**Game play logic**

Use the boolean in character to check if character is carrying the objective.

So in the HandleOverlap method add the following.

AFPSObjectiveCharacter\* MyPawn = Cast<AFPSObjectiveCharacter>(OtherActor);

we need to include the character

#include "FPSObjectiveCharacter.h"

Now we need 2 checks

If the variable MyPawn not null pointer and the boolean in it is 'true'

So do the following

if (MyPawn != nullptr && MyPawn->bIsCarryingObjective)

{

}

We need to disable the input if the condition is true.

We will do that in GameMode.

So go to GameMode.h and add this method.

void MissionComplete(APawn\* InstigatorPawn);

and do the definition in cpp file.

void AFPSObjectiveGameMode::MissionComplete(APawn \* InstigatorPawn)

{

if (InstigatorPawn)

{

InstigatorPawn->DisableInput(nullptr);

}

}

Argument passed to the method can be pointer to the player controller or a null pointer.

If nullpointer is passed, then input will be disabled.

We can do some extra logic in BP. For that add another function in game mode header file as follows.

UFUNCTION(BlueprintImplementableEvent, Category = "GameMode")

void OnMissionCompleted(APawn\* InstigatorPawn);

We call this function in MissionComplete method we created earlier after the if condition.

OnMissionCompleted(InstigatorPawn);

This function is exposed to BP. There will be no implementation in cpp.

Now we need to call this function from ExtractionZone.cpp.

To do that add the following code in the if condition of the HandleOverlap function.

AFPSObjectiveGameMode\* GM = Cast<AFPSObjectiveGameMode>(GetWorld()->GetAuthGameMode());

if (GM)

{

GM->MissionComplete(MyPawn);

}

Include needed for GameMode

#include "FPSObjectiveGameMode.h"

Final script in handle overlap function will be like the following.

void AFPSExtraction::HandleOverlap(UPrimitiveComponent \* OverlappedComponent, AActor \* OtherActor, UPrimitiveComponent \* OtherComp,

int32 OtherBodyIndex, bool bFromSweep, const FHitResult & SweepResult)

{

UE\_LOG(LogTemp, Warning, TEXT("Overlapped with extraction zone"));

AFPSObjectiveCharacter\* MyPawn = Cast<AFPSObjectiveCharacter>(OtherActor);

if (MyPawn != nullptr && MyPawn->bIsCarryingObjective)

{

AFPSObjectiveGameMode\* GM = Cast<AFPSObjectiveGameMode>(GetWorld()->GetAuthGameMode());

if (GM)

{

GM->MissionComplete(MyPawn);

}

}

}

Now the game should be working.

We need a Mission Complete message to display.

**Screen Message using Widget**

Now let’s show some on screen messages for extraction zone.

Create widget named "Game Over"

On design tab just add a text that says "Mission Completed"



Go to GameMode BP and add blueprint to create widget and add to view port.



Now when you extract we should be getting the widget.

**Adding Sound**

Now we need to give a sound when player enters extraction zone without objective.

So go to ExtractionZone.h file and add the USoundBase pointer like the following

UPROPERTY(EditDefaultsOnly, Category = "Sounds")

USoundBase\* ObjectiveMissingSound;

Now go to the Extraction Zone.cpp file and add another if statement at start of HandleOverlap function

if (MyPawn == nullptr)

{

return;

}

Add an “else” to current if statement to play sound.

add the following to the else part

UGameplayStatics::PlaySound2D(this, ObjectiveMissingSound);

May need include statement

#include "Kismet/GameplayStatics.h"

final if statement will look like this

if (MyPawn != nullptr && MyPawn->bIsCarryingObjective)

{

AFPSObjectiveGameMode\* GM = Cast<AFPSObjectiveGameMode>(GetWorld()->GetAuthGameMode());

if (GM)

{

GM->MissionComplete(MyPawn);

}

}

else

{

UGameplayStatics::PlaySound2D(this, ObjectiveMissingSound);

}

Now in the FPSExtractionZone BP add a sound in the defaults. (UPROPERTY in header file of extraction zone.)

Game should play the sound when entering extraction zone without carrying objective.

(Make sure we have added a blueprint version of the extraction zone in the level.)

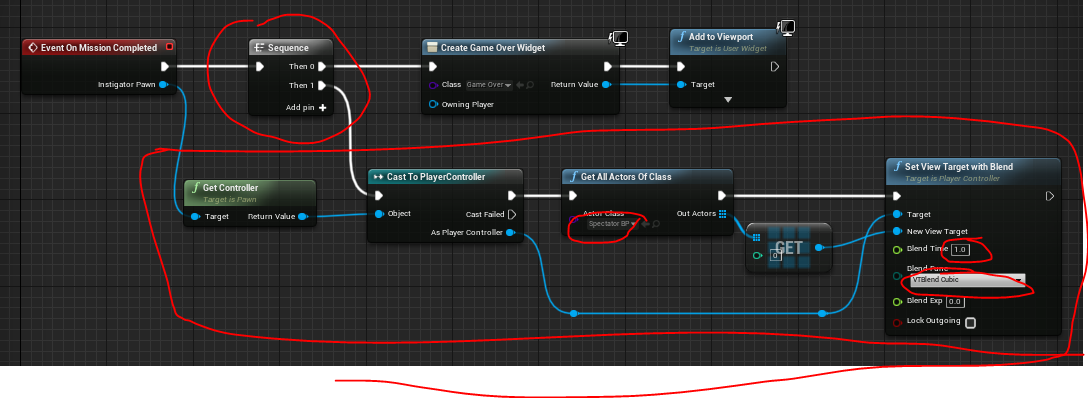
**Overview in Post Mission**

Camera moves from first person view to third person view.

**Prototype in BP**

Add a camera that focus on entire level and convert it to a BP named “SpectatorBP”

In GameModeBP on OnMissionComplete event edit BP as following.



Prototyping is done now. Game should work.

**Implementing in CPP**

First we need to get the PlayerController

To get player controller go to GameMode cpp to CompleteMission function and in if statement

APlayerController\* PC = Cast<APlayerController>(InstigatorPawn->GetController());

Now we need to SetViewTargetWithBlend to player controller. (similar function used in bp)

PC->SetViewTargetWithBlend(NewViewTarget, 5.0f, EViewTargetBlendFunction::VTBlend\_Cubic);

We can do a check for pointer before using variable PC

if (PC)

{

PC->SetViewTargetWithBlend(NewViewTarget, 5.0f, EViewTargetBlendFunction::VTBlend\_Cubic);

}

' NewViewTarget ' is an actor variable which is not declared at this point.

So lets declare it as a pointer to actor inside the “if” as follows

AActor\* NewViewTarget;

To get a value into the NewViewTarget we need to get all actors of class "SpectatorBP" (class we created in editor with camera in it) and get its first value just like we did in BP.

So add following statement

UGameplayStatics::GetAllActorsOfClass(this, SpectatingCameraClass, ReturnedActors);

Will need include

#include "Kismet/GameplayStatics.h"

Variables SpectatingCameraClass & ReturnedActors need to be declared.

In header file we need to expose SpectatingCameraClass to BP so that we can select a class in the editor.

UPROPERTY(EditDefaultsOnly, Category = "Spectating")

TSubclassOf<AActor> SpectatingCameraClass;

In CPP file inside ”if” we need to declare an array of actors that will be returned in the function GetAllActorsOfClass

TArray<AActor\*> ReturnedActors;

To get first element in ReturnedActors array do the following variable.

NewViewTarget = ReturnedActors[0];

To make sure there are atleast one actor in the array we check it with if

if (ReturnedActors.Num() > 0)

{

NewViewTarget = ReturnedActors[0];

}

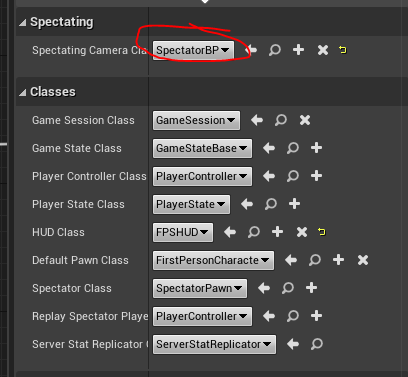
We can move the PC variable setting and setting NewViewTarget inside the above if statement

We can move all code in **MissionComplete** to an if condition that checks for spectator Camera class nullptr

Add and else part to the if to LOG a message.

UE\_LOG(LogTemp, Warning, TEXT("Cannot find Spectating Camera Class. Please Update Game Mode"))

We need to select the Spectator Class in GameMode default settings



The final finished code in MissionComplete function is as follows.

void AFPSObjectiveGameMode::MissionComplete(APawn \* InstigatorPawn)

{

if (InstigatorPawn)

{

InstigatorPawn->DisableInput(nullptr);

if (SpectatingCameraClass)

{

AActor\* NewViewTarget = nullptr;

TArray<AActor\*> ReturnedActors;

UGameplayStatics::GetAllActorsOfClass(this, SpectatingCameraClass, ReturnedActors);

if (ReturnedActors.Num() > 0)

{

NewViewTarget = ReturnedActors[0];

APlayerController\* PC = Cast<APlayerController>(InstigatorPawn->GetController());

if (PC)

{

PC->SetViewTargetWithBlend(NewViewTarget, 1.0f, EViewTargetBlendFunction::VTBlend\_Cubic);

}

}

}

else

{

UE\_LOG(LogTemp, Warning, TEXT("Cannot find Spectating Camera Class. Please Update Game Mode"))

}

}

//OnMissionCompleted(InstigatorPawn);

}

**Reference**

Check the documentation (https://docs.unrealengine.com/en-US/API/Runtime/Engine/Kismet/UGameplayStatics/GetAllActorsOfClass/index.html)