created Character and Action Objects

As it is right now there is no character or player select screen. To use the created character you will need to select the “Player(Temp)” object already placed in the scene and drag your character into the Character field inside its Player script.



Once an action has been setup it can be dragged onto any of the “Input” fields in a Character object.

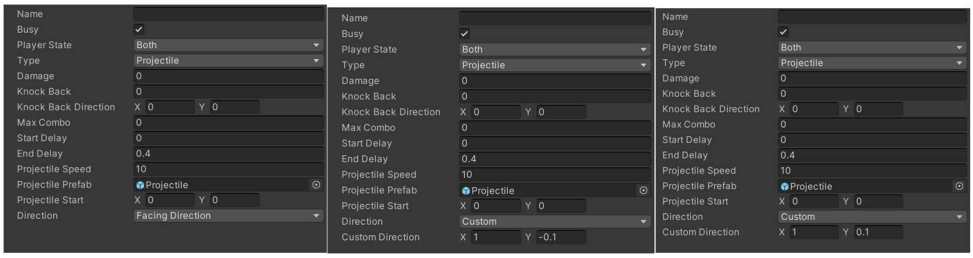


Adding multiple Actions on one input

Multiple actions can be added to one input. This allows for different actions to be triggered by the same input depending on player state, or for multiple actions to be triggered together for more complicated moves. Some examples of this Include:

Triple Shot:

Three projectile actions, each with no start delay and the same projectile prefab (this is just Unity's default circle sprite + the projectile.cs script). But the projectile direction is slightly different for each, one being "Facing Direction" and the others being X: +1, Y-0.1 and X+1,Y+0.1. this results in the projectiles spreading out as they travel forward from the player.



Dragon Uppercut:

An attempt to replicate the Ken/Ryu uppercut from Street fighter. This one uses 5 hit points to define the hitbox path (<0.5,-0.4> <1.3,0><1.5,0.8><1.7,1.6><1.7,1.6>). It has the same value twice at the end, causing the hitbox to be stationary (relative to the player) at the end of its path for a brief instant. It also has a “Move To Point” X:0 Y:+5 action causing the player to move up by 5 units. The movement action is on a longer start delay that the melee one, so the “punch” begins before the player moves upwards.