Unreal Engine First-Person Shooter Plugin Documentation

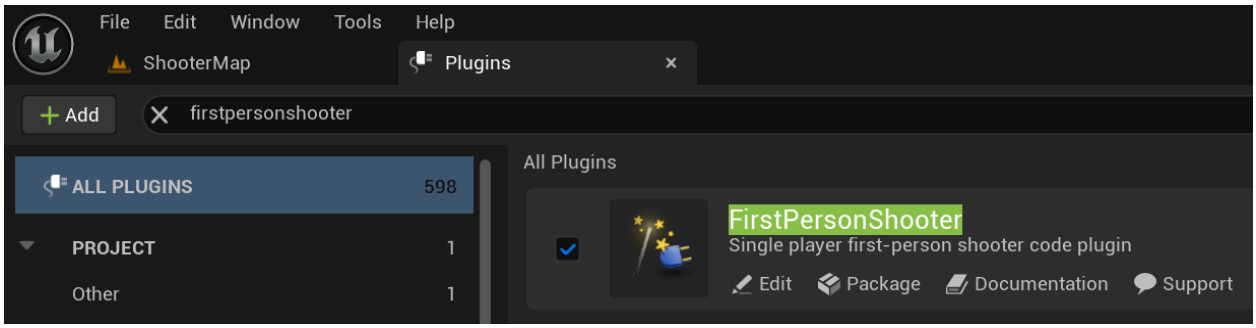
Released: September 1, 2022
Last modified: May 27, 2024

1. Enabling the Plugin

1.1. Install the FirstPersonShooter plugin to the engine from the Epic Games Launcher library.

1.2. Open your project, go to Edit, Plugins, and search for FirstPersonShooter.

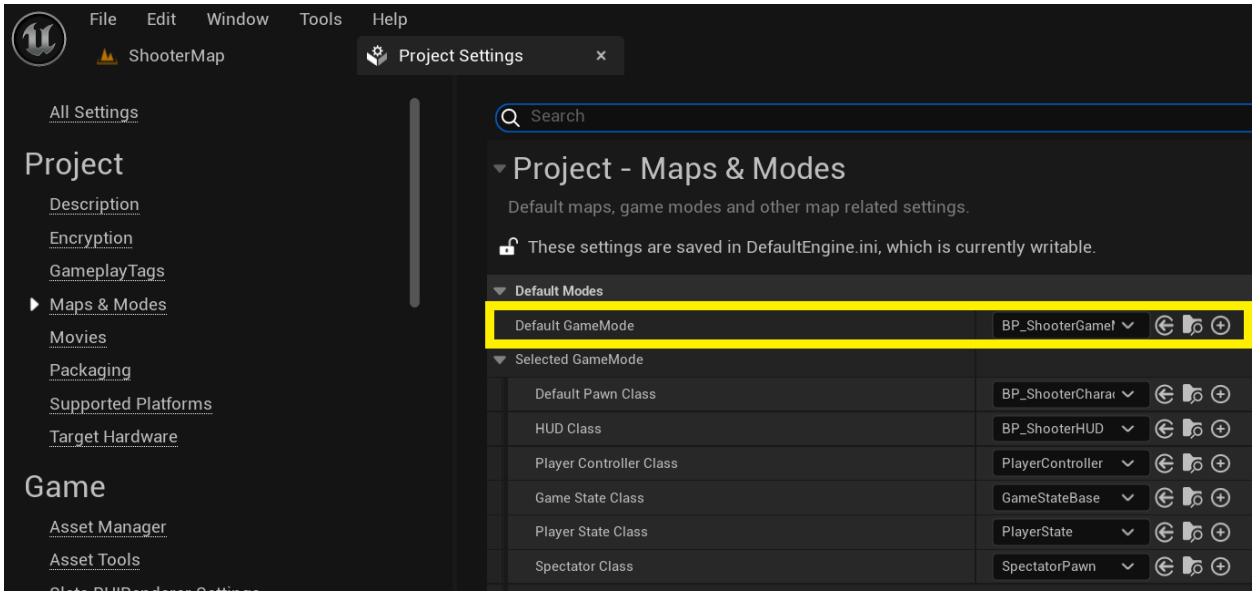
1.3. Enable the FirstPersonShooter plugin and restart the editor.



2. Setting up the Game Mode

2.1. Create a blueprint based on the GameModeBase class.

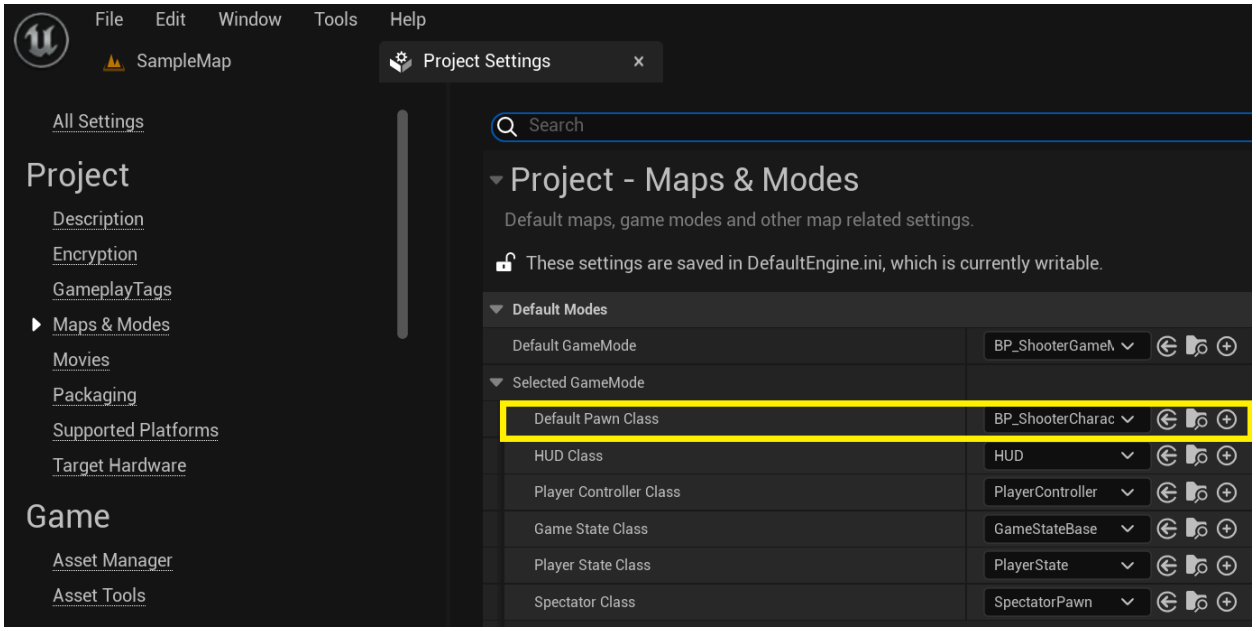
2.2. Click Edit, Project Settings, Maps & Modes, and set the Default GameMode to the GameModeBase blueprint you created.



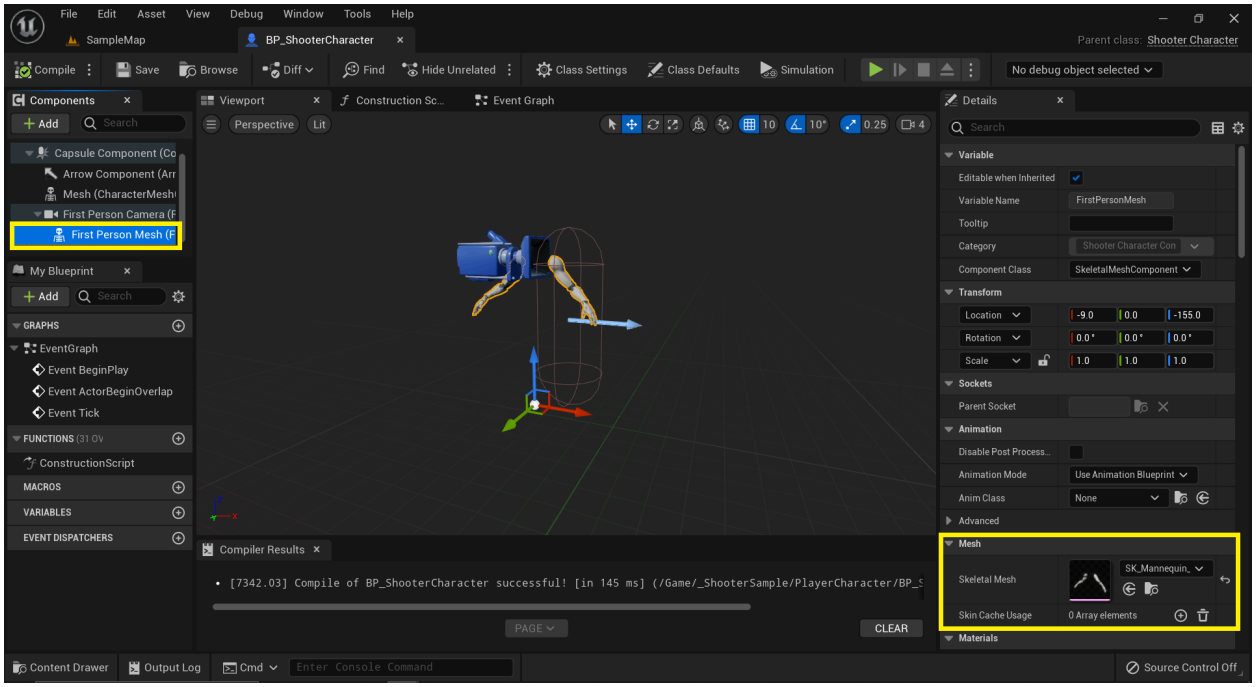
3. Setting up the Shooter Character

3.1. Create a blueprint based on the ShooterCharacter class.

3.2. Click Edit, Project Settings, Maps & Modes, and set the Default Pawn Class to the shooter character blueprint you created.



3.3. Set the shooter character's first-person mesh.

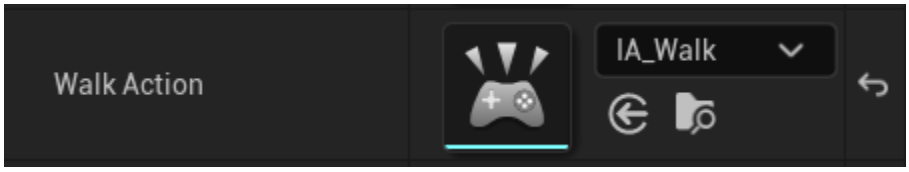


3.4. Setting up the Shooter Character's Input

The ShooterCharacter class has an Input Action member variable for each of its actions that are handled by the plugin's source code. Follow these steps in order to create and set up these actions. You can see an example of a ShooterCharacter Input Mapping Context in the example project:

3.4.1. Right click in the Content Browser, hover over Input, and select Input Action.

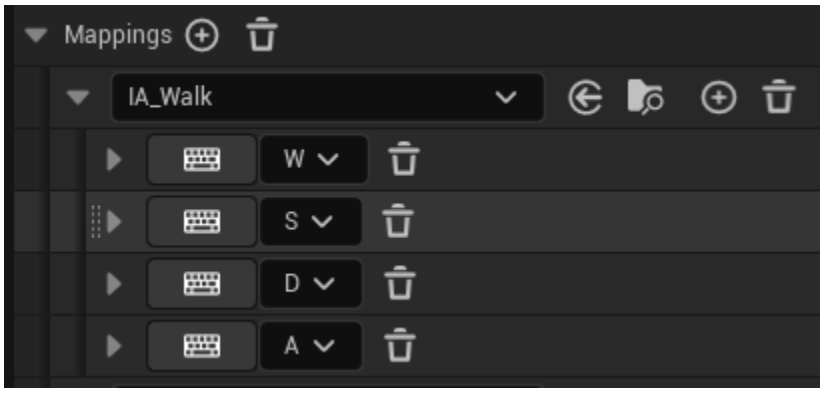
3.4.2. Open your ShooterCharacter's blueprint and set the input action variable you wish to set up to the one you just created.



3.4.3. Right click in the Content Browser, hover over Input, and select Input Action Mapping.

3.4.4. Open your ShooterCharacter's blueprint and set the PlayerInputMappingContext variable to the Input Mapping Context you just created.

3.4.5. Open the Input Mapping Context and add a Mapping for the action you are setting up and bind the key(s) that the player will press to perform this action.



The shooter character's blueprint-exposed variables can be found in the Shooter Character Properties section of its blueprint.

Shooter Character Blueprint-Exposed Functions:

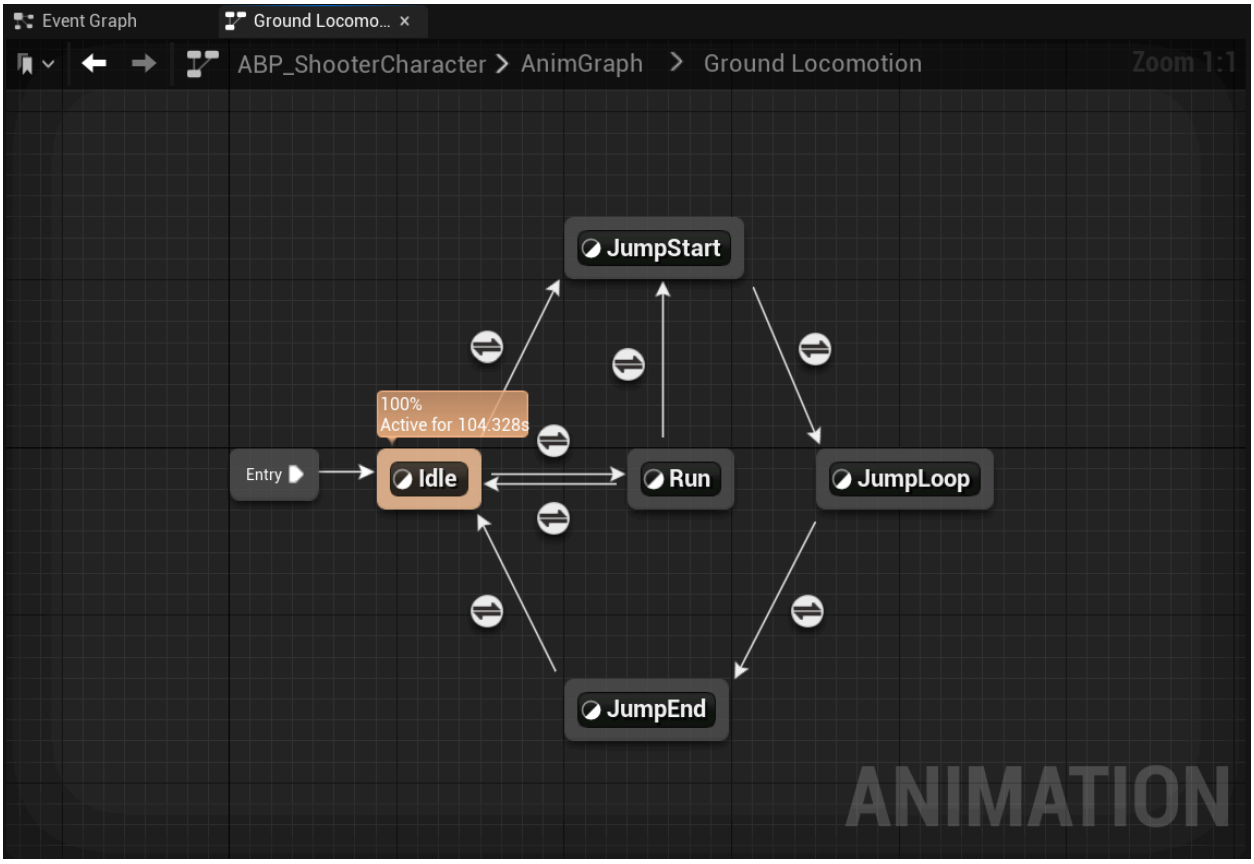
- Float **GetMissingHealthAmount**();
- Bool **IsDead**() const;
- Int32 **GetReservedAmmoAmount**(const FName AmmoType) const;
- Bool **HasWeaponOfType**(const FName WeaponType) const;
- Void **EquipWeaponAtIndex**(const int32 InWeaponIndex);
- Void **EquipWeaponAtPreviousIndex**();
- Void **EquipWeaponAtNextIndex**();
- Void **FireEquippedWeapon**();
- FHitResult **LineTraceFromScreen**(const ECollisionChannel InCollisionChannel) const;
- Void **DropEquippedWeapon**();
- Void **ReloadEquippedWeapon**();
- Void **Die**();

3.5. Setting up the Shooter Character's animation blueprint

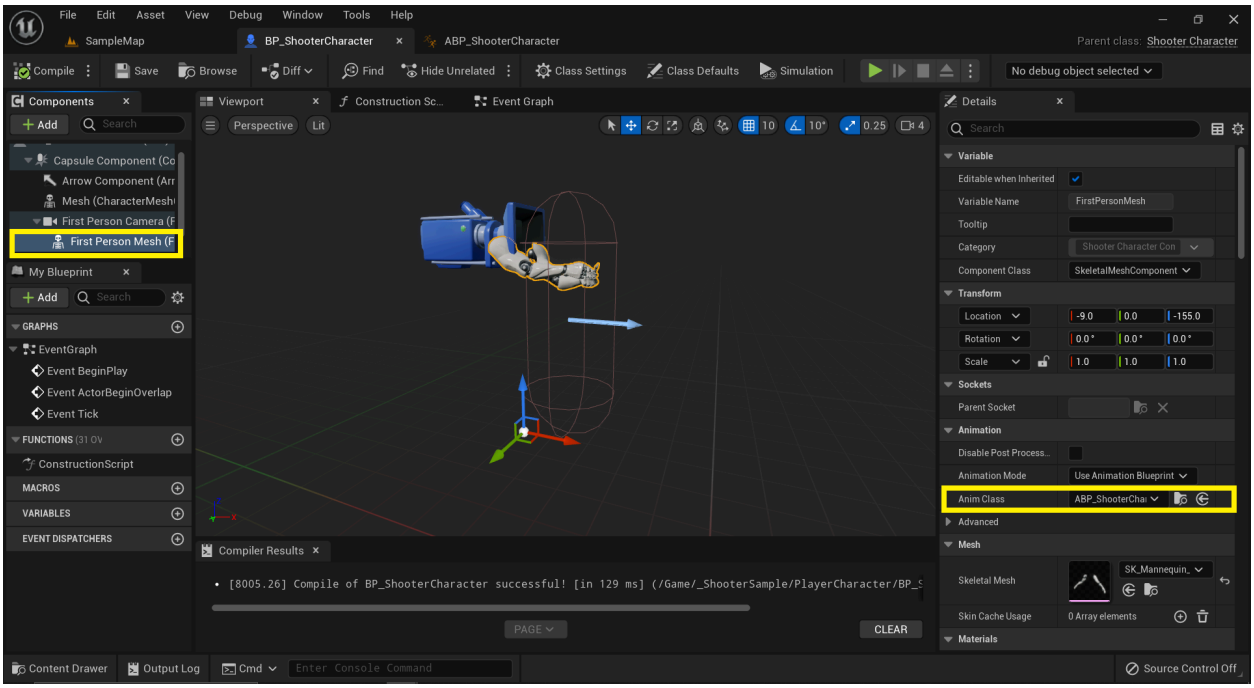
3.5.1. Create an animation blueprint that uses your shooter character's first-person mesh's skeleton and ShooterCharacterAnimInstance as the parent class.

3.5.2. Open your Shooter Character Animation Blueprint, go to the Event Graph, drag off of the Event Blueprint Update Animation node's execution pin, call the Update Animation Properties method from the ShooterCharacterAnimInstance class.

3.5.3. Set up the shooter character's first-person mesh's animation blueprint's Anim Graph (this is just an example, you can set up your Anim Graph however you prefer. You can also see this AnimGraph in the example project).



3.5.4. Set your shooter character's first-person mesh's animation blueprint.



Shooter Character Animation Variables:

- **Shooter Character Owner** (Character): Shooter Character that is using the Shooter Character Animation Instance class.
- **Is In Air** (Bool): True when the shooter character jumps into the air or is falling through the air. False otherwise.
- **Is Moving** (Bool): True when the shooter character's current acceleration vector is greater than zero.
- **Is Sprinting** (Bool): True when the shooter character's current speed matches the shooter character's sprinting movement speed.
- **Current Movement Speed** (Double): Centimeters per second that the shooter character is currently moving at.
- **Equipped Weapon Type** (FName): Type of weapon that the shooter character currently has equipped.
- **Can Use FABRIK** (Bool): True when the shooter character's combat state variable is set to unoccupied or firing.
    - Used to keep the shooter character's hand at a certain location depending on the type of weapon the shooter character has equipped.

Shooter Character Anim Instance Functions:

- Void **UpdateAnimationProperties**()

3.6. Setting up a Shooter Character animation montage

3.6.1. Create an animation montage that uses the same skeleton as your shooter character's first-person mesh.

3.6.2. Drag the animation you want to use from the Asset Browser into the montage.

3.6.3. Create a montage section for each animation. (Example: EquipPistol or ReloadPistol).

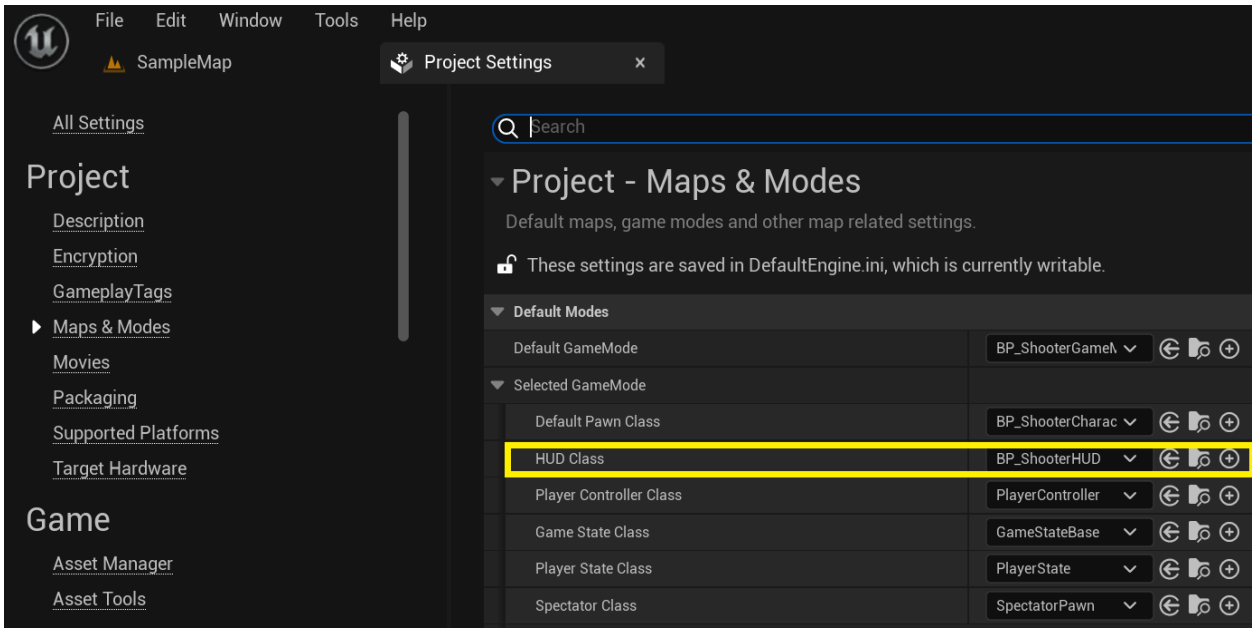
3.6.4. Open the shooter character's blueprint and set the appropriate montage variable (EquipWeaponMontage, FireWeaponMontage, or ReloadWeaponMontage).

3.6.5. Open the blueprint of the weapon that corresponds with the created montage section and set the appropriate montage section name variable. (EquipMontageSectionName, FireMontageSectionName, or ReloadMontageSectionName).

4. Setting up the HUD

4.1. Create a blueprint based on the ShooterHUD class.

4.2. Set your ShooterHUD blueprint as the default HUD Class.



4.3. Import your reticle texture.

4.4. Open your reticle texture and set the Compression Settings to UserInterface2D (RGBA).

4.5. Open your ShooterHUD blueprint and set the Reticle Texture variable to the reticle texture you imported.

4.6. Create a User Widget blueprint and set it as the HUDOverlayClass. (All user widget that you wish to display on the screen must be put on this user widget.)

5. Setting up Weapons

There are three different types of firing methods, hitscan, spread hitscan, and projectile. Hitscan weapons hit their target immediately upon firing, spread hitscan weapons operate in the same manner as hitscan weapons but can fire multiple hitscans at once, projectile weapons fire projectiles that travel to their target at a specified speed. The ShooterCharacter class has a weapon array, so multiple weapons can be carried at once.

5.1. Setting up a Hitscan Weapon

5.1.1. Create a blueprint based on the HitscanWeapon class.

5.1.2. The hitscan weapon's settings can be toggled in the Hitscan Weapon Properties section of the hitscan weapon's blueprint.

5.2. Setting up a Hitscan Spread Weapon

5.2.1. Create a blueprint based on the Shotgun class.

5.2.2. The amount of pellets the shotgun fires can be set in the Shotgun Properties section of the shotgun's blueprint.

5.3. Setting up a Projectile Weapon

5.3.1. Create a blueprint based on the ProjectileWeapon class.

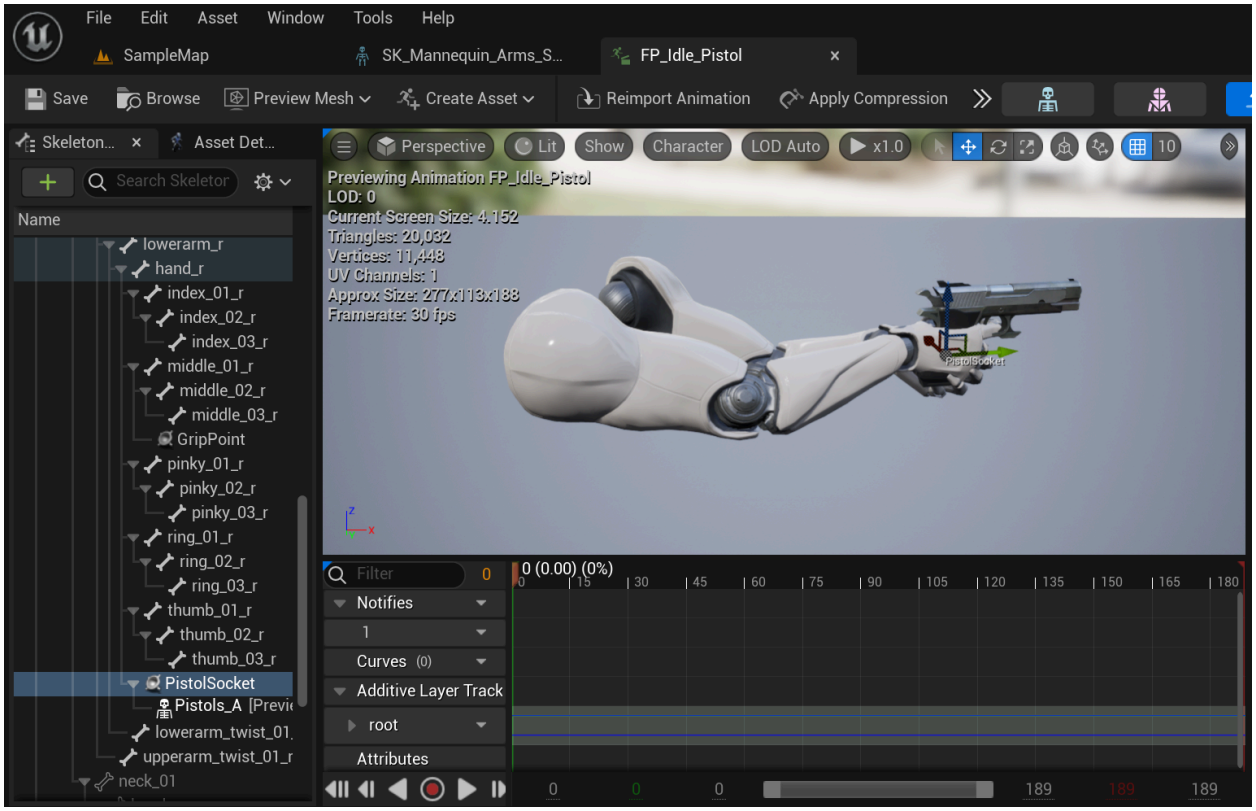
5.3.2. The projectiles that the projectile weapon uses can be set in the Projectile Weapon properties section of the projectile weapon's blueprint.

Weapon notes:

- The weapon's blueprint-exposed variables can be found in the Weapon Properties section of its blueprint.
- If the weapon asset you are using has a firing animation, then you can play the weapon's fire sound and spawn the weapon's muzzle flash particles within the weapon's fire animation using Notifies and leave those variables blank in the weapon's blueprint. But if the weapon asset you are using does not have a fire animation, then you can select the fire sound and muzzle flash particles individually in the weapon's blueprint or even leave them blank if you do not have such assets.
- Giving a weapon the WeaponType "SniperRifle" will give that weapon the ability to zoom in by pressing the secondary fire action button.

Any mesh that you wish to attach weapons to needs a socket that the weapon can attach to. To add a socket to a character's mesh, open the skeleton that the character's mesh uses, right-click the bone in the Skeleton Tree that you wish to add a socket to, select Add Socket, and give the socket a name such as "PistolSocket" or "AssaultRifleSocket."

You can then add a preview asset to the socket such as the mesh of the weapon so you can see how the weapon will look once it has attached to the character's mesh. To add a preview asset to the socket, right-click on the added socket, select Add Preview Asset, select the mesh of the weapon you wish to equip. Then move and rotate the socket until you are satisfied with the way it looks.



Weapon Blueprint-Exposed Functions:

- FVector **GetScatterShotLocation**(const FVector& FireHitLocation, const double InFireSphereRadius);
- Void **UpdateClipAmmoAmount**(const int32 AmmoAmountToAdd);
- Void **SpawnBulletCasing**();
- FVector **GetMuzzleSocketLocation**();

6. Setting up Projectiles

There are two different types of projectiles, regular projectiles that are designed to apply damage only to an area of its size, and explosive projectiles that are designed to apply damage to all characters in a radius that the user can specify.

6.1. Setting up a Projectile

6.1.1. Create a blueprint based on the Projectile class.

6.2. Setting up an Explosive Projectile

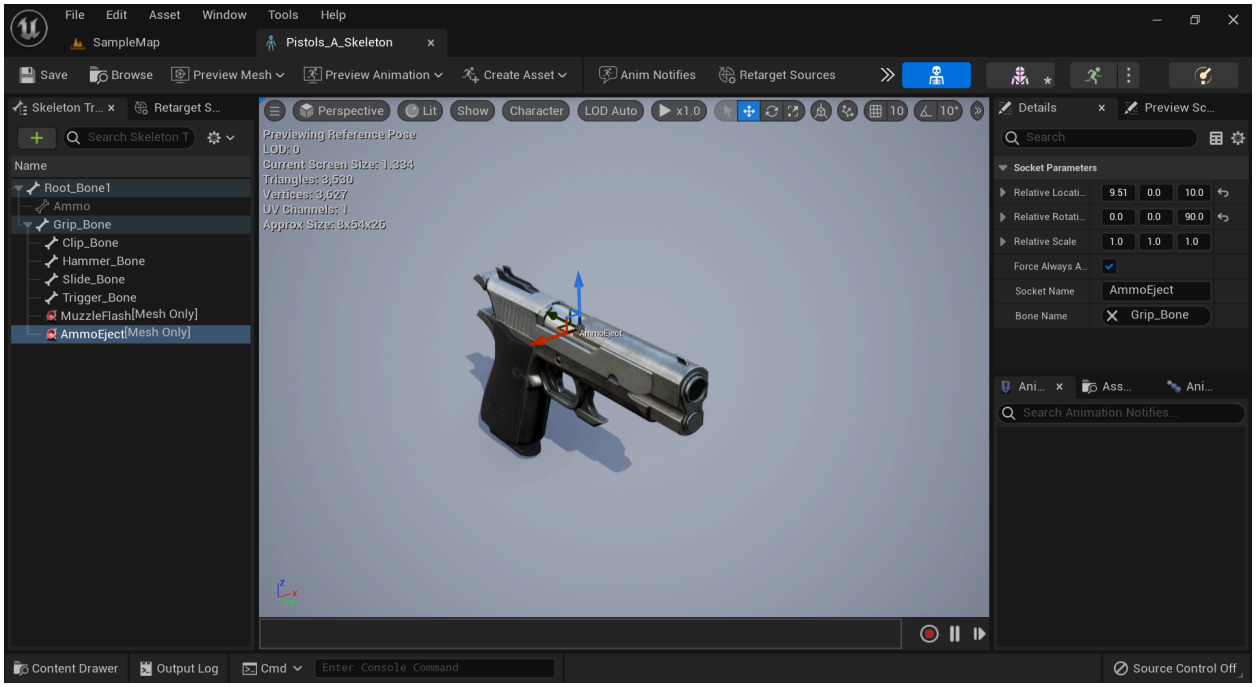
6.2.1. Create a blueprint based on the ExplosiveProjectile class.

Select the Projectile Movement Component in the Components tab, and adjust the Initial Speed and Max Speed to your liking.

The projectile's blueprint-exposed variables can be found in the Projectile Properties section of its blueprint.

7. Setting up a Bullet Casing

Make sure the socket on your weapon's skeleton that the casing will eject from is rotated in a way where the x-axis (red arrow) is facing the direction you want the casing to travel in once it is ejected.



7.1. Create a blueprint based off of the BulletCasing class.

7.2. Set the bullet casing's mesh.

7.3. Open the blueprint of the weapon this casing should eject from and set the Bullet Casing Class to the bullet casing blueprint you created.

The bullet casing's blueprint-exposed variables can be found in the Bullet Casing Properties section of its blueprint.

8. Setting up NPCs

There are two types of NPCs: allies and enemies. Allies can join and follow the player as well as target and fight enemies. Enemies will target the player and allies. There are two types of enemies: melee enemies and shooter enemies.

All NPCs have a Focus Sphere component which can set the NPC's target and an Attack Sphere component that when overlapped by their target will cause the NPC to attack.

The NPC's blueprint-exposed variables can be found in the NPC Properties section of its blueprint.

NPC Blueprint-Exposed Functions:

- Void **Attack**()
- Void **Die**()
- Void **MoveToNextPatrolLocation**()
- Void **MoveToPatrolLocationAtIndex**(const int32 InPatrolLocationIndex)
- Void **SetTarget**(const AActor* Target)

8.1. Setting up an Enemy

There are two types of enemies: melee enemies and shooter enemies. Melee enemies are designed to run up to the player and attack at close-range, shooter enemies are designed to fire their gun at the player from medium or long-range.

8.1.1. Setting up a Melee Enemy

Some components of the melee enemy class are the Melee Weapon Mesh, the Damage Box, which is designed to be activated in the melee enemy's attack animation montage so it can apply damage to their target.

8.1.1.1. Create a blueprint based on the MeleeEnemy class.

8.1.1.2. Select the melee enemy's skeletal mesh.

8.1.1.3. Move and rotate the melee enemy's mesh so it fits within the melee enemy's capsule component and is facing the same direction as the Arrow Component.

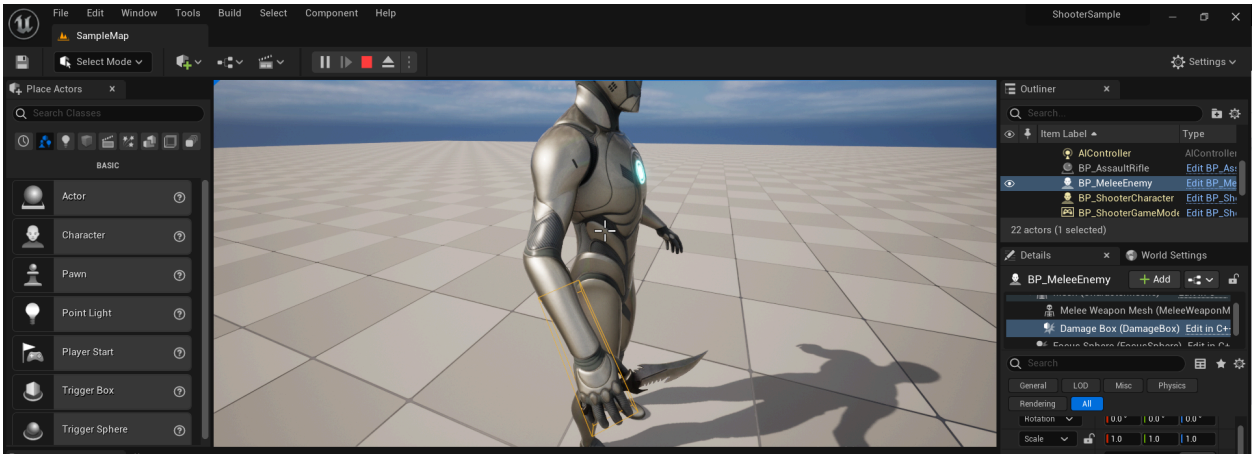
8.1.1.4. Select the melee enemy's melee weapon mesh.

8.1.1.5. Add a socket on the melee enemy's skeleton that the melee weapon's mesh can attach to.

8.1.1.6. Set the melee enemy's Weapon Attach Socket Name variable to the name of the socket that you added to the melee enemy's skeleton.

8.1.1.7. Add a socket on the melee enemy's skeleton that the Damage Box can attach to.

8.1.1.8. Set the melee enemy's Damage Box Socket Name variable to the name of the damage box socket that you added to the melee enemy's skeleton.



8.1.1.9. Adjust the size of the melee enemy's Damage Box to your liking. (You can select the Damage Box in the melee enemy's blueprint and set Hidden in Game in the Details panel to false to see how the Damage Box looks on the mesh in-game.)

Setting up a melee enemy attack animation montage

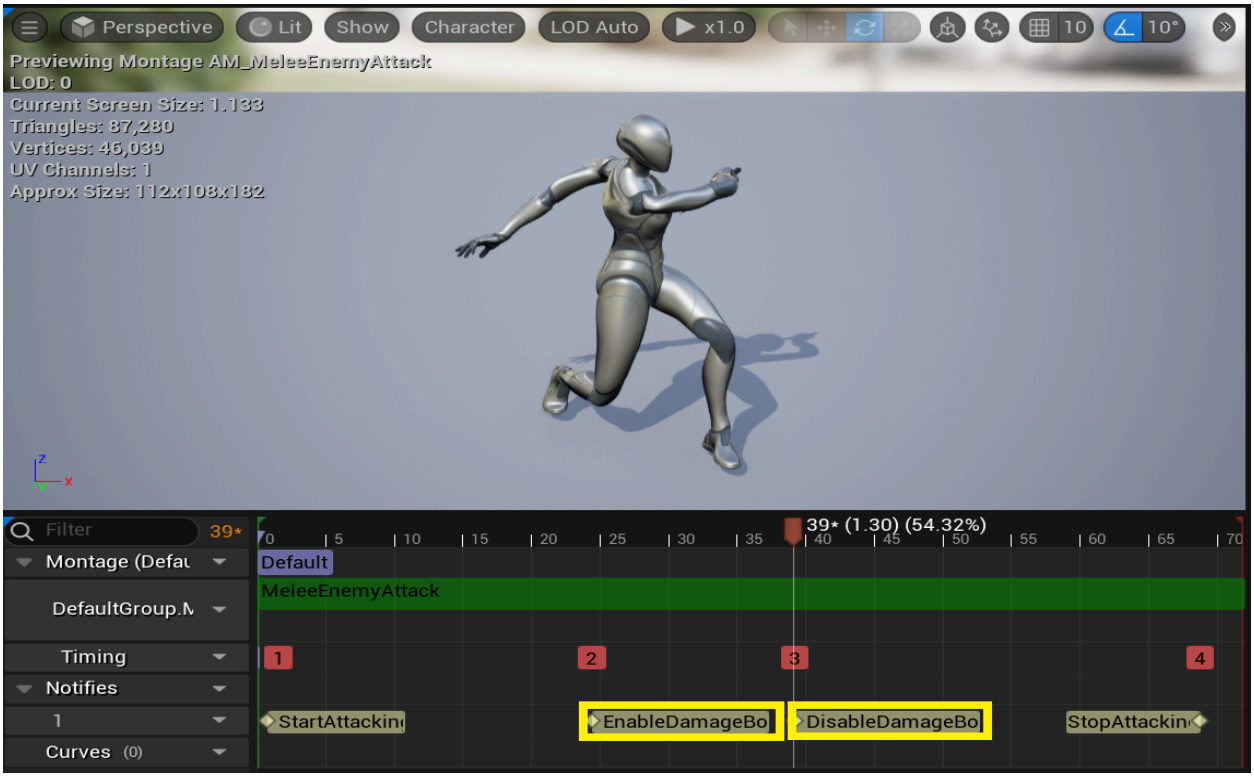
8.1.1.10. Create an animation montage that uses the same skeleton as your melee enemy.

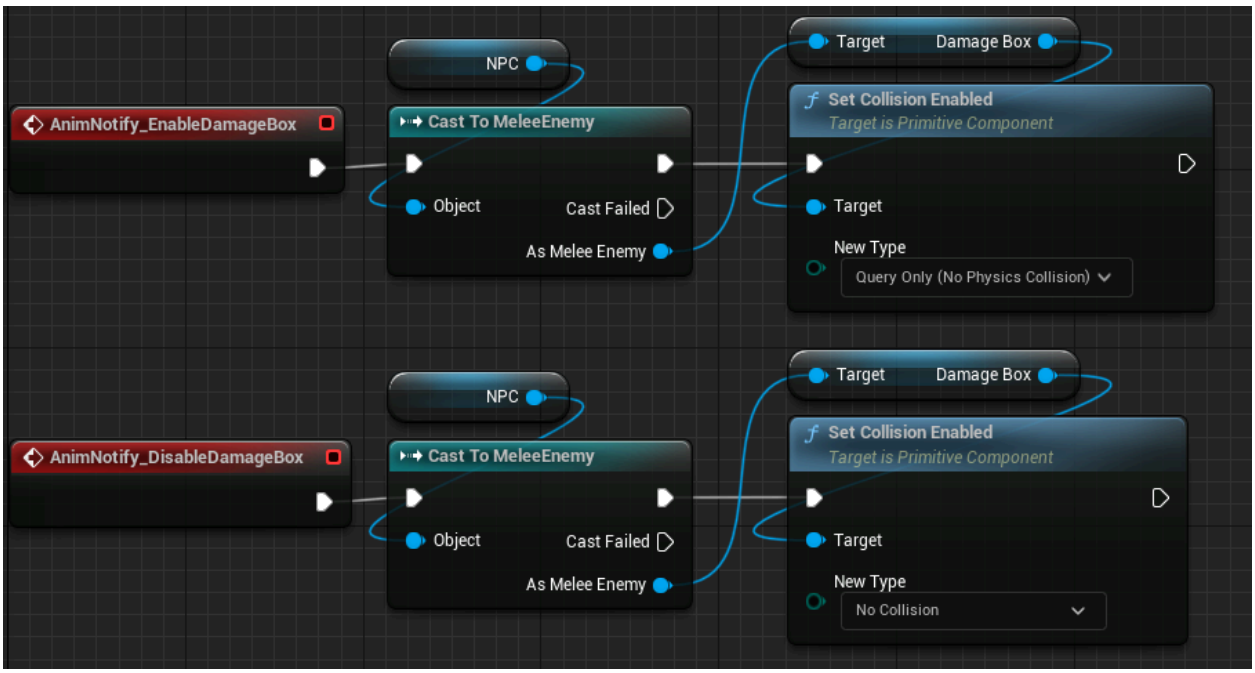
8.1.1.11. Drag your melee enemy's attack animations into the montage.

8.1.1.12. Use the start and stop attacking notifies in the Event Graph of the melee enemy's mesh's animation blueprint.

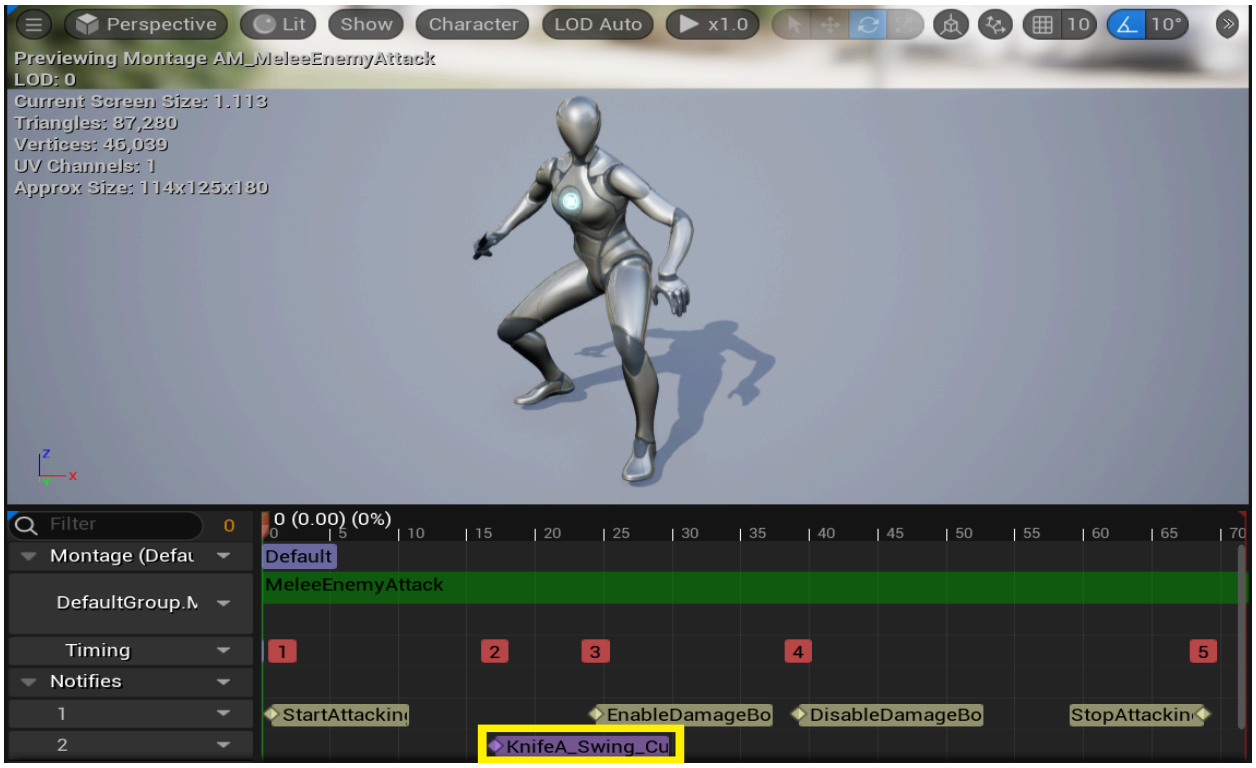
8.1.1.13. Create notifies at points in the attack animations where the melee enemy's melee damage box should be enabled and disabled.

8.1.1.14. Use the enable and disable damage box notifies in the Event Graph of the melee enemy's mesh's animation blueprint.





8.1.1.15. (Optional) Add a Play Sound notify that will play a weapon swinging noise.



8.1.1.16. Set the melee enemy's Attack Animation Montage variable to the attack animation montage you created for it.

8.1.1.17. Create and set the melee enemy's behavior tree variable.

8.1.2. Setting up a Shooter Enemy

8.1.2.1. Create a blueprint based on the ShooterEnemy class.

8.1.2.2. Select the shooter enemy's skeletal mesh.

8.1.2.3. Move and rotate the shooter enemy's skeletal mesh so it fits inside the capsule component and is facing the same direction as the Arrow Component.

8.1.2.4. Add a socket to the shooter enemy's skeleton that a weapon can attach to.

8.1.2.5. Set the shooter enemy's Weapon Socket Name variable to the name you gave the socket on the shooter enemy's skeleton that you just added.

8.1.2.6. Set the shooter enemy's Weapon Class variable to a blueprint of the weapon you want the shooter enemy to equip.

8.1.2.7. Adjust the Focus Sphere radius to a size from which you would like the enemy to shoot from.

8.1.2.8. Create the shooter enemy's attack animation montage.

8.1.2.9. Set the shooter enemy's Attack Montage variable to the attack animation montage you created for it.

8.1.2.10. Create and set the shooter enemy's behavior tree variable.

8.1.2.11. (Optional) Set the shooter enemy's Reload Montage variable.

8.2. Setting up an Ally

There are two types of allies: Regular allies that join the player and attack enemies, and medic allies that do the same but can also heal the player. The player can make an ally join them by overlapping with the ally's focus sphere and pressing the Select button.

In order to make an ally join you, make sure the shooter character's ally trace collision channel variable is set to a collision channel that is blocked by the ally's capsule or mesh and press the select button.

8.2.1. Setting up a default Ally

8.2.1.1. Create a blueprint based on the Ally class.

8.2.1.2. Select the ally's skeletal mesh.

8.2.1.3. Move and rotate the ally's skeletal mesh so it fits inside the capsule component and is facing the same direction as the Arrow Component.

8.2.1.4. Add a socket to the ally's skeleton for their weapon to attach to.

8.2.1.5. Set the ally's Weapon Socket Name variable to the name you gave the socket on the ally's skeleton that you just added.

8.2.1.6. Set the ally's Weapon Class variable to a blueprint of the weapon you want the shooter ally to equip.

8.2.1.7. Adjust the Focus Sphere radius to a size from which you would like the ally to shoot from.

8.2.1.8. Create the shooter ally's attack animation montage.

8.2.1.9. Set the shooter ally's Attack Montage variable to the attack animation montage you created for it.

8.2.1.10. Set the shooter ally's AI Controller in the Details panel to a blueprint based on the AllyController class.

8.2.1.11. Create and set the shooter ally's behavior tree variable.

8.2.1.12. (Optional) Set the ally's Reload Montage variable.

The ally's blueprint-exposed variables can be found in the Ally Properties section of its blueprint.

8.2.2. Setting up a Medic Ally

Medic allies act just as regular allies do, but they have an additional sphere component known as the Heal Sphere. When this sphere is overlapped by the player, the medic ally will check if the player can be healed, then heal the player if they can.

8.2.2.1. Create a blueprint based on the MedicAlly class.

8.2.2.2. Set the Heal Sphere component's radius to a size that the medic ally should be able to heal from.

8.2.2.3. Set the variables in the Medic Ally Properties section.

8.2.2.4. Follow the steps of setting up a default ally. (Section 8.2.1)

The medic ally's blueprint-exposed variables can be found in the Medic Ally Properties section of its blueprint.

8.3. Setting up an NPC animation blueprint

8.3.1. Create an animation blueprint that uses the same skeleton as your NPC's mesh and NPCAnimInstance as the parent class.

8.3.2. Open the NPC's animation blueprint, go to the event graph, drag off of the Event Blueprint Update Animation node's execution pin, call Update Animation Properties from the NPCAnimInstance class.

NPC Animation Variables:

- **NPC Owner** (Character): NPC that is using the animation blueprint.
- **Current Movement Speed** (Double): Centimeters per second that the NPC is moving at.

8.4. Setting up an NPC behavior tree

8.4.1. Create a blackboard.

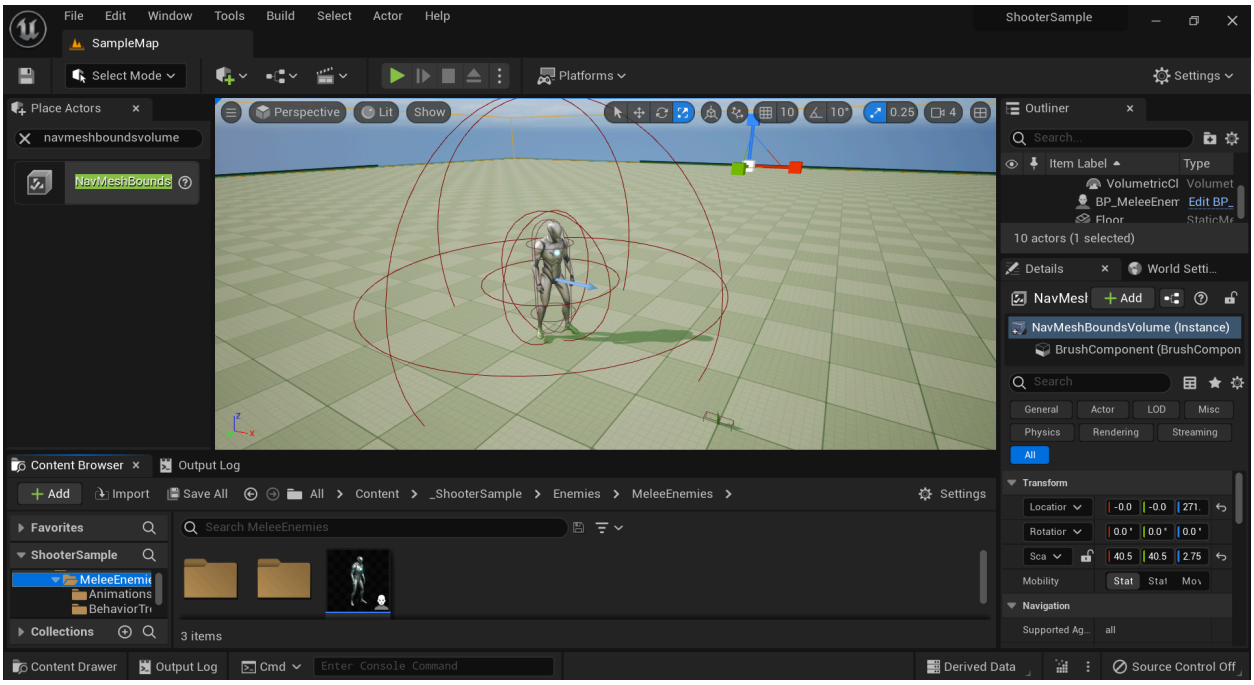
8.4.2. Use the New Key button to add the following variables:

- PatrolLocation (Vector)
- Target (Object) (Select Target, click the drop-down menu next to Key Type in the Blackboard Details tab, and set Base Class to Actor.)
- IsDead (Bool)
- ShooterCharacterIsDead (Bool)
- TargetIsVisible (Bool)
- NPCCombatState (Enum) (Select NPCCombatState, click the drop-down menu next to Key Type in the Blackboard Details tab, and set the Enum Name to ECombatState.)

8.4.3. Create a behavior tree.

8.4.4. Open the behavior tree, select the drop-down menu next to BehaviorTree in the Details panel, make sure the Blackboard Asset is set to the blackboard you just created.

8.4.5. Drag a Nav Mesh Bounds Volume actor into the map and scale it across the entire area that you wish to have NPCs traverse. (You can press P with the Nav Mesh Bounds Volume selected to see the area where NPCs can traverse.)



9. Setting up Pickups

There are two different types of pickups: health pickups and ammo pickups. Every pickup has an Overlap Sphere and a PickUp Sphere. When the shooter character overlaps with the Overlap Sphere, the pickup can be picked up by pressing the Select button. When the shooter character overlaps with the PickUp Sphere, the pickup will automatically be picked up.

9.1. Setting up Health Pickups

9.1.1. Create a blueprint based on the HealthPickup class.

9.1.2. Set the health pickup's Pickup Name variable.

9.1.3. Set the health pickup's Pickup Mesh.

9.2. Setting up Ammo Pickups

9.2.1. Create a blueprint based on the AmmoPickup class.

9.2.2. Set the Ammo Type variable to

9.2.3. Set the Pickup Name variable (ideally to the same name as the type of ammo that it is.)

9.2.4. Set the Pickup Amount to the amount that you wish to be added to the shooter character's ammo map when the ammo pickup is picked up.

9.2.5. Set the ammo pickup's Pickup Mesh.

The pickup's blueprint-exposed variables can be found in the Pickup Properties section of its blueprint.

10. Setting up Explosives

Explosives are designed to apply damage and impulse to all surrounding actors within a specified radius upon exploding.

10.1. Create a blueprint based on the Explosive class.

10.2. Set the Explosive's mesh.

The explosive's blueprint-exposed variables can be found in the Explosive Properties section of its blueprint.