Fight Component

Documentation

Documentation version: 1.1.1

(ENG)

1. Introduction:

The **FightComponent** project was specifically designed to provide developers with a lightweight and flexible hand-to-hand combat system, the feature of which is its ease of integration into any project.

This product contains:

Base character class "Character" - The character was based on the base Character class, as well as the basic **CharacterMovement** movement for ease of use.

Health system - Demo health system implemented for example of taking damage.

Damage system - The ability to injure one or more opponents, depending on their distance from the character.

Stealth Assassination System - Thanks to this feature, the character has a chance to quickly and quietly remove his opponent.

Impact blocking system - The basic system of blocking a blow, thanks to which the character receives damage several times less than the original.

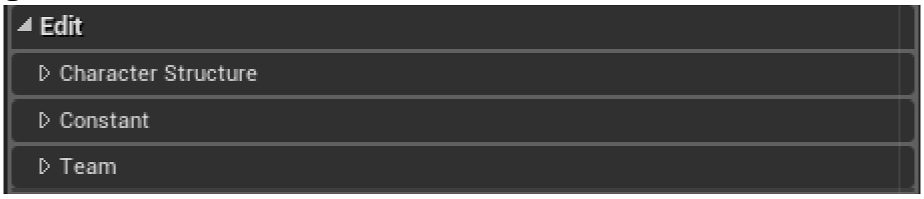
Guidance system - Automatic system of aiming and keeping the target in the character's field of vision.

Artificial Intelligence - A basic AI that can see, hear, search, attack, dodge, block, and stealth kills.

2. Modifiable and immutable parameters of the component FightComponent:

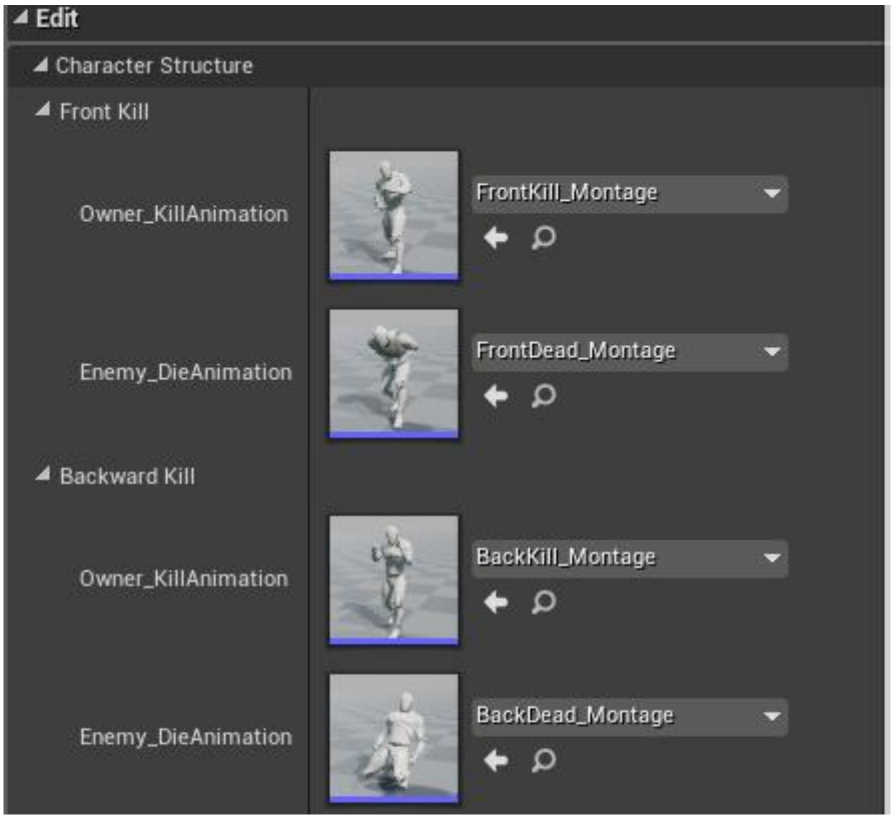
2.1 Edit

This list contains several sub-lists, which contain the main variable parameters of the component. These parameters can be changed both dynamically and before the start of the game.



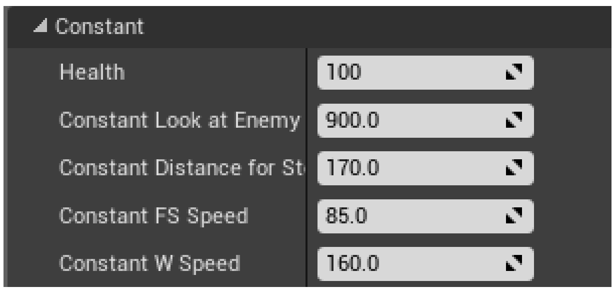
Edit
- Character Structure
- Constant
- Team

Character Structure- Contains two structures responsible for stealth kill animations. In this sublist, you can set or change the stealth kill animations for both the front side and the back.

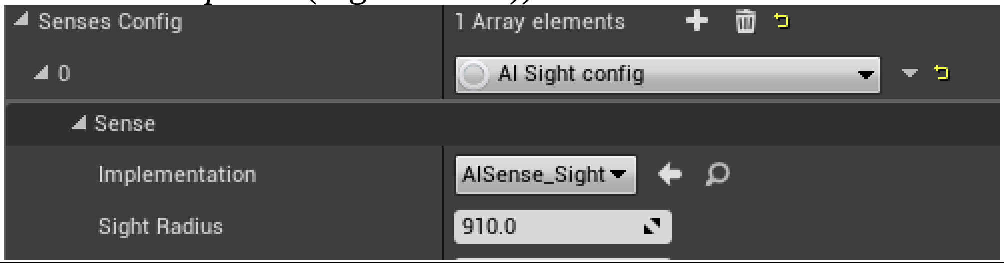


Edit
Character Structure
Front Kill
Owner_KillAnimation: FrontKill_Montage
Enemy_DieAnimation: FrontDead_Montage
Backward Kill
Owner_KillAnimation: BackKill_Montage
Enemy_DieAnimation: BackDead_Montage

Constant- The main variables responsible for life, auto guidance and speed of the character.



Constant	
Health	100
Constant Look at Enemy	900.0
Constant Distance for St	170.0
Constant FS Speed	85.0
Constant W Speed	160.0

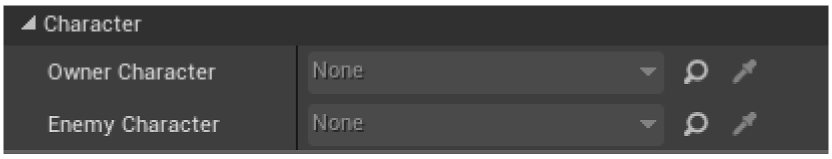
Health - Character lives
Constant look at enemy - The distance at which the character can aim at the enemy (**IMPORTANT!!!** The value of this variable must not exceed the value of the field of view *AiPerception*! (Sight Radius)).  Senses Config — 1 Array elements 0 — AI Sight config Sense Implementation — AISense_Sight Sight Radius — 910.0
Constant Distance for Stealth Kill - The maximum distance between the character and the target, in which it is possible to make a stealth kill.
Constant FS Speed - The maximum speed of movement of the character in a combat stance.
Constant W Speed- Maximum movement speed in a relaxed stance.

Team- Contains an array of friendly names that cannot be targeted or inflicted any damage, and also contains a boolean variable **Friendly Fire**, thanks to which it becomes possible to deal damage to friendly characters, but it remains impossible to target and play a stealth kill.

2.2Non Edit

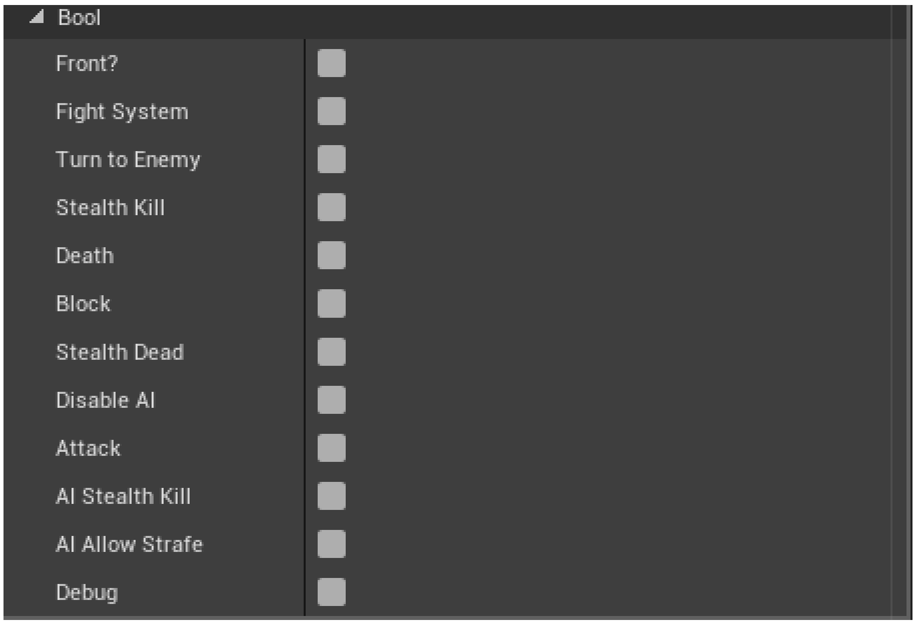
It contains sublists of different types of parameters, into which it is **STRONGLY FORBIDDEN** to make any changes. Changes to these parameters are made automatically depending on the situations and interactions of the character.

Character- Variables that contain the owner of the component (character), as well as the purpose of this owner.



Character
Owner Character — None
Enemy Character — None

Bool- Boolean variables responsible for enabling / disabling certain states of the character.



Bool

Front?	☐
Fight System	☐
Turn to Enemy	☐
Stealth Kill	☐
Death	☐
Block	☐
Stealth Dead	☐
Disable AI	☐
Attack	☐
AI Stealth Kill	☐
AI Allow Strafe	☐
Debug	☐

Front- Variable responsible for choosing the closest side to play a stealth kill.
Fight System -Variable responsible for the state of the character. Whether he is in a fighting stance or relaxed.
Turn to Enemy- Smooth movement of the character to the target for correct reproduction of stealthy murder.
Stealth kill- Variable responsible for the reproduction of a stealth murder. Whether a stealth kill is currently being played.
Death- The state of the character. Alive or dead character.
Block -Variable responsible for the state of blocking strikes. Whether the character is blocking the opponent's attacks at the moment.
Stealth dead- Whether the character is currently a victim of a stealth murder.
Disable AI- Variable that is responsible for disabling / enabling AI. Disabled during stealth kill or death.
Attack- Whether the character attacks with normal attacks.
AI Stealth Kill- A variable that is responsible for the ability to reproduce covert murder by artificial intelligence.
AI Allow Strafe- Can artificial intelligence dodge a blow flying in its direction.
Debug- A variable responsible for enabling / disabling **FightComponent** debugging.

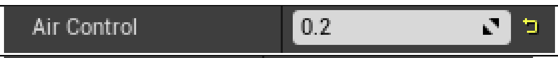
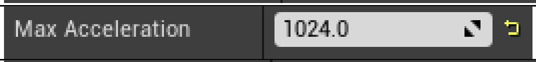
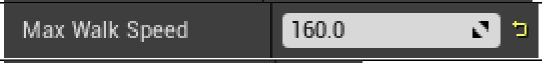
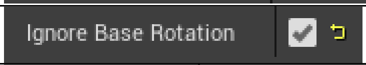
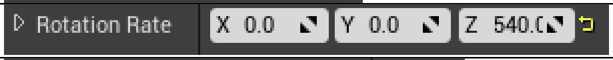
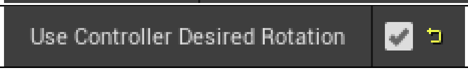
2.3Tags

Component Tags- An array of names that are assigned to a character for convenient division into teams (for example, the **Player** command is assigned to a playable character, which makes it possible to attack all other characters except friendly ones (if **Friendly Fire** is not enabled). You can also assign more than one command, but several (take a look on **Component Tags** of neutral AI)).

3.Parameters of the game character Parent_Player:

3.1Character Movement

In the player character, the base class **CharacterMovement** from *Epic Games* is responsible for the movement system. In this regard, all movement parameters are configured in this component. The following parameters have been changed in the **CharacterMovement** to make the **FightComponent** function correctly:

AirControl= 0.2	 Air Control 0.2
MaxAcceleration= 1024	 Max Acceleration 1024.0
MaxWalkSpeed= 160	 Max Walk Speed 160.0
IgnoreBaseRotation= True	 Ignore Base Rotation ✓
RotationRate= 0.0; 0.0; 540.0;	 Rotation Rate X 0.0 Y 0.0 Z 540.0
UseControllerDesiredRotation= True	 Use Controller Desired Rotation ✓

Also in the game character's **ClassDefaults UseControllerRotationYaw** = False

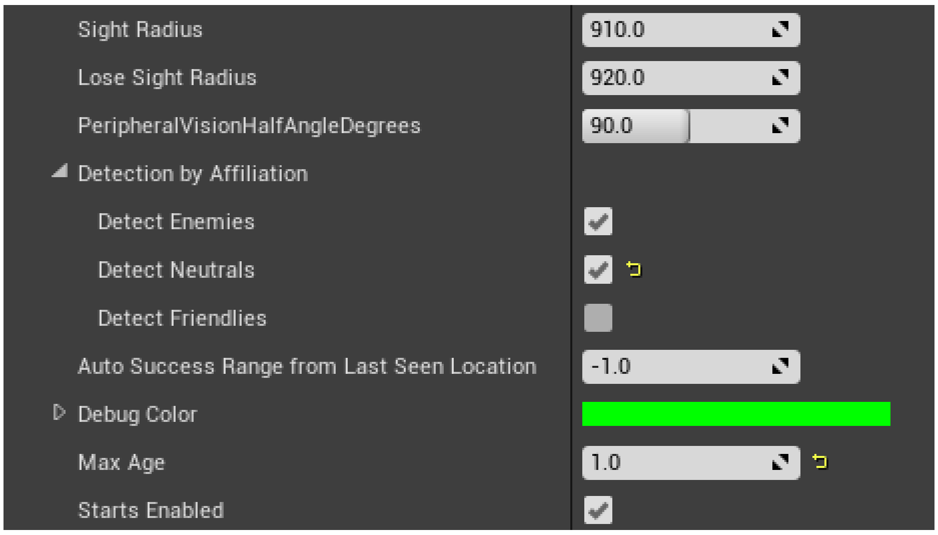
IMPORTANT!!!!
Several **CharacterMovement** parameters change dynamically during gameplay, such as:

MaxWalkSpeed = 160/85	Changed in the **FightComponent** of the boolean variable **FightSystem** depending on which stance the character is in.
UseControllerDesiredRotation = True / False	Changed in **AnimBP_Player** boolean variable **IsAccelerating** depending on the state of movement. Whether the character is standing still or not.

3.2AIPerception

In the playable character, **AIPerception** is only used to find the nearest target and focus on it. For this, only one basic **AISense_Sight** config is set in **SensesConfig**. For comfortable use of the **FightComponent**, the following values were chosen:

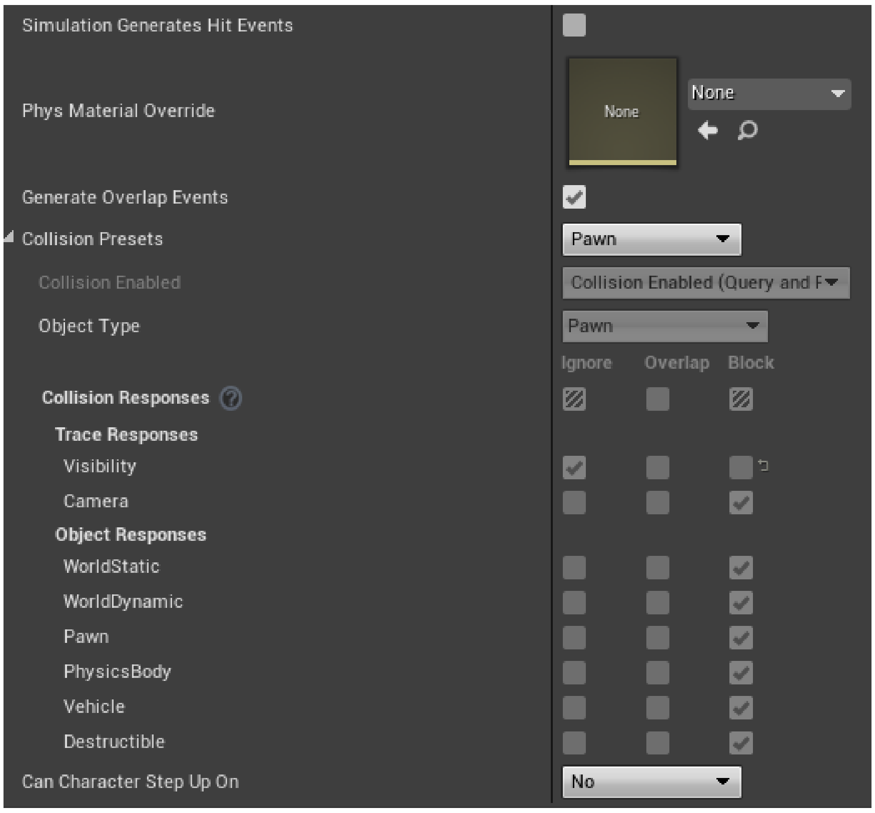
SightRadius -910.0
LoseSightRadius -920.0
PeripheralVisionHalfAngleDegrees -90.0
DetectionByAffiliation - Detect Enemies - True Detect Neutrals - True Detect Friendlies - False
AutoSuccessRangeFromLastSeenLocation --1.0
MaxAge --1.0
StartsEnabled - True



Sight Radius	910.0
Lose Sight Radius	920.0
PeripheralVisionHalfAngleDegrees	90.0
Detection by Affiliation	
Detect Enemies	✓
Detect Neutrals	✓
Detect Friendlies	☐
Auto Success Range from Last Seen Location	-1.0
Debug Color	🟩
Max Age	1.0
Starts Enabled	✓

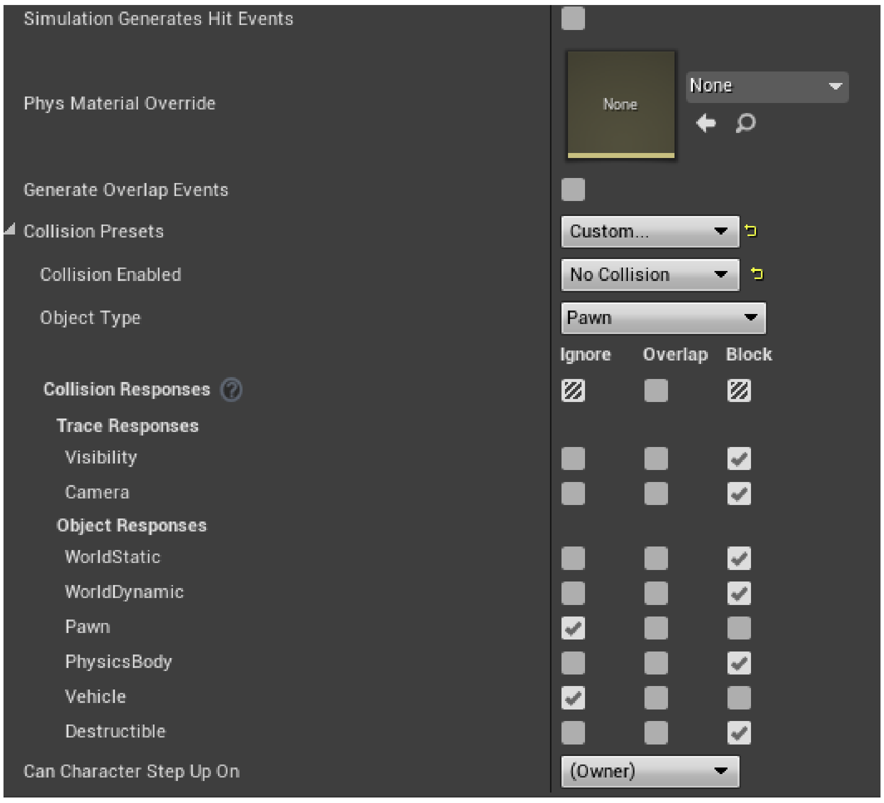
3.3 collision

The **CapsuleComponent** collision has not been changed and is defaulted.



Simulation Generates Hit Events			
Phys Material Override	None		
Generate Overlap Events	✓		
Collision Presets	Pawn		
Collision Enabled	Collision Enabled (Query and F		
Object Type	Pawn		
	Ignore	Overlap	Block
Collision Responses	▨		▨
Trace Responses			
Visibility	✓		
Camera			✓
Object Responses			
WorldStatic			✓
WorldDynamic			✓
Pawn			✓
PhysicsBody			✓
Vehicle			✓
Destructible			✓
Can Character Step Up On	No		

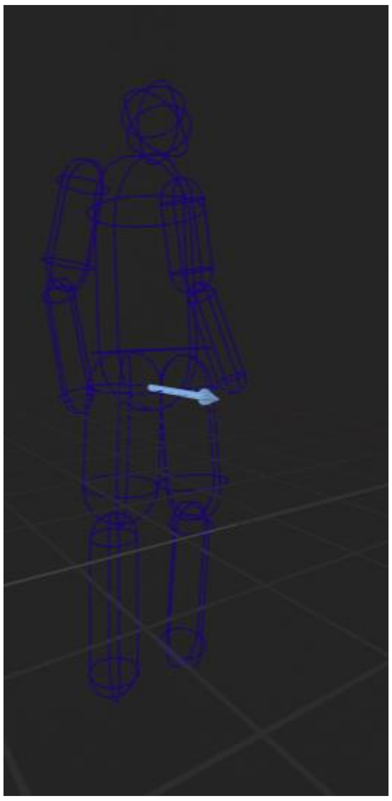
Mesh collision was also almost unaffected, except for disabling collision in **CollisionEnabled** = No Collision.



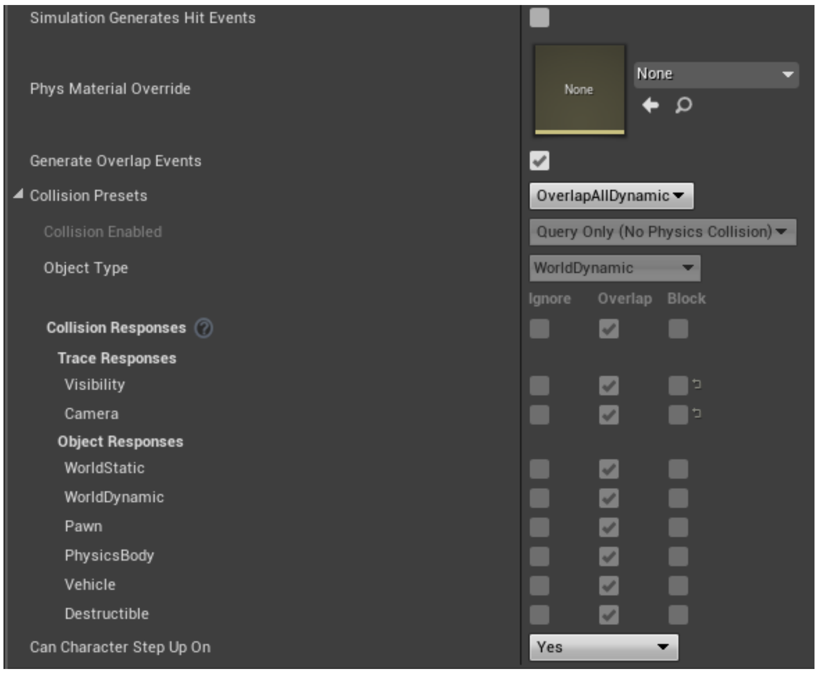
Simulation Generates Hit Events			
Phys Material Override	None		
Generate Overlap Events			
Collision Presets	Custom...		
Collision Enabled	No Collision		
Object Type	Pawn		
	Ignore	Overlap	Block
Collision Responses	▨		▨
Trace Responses			
Visibility			✓
Camera			✓
Object Responses			
WorldStatic			✓
WorldDynamic			✓
Pawn	✓		
PhysicsBody			✓
Vehicle	✓		
Destructible			✓
Can Character Step Up On	(Owner)		

3.4 Hitbox

Hitboxes are used to register a hit on a character or enemy. Own hitboxes were created in order to optimize the calculation of the target hit.



Hitbox collision is set to register absolutely all objects overlapping them.



Simulation Generates Hit Events	☐		
Phys Material Override	None		
Generate Overlap Events	✓		
Collision Presets	OverlapAllDynamic		
Collision Enabled	Query Only (No Physics Collision)		
Object Type	WorldDynamic		
	Ignore	Overlap	Block
Collision Responses	☐	✓	☐
Trace Responses			
Visibility	☐	✓	☐
Camera	☐	✓	☐
Object Responses			
WorldStatic	☐	✓	☐
WorldDynamic	☐	✓	☐
Pawn	☐	✓	☐
PhysicsBody	☐	✓	☐
Vehicle	☐	✓	☐
Destructible	☐	✓	☐
Can Character Step Up On	Yes		

There are 10 hitboxes on the character's entire body for full coverage.

Hitbox_Body
Hitbox_Upperarm_R
Hitbox_Lowerarm_R
Hitbox_Head
Hitbox_Upperarm_L
Hitbox_Lowerarm_L
Hitbox_Calf_R
Hitbox_Thigh_R
Hitbox_Calf_L
Hitbox_Thigh_L

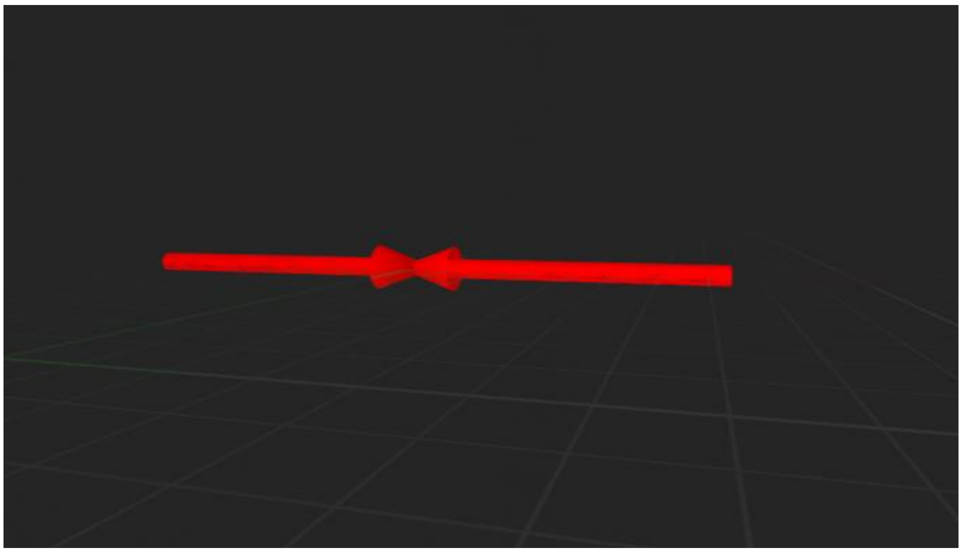
Hitbox_Body
Hitbox_Upperarm_R
Hitbox_Lowerarm_R
Hitbox_Head
Hitbox_Upperarm_L
Hitbox_Lowerarm_L
Hitbox_Calf_R
Hitbox_Thigh_R
Hitbox_Calf_L
Hitbox_Thigh_L

3.5ArrowComponent

Arrows attached to the **CapsuleComponent** serve as the position and direction for the stealth kill. In the game character, as well as in the AI, there are only two of them:

Front- the front side of the character.
Back- the back of the character.

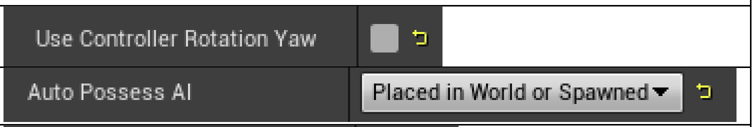
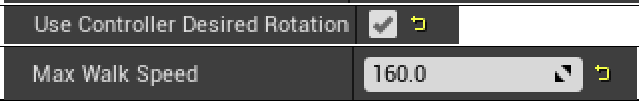
These arrows can be much more, depending on the types of stealth killings and their directions.



4. Artificial intelligence:

4.1Parent_AI

The AI was based on the base **Character** class from "*Epic Games*". **Parent_AI** also uses the base controller **AIController** (All the logic for launching and operating the AI is written in the **Parent_AI** itself). In this AI class, the basic parameters that the **FightComponent** uses have been set and configured.

FightComponent	
AIPerception	
ArrowComponent (Front; Back)	
BI_Fight	
Custom **ClassDefaults**	 Use Controller Rotation Yaw ☐ ; Auto Possess AI: Placed in World or Spawned
Custom **CharacterMovement**	 Use Controller Desired Rotation ✓ ; Max Walk Speed: 160.0

Since this is the parent AI class and does not have a **Mesh**, there is no hitbox collision.

4.1.1 Parent_AI variables

The parent AI class **Parent_AI** has only two available and editable variables.

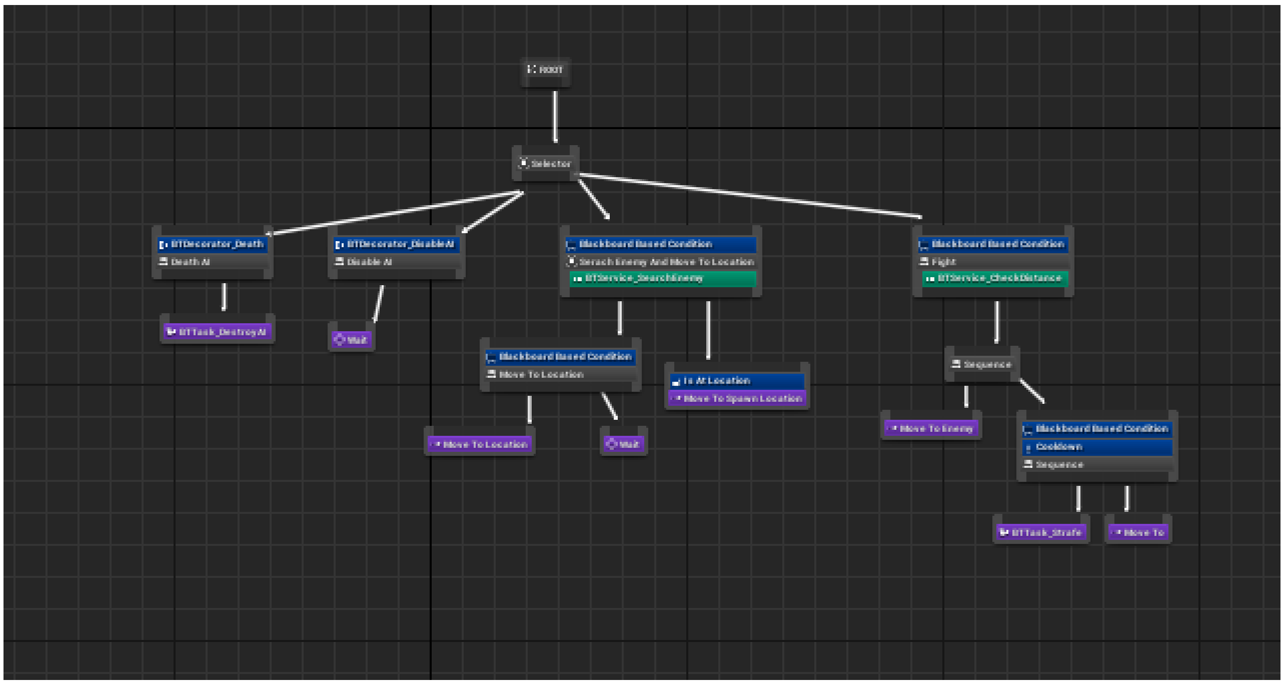
AI_AllowStrafe- boolean variable that allows / prohibits the AI to evade enemy strikes.
AI_SteathKill Is a boolean variable that allows / disables the AI to play stealth kill.

4.2 AI brain

It was decided to use **Behavior Tree** and **Blackboard** as the basis for AI control.

4.2.1AI_BT (Behavior Tree)

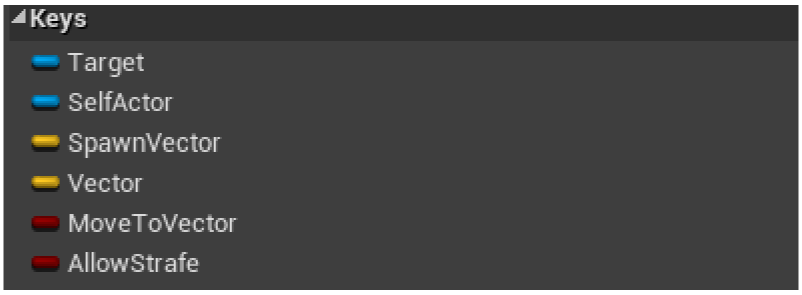
In this tree of behavior, you can see all the capabilities of Artificial Intelligence, such as: **death, shutdown, search, attack**.



4.2.2AI_BB (Blackboard)

All keys used can be seen on this AI board.

Target - Purpose.
Self -Link to yourself.
SpawnVector- the place where the AI appeared.
Vector -dynamic location variable where the AI can go (e.g. rustle).
MoveToVector -the variable responsible for the ability to get to the location, which is set in the **Vector** variable.
AllowStrafe- the ability to evade enemy strikes.



Keys
Target
SelfActor
SpawnVector
Vector
MoveToVector
AllowStrafe

4.2.3 AI Decorators

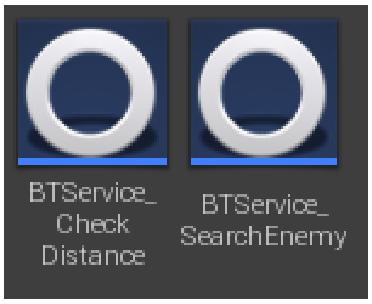
In the **FightComponent** project, the AI has two decorators that enable or disable the behavior of the AI.

BTDecorator_Death- Checks if the AI is alive.
BTDecorator_DisableAI- Checks the AI for activity (for example, the AI is disabled during stealth kill or stealth death).

4.2.4 AI Services

In the **FightComponent** project, the AI has two services that execute the secondary AI logic.

BTService_CheckDistance- Checks the distance to the target and attacks.
BTService_SearchEnemy- Looking for a new closest target.



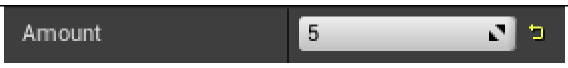
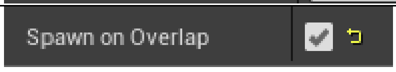
4.2.5 AI Tasks

In the **FightComponent** project, the AI has two tasks that perform the basic logic of the AI.

BTTask_DestroyAI- Disables and removes AI upon death.
BTTask_Strafe- Finds a new position before dodging a blow.

4.3 AI Spawner

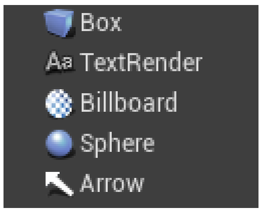
The **FightComponent** project also contains special AI spawners with some tweaks.

Amount- the number of AI that will be spawned (max - 5).	 Amount 5
AI- AI choice (e.g. friendly).	AI AI_Human_Friendly
SpawnOnOverlap- AI spawn after the character crosses the trigger.	 Spawn on Overlap ✓

Also, all AIs that have been spawned will be written to a separate **AIMassive** array for easy manipulation with them.

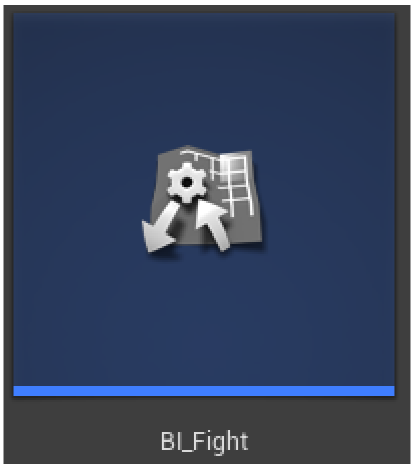
The AI spawner includes components such as:

Box - an area of a certain size on which the AI can be spawned.
TextRender - the name of the spawner.
Billboard - spawner icon
Sphere - a trigger upon crossing which the AI will spawn.
Arrow - the direction of spawn AI

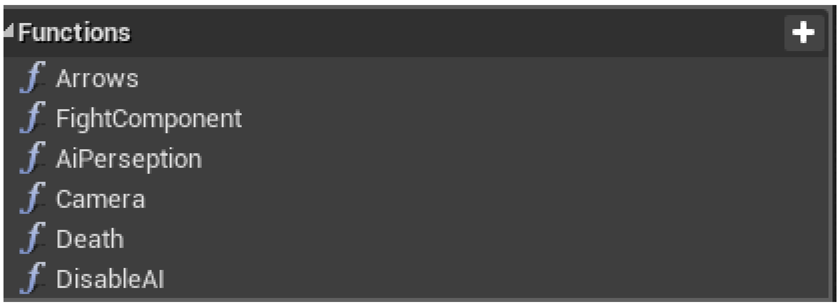


5.Blueprint interface “BI_Fight”:

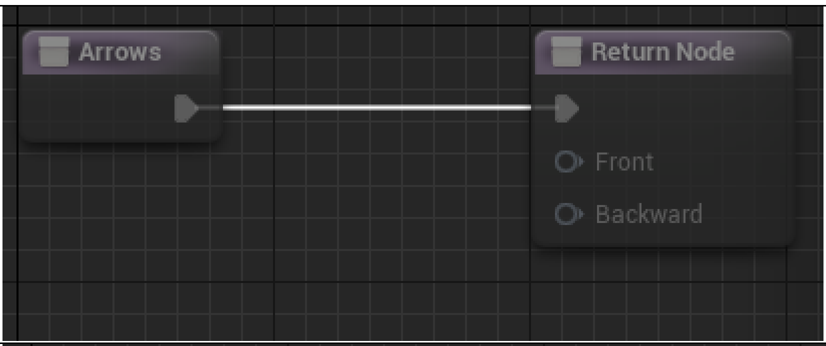
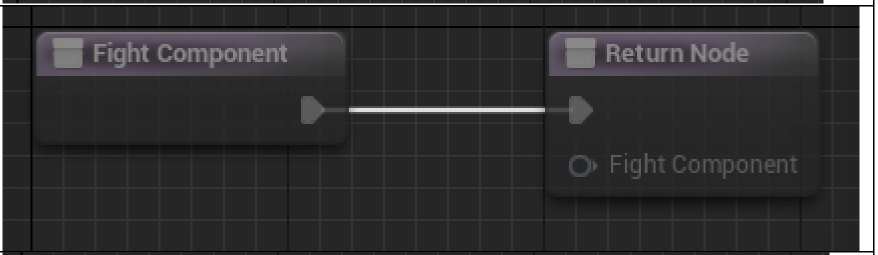
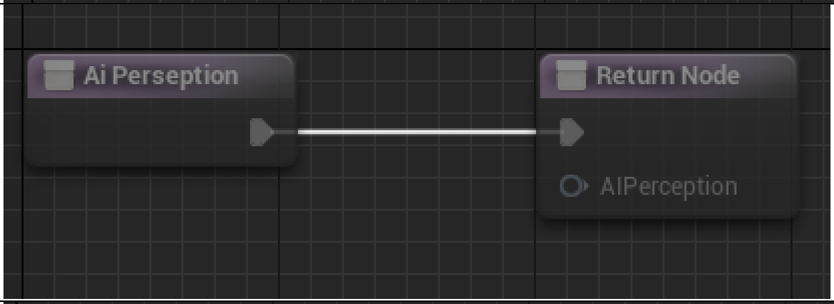
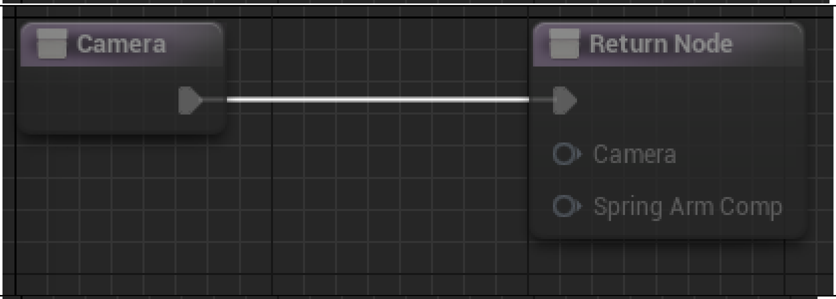
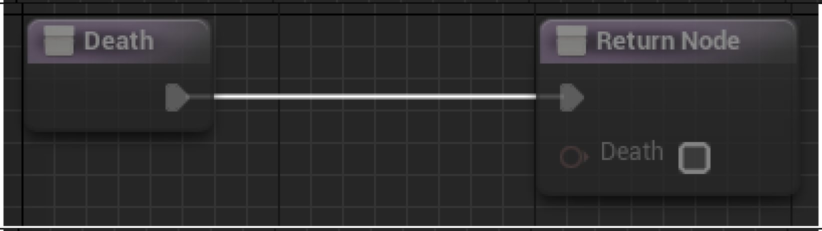
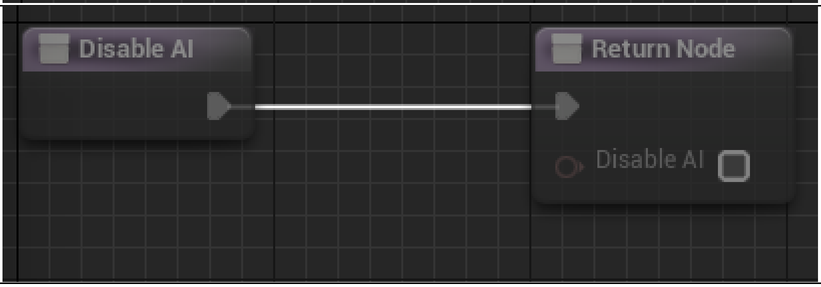
This interface is used for easy access to commonly used components and parameters of a character or target.



The interface has only 6 required functions.



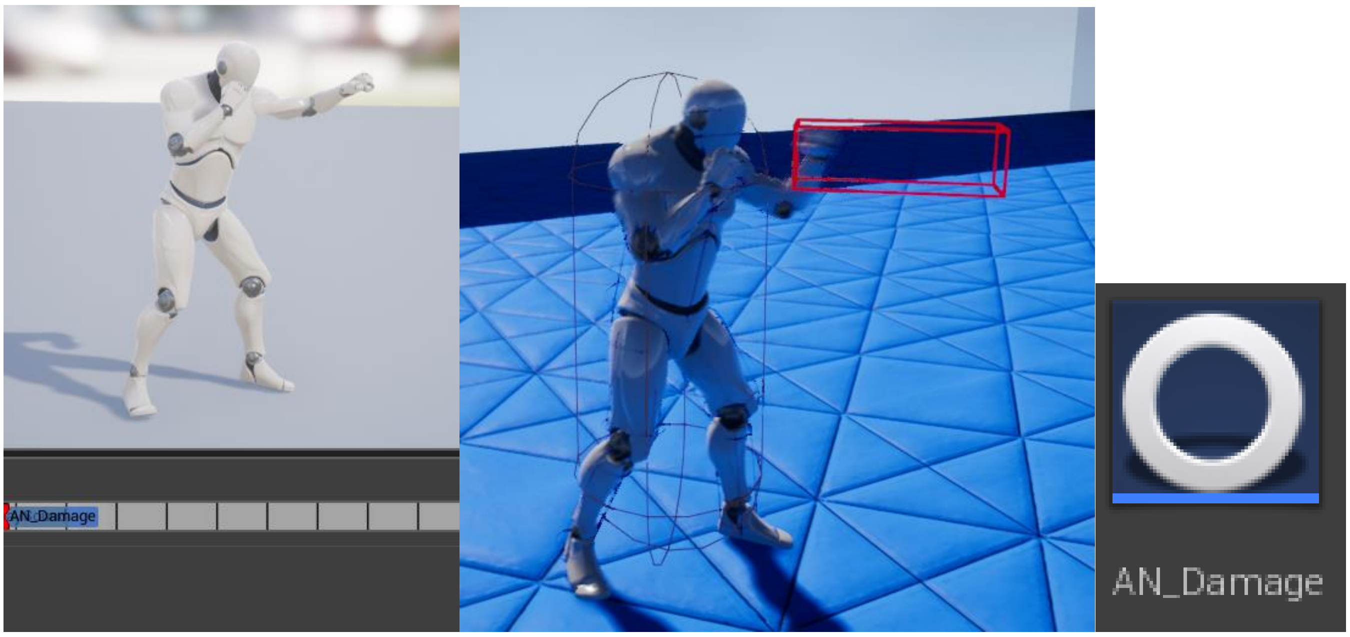
Functions
Arrows
FightComponent
AiPerseption
Camera
Death
DisableAI

Arrows- function for passing direction arrows **ArrowComponent** (In this case, **Front** and **Back**)	 Arrows → Return Node: Front, Backward
FightComponent- a function to pass the entire **FightComponent.**	 Fight Component → Return Node: Fight Component
AIPerseprion- a function to pass the entire component **AIPerception**	 Ai Perseption → Return Node: AIPerception
Camera- function for passing **Camera** and **SpringArmComp**	 Camera → Return Node: Camera, Spring Arm Comp
Death- Function for passing a boolean parameter of the character's death.	 Death → Return Node: Death
DisableAI- a function for passing a boolean parameter to disable the AI. (Only used in AI)	 Disable AI → Return Node: Disable AI

6.Anim Notify:

6.1 AN_Damage

AN_Damage- is responsible for the appearance of collision impact (Calls **DamageRadius** in **FightComponent**) and is located in animations.



Also **AN_Damage** has a parameter - a structure that includes

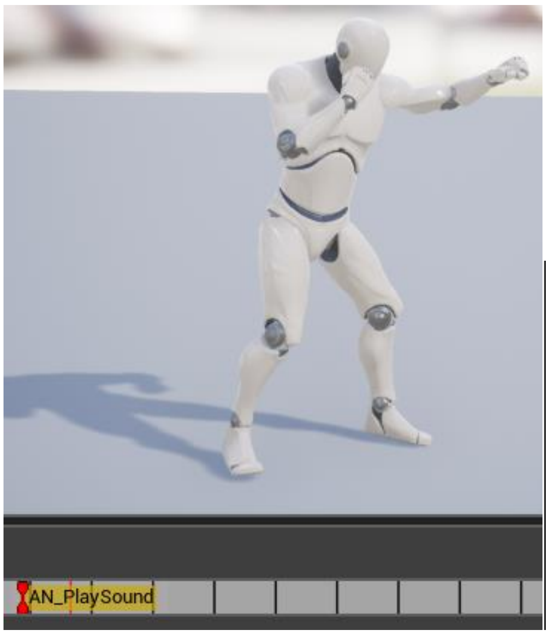
BoxExtent- The size of the collision.
Distance- Distance of the collision appearance from the initiator.
Height- The height of the collision appearance from the initiator.
Damage- Damage.
DirectionOfImpact- Impact side.

6.2 AN_PlaySound

AN_PlaySound- is located in animations and is responsible for the appearance of surround sound in the world.

It includes parameters such as:

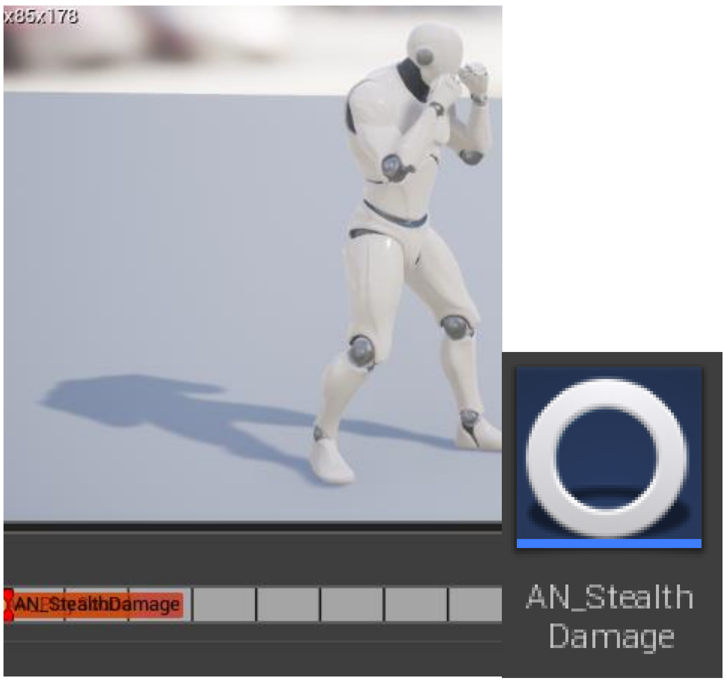
WillAIHear?- Can the AI hear this sound?
Loudness - Sound volume.
MaxRange - Maximum sound range.
Sound- Sound.
VolumeMultiplier- Volume multiplier.
PitchMultiplier- Pitch multiplier.
SwitchNameInCue- **Switch** titles in the installed **Cue**.
SwitchType- Sound selection.



6.3 AN_StealthDamage

AN_StealthDamage- Located in animations and is responsible for the damage done to the target during a stealth kill (Calls **ApplyDamage** from **FightComponent**). Inside it is a parameter such as:

Kill?- Kills the target in the final hit of a stealth kill.



7.Animation Blueprint:

The **FightComponent** project uses the animation blueprint **AnimBP_Parent_Character** to control character animations. It contains the parameters and logic required for proper functioning.

7.1 EventGraph

Inside **EventGraph** is the main logic for initializing and updating parameters. In initialization, the main variables are set, such as:

Character- Link to the owning character.
FightComponent- Link to **FightComponent**.

The following basic variables are updated every frame:

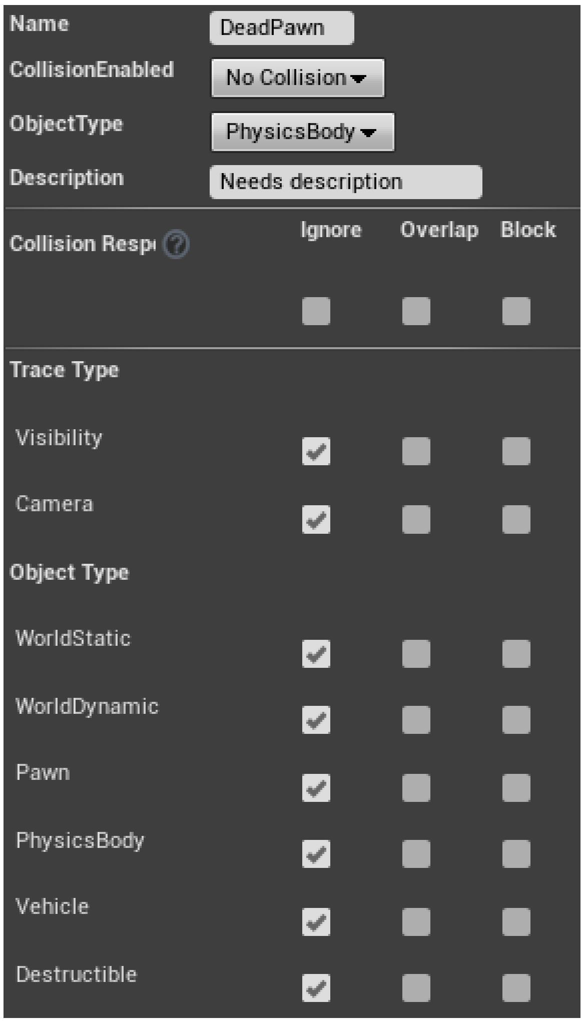
IsInAir? - Boolean variable that checks if the character is in the air.
IsAccelerating- Boolean variable that checks if the character is moving.
FightSystems- Boolean variable that checks if the character is in combat stance.
Velocity -Velocity change in vector.
Direction- Direction of movement.
Speed- Speed.
Lean- Leaning in the direction of the jump.

7.2 AnimGraph

Inside **AnimGraph** there are animations that change depending on the state of the variables.
Locomotion SM is a *State Machine*, inside of which the main movement animations are located. **Cache Pose** is used to play *AnimMontage* (in our case, this is animation of strikes, blocks and stealth kill).

8.Own collision profile:

For the correct functioning of the **FightComponent** project, it was decided to add its own **DeadPawn** collision profile, which is used after the death of the character. This profile does not have any collision and ignores all objects that intersect it.



Name	DeadPawn
CollisionEnabled	No Collision
ObjectType	PhysicsBody
Description	Needs description

Collision Resp	Ignore	Overlap	Block
	☐	☐	☐
Trace Type			
Visibility	✓	☐	☐
Camera	✓	☐	☐
Object Type			
WorldStatic	✓	☐	☐
WorldDynamic	✓	☐	☐
Pawn	✓	☐	☐
PhysicsBody	✓	☐	☐
Vehicle	✓	☐	☐
Destructible	✓	☐	☐

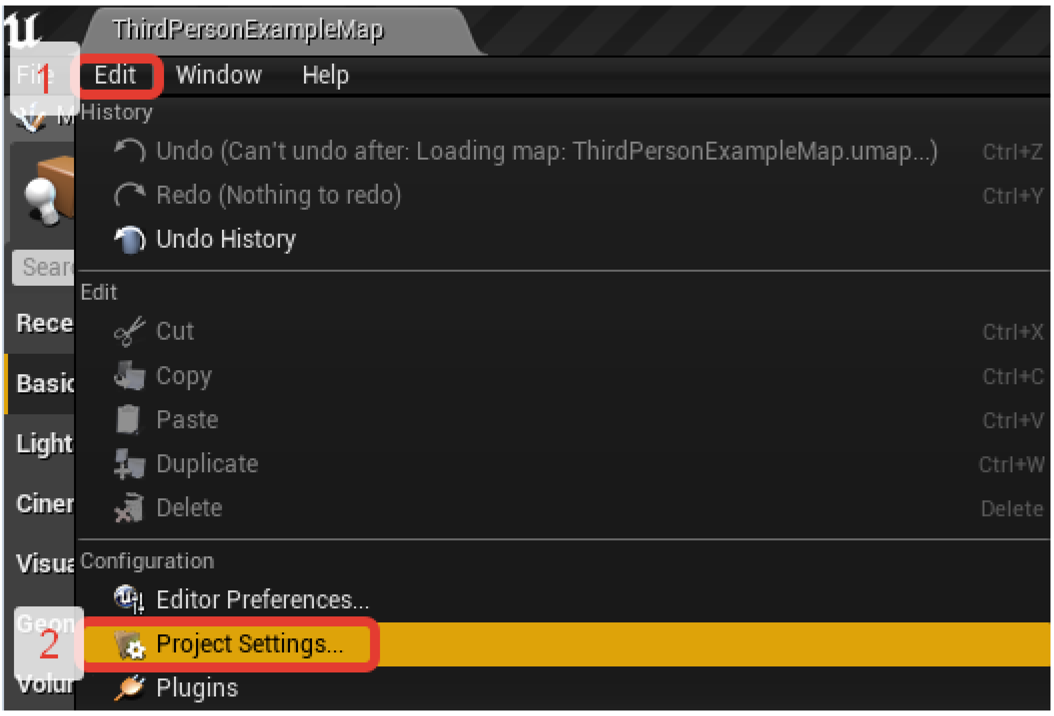
9.Integrating FightComponent into another project:

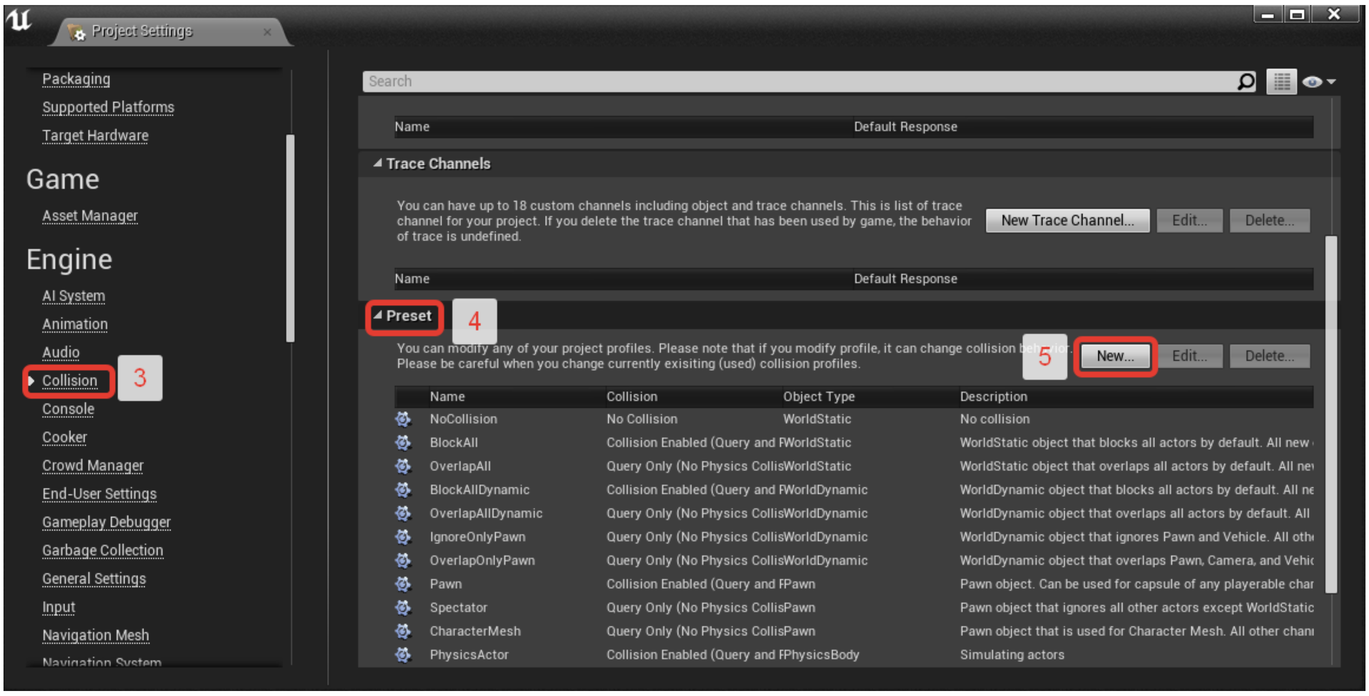
You can watch a video with the integration of FightComponent in another project on YouTube: https://www.youtube.com/watch?v=ewy3_uphjAg&t=342s

For example, let's take the ready-made **ThirdPerson** project from “*Epic Games*” and integrate **FightComponent** into it.

9.1 Creating your own clash profile

The first step is to immediately create a custom **DeadPawn** collision profile. To do this, open *Edit -> Project Settings .. -> Collision -> Preset -> New ...*





Name | Default Response

Trace Channels

You can have up to 18 custom channels including object and trace channels. This is list of trace channel for your project. If you delete the trace channel that has been used by game, the behavior of trace is undefined.

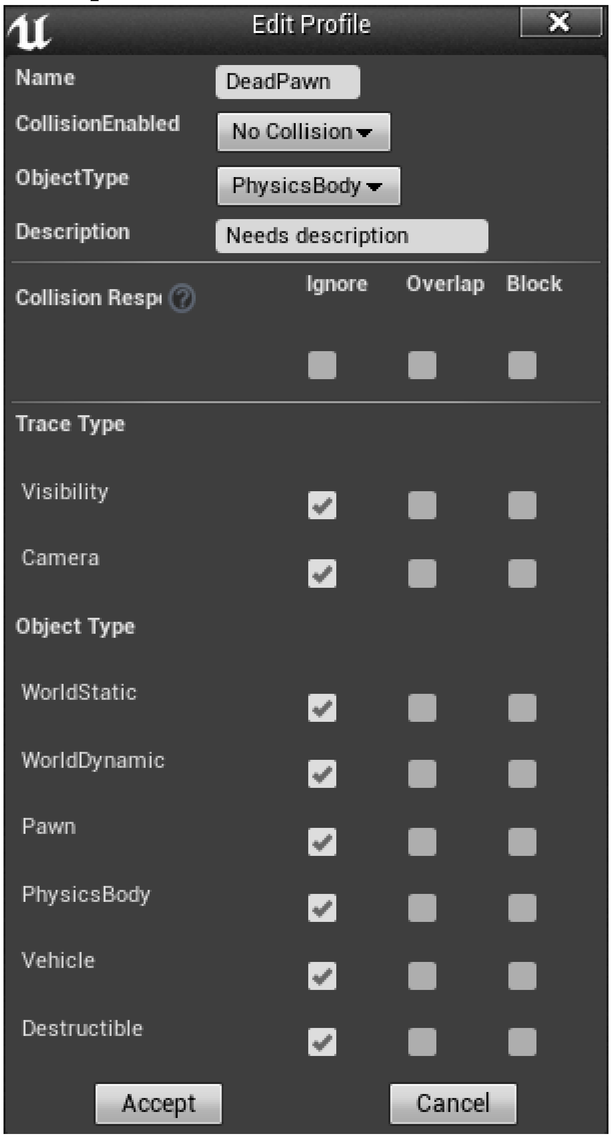
Name | Default Response

Preset

You can modify any of your project profiles. Please note that if you modify profile, it can change collision behavior. Please be careful when you change currently exisiting (used) collision profiles.

Name	Collision	Object Type	Description
NoCollision	No Collision	WorldStatic	No collision
BlockAll	Collision Enabled (Query and P	WorldStatic	WorldStatic object that blocks all actors by default. All new
OverlapAll	Query Only (No Physics Collis	WorldStatic	WorldStatic object that overlaps all actors by default. All nev
BlockAllDynamic	Collision Enabled (Query and P	WorldDynamic	WorldDynamic object that blocks all actors by default. All ne
OverlapAllDynamic	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that overlaps all actors by default. All
IgnoreOnlyPawn	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that ignores Pawn and Vehicle. All oth
OverlapOnlyPawn	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that overlaps Pawn, Camera, and Vehic
Pawn	Collision Enabled (Query and P	Pawn	Pawn object. Can be used for capsule of any playerable char
Spectator	Query Only (No Physics Collis	Pawn	Pawn object that ignores all other actors except WorldStatic
CharacterMesh	Query Only (No Physics Collis	Pawn	Pawn object that is used for Character Mesh. All other chan
PhysicsActor	Collision Enabled (Query and P	PhysicsBody	Simulating actors

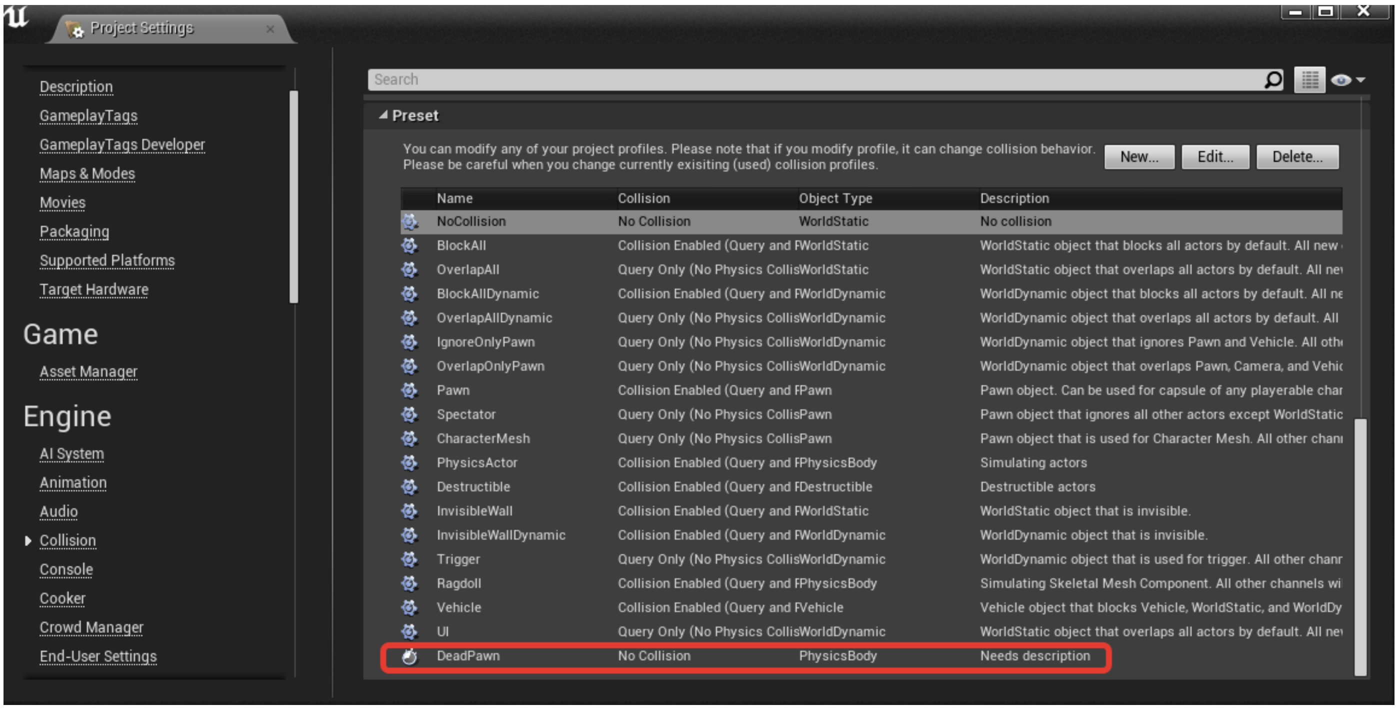
Change all parameters of the collision profile as shown in the example and click the “*Accept*” button.



Name	DeadPawn
CollisionEnabled	No Collision
ObjectType	PhysicsBody
Description	Needs description

Collision Resp	Ignore	Overlap	Block
	☐	☐	☐
Trace Type			
Visibility	✓	☐	☐
Camera	✓	☐	☐
Object Type			
WorldStatic	✓	☐	☐
WorldDynamic	✓	☐	☐
Pawn	✓	☐	☐
PhysicsBody	✓	☐	☐
Vehicle	✓	☐	☐
Destructible	✓	☐	☐

After clicking the “*Accept*” button, you should have a new collision profile in the “*Preset*” list:

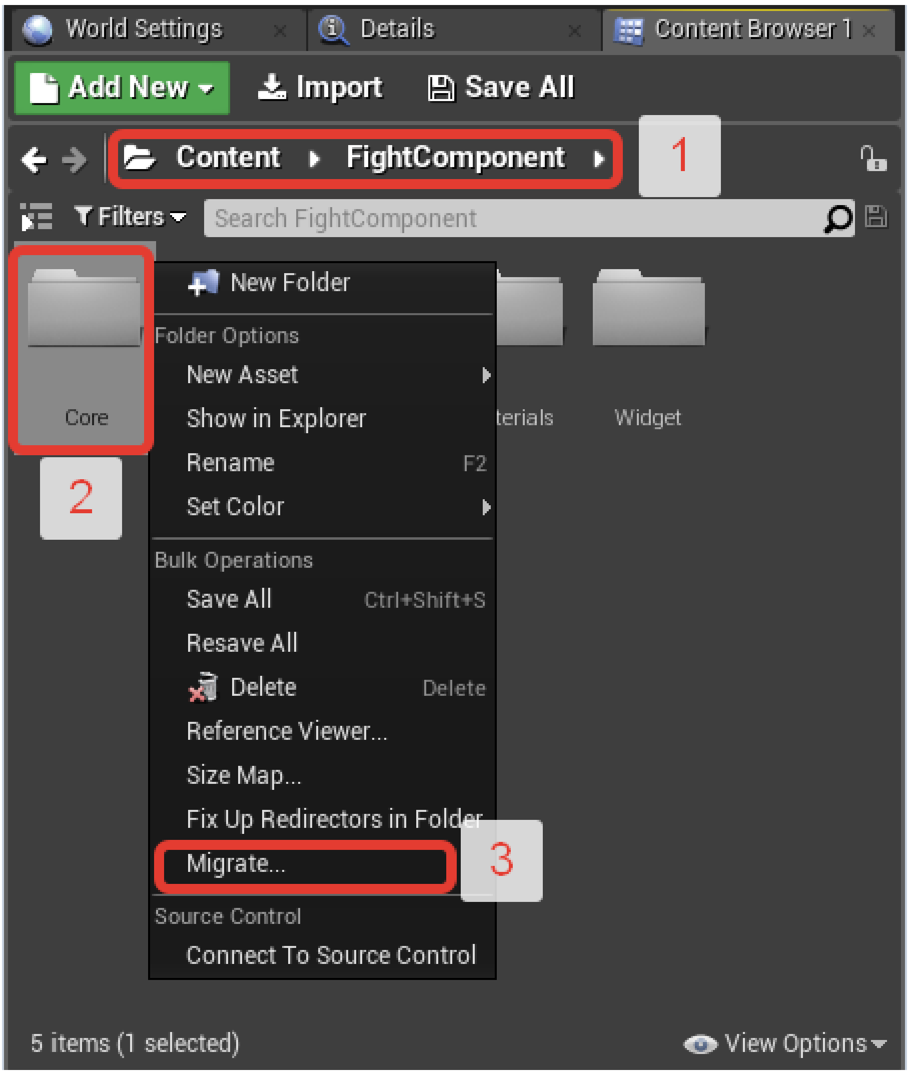


You can modify any of your project profiles. Please note that if you modify profile, it can change collision behavior. Please be careful when you change currently exisiting (used) collision profiles.

New... Edit... Delete...

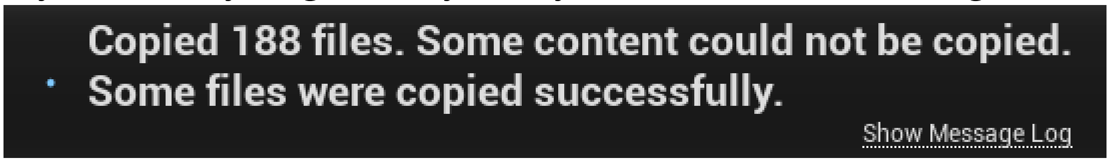
Name	Collision	Object Type	Description
NoCollision	No Collision	WorldStatic	No collision
BlockAll	Collision Enabled (Query and P	WorldStatic	WorldStatic object that blocks all actors by default. All new
OverlapAll	Query Only (No Physics Collis	WorldStatic	WorldStatic object that overlaps all actors by default. All nev
BlockAllDynamic	Collision Enabled (Query and P	WorldDynamic	WorldDynamic object that blocks all actors by default. All ne
OverlapAllDynamic	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that overlaps all actors by default. All
IgnoreOnlyPawn	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that ignores Pawn and Vehicle. All othe
OverlapOnlyPawn	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that overlaps Pawn, Camera, and Vehic
Pawn	Collision Enabled (Query and P	Pawn	Pawn object. Can be used for capsule of any playerable char
Spectator	Query Only (No Physics Collis	Pawn	Pawn object that ignores all other actors except WorldStatic
CharacterMesh	Query Only (No Physics Collis	Pawn	Pawn object that is used for Character Mesh. All other chan
PhysicsActor	Collision Enabled (Query and P	PhysicsBody	Simulating actors
Destructible	Collision Enabled (Query and P	Destructible	Destructible actors
InvisibleWall	Collision Enabled (Query and P	WorldStatic	WorldStatic object that is invisible.
InvisibleWallDynamic	Collision Enabled (Query and P	WorldDynamic	WorldDynamic object that is invisible.
Trigger	Query Only (No Physics Collis	WorldDynamic	WorldDynamic object that is used for trigger. All other chanr
Ragdoll	Collision Enabled (Query and P	PhysicsBody	Simulating Skeletal Mesh Component. All other channels wi
Vehicle	Collision Enabled (Query and P	Vehicle	Vehicle object that blocks Vehicle, WorldStatic, and WorldDy
UI	Query Only (No Physics Collis	WorldDynamic	WorldStatic object that overlaps all actors by default. All nev
DeadPawn	No Collision	PhysicsBody	Needs description

FightComponent into your project. To do this, create a new **FightComponent** project, open it, go to the **FightComponent** folder, right-click on the **Core** folder and select the “*Migrate*” button.



After that click "*OK*", find your project in the explorer and select the "*Content*" folder.

If you did everything correctly, then you should see the following notification:



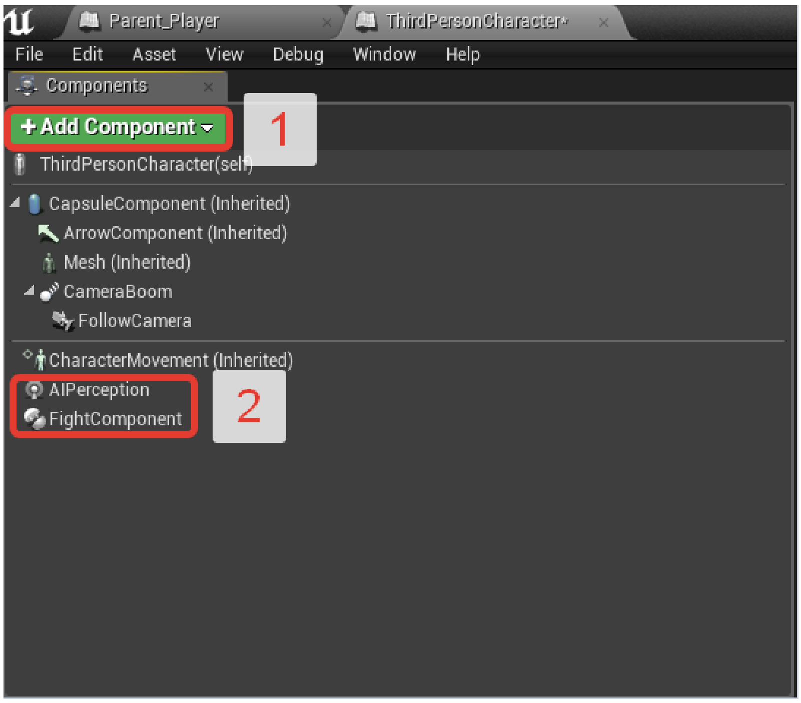
Copied 188 files. Some content could not be copied. Some files were copied successfully.

Next, open the **FightComponent** located in *Content* -> *FightComponent* -> *Core* -> *Blueprints* -> *Component* and compile it.

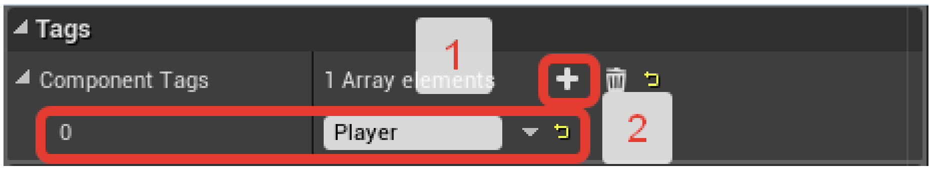
9.3 Customization of the game character.

Now that you have a **FightComponent** in your project, you can start customizing your game character. To do this, open your character, and also open the **Parent_Player** character, which is located in *Content -> FightComponent -> Core -> Blueprints -> Player*.

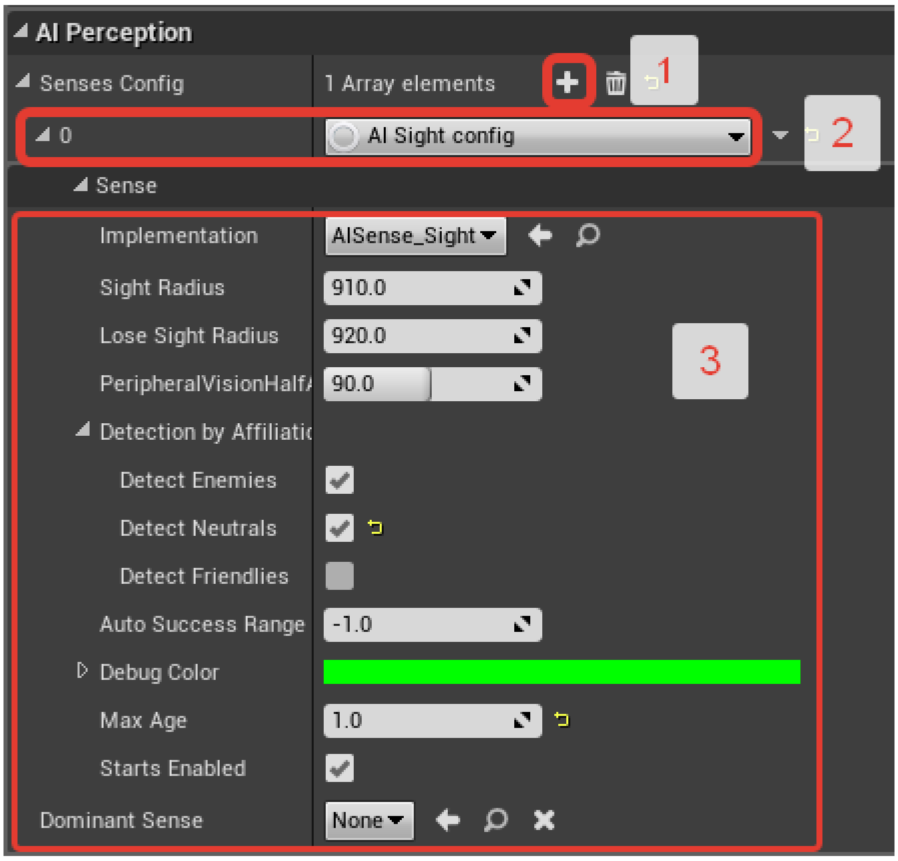
In your character, click the *"+ Add Component"* button and add a **FightComponent** as well as an **AIPerception**.



Since the **FightComponent** is configured automatically, you only need to add the **Player** tag. To do this, click on the **FightComponent**, find the **Tags** tab, click on the "+" and enter **Player** in the line that appears.



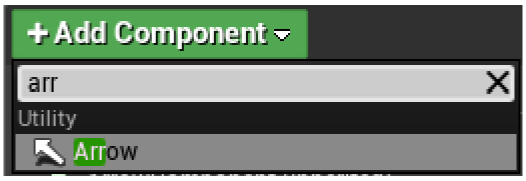
When you're done setting up the **Tags** in the **FightComponent**, click on the **AIPerception**. In the opened tabs find **AI Perception**, which will contain the **Senses Config** parameter and click on the "+". Next, you must select **AI Sight Config**, and also open the tab that appears. After that, you need to configure this config as shown in the example:



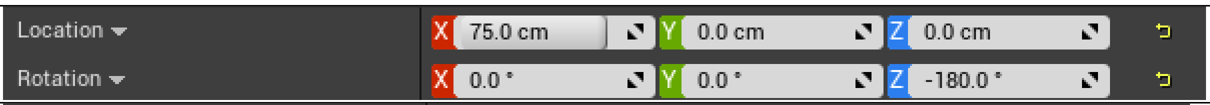
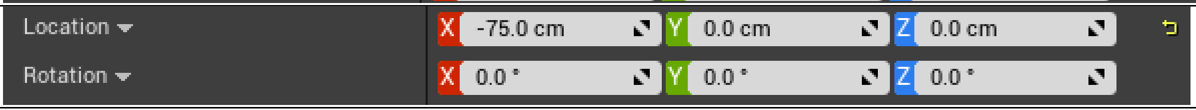
AI Perception

Senses Config	1 Array elements
0	AI Sight config
Sense	
Implementation	AISense_Sight
Sight Radius	910.0
Lose Sight Radius	920.0
PeripheralVisionHalfA	90.0
Detection by Affiliatic	
Detect Enemies	✓
Detect Neutrals	✓
Detect Friendlies	
Auto Success Range	-1.0
Debug Color	🟩
Max Age	1.0
Starts Enabled	✓
Dominant Sense	None

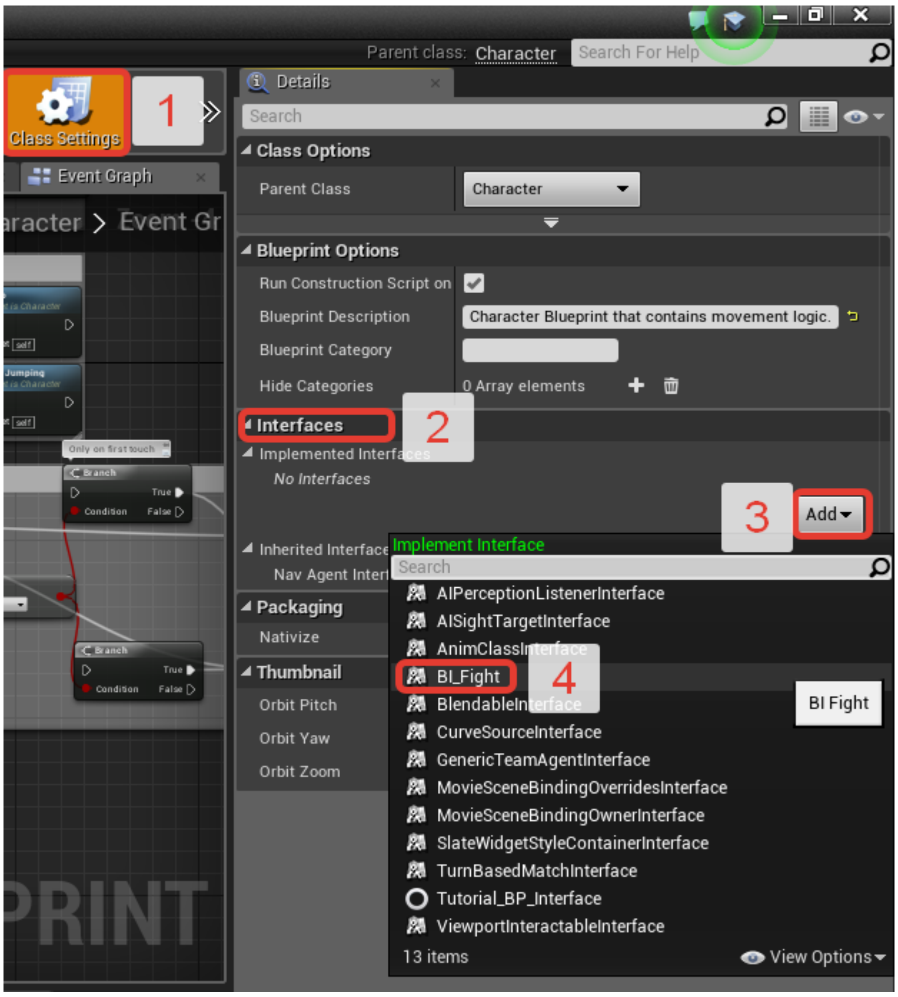
Next, you need to add **ArrowComponent** arrows to your character, which are responsible for the location and direction of the stealth kill. To add them, you need to click on the *"+ Add Component*" button again
and enter “*Arrow*” in the search.



Add two arrows to your character and rename them **Front** and **Back**.
(IMPORTANT!!! Both arrows added must be children of the **CapsuleComponent**!)
Select any arrow and, depending on its direction, change its “*Location*” and “*Rotation*” in the “*Transform*” list.

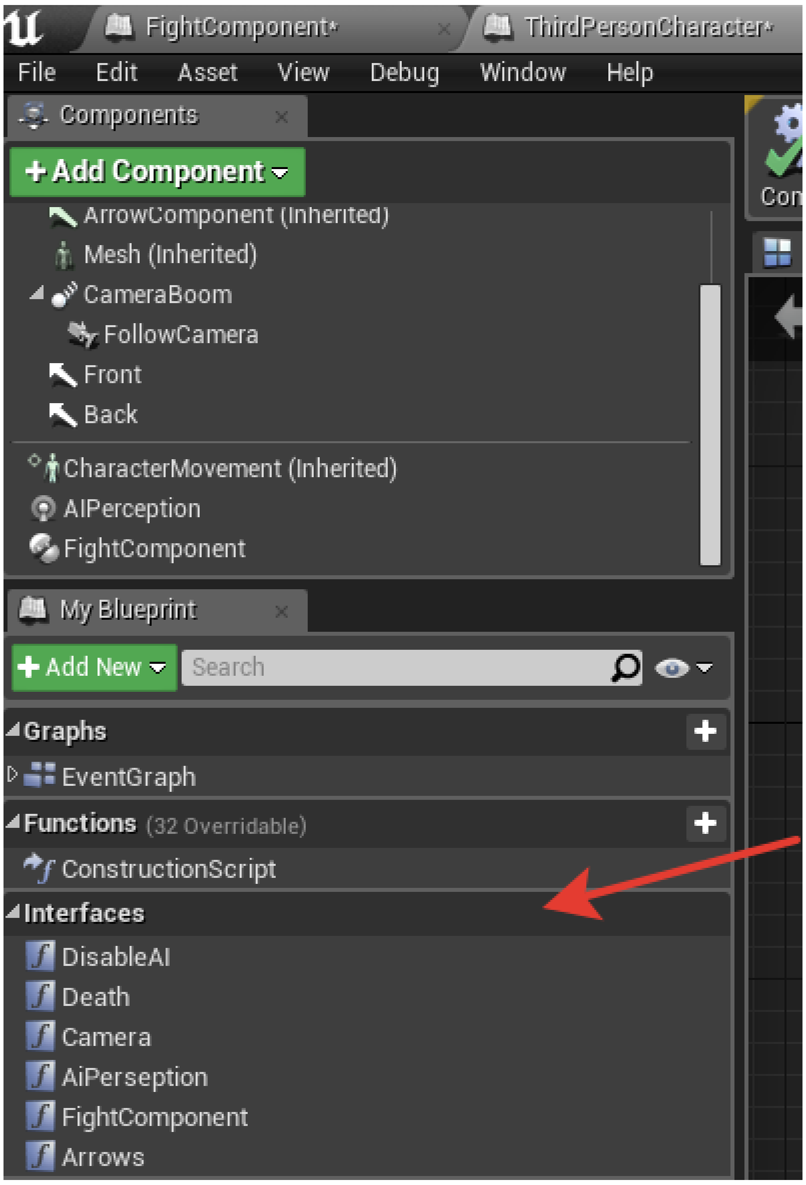
Front	 Location: X 75.0 cm, Y 0.0 cm, Z 0.0 cm; Rotation: X 0.0 °, Y 0.0 °, Z -180.0 °
Back	 Location: X -75.0 cm, Y 0.0 cm, Z 0.0 cm; Rotation: X 0.0 °, Y 0.0 °, Z 0.0 °

Now that you are done with the settings for the main components, you should add the **BI_Fight** interface to your character. To do this, open **Class Settings**, find the "*Interfaces*" tab, click on the "*Add*" button, and select **BI_Fight**.

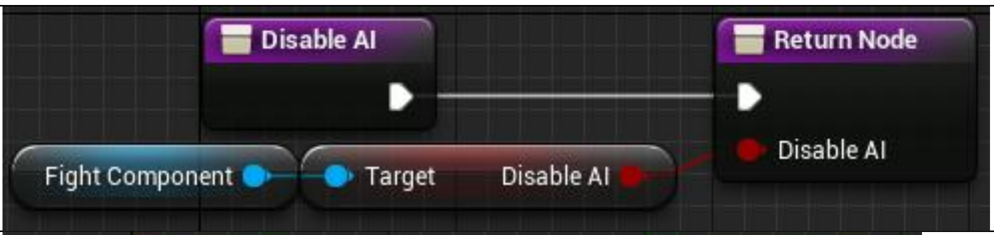
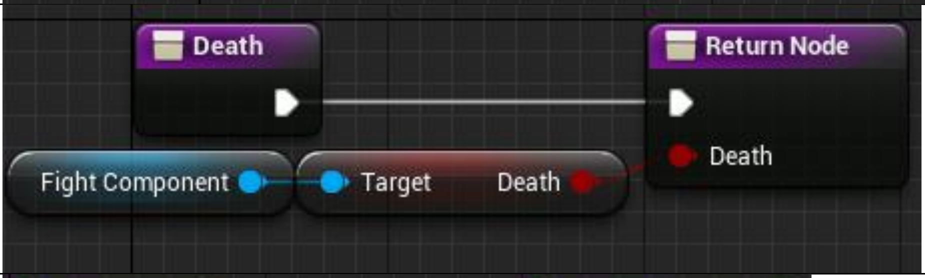
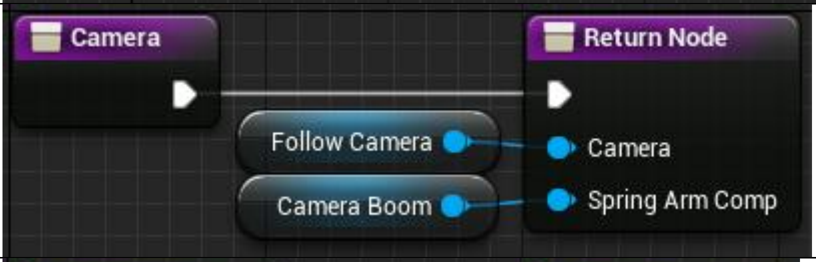
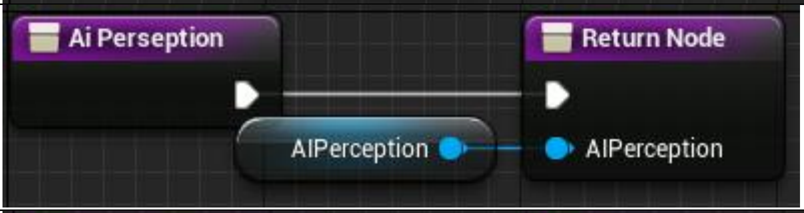
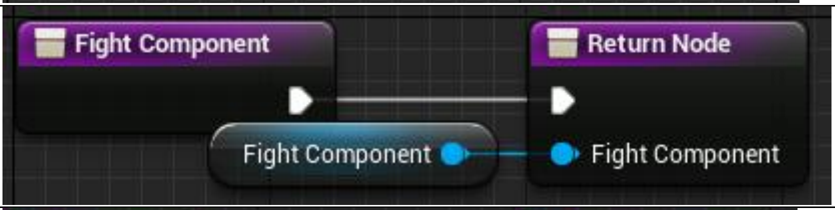
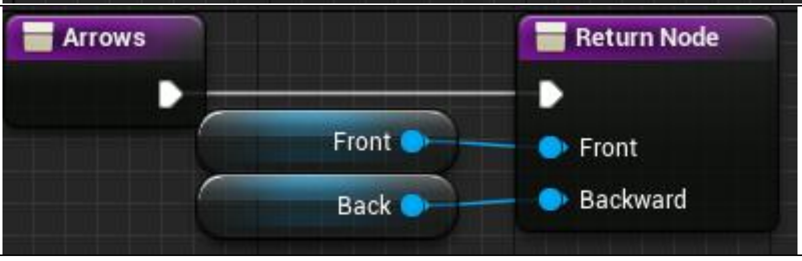


Parent Class: Character
Run Construction Script on Drag: ✓
Blueprint Description: Character Blueprint that contains movement logic.
Blueprint Category:
Hide Categories: 0 Array elements

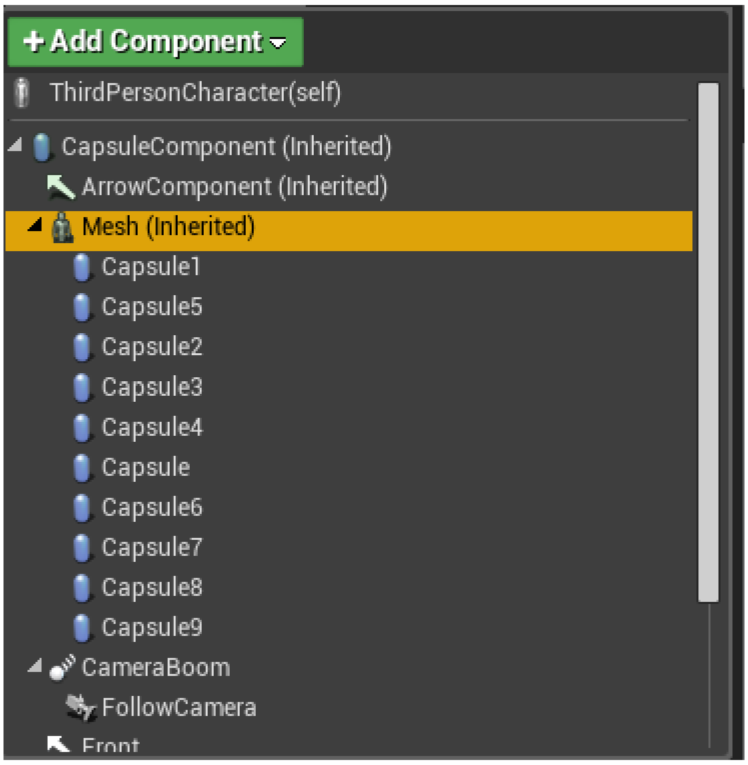
After that, you should compile your character again. After successful compilation, you will see that on the left side of the screen your character has a list of "*Interfaces*".



Now your task is to fill in all these interface functions. How to fill them is shown below:

DisableAI	 Disable AI Return Node Disable AI Fight Component Target Disable AI
Death	 Death Return Node Death Fight Component Target Death
Camera	 Camera Return Node Follow Camera Camera Camera Boom Spring Arm Comp
AIPerseption	 Ai Perseption Return Node AIPerception AIPerception
FightComponent	 Fight Component Return Node Fight Component Fight Component
Arrows	

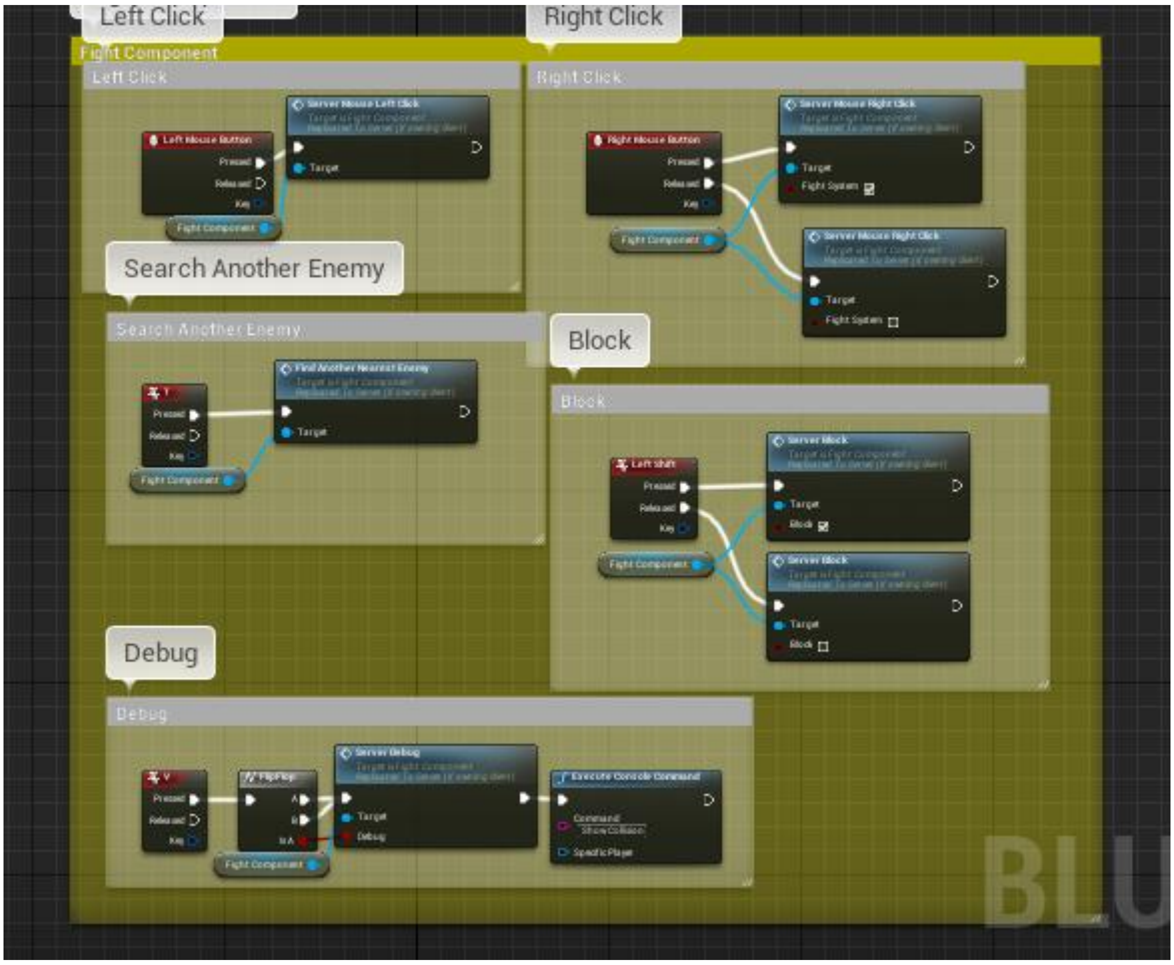
When you're done setting up all of the above components, you'll need to add hitboxes to your character. To do this, open **Parent_AI** and copy all the hitboxes into your character by first clicking on the **Mesh** component (this is necessary in order for the hitboxes to become children and inherit the animation of the skeleton). You should have something like this:



Now your task is to set up these hitboxes the same as in the game's **Parent_Player** character (Name, size, parent socket, location).

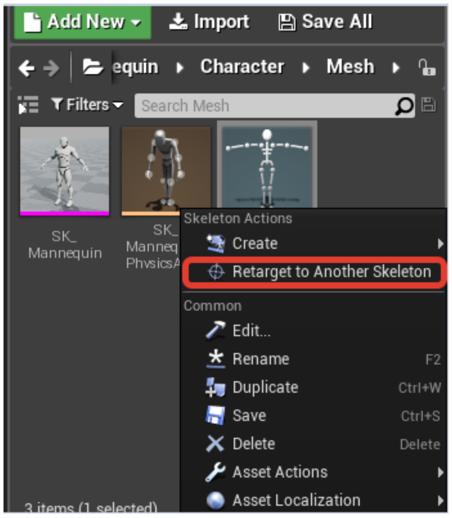
A video detailing how this is done can be viewed here: https://youtu.be/Brtx41XbwjY

Now that the character is fully configured, all that remains is to transfer control from the **Parent_Player** to your character.

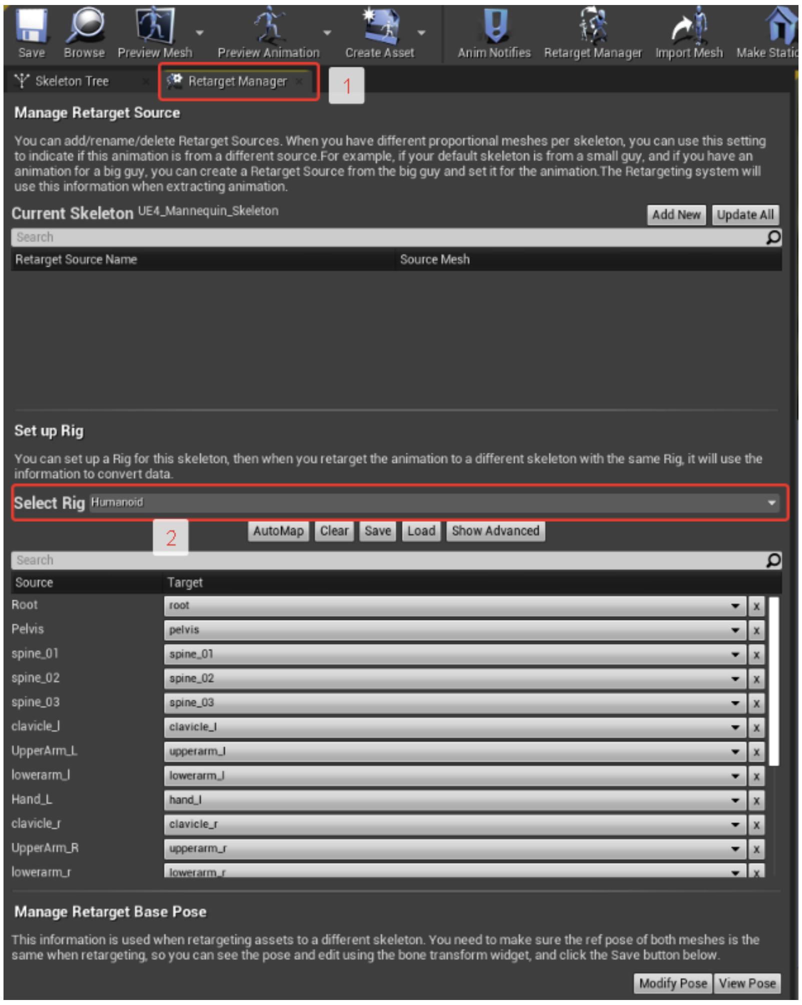


9.4 Configuring Animation Blueprint.

Since the character has been fully configured, now you should go to animations and **AnimBP_Parent_Character**, but before that you should retarget the **FightComponent** skeleton to the skeleton of your game character. To do this, go to *Content* -> *FightComponent* -> *Core* -> *Animations* -> *Manequin* -> *Character* -> *Mesh*.
Right click on the **UE4_Mannequin_Skeleton** skeleton and click on "*Retarget to another skeleton*".



In the window that opens, select the skeleton of your character and click "*Retarget*".
If your skeleton is not in the list, open the skeleton of your character, as well as the skeleton of the **FightComponent** character, go to *Retarget Manager* and set *Select Rig* to *Humanoid* in both skeletons.



Manage Retarget Source

You can add/rename/delete Retarget Sources. When you have different proportional meshes per skeleton, you can use this setting to indicate if this animation is from a different source.For example, if your default skeleton is from a small guy, and if you have an animation for a big guy, you can create a Retarget Source from the big guy and set it for the animation.The Retargeting system will use this information when extracting animation.

Current Skeleton UE4_Mannequin_Skeleton

Retarget Source Name	Source Mesh

Set up Rig

You can set up a Rig for this skeleton, then when you retarget the animation to a different skeleton with the same Rig, it will use the information to convert data.

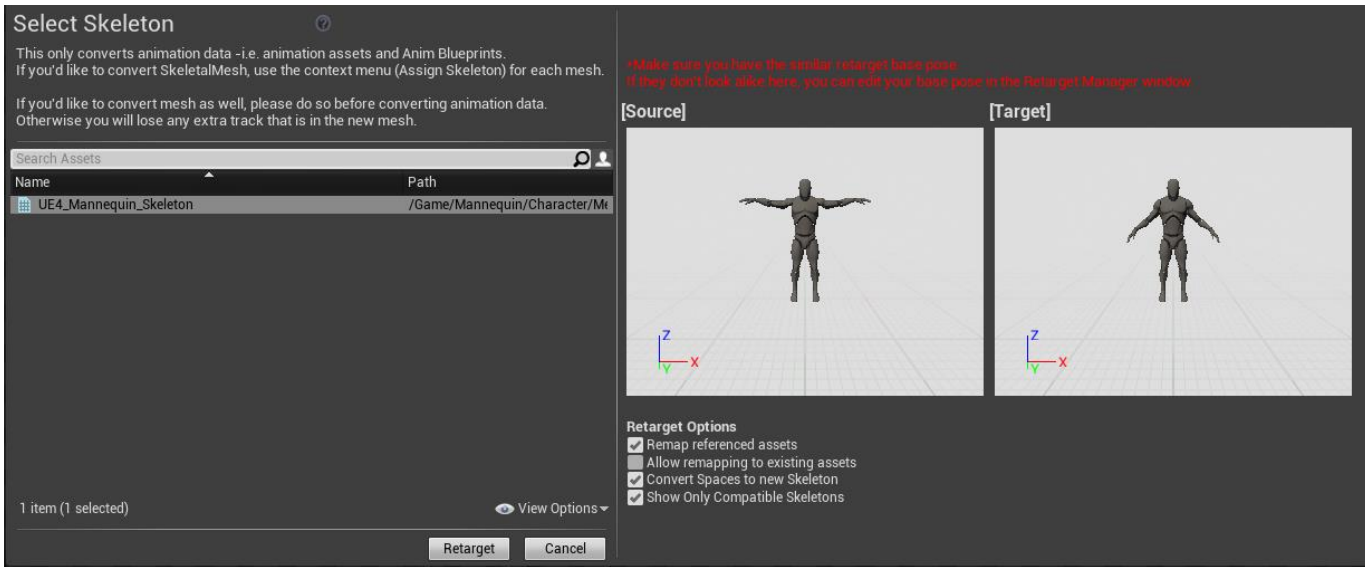
Select Rig Humanoid

Source	Target
Root	root
Pelvis	pelvis
spine_01	spine_01
spine_02	spine_02
spine_03	spine_03
clavicle_l	clavicle_l
UpperArm_L	upperarm_l
lowerarm_l	lowerarm_l
Hand_L	hand_l
clavicle_r	clavicle_r
UpperArm_R	upperarm_r
lowerarm_r	lowerarm_r

Manage Retarget Base Pose

This information is used when retargeting assets to a different skeleton. You need to make sure the ref pose of both meshes is the same when retargeting, so you can see the pose and edit using the bone transform widget, and click the Save button below.

After that, your character's skeleton should appear in the list.



Select Skeleton

This only converts animation data -i.e. animation assets and Anim Blueprints.
If you'd like to convert SkeletalMesh, use the context menu (Assign Skeleton) for each mesh.

If you'd like to convert mesh as well, please do so before converting animation data.
Otherwise you will lose any extra track that is in the new mesh.

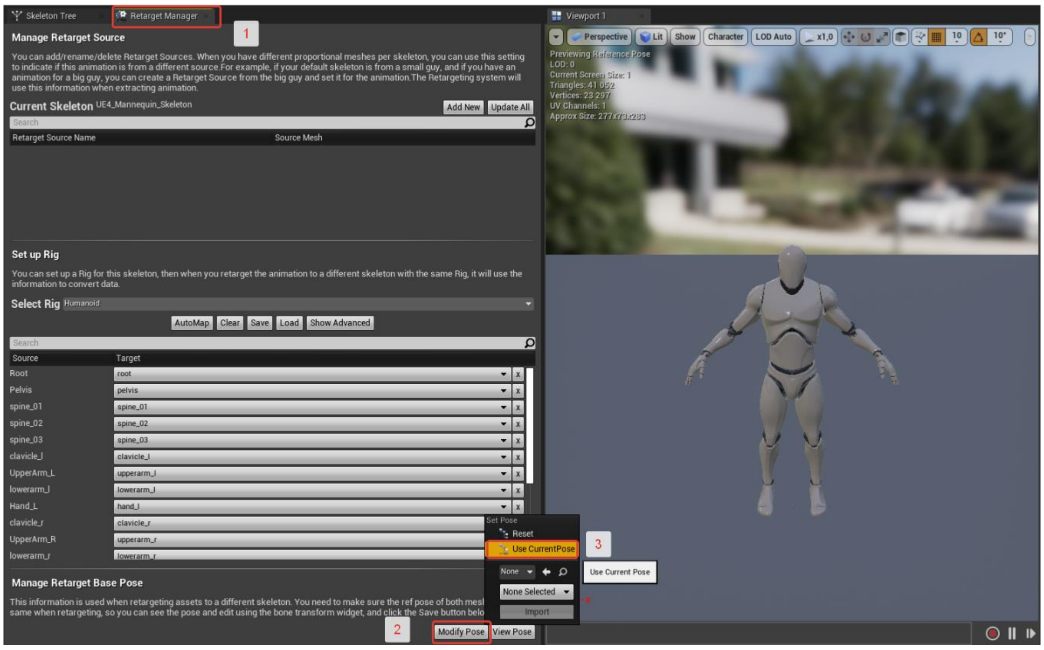
Name	Path
UE4_Mannequin_Skeleton	/Game/Mannequin/Character/Me

1 item (1 selected)

*Make sure you have the similar retarget base pose
If they don't look alike here, you can edit your base pose in the Retarget Manager window

Retarget Options
- ✓ Remap referenced assets
- Allow remapping to existing assets
- ✓ Convert Spaces to new Skeleton
- ✓ Show Only Compatible Skeletons

If the characters ' poses are different, then you need to go to the **FightComponent** skeleton again, open the *Retarget Manager* section, click the *Modify Pose* button and select *Use Current Pose* (If the skeleton is in the T-pose in Viewport, then before clicking the *Modify Pose* button, you must click the *View Pose* button to change the pose to the A-pose).



Manage Retarget Source

You can add/rename/delete Retarget Sources. When you have different proportional meshes per skeleton, you can use this setting to indicate if this animation is from a different source.For example, if your default skeleton is from a small guy, and if you have an animation for a big guy, you can create a Retarget Source from the big guy and set it for the animation.The Retargeting system will use this information when extracting animation.

Current Skeleton UE4_Mannequin_Skeleton

Retarget Source Name	Source Mesh

Set up Rig

You can set up a Rig for this skeleton, then when you retarget the animation to a different skeleton with the same Rig, it will use the information to convert data.

Select Rig Humanoid

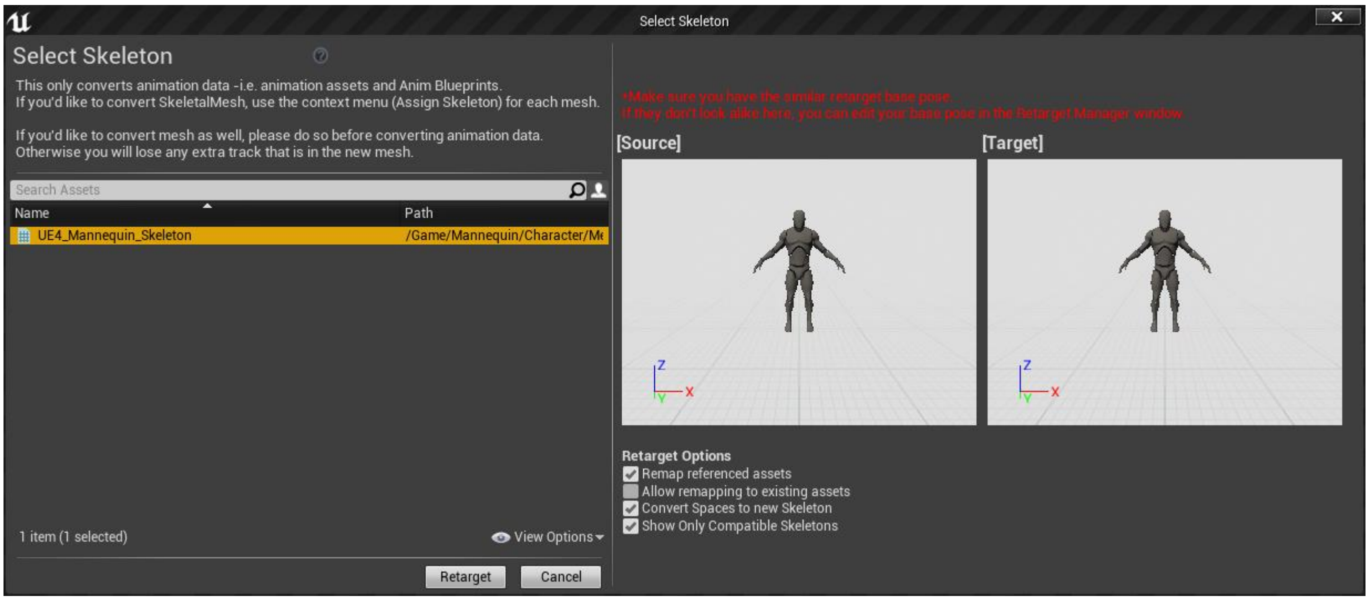
Source	Target
Root	root
Pelvis	pelvis
spine_01	spine_01
spine_02	spine_02
spine_03	spine_03
clavicle_l	clavicle_l
UpperArm_L	upperarm_l
lowerarm_l	lowerarm_l
Hand_L	hand_l
clavicle_r	clavicle_r
UpperArm_R	upperarm_r
lowerarm_r	lowerarm_r

Manage Retarget Base Pose

This information is used when retargeting assets to a different skeleton. You need to make sure the ref pose of both mesl same when retargeting, so you can see the pose and edit using the bone transform widget, and click the Save button belo

Previewing Reference Pose
LOD: 0
Current Screen Size: 1
Triangles: 41 052
Vertices: 23 297
UV Channels: 1
Approx Size: 277x73x283

Now you can make a *Retarget*.



Select Skeleton

This only converts animation data -i.e. animation assets and Anim Blueprints.
If you'd like to convert SkeletalMesh, use the context menu (Assign Skeleton) for each mesh.

If you'd like to convert mesh as well, please do so before converting animation data.
Otherwise you will lose any extra track that is in the new mesh.

Name	Path
UE4_Mannequin_Skeleton	/Game/Mannequin/Character/Me

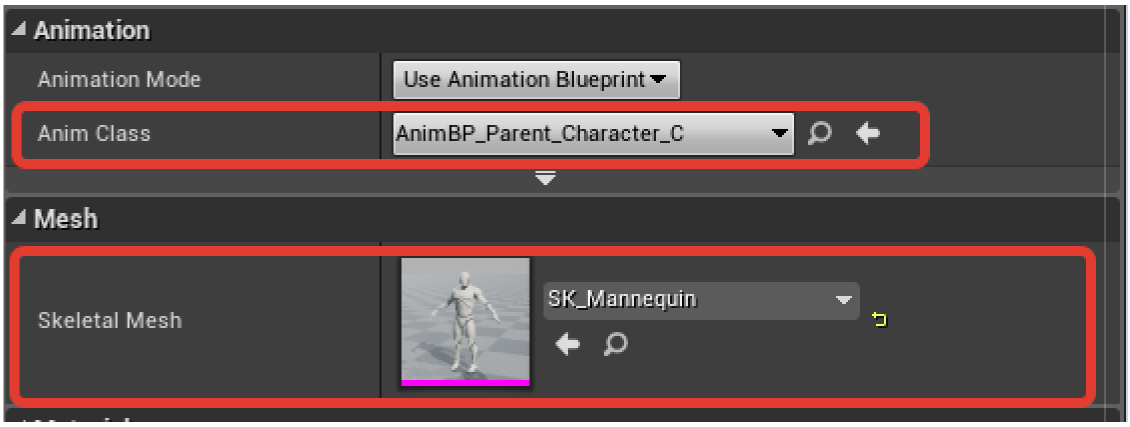
1 item (1 selected)

*Make sure you have the similar retarget base pose.
If they don't look alike here, you can edit your base pose in the Retarget Manager window.

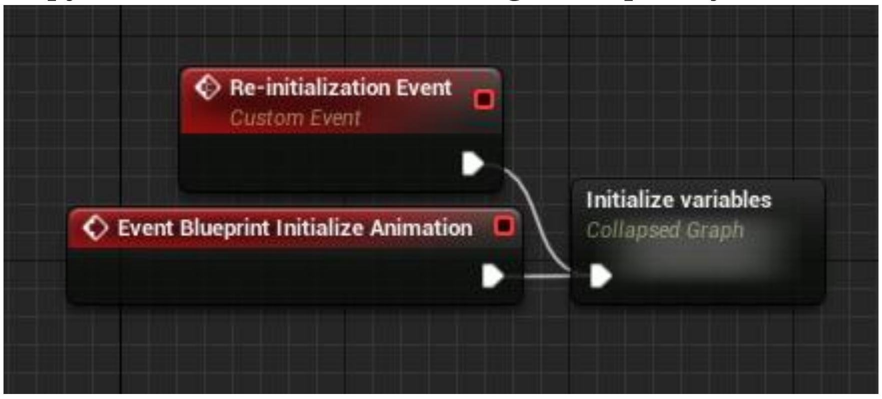
[Source] [Target]

Retarget Options
- [✓] Remap referenced assets
- [ ] Allow remapping to existing assets
- [✓] Convert Spaces to new Skeleton
- [✓] Show Only Compatible Skeletons

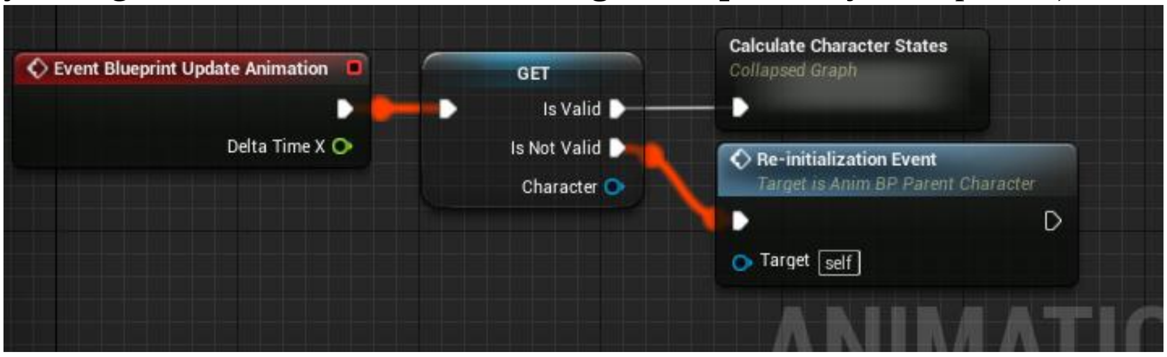
IMPORTANT!!!! (After retargeting to your skeleton, you should change "*Skeletal Mesh*" in **Parent_AI_Human** to your "*Skeletal Mesh*", and also select "*Anim Class*" **AnimBP_Parent_Character**)


Animation
Animation Mode: Use Animation Blueprint
Anim Class: AnimBP_Parent_Character_C
Mesh
Skeletal Mesh: SK_Mannequin

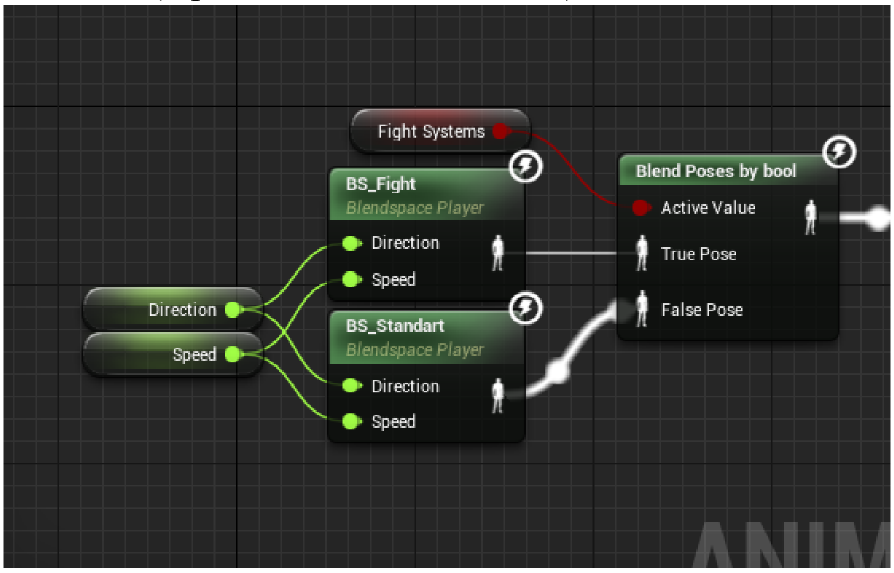
After these manipulations, you can proceed to setting up the animated blueprint. Open **AnimBP_Parent_Character** (*Content* -> *FightComponent* -> *Core* -> *Animations*) and also your animated blueprint for your character.
Copy the variable initialization logic and paste your character into **AnimBP**.



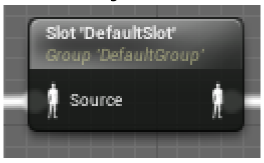
Also copy the logic that updates every frame and paste it into your animation blueprint (If your logic has similar variables with **FightComponent**, just skip them).


Event Blueprint Update Animation
Delta Time X
GET
Is Valid
Is Not Valid
Character
Calculate Character States
Collapsed Graph
Re-initialization Event
Target is Anim BP Parent Character
Target
self

Copy the **FightComponent** blendspaces and animations, and paste them into your **AnimBP** (repeat this for each "*State*").



Since the **Anim Montages** use the "*DefaultSlot*", create and connect the *"DefaultSlot" Slot* in your animation blueprint.



If all steps were completed without errors, then the integration of **FightComponent** to your project was completed successfully!