**Tanks Vs Zombies**

Create a basic C++ project name it Tank

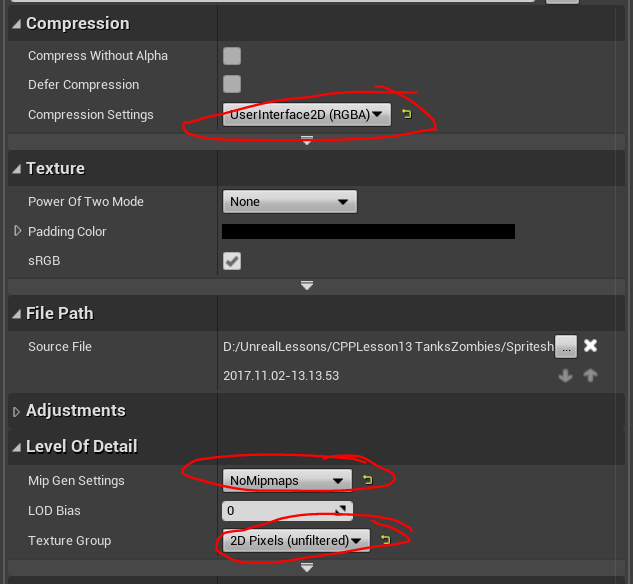
Create a C++ GameMode and name it TankGameMode

Open TankBuild.cs file and add "Paper2D" at the end of PublicDependencyModule as follows

PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore", "Paper2D" });

Close VS and right click on uproject file and generate VisualStudioProjectFiles

Import sprite assets Open sprite and change the following.



Rightclick on sprite texture and select SpriteActions/ExtractSprite

Select Grid or Auto and click extract

Name sprites accordingly

Create a Pawn C++ class

Add following code to header file

private:

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank", meta = (AllowPrivateAccess = "true"))

UArrowComponent\* TankDirection;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank", meta = (AllowPrivateAccess = "true"))

UPaperSpriteComponent\* TankSprite;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Turret", meta = (AllowPrivateAccess = "true"))

UChildActorComponent\* ChildTurret;

Need forward decleration

class UArrowComponent;

class UPaperSpriteComponent;

Add the following code to constructor

if (!RootComponent) {

RootComponent = CreateDefaultSubobject<USceneComponent>(TEXT("TankBase"));

}

TankDirection = CreateDefaultSubobject<UArrowComponent>(TEXT("TankDirection"));

TankDirection->AttachTo(RootComponent);

TankSprite = CreateDefaultSubobject<UPaperSpriteComponent>(TEXT("TankSprite"));

TankSprite->AttachTo(TankDirection);

ChildTurret = CreateDefaultSubobject<UChildActorComponent>(TEXT("Turret"));

ChildTurret->AttachTo(TankDirection);

Needs some includes

#include "Tank.h"

#include "Tanks.h"

#include "Components/ArrowComponent.h"

#include "PaperSpriteComponent.h"

Create an Actor class called Turret

Add following to header file

private:

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Turret", meta = (AllowPrivateAccess = "true"))

UArrowComponent\* TurretDirection;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Turret", meta = (AllowPrivateAccess = "true"))

UPaperSpriteComponent\* TurretSprite;

Add following to C++ file constructor

TurretDirection = CreateDefaultSubobject<UArrowComponent>(TEXT("TurretDirection"));

RootComponent = TurretDirection;

TurretSprite = CreateDefaultSubobject<UPaperSpriteComponent>(TEXT("TurretSprite"));

TurretSprite->AttachTo(TurretDirection);

Create BP version of the tank

Select the tank sprite from the sprite

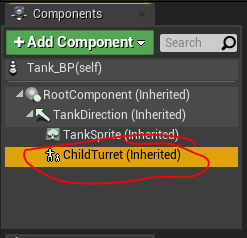
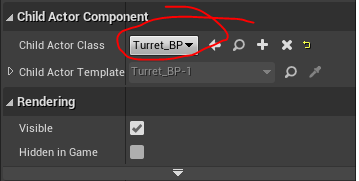
Make rotations to match the level

Create a BP version of the Turret

Select the sprite

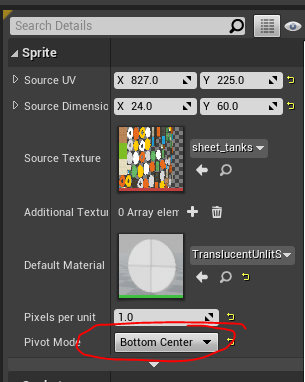
Adjust rotation

Select the Turret\_BP in the child actor in Tank\_BP

To change the pivot of the turret

Open the sprite and select the pivot as follows



We need to add camera. Add the following code to Tank.h

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank", meta = (AllowPrivateAccess = "true"))

UCameraComponent\* CameraComponent;

USpringArmComponent\* SpringArm;

Add the following code to Tank.cpp

SpringArm = CreateDefaultSubobject<USpringArmComponent>(TEXT("SpringArm"));

SpringArm->TargetArmLength = 500.0f;

SpringArm->bEnableCameraLag = true;

SpringArm->bEnableCameraRotationLag = false;

SpringArm->CameraLagSpeed = 2.0f;

SpringArm->bDoCollisionTest = false;

SpringArm->bUsePawnControlRotation = false;

SpringArm->AttachTo(RootComponent);

SpringArm->SetWorldRotation(FRotator(-90.0f, 0.0f, 0.0f));

CameraComponent = CreateDefaultSubobject< UCameraComponent>(TEXT("CameraComponent"));

CameraComponent->bUsePawnControlRotation = false;

CameraComponent->AttachTo(SpringArm,USpringArmComponent::SocketName);

CameraComponent->ProjectionMode = ECameraProjectionMode::Orthographic;

CameraComponent->OrthoWidth = 1024.0f;

CameraComponent->AspectRatio = 3.0f / 4.0f;

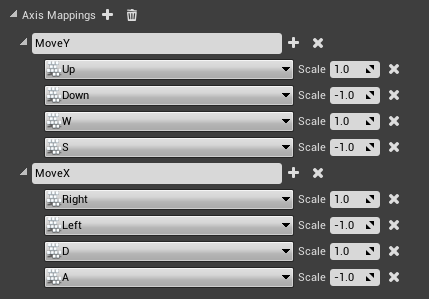
Create a BP version of the TankGameMode

Select TankBP as the default pawn.

Select the TankGameMode\_BP as the game mode

Tank should be the default pawn

Add the following input in project settings



Add following to Tank.h

private:

void MoveX(float AxisValue);

void MoveY(float AxisValue);

Add the following to SetupPlayerInputComponent method in Tank.cpp

PlayerInputComponent->BindAxis("MoveX", this, &ATank::MoveX);

PlayerInputComponent->BindAxis("MoveY", this, &ATank::MoveY);

Add the following Struct to Tank.h file

USTRUCT()

struct FTankInput

{

GENERATED\_BODY()//To recognise by Unreal

public:

UPROPERTY(Visiblenywhere, BlueprintReadOnly, Category = "Tank Input")

FVector2D MovementInput;

void Sanitize();

void MoveX(float AxisValue);

void MoveY(float AxisValue);

private:

FVector2D RawMovementInput;

};

Create a variable of the struct at the end of Tank.h

protected:

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank Input")

FTankInput TankInput;

Add the implementation of Move methods as follows

void ATank::MoveX(float AxisValue)

{

TankInput.MoveX(AxisValue);

}

void ATank::MoveY(float AxisValue)

{

TankInput.MoveY(AxisValue);

}

Create implementation of functions in struct in the Tank.cpp file as follows

void FTankInput::Sanitize()

{

MovementInput = RawMovementInput.ClampAxes(-1.0f, 1.0f);

MovementInput = MovementInput.GetSafeNormal();

RawMovementInput.Set(0.0f, 0.0f);

}

void FTankInput::MoveX(float AxisValue)

{

RawMovementInput.X += AxisValue;

}

void FTankInput::MoveY(float AxisValue)

{

RawMovementInput.Y += AxisValue;

}

Add the following in Tick to test

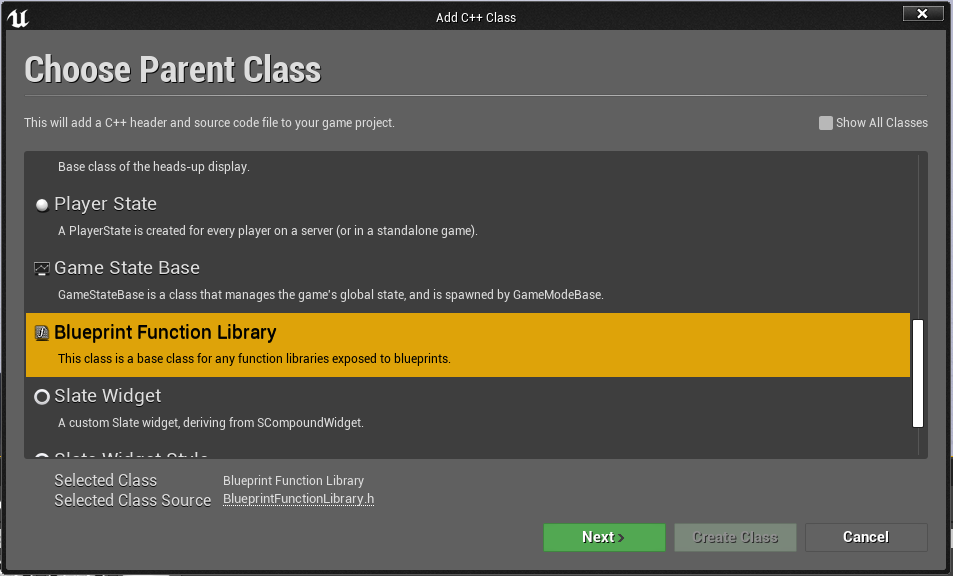
TankInput.Sanitize();

UE\_LOG(LogTemp, Warning, TEXT("Movement : (%f, %f)"), TankInput.MovementInput.X, TankInput.MovementInput.Y);

Output log should give the directions of the key press.

Now we need the tank to move

Create a Blueprint Function Library name it “TankStatics”



Add following to Tanks.h

#include "TankStatics.h"

Add include in cpp file

#include "Tanks.h"

Add following functions to Header file

public:

UFUNCTION(BlueprintCallable, Category = "Math")

static float FindDeltaAngleDegrees(float A1, float A2);

UFUNCTION(BlueprintPure, Category = "Math|Rotator", meta = (Keywords = "rotation rotate"))

static bool FindLookAtAngle2D(const FVector2D& Start, const FVector2D& Target, float& Angle);

Create implementation as follows

float UTankStatics::FindDeltaAngleDegrees(float A1, float A2)

{

float Delta = A2 - A1;

if (Delta > 180.0f) {

Delta = Delta - 360.0f;

}

else if (Delta < -180.0f)

{

Delta = Delta + 360.0f;

}

return Delta;

}

bool UTankStatics::FindLookAtAngle2D(const FVector2D & Start, const FVector2D & Target, float & Angle)

{

FVector2D Normal = (Target - Start).GetSafeNormal();

if (!Normal.IsNearlyZero())

{

Angle = FMath::RadiansToDegrees(FMath::Atan2(Normal.Y, Normal.X));

return true;

}

return false;

}

Add the following code to Tick of the Tank.cpp for movement and Rotation

{ // Variables declare will exist only within this score.

FVector DesiredMovementDirection = FVector(TankInput.MovementInput.X, TankInput.MovementInput.Y, 0.0f);

if (!DesiredMovementDirection.IsNearlyZero())

{

//Rotate the Tank

FRotator MovementAngle = DesiredMovementDirection.Rotation();

float DeltaYaw = UTankStatics::FindDeltaAngleDegrees(TankDirection->GetComponentRotation().Yaw,MovementAngle.Yaw);

bool bReverse = false;

if (DeltaYaw != 0.0f)

{

float AdjustedDeltaYaw = DeltaYaw;

if (AdjustedDeltaYaw < -90.0f)

{

AdjustedDeltaYaw += 180.0f;

bReverse = true;

}

else if (AdjustedDeltaYaw > 90.0f)

{

AdjustedDeltaYaw -= 180.0f;

bReverse = true;

}

float MaxYawThisFrame = YawSpeed \* DeltaTime;

if (MaxYawThisFrame >= FMath::Abs(AdjustedDeltaYaw))

{

if (bReverse)

{

// move backward

FRotator FacingAngle = MovementAngle;

FacingAngle.Yaw = MovementAngle.Yaw + 180.0f;

TankDirection->SetWorldRotation(FacingAngle);

}

else

{

TankDirection->SetWorldRotation(MovementAngle);

}

}

else

{

// can reach out desired angle this frame , rotate part way.

TankDirection->AddLocalRotation(FRotator(0.0f, FMath::Sign(AdjustedDeltaYaw)\*MaxYawThisFrame, 0.0f));

}

// Move the tank

{

FVector MovementDirection = TankDirection->GetForwardVector()\* (bReverse ? -1.0f : 1.0f);

FVector Pos = GetActorLocation();

Pos.X += MovementDirection.X \* MoveSpeed \* DeltaTime;

Pos.Y += MovementDirection.Y \* MoveSpeed \* DeltaTime;

SetActorLocation(Pos);

}

}

Some variables need to be declared in Header file

//maximum turn rate of the tank

UPROPERTY(EditAnywhere, BlueprintReadOnly, Category = "Tank", meta = (ClampMin= "0.0"))

float YawSpeed;

// movement speed

UPROPERTY(EditAnywhere, BlueprintReadOnly, Category = "Tank", meta = (ClampMin = "0.0"))

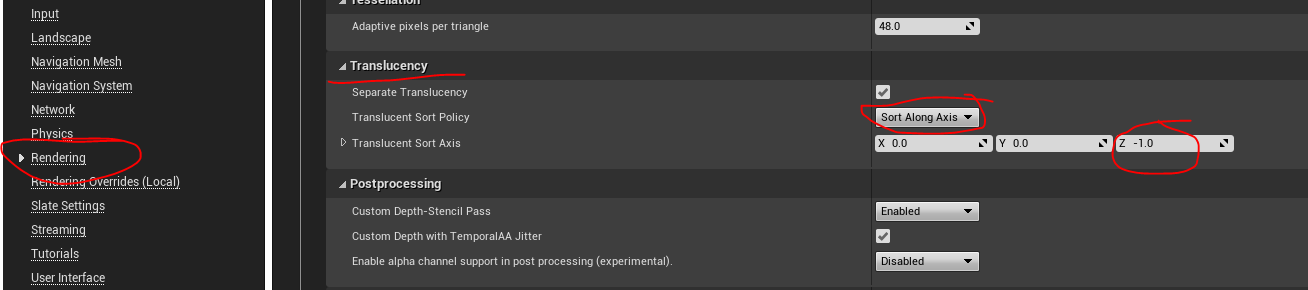
float MoveSpeed;

In the Tank\_BP change Yaw speed and MoveSpeed to 180 and 100 respectively

When playing tank should move

Might need to adjust orientation of the sprite and arrow.

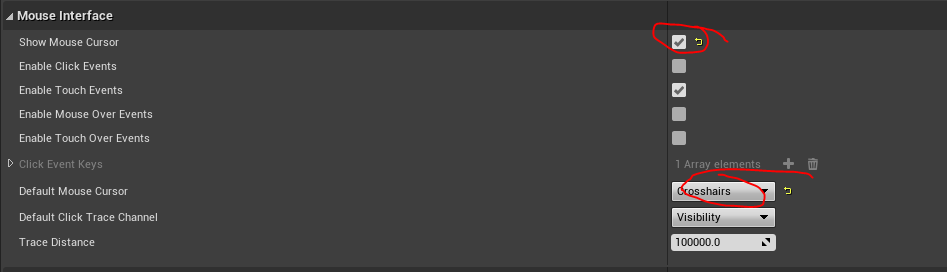
For Turret disappearing change project settings as follows



**Tank Turret Scripting**

Create a Player Controller and add that into the game mode

Make the following changes in Player Controller



Make sure player controller is set in game mode and game mode is set in world settings

Open TankStatics.h and add following function

UFUNCTION(BlueprintPure, Category = "Math|Rotator", meta = (Keywords = "rotation rotate"))

static bool FindLookAtAngle2D(const FVector2D& Start, const FVector2D& Target, float& Angle);

Add the following implememtation in cpp file

bool UTankStatics::FindLookAtAngle2D(const FVector2D & Start, const FVector2D & Target, float & Angle)

{

FVector2D Normal = (Target - Start).GetSafeNormal();

if (!Normal.IsNearlyZero())

{

Angle = FMath::RadiansToDegrees(FMath::Atan2(Normal.Y, Normal.X));

return true;

}

return false;

}

Declare a variable in Turret.h as follows

// Max turn rate in degrees per second for the turret

UPROPERTY(EditAnywhere, BlueprintReadOnly, Category = "Turret")

float YawSpeed;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Turret")

ATank\* Tank;

Forward decleration might be required

class ATank;

Add the following code to Beginplay of Turret.cpp

Tank = Cast<ATank>(GetParentComponent()->GetOwner());

check(Tank);

Initialise YawSpeed in constructor as follows

YawSpeed = 180.0f;

Add the following code to Tick of Turret.cpp

check(TurretDirection);

if (Tank != nullptr)

{

if (APlayerController\* PC = Cast<APlayerController>(Tank->GetController()))

{

FVector2D AimLocation;

if (PC->GetMousePosition(AimLocation.X, AimLocation.Y))

{

FVector2D TurretLocation = FVector2D::ZeroVector;

UGameplayStatics::ProjectWorldToScreen(PC, TurretDirection->GetComponentLocation(), TurretLocation);

float DesiredYaw;

if (UTankStatics::FindLookAtAngle2D(TurretLocation, AimLocation, DesiredYaw))

{

FRotator CurrentRotation = TurretDirection->GetComponentRotation();

float DeltaYaw = UTankStatics::FindDeltaAngleDegrees(CurrentRotation.Yaw, DesiredYaw);

float MaxDeltaYawThisFrame = YawSpeed \* DeltaTime;

if (MaxDeltaYawThisFrame >= FMath::Abs(DesiredYaw))

{

//We can get there on this frame, so just set it.

CurrentRotation.Yaw = DesiredYaw;

}

else

{

CurrentRotation.Yaw += FMath::Sign(DesiredYaw)\* MaxDeltaYawThisFrame;

TurretDirection->SetWorldRotation(CurrentRotation);

}

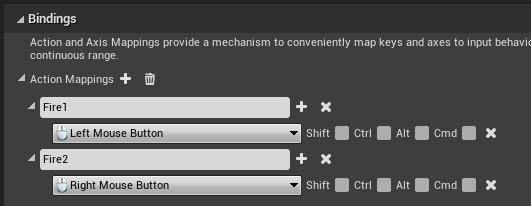
}

}

}

}

Add new action inputs as follows



Add the following code to Tank.h file as public

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank input")

uint32 bFire1 : 1; // declare a variable of type uint32 of size 1

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Tank input")

uint32 bFire2 : 1;

declare 2 methods in public as follows

void Fire1(bool bPressed);

void Fire2(bool bPressed);

Declare 4 methods as private as follows

void Fire1Pressed();

void Fire1Release();

void Fire2Pressed();

void Fire2Release();

Create implementation as follows

void FTankInput::Fire1(bool bPressed)

{

bFire1 = bPressed;

}

void FTankInput::Fire2(bool bPressed)

{

bFire2 = bPressed;

}

void ATank::Fire1Pressed()

{

TankInput.Fire1(true);

}

void ATank::Fire1Release()

{

TankInput.Fire1(false);

}

void ATank::Fire2Pressed()

{

TankInput.Fire2(true);

}

void ATank::Fire2Release()

{

TankInput.Fire2(false);

}

Add the following to SetupPlayerInputComponent method

PlayerInputComponent->BindAction("Fire1", EInputEvent::IE\_Pressed, this, &ATank::Fire1Pressed);

PlayerInputComponent->BindAction("Fire1", EInputEvent::IE\_Released, this, &ATank::Fire1Release);

PlayerInputComponent->BindAction("Fire2", EInputEvent::IE\_Pressed, this, &ATank::Fire2Pressed);

PlayerInputComponent->BindAction("Fire2", EInputEvent::IE\_Released, this, &ATank::Fire2Release);

**Missile**

Create a new Actor class called Missile

Add the following as public in Missile.h

UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Projectile")

float Speed;

UFUNCTION(BlueprintNativeEvent, Category = "Projectile")

void Explode();

void Explode\_Implementation();

int TotalShellsFired;

Add the following in constructor

Speed = 200.0f;

Add the following to begin play

FTimerHandle DummyTimerHandle;

GetWorldTimerManager().SetTimer(DummyTimerHandle, this, &AMissile::Explode, 1.0f);

Add the following to Tick

FVector Loc = GetActorLocation();

Loc += (DeltaTime \* Speed) \* GetTransform().GetUnitAxis(EAxis::X);

SetActorLocation(Loc);

Add the following to implementation

void AMissile::Explode\_Implementation()

{

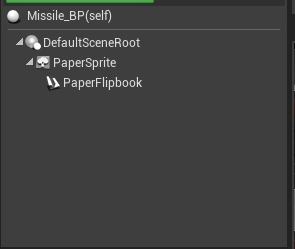
Destroy();

++TotalShellsFired;

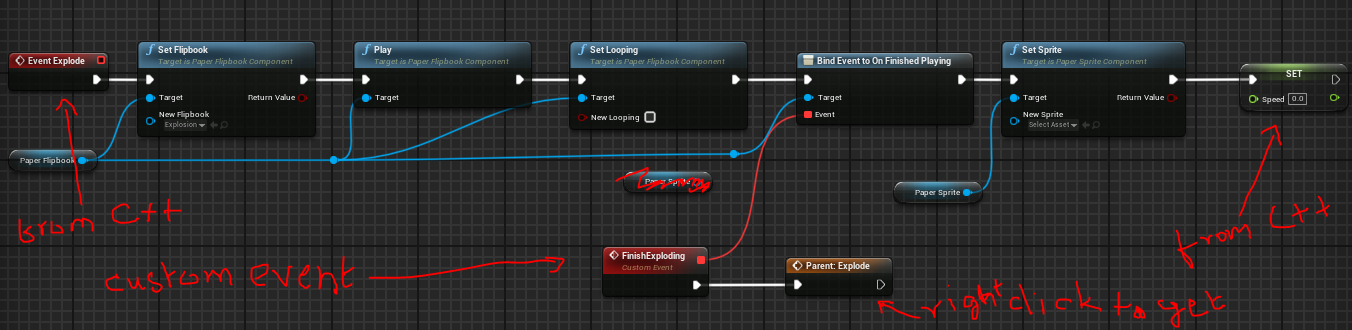
}

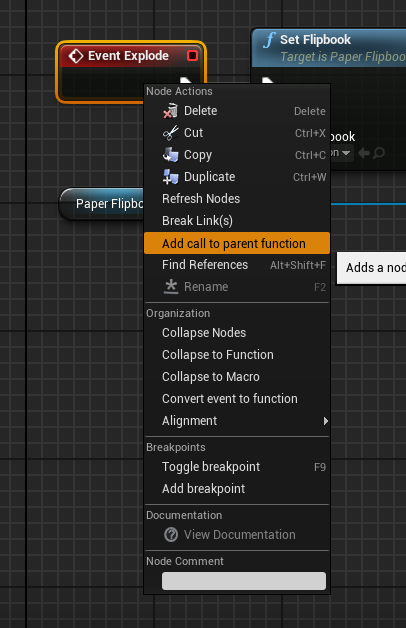
Create a BP version of the Missile

Add sprite and papersprite and flipbook components

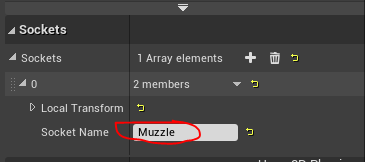


Add following BP





Add a socket to TurretSprite



Add following code to Tank.h as public

UFUNCTION(BlueprintCallable, Category = "Tank")

const FTankInput& GetCurrentInput() { return TankInput; }

Add the following to Turret.h

UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Turret")

TSubclassOf<AActor> Projectile;

Forward declaration required for Missile

Add the following code to Turret.cpp

**Reference**

<https://www.youtube.com/watch?v=NyXq0Hy9xQs&list=PLZlv_N0_O1gaz3ydgU5wt6c_JtJzwXUKW&ab_channel=UnrealEngine>

<https://github.com/aa1000/GASTanksVsZombies>