Weapon Refactor Cheat sheet

**In A\*yourproject\*Character.h**

**Copy the following into your weaponBase.h**

/\*\* Location on gun mesh where projectiles should spawn. \*/

UPROPERTY(VisibleDefaultsOnly, Category = Mesh)

class USceneComponent\* FP\_MuzzleLocation;

…

/\*\* Gun muzzle's offset from the characters location \*/

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Gameplay)

FVector GunOffset;

/\*\* Projectile class to spawn \*/

UPROPERTY(EditDefaultsOnly, Category=Projectile)

TSubclassOf<class AWeaponRefactorProjectile> ProjectileClass;

/\*\* Sound to play each time we fire \*/

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Gameplay)

class USoundBase\* FireSound;

**Change** AWeaponRefactorProjectile **to your projects projectile class, also include it in the .cpp**

**Comment/delete the following from character.h**

/\*\* Gun mesh: 1st person view (seen only by self) \*/

UPROPERTY(VisibleDefaultsOnly, Category = Mesh)

class USkeletalMeshComponent\* FP\_Gun;

/\*\* Location on gun mesh where projectiles should spawn. \*/

UPROPERTY(VisibleDefaultsOnly, Category = Mesh)

class USceneComponent\* FP\_MuzzleLocation;

/\*\* Gun mesh: VR view (attached to the VR controller directly, no arm, just the actual gun) \*/

UPROPERTY(VisibleDefaultsOnly, Category = Mesh)

class USkeletalMeshComponent\* VR\_Gun;

/\*\* Location on VR gun mesh where projectiles should spawn. \*/

UPROPERTY(VisibleDefaultsOnly, Category = Mesh)

class USceneComponent\* VR\_MuzzleLocation;

…

/\*\* Gun muzzle's offset from the characters location \*/

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Gameplay)

FVector GunOffset;

/\*\* Projectile class to spawn \*/

UPROPERTY(EditDefaultsOnly, Category=Projectile)

TSubclassOf<class AWeaponRefactorProjectile> ProjectileClass;

/\*\* Sound to play each time we fire \*/

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Gameplay)

class USoundBase\* FireSound;

**In A\*yourproject\*Character.cpp**

**Constuctor:**

**Comment out/delete the following**

// Create a gun mesh component

FP\_Gun = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("FP\_Gun"));

FP\_Gun->SetOnlyOwnerSee(true); // only the owning player will see this mesh

FP\_Gun->bCastDynamicShadow = false;

FP\_Gun->CastShadow = false;

// FP\_Gun->SetupAttachment(Mesh1P, TEXT("GripPoint"));

FP\_Gun->SetupAttachment(RootComponent);

VR\_Gun = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("VR\_Gun"));

VR\_Gun->SetOnlyOwnerSee(true); // only the owning player will see this mesh

VR\_Gun->bCastDynamicShadow = false;

VR\_Gun->CastShadow = false;

VR\_Gun->SetupAttachment(R\_MotionController);

VR\_Gun->SetRelativeRotation(FRotator(0.0f, -90.0f, 0.0f));

VR\_MuzzleLocation = CreateDefaultSubobject<USceneComponent>(TEXT("VR\_MuzzleLocation"));

VR\_MuzzleLocation->SetupAttachment(VR\_Gun);

VR\_MuzzleLocation->SetRelativeLocation(FVector(0.000004, 53.999992, 10.000000));

VR\_MuzzleLocation->SetRelativeRotation(FRotator(0.0f, 90.0f, 0.0f)); // Counteract the rotation of the VR gun model.

**Copy the following into the constructor of WeaponBase**

FP\_MuzzleLocation = CreateDefaultSubobject<USceneComponent>(TEXT("MuzzleLocation"));

FP\_MuzzleLocation->SetupAttachment(FP\_Gun);

FP\_MuzzleLocation->SetRelativeLocation(FVector(0.2f, 48.4f, -10.6f));

GunOffset = FVector(100.0f, 0.0f, 10.0f);

**Change FP\_Gun to MeshComp;**

**BeginPlay():**

**Comment out/delete the following**

//Attach gun mesh component to Skeleton, doing it here because the skeleton is not yet created in the constructor

FP\_Gun->AttachToComponent(Mesh1P, FAttachmentTransformRules(EAttachmentRule::SnapToTarget, true), TEXT("GripPoint"));

// Show or hide the two versions of the gun based on whether or not we're using motion controllers.

if (bUsingMotionControllers)

{

VR\_Gun->SetHiddenInGame(false, true);

Mesh1P->SetHiddenInGame(true, true);

}

else

{

VR\_Gun->SetHiddenInGame(true, true);

Mesh1P->SetHiddenInGame(false, true);

}

**Create a public function in WeaponBase for firing of that weapon, call it FireWeapon(), it should return a bool, The bool represents is if it successfully fired.**

**In OnFire() in character**

**comment/delete the following:**

// try and fire a projectile

if (ProjectileClass != NULL)

{

UWorld\* const World = GetWorld();

if (World != NULL)

{

if (bUsingMotionControllers)

{

const FRotator SpawnRotation = VR\_MuzzleLocation->GetComponentRotation();

const FVector SpawnLocation = VR\_MuzzleLocation->GetComponentLocation();

World->SpawnActor<AWeaponRefactorProjectile>(ProjectileClass, SpawnLocation, SpawnRotation);

}

else

{

const FRotator SpawnRotation = GetControlRotation();

// MuzzleOffset is in camera space, so transform it to world space before offsetting from the character location to find the final muzzle position

const FVector SpawnLocation = ((FP\_MuzzleLocation != nullptr) ? FP\_MuzzleLocation->GetComponentLocation() : GetActorLocation()) + SpawnRotation.RotateVector(GunOffset);

//Set Spawn Collision Handling Override

FActorSpawnParameters ActorSpawnParams;

ActorSpawnParams.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButDontSpawnIfColliding;

// spawn the projectile at the muzzle

World->SpawnActor<AWeaponRefactorProjectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);

}

}

}

// try and play the sound if specified

if (FireSound != NULL)

{

UGameplayStatics::PlaySoundAtLocation(this, FireSound, GetActorLocation());

}

// try and play a firing animation if specified

if (FireAnimation != NULL)

{

// Get the animation object for the arms mesh

UAnimInstance\* AnimInstance = Mesh1P->GetAnimInstance();

if (AnimInstance != NULL)

{

AnimInstance->Montage\_Play(FireAnimation, 1.f);

}

}

**Replace it with:**

if (Weapon->FireWeapon())

{

WeaponFired(); //your blueprint implementable event...

// try and play a firing animation if specified

if (FireAnimation != NULL)

{

// Get the animation object for the arms mesh

UAnimInstance\* AnimInstance = Mesh1P->GetAnimInstance();

if (AnimInstance != NULL)

{

AnimInstance->Montage\_Play(FireAnimation, 1.f);

}

}

}

**In FireWeapon in weaponBase.cpp**

// try and fire a projectile

if (ProjectileClass != NULL)

{

UWorld\* const World = GetWorld();

if (World != NULL)

{

const FRotator SpawnRotation = UGameplayStatics::GetPlayerController(GetWorld(), 0)->GetControlRotation();

// MuzzleOffset is in camera space, so transform it to world space before offsetting from the character location to find the final muzzle position

const FVector SpawnLocation = ((FP\_MuzzleLocation != nullptr) ? FP\_MuzzleLocation->GetComponentLocation() : GetActorLocation()) + SpawnRotation.RotateVector(GunOffset);

//Set Spawn Collision Handling Override

FActorSpawnParameters ActorSpawnParams;

ActorSpawnParams.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButDontSpawnIfColliding;

// spawn the projectile at the muzzle

World->SpawnActor<AWeaponRefactorProjectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);

return true;

}

}

return false;