Created 2020/09/04 – Last revision on 2020/10/04

Steps For a Simple C++ Test of the UE4.24.3 HTN Planner Plugin

@NoFrogPlansMore

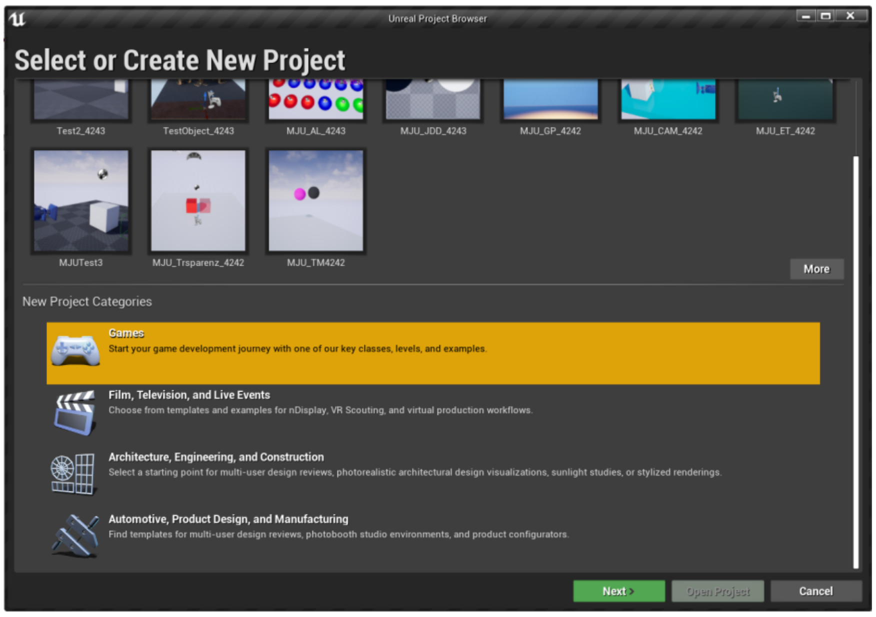
Objective: A First C++ test of the UE4.24.3 HTN Planner Plugin

Content:

Section 1 : Required Project Setup (≤1/4h) (Unreal Documentation – Programming Quick Start)

You should have a copy of **Visual Studio 2017** for Desktop (Preferably Community or Professional Edition) installed before starting this tutorial. For setup instructions, please see Setting Up Visual Studio for UE4 .

1. Open **Unreal Engine** from the Launcher. The Unreal Project Browser window will appear. Select "Games" in New Project Categories and click on "Next".



New Project Categories

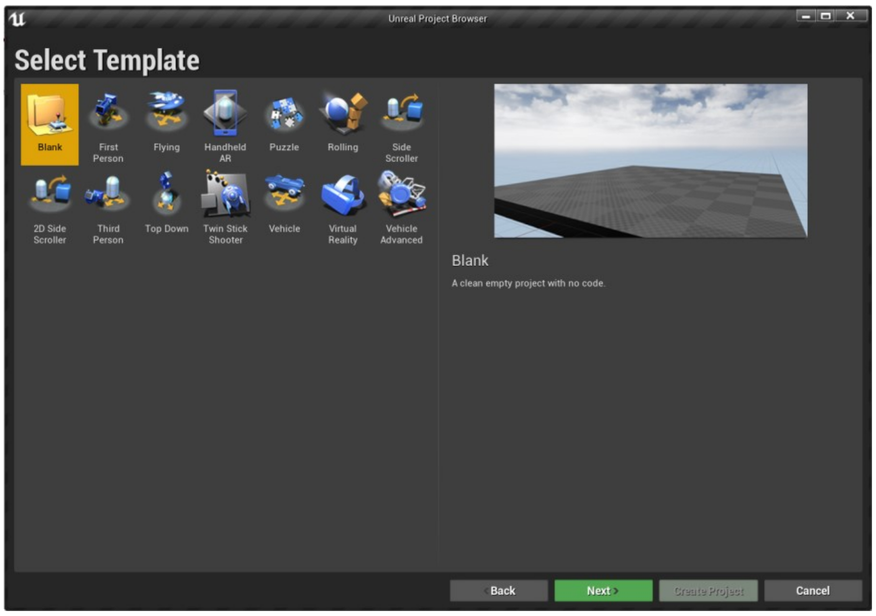
Games
Start your game development journey with one of our key classes, levels, and examples.

Film, Television, and Live Events
Choose from templates and examples for nDisplay, VR Scouting, and virtual production workflows.

Architecture, Engineering, and Construction
Select a starting point for multi-user design reviews, photorealistic architectural design visualizations, sunlight studies, or stylized renderings.

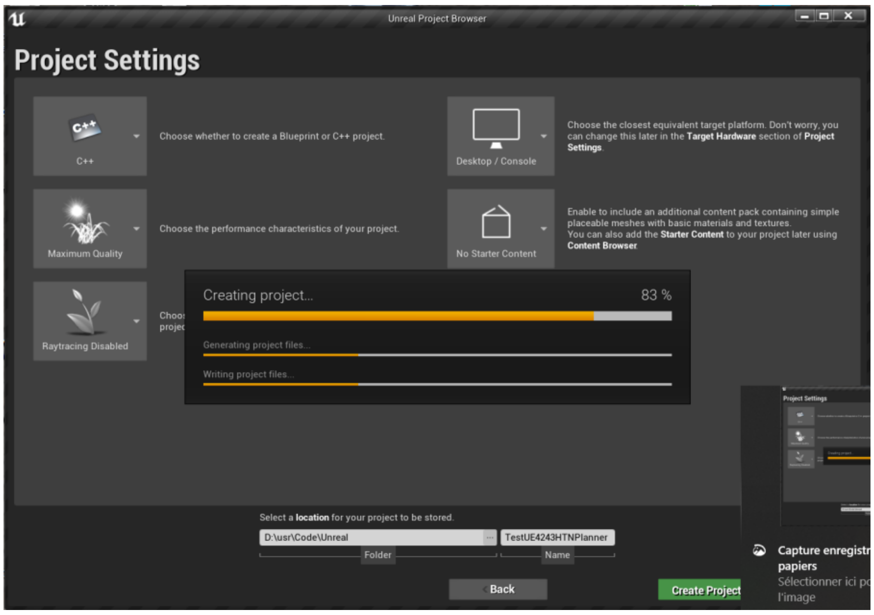
Automotive, Product Design, and Manufacturing
Find templates for multi-user design reviews, photobooth studio environments, and product configurators.

2. Select the Blank Template and then click on "Next":



Blank
A clean empty project with no code.

3. In the Project Settings window, make sure to select **C++** and as this is a simple test, **Without Starter Content**. Eventually enter a name for your project; I here chose "TestUE4243HTNPlanner" as a project name. We can now click **Create Project** and get started:



Unreal Project Browser

Project Settings

C++ — Choose whether to create a Blueprint or C++ project.

Desktop / Console — Choose the closest equivalent target platform. Don't worry, you can change this later in the **Target Hardware** section of **Project Settings**.

Maximum Quality — Choose the performance characteristics of your project.

No Starter Content — Enable to include an additional content pack containing simple placeable meshes with basic materials and textures. You can also add the **Starter Content** to your project later using **Content Browser**.

Raytracing Disabled

Creating project... 83 %

Generating project files...

Writing project files...

Select a **location** for your project to be stored.

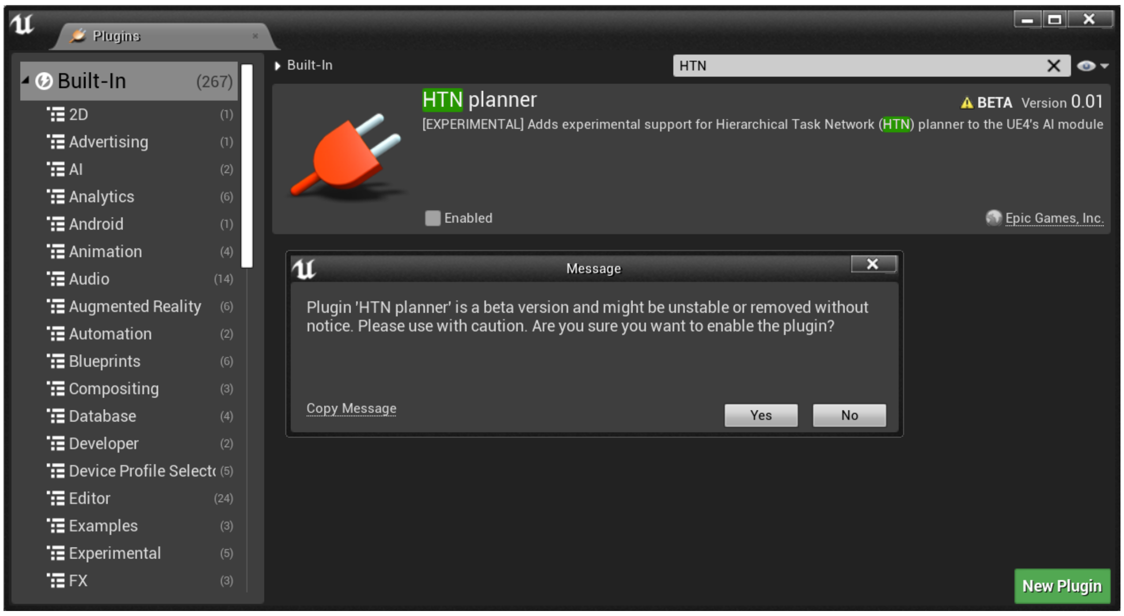
Folder: D:\usr\Code\Unreal

Name: TestUE4243HTNPlanner

The **Unreal Editor** will now open our new project. **Visual Studio** will also open and load the solution file that our project has created.

<u>Subsection 1.1 :</u> Make the HTN Planner Plugin available for your project

1. Left-click sur Settings > Plugins et look for HTN; when you click on the "Enabled" box, a Message appears telling you the Plugin is a beta version:



HTN planner — BETA Version 0.01

[EXPERIMENTAL] Adds experimental support for Hierarchical Task Network (HTN) planner to the UE4's AI module

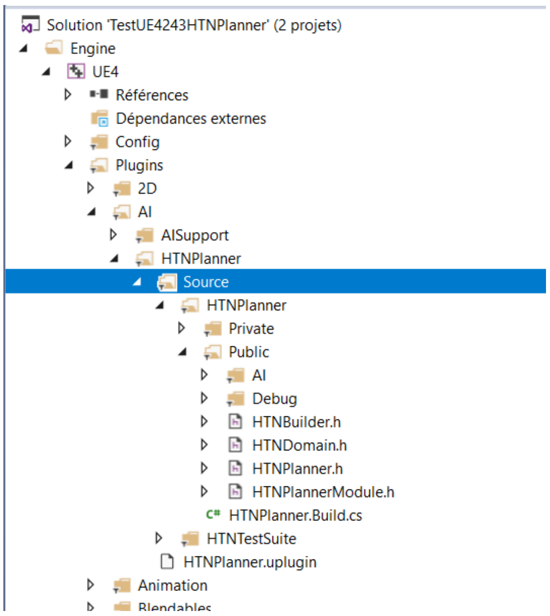
Enabled — Epic Games, Inc.

Message

Plugin 'HTN planner' is a beta version and might be unstable or removed without notice. Please use with caution. Are you sure you want to enable the plugin?

2. Left-click on "Yes" and then on "Restart Now". Unreal shuts down and immediately restarts with the HTN Planner Plugin enabled. Close the Plugins window.

3. In the Solution explorer of **Visual Studio**, you can check the access of the C++ code of the "HTNPlanner" from Engine ▸ UE4 ▸ Plugins ▸ AI ▸ HTNPlanner ▸ Source:



4. In Visual Studio, open the file:

   Games ▸ TestUE4HTNPlanner ▸ Source ▸ TestUE4HTNPlanner ▸ TestUE4HTNPlanner.Build.cs

5. Line 13, add `"HTNPlanner"` to the list of Private Dependency Module Names:

```
PrivateDependencyModuleNames.AddRange(new string[] { "HTNPlanner" });
```

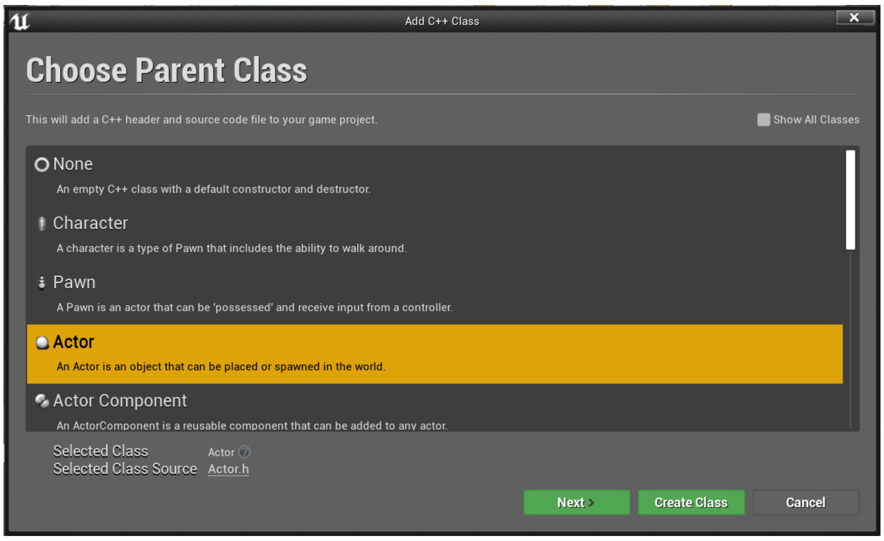
   If you forgot to do this, the compiler won't be able to `#include` the necessary HTN Planner Plugin public files for this test.

6. Get back to the Unreal Editor to left-click on "Compile". Wait for the compilation to complete successfully.

<u>Subsection 1.2 :</u> Add a C++ class to the project

We now add a class to our project to later call the HTN Planner with the `BeginPlay()` method.

1. In the Unreal Editor, Left-Click File > New C++ Class... Choose "Actor" as a Parent Class:



This will add a C++ header and source code file to your game project.

None
An empty C++ class with a default constructor and destructor.

Character
A character is a type of Pawn that includes the ability to walk around.

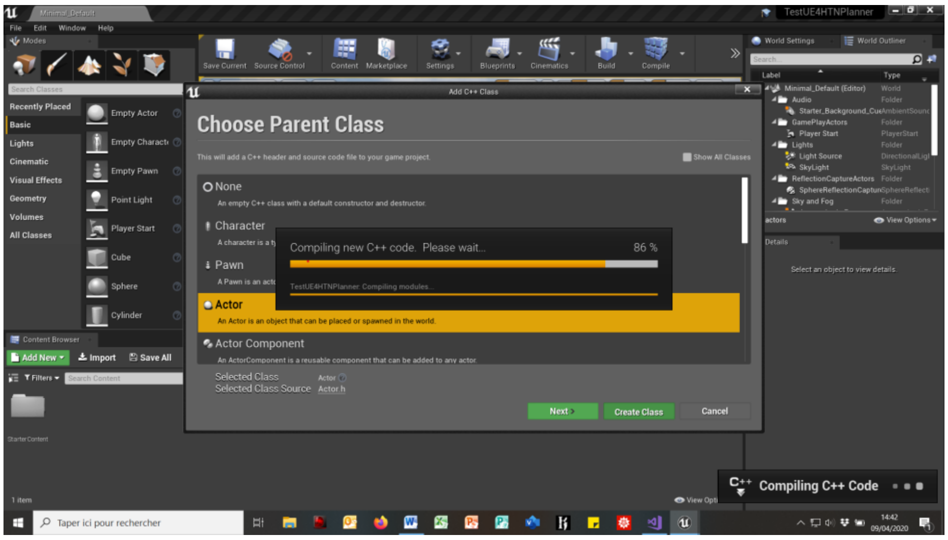
Pawn
A Pawn is an actor that can be 'possessed' and receive input from a controller.

Actor
An Actor is an object that can be placed or spawned in the world.

Actor Component
An ActorComponent is a reusable component that can be added to any actor.

Selected Class Actor
Selected Class Source Actor.h

2. Left-Click on “Create Class” (UE4 proposes to name your Actor Class “MyActor” which is just fine for this test) to generate and compile various files to your project:



This will add a C++ header and source code file to your game project.

None
An empty C++ class with a default constructor and destructor.

Character
A character is a t

Pawn
A Pawn is an acto

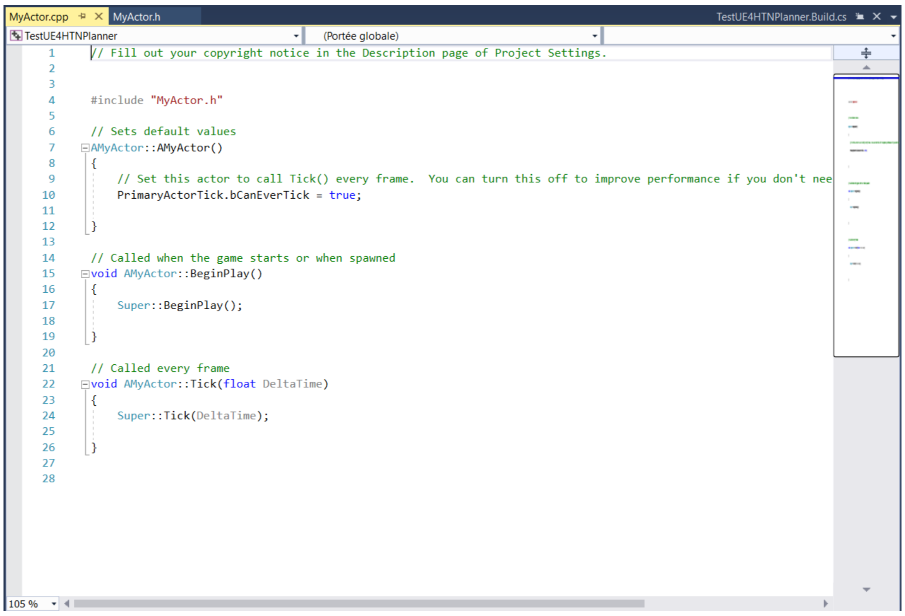
Compiling new C++ code. Please wait... 86 %
TestUE4HTNPlanner: Compiling modules...

Actor
An Actor is an object that can be placed or spawned in the world.

Actor Component
An ActorComponent is a reusable component that can be added to any actor.

Selected Class Actor
Selected Class Source Actor.h

3. Switch to Visual Studio and check file ‘’MyActor.cpp” is open and looks like this:



```cpp
// Fill out your copyright notice in the Description page of Project Settings.


#include "MyActor.h"

// Sets default values
AMyActor::AMyActor()
{
    // Set this actor to call Tick() every frame.  You can turn this off to improve performance if you don't nee
    PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void AMyActor::BeginPlay()
{
    Super::BeginPlay();

}

// Called every frame
void AMyActor::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

}

```

<u>Section 2 :</u> Testing plan generation (≤3/4h)

<u>Subsection 2.1 :</u> Enabling console logging

1. Open “MyActor.h” in Visual Studio and insert the following after `#include "MyActor.generated.h"` (which you can find in line 7) :

   ```
   DECLARE_LOG_CATEGORY_EXTERN(MyLogHTNPlanner, Log, All);
   ```

   This could be inserted line 9, for instance.

2. Add the following to “MyActor.cpp” , just after `#include "MyActor.h"` (ligne 4):

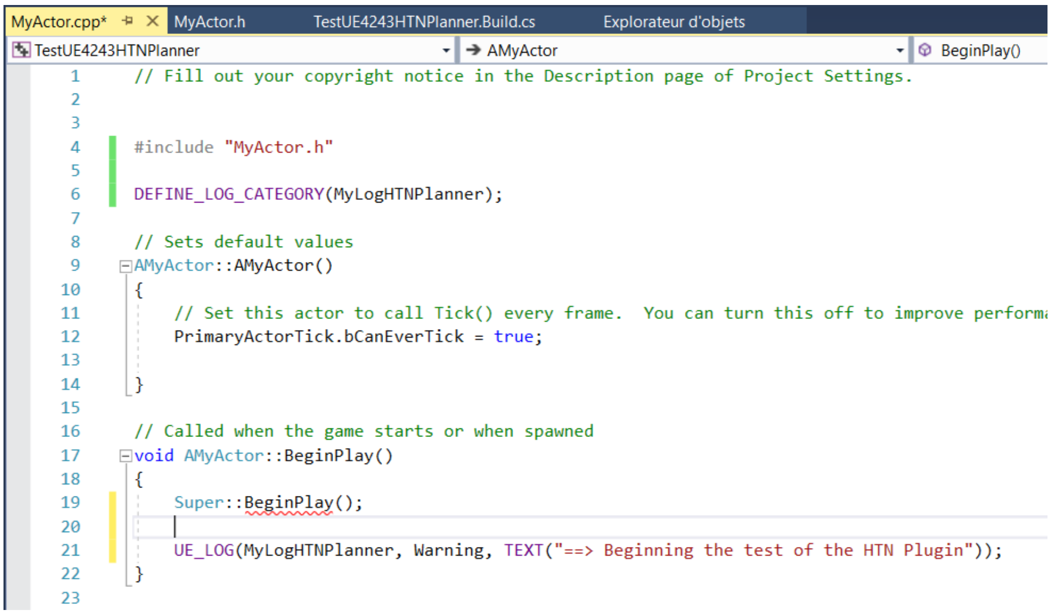
   ```
   DEFINE_LOG_CATEGORY(MyLogHTNPlanner);
   ```

   This could be inserted line 6, for instance.

3. Add the following to the body of `BeginPlay()` in “MyActor.cpp”:

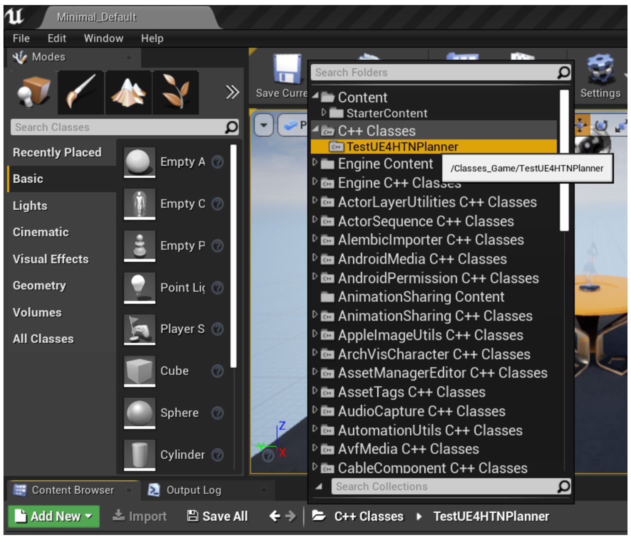
   ```
   UE_LOG(MyLogHTNPlanner, Warning, TEXT("==> Beginning the test of the HTN Plugin"));
   ```

   This could be inserted line 21, for instance.

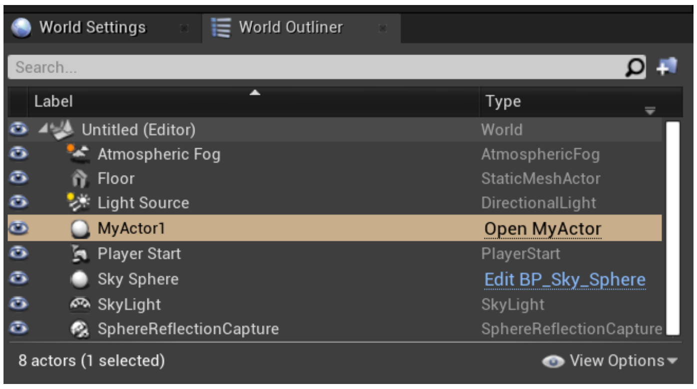


```
1   // Fill out your copyright notice in the Description page of Project Settings.
2
3
4   #include "MyActor.h"
5
6   DEFINE_LOG_CATEGORY(MyLogHTNPlanner);
7
8   // Sets default values
9   AMyActor::AMyActor()
10  {
11      // Set this actor to call Tick() every frame.  You can turn this off to improve performa
12      PrimaryActorTick.bCanEverTick = true;
13
14  }
15
16  // Called when the game starts or when spawned
17  void AMyActor::BeginPlay()
18  {
19      Super::BeginPlay();
20      
21      UE_LOG(MyLogHTNPlanner, Warning, TEXT("==> Beginning the test of the HTN Plugin"));
22  }
23
```

4. Back to the Unreal Editor, make sure C++ Classes > TestUE4HTNPlanner is open in the Content Browser:



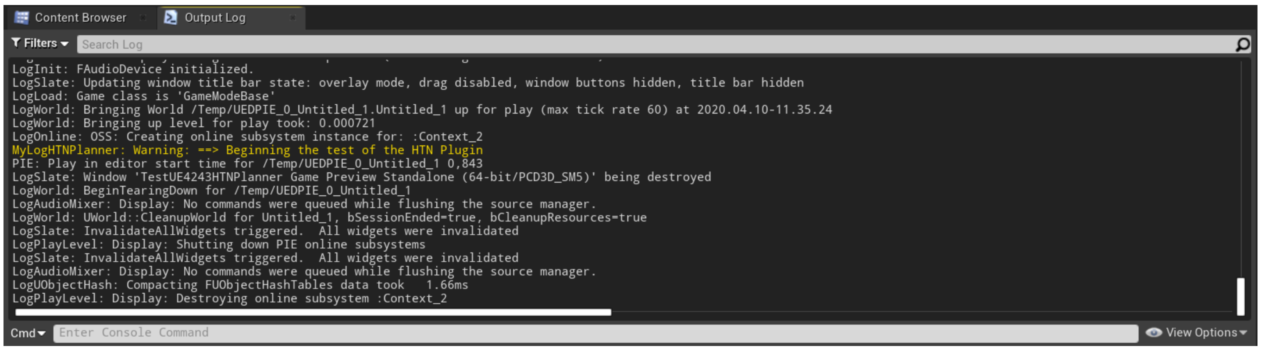
5. Drag and drop MyActor anywhere in the game scene; “MyActor1” must now appear in the “World Outliner”:



Label	Type
Untitled (Editor)	World
Atmospheric Fog	AtmosphericFog
Floor	StaticMeshActor
Light Source	DirectionalLight
MyActor1	Open MyActor
Player Start	PlayerStart
Sky Sphere	Edit BP_Sky_Sphere
SkyLight	SkyLight
SphereReflectionCapture	SphereReflectionCapture

6. If the Output Log tab isn’t already available right to the Content Browser tab, make it so: Left-Click on Window > Developer Tools > Output Log; then Click on the Output Log tab to see its contents.

7. Now, Left-Click on “Compile’’ and wait for compilation to complete; then Left-Clik on “Play” to see the `BeginPlay()` log from “MyActor.cpp”:



```
LogInit: FAudioDevice initialized.
LogSlate: Updating window title bar state: overlay mode, drag disabled, window buttons hidden, title bar hidden
LogLoad: Game class is 'GameModeBase'
LogWorld: Bringing World /Temp/UEDPIE_0_Untitled_1.Untitled_1 up for play (max tick rate 60) at 2020.04.10-11.35.24
LogWorld: Bringing up level for play took: 0.000721
LogOnline: OSS: Creating online subsystem instance for: :Context_2
MyLogHTNPlanner: Warning: ==> Beginning the test of the HTN Plugin
PIE: Play in editor start time for /Temp/UEDPIE_0_Untitled_1 0,843
LogSlate: Window 'TestUE4243HTNPlanner Game Preview Standalone (64-bit/PCD3D_SM5)' being destroyed
LogWorld: BeginTearingDown for /Temp/UEDPIE_0_Untitled_1
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogWorld: UWorld::CleanupWorld for Untitled_1, bSessionEnded=true, bCleanupResources=true
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogPlayLevel: Display: Shutting down PIE online subsystems
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogUObjectHash: Compacting FUObjectHashTables data took   1.66ms
LogPlayLevel: Display: Destroying online subsystem :Context_2
```

Subsection 2.2 : (Empty) Test of the HTN Planner

We’re now to make an empty call to the planner and log this call.

1. In Visual Studio, add the following inclusion of “. h” files to “MyActor.cpp” in order to enable the declaration and the building of a planning domain, and the call to the planner:

```
// => HTN Planning includes
#include "HTNDomain.h"
#include "HTNBuilder.h"
#include "HTNPlanner.h"
// <= HTN Planning includes
```

This can start line 6, for instance.

2. Add the following to the body of in “MyActor.cpp”:

```
FHTNBuilder_Domain DomainBuilder;
FHTNWorldState WorldState;
FHTNResult Result;
FHTNPlanner MyHTNPlanner;
```

This can start line 29, for instance.

3. Back to the Unreal Editor to Click on “Compile” and check that the compilation completed successfully; that is, both the inclusion of the HTN Plugin public files and the above delcarations were successful.

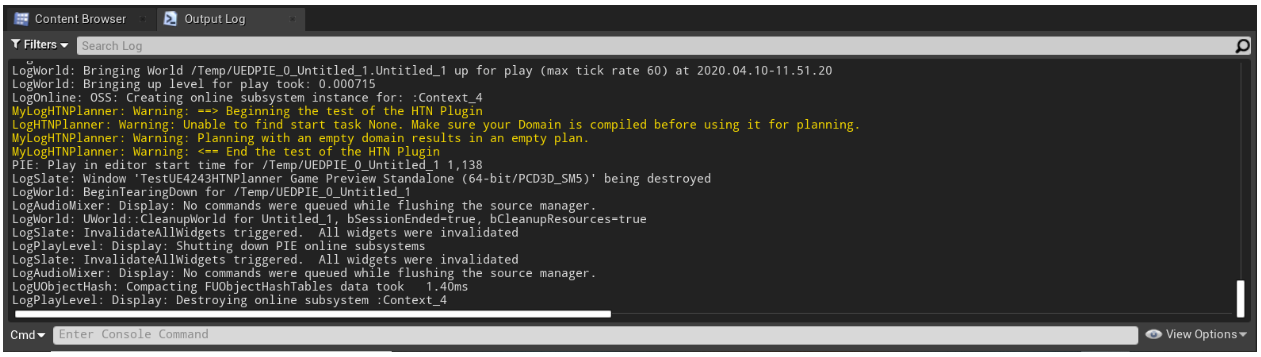
4. Now let’s add an empty call to the HTN Planner to test the above variables; for instance, this can start in line 34 of “MyActor.cpp”:

```
MyHTNPlanner.GeneratePlan(*(DomainBuilder.DomainInstance), WorldState, Result);
if (Result.TaskIDs.Num() == 0)
        UE_LOG(MyLogHTNPlanner, Warning, TEXT("Planning with an empty domain results in an empty plan."))
else
        UE_LOG(MyLogHTNPlanner, Warning, TEXT("Ooops !!!!!"));
```

5. Add the following log to the body of BeginPlay() in “MyActor.cpp”, for instance line 40:

```
UE_LOG(MyLogHTNPlanner, Warning, TEXT("<== End the test of the HTN Plugin"));
```

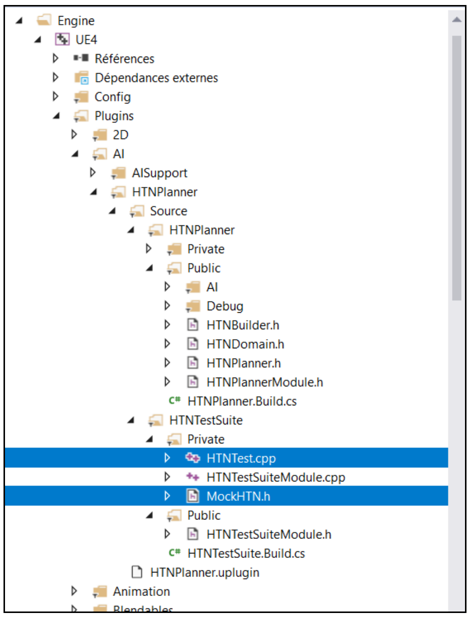
6. Back to the Unreal Editor to Left-Click on “Compile” and check that the compilation completed successfully; then Left-Click on “Play” and check the logs:



```
LogWorld: Bringing World /Temp/UEDPIE_0_Untitled_1.Untitled_1 up for play (max tick rate 60) at 2020.04.10-11.51.20
LogWorld: Bringing up level for play took: 0.000715
LogOnline: OSS: Creating online subsystem instance for: :Context_4
MyLogHTNPlanner: Warning: ==> Beginning the test of the HTN Plugin
LogHTNPlanner: Warning: Unable to find start task None. Make sure your Domain is compiled before using it for planning.
MyLogHTNPlanner: Warning: Planning with an empty domain results in an empty plan.
MyLogHTNPlanner: Warning: <== End the test of the HTN Plugin
PIE: Play in editor start time for /Temp/UEDPIE_0_Untitled_1 1,138
LogSlate: Window 'TestUE4243HTNPlanner Game Preview Standalone (64-bit/PCD3D_SM5)' being destroyed
LogWorld: BeginTearingDown for /Temp/UEDPIE_0_Untitled_1
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogWorld: UWorld::CleanupWorld for Untitled_1, bSessionEnded=true, bCleanupResources=true
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogPlayLevel: Display: Shutting down PIE online subsystems
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogUObjectHash: Compacting FUObjectHashTables data took   1.40ms
LogPlayLevel: Display: Destroying online subsystem :Context_4
```

Subsection 2.3: Testing a simple planning domain

1. Give a look at the test code which comes with the HTN Planner Plugin; for instance, give a look (or read, or study) files “MokHTN.h” and “HTNTest.cpp”:



```
Engine
  UE4
    Références
    Dépendances externes
    Config
    Plugins
      2D
      AI
        AISupport
        HTNPlanner
          Source
            HTNPlanner
              Private
              Public
                AI
                Debug
                HTNBuilder.h
                HTNDomain.h
                HTNPlanner.h
                HTNPlannerModule.h
              HTNPlanner.Build.cs
            HTNTestSuite
              Private
                HTNTest.cpp
                HTNTestSuiteModule.cpp
                MockHTN.h
              Public
                HTNTestSuiteModule.h
              HTNTestSuite.Build.cs
          HTNPlanner.uplugin
      Animation
      Blendables
```

2. Copy lines 11 to 36 from “MokHTN.h” and paste them in “MyActor.cpp”, for instance starting line 14:

```
enum class EMockHTNWorldState : uint8
{
	EnemyHealth,
	EnemyActor,
	Ammo,
	AbilityRange,
	HasWeapon,
	MoveDestination,
	PickupLocation,
	CurrentLocation,
	CanSeeEnemy,

	MAX
};

enum class EMockHTNTaskOperator : uint8
{
	DummyOperation,
	FindPatrolPoint,
	FindWeapon,
	NavigateTo,
	PickUp,
	UseWeapon,

	MAX
};
```

3. Copy lines 39 à 106 from “HTNTest.cpp’’ and paste them in the body of `BeginPlay()` in “MyActor.cpp”, for instance starting line 69:

```
DomainBuilder.SetRootName(TEXT("Root"));
{
	FHTNBuilder_CompositeTask& CompositeTaskBuilder = DomainBuilder.AddCompositeTask(TEXT("Root"));
	{
		FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod(
			TArray<FHTNCondition>({
			FHTNCondition(EMockHTNWorldState::EnemyHealth, EHTNWorldStateCheck::Greater).SetRHSAsValue(0)
			, FHTNCondition(EMockHTNWorldState::EnemyActor, EHTNWorldStateCheck::IsTrue)
		}));
		MethodsBuilder.AddTask(TEXT("AttackEnemy"));
	}
	{
		FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod();
		MethodsBuilder.AddTask(TEXT("FindPatrolPoint"));
		MethodsBuilder.AddTask(TEXT("NavigateToMoveDestination"));
	}
}
{
	FHTNBuilder_CompositeTask& CompositeTaskBuilder = DomainBuilder.AddCompositeTask(TEXT("AttackEnemy"));
```

```
			{
				FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod(FHTNCondition(EMockHTNWorldState::HasWeapon, 
EHTNWorldStateCheck::IsTrue));
				MethodsBuilder.AddTask(TEXT("NavigateToEnemy"));
				MethodsBuilder.AddTask(TEXT("UseWeapon"));
				MethodsBuilder.AddTask(TEXT("Root"));
			}
			{
				FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod();
				MethodsBuilder.AddTask(TEXT("FindWeapon"));
				MethodsBuilder.AddTask(TEXT("NavigateToWeapon"));
				MethodsBuilder.AddTask(TEXT("PickUp"));
				MethodsBuilder.AddTask(TEXT("AttackEnemy"));
			}
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("FindPatrolPoint"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::FindPatrolPoint, EMockHTNWorldState::MoveDestination);
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("FindWeapon"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::FindWeapon, EMockHTNWorldState::PickupLocation);
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("NavigateToMoveDestination"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, EMockHTNWorldState::MoveDestination);	// Local Variables?
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::MoveDestination));
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("NavigateToEnemy"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, EMockHTNWorldState::EnemyActor);
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::EnemyActor));
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CanSeeEnemy, EHTNWorldStateOperation::Set).SetRHSAsValue(1));
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("NavigateToWeapon"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, EMockHTNWorldState::PickupLocation);
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::PickupLocation));
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("PickUp"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::PickUp, EMockHTNWorldState::PickupLocation);
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::HasWeapon, EHTNWorldStateOperation::Set).SetRHSAsValue(1));
		}
		{
			FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("UseWeapon"));
			PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::UseWeapon, EMockHTNWorldState::EnemyActor);
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::Ammo, EHTNWorldStateOperation::Decrease).SetRHSAsValue(1));
			PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::EnemyHealth, EHTNWorldStateOperation::Decrease).SetRHSAsValue(1));
		}
```

4. Following the previous insertion, add the following lines (the if can be found line 108 to 111 in "HTNTest.cpp") to the body of `BeginPlay()` in "MyActor.cpp", for instance starting line 138:

```
bool bCompile = true;
if (bCompile)
{
	DomainBuilder.Compile();
}
```

5. Now add the following call to the HTN Planner, for instance starting line 144 in the body of `BeginPlay()` in "MyActor.cpp":

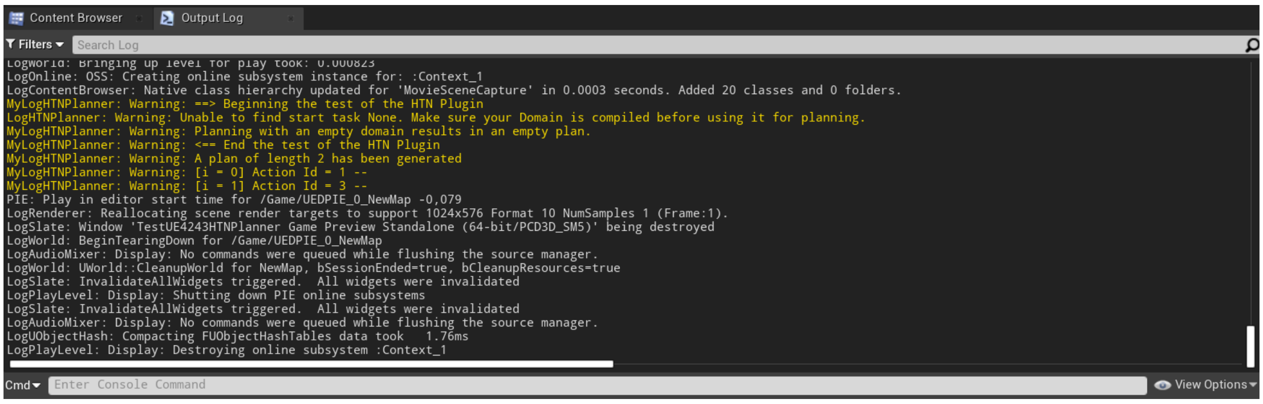
```
MyHTNPlanner.GeneratePlan(*(DomainBuilder.DomainInstance), WorldState, Result);
```

6. Eventually insert a log of the length of the plan and its actions, for instance starting line 146 in the body of `BeginPlay()` in "MyActor.cpp":

```
UE_LOG(MyLogHTNPlanner, Warning, TEXT("A plan of length %d has been generated"), Result.TaskIDs.Num())

	for (int i = 0; i < Result.TaskIDs.Num(); ++i)
	{
		UE_LOG(MyLogHTNPlanner, Warning, TEXT("[i = %d] Action Id = %d -- "), i, 
Result.ActionsSequence[i].ActionID)
	}
```

7. Back to the Unreal Editor, Left-Click on "Compile" (which should result in a successful compilation) and then "Play" to check the following logs:



```
LogWorld: Bringing up level for play took: 0.000823
LogOnline: OSS: Creating online subsystem instance for: :Context_1
LogContentBrowser: Native class hierarchy updated for 'MovieSceneCapture' in 0.0003 seconds. Added 20 classes and 0 folders.
MyLogHTNPlanner: Warning: ==> Beginning the test of the HTN Plugin
LogHTNPlanner: Warning: Unable to find start task None. Make sure your Domain is compiled before using it for planning.
MyLogHTNPlanner: Warning: Planning with an empty domain results in an empty plan.
MyLogHTNPlanner: Warning: <== End the test of the HTN Plugin
MyLogHTNPlanner: Warning: A plan of length 2 has been generated
MyLogHTNPlanner: Warning: [i = 0] Action Id = 1 --
MyLogHTNPlanner: Warning: [i = 1] Action Id = 3 --
PIE: Play in editor start time for /Game/UEDPIE_0_NewMap -0,079
LogRenderer: Reallocating scene render targets to support 1024x576 Format 10 NumSamples 1 (Frame:1).
LogSlate: Window 'TestUE4243HTNPlanner Game Preview Standalone (64-bit/PCD3D_SM5)' being destroyed
LogWorld: BeginTearingDown for /Game/UEDPIE_0_NewMap
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogWorld: UWorld::CleanupWorld for NewMap, bSessionEnded=true, bCleanupResources=true
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogPlayLevel: Display: Shutting down PIE online subsystems
LogSlate: InvalidateAllWidgets triggered.  All widgets were invalidated
LogAudioMixer: Display: No commands were queued while flushing the source manager.
LogUObjectHash: Compacting FUObjectHashTables data took   1.76ms
LogPlayLevel: Display: Destroying online subsystem :Context_1
```

8. That’s All Folks!

<u>Appendix 1:</u> “MyActor.h”

```
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "MyActor.generated.h"

DECLARE_LOG_CATEGORY_EXTERN(MyLogHTNPlanner, Log, All);

UCLASS()
class TESTUE4243HTNPLANNER_API AMyActor : public AActor
{
	GENERATED_BODY()

public:
	// Sets default values for this actor's properties
	AMyActor();

protected:
	// Called when the game starts or when spawned
	virtual void BeginPlay() override;

public:
	// Called every frame
	virtual void Tick(float DeltaTime) override;

};
```

Appendix 2 : “MyActor.cpp”

```
// Fill out your copyright notice in the Description page of Project Settings.


#include "MyActor.h"

// => HTN Planning includes
#include "HTNDomain.h"
#include "HTNBuilder.h"
#include "HTNPlanner.h"
// <= HTN Planning includes

DEFINE_LOG_CATEGORY(MyLogHTNPlanner);

enum class EMockHTNWorldState : uint8
{
	EnemyHealth,
	EnemyActor,
	Ammo,
	AbilityRange,
	HasWeapon,
	MoveDestination,
	PickupLocation,
	CurrentLocation,
	CanSeeEnemy,

	MAX
};

enum class EMockHTNTaskOperator : uint8
{
	DummyOperation,
	FindPatrolPoint,
	FindWeapon,
	NavigateTo,
	PickUp,
	UseWeapon,

	MAX
};

// Sets default values
AMyActor::AMyActor()
{
	// Set this actor to call Tick() every frame.  You can turn this off to improve performance if you don't need
it.
	PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void AMyActor::BeginPlay()
{
	Super::BeginPlay();

	UE_LOG(MyLogHTNPlanner, Warning, TEXT("==> Beginning the test of the HTN Plugin"));

	FHTNBuilder_Domain DomainBuilder;
	FHTNWorldState WorldState;
	FHTNResult Result;
	FHTNPlanner MyHTNPlanner;

	MyHTNPlanner.GeneratePlan(*(DomainBuilder.DomainInstance), WorldState, Result);
	if (Result.TaskIDs.Num() == 0)
		UE_LOG(MyLogHTNPlanner, Warning, TEXT("Planning with an empty domain results in an empty plan."))
	else
		UE_LOG(MyLogHTNPlanner, Warning, TEXT("Ooops !!!!!"));

	UE_LOG(MyLogHTNPlanner, Warning, TEXT("<== End the test of the HTN Plugin"));

	DomainBuilder.SetRootName(TEXT("Root"));
	{
		FHTNBuilder_CompositeTask& CompositeTaskBuilder = DomainBuilder.AddCompositeTask(TEXT("Root"));
		{
			FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod(
				TArray<FHTNCondition>({
				FHTNCondition(EMockHTNWorldState::EnemyHealth, 
EHTNWorldStateCheck::Greater).SetRHSAsValue(0)
				, FHTNCondition(EMockHTNWorldState::EnemyActor, EHTNWorldStateCheck::IsTrue)
					}));
			MethodsBuilder.AddTask(TEXT("AttackEnemy"));
		}
		{
			FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod();
			MethodsBuilder.AddTask(TEXT("FindPatrolPoint"));
			MethodsBuilder.AddTask(TEXT("NavigateToMoveDestination"));
		}
	}
	{
		FHTNBuilder_CompositeTask& CompositeTaskBuilder = DomainBuilder.AddCompositeTask(TEXT("AttackEnemy"));
		{
			FHTNBuilder_Method& MethodsBuilder = 
CompositeTaskBuilder.AddMethod(FHTNCondition(EMockHTNWorldState::HasWeapon, EHTNWorldStateCheck::IsTrue));
			MethodsBuilder.AddTask(TEXT("NavigateToEnemy"));
			MethodsBuilder.AddTask(TEXT("UseWeapon"));
			MethodsBuilder.AddTask(TEXT("Root"));
		}
		{
			FHTNBuilder_Method& MethodsBuilder = CompositeTaskBuilder.AddMethod();
			MethodsBuilder.AddTask(TEXT("FindWeapon"));
			MethodsBuilder.AddTask(TEXT("NavigateToWeapon"));
			MethodsBuilder.AddTask(TEXT("PickUp"));
			MethodsBuilder.AddTask(TEXT("AttackEnemy"));
		}
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = 
DomainBuilder.AddPrimitiveTask(TEXT("FindPatrolPoint"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::FindPatrolPoint, 
EMockHTNWorldState::MoveDestination);
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("FindWeapon"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::FindWeapon, 
EMockHTNWorldState::PickupLocation);
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = 
DomainBuilder.AddPrimitiveTask(TEXT("NavigateToMoveDestination"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, 
EMockHTNWorldState::MoveDestination);	// Local Variables?
		PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::MoveDestination));
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = 
DomainBuilder.AddPrimitiveTask(TEXT("NavigateToEnemy"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, EMockHTNWorldState::EnemyActor);
		PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::EnemyActor));
		PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CanSeeEnemy, 
EHTNWorldStateOperation::Set).SetRHSAsValue(1));
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = 
DomainBuilder.AddPrimitiveTask(TEXT("NavigateToWeapon"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::NavigateTo, 
EMockHTNWorldState::PickupLocation);
		PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::CurrentLocation, 
EHTNWorldStateOperation::Set).SetRHSAsWSKey(EMockHTNWorldState::PickupLocation));
	}
	{
		FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("PickUp"));
		PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::PickUp, EMockHTNWorldState::PickupLocation);
		PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::HasWeapon, 
EHTNWorldStateOperation::Set).SetRHSAsValue(1));
```

```
        }
        {
                FHTNBuilder_PrimitiveTask& PrimitiveTaskBuilder = DomainBuilder.AddPrimitiveTask(TEXT("UseWeapon"));
                PrimitiveTaskBuilder.SetOperator(EMockHTNTaskOperator::UseWeapon, EMockHTNWorldState::EnemyActor);
                PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::Ammo,
EHTNWorldStateOperation::Decrease).SetRHSAsValue(1));
                PrimitiveTaskBuilder.AddEffect(FHTNEffect(EMockHTNWorldState::EnemyHealth,
EHTNWorldStateOperation::Decrease).SetRHSAsValue(1));
        }

        bool bCompile = true;
        if (bCompile)
        {
                DomainBuilder.Compile();
        }

        MyHTNPlanner.GeneratePlan(*(DomainBuilder.DomainInstance), WorldState, Result);

        UE_LOG(MyLogHTNPlanner, Warning, TEXT("A plan of length %d has been generated"), Result.TaskIDs.Num())

                for (int i = 0; i < Result.TaskIDs.Num(); ++i)
                {
                        UE_LOG(MyLogHTNPlanner, Warning, TEXT("[i = %d] Action Id = %d -- "), i,
Result.ActionsSequence[i].ActionID)
                }

}

// Called every frame
void AMyActor::Tick(float DeltaTime)
{
        Super::Tick(DeltaTime);

}
```