**Guards distraction.**

To look at where the noise came from we add the following code to **OnNoiseHear** method in **AIGuard.cpp**

**FRotationMatrix::MakeFromX(X).Rotator();**

X is a direction vector. But location is a unit vector. So to change Unit vector to direction vector add the following code

**FVector Direction = Location - GetActorLocation();**

**Direction.Normalize();**

We have to create a rotator variable from the earlier code. For that add the following code.

**FRotator NewLookAt = FRotationMatrix::MakeFromX(Direction).Rotator();**

Now we use the above variable to use in the **SetActorRotation** method. Final set of codes will look like this

**FVector Direction = Location - GetActorLocation();**

**Direction.Normalize();**

**FRotator NewLookAt = FRotationMatrix::MakeFromX(Direction).Rotator();**

**SetActorRotation(NewLookAt);**

The AIGuard should look at the direction of the noise. One problem is the guard looks up if the bullet hits a wall up.

Exercise: fix it.

Ans

**NewLookAt.Pitch = 0;**

**NewLookAt.Roll = 0;**

Now we need to turn the guard back to its initial rotation after a certain amount of time. So first go to the header file and create a variable as follows.

**FRotator OriginalRotation;**

Now we initialise the variable in the begin play method as follows.

**OriginalRotation = GetActorRotation();**

This will set the variable with the initial rotation of the guard.

Now we must set timer. So create a timer handle in the header file method

**FTimerHandle TimerHandle\_ResetOrientation;**

Now include the timemanager header file

**#include "TimerManager.h"**

Now go to the OnNoiseHear method and add the following codes

**GetWorldTimerManager().ClearTimer(TimerHandle\_ResetOrientation);**

**GetWorldTimerManager().SetTimer(TimerHandle\_ResetOrientation,this, &AFPSAIGuard::ResetOrientation, 3.0f);**

We need to create a function called **ResetOrientaion** that is passed into the method **SetTimer**.

Declare the function as follows in the header file.

**UFUNCTION()**

**void ResetOrientation();**

Create its implementation as follows in the C++ file.

**void AFPSAIGuard::ResetOrientation()**

**{**

**SetActorRotation(OriginalRotation);**

**}**

Now we need to add game over when the guard spots the player.

We are adding a rule to the game. This is done in gamemode. So in gamemode, we need to add a Boolean called **MissionComplete** to method **CompleteMission**

**void CompleteMission(APawn\* InstigatorPawn, bool MissionComplete);**

We need to add the same variable in declaration and implementation.

We need to add this variable to the BP version of the function **OnMissionCompleted** in the gamemode.

**void OnMissionCompleted(APawn\* InstigatorPawn, bool MissionComplete);**

In the gamemode.cpp file we need to add the variable in the **OnMissionControl** BP function call at the end inside the CompleteMission function as follows.

**OnMissionCompleted(InstigatorPawn, MissionSuccess);**

We are calling the MissionComplete function in **ExtractionZone**. So we need to pass the boolean while calling the function in the **ExtractionZone.cpp**  in the **HandleOverlap** function.

**GM->CompleteMission(MyPawn, true);**

Now we need to specify the failure state when the guard sees the player. So head into the **AIGuard.cpp** file and add the following.

**ACPPLesson02GameMode\* GM = Cast<ACPPLesson02GameMode>(GetWorld()->GetAuthGameMode());**

**if (GM)**

**{**

**GM->CompleteMission(SeenPawn, false);**

**}**

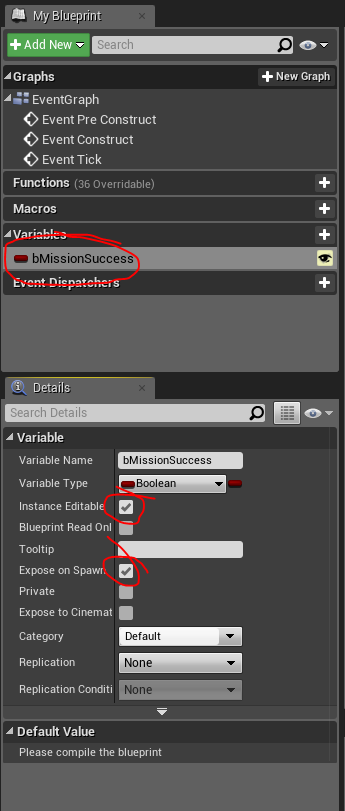
Need to include 2 header files in the AIGuard.cpp

**#include "CPPLesson02GameMode.h"**

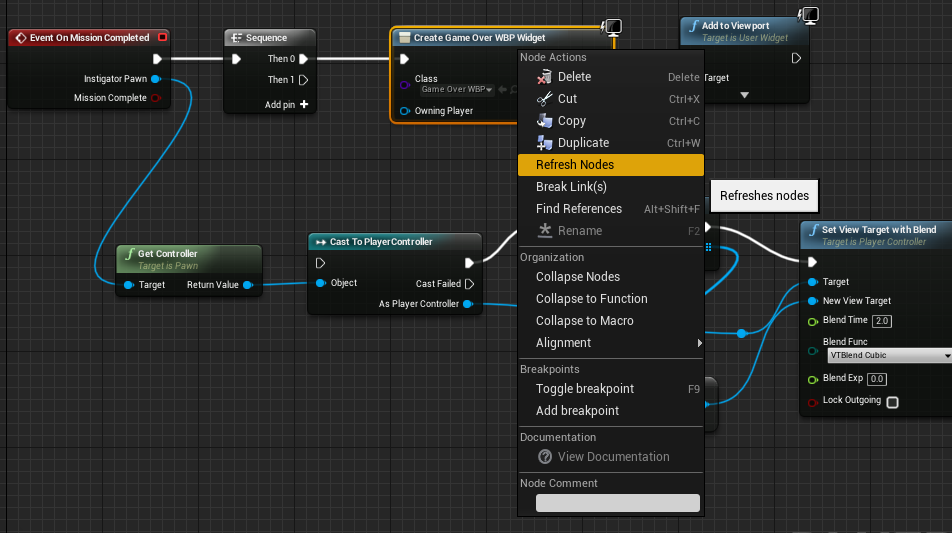
**#include "GameFramework/Actor.h"**

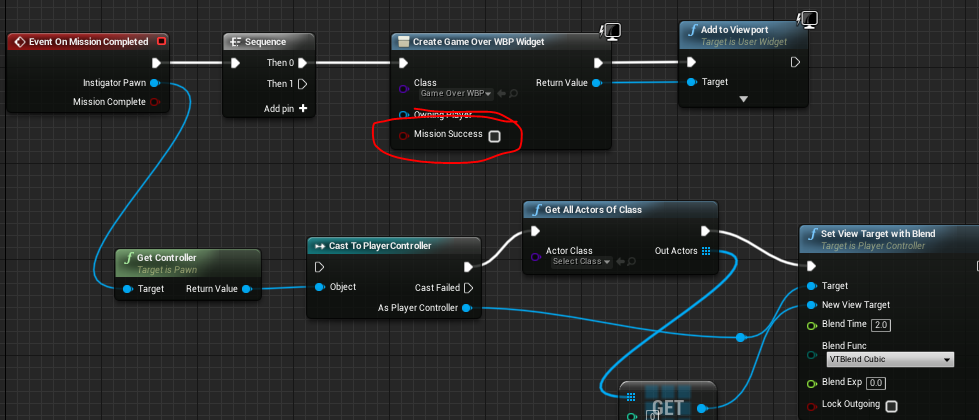
Now the when the AIGuard sees the player the game will be over.

Next we need to change the text when the game is over. It should tell the player whether the game is over or mission completed. To do this go to GameOverWidgetBP and create a Boolean variable called **bMissionSuccess** and expose on spawn and instance editable.



Now go to BP\_GameMode and rightclick on the create widget node an refreshnode and we will get the new variable create in the widget visible.





And connect the Boolean

