Hostage Mechanic

Technical documentation

Fabio Mangiameli

Fm083@hdm-stuttgart.de

Matrikel-Nr.: 39295

Content

1 Player Character

The Player Character class is located in \Source\HostageMechanic\HostageMechanicCharacter.h and is a modified version of the first-person template player class. The Actor which can be possessed by the player is a blueprint which inherits from HostageMechanicCharacter.h called FirstPersonCharacter. It can be found under \Content\MyContent\Blueprints

1.1 Functions

Setup Player Input

Inputs to for grabbing and restarting the game

```
void AHostageMechanicCharacter::SetupPlayerInputComponent(class
UInputComponent* PlayerInputComponent)
{

…

// Toggle E to grab hostage
PlayerInputComponent->BindAction("GrabHostage", IE_Pressed, this,
&AHostageMechanicCharacter::GrabHostage);

// Press O to restart the Level
PlayerInputComponent->BindAction("Restart", IE_Pressed, this,
&AHostageMechanicCharacter::RestartGame);
}
```

Restart Game

Function will load the level again and restarts this way the game.

```
void AHostageMechanicCharacter::RestartGame()
{
   UGameplayStatics::OpenLevel(this, FName(*GetWorld()->GetName()),false);
}
```

Camera Shake Grabbed

The function will trigger the camera shake which happens when you have a hostage grabbed.

```
void AHostageMechanicCharacter::CameraShakeGrabbed()
{
   GetWorld()->GetFirstPlayerController()->PlayerCameraManager-
>StartCameraShake(CamShake,1);
}
```

Take Damage

Override of the unreal take damage function which will be triggered when a damage event happens. It will reduce the player's health with a defined amount of damage. If the player is grabbing a hostage, he will drop it. If the health is equal or below 0 the player is dead which instantly restarts the game.

```cpp
float AHostageMechanicCharacter::TakeDamage(float DamageAmount,
FDamageEvent const& DamageEvent,
   AController* EventInstigator, AActor* DamageCauser)
{
   Health = Health - DamageAmount;

      if(HostageIsGrabbed)
      {
         UE_LOG(LogTemp,Warning,TEXT("Release"))
         GrabHostage();
      }
       if(Health <= 0)
       {
          RestartGame();
       }

       return DamageAmount;
}
```

Grab Hostage

The function is split into two parts the first one will handle the release of the hostage and the second part handles the logic for grabbing.

Grabbing

A 200-unit line trace is shot from the player in the forward direction. If a hit is detected a cast will prove that the hit actor is from type AHostagePawn otherwise the declaration of GrabbedHosage will not work.

If GrabbedHostage is declared and the player is not already grabbing a Hostage, GrabbedHostage gets attached to the player mesh in a socket called “GrabSocket”. Also, the whole collision for the hostage mesh will be turned off except for the camera trace channel. This is later important for the Behavior Tree Task “BTT_Attack”.

Now the player penalty gets set up. The character movement, turn, and look-up rate will be slowed, and a timer will call the CameraShakeGrabbed function.

```
void AHostageMechanicCharacter::GrabHostage()
{

…

//If Hostage is not Grabbed attach it to the player
   else
   {
      FHitResult OutHit;
      FVector Start = FP_Gun->GetComponentLocation();

      FVector ForwardVector = FirstPersonCameraComponent->GetForwardVector();
      FVector End = ((ForwardVector * 200) + Start);
      FCollisionQueryParams CollisionParams;

      if(GetWorld()->LineTraceSingleByChannel(OutHit, Start, End, ECC_Camera,
CollisionParams))
      {
         GrabbedHostage = Cast<AHostagePawn>(OutHit.GetActor());

         if(GrabbedHostage && !HostageIsGrabbed)
         {
            HostageIsGrabbed = true;

            GrabbedHostage->GetMesh()->SetCollisionResponseToAllChannels(ECR_Ignore);
            GrabbedHostage->GetMesh()->SetCollisionResponseToChannel(ECC_Camera,
ECR_Block);
            GrabbedHostage->AttachToComponent(Mesh1P,
FAttachmentTransformRules::SnapToTargetIncludingScale, "GrabSocket");

            GetCharacterMovement()->MaxWalkSpeed = SlowedMovementSpeed;

            BaseTurnRate = 10.f;
            BaseLookUpRate = 10.f;

            Turn(1);
            LookUp(-1);

            GetWorldTimerManager().SetTimer(
               CameraShakeTimer, this, &AHostageMechanicCharacter::CameraShakeGrabbed,
0.25,true);
         }
      }
   }
}
```

Release Hostage

To have a proper position after the release, the rotation of the hostage has to be changed. The rotation gets rested to 0 except for the yaw rotation.

For the new location, a line trace is shot 1000 units down from the hostage. The hit result will be saved as the new location for the hostage.

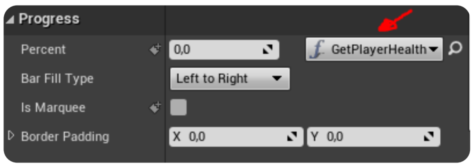
Now the hostage gets detached from the player mesh and set to the new location. After that, the collision and all player properties are reset too normal.

```
void AHostageMechanicCharacter::GrabHostage()
{
    //If Hostage is Grabbed, detach it and reset the location.
    if(HostageIsGrabbed)
    {
        FRotator NewRotation = FRotator(0,GrabbedHostage->GetActorRotation().Yaw,0);
        GrabbedHostage->SetActorRotation(NewRotation);

        FHitResult OutHit;
        FVector Start = GrabbedHostage->GetRootComponent()->GetComponentLocation();

        FVector DownVector = GrabbedHostage->GetRootComponent()->GetUpVector();
        FVector End = ((DownVector * -1000) + Start);
        FCollisionQueryParams CollisionParams;

        GetWorld()->LineTraceSingleByChannel(OutHit, Start, End, ECC_Visibility, Colli-
sionParams);

        GrabbedHostage->DetachFromActor(FDetachmentTransformRules::KeepWorldTransform);

        FVector NewPosition = FVector(OutHit.Location);

        GrabbedHostage->SetActorLocation(NewPosition);
        GrabbedHostage->GetMesh()->SetCollisionResponseToAllChannels(ECR_Block);
        GrabbedHostage->GetMesh()->SetCollisionResponseToChannel(ECollisionChannel::EN-
EMY_LINE_OF_SIGHT, ECollisionResponse::ECR_Ignore);

        GetCharacterMovement()->MaxWalkSpeed = NormalMovementSpeed;

        GetWorldTimerManager().ClearTimer(CameraShakeTimer);

        BaseTurnRate = 60.f;
        BaseLookUpRate = 60.f;

        Turn(1);
        LookUp(-1);

        HostageIsGrabbed = false;
    }

…

}
```

1.2 Health Bar Widget

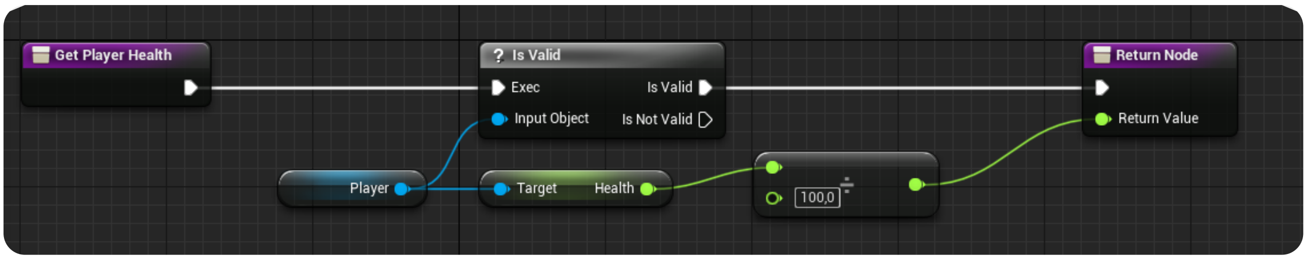
The health bar is located in \Content\MyContent\Blueprint\HealthBar.

It is a progress bar and is using the player health as percent value.



Progress

Percent	0,0	GetPlayerHealth
Bar Fill Type	Left to Right	
Is Marquee		
Border Padding	X 0,0	Y 0,0



2 Enemy AI

The Enemy AI can be split into three big parts. The Enemy class, written in C++ includes all components, basic functions, and important variables. The AI Controller, made with Blueprints, which will control the Enemy Class, and the most important part the Behavior Tree, which is created with the Unreal Behavior Tree Tool and Blueprints for the different Behavior Tree Tasks.

2.1 Enemy Class

The Enemy Class is located in \Source\HostageMechanic\EnemyCharacter.h. To place enemies in the level there is a Blueprint called BP_EnemyAI which is located in \Content\MyContent\EnemyAI\AI.

Shoot

The function will spawn a projectile at the location of the weapon muzzle.

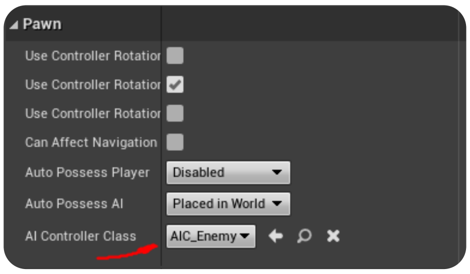
```
void AEnemyCharacter::Shoot(AHostageMechanicCharacter* Player)
{
   // try and fire a projectile
   if (ProjectileClass != nullptr)
   {
      UWorld* const World = GetWorld();
      if (World != nullptr)
      {
         const FRotator SpawnRotation = UKismetMathLibrary::FindLookAtRotation(Muzzle-
>GetComponentLocation(), Player->GetRootComponent()->GetComponentLocation());

         const FVector SpawnLocation = Muzzle != nullptr ? Muzzle->GetComponentLocation() :
GetActorLocation();

         //Set Spawn Collision Handling Override
         FActorSpawnParameters ActorSpawnParams;
         ActorSpawnParams.SpawnCollisionHandlingOverride =
ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButAlwaysSpawn;

         // spawn the projectile at the muzzle
         World->SpawnActor<AHostageMechanicProjectile>(ProjectileClass, SpawnLocation,
SpawnRotation, ActorSpawnParams);
      }
   }
}
```

Take Damage

Override of the unreal take damage function which will be triggered when a damage event happens. It will reduce the enemy's health with a defined amount of damage. If Health is below or equal to 0 the enemy is dead. The bool IsDead is set to true. This variable is important for the Animation State Machine and the Behavior Tree. The AI Stats Widget gets destroyed and after 5 seconds the whole Enemy Class gets destroyed.

```
float AEnemyCharacter::TakeDamage(float DamageAmount, FDamageEvent const& DamageEvent, AController*
EventInstigator,
   AActor* DamageCauser)
{
   Health = Health - DamageAmount;
   if(Health <= 0)
   {
      IsDead = true;
      SetActorEnableCollision(false);

      Widget->DestroyComponent();

      FTimerHandle UnusedHandle;
      GetWorldTimerManager().SetTimer(UnusedHandle, this, &AEnemyCharacter::DestroyCharacter, 5,
false);
   }
   return DamageAmount;
}
```

2.2 AI Controller

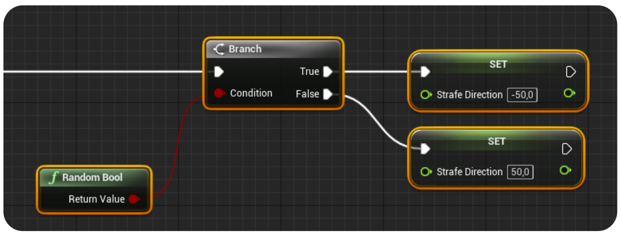


Pawn	
Use Controller Rotatio	☐
Use Controller Rotatio	✓
Use Controller Rotatio	☐
Can Affect Navigation	☐
Auto Possess Player	Disabled
Auto Possess AI	Placed in World
AI Controller Class	AIC_Enemy

The AI Controller is located under \Content\MyContent\EnemyAI\AI\AIC_Enemy.

To use the AI Controller for the Enemy Class you have to select it inside of the BP_Enemy detail panel under Pawn.

Event Begin Play

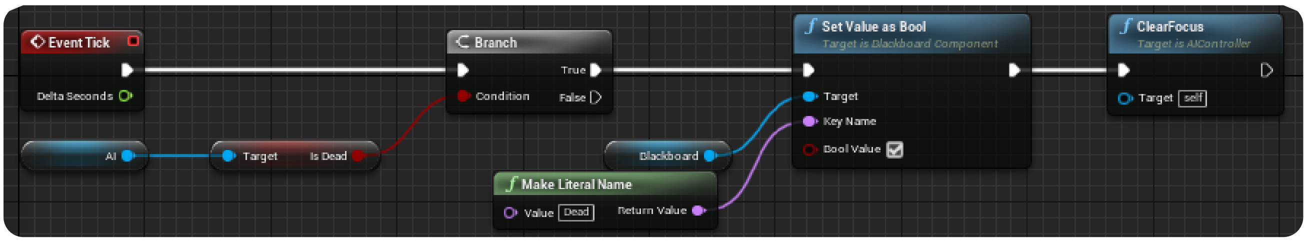


The event node will start the Behavior Tree and declare variables that are important for the AI Controller.

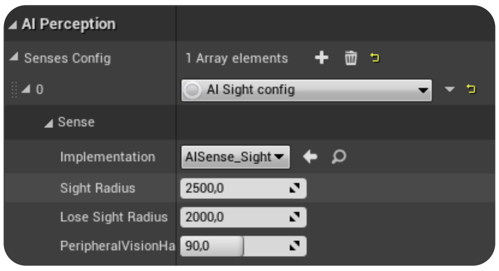
The enemy can strafe. To define the strafe direction randomly the node Random Bool is used.

Event Tick

Every Tick the AI Controller will check if the enemy is dead. If it is true, the bool Dead in the blackboard is set which will cause the death behavior in the behavior tree to happen.



AI Perception



AI Perception	
Senses Config	1 Array elements
0	AI Sight config
Sense	
Implementation	AISense_Sight
Sight Radius	2500,0
Lose Sight Radius	2000,0
PeripheralVisionHa	90,0

The AI Perception component is used for the Sight Sense of the AI. Inside of the details the radius in which the AI should detect actors can be defined in the details panel as Sight Radius.

However, I encountered strange behavior with the AI Perception where it sometimes would lose sight when the player is standing right in front of the AI. That is why I implemented a “line of sight check” myself inside of the Behavior Tree. Now the perception is only used to set the focus for the AI on the player.

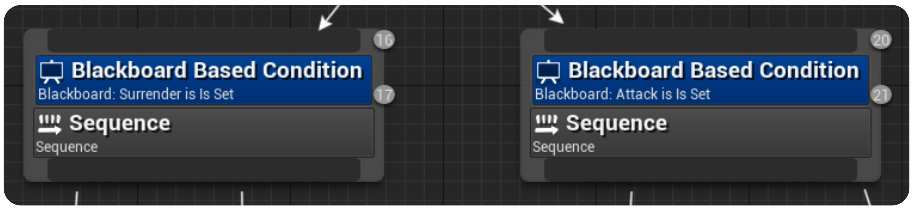
2.3 Behavior Tree

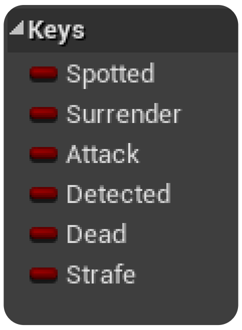
The Behavior Tree is located under \Content\MyContent\EnemyAI\AI\BT_Enemy. A Blackboard is used which can be found in the same folder with the name BB_Enemy. All Behavior Tree Tasks can be found in the folder BTT.

2.3.1 Blackboard

In the Blackboard are several Booleans stored. These are used for the Transitions in the Behavior Tree.

For example, if the AI should surrender or Attack.


Blackboard Based Condition
Blackboard: Surrender is Is Set
Sequence
Sequence
Blackboard Based Condition
Blackboard: Attack is Is Set
Sequence
Sequence

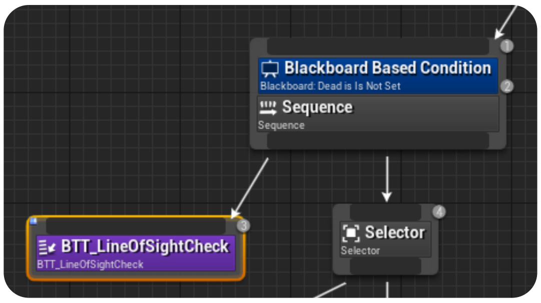


Blackboard values

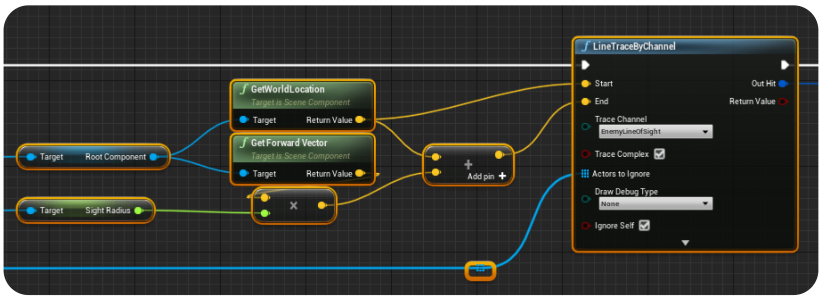
2.3.2 Behavior Tree Tasks

Behavior Tree Tasks are used to make the AI do specific things at a specific moment.

For example, in the picture on the right, the Task “BTT_LineOfSightCheck” will be executed as long as the Blackboard value Dead is not true.



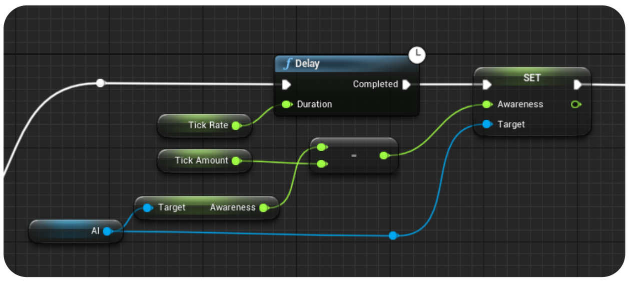
Line Of Sight Check



This task will check if the enemy can see the player or not. To do that a line trace is shot in forward direction starting from the AI. Because of the AI focus on the player, the line trace will always go in the direction of the player. If the line of sight check was successful, the blackboard value spotted is set to true.

Decrease and Increase Awarness

To increase and decrease the enemy awareness two Behavior Tree Tasks are used. As long as the Blackboard value Spotted is true the task Increase Awareness will be called. If Spotted is false Decrease Awareness is called.

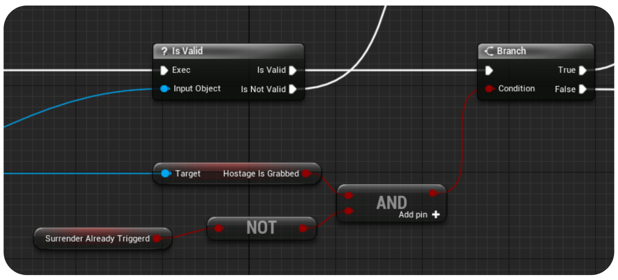


The Tick Rate is the seconds between every tick and the Tick Amount is how much will be added or deducted to the awareness of every Tick. If the awareness reaches 1,0 the blackboard value Detected is set true and awareness can't be increased anymore. If the awareness is dropping Detected will be instantly set false.

Reset Attack Status

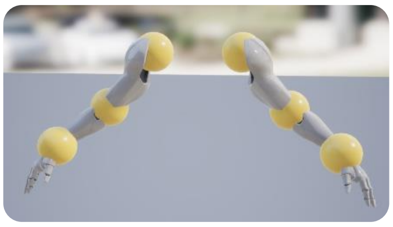
This task is only to reset the Blackboard values Surrender and Attack if the player isn't detected anymore.

Surrender Trigger

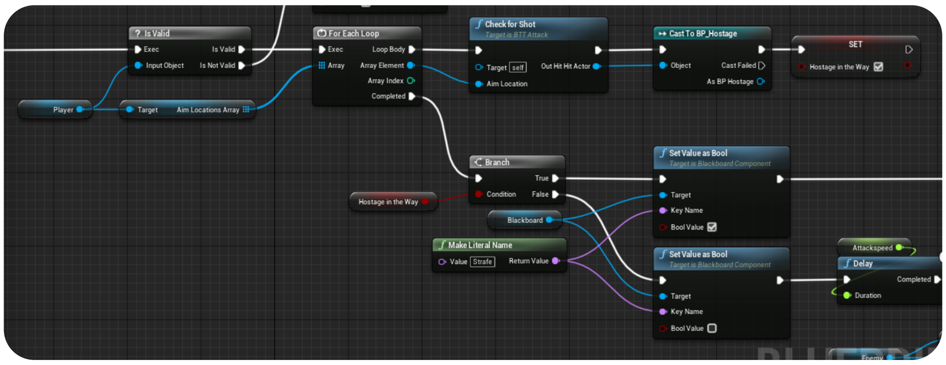


The task is used to check if the player has a hostage grabbed and if the Surrender event hasn't already been triggered. If yes the Blackboard value Surrender is set to true and Attack false. Otherwise, Surrender is set false and Attack true.

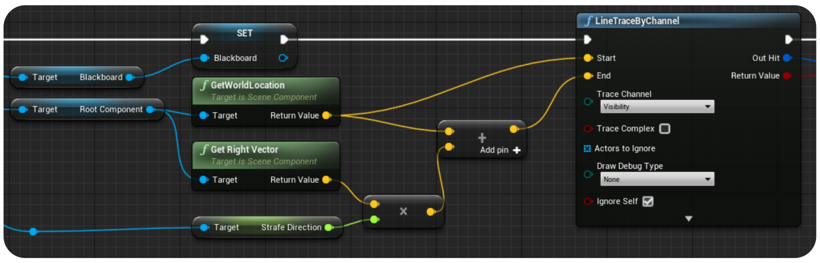
Attack



In this task, the enemy will check if the player can't be shot because the hostage is in the way. To do that six line traces are shot with a for each loop from the enemy to the yellow marked locations in the picture. With more than one line trace it's much more precise for the enemy to see if the Player Mesh is covered by the Hostage or not. If one line trace detects that the hit actor is a BP_Hostage the bool "Hostage in the way" is set true. If it's true the enemy can't attack and the Blackboard value Strafe is set to true. If the hostage is not in the way the Attack Montage is played which will result in the enemy shooting at the player.



Strafe

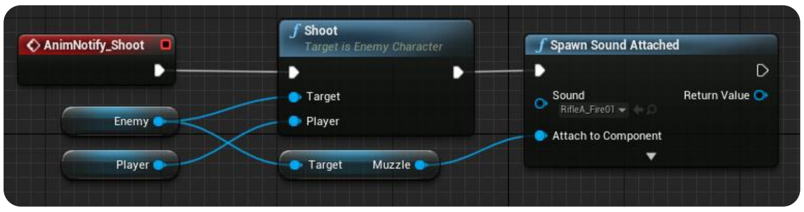


To get a free shot the enemy will strafe around the player. To find the location where to move at a line trace from the enemy is shoot to the side. The direction is depending on the earlier explained Strafe Direction variable. If the line trace hits a wall Strafe Direction gets switched.

2.4 Animation Blueprint

The Animation Blueprint is located in /Content/MyContent/Animations/ABP_EnemyAI.

AnimNotify Shoot



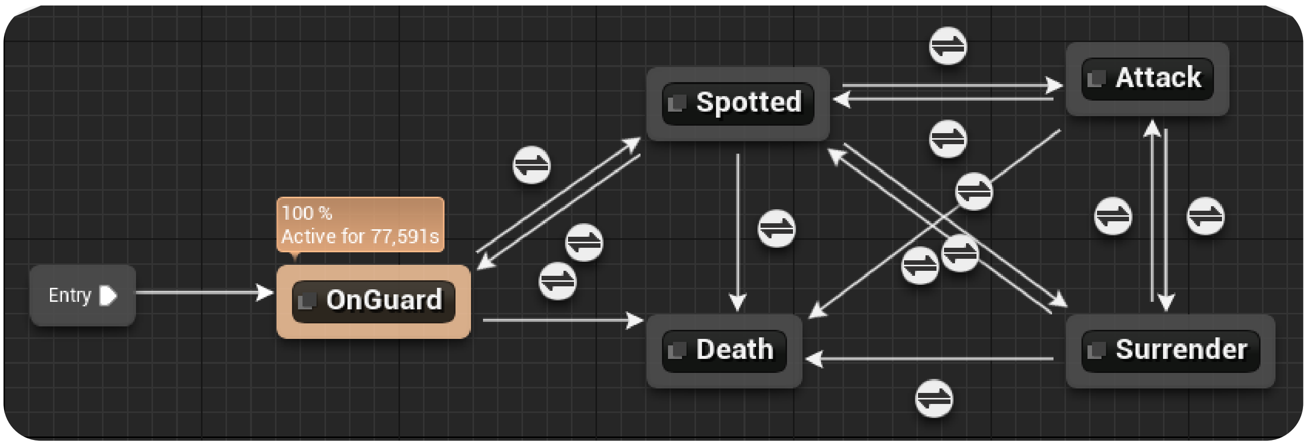
This node will be triggered if the Attack Montage which is played in the Behavior Tree Task BTT_Attack reaches a certain point. It will call the Shoot function in the Enemy Class and spawn a sound at the weapon muzzle.

Event Blueprint Update Animation

The different variables for the State Machine Transitions have to be updated. This event will extract every tick all the important information's out of the Blackboard and also checks if the enemy is moving.

State Machine

All transitions check if a bool from the Blackboard is set and change state regarding to the result.



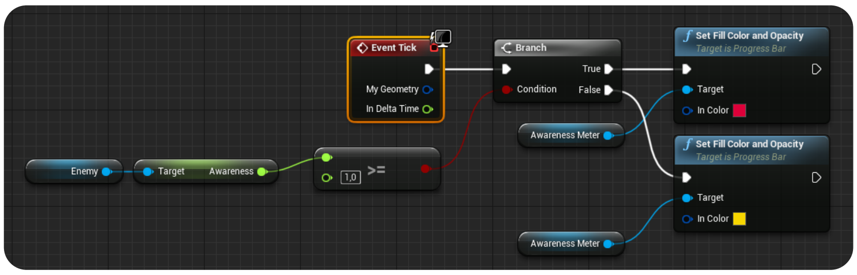
2.5 AI Stats Widget

The AI Stats Widget is located in /Content/MyContent/EnemyAI and is used to show the Health and Awareness of the enemy to the player.

Two Progress Bars are used. One has a binding to the enemy variable Health and the other one to the enemy variable Awareness. For Health it is the same implementation as in the "Player Health Bar". The Awareness Progress Bar has a direct binding to the enemy variable Awareness.

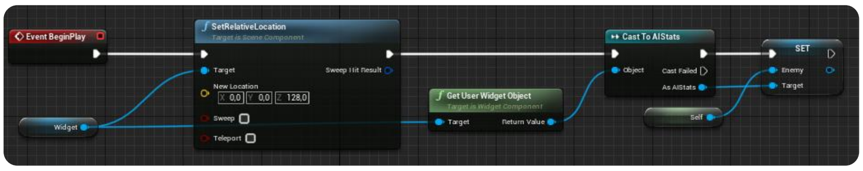
Event Tick

To turn the Awareness Meter red when the enemy has detected the player the Event Tick is used and will turn the color into red when the awareness is more or equal to 1,0.



Setup Widget

The reference for the Enemy which is using the Widget is handed over in the event graph of BP_Enemy. Also, I had to relocate the Widget because of a bug that completely changes the location of it when BP_Enemy is placed in the World.



3 Hostage

The Hostage Class is located in /Source/HostageMechanic/HostagePawn.h. A Blueprint called BP_Hostage which inherits from HostagePawn.h is located in Content/MyContent/Hostage and is used to place hostages in the world.

The class only has a getter for the Hostage Mesh to use in the grab function inside the Player Class.

4 Other Classes

4.1 Projectile Class

The Projectile Class is located in /Source/HostageMechanic/HostageMechanicProjectile.h and is a modified version of the “First Person Template Projectile”. It is used for the player and the enemies.

On Hit

The projectile will trigger an Overlap Event and check if the overlapping actor can be damaged. If yes, the Take Damage function of the Actor is called.

The event will always spawn a particle effect and destroy the projectile.

```
void AHostageMechanicProjectile::OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor,
UPrimitiveComponent* OtherComp, FVector NormalImpulse, const FHitResult& Hit)
{
		const FVector WorldLocationProjectile = RootComponent->GetComponentLocation();

	if (OtherActor && OtherActor->CanBeDamaged() && !OtherActor->IsPendingKill())
	{
		TSubclassOf<UDamageType> DmgTypeClass = UDamageType::StaticClass();
		OtherActor->TakeDamage(Damage, FDamageEvent(DmgTypeClass), DamageInstigator, this);
	}

	UGameplayStatics::SpawnEmitterAtLocation(GetWorld(), ParticleEffect, WorldLocationProjectile);

	Destroy();
}
```

4.2 Camera Shake

The Camera Shake Class is located in /Source/HostageMechanic/CameraShakeGrabbed.h. Inside the folder /Content/MyContent/Blueprint is a Blueprint called BP_CameraShakeGrabbed which inherits from CameraShakeGrabbed.h. The Blueprint can be used to easily change all the values.

5 Used Content

List of content I used which was not created by me.

I used the Unreal First-Person Shooter Template and modified some of the content.

Animation Starter Pack from Epic Games:

- Hostage_Mannequin
- Hostage_PhysicsAsset
- Hostage_Skeleton
- Idle_Rifle_Hip

Prototype Weapons from Ying Pei Games:

- Everything in the folder /Content/PrototypeWeap

Animations and Mesh from Mixamo

- Everything in the folder /Content/MyContent/EnemyAI/Mesh
- Attack
- Attack_Idle
- Cheering
- Clapping
- Death
- OnGuard
- Spotted
- Strafe_Left
- Surrender