Whiskey Business

Architecture/Design Document

Weapons System

Table of Contents

Change History

Version: 0.1

Modifier: Filipe Botelho

Date: 2025-02-17

Description of Change: Initial setup of Base and derived classes

Version: 0.2

Modifier: Filipe Botelho

Date: 2025-02-24

Description of Change: Added Aiming with a laser pointer

Version: 0.3

Modifier: Filipe Botelho

Date: 2025-03-02

Description of Change: changed laser pointer to draw line for more accuracy and changed activation times using AnimNotifyState

Version: 0.4

Modifier: Filipe Botelho

Date: 2025-03-22

Description of Change: Created Delegates for attack and special attack and implemented the new Custom Damage System component

Version: 0.5

Modifier: Filipe Botelho

Date: 2025-03-23

Description of Change: Created Timers for attacks for cooldowns and to help with the animation lock

1. Introduction

This Document is to describe the Design of the Weapons System in the game Whiskey Business. It will go over the Design Goals and the decisions made in creating the weapons and how they interact in the game. It will also aid future development in creating new weapons in the game.

2. Design Goals

- Transitive Design: Weapons will be using Inheritance OOP design. Sharing as many similarities in the base class to all weapon types.
- Ease of Addition: The Design will make it as seamless as possible to add additional weapons in the future.

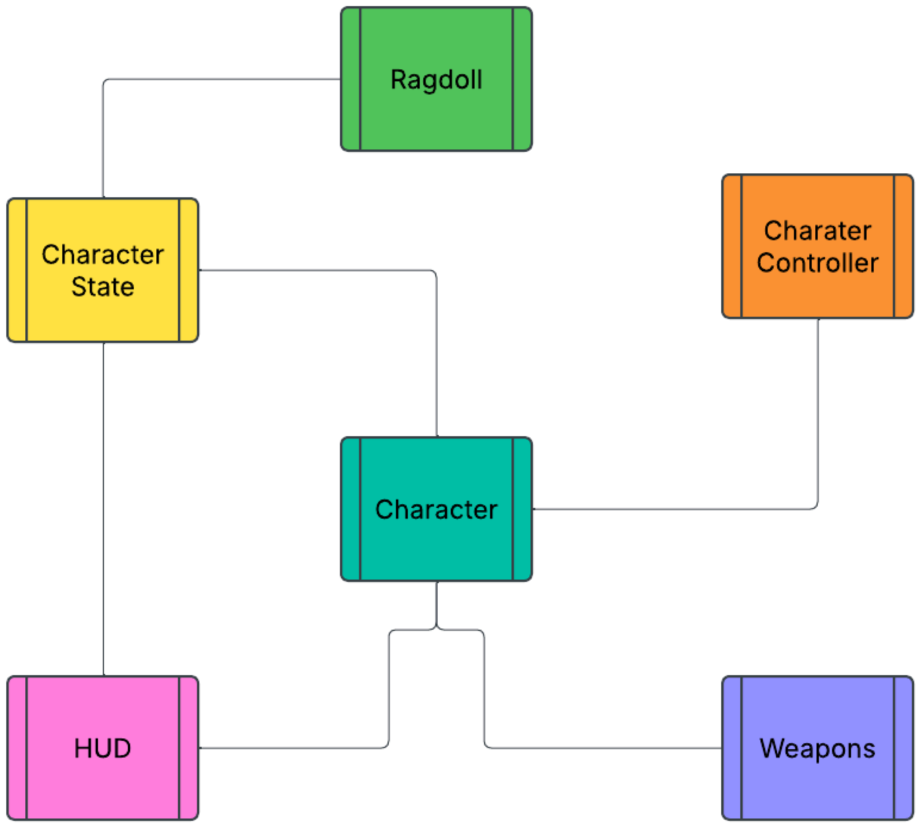
3. System Behavior

Weapons will appear on the map and can be picked up by the players. Using weapons will apply varying amounts of Damage, Stagger Damage (for ragdolling) and force dead on targets depending on the weapon.

4. Logical View

Diagrams showing the relationships and interaction between systems.

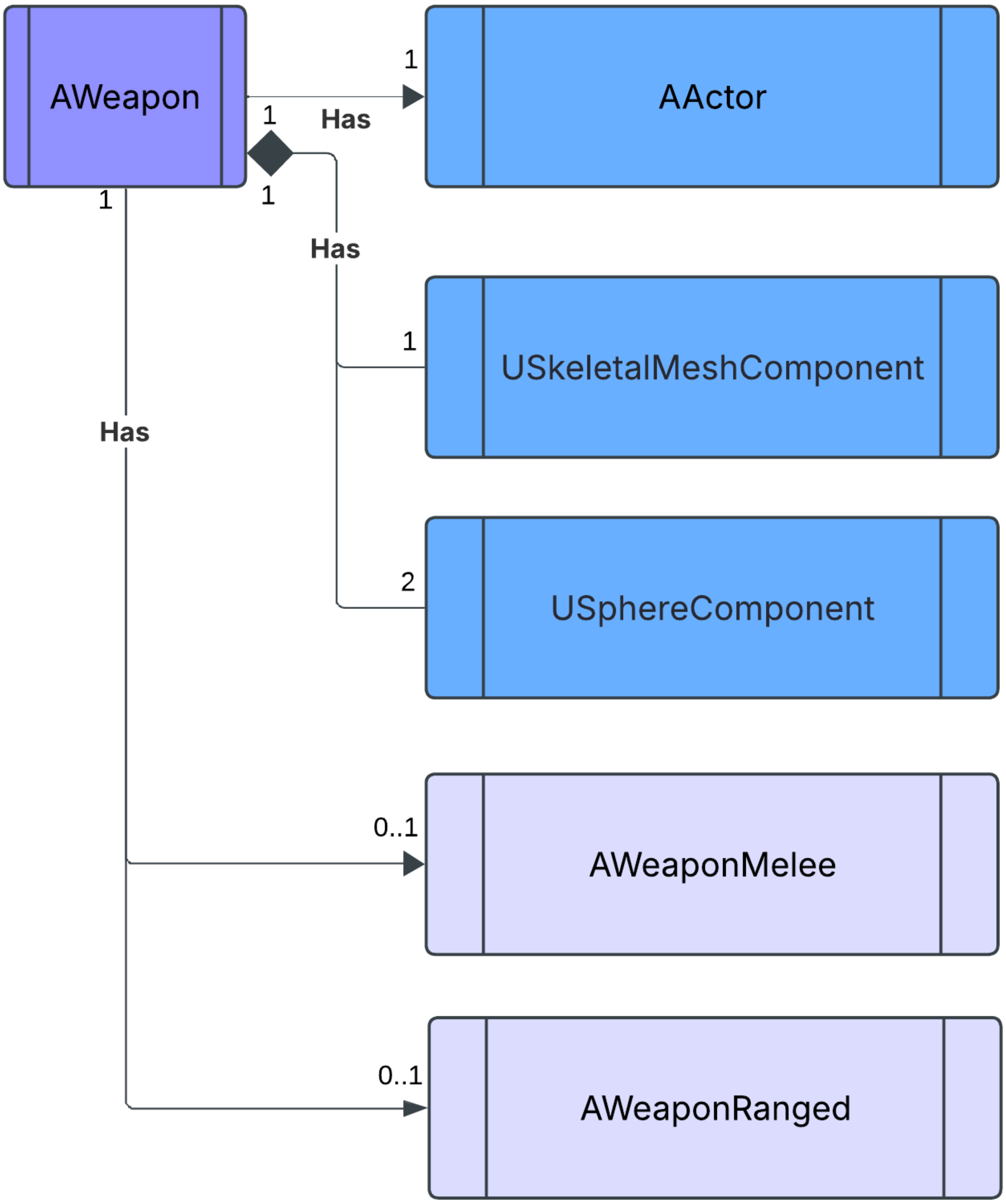
4.1 High-Level Design



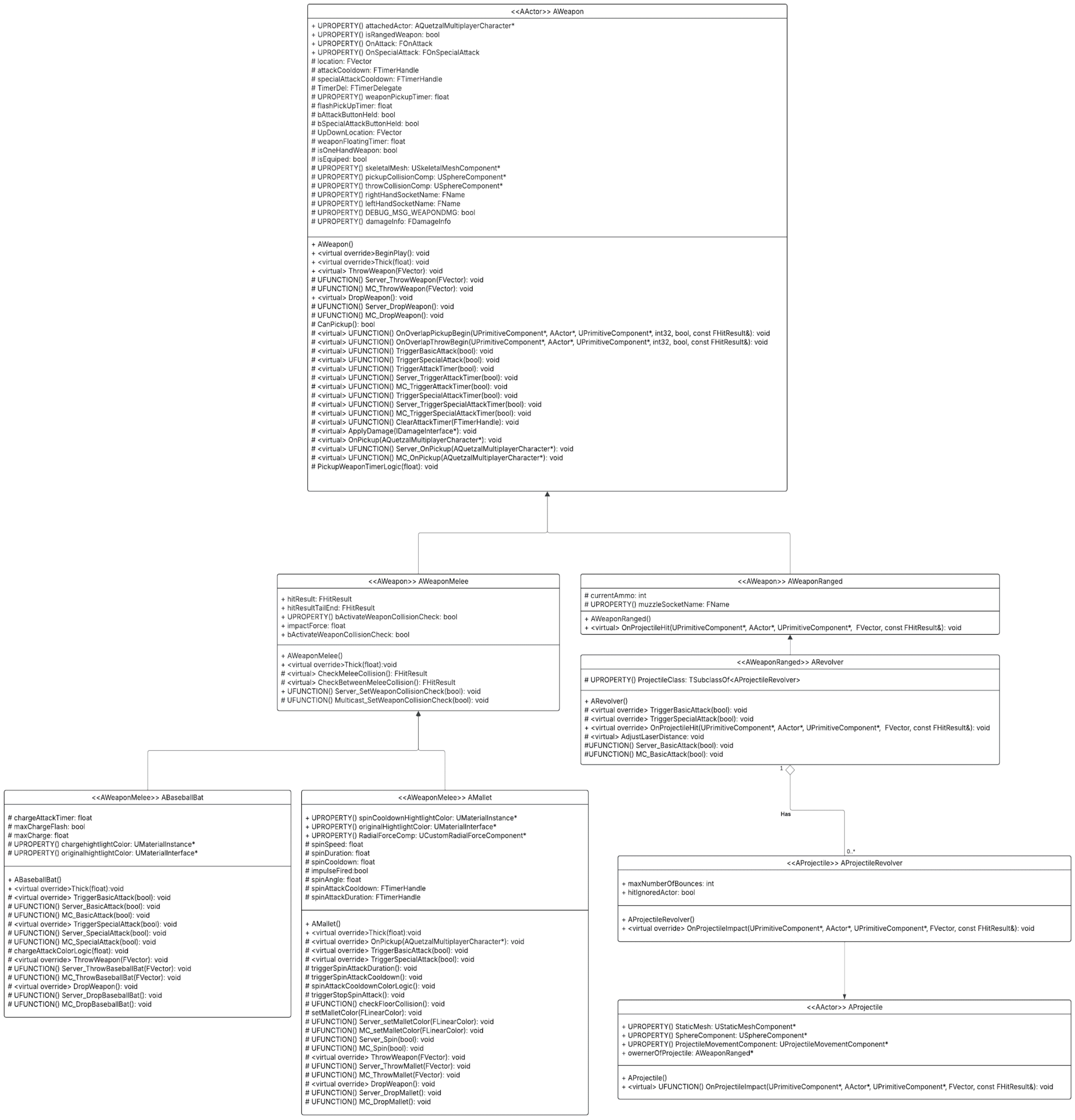
High-Level Design of varying systems in the game.

- The Character Controller will handle the input from the user that will control the player.
- The HUD will keep the players informed of the current health and score of the players using values from Character and Character State.
- Weapons will interact with the player by activating equipped weapons or receiving damage from weapons.
- Depending on the Character State the player can enter into a Ragdoll state.
- At the center is the Character interacting with all systems in some varying ways.

4.2 Mid-Level Design



4.3 Detailed Class Design



```
<<AActor>> AWeapon
+ UPROPERTY() attachedActor: AQuetzalMultiplayerCharacter*
+ UPROPERTY() isRangedWeapon: bool
+ UPROPERTY() OnAttack: FOnAttack
+ UPROPERTY() OnSpecialAttack: FOnSpecialAttack
# location: FVector
# attackCooldown: FTimerHandle
# specialAttackCooldown: FTimerHandle
# TimerDel: FTimerDelegate
# UPROPERTY() weaponPickupTimer: float
# flashPickUpTimer: float
# bAttackButtonHeld: bool
# bSpecialAttackButtonHeld: bool
# UpDownLocation: FVector
# weaponFloatingTimer: float
# isOneHandWeapon: bool
# isEquiped: bool
# UPROPERTY() skeletalMesh: USkeletalMeshComponent*
# UPROPERTY() pickupCollisionComp: USphereComponent*
# UPROPERTY() throwCollisionComp: USphereComponent*
# UPROPERTY() rightHandSocketName: FName
# UPROPERTY() leftHandSocketName: FName
# UPROPERTY() DEBUG_MSG_WEAPONDMG: bool
# UPROPERTY() damageInfo: FDamageInfo

+ AWeapon()
+ <virtual override>BeginPlay(): void
+ <virtual override>Thick(float): void
+ <virtual> ThrowWeapon(FVector): void
# UFUNCTION() Server_ThrowWeapon(FVector): void
# UFUNCTION() MC_ThrowWeapon(FVector): void
+ <virtual> DropWeapon(): void
# UFUNCTION() Server_DropWeapon(): void
# UFUNCTION() MC_DropWeapon(): void
# CanPickup(): bool
# <virtual> UFUNCTION() OnOverlapPickupBegin(UPrimitiveComponent*, AActor*, UPrimitiveComponent*, int32, bool, const FHitResult&): void
# <virtual> UFUNCTION() OnOverlapThrowBegin(UPrimitiveComponent*, AActor*, UPrimitiveComponent*, int32, bool, const FHitResult&): void
# <virtual> UFUNCTION() TriggerBasicAttack(bool): void
# <virtual> UFUNCTION() TriggerSpecialAttack(bool): void
# <virtual> UFUNCTION() TriggerAttackTimer(bool): void
# <virtual> UFUNCTION() Server_TriggerAttackTimer(bool): void
# <virtual> UFUNCTION() MC_TriggerAttackTimer(bool): void
# <virtual> UFUNCTION() TriggerSpecialAttackTimer(bool): void
# <virtual> UFUNCTION() Server_TriggerSpecialAttackTimer(bool): void
# <virtual> UFUNCTION() MC_TriggerSpecialAttackTimer(bool): void
# <virtual> UFUNCTION() ClearAttackTimer(FTimerHandle): void
# <virtual> ApplyDamage(IDamageInterface*): void
# <virtual> OnPickup(AQuetzalMultiplayerCharacter*): void
# <virtual> UFUNCTION() Server_OnPickup(AQuetzalMultiplayerCharacter*): void
# <virtual> UFUNCTION() MC_OnPickup(AQuetzalMultiplayerCharacter*): void
# PickupWeaponTimerLogic(float): void
```

```
<<AWeapon>> AWeaponMelee
+ hitResult: FHitResult
+ hitResultTailEnd: FHitResult
+ UPROPERTY() bActivateWeaponCollisionCheck: bool
+ impactForce: float
+ bActivateWeaponCollisionCheck: bool

+ AWeaponMelee()
+ <virtual override>Thick(float):void
# <virtual> CheckMeleeCollision(): FHitResult
# <virtual> CheckBetweenMeleeCollision(): FHitResult
+ UFUNCTION() Server_SetWeaponCollisionCheck(bool): void
# UFUNCTION() Multicast_SetWeaponCollisionCheck(bool): void
```

```
<<AWeapon>> AWeaponRanged
# currentAmmo: int
# UPROPERTY() muzzleSocketName: FName

+ AWeaponRanged()
+ <virtual> OnProjectileHit(UPrimitiveComponent*, AActor*, UPrimitiveComponent*,  FVector, const FHitResult&): void
```

```
<<AWeaponRanged>> ARevolver
# UPROPERTY() ProjectileClass: TSubclassOf<AProjectileRevolver>

+ ARevolver()
# <virtual override> TriggerBasicAttack(bool): void
# <virtual override> TriggerSpecialAttack(bool): void
+ <virtual override> OnProjectileHit(UPrimitiveComponent*, AActor*, UPrimitiveComponent*,  FVector, const FHitResult&): void
# <virtual> AdjustLaserDistance: void
#UFUNCTION() Server_BasicAttack(bool): void
#UFUNCTION() MC_BasicAttack(bool): void
```

```
<<AWeaponMelee>> ABaseballBat
# chargeAttackTimer: float
# maxChargeFlash: bool
# maxCharge: float
# UPROPERTY() chargehightlightColor: UMaterialInstance*
# UPROPERTY() originalhightlightColor: UMaterialInterface*

+ ABaseballBat()
+ <virtual override>Thick(float):void
# <virtual override> TriggerBasicAttack(bool): void
# UFUNCTION() Server_BasicAttack(bool): void
# UFUNCTION() MC_BasicAttack(bool): void
# <virtual override> TriggerSpecialAttack(bool): void
# UFUNCTION() Server_SpecialAttack(bool): void
# UFUNCTION() MC_SpecialAttack(bool): void
# chargeAttackColorLogic(float): void
# <virtual override> ThrowWeapon(FVector): void
# UFUNCTION() Server_ThrowBaseballBat(FVector): void
# UFUNCTION() MC_ThrowBaseballBat(FVector): void
# <virtual override> DropWeapon(): void
# UFUNCTION() Server_DropBaseballBat(): void
# UFUNCTION() MC_DropBaseballBat(): void
```

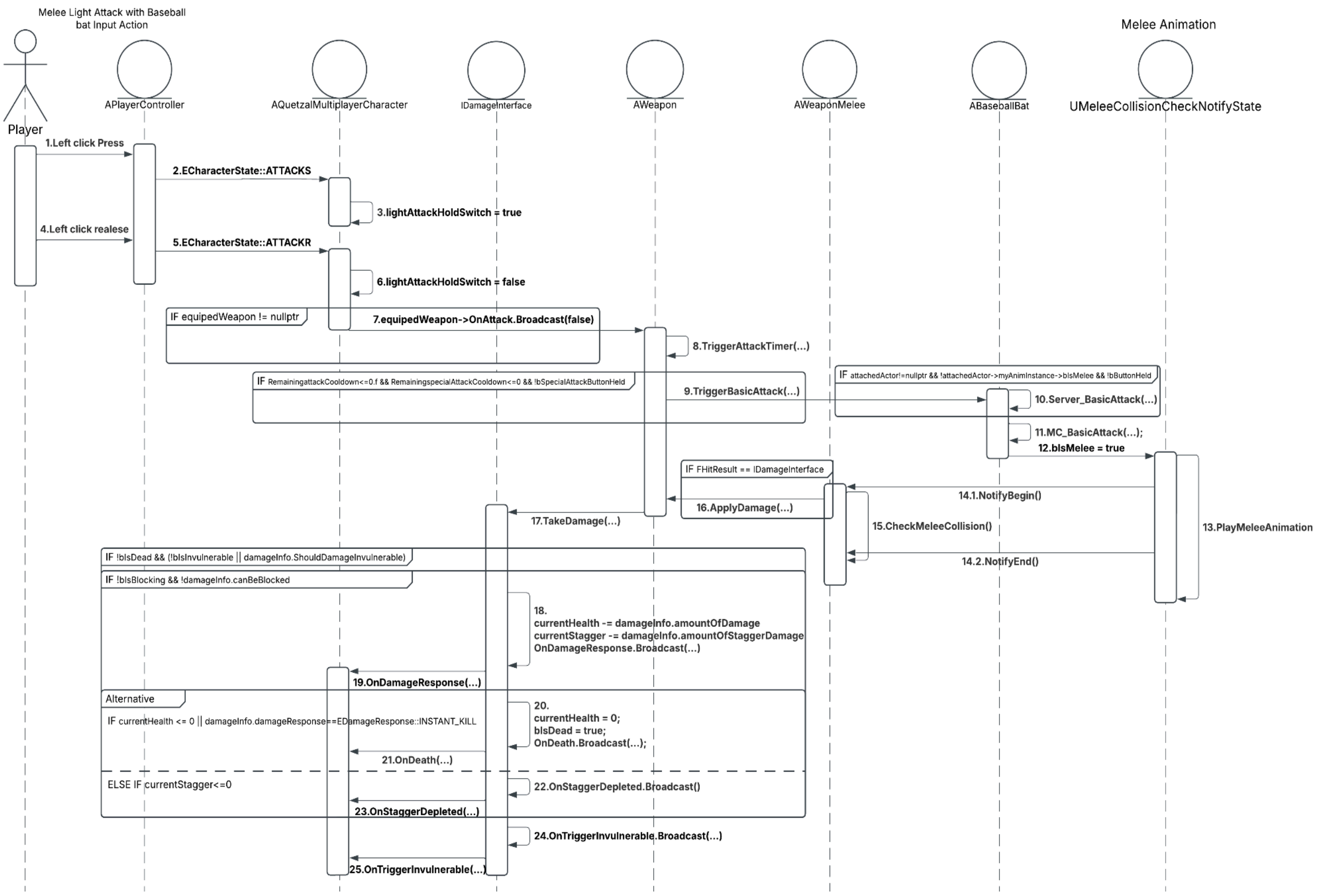
```
<<AWeaponMelee>> AMallet
+ UPROPERTY() spinCooldownHightlightColor: UMaterialInstance*
+ UPROPERTY() originalHightlightColor: UMaterialInterface*
+ UPROPERTY() RadialForceComp: UCustomRadialForceComponent*
# spinSpeed: float
# spinDuration: float
# spinCooldown: float
# impulseFired:bool
# spinAngle: float
# spinAttackCooldown: FTimerHandle
# spinAttackDuration: FTimerHandle

+ AMallet()
+ <virtual override>Thick(float):void
# <virtual override> OnPickup(AQuetzalMultiplayerCharacter*): void
# <virtual override> TriggerBasicAttack(bool): void
# <virtual override> TriggerSpecialAttack(bool): void
# triggerSpinAttackDuration(): void
# triggerSpinAttackCooldown(): void
# spinAttackCooldownColorLogic(): void
# triggerStopSpinAttack(): void
# UFUNCTION() checkFloorCollision(): void
# setMalletColor(FLinearColor): void
# UFUNCTION() Server_setMalletColor(FLinearColor): void
# UFUNCTION() MC_setMalletColor(FLinearColor): void
# UFUNCTION() Server_Spin(bool): void
# UFUNCTION() MC_Spin(bool): void
# <virtual override> ThrowWeapon(FVector): void
# UFUNCTION() Server_ThrowMallet(FVector): void
# UFUNCTION() MC_ThrowMallet(FVector): void
# <virtual override> DropWeapon(): void
# UFUNCTION() Server_DropMallet(): void
# UFUNCTION() MC_DropMallet(): void
```

```
<<AProjectile>> AProjectileRevolver
+ maxNumberOfBounces: int
+ hitIgnoredActor: bool

+ AProjectileRevolver()
+ <virtual override> OnProjectileImpact(UPrimitiveComponent*, AActor*, UPrimitiveComponent*, FVector, const FHitResult&): void
```

```
<<AActor>> AProjectile
+ UPROPERTY() StaticMesh: UStaticMeshComponent*
+ UPROPERTY() SphereComponent: USphereComponent*
+ UPROPERTY() ProjectileMovementComponent: UProjectileMovementComponent*
+ owernerOfProjectile: AWeaponRanged*

+ AProjectile()
+ <virtual> UFUNCTION() OnProjectileImpact(UPrimitiveComponent*, AActor*, UPrimitiveComponent*, FVector, const FHitResult&): void
```

5. Process View of the Weapons System

Light Melee Attack

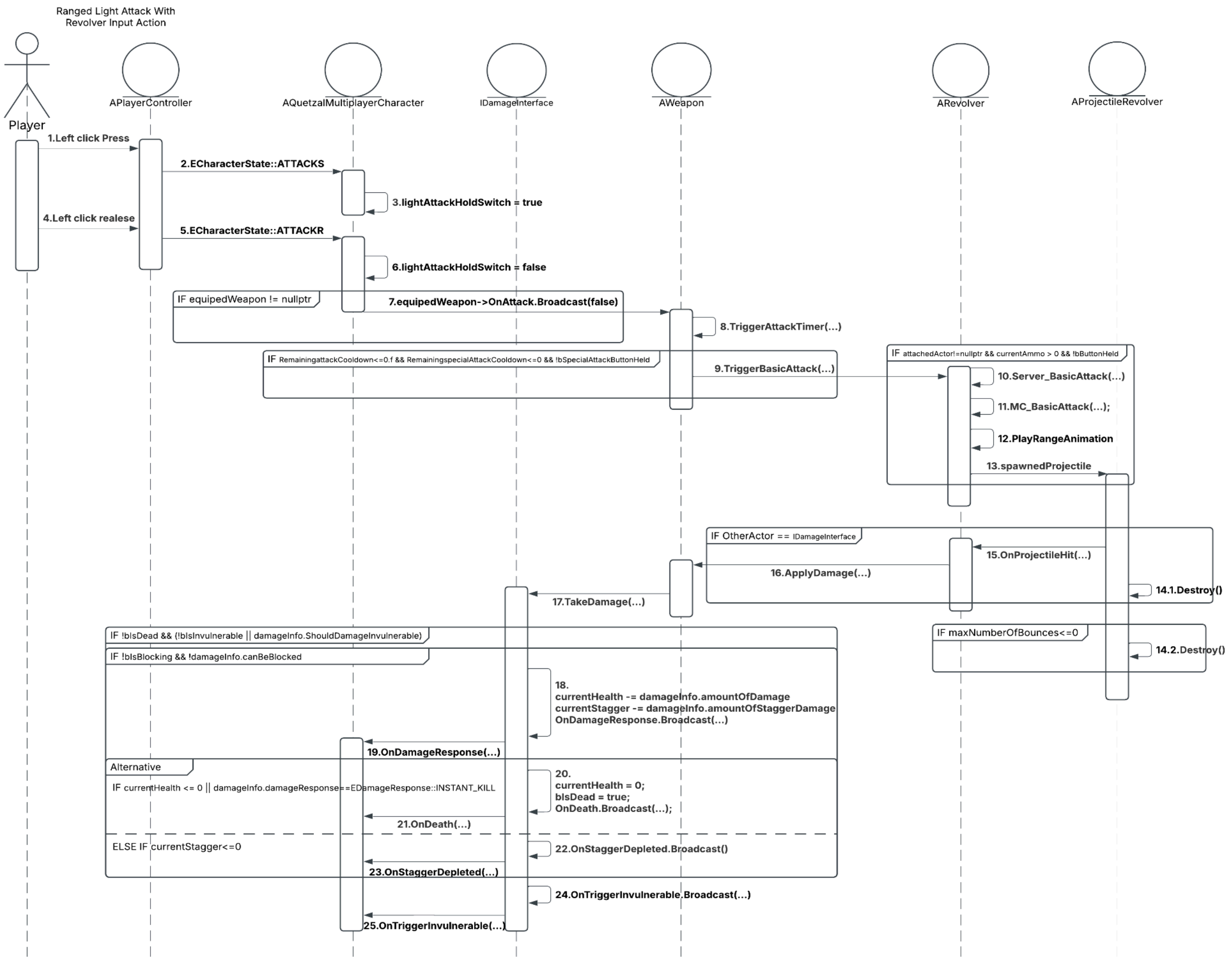


Melee Light Attack with Baseball bat Input Action
Melee Animation
Player
APlayerController
AQuetzalMultiplayerCharacter
IDamageInterface
AWeapon
AWeaponMelee
ABaseballBat
UMeleeCollisionCheckNotifyState
1.Left click Press
2.ECharacterState::ATTACKS
3.lightAttackHoldSwitch = true
4.Left click realese
5.ECharacterState::ATTACKR
6.lightAttackHoldSwitch = false
IF equipedWeapon != nullptr
7.equipedWeapon->OnAttack.Broadcast(false)
8.TriggerAttackTimer(...)
IF RemainingattackCooldown<=0.f && RemainingspecialAttackCooldown<=0 && !bSpecialAttackButtonHeld
9.TriggerBasicAttack(...)
IF attachedActor!=nullptr && !attachedActor->myAnimInstance->bIsMelee && !bButtonHeld
10.Server_BasicAttack(...)
11.MC_BasicAttack(...);
12.bIsMelee = true
13.PlayMeleeAnimation
IF FHitResult == IDamageInterface
14.1.NotifyBegin()
15.CheckMeleeCollision()
14.2.NotifyEnd()
16.ApplyDamage(...)
17.TakeDamage(...)
IF !bIsDead && (!bIsInvulnerable || damageInfo.ShouldDamageInvulnerable)
IF !bIsBlocking && !damageInfo.canBeBlocked
18. currentHealth -= damageInfo.amountOfDamage currentStagger -= damageInfo.amountOfStaggerDamage OnDamageResponse.Broadcast(...)
19.OnDamageResponse(...)
Alternative
IF currentHealth <= 0 || damageInfo.damageResponse==EDamageResponse::INSTANT_KILL
20. currentHealth = 0; bIsDead = true; OnDeath.Broadcast(...);
21.OnDeath(...)
ELSE IF currentStagger<=0
22.OnStaggerDepleted.Broadcast()
23.OnStaggerDepleted(...)
24.OnTriggerInvulnerable.Broadcast(...)
25.OnTriggerInvulnerable(...)

1. Player presses left click
2. Apply State change to character to attack start
3. Switch boolean to true that the button is being held
4. Player releases left click
5. Apply State change to character to attack release
6. Switch boolean to false that the button is being held
7. Check if a weapon is equipped, if not null broadcast OnAttach to equipped weapon with false. Note: false value is telling the weapon that the button is not being held.
8. Weapon triggers Attack timer, used to prevent players from spamming the attack button.
9. Check if the attack is off cooldown and that the right click isn't pressed, then trigger basic attack.
10. , 11. & 12. The baseball bat checks if the button is released to determine the type of attack. It will then play and multicast the melee animation.

13. The weapon will trigger an animation for light melee attack which at a certain point (frame 10) will trigger 14.1.NotifyBegin from MeleeCollisionCheckNotifyState.
14. In turn will initiate the CheckMeleeCollision function in WeaponMelee. It will keep checking until 14.2.NotifyEnd from MeleeCollisionCheckNotifyState is called.
15. & 16.Anything with a IDamageInterface hit will trigger the apply damage function from the weapon.
17. Weapon will then call the TakeDamage() function from the IDamageInterface.
18. If the target is not dead or invulnerable or blocking, Lower the health, stagger and broadcast the damageResponse to the inherited Object.
19. Character receives the DamageResponse broadcast.
20. If the Object no longer has health or the damage response was INSTANT_KILL, set the health to 0, set bIsDead to true and broadcast OnDeath().
21. Character receives the OnDeath broadcast.
22. If the previous condition is false then we check if the stagger is 0 or less, if true broadcast OnStaggerDepleted.
23. Character receives the OnStaggerDepleted broadcast.
24. If the Object is still alive, broadcast OnTriggerInvulnerable.
25. Character receives OnTriggerInvulnerable broadcast.

Light Ranged Attack


Ranged Light Attack With Revolver Input Action
Player
APlayerController
AQuetzalMultiplayerCharacter
IDamageInterface
AWeapon
ARevolver
AProjectileRevolver
1.Left click Press
2.ECharacterState::ATTACKS
3.lightAttackHoldSwitch = true
4.Left click realese
5.ECharacterState::ATTACKR
6.lightAttackHoldSwitch = false
IF equipedWeapon != nullptr
7.equipedWeapon->OnAttack.Broadcast(false)
8.TriggerAttackTimer(...)
IF RemainingattackCooldown<=0.f && RemainingspecialAttackCooldown<=0 && !bSpecialAttackButtonHeld
9.TriggerBasicAttack(...)
IF attachedActor!=nullptr && currentAmmo > 0 && !bButtonHeld
10.Server_BasicAttack(...)
11.MC_BasicAttack(...);
12.PlayRangeAnimation
13.spawnedProjectile
IF OtherActor == IDamageInterface
15.OnProjectileHit(...)
16.ApplyDamage(...)
14.1.Destroy()
17.TakeDamage(...)
IF maxNumberOfBounces<=0
14.2.Destroy()
IF !bIsDead && (!bIsInvulnerable || damageInfo.ShouldDamageInvulnerable)
IF !bIsBlocking && !damageInfo.canBeBlocked
18.
currentHealth -= damageInfo.amountOfDamage
currentStagger -= damageInfo.amountOfStaggerDamage
OnDamageResponse.Broadcast(...)
19.OnDamageResponse(...)
Alternative
IF currentHealth <= 0 || damageInfo.damageResponse==EDamageResponse::INSTANT_KILL
20.
currentHealth = 0;
bIsDead = true;
OnDeath.Broadcast(...);
21.OnDeath(...)
ELSE IF currentStagger<=0
22.OnStaggerDepleted.Broadcast()
23.OnStaggerDepleted(...)
24.OnTriggerInvulnerable.Broadcast(...)
25.OnTriggerInvulnerable(...)

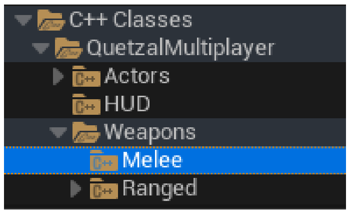
1. Player presses left click
2. Apply State change to character to attack start
3. Switch boolean to true that the button is being held
4. Player releases left click
5. Apply State change to character to attack release
6. Switch boolean to false that the button is being held
7. Check if a weapon is equipped, if not null broadcast OnAttach to equipped weapon with false. Note: false value is telling the weapon that the button is not being held.
8. Weapon triggers Attack timer, used to prevent players from spamming the attack button.
9. Check if the attack is off cooldown and that the right click isn't pressed, then trigger basic attack.
10. , 11. & 12. The Revolver checks if the button is released and the amount of ammo left to determine the type of attack. It will then play and multicast the Ranged animation.

13. The weapon spawns a revolver projectile.
14. 1 & 14.2 if the projectile hits an object with IDamageInterface or reaches the max number of bounces call Destroy().
15. & 16.Anything with a IDamageInterface hit will trigger the apply damage function from the weapon.
17. Weapon will then call the TakeDamage() function from the IDamageInterface.
18. If the target is not dead or invulnerable or blocking, Lower the health, stagger and broadcast the damageResponse to the inherited Object.
19. Character receives the DamageResponse broadcast.
20. If the Object no longer has health or the damage response was INSTANT_KILL, set the health to 0, set bIsDead to true and broadcast OnDeath().
21. Character receives the OnDeath broadcast.
22. If the previous condition is false then we check if the stagger is 0 or less, if true broadcast OnStaggerDepleted.
23. Character receives the OnStaggerDepleted broadcast.
24. If the Object is still alive, broadcast OnTriggerInvulnerable.
25. Character receives OnTriggerInvulnerable broadcast.

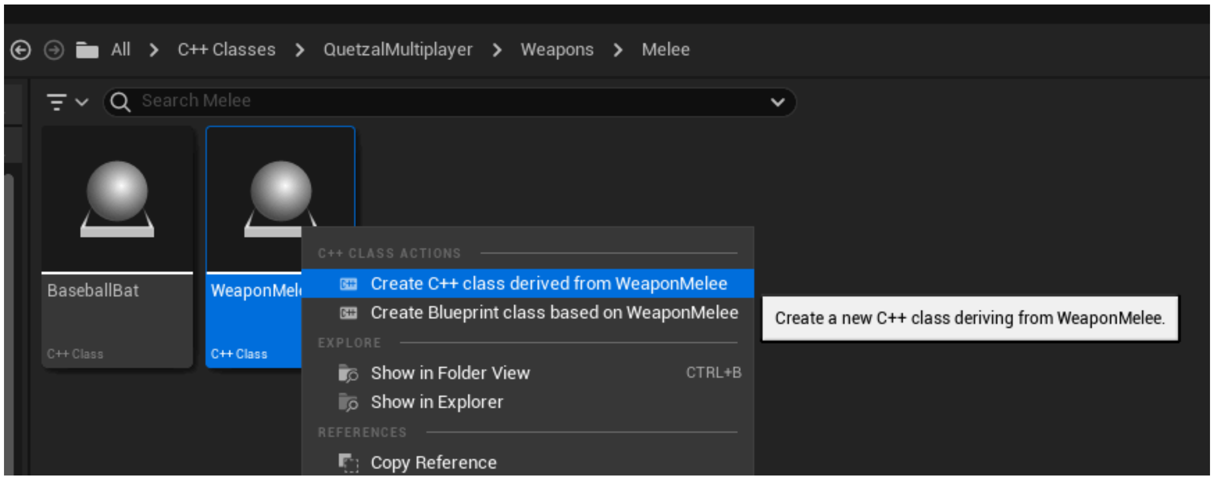
6. Use Case View

6.1 Adding New Weapons

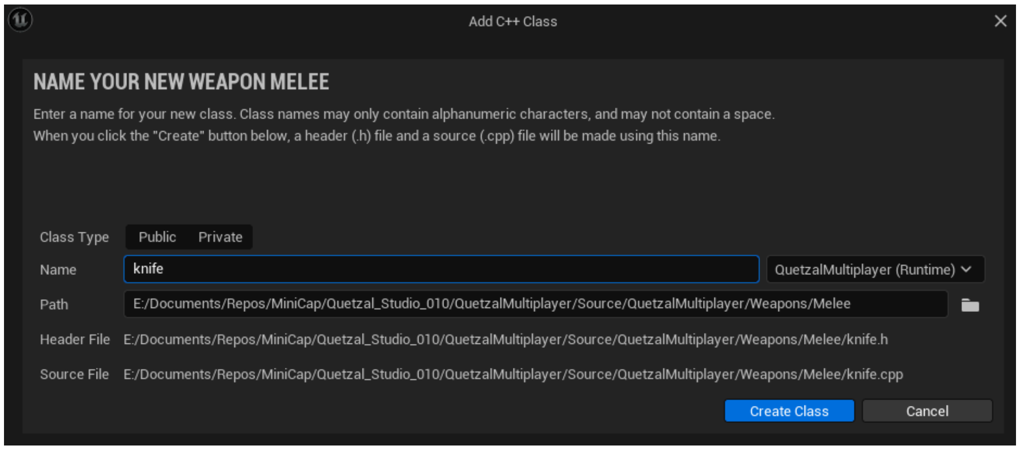
1. In Unreal, navigate to the melee c++ folder



2. Right-Click WeaponMelee class and click on Create C++ derived from WeaponMelee



3. Give the weapon a name

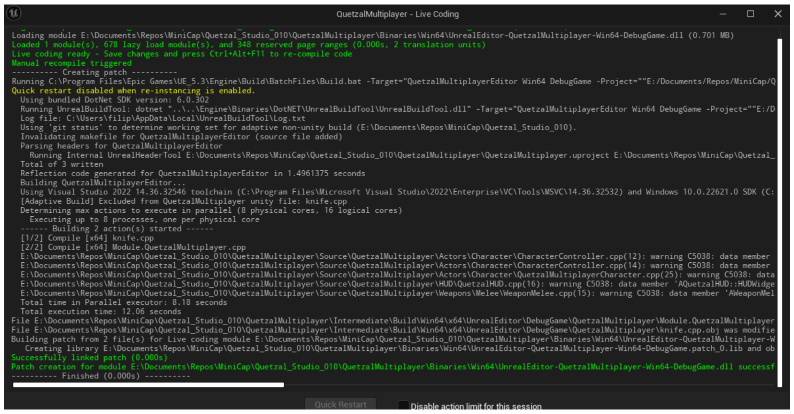

Add C++ Class

NAME YOUR NEW WEAPON MELEE

Enter a name for your new class. Class names may only contain alphanumeric characters, and may not contain a space.
When you click the "Create" button below, a header (.h) file and a source (.cpp) file will be made using this name.

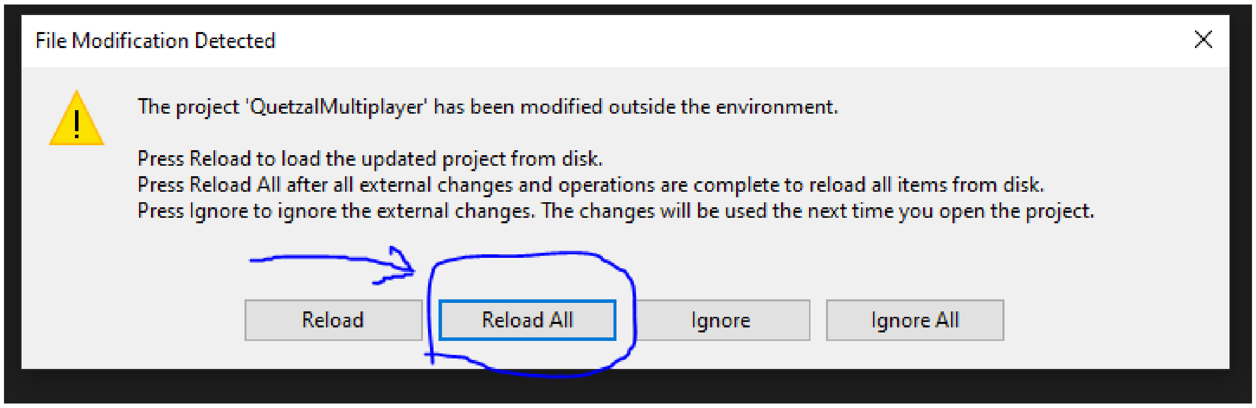
Class Type: Public Private
Name: knife — QuetzalMultiplayer (Runtime)
Path: E:/Documents/Repos/MiniCap/Quetzal_Studio_010/QuetzalMultiplayer/Source/QuetzalMultiplayer/Weapons/Melee
Header File: E:/Documents/Repos/MiniCap/Quetzal_Studio_010/QuetzalMultiplayer/Source/QuetzalMultiplayer/Weapons/Melee/knife.h
Source File: E:/Documents/Repos/MiniCap/Quetzal_Studio_010/QuetzalMultiplayer/Source/QuetzalMultiplayer/Weapons/Melee/knife.cpp

4. Wait for Unreal to finish doing it's thing… Once Successful reload all in Visual Studio.



```
Loading module E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Binaries\Win64\UnrealEditor-QuetzalMultiplayer-Win64-DebugGame.dll (0.701 MB)
Loaded 1 module(s), 678 lazy load module(s), and 348 reserved page ranges (0.000s, 2 translation units)
Live coding ready - Save changes and press Ctrl+Alt+F11 to re-compile code
Manual recompile triggered
---------- Creating patch ----------
Running C:\Program Files\Epic Games\UE_5.3\Engine\Build\BatchFiles\Build.bat -Target="QuetzalMultiplayerEditor Win64 DebugGame -Project=""E:/Documents/Repos/MiniCap/Q
Quick restart disabled when re-instancing is enabled.
  Using bundled DotNet SDK version: 6.0.302
  Running UnrealBuildTool: dotnet "..\..\Engine\Binaries\DotNET\UnrealBuildTool\UnrealBuildTool.dll" -Target="QuetzalMultiplayerEditor Win64 DebugGame -Project=""E:/D
  Log file: C:\Users\filip\AppData\Local\UnrealBuildTool\Log.txt
  Using 'git status' to determine working set for adaptive non-unity build (E:\Documents\Repos\MiniCap\Quetzal_Studio_010).
  Invalidating makefile for QuetzalMultiplayerEditor (source file added)
  Parsing headers for QuetzalMultiplayerEditor
    Running Internal UnrealHeaderTool E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\QuetzalMultiplayer.uproject E:\Documents\Repos\MiniCap\Quetzal_
  Total of 3 written
  Reflection code generated for QuetzalMultiplayerEditor in 1.4961375 seconds
  Building QuetzalMultiplayerEditor...
  Using Visual Studio 2022 14.36.32546 toolchain (C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Tools\MSVC\14.36.32532) and Windows 10.0.22621.0 SDK (C:
  [Adaptive Build] Excluded from QuetzalMultiplayer unity file: knife.cpp
  Determining max actions to execute in parallel (8 physical cores, 16 logical cores)
    Executing up to 8 processes, one per physical core
  ------ Building 2 action(s) started ------
  [1/2] Compile [x64] knife.cpp
  [2/2] Compile [x64] Module.QuetzalMultiplayer.cpp
  E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Source\QuetzalMultiplayer\Actors\Character\CharacterController.cpp(12): warning C5038: data member
  E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Source\QuetzalMultiplayer\Actors\Character\CharacterController.cpp(14): warning C5038: data member
  E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Source\QuetzalMultiplayer\Actors\Character\QuetzalMultiplayerCharacter.cpp(25): warning C5038: data
  E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Source\QuetzalMultiplayer\HUD\QuetzalHUD.cpp(16): warning C5038: data member 'AQuetzalHUD::HUDWidge
  E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Source\QuetzalMultiplayer\Weapons\Melee\WeaponMelee.cpp(15): warning C5038: data member 'AWeaponMel
  Total time in Parallel executor: 8.18 seconds
  Total execution time: 12.06 seconds
File E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Intermediate\Build\Win64\x64\UnrealEditor\DebugGame\QuetzalMultiplayer\Module.QuetzalMultiplayer
File E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Intermediate\Build\Win64\x64\UnrealEditor\DebugGame\QuetzalMultiplayer\knife.cpp.obj was modifie
Building patch from 2 file(s) for Live coding module E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Binaries\Win64\UnrealEditor-QuetzalMultiplayer-W
   Creating library E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Binaries\Win64\UnrealEditor-QuetzalMultiplayer-Win64-DebugGame.patch_0.lib and ob
Successfully linked patch (0.000s)
Patch creation for module E:\Documents\Repos\MiniCap\Quetzal_Studio_010\QuetzalMultiplayer\Binaries\Win64\UnrealEditor-QuetzalMultiplayer-Win64-DebugGame.dll successf
---------- Finished (0.000s) ----------
```

Quick Restart
Disable action limit for this session

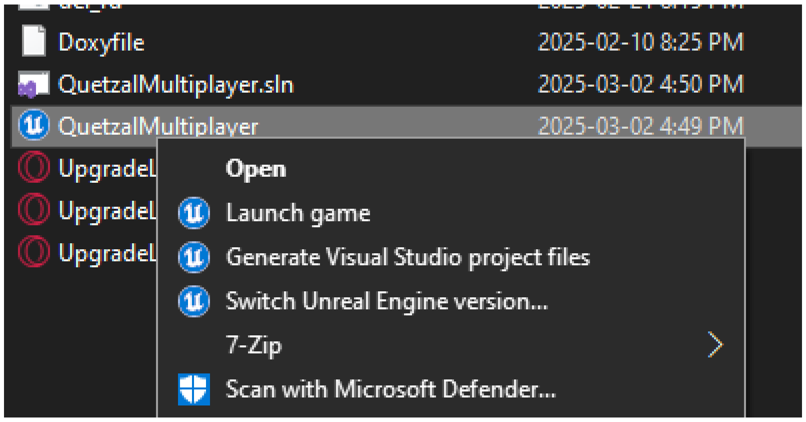

File Modification Detected

The project 'QuetzalMultiplayer' has been modified outside the environment.

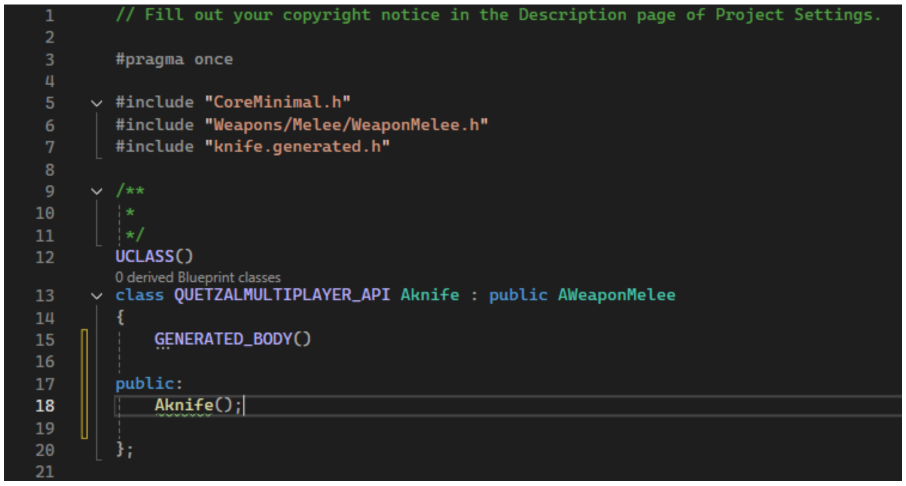
Press Reload to load the updated project from disk.
Press Reload All after all external changes and operations are complete to reload all items from disk.
Press Ignore to ignore the external changes. The changes will be used the next time you open the project.

Reload
Reload All
Ignore
Ignore All

Tip: if you don't see the new cpp and h file, Generate Visual Studio project files again.

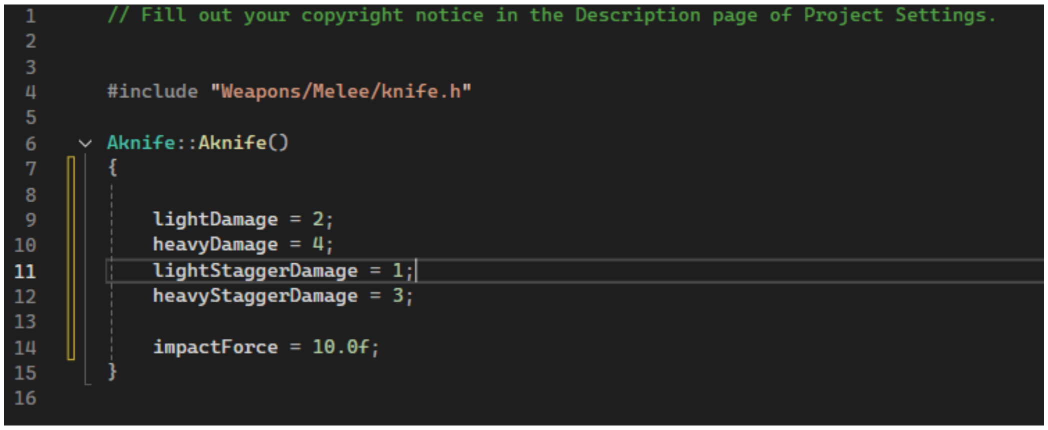

Doxyfile 2025-02-10 8:25 PM
QuetzalMultiplayer.sln 2025-03-02 4:50 PM
QuetzalMultiplayer 2025-03-02 4:49 PM
UpgradeL
UpgradeL
UpgradeL
Open
Launch game
Generate Visual Studio project files
Switch Unreal Engine version...
7-Zip
Scan with Microsoft Defender...

5. In the header and cpp file, create the default constructor.



```cpp
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "Weapons/Melee/WeaponMelee.h"
#include "knife.generated.h"

/**
 *
 */
UCLASS()
0 derived Blueprint classes
class QUETZALMULTIPLAYER_API Aknife : public AWeaponMelee
{
	GENERATED_BODY()

public:
	Aknife();

};
```

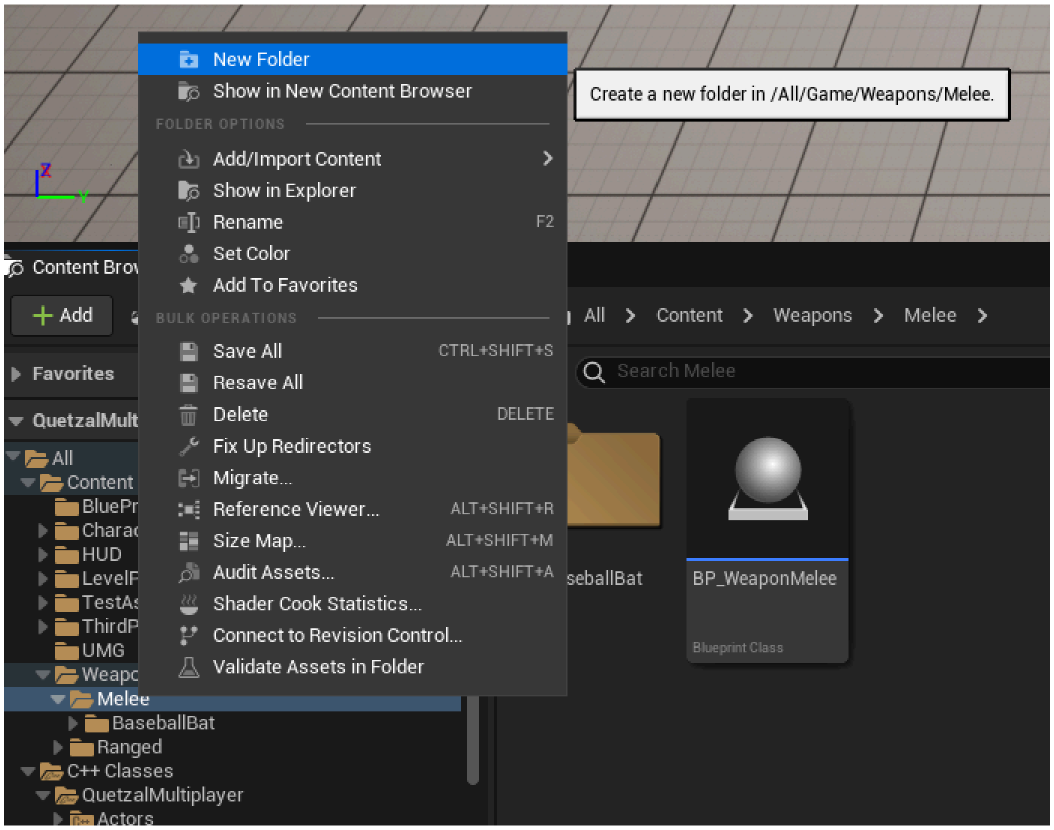
6. Change the damage values for the weapon in the constructor.



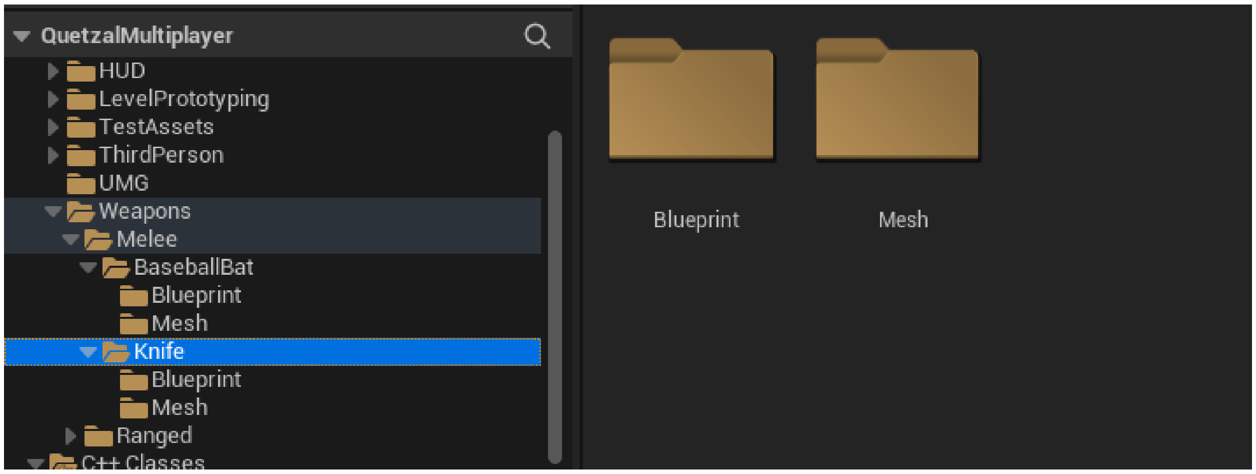
```cpp
// Fill out your copyright notice in the Description page of Project Settings.


#include "Weapons/Melee/knife.h"

Aknife::Aknife()
{

	lightDamage = 2;
	heavyDamage = 4;
	lightStaggerDamage = 1;
	heavyStaggerDamage = 3;

	impactForce = 10.0f;
}
```

7. Add any special feature for the weapon if needed.

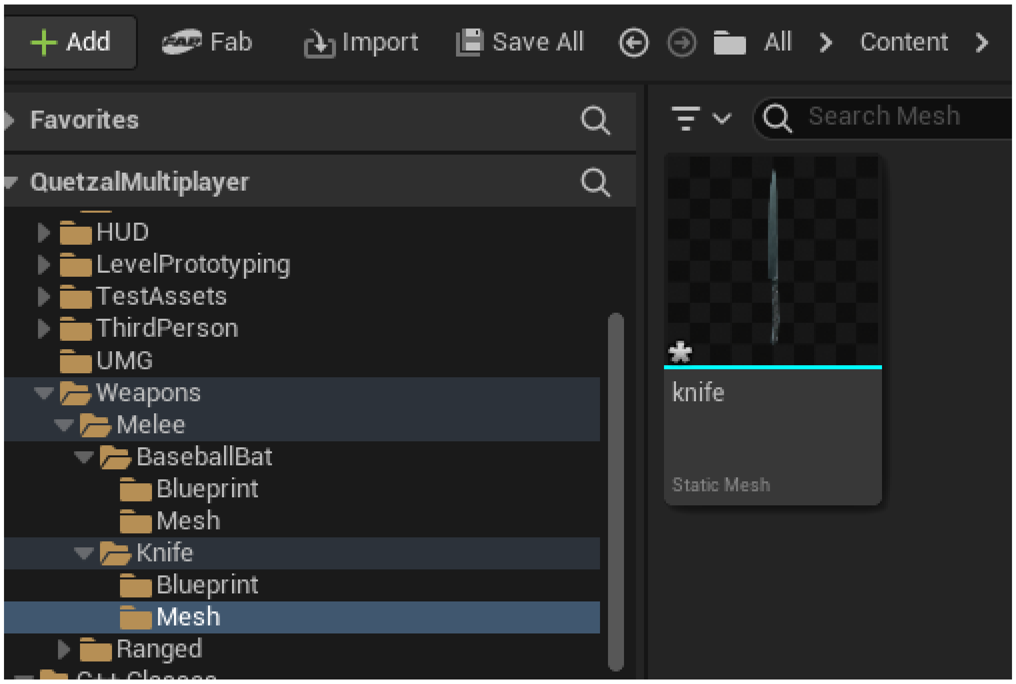
8. Back in Unreal, Create a new folder in the melee folder and give it the weapon name


New Folder
Show in New Content Browser
FOLDER OPTIONS
Add/Import Content
Show in Explorer
Rename F2
Set Color
Add To Favorites
BULK OPERATIONS
Save All CTRL+SHIFT+S
Resave All
Delete DELETE
Fix Up Redirectors
Migrate...
Reference Viewer... ALT+SHIFT+R
Size Map... ALT+SHIFT+M
Audit Assets... ALT+SHIFT+A
Shader Cook Statistics...
Connect to Revision Control...
Validate Assets in Folder
Create a new folder in /All/Game/Weapons/Melee.
All > Content > Weapons > Melee >
Search Melee
BP_WeaponMelee
Blueprint Class

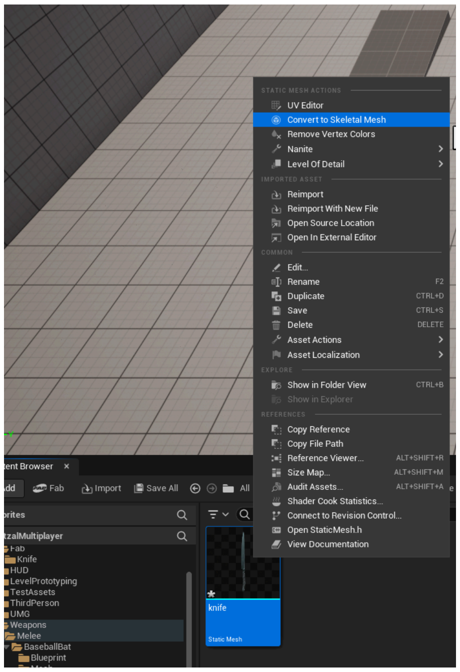
9. In the newly created folder, create a Blueprint and a Mesh folder.



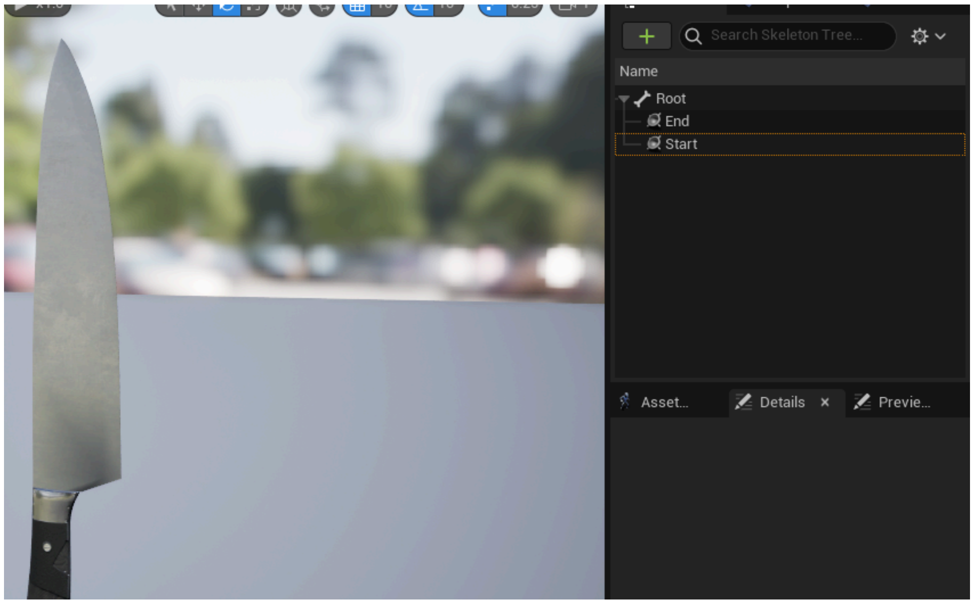
10. Put the mesh of the new weapon in the mesh folder.



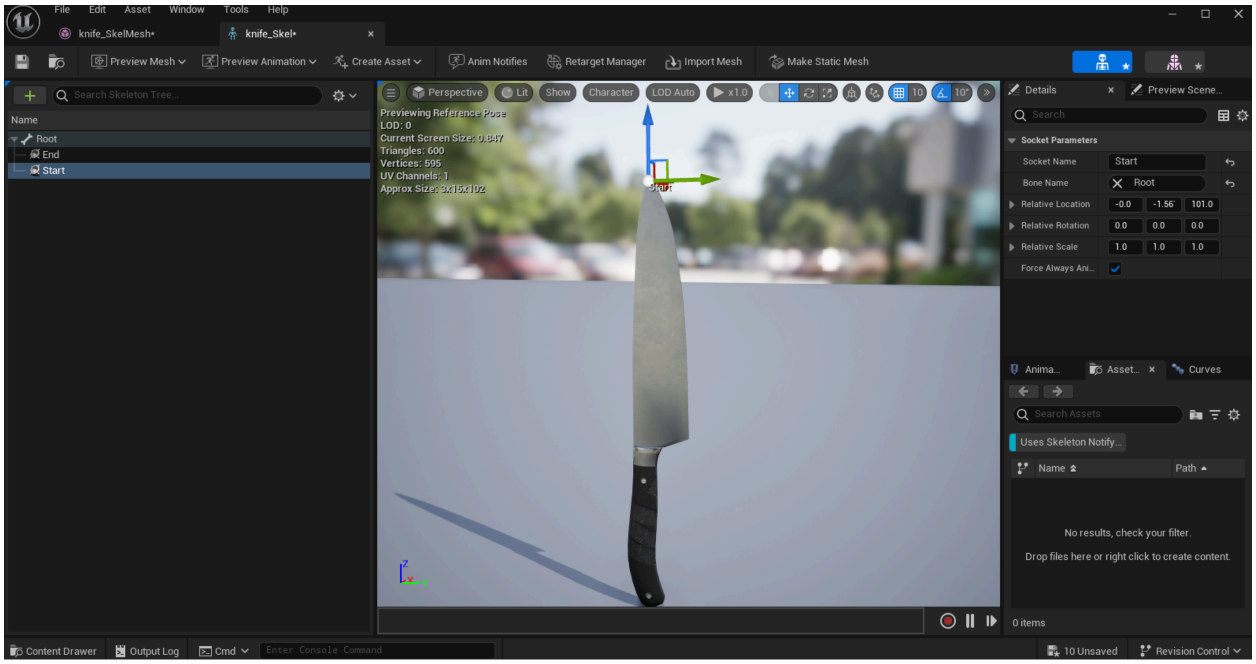
Note: Convert Static Mesh into Skeletal Mesh



11. Double click the skeletal mesh, click on the root socket, click the green plus symbol then add 2 Sockets. One Called End the other Start.



12. Next click on the Skeleton button top right, then click on the Start socket and drag it to the tip of the weapon.



```
Previewing Reference Pose
LOD: 0
Current Screen Size: 0.847
Triangles: 600
Vertices: 595
UV Channels: 1
Approx Size: 3x15x102
```

Socket Name	Start		
Bone Name	Root		
Relative Location	-0.0	-1.56	101.0
Relative Rotation	0.0	0.0	0.0
Relative Scale	1.0	1.0	1.0
Force Always Ani...	✓		

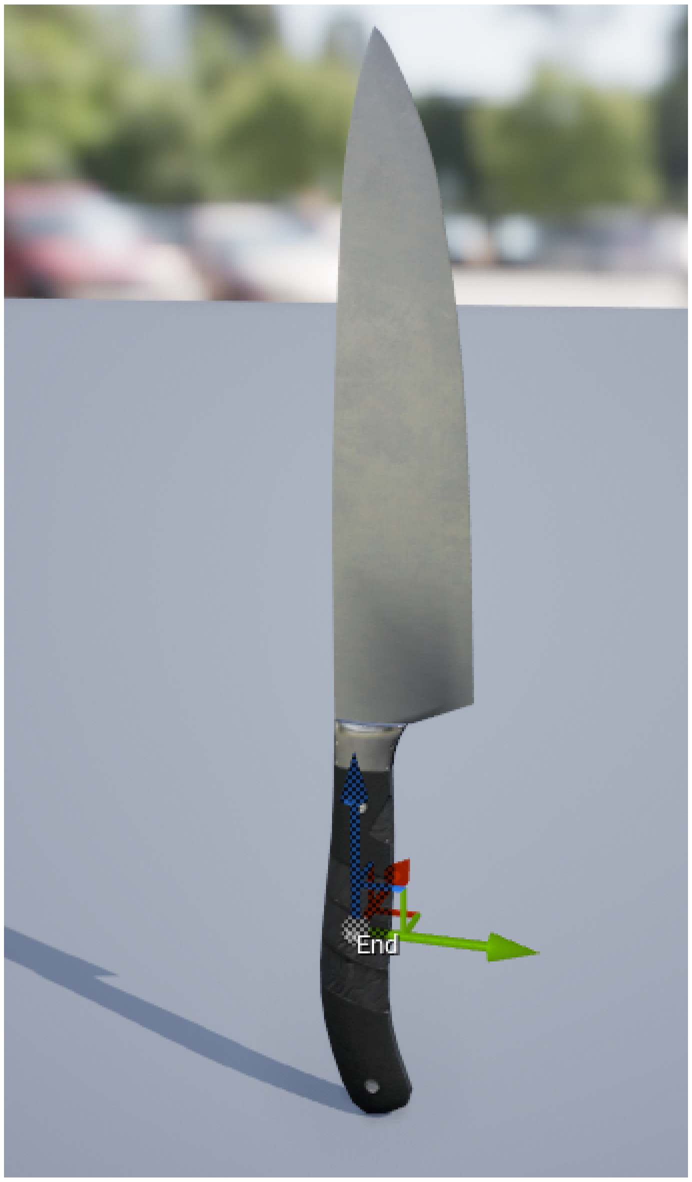
No results, check your filter.

Drop files here or right click to create content.

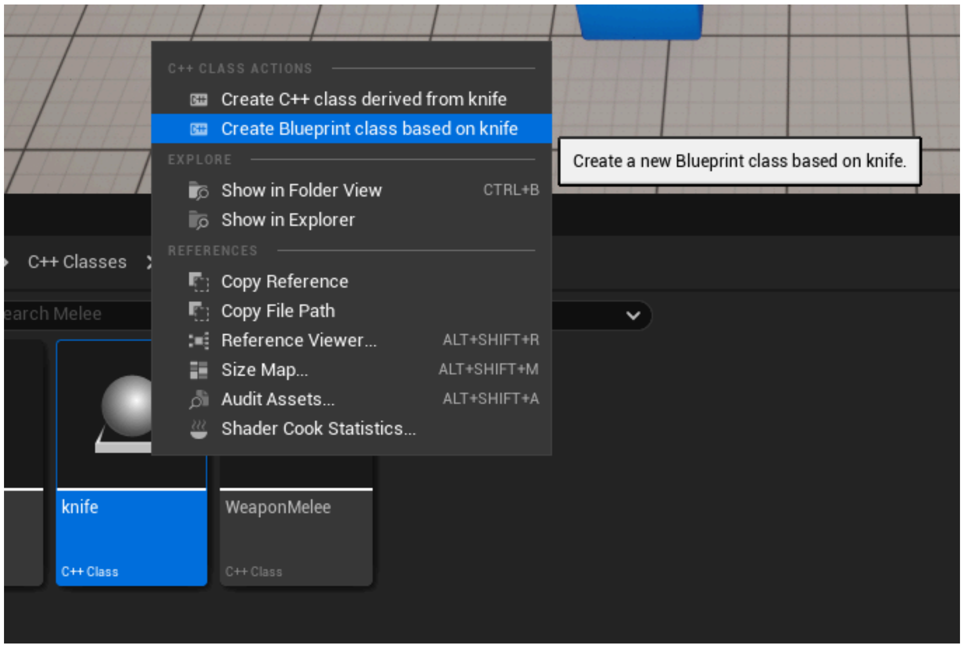
0 items

13. Now drag the End socket to the handle of the weapon. Save and close when done.

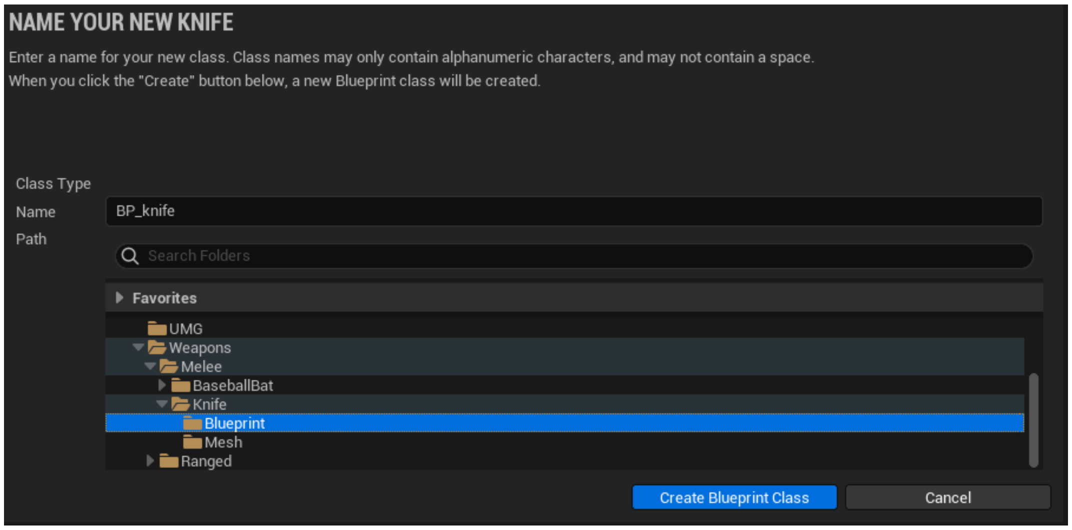
Note: The End socket is where it will attach to the character.



14. Navigate to the cpp file of the new weapon, create a Blueprint of the new weapon.


C++ CLASS ACTIONS
Create C++ class derived from knife
Create Blueprint class based on knife
Create a new Blueprint class based on knife.
EXPLORE
Show in Folder View CTRL+B
Show in Explorer
REFERENCES
Copy Reference
Copy File Path
Reference Viewer... ALT+SHIFT+R
Size Map... ALT+SHIFT+M
Audit Assets... ALT+SHIFT+A
Shader Cook Statistics...
C++ Classes
earch Melee
knife
C++ Class
WeaponMelee
C++ Class

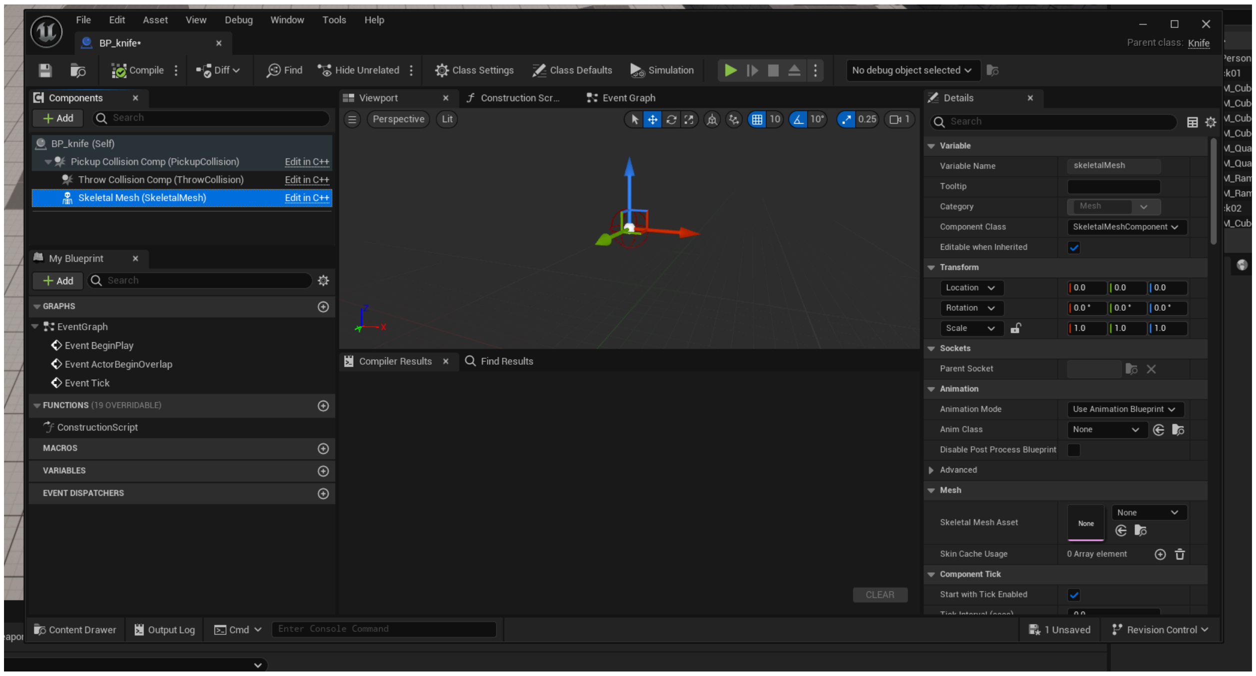
15. Give the blueprint a name, BP_WeaponName. And put it in the newly created blueprint folder.


NAME YOUR NEW KNIFE

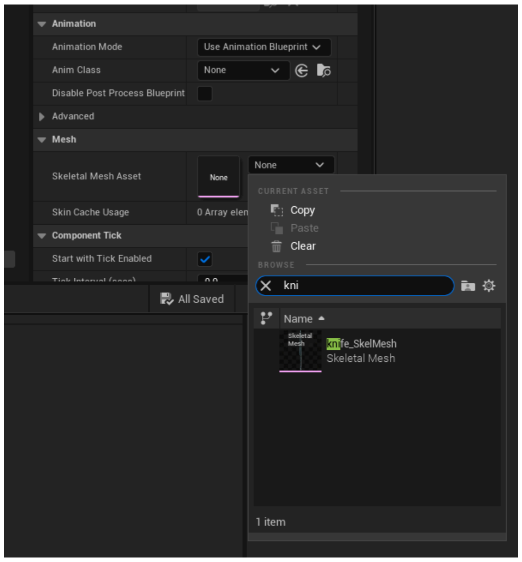
Enter a name for your new class. Class names may only contain alphanumeric characters, and may not contain a space.
When you click the "Create" button below, a new Blueprint class will be created.

Class Type
Name BP_knife
Path

16. A blueprint window will popup, click on SkeletalMesh in the left window.



17. Under Mesh- Skeletal mesh on the right side, search for the new skeletal mesh weapon created earlier. Once Selected, Save and close the window.



18. Congrats! You've created a new weapon for the game. Drag and drop your blueprint of the new weapon into the map and test it out.