Dynamic Combat System

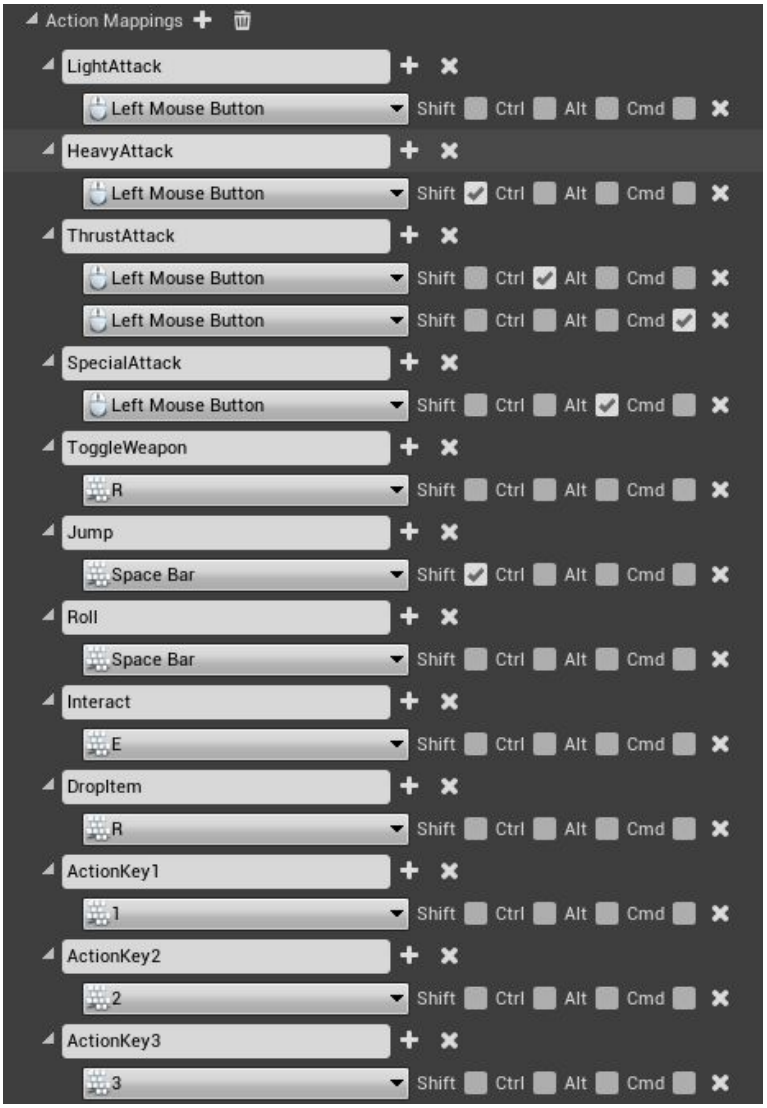
Contact email: szewczykgrzesiek@gmail.com

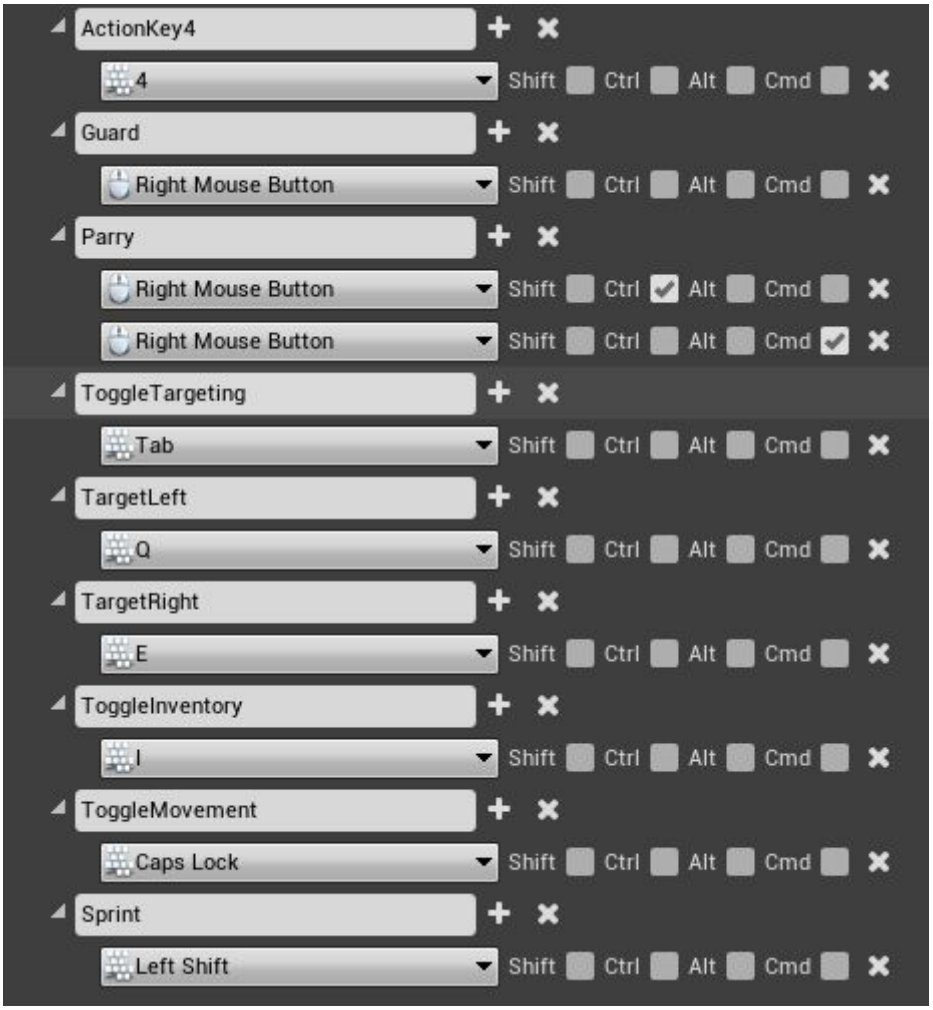
Player

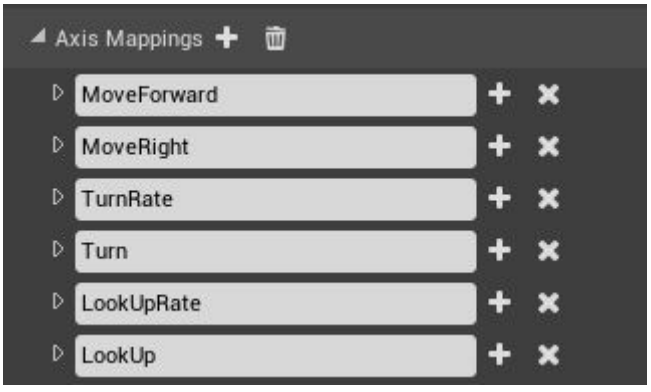
Keybindings

All used input keys are stored in config file (Edit->ProjectSettings->Input)

To transfer them into different project, copy DefaultInput.ini (DynamicCombatSystem/Config) into different project Config folder. Keep in mind that it will override input settings, safer option would be to rewrite them manually.




ActionKey4
4
Shift Ctrl Alt Cmd
Guard
Right Mouse Button
Shift Ctrl Alt Cmd
Parry
Right Mouse Button
Shift Ctrl Alt Cmd
Right Mouse Button
Shift Ctrl Alt Cmd
ToggleTargeting
Tab
Shift Ctrl Alt Cmd
TargetLeft
Q
Shift Ctrl Alt Cmd
TargetRight
E
Shift Ctrl Alt Cmd
ToggleInventory
I
Shift Ctrl Alt Cmd
ToggleMovement
Caps Lock
Shift Ctrl Alt Cmd
Sprint
Left Shift
Shift Ctrl Alt Cmd


Axis Mappings
MoveForward
MoveRight
TurnRate
Turn
LookUpRate
LookUp

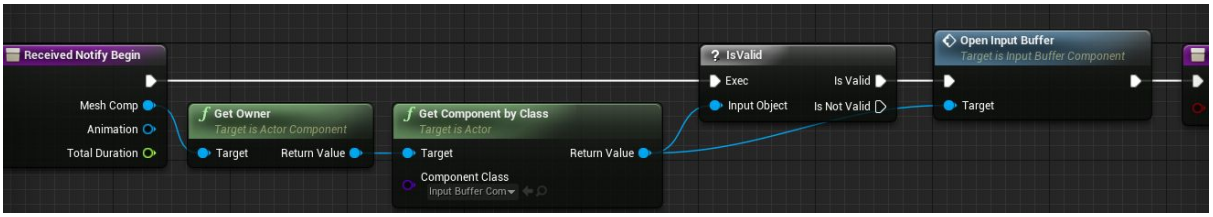
InputBuffer

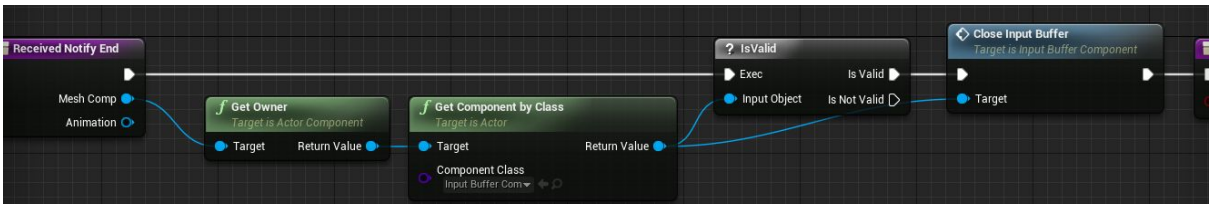
Player is using InputBufferComponent.

This component allows to store key(EInputBufferKey) in buffer and perform action based on that key when buffer is closed.

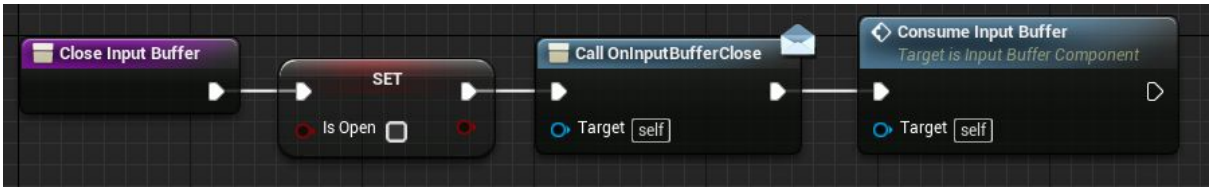
e.g. when input buffer is open, and character is attacking, you can press roll input, then roll key will be stored in buffer, and when buffer closes, checks if there is any key inside, in this case there would be roll key and Roll function will be called.
It gives feeling of better control over the character.

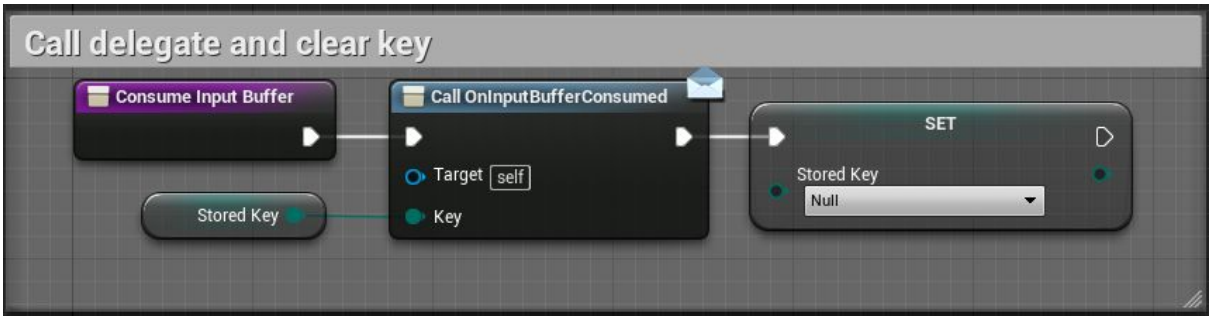
To open and close InputBuffer, use ANS_InputBuffer.
In this example InputBuffer is open from 50-70% of the montage.

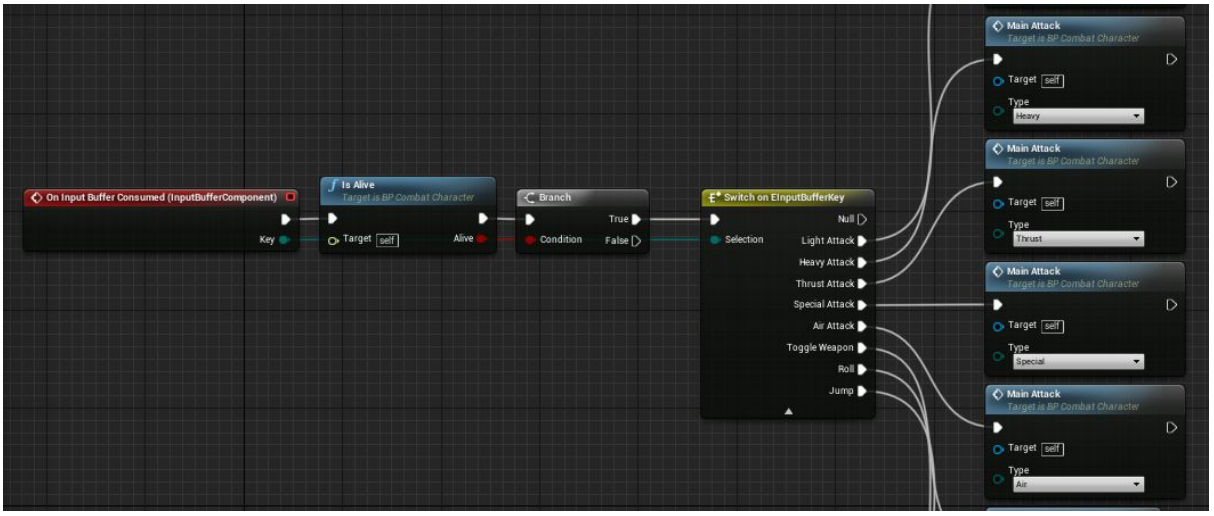




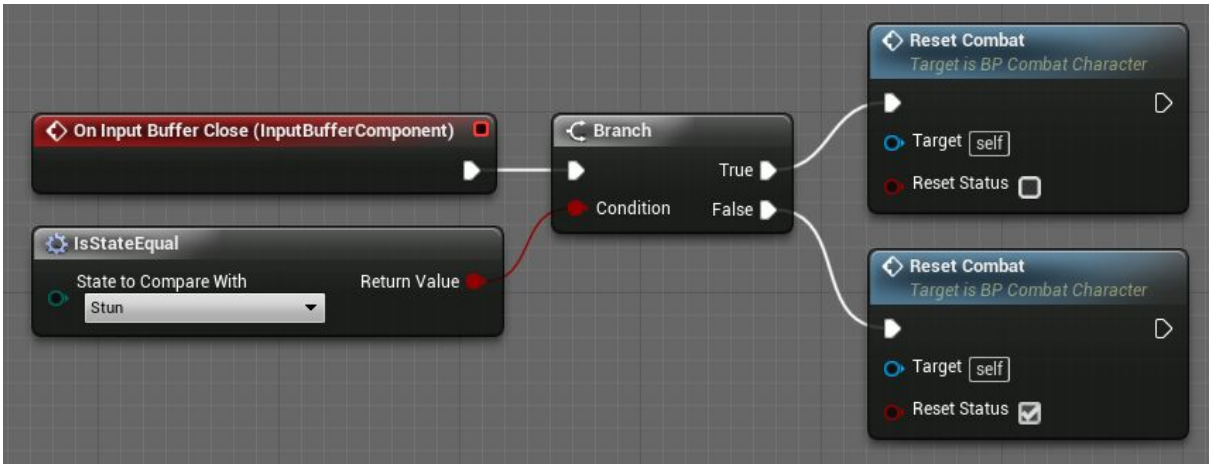
So actions like attack/roll/jump etc. are not calling their functions directly. They update key stored in InputBuffer, and when it closes, key is consumed, calling the proper function.

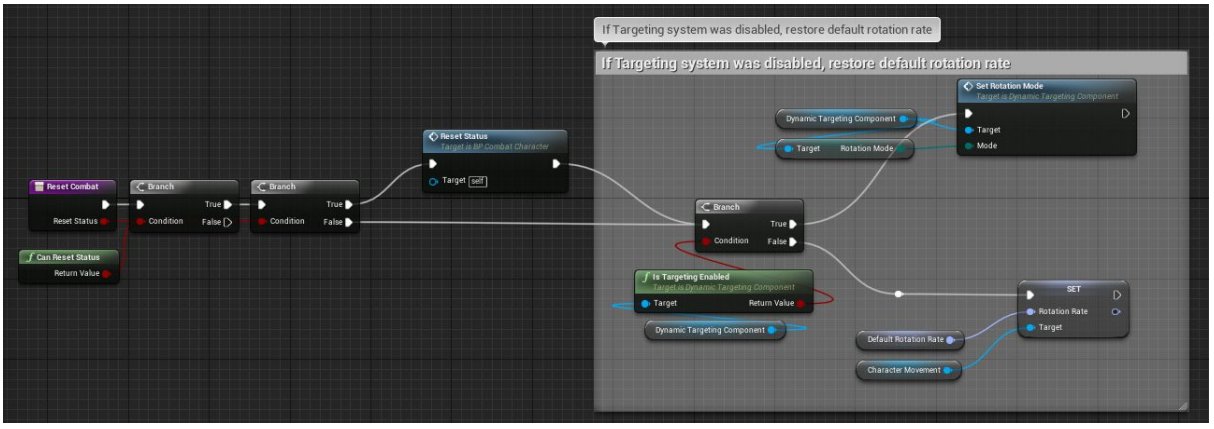


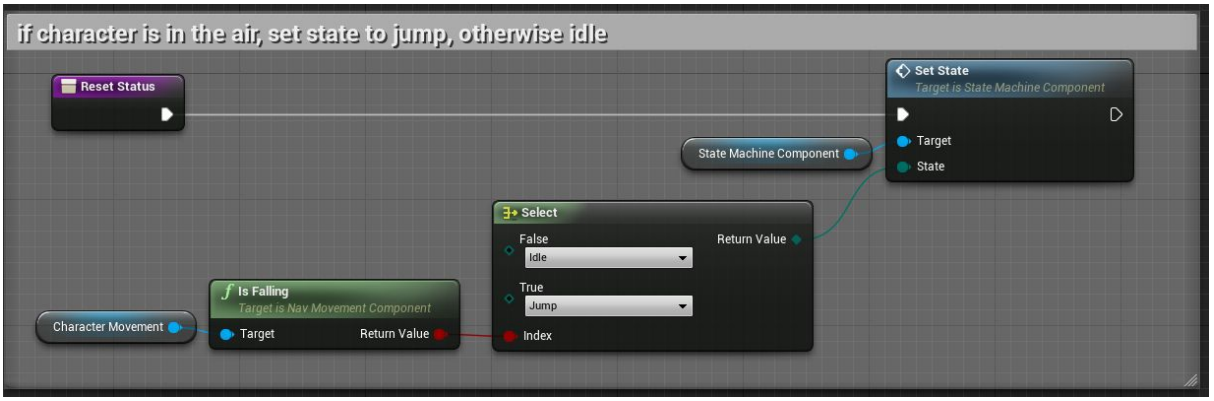




Additionaly when InputBuffer closes, ResetCombat function is called.







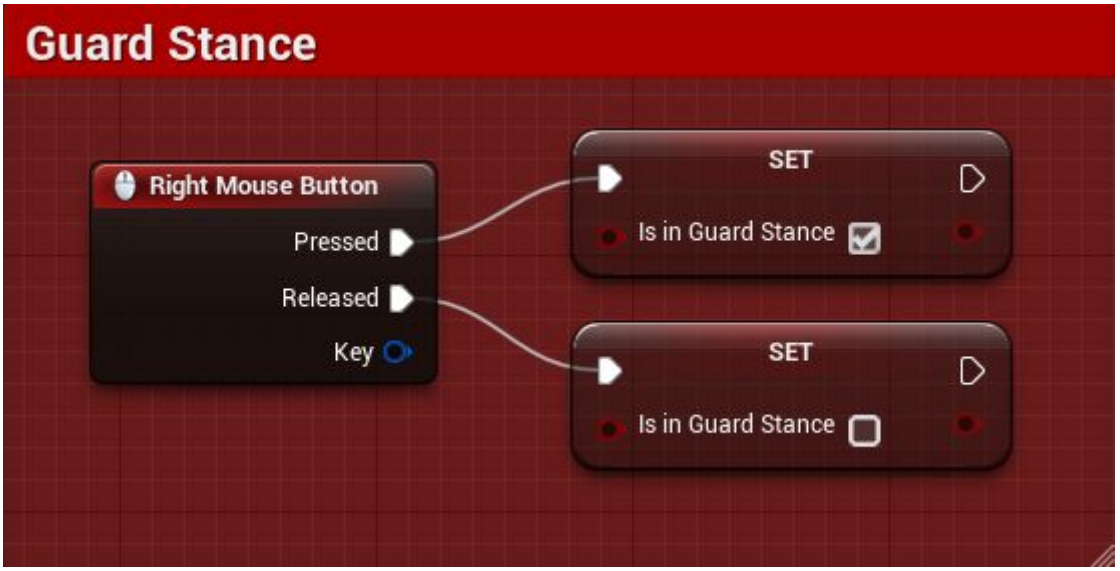
Notice that if character was in Stun state, status won't be resetted.
This is problem with anim notify states, when it is interrupted Notify End Event will be called on next frame.

Let's say that character was attacking and Input buffer was open, but then got hit and state changed to stun, also playing stun animation.
On the next frame InputBuffer end event would be called, calling function ResetCombat which would reset state back to Idle.
It means that stun state would last only for 1 frame.

That's why on montages like Stun/Impact you shouldn't use ANS_InputBuffer, but AN_ResetCombat instead.

Guard Stance

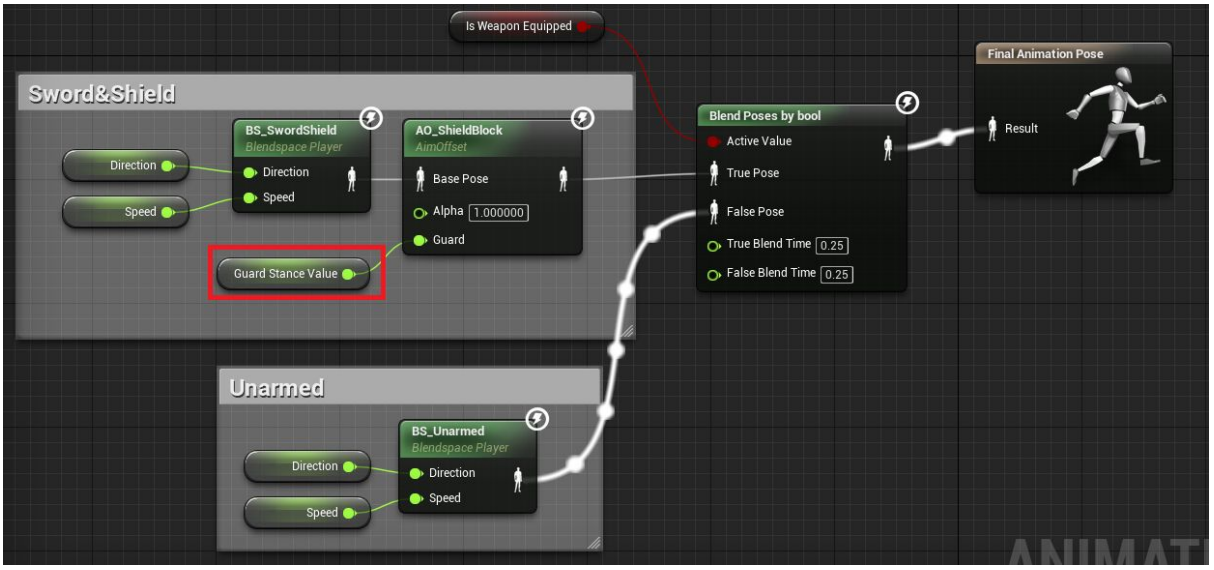
Enabled and disabled by pressing input key.



Based on that bool value, variable GuardStanceValue is smoothly changed between 0 and 1.

When character gets hit while GuardStanceValue equals 1, it means that hit was blocked.
If there is enough stamina, it will be removed instead of health and stun effect won't be applied.

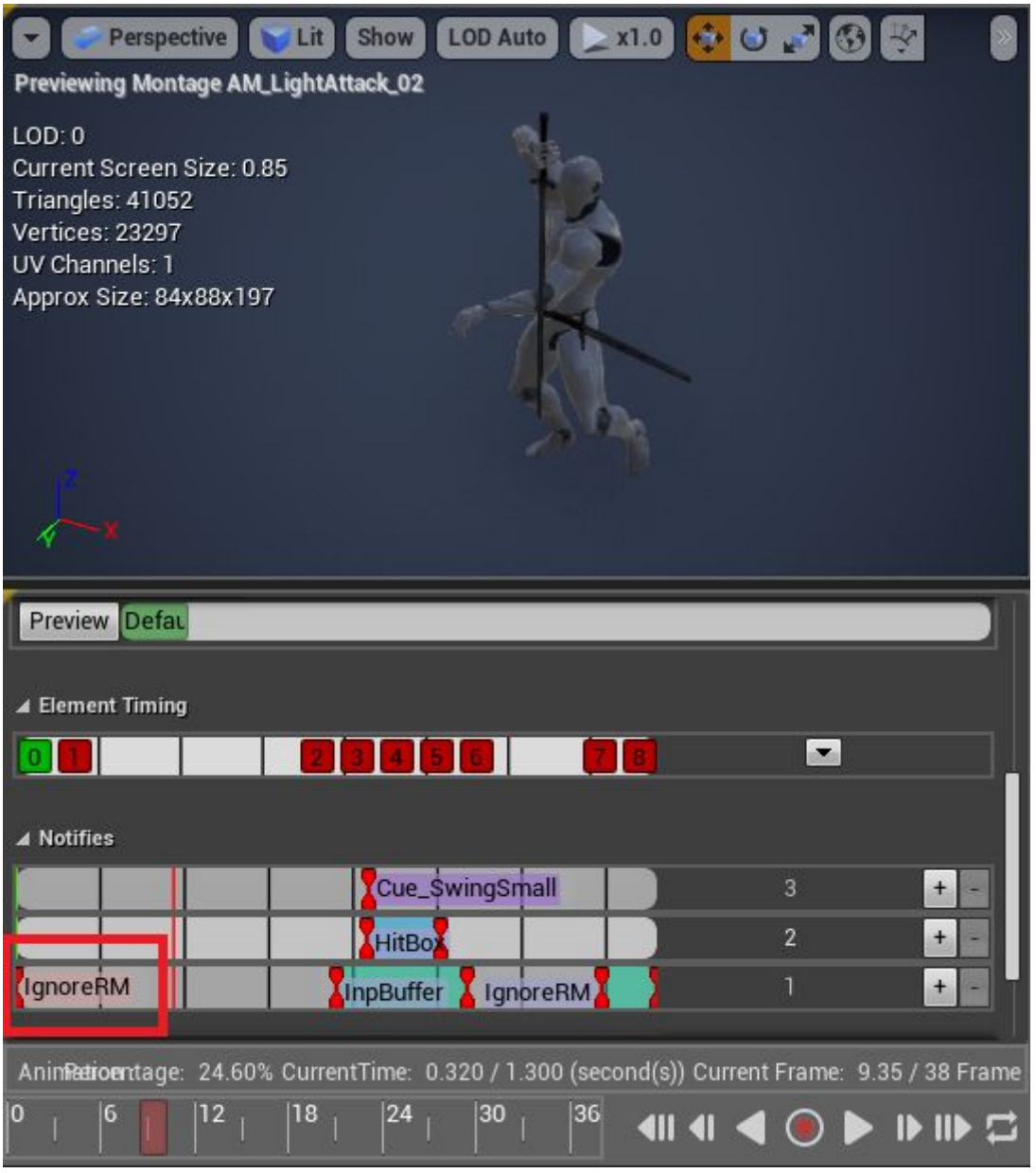
This value is also used in animation blueprint to rise/lower shield



Ignore Root Motion

System which allows character to turn into desired direction at the beginning of the montage like attack/roll etc.

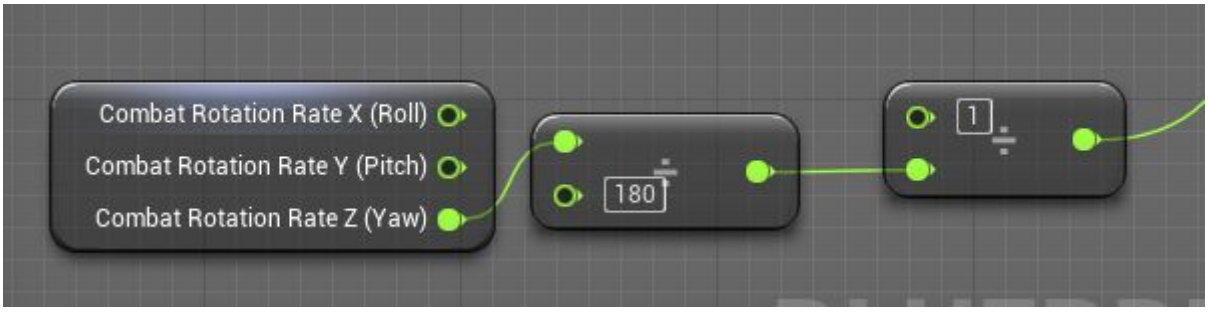
Firstly AN_IgnoreRootMotion placed at the begging of the montage is calling IgnoreRootMotion function from BP_CombatCharacter.



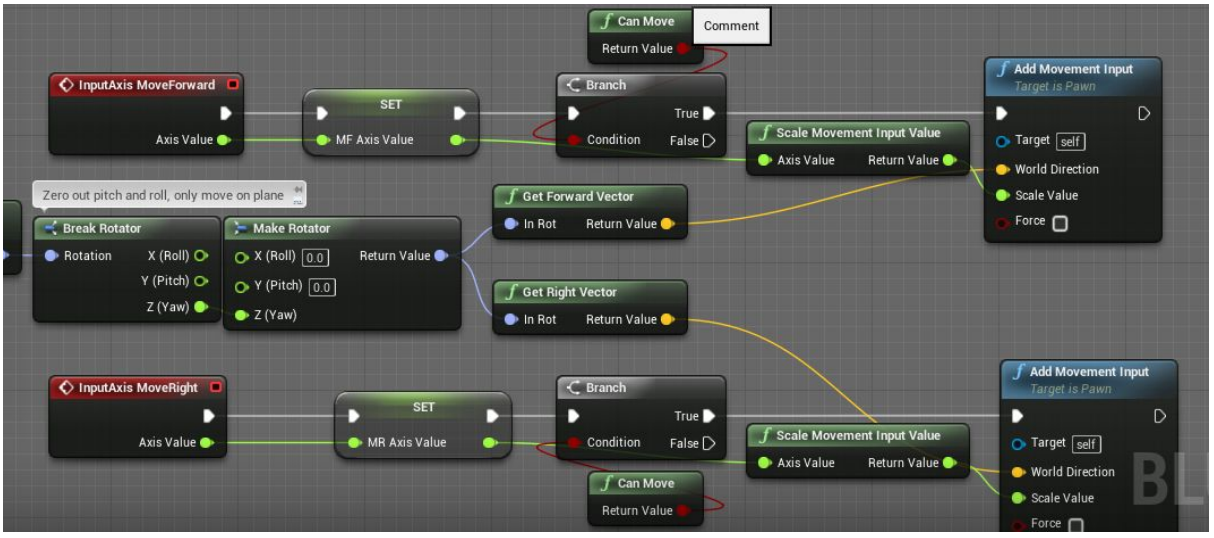
Function is calculating for how long root motion should be ignored based on variable CombatRotationRate (Rotator used as RotationRate in

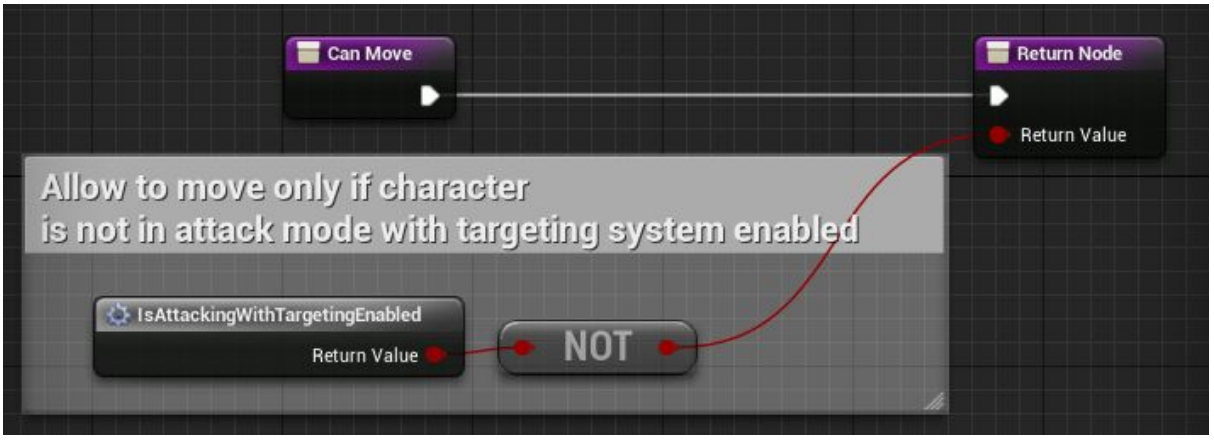
MovementComponent when character enters Roll/Attack state, by default it is 0,0,900)

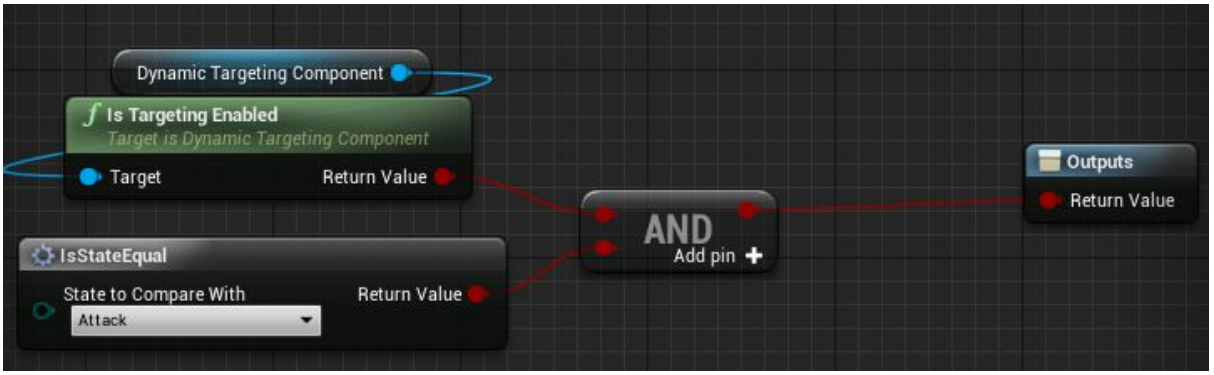
With 900 rotation rate yaw, because we only care about that, it takes 0.2sec to turn 180 degrees(max)
[1/(900/180)].



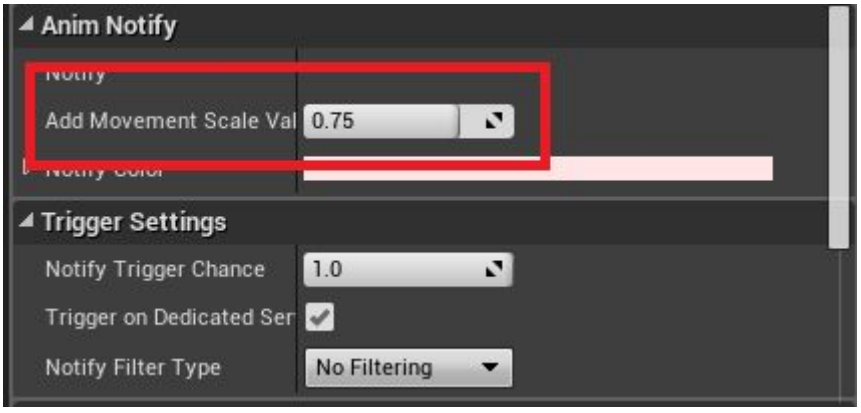
During that time, if targeting system is enabled and player is in attack state, character can't be moved using axis inputs, to prevent moving right/left/back.



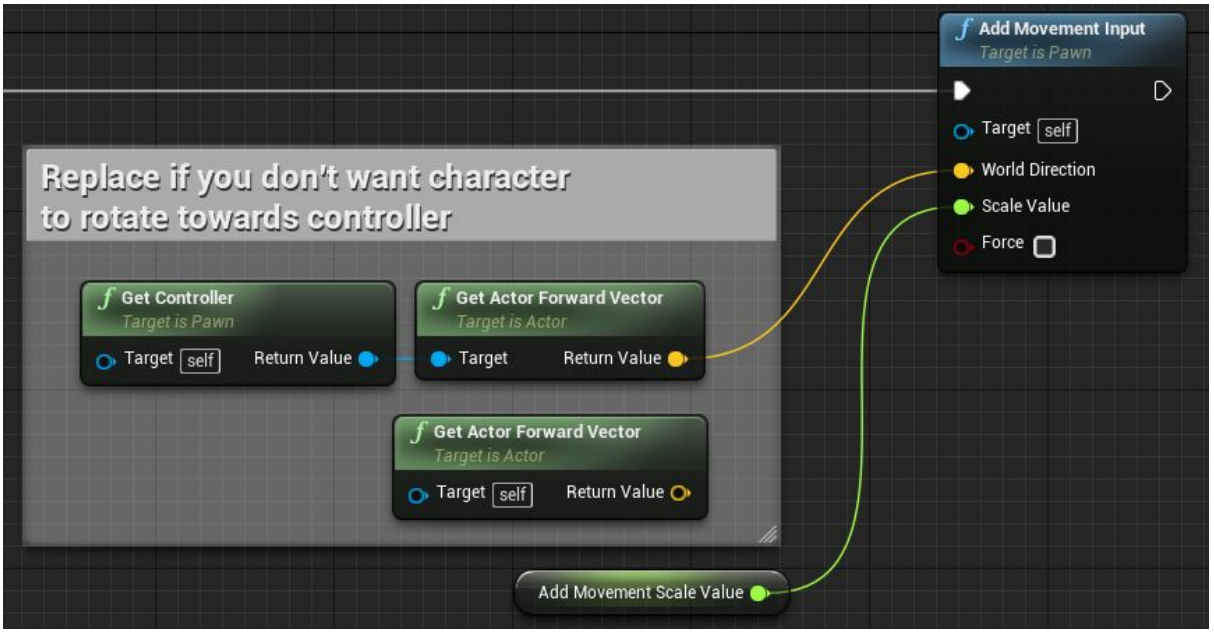




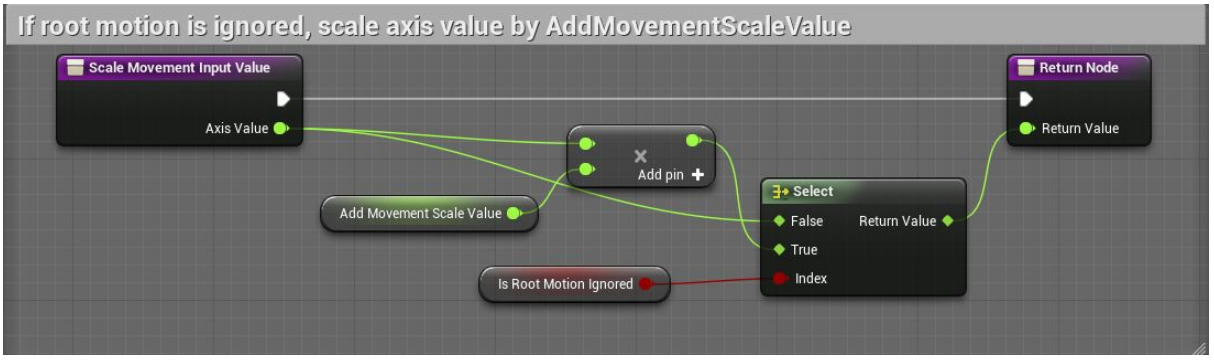
Instead character will be moved by ticking function using AddMovementScaleValue provided by [AN_IgnoreRootMotion].



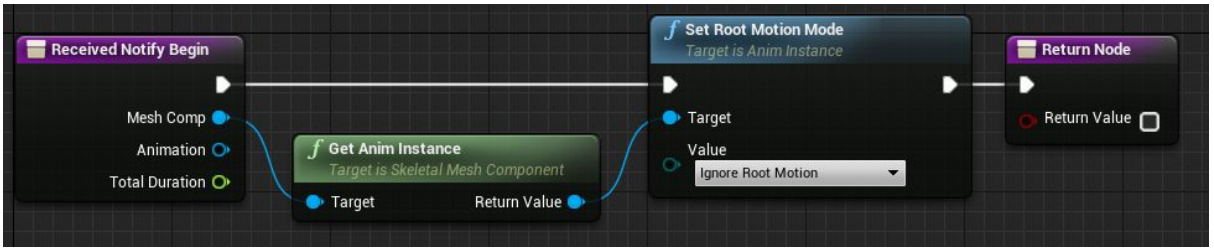
1 means move character(useful if you want character to move while root motion is ignored e.g in roll)
0 don't move at all (useful if you have InPlace attack animations)

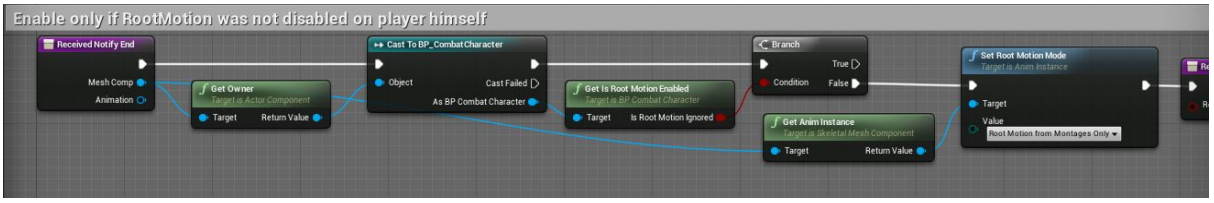


Otherwise if player is not in attack state, axis value will be scaled by AddMovementScaleValue, it means that if value on notify was 0, we won't be able to move character anyway.

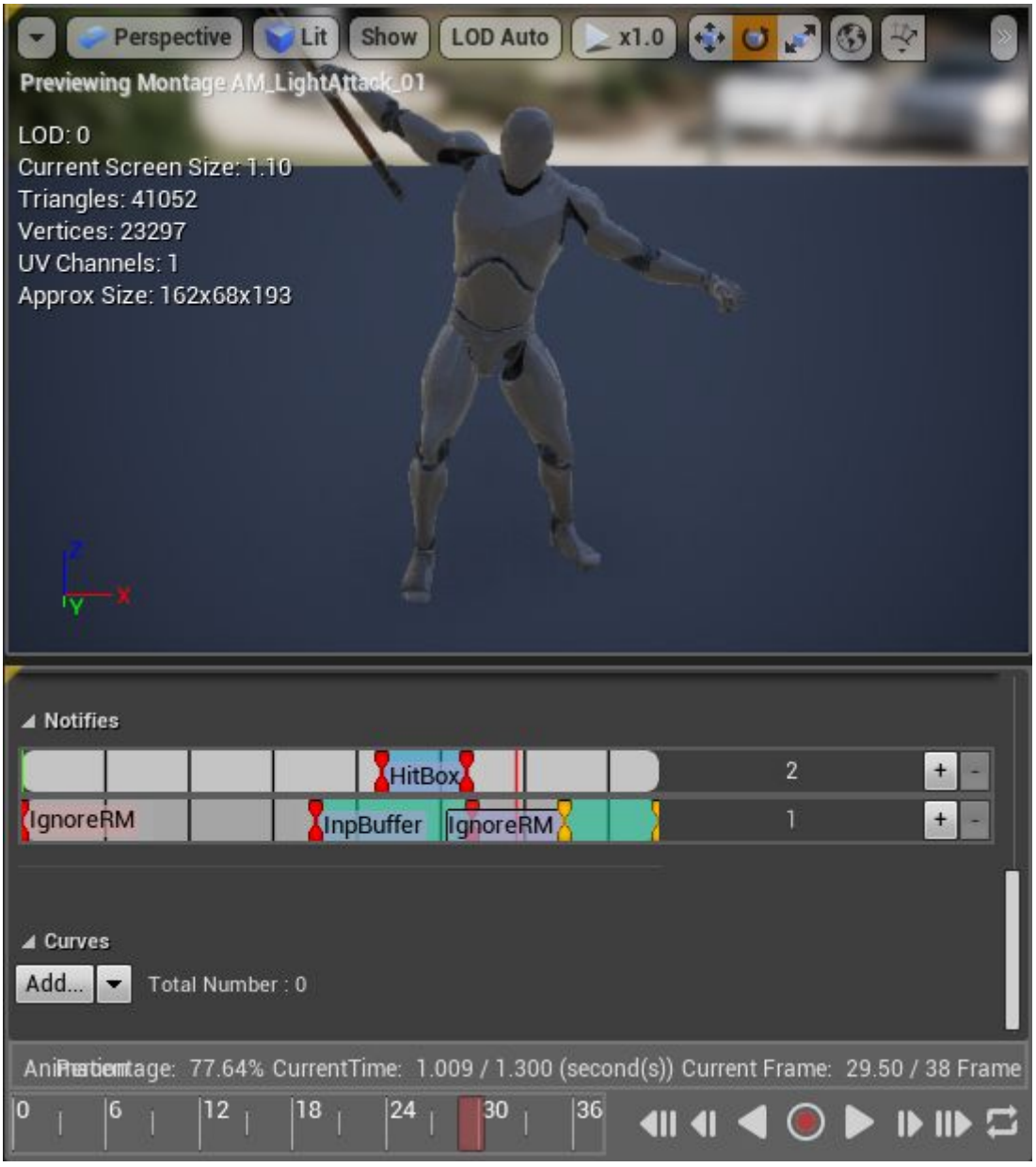


There is also another notify(state) which in contrast to AN_IgnoreRootMotion, is not calling player's blueprint function, but just ignores root motion on begin event, and enable it back on end event, if root motion was not already disabled by player himself.



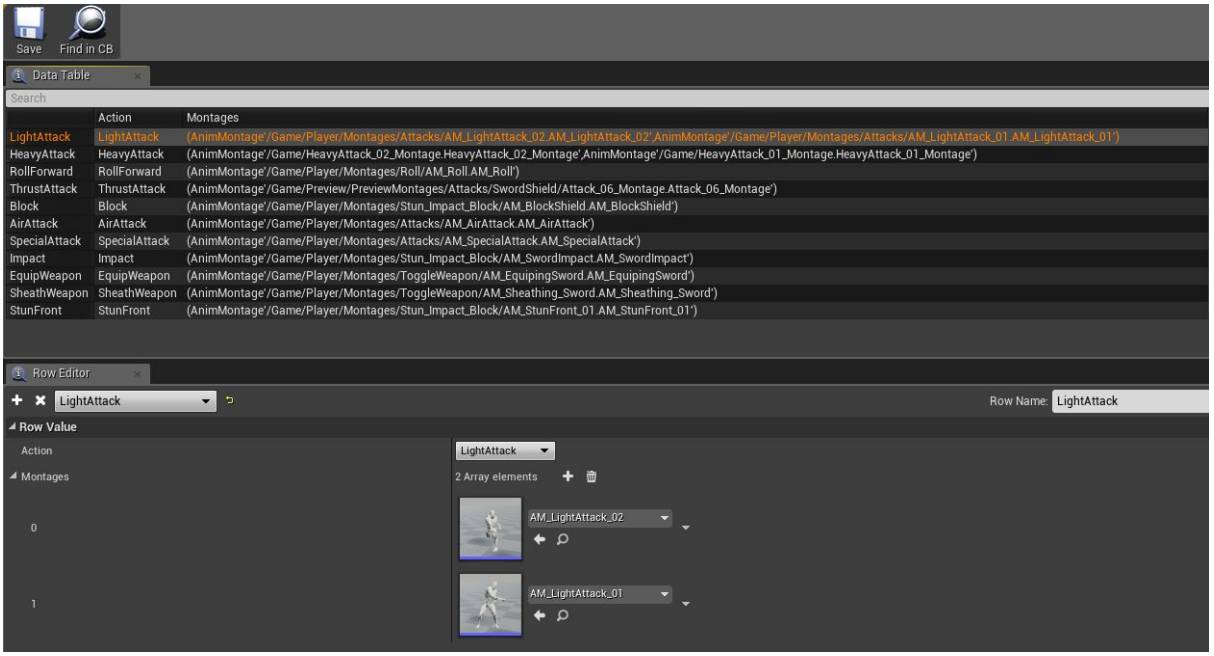


Using this notify is optional, it is most useful at the end (last 10-15%) of montages using root motion, like attack, roll to give player faster control over the character while montage is blending out.


Perspective
Lit
Show
LOD Auto
x1.0
Previewing Montage AM_LightAttack_01
LOD: 0
Current Screen Size: 1.10
Triangles: 41052
Vertices: 23297
UV Channels: 1
Approx Size: 162x68x193
Notifies
HitBox
2
IgnoreRM
InpBuffer
IgnoreRM
1
Curves
Add...
Total Number : 0
Percentage: 77.64% CurrentTime: 1.009 / 1.300 (second(s)) Current Frame: 29.50 / 38 Frame
0
6
12
18
24
30
36

Montage Actions used by player

All montages used by player are stored in DataTable (DataTables/DT_CombatCharacter), using MontageManagerComponent.
I highly recommend to keep that convection if you are going to add new montages.



Take note that RowName must be the same as MotageAction value.

Important Montages

If any of those montages will not be valid, action will simply not be performed.

Light Attack (Slot: FullBody, with root motion)
Heavy Attack (Slot: FullBody, with root motion)
Thrust Attack (Slot: FullBody, with root motion)
Special Attack (Slot: FullBody, with root motion)
Air Attack (Slot: UpperBody, no root motion)
Impact (Slot: FullBody, with root motion)
Stun Front(Slot: FullBody, with root motion)

Block (Slot: UpperBody, no root motion)
Equip Weapon (Slot: UpperBody, no root motion)
Sheath Weapon (Slot: UpperBody, no root motion)
Roll Forward (Slot: FullBody, with root motion)

Additional Montages

For directional stun and roll montages there is already implemented logic inside player character, just set animations with proper enum value, Row Name and it will work.

Stun Back (Slot: FullBody, with root motion)
Stun Left (Slot: FullBody, with root motion)
Stun Right (Slot: FullBody, with root motion)

Roll Forward Left (Slot: FullBody, with root motion)
Roll Forward Right (Slot: FullBody, with root motion)
Roll Left (Slot: FullBody, with root motion)
Roll Right (Slot: FullBody, with root motion)
Roll Back (Slot: FullBody, with root motion)
Roll Back Left (Slot: FullBody, with root motion)
Roll Back Right (Slot: FullBody, with root motion)

AI

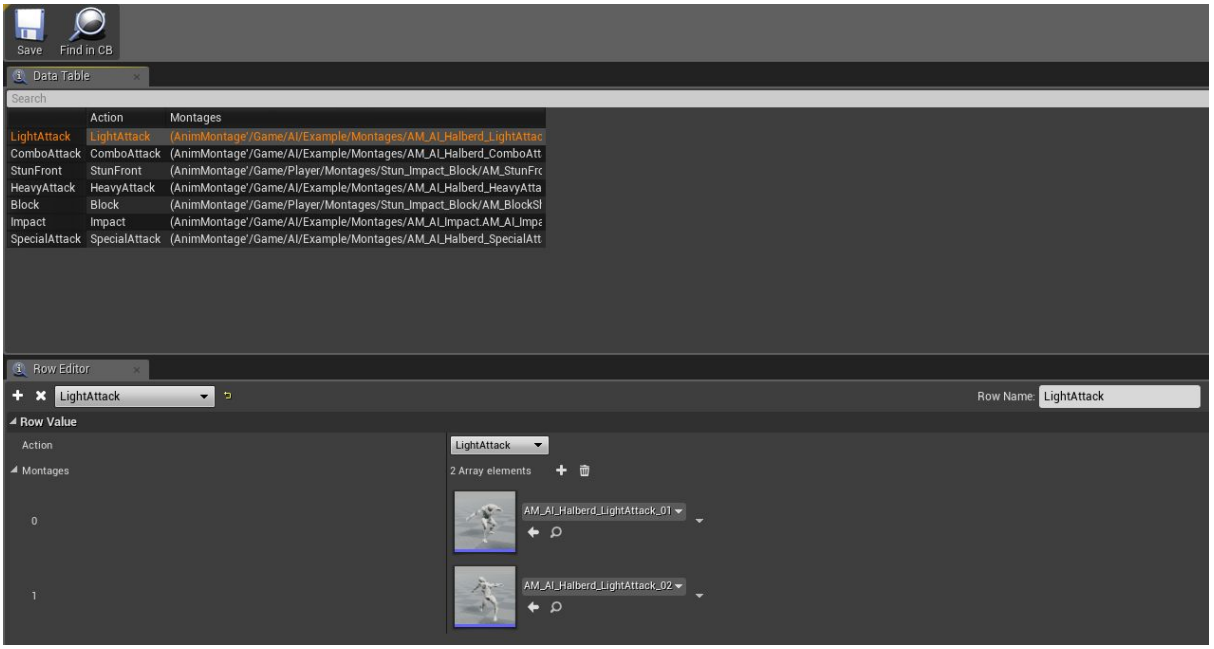
Remember to enable EQS.
https://docs.unrealengine.com/latest/INT/Engine/AI/EnvironmentQuerySystem/QuickStart/2

Creating new AI

1. Create new blueprint based on BP_BaseAI
2. Create new BehaviorTree with BB_Base as blackboard or use BT_ExampleAI
3. In AI defaults set BehaviorTree pointer to created one
4. By default AI will have BP_BaseAIController as AIController, it will set player as target value in blackboard on detection
5. Set AI mesh, anim instance, play around with default settings
6. Set Montages DataTable in MontagesManagerComponent to DT_ExampleAI or create your own with different animations
7. After those steps AI should be ready, and you can start building Behavior Tree using included tasks/services/decorators if you didn't use BT_ExampleAI

Animations used by AI

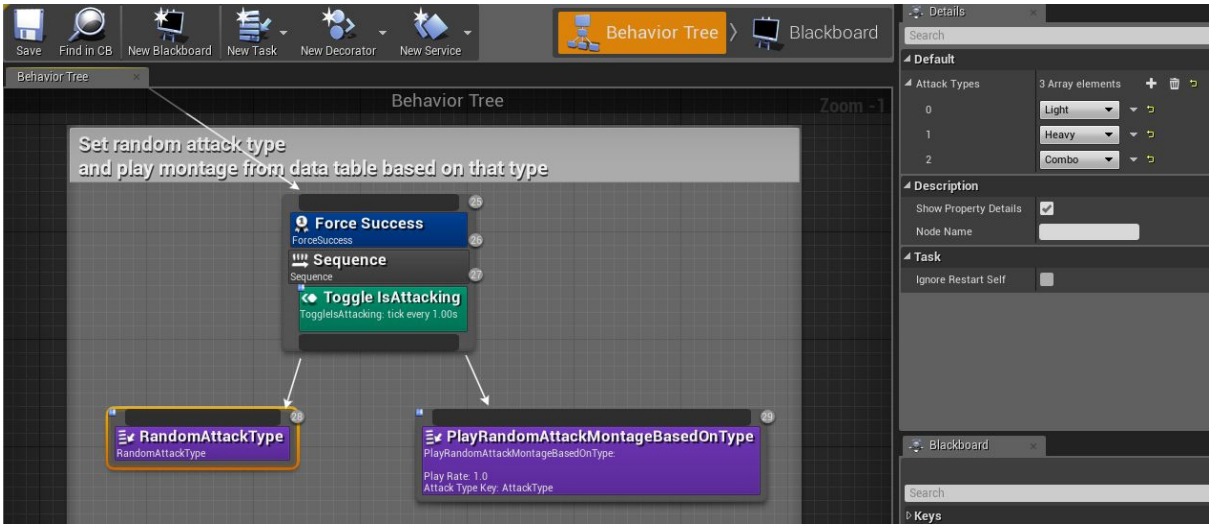
Same as player, all montages used by AI are stored in DataTable (DataTables/DT_ExampleAI), using MontageManagerComponent.
I highly recommend to keep that convection if you are going to add new montages.

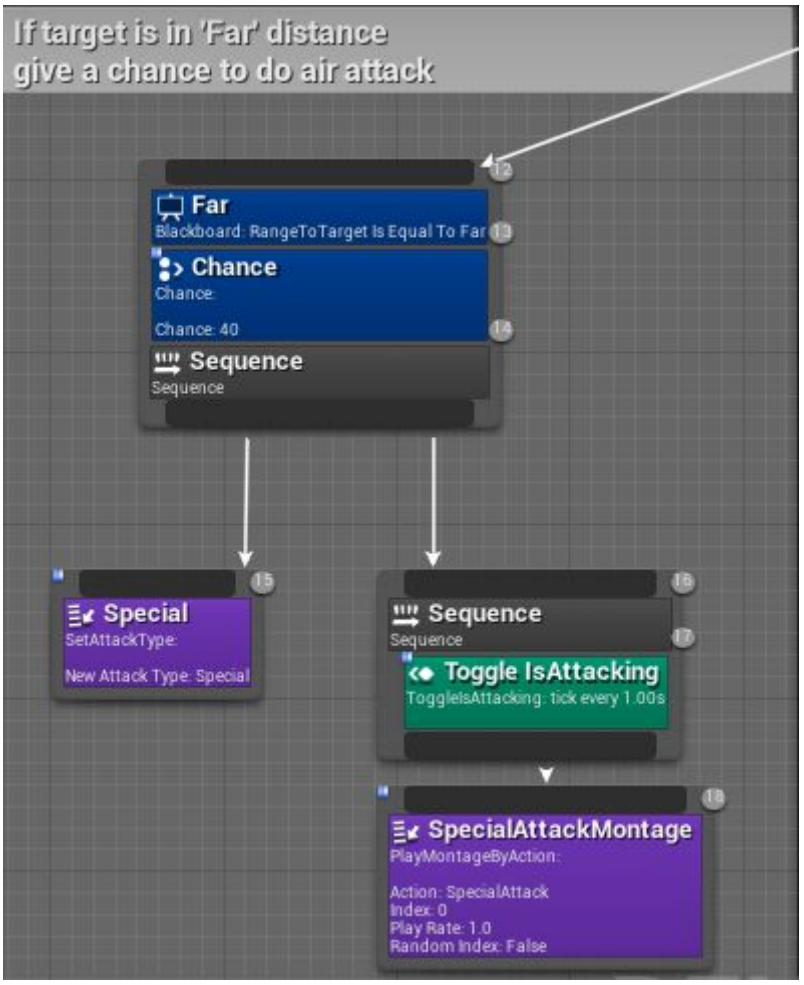


Take note that RowName must have same name as Action value.

Important Montages

If you take a look at BT_ExampleAI, it uses light/heavy/combo/special attack types, so he will need to set those in DataTable as well.





Light Attack (Slot: FullBody, with root motion)
Heavy Attack (Slot: FullBody, with root motion)
Special Attack (Slot: FullBody, with root motion)
Combo Attack (Slot: FullBody, with root motion)
Impact (Slot: FullBody, with root motion)
Stun Front(Slot: FullBody, with root motion)
Block (Slot: UpperBody, no root motion)

Additional Montages

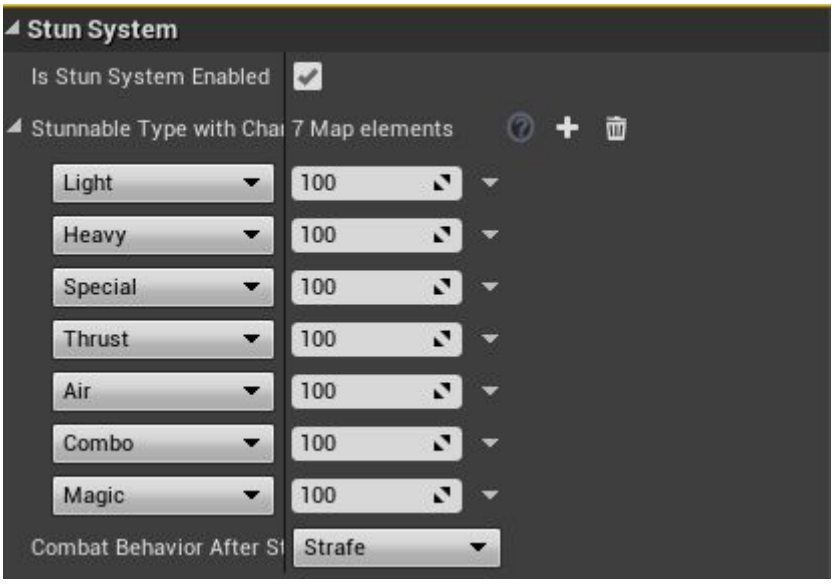
For directional stun there is already implemented logic inside player character, just set animations with proper enum value and Row Name and it will work.

Stun Back (Slot: FullBody, with root motion)

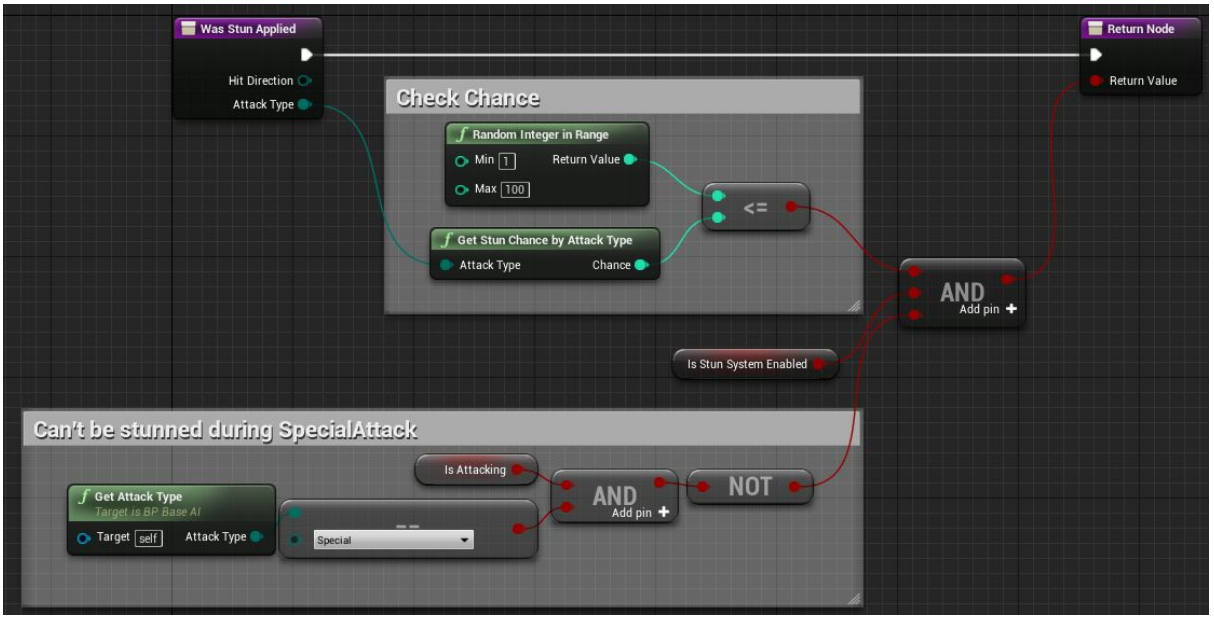
Stun Left (Slot: FullBody, with root motion)
Stun Right (Slot: FullBody, with root motion)

Stun System

In AI default settings stun system can be enabled/disabled.
If it's enabled, you can set chances for this character to get stunned and CombatBehavior after stun ends.

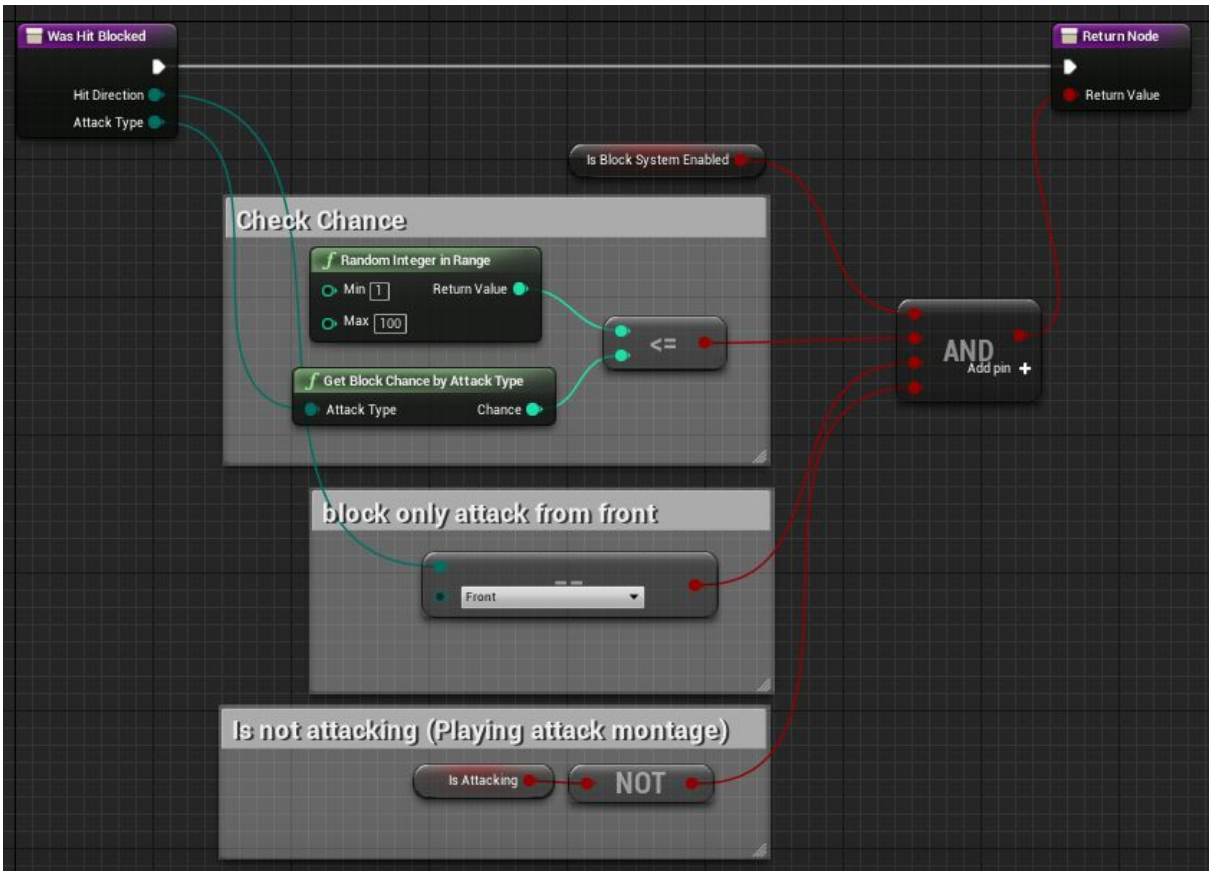


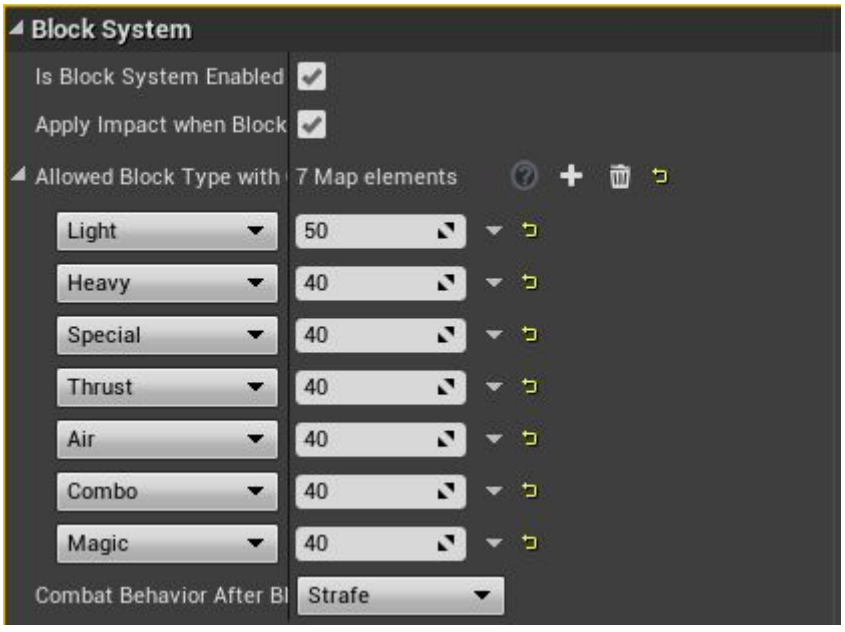
Additionally in function WasStunApplied you can add other conditions which must be fulfilled to apply stun effect
(e.g. never stun character while special attack is performed).



Block System

Very similar to stun system but to change block conditions use WasHitBlocked function instead.

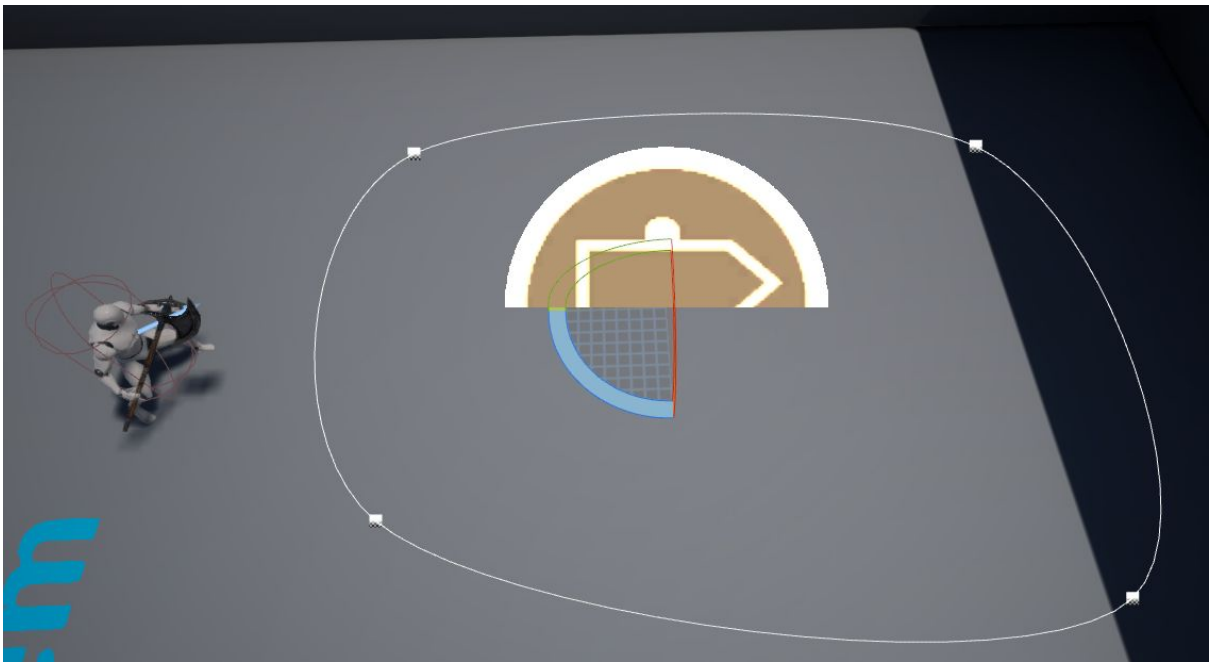




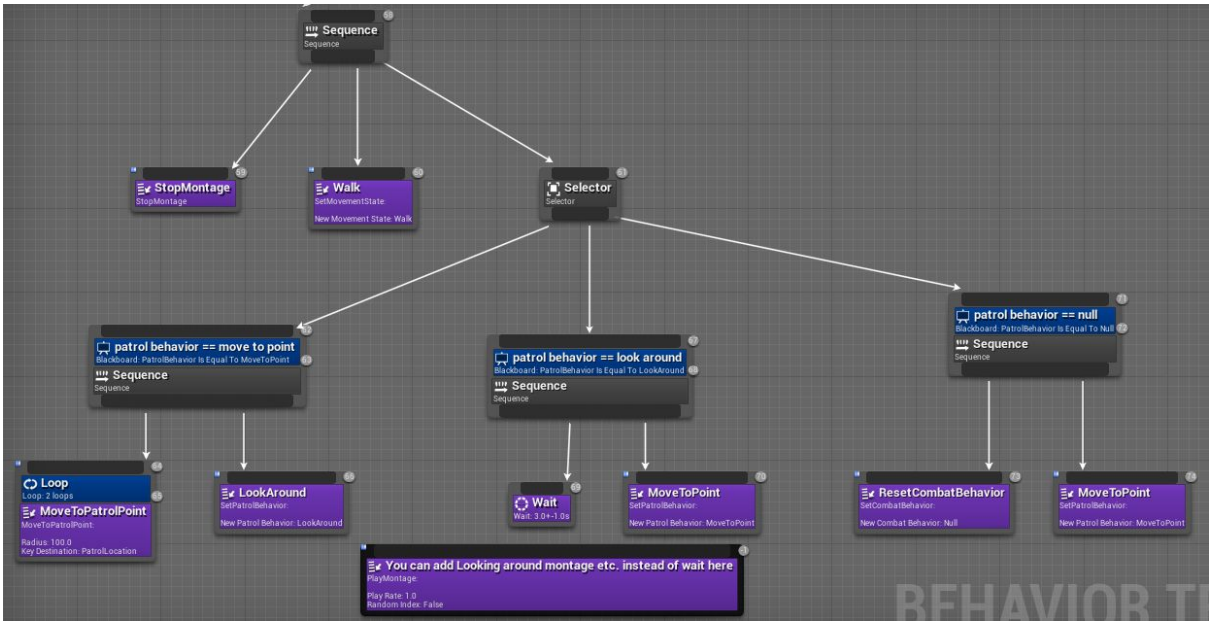
If ApplyImpactWhenBlock is enabled, Impact function from IsAttackable interface will be called on attacker.
It can be disabled in case there is Dodge/Evade block montage played instead of blocking with shield/sword etc.

Patrol

Allows AI to patrol using spline from BP_PatrolPath blueprint.
Drag and drop it into the level creating patrol spline, then set it with eye dropper in AI default settings.

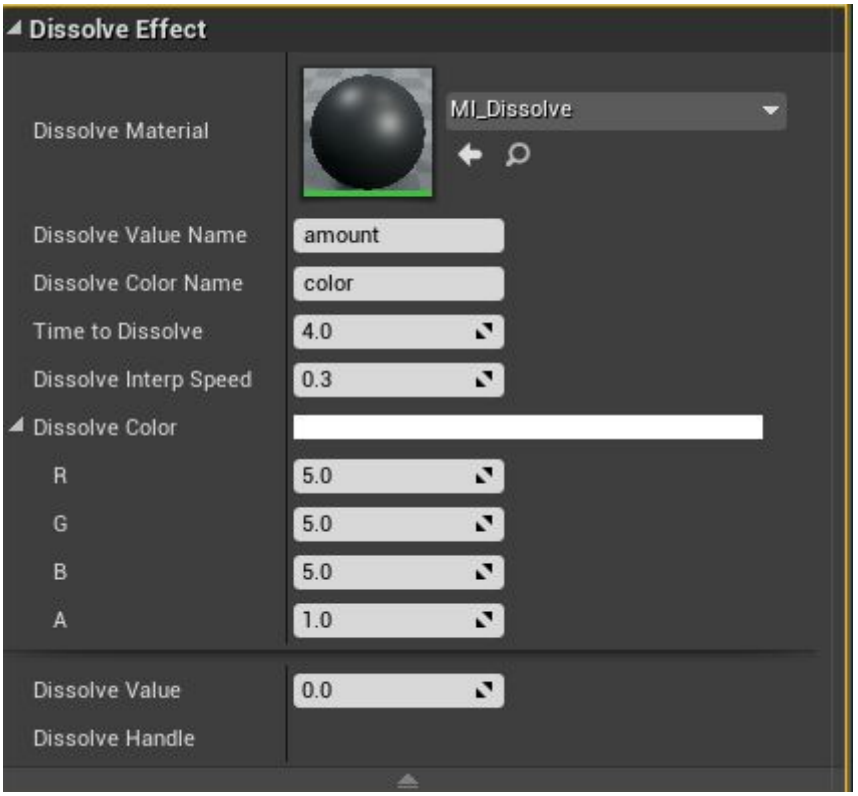


In BT_ExampleAI by default if there is not any enemy around, character will be moving towards 2 patrol points, wait and repeat.


Sequence
Sequence
StopMontage
StopMontage
Walk
SetMovementState:
New Movement State: Walk
Selector
Selector
patrol behavior == move to point
Blackboard: PatrolBehavior Is Equal To MoveToPoint
Sequence
Sequence
patrol behavior == look around
Blackboard: PatrolBehavior Is Equal To LookAround
Sequence
Sequence
patrol behavior == null
Blackboard: PatrolBehavior Is Equal To Null
Sequence
Sequence
Loop
Loop: 2 loops
MoveToPatrolPoint
MoveToPatrolPoint:
Radius: 100.0
Key Destination: PatrolLocation
LookAround
SetPatrolBehavior:
New Patrol Behavior: LookAround
Wait
Wait: 3.0+-1.0s
MoveToPoint
SetPatrolBehavior:
New Patrol Behavior: MoveToPoint
ResetCombatBehavior
SetCombatBehavior:
New Combat Behavior: Null
MoveToPoint
SetPatrolBehavior:
New Patrol Behavior: MoveToPoint
You can add Looking around montage etc. instead of wait here
PlayMontage:
Play Rate: 1.0
Random Index: False

Dissolve Effect

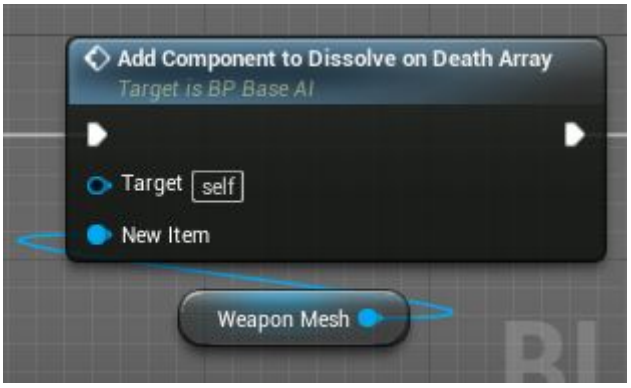
Effect applied after character dies.



DissolveValueName - parameter name of dissolve effect value material
DissolveColorName - parameter name of dissolve color in material

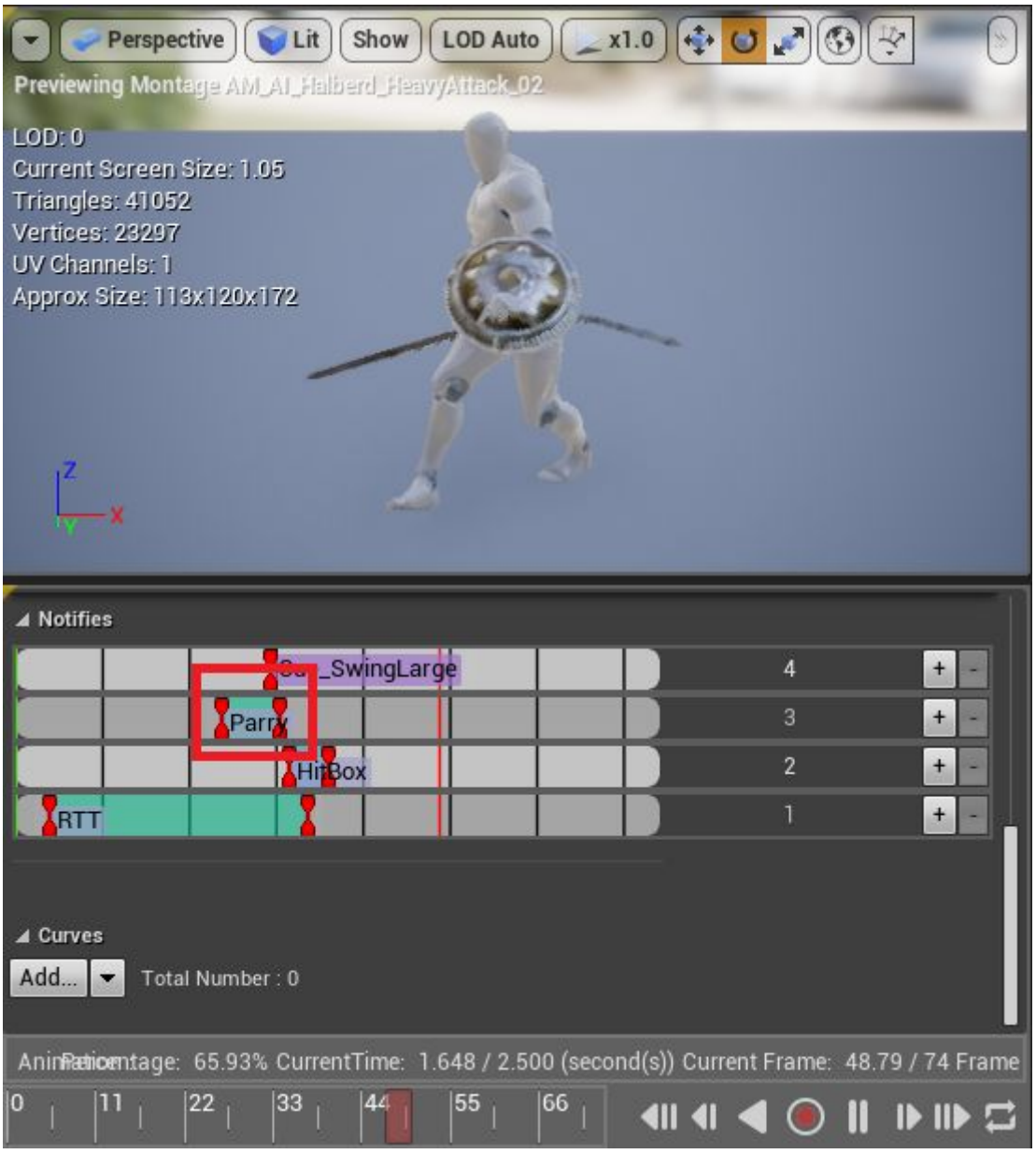
TimeToDissolve - time after dissolve effect will start be applying
DissolveInterpSpeed - how fast dissolve material will be applied
Dissolve Color - dissolve material color

New Primitive Components(attachments like swords/shields) can be added with AddComponentToDissolveOnDeathArray



Parry

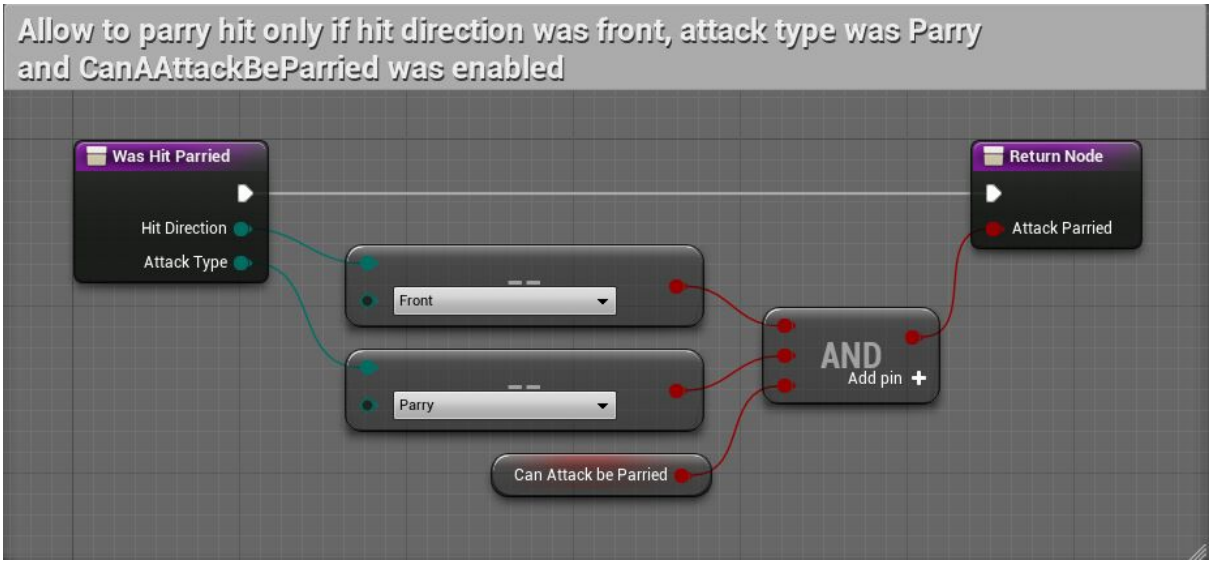
If AI is attacking and gets hit while is in notify state ANS_ParrySusceptible (and attack type was Parry), then AI gets stun and plays ParryGetHit montage from DataTable.



ANS_ParrySusceptible is calling functions from interface IsSusceptibleToParry (Parry SusceptibleStart/End)


Parry Susceptible Start
SET
Can Attack be Parried
Return Node
Dummy


Parry Susceptible End
SET
Can Attack be Parried
Return Node
Dummy


Allow to parry hit only if hit direction was front, attack type was Parry and CanAAttackBeParried was enabled
Was Hit Parried
Hit Direction
Attack Type
Front
Parry
AND
Add pin
Can Attack be Parried
Return Node
Attack Parried

Interfaces

IsAttackable

Defines if owning class can be attacked.
Functions:
HandleHit - called when character receives damage..
Impact - called e.g. when attacked actor blocked hit.
IsAlive - should be implemented to return true if character is alive, false otherwise.

CanBeImmortal

EnableImmortality - makes character immortal
DisableImmortality - disable immortality on character

IsTargetable

Defines if character can be targeted by dynamic targeting component.
More info in DynamicTargetingComponent.

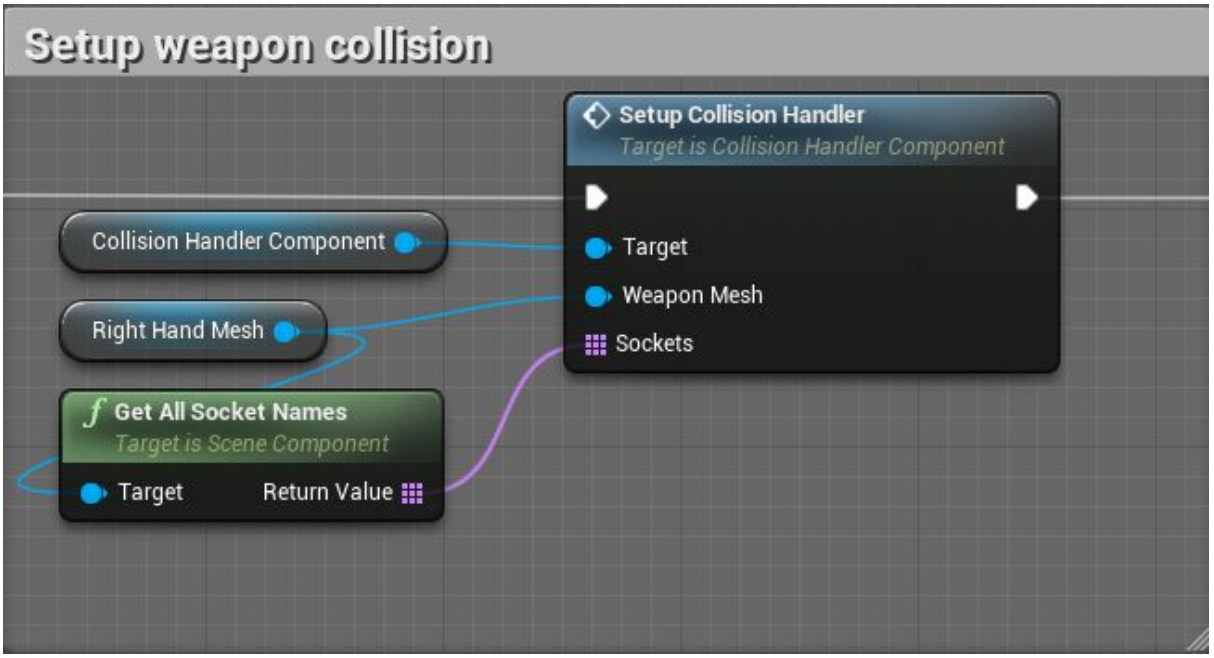
IsSusceptibleToParry

Described in Parry

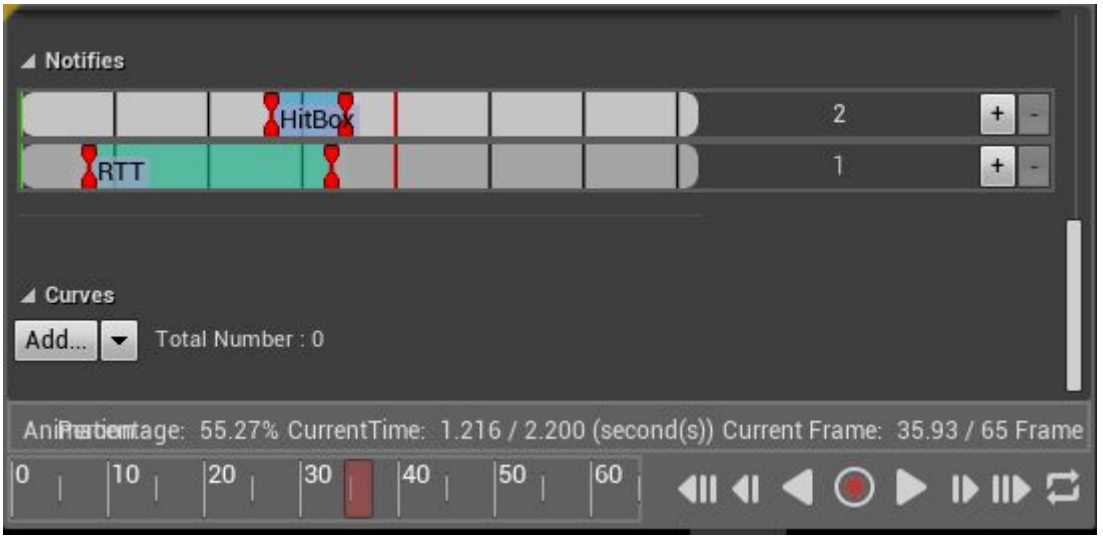
Components

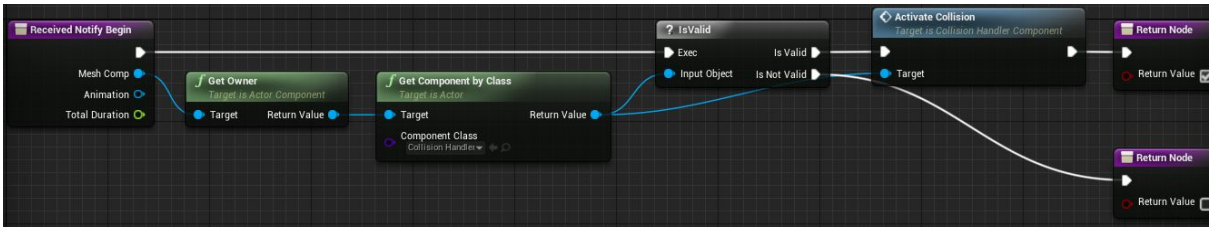
CollisionHandlerComponent

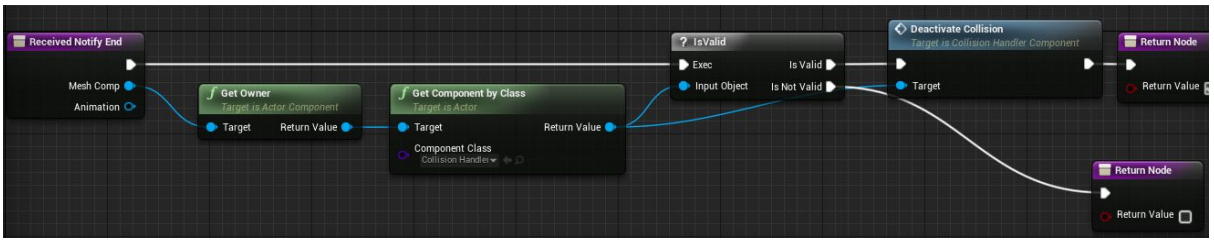
Component used in both Player and AI which allows to detect collision with other actors based on given sockets of mesh(weapon)
e.g. to apply damage on them.



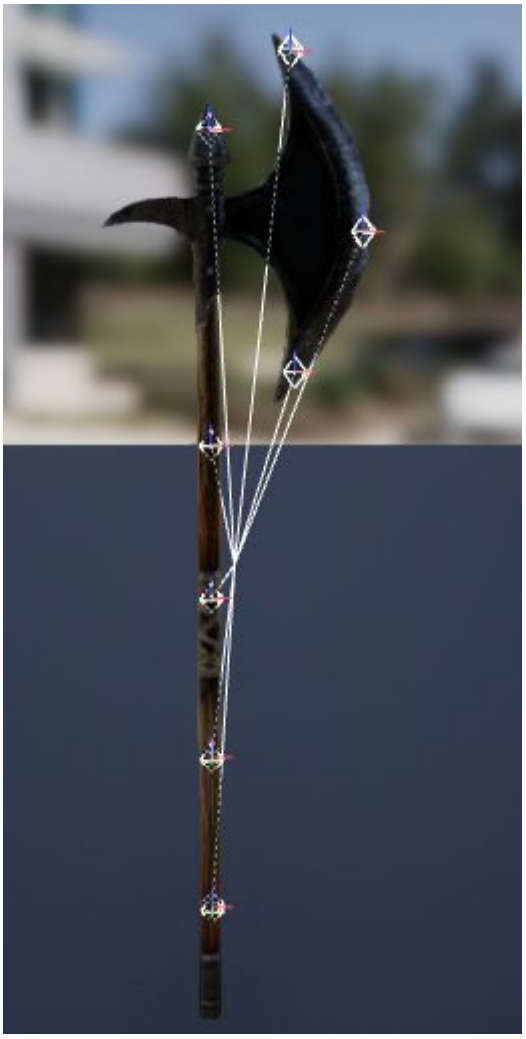
To enable/disable detecting collision,use ANS_HitBox



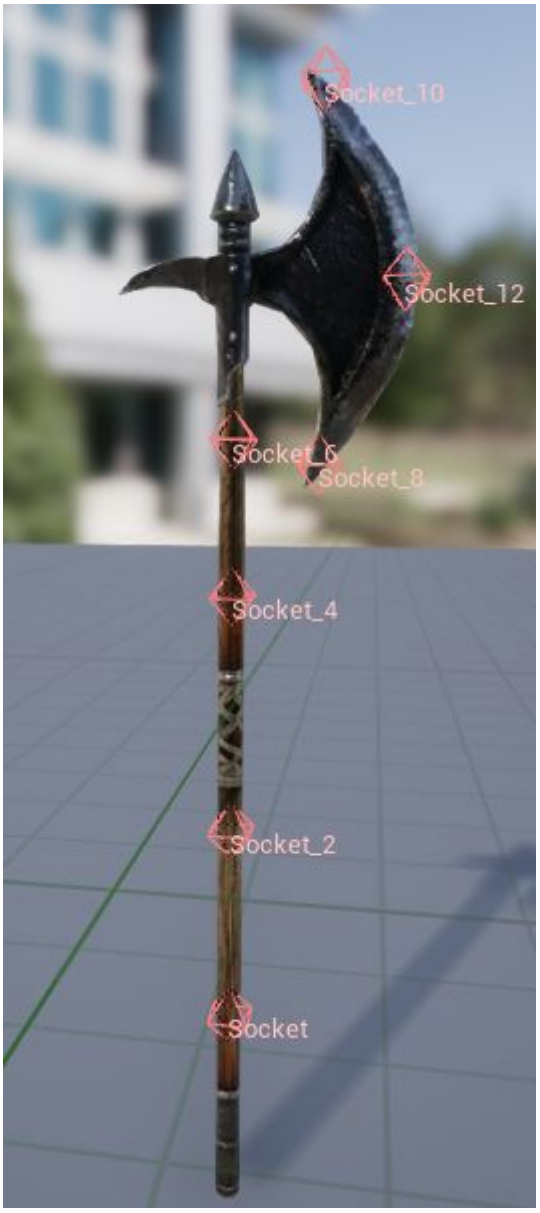




Item used by this component should be skeletal mesh.
Example weapon with few sockets on it.

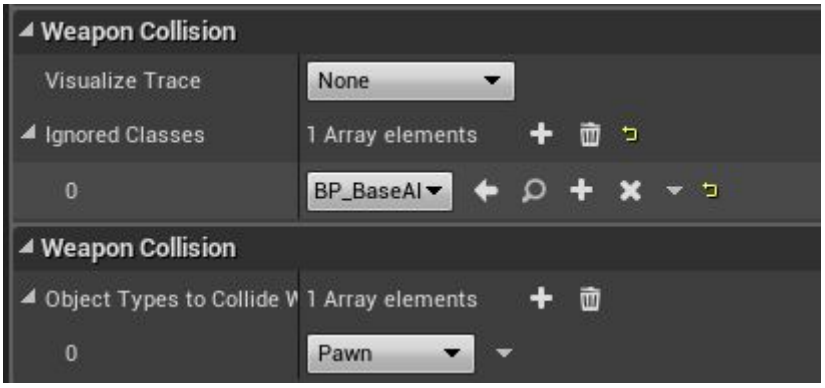


Sockets on Skeletal Mesh



Sockets on Static Mesh

To avoid calling OnHit event when collision is detected on friendly actors(companion etc.),
their class can be added to IgnoredClasses array(all classes inheriting from this class will also be ignored)



DynamicTargetingComponent

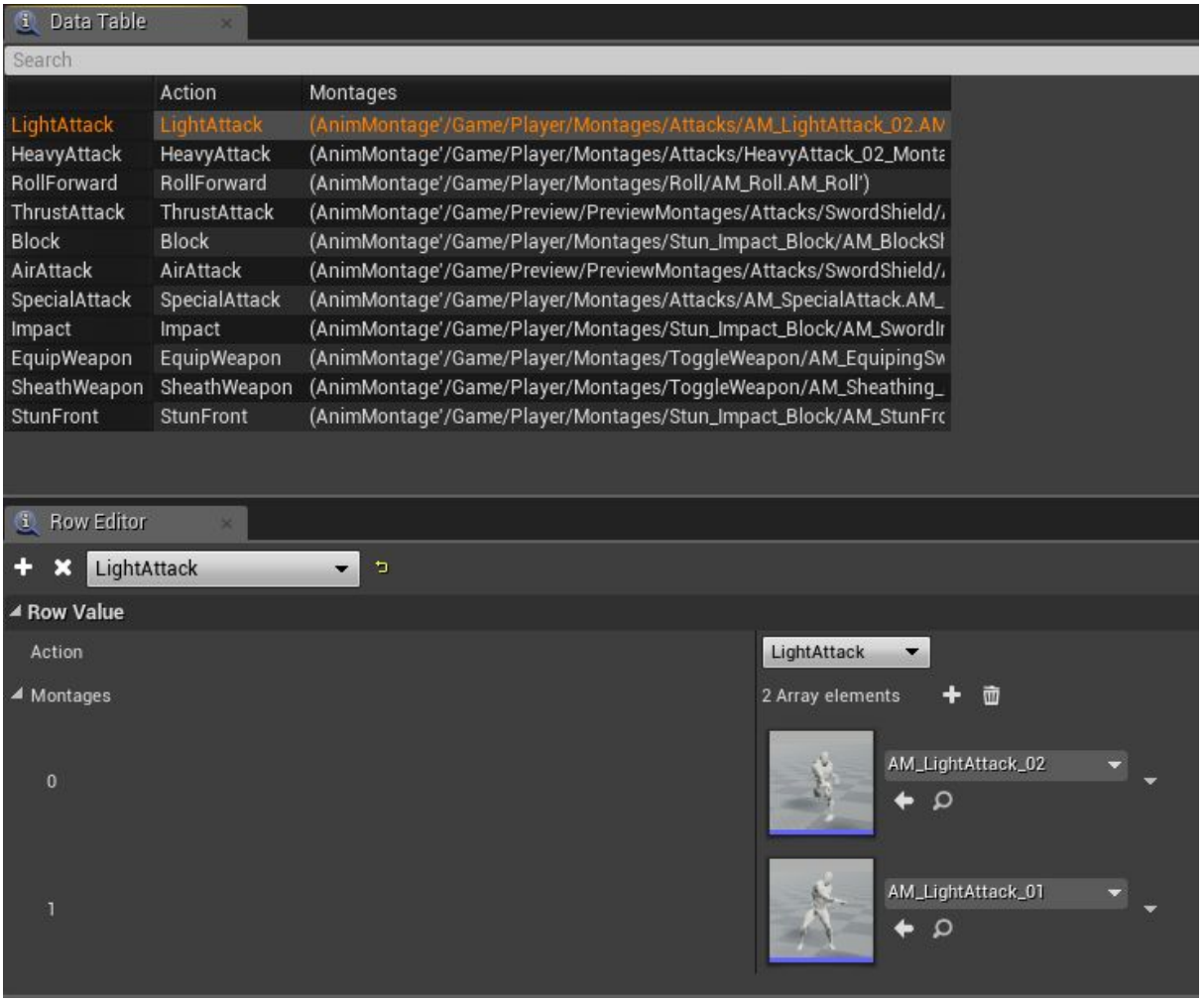
https://www.unrealengine.com/marketplace/dynamic-targeting
Implementation video:
https://www.youtube.com/watch?v=sp6SicLCjJQ

InputBufferComponent

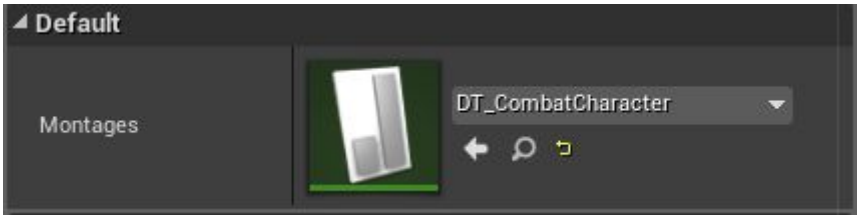
Described in InputBuffer section.

MontageManagerComponent

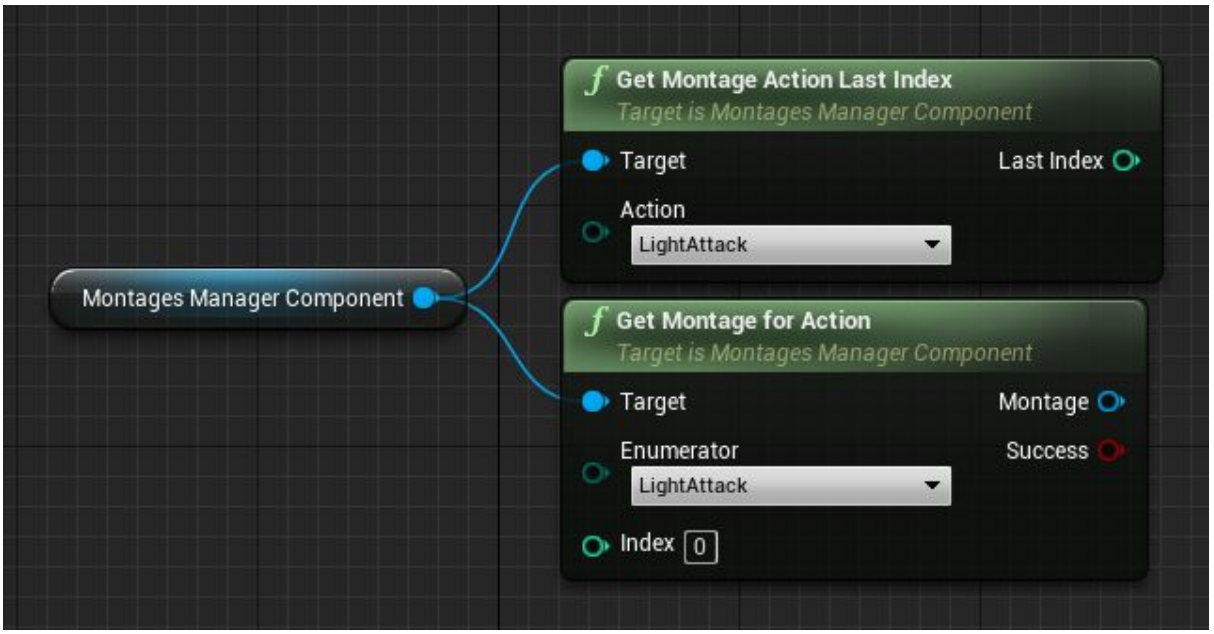
Component used both by AI and Player, it helps to keep code clean and easily modifiable by storing all montages in one place - DataTables

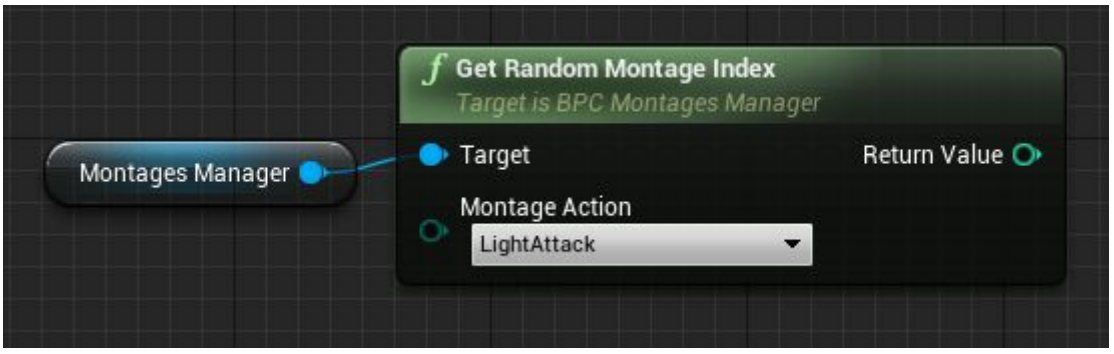


To use, set Data Table which will be used by owner.



Getting Montage or last index of the array, using EMontageAction value

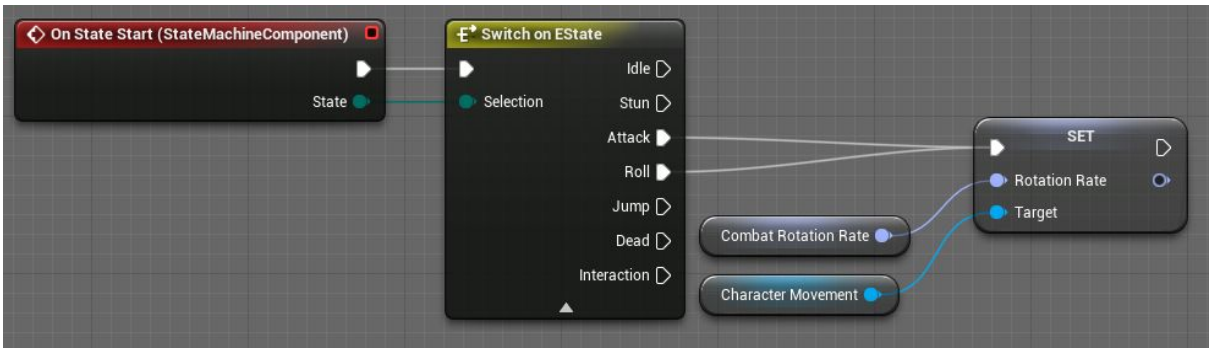

Get Montage Action Last Index
Target is Montages Manager Component
Target
Last Index
Action
LightAttack
Montages Manager Component
Get Montage for Action
Target is Montages Manager Component
Target
Montage
Enumerator
Success
LightAttack
Index
0

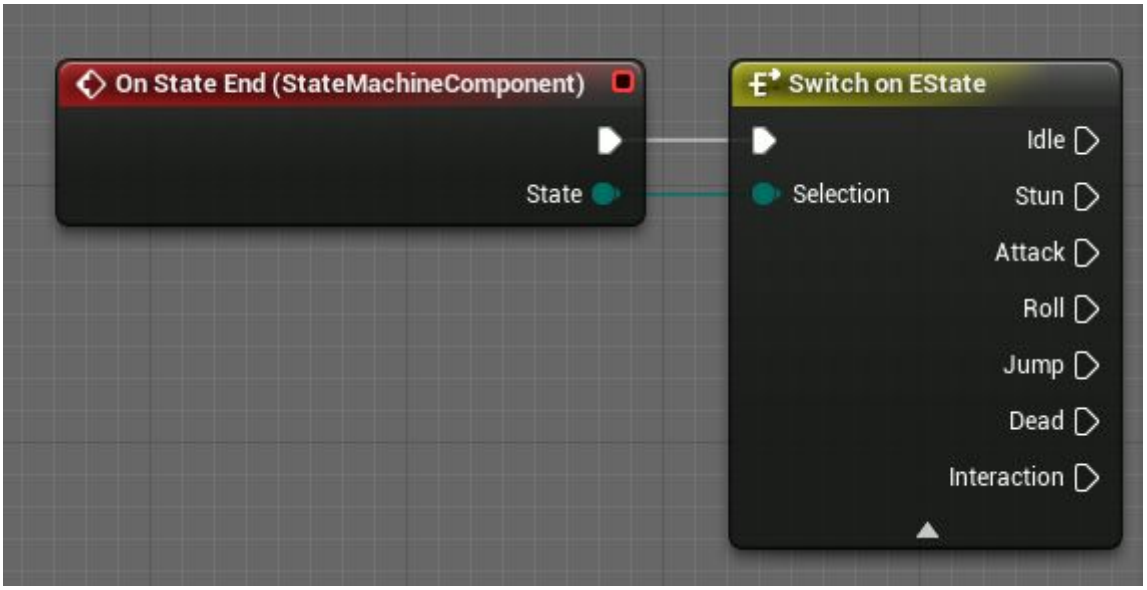

Get Random Montage Index
Target is BPC Montages Manager
Montages Manager
Target
Return Value
Montage Action
LightAttack

StateMachineComponent

Component used by player to define his current state.

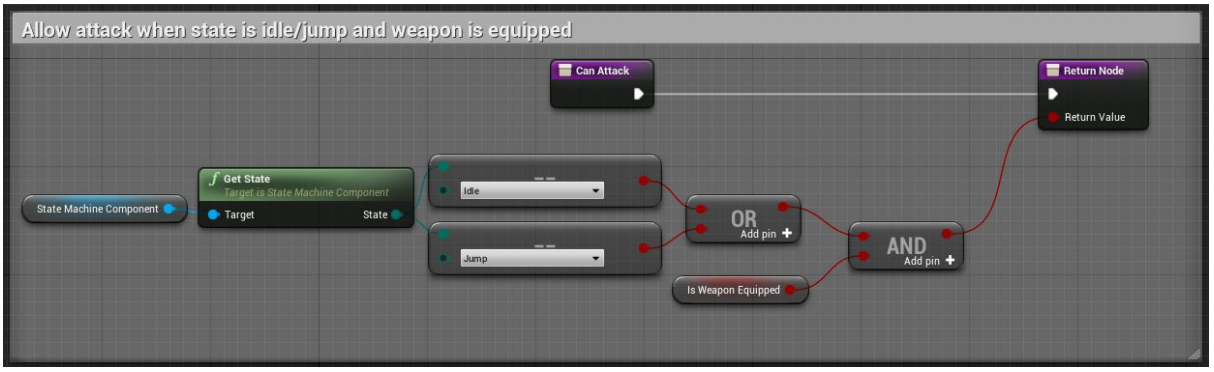
Allow to call additional functions when state ends/starts





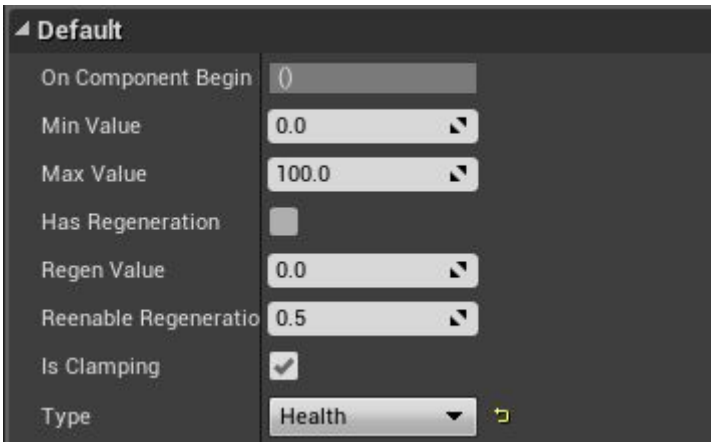
Mostly used to define what player can/can't do in certain states.

e.g. allow attack only in Idle/Air state if weapon is equipped

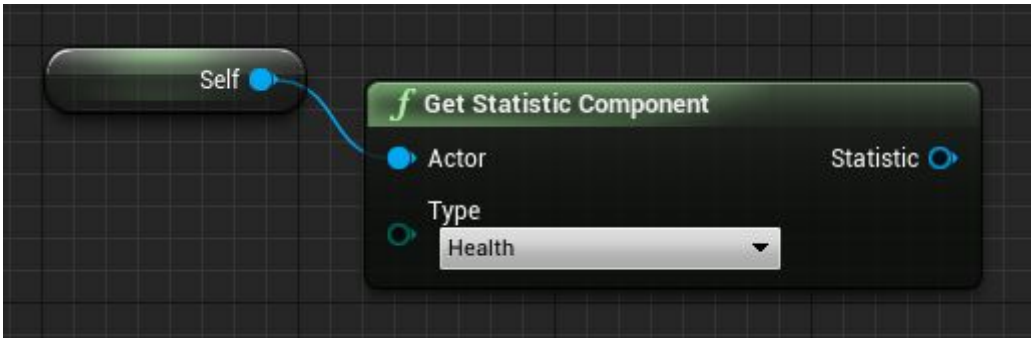


StatisticComponent

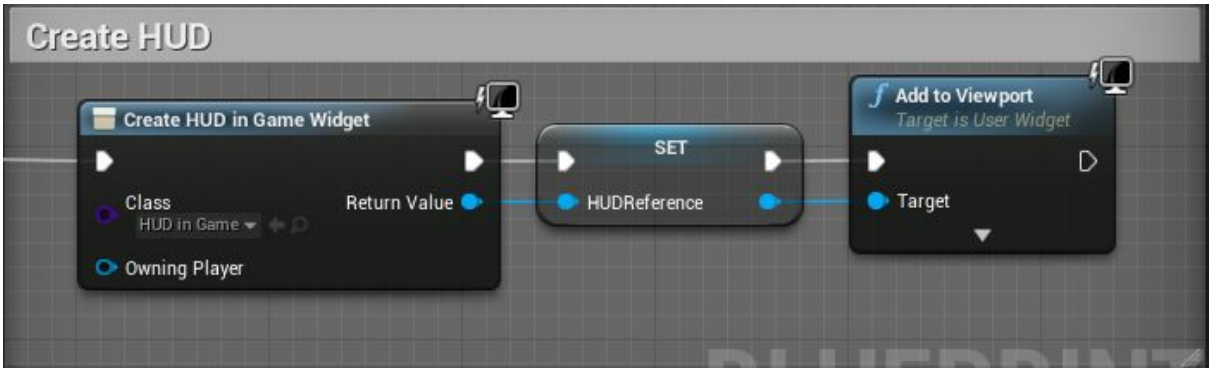
Component used both by player(stamina/health) and AI(health).
New stats like Mana/Exp can be also be added.

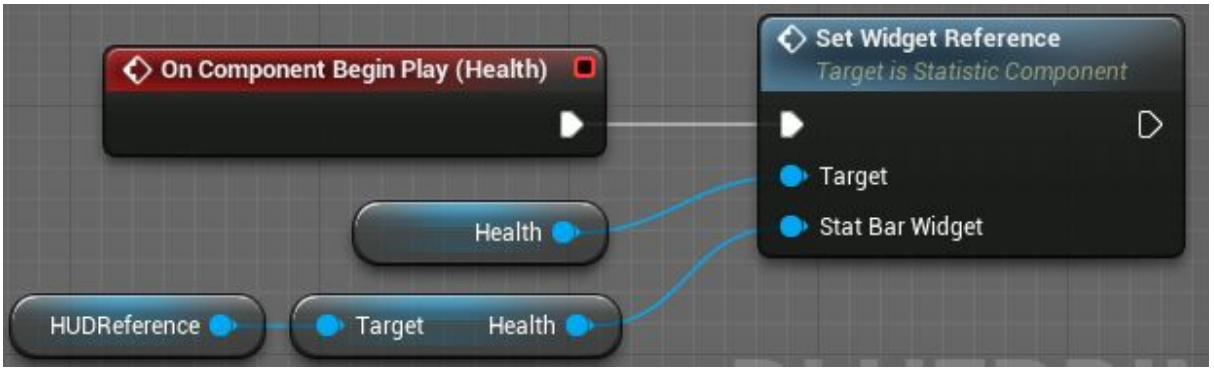


Defining type of component (like Health above) helps to get that component with function ‘GetStatisticComponent’ from global library

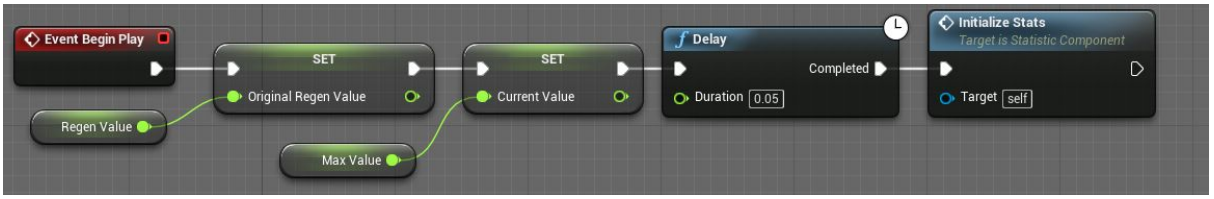


Component requires to set BaseStatBar pointer in order to visualize it.
To do that use function SetWidgetReference



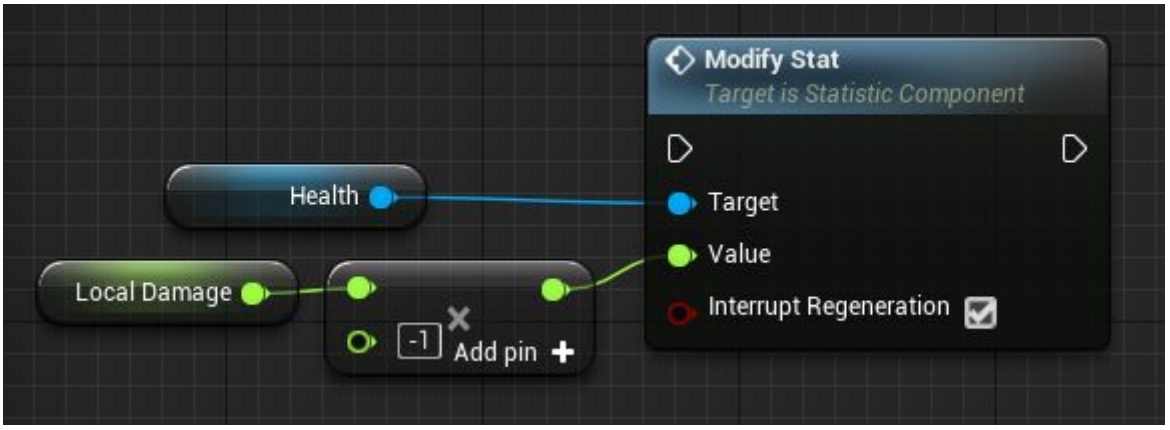


Event OnComponentBeginPlay is called with little delay, to avoid calling it before owner's BeginPlay.



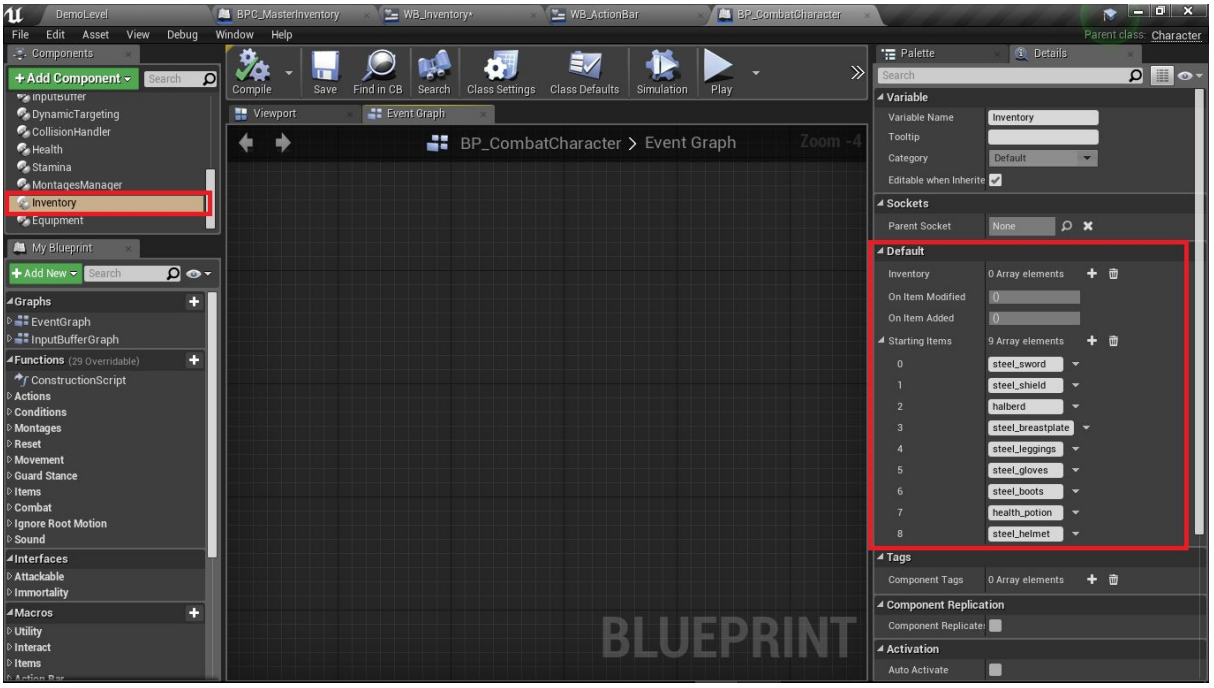
You could also initialize it on your own with above function InitializeStats which is public.

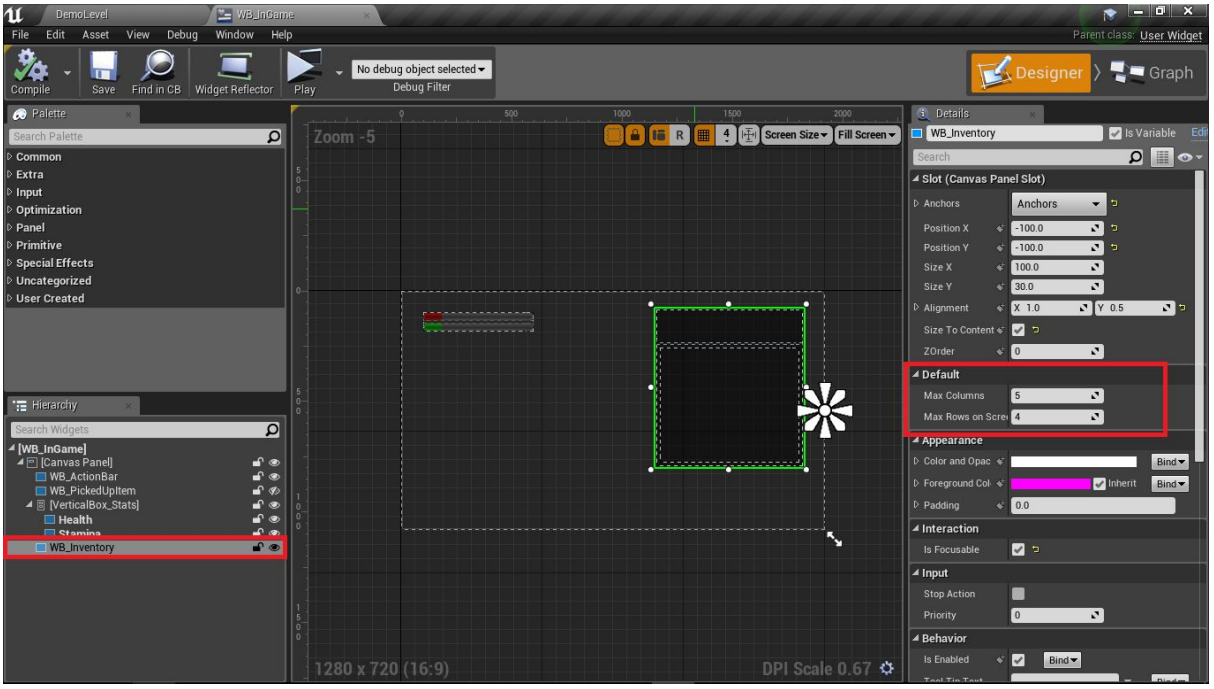
Example of modyfing stat.



InventoryComponent

Component used by player, which allows to store, use and drop items.





To modify size of inventory change MaxColumns/MaxRowsOnScreen values (default 5:4)

Adding new Item Category

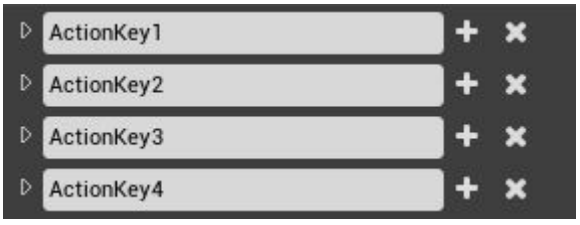
After adding new enum value system will dynamically add new Category button above inventory slots

Action Bar

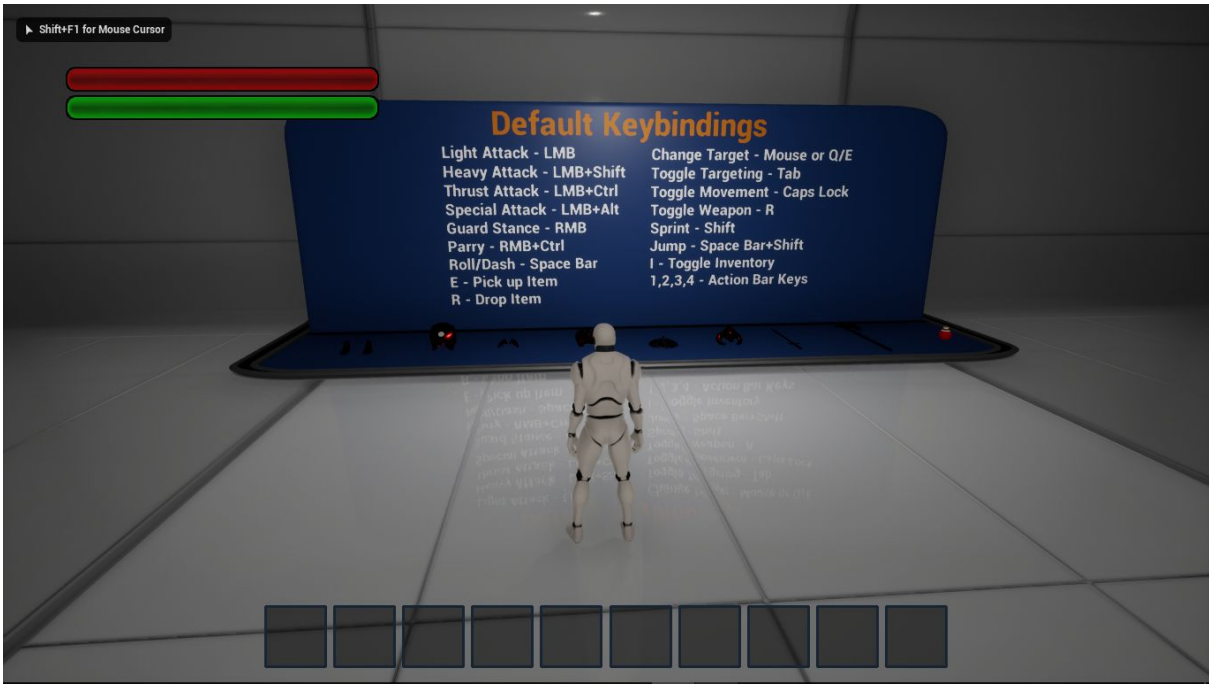
Allows to use items from slots, without opening inventory

By default there are 4 action bar slots, to modify their number open WB_Inventory, select WB_ActionBar and change value of ‘ActionBarSlots’

To be able to use them with input, add new Action Mappings inside Input settings.



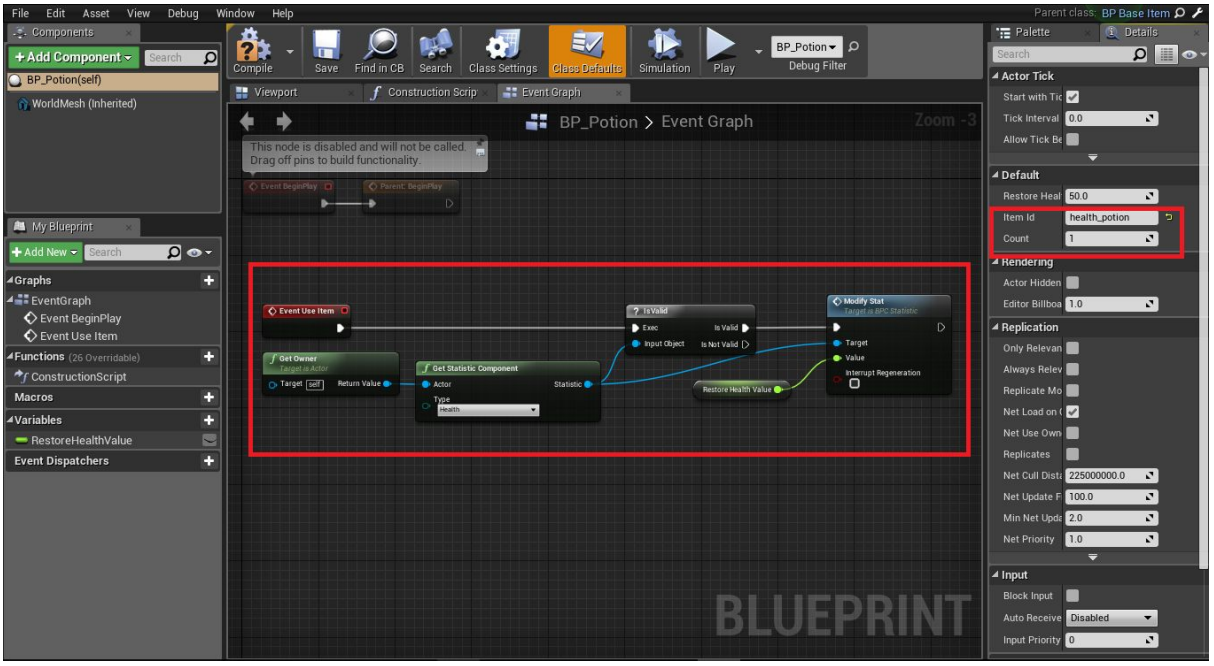
It’s important to call them ActionKeyX, where X means slot number (Starts from 1)



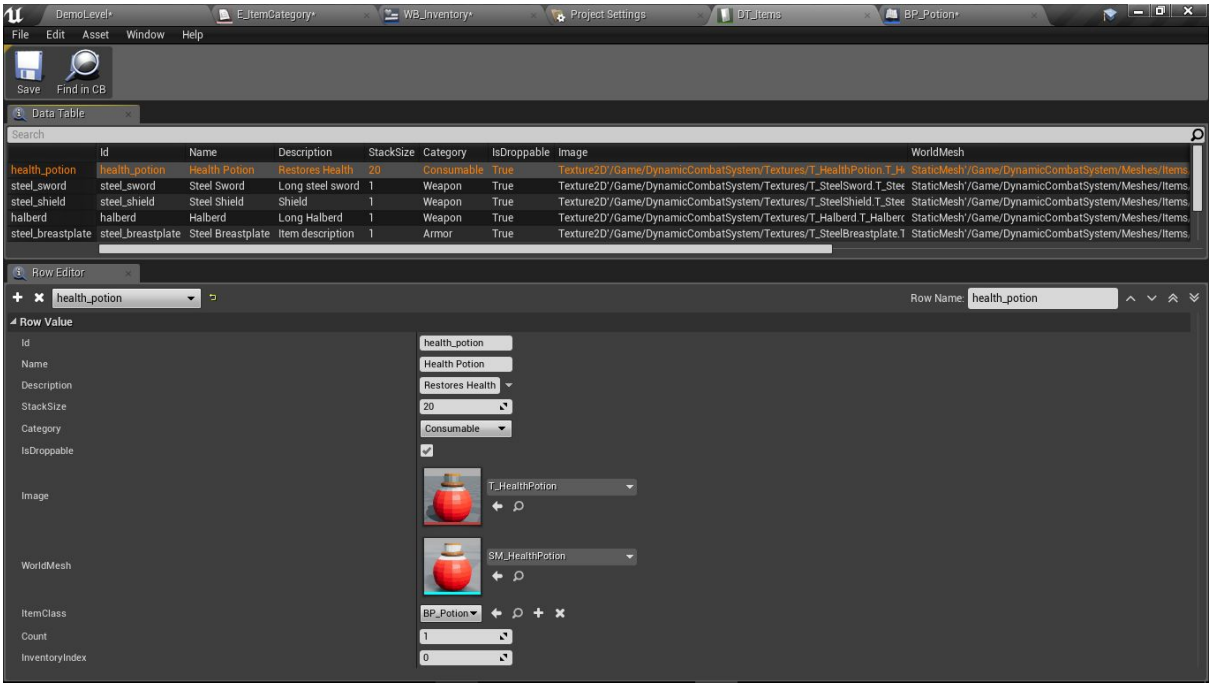
Add New Item

Example: Health Potion
Id: health_potion

Create New Child Blueprint of BP_BaseItem



Set Item Id to health_potion, count to 1 and implement UseItem event
Next create new Row in DT_Items
It’s important to set RowName and Id same as Id in item blueprint (health_potion)



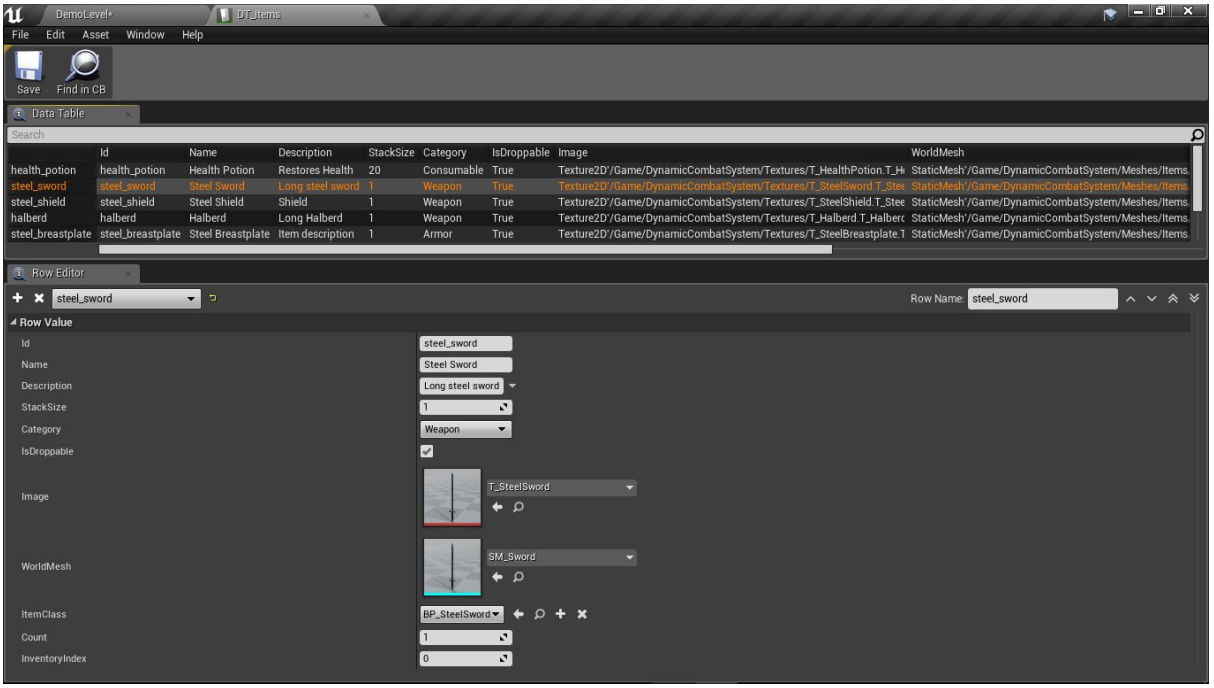
As Item class select previously created BP_HealthPotion
Then set Count to 1, leave InventoryIndex as it is (0)

After that item can be placed in the world, picked and used

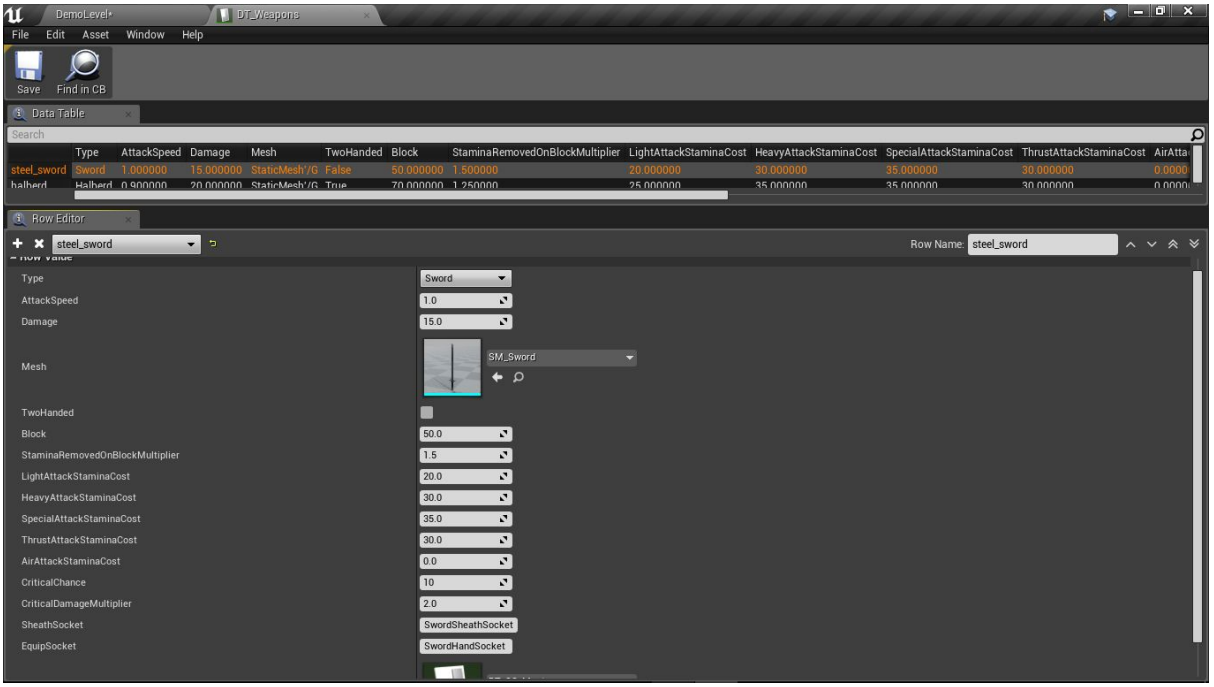
Add New Weapon

Scheme is same as above with few differences.
Instead of creating blueprint from BP_BaseItem, use BP_BaseWeapon (BP_BaseShield for shields)
Don’t override UseItem event there!

After adding item to DT_Items

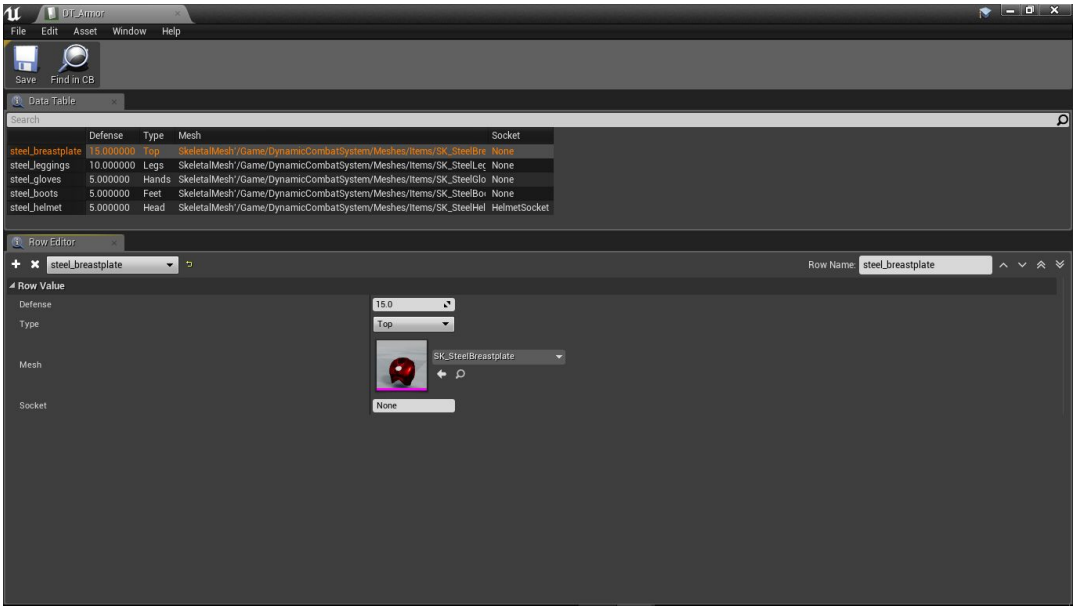


Add it also to DT_Weapons(DT_Shields for shield) (RowName must be the same as Id used in DT_Items)



Add New Armor

Scheme is the same as adding weapon, instead of using adding Row to DT_Weapons, add it to DT_Armor.



Socket name is used just by Helmet to attach it to the character (rest of armor parts are using Character Mesh as Master Pose Component)

As base blueprint classes use
BP_BaseArmor

EquipmentComponent

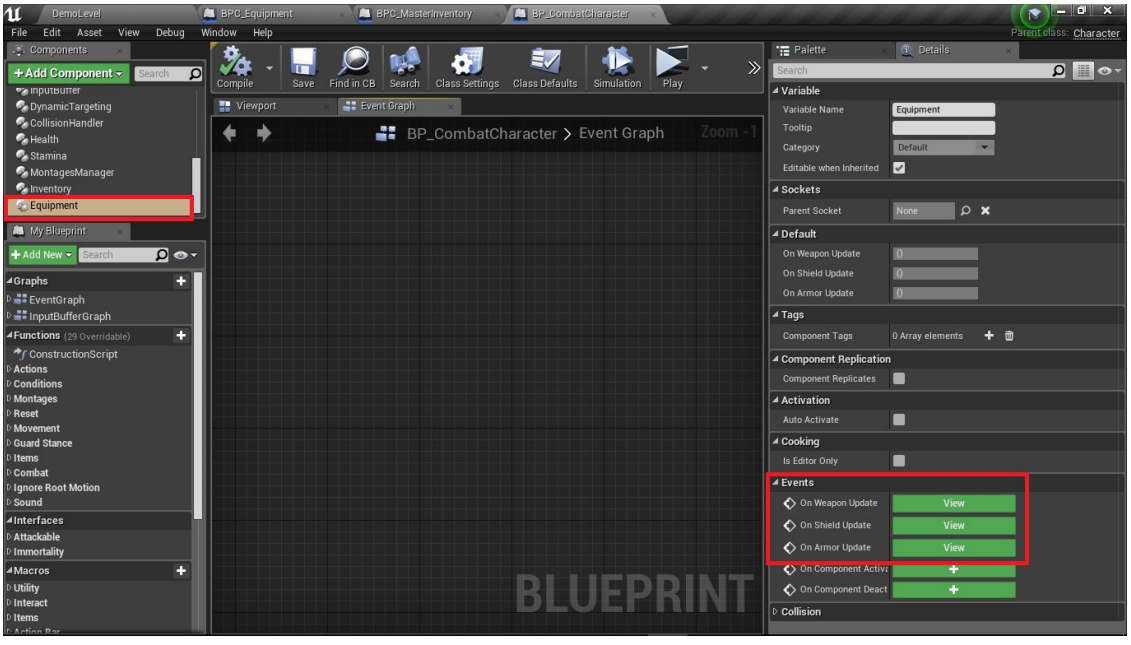
Component used by player, which allows to equip weapon/armor parts.

By default there are 7 item slots:

- Left Hand
- Right Hand
- Head
- Top
- Legs
- Hands
- Feet

Weapons are equipped to Right Hand slot, while shields to Left Hand slot.

Weapon can be 1 or 2 handed.
2-handed weapon can’t be equipped with shield at the same time.

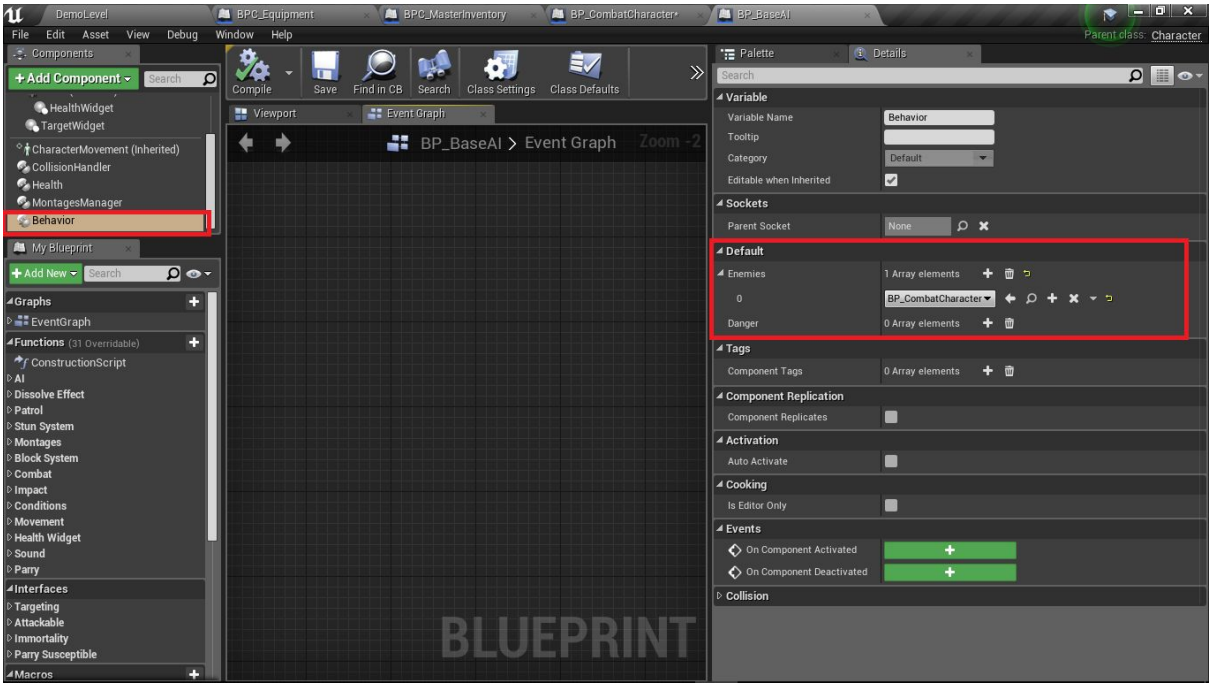


Whenever new item is equipped, one of three available events is called

- OnShieldUpdate
- OnWeaponUpdate
- OnArmorUpdate

BehaviorComponent

Component used by AI, which allows to specify danger/enemy classes



For now only enemies are used in AI controller.

This Component will be extended in next update.

Animation Notifies

AN_IgnoreRootMotion

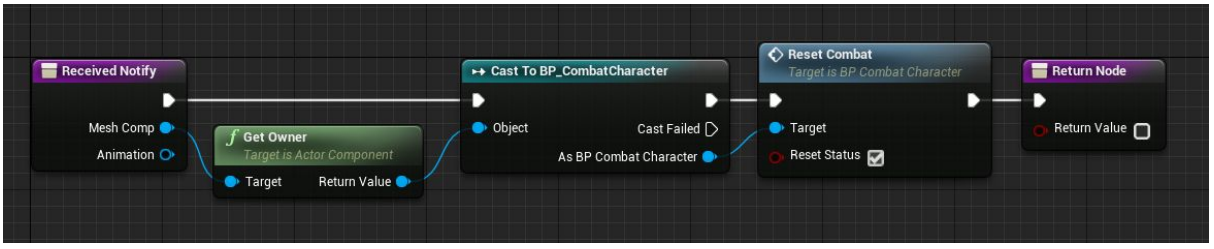
This system is described in Player's IgnoreRootMotion section.

ANS_IgnoreRootMotion

This system is described in Player's IgnoreRootMotion section.

AN_ResetCombat

Notify calling ResetCombat function from player character.
This notify should be used on Impact/Stun player's montage, instead of ANS_InputBuffer.



More info in [InputBuffer](#).

ANS_ParrySusceptible

Described in [Parry](#)

ANS_HitBox

Described in [CollisionHandlerComponent](#).

ANS_InputBuffer

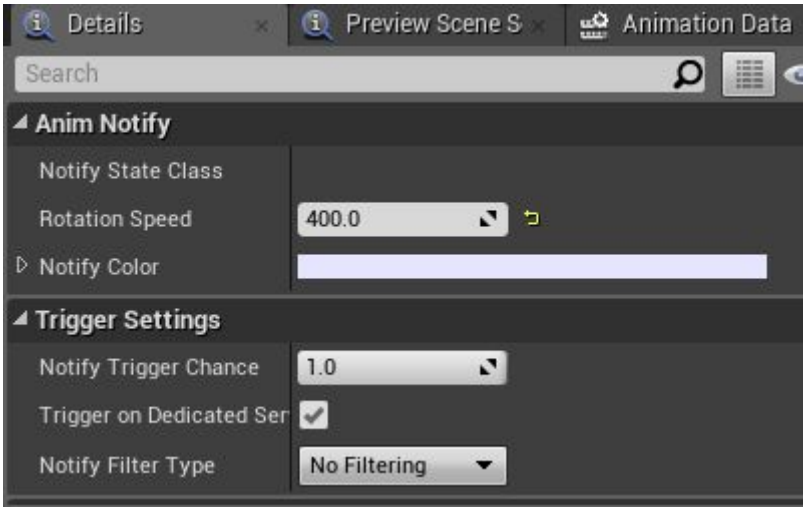
Described in InputBuffer section.

ANS_RotateTowardsTarget

Allows to rotate AI smoothly towards target from blackboard.

Mostly used on attack animations.

Rotation speed(how fast character will be rotating) can be specifed in anim notify section.

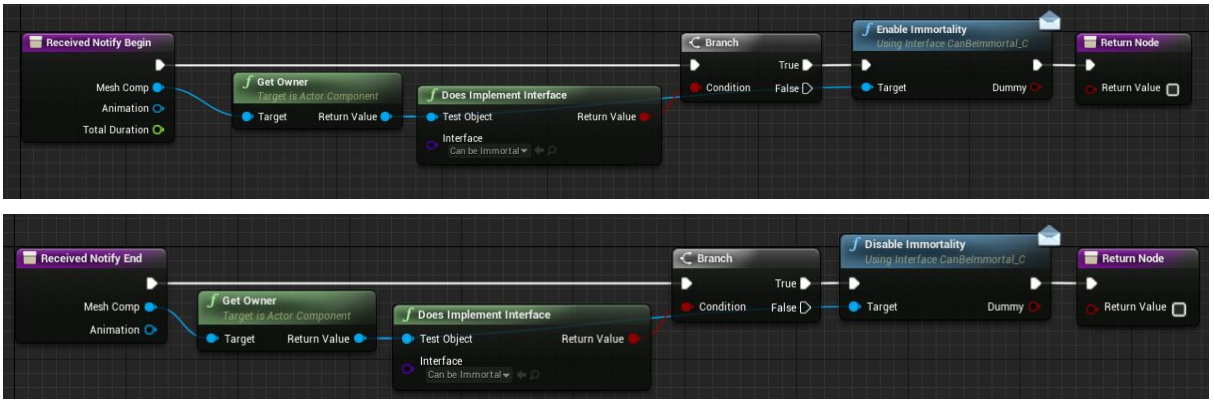


It is recommended to add this state in the moment when character is moving to avoid foot sliding effect.

In example above state is added when character has 1 feet up, with rotation speed of 400.

ANS_ToggleImmortality

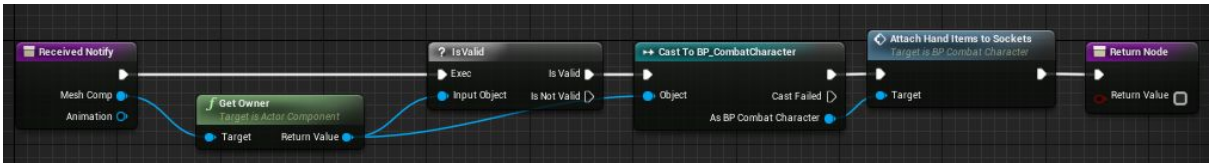
State which calls CanBeImmortal interface functions.
On Begin - EnableImmortality
On End - DisableImmortality



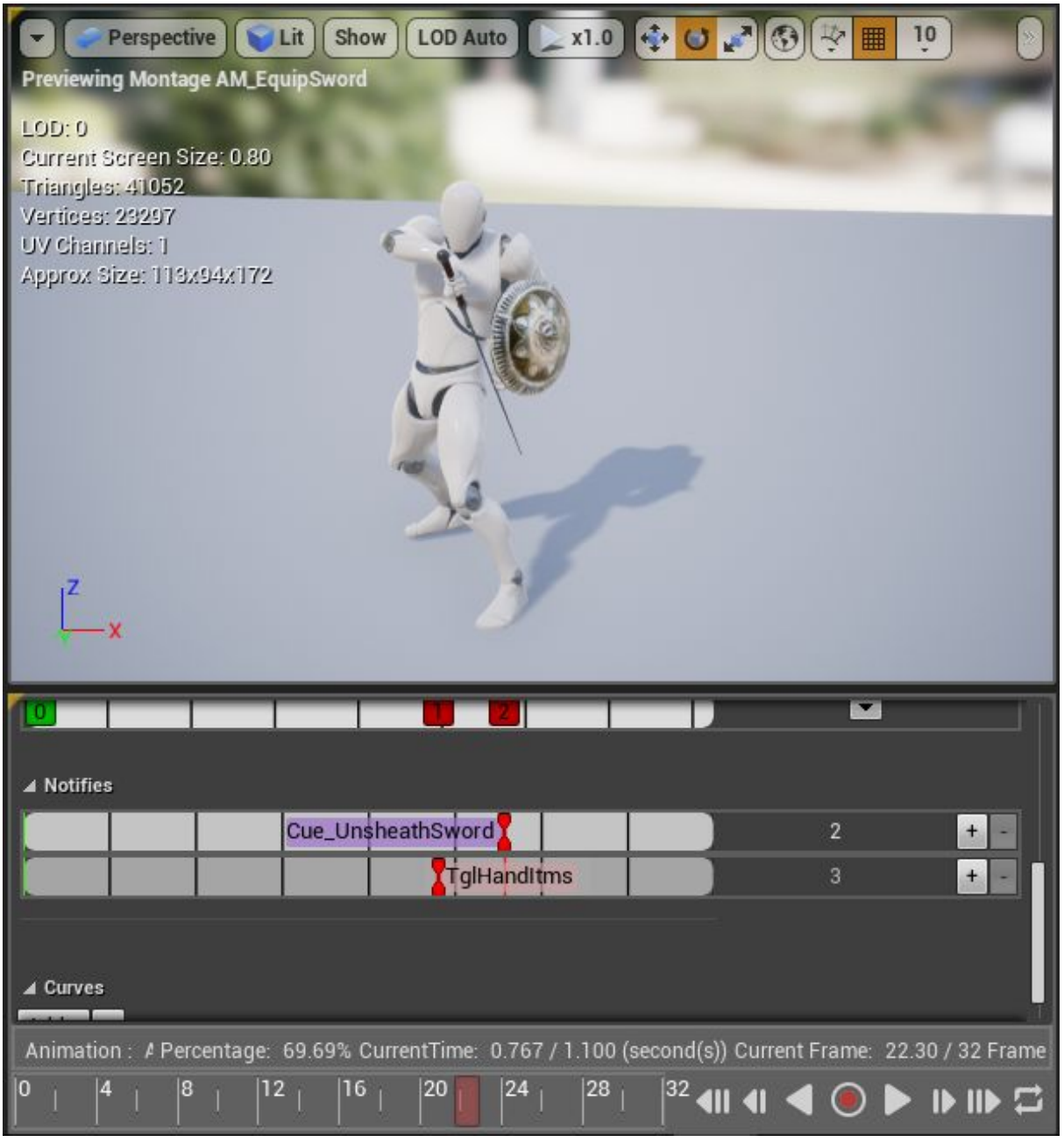
Used on roll/dash montages to make character immortal.

AN_ToggleHandItems

Called on equipping/sheathing montages, used by player.



On received notify calling function ‘AttachHandItemsToSockets’, which will attach right/hand weapon meshes to sheath/hand.


Perspective
Lit
Show
LOD Auto
x1.0
10
Previewing Montage AM_EquipSword
LOD: 0
Current Screen Size: 0.80
Triangles: 41052
Vertices: 23297
UV Channels: 1
Approx Size: 113x94x172
Z
X
Notifies
Cue_UnsheathSword
2
TglHandItms
3
Curves
Animation : Percentage: 69.69% CurrentTime: 0.767 / 1.100 (second(s)) Current Frame: 22.30 / 32 Frame
0
4
8
12
16
20
24
28
32

Enums

EAICombatBehavior

Behaviors of AI while it is in combat state, e.g. approach/strafe/attack etc.
Can be used in behavior tree.

EAIHealthState

Health state of AI, low/medium/high.
Can be used in behavior tree.

EAIPatrolBehavior

Behaviors of AI while it is in combat state, e.g. look around/wait/move to waypoint etc.
Can be used in behavior tree.

EAIStatus

Main status of AI e.g. combat/patrol/Investigate/flee etc.
Can be used in behavior tree.

EAttackType

Used to perform correct attack, modify damage based on type, define which types can stun opponent or be blocked etc.

EInputBufferKey

Used in InputBufferComponent to choose action based on stored key.

EMontageAction

Used by MontageManagerComponent to choose proper montage from DataTable based on given value.
More info in MontageManagerComponent.

ERotationMode

Used by DynamicTargetingComponent.

EState

Used by StateMachineComponent to define current state of the owner. Mainly used in player blueprint.

EStatistic

Used by StatisticComponent to specify type of stat.