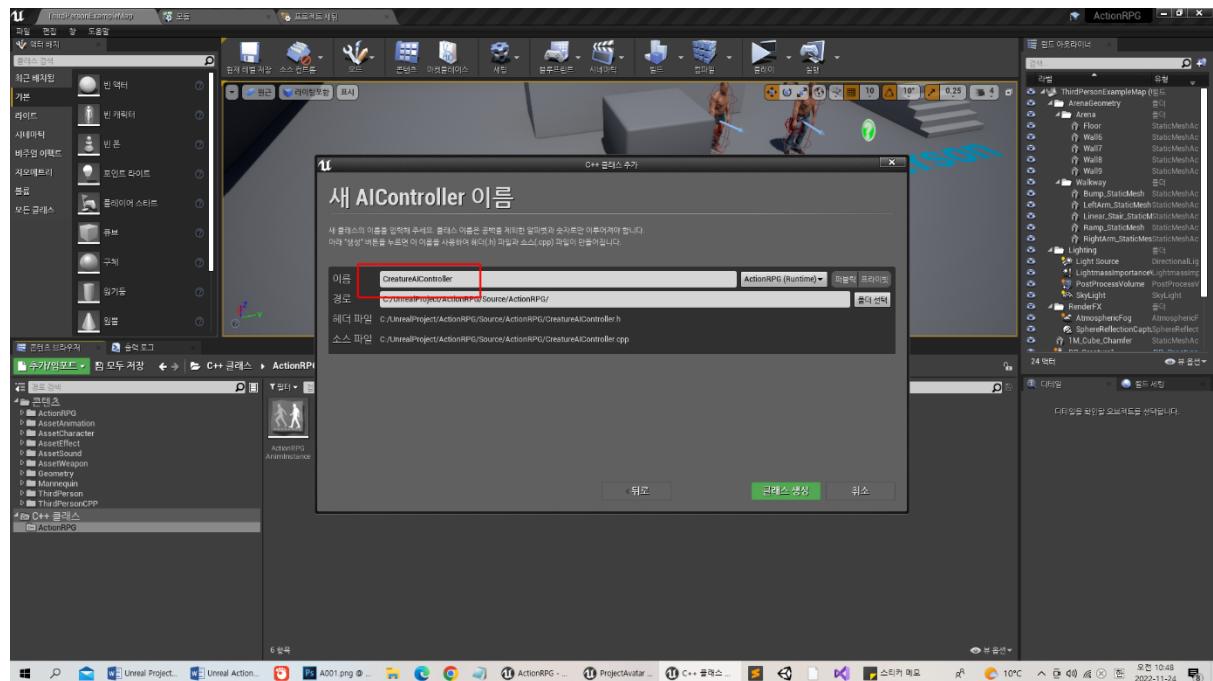
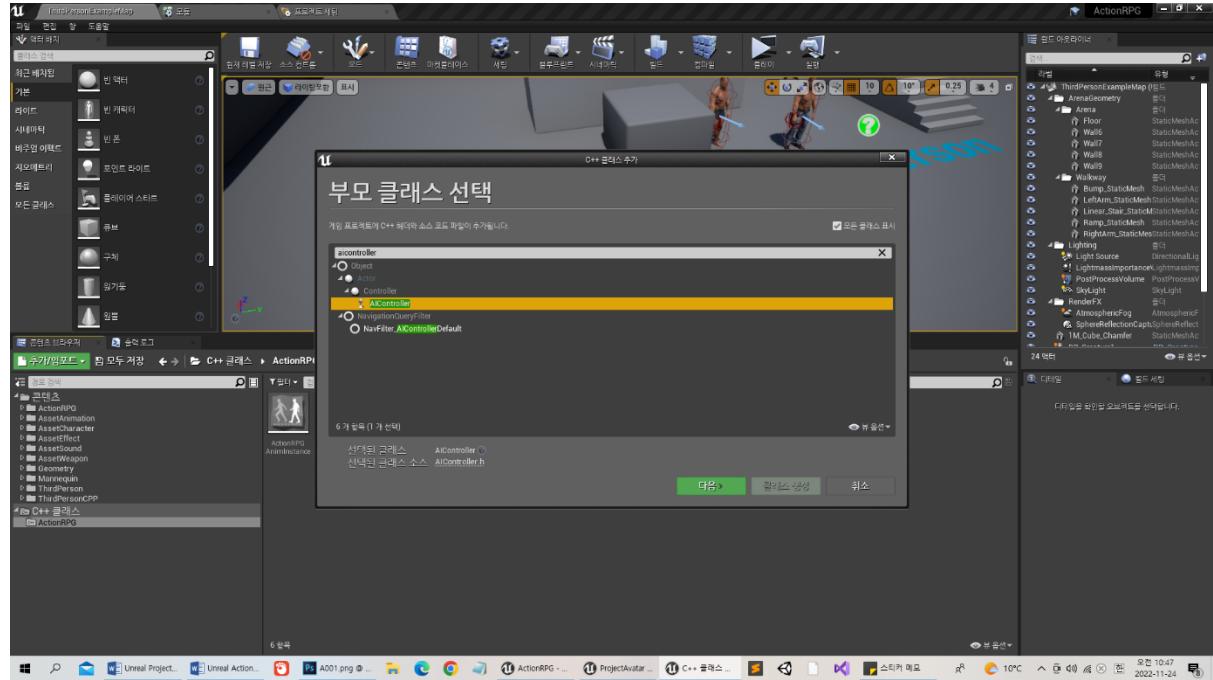
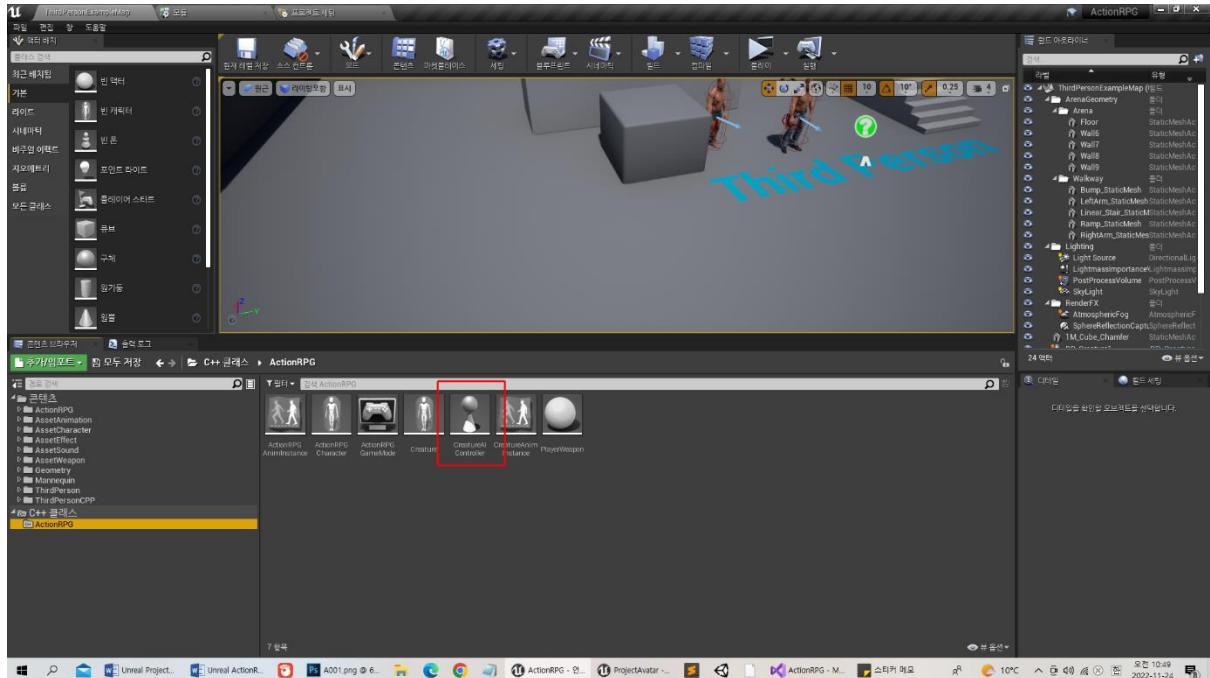


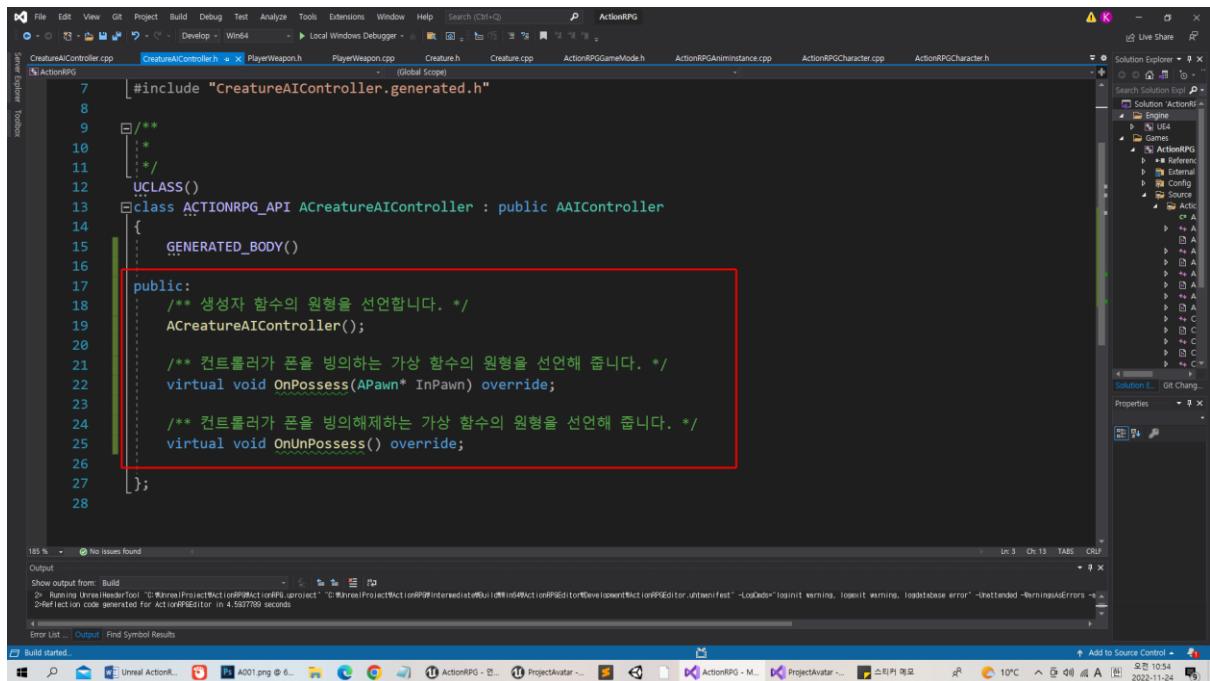
이번 시간에는 인공지능에 대해서 알아 보도록 합니다.

AIController를 부모 클래스로 상속받는 CreatureAIController이라는 이름의 클래스를 정의해 주도록 합니다.

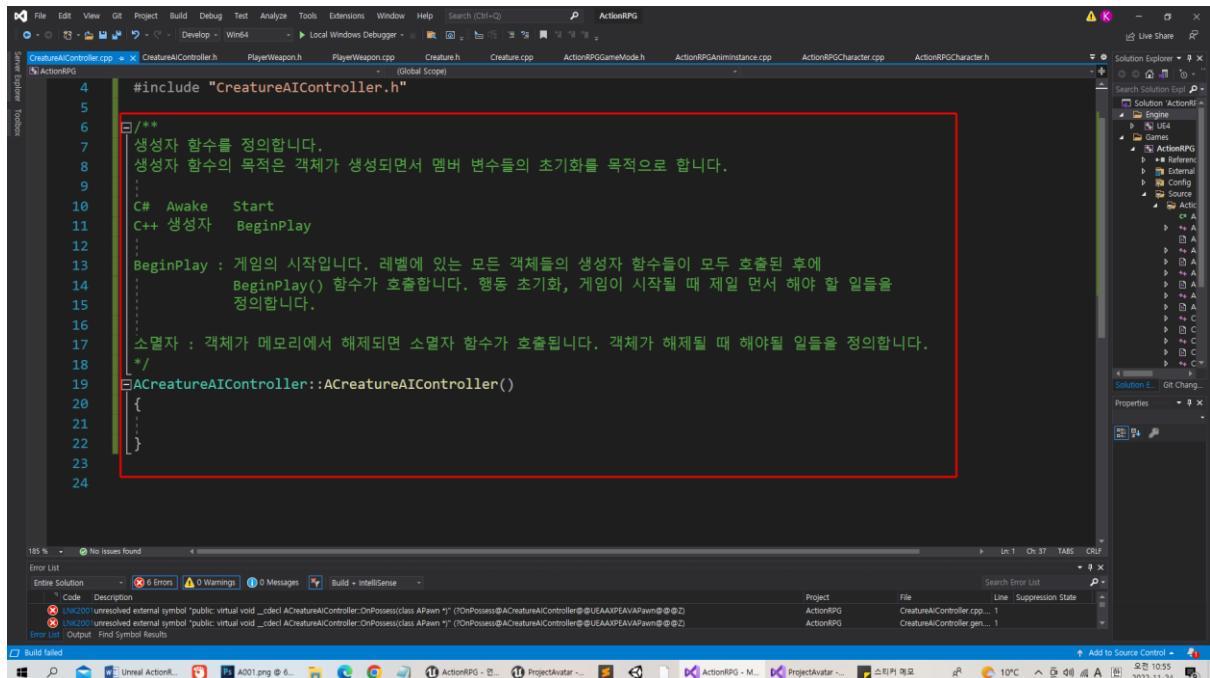




필요한 멤버 함수의 원형을 선언합니다.



구현해 줍니다.



```
4 #include "CreatureAIController.h"
5
6 /**
7  * 생성자 함수를 정의합니다.
8  * 생성자 함수의 목적은 객체가 생성되면서 멤버 변수들의 초기화를 목적으로 합니다.
9
10 C# Awake Start
11 C++ 생성자 BeginPlay
12
13 BeginPlay : 게임의 시작입니다. 레벨에 있는 모든 객체들의 생성자 함수들이 모두 호출된 후에
14     BeginPlay() 함수가 호출합니다. 행동 초기화, 게임이 시작될 때 제일 먼저 해야 할 일들을
15     정의합니다.
16
17 소멸자 : 객체가 메모리에서 해제되면 소멸자 함수가 호출됩니다. 객체가 해제될 때 해야 할 일들을 정의합니다.
18 */
19 ACreatureAIController::ACreatureAIController()
20 {
21
22 }
```

100% No issues found

Error List

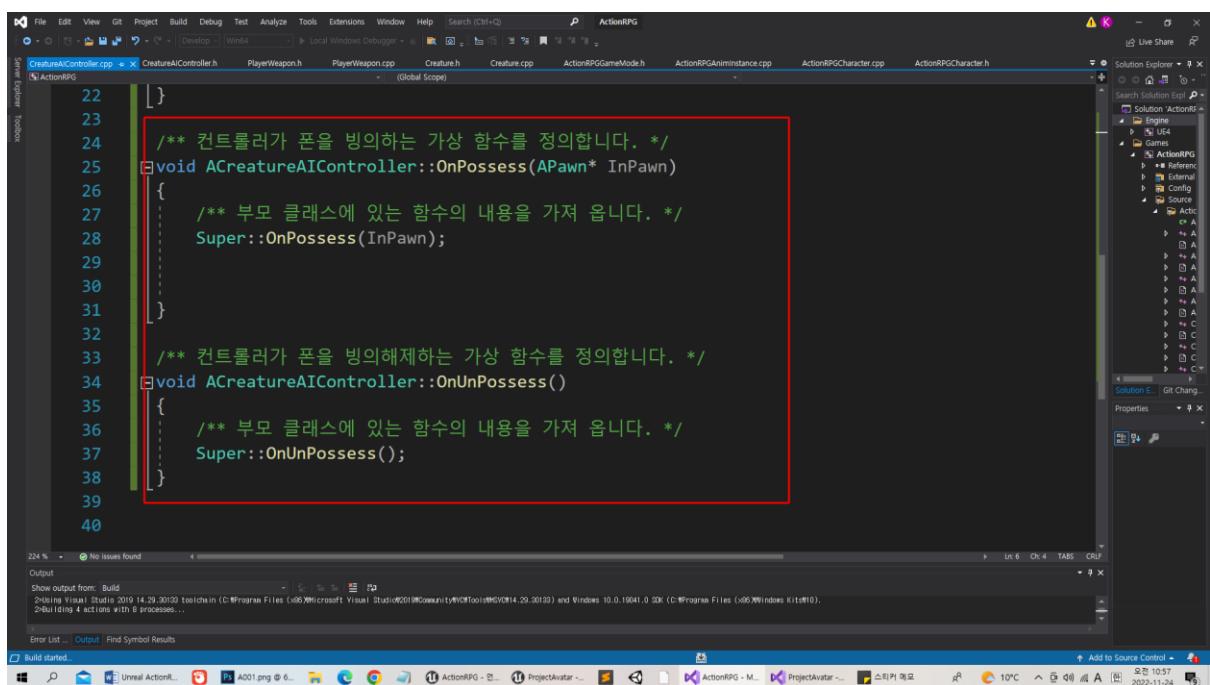
Code Description	Project	File	Line	Suppression State
UNK001 unresolved external symbol "public: virtual void __cdecl ACreatureAIController::OnPossess(class APawn *)" (?OnPossess@ACreatureAIController@@UEAAXPEAVAPawn@@@Z)	ActionRPG	CreatureAIController.cpp	1	
UNK002 unresolved external symbol "public: virtual void __cdecl ACreatureAIController::OnUnpossess(class APawn *)" (?OnUnpossess@ACreatureAIController@@UEAAXPEAVAPawn@@@Z)	ActionRPG	CreatureAIController.gen...	1	

Output

Build failed

Build output: Unreal ActionRPG

10:55 10°C 2022-11-24



```
22 }
23
24 /**
25  * 컨트롤러가 폰을 빙의하는 가상 함수를 정의합니다. */
26 void ACreatureAIController::OnPossess(APawn* InPawn)
27 {
28     /**
29      * 부모 클래스에 있는 함수의 내용을 가져옵니다.
30      */
31     Super::OnPossess(InPawn);
32
33 /**
34  * 컨트롤러가 폰을 빙의해제하는 가상 함수를 정의합니다. */
35 void ACreatureAIController::OnUnpossess()
36 {
37     /**
38      * 부모 클래스에 있는 함수의 내용을 가져옵니다.
39      */
40     Super::OnUnpossess();
41 }
```

224% No issues found

Output

Show output from: Build

Starting Visual Studio 2019 14.29.26526.101 chain (C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30130) and Windows 10.0.19041.0.0 SDK (C:\Program Files (x86)\Windows Kits\10).

Building 4 actions with 0 processes...

Error List

Output

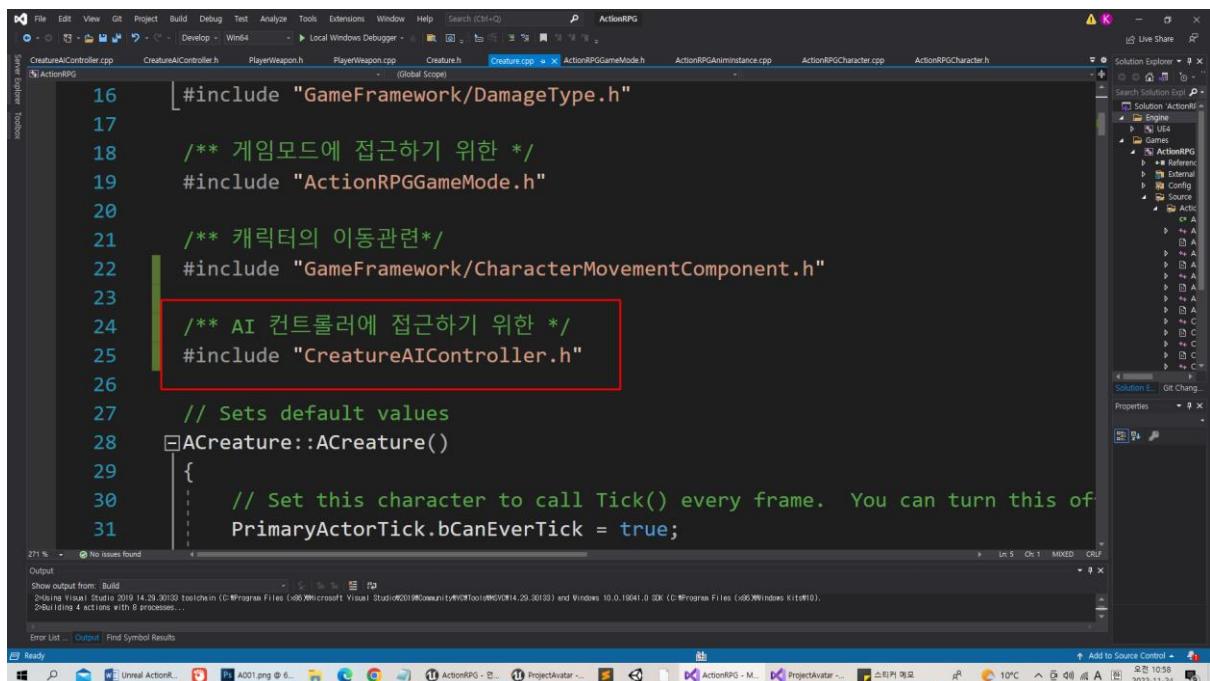
Find Symbol Results

Build started.

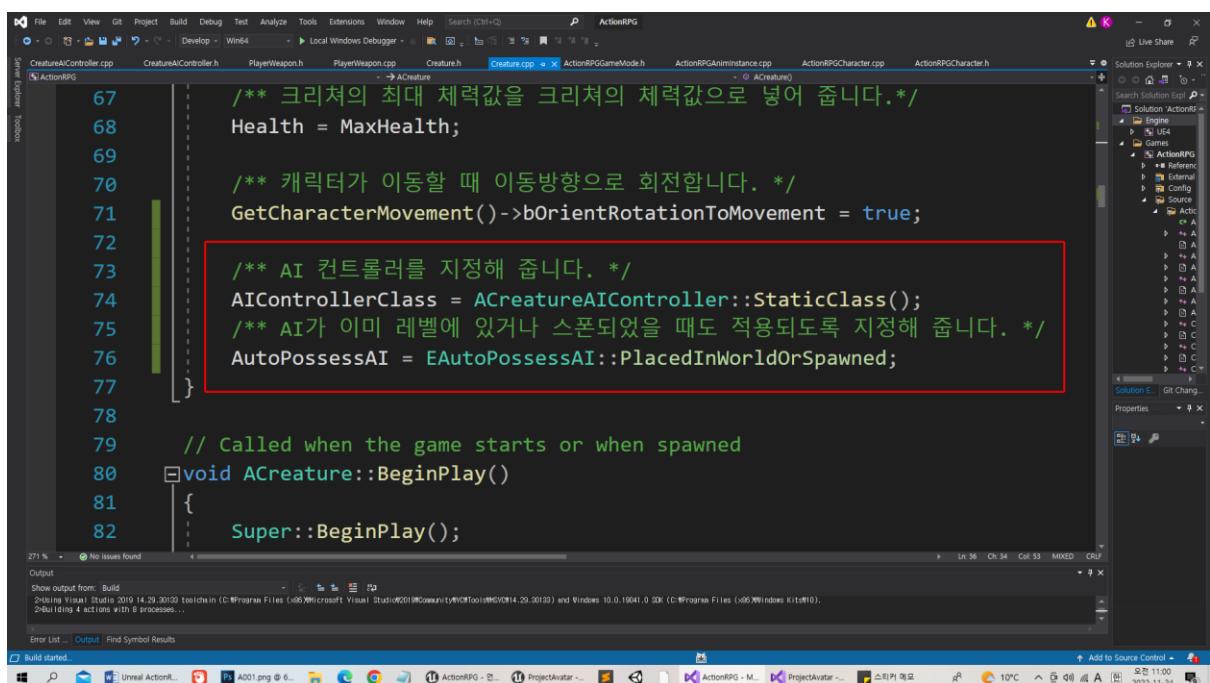
Build output: Unreal ActionRPG

10:57 10°C 2022-11-24

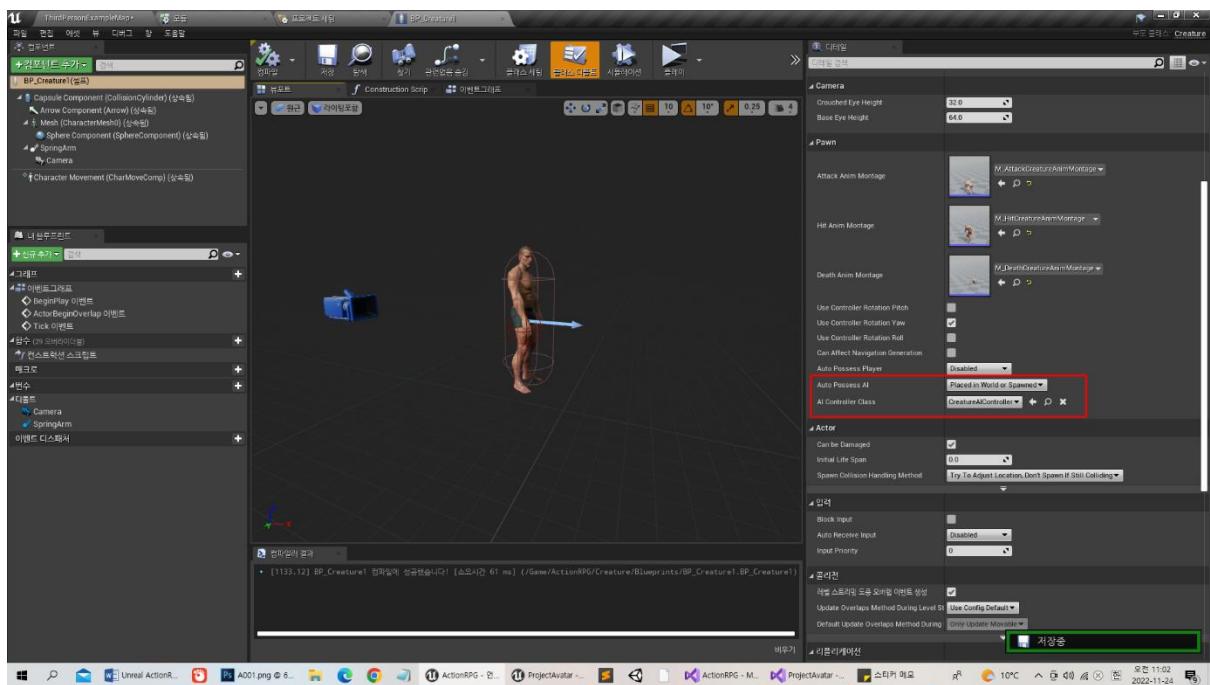
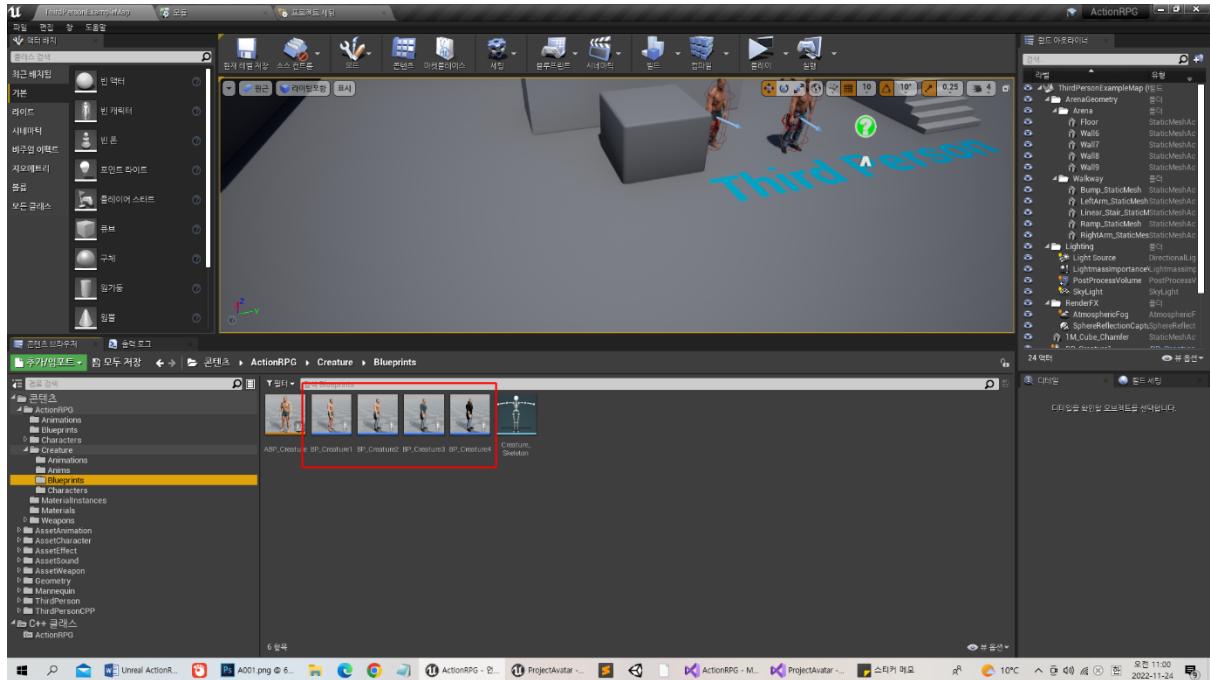
Creature 클래스에서 AIController를 적용해 주도록 합니다.



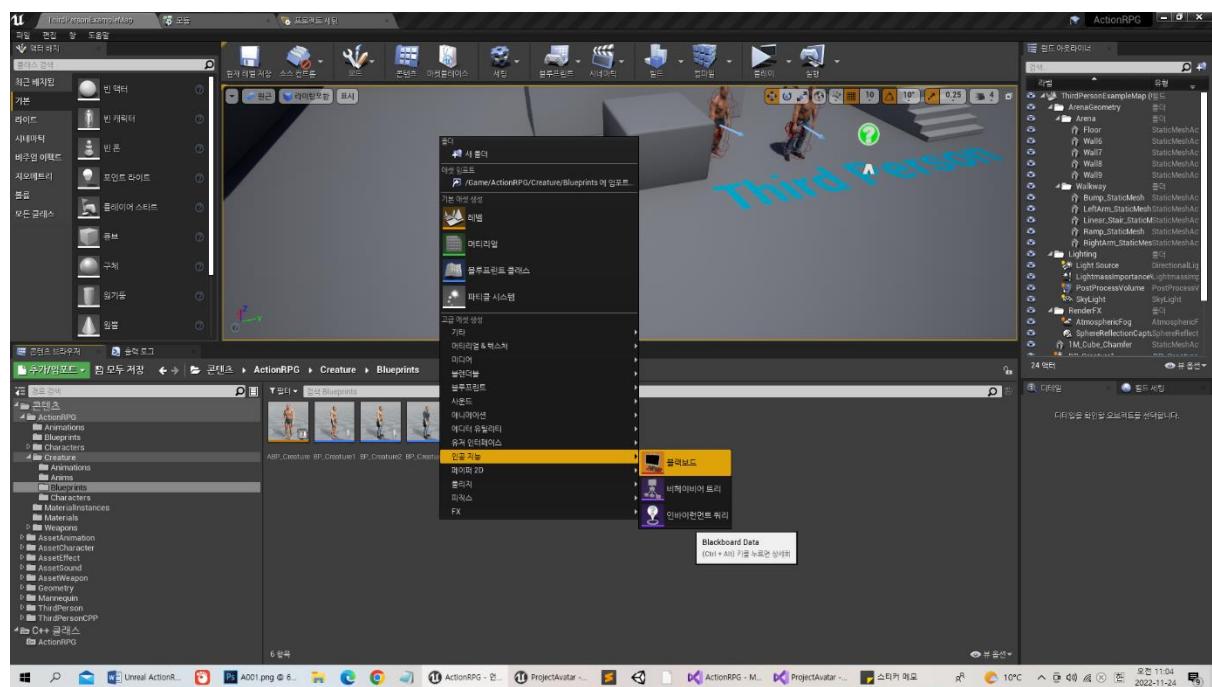
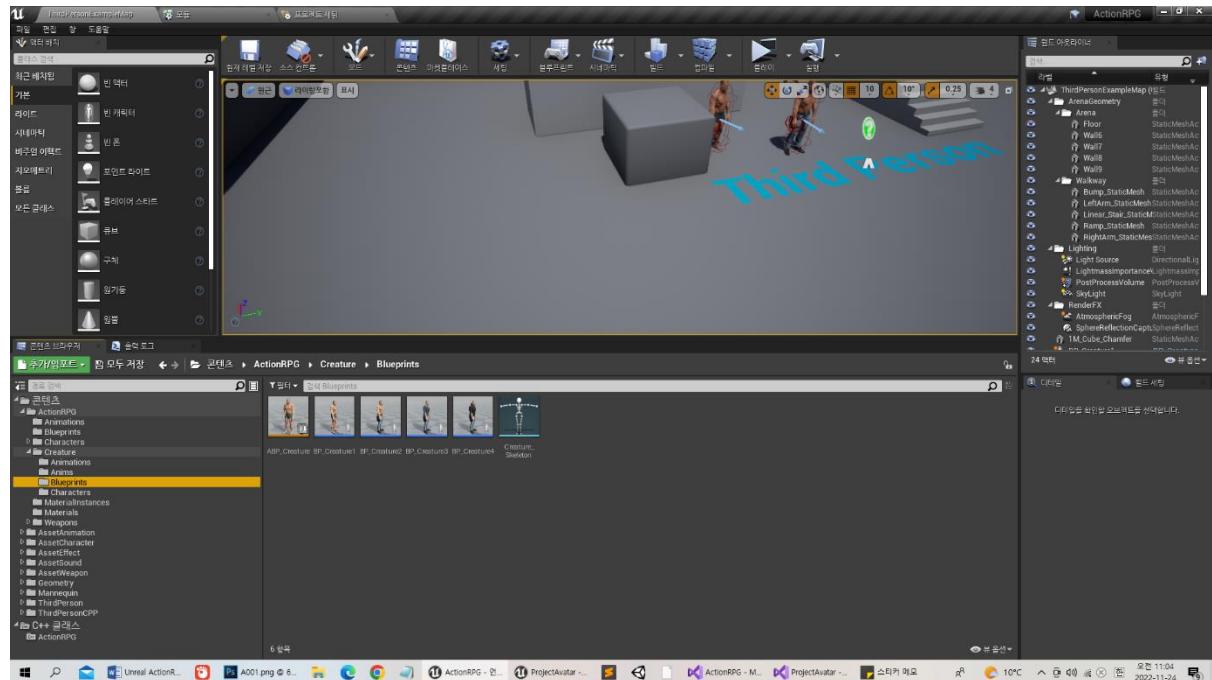
```
16 #include "GameFramework/DamageType.h"
17
18 /** 게임모드에 접근하기 위한 */
19 #include "ActionRPGGameMode.h"
20
21 /** 캐릭터의 이동관련*/
22 #include "GameFramework/CharacterMovementComponent.h"
23
24 /** AI 컨트롤러에 접근하기 위한 */
25 #include "CreatureAIController.h"
26
27 // Sets default values
28 ACreature::ACreature()
29 {
30     // Set this character to call Tick() every frame. You can turn this off
31     PrimaryActorTick.bCanEverTick = true;
32 }
33
34 // Called when the game starts or when spawned
35 void ACreature::BeginPlay()
36 {
37     Super::BeginPlay();
38 }
```

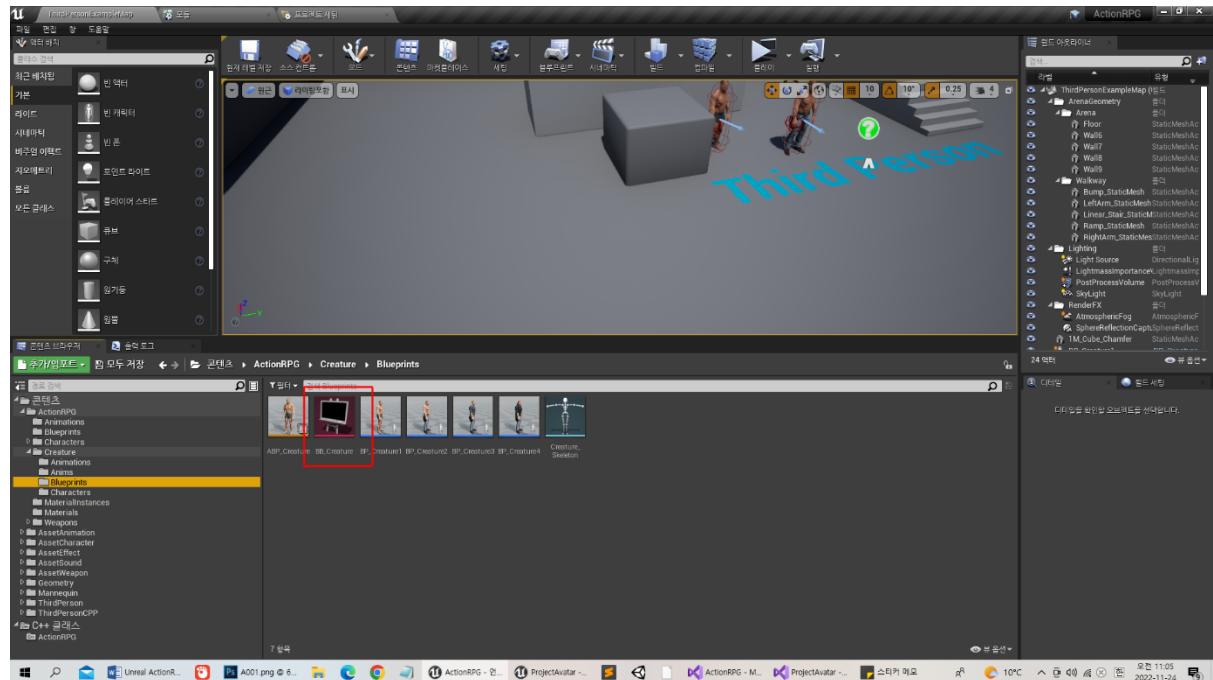


```
67 /**
68  * 크리쳐의 최대 체력값을 크리쳐의 체력값으로 넣어 줍니다.
69  */
70 Health = MaxHealth;
71
72 /**
73  * 캐릭터가 이동할 때 이동방향으로 회전합니다.
74  */
75 GetCharacterMovement()->bOrientRotationToMovement = true;
76
77 /**
78  * AI 컨트롤러를 지정해 줍니다.
79  */
80 AIControllerClass = ACreatureAIController::StaticClass();
81
82 /**
83  * AI가 이미 레벨에 있거나 스폰되었을 때도 적용되도록 지정해 줍니다.
84  */
85 AutoPossessAI = EAutoPossessAI::PlacedInWorldOrSpawned;
86
87
88 /**
89  * Called when the game starts or when spawned
90  */
91 void ACreature::BeginPlay()
92 {
93     Super::BeginPlay();
94 }
```

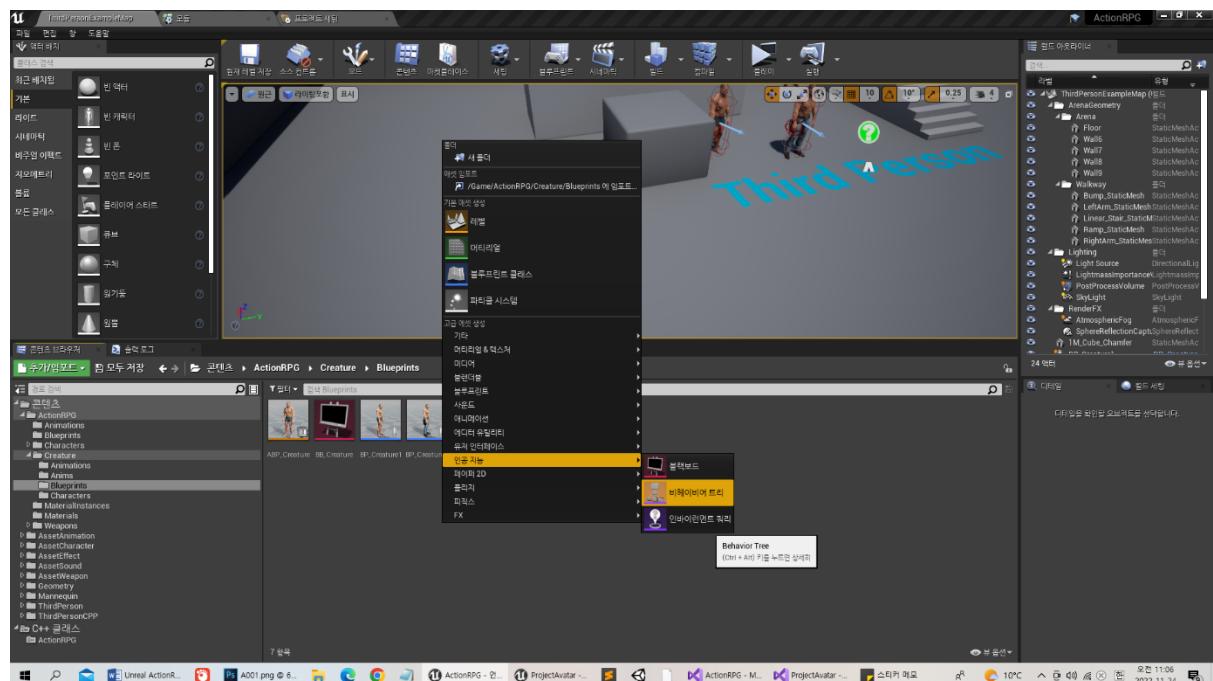


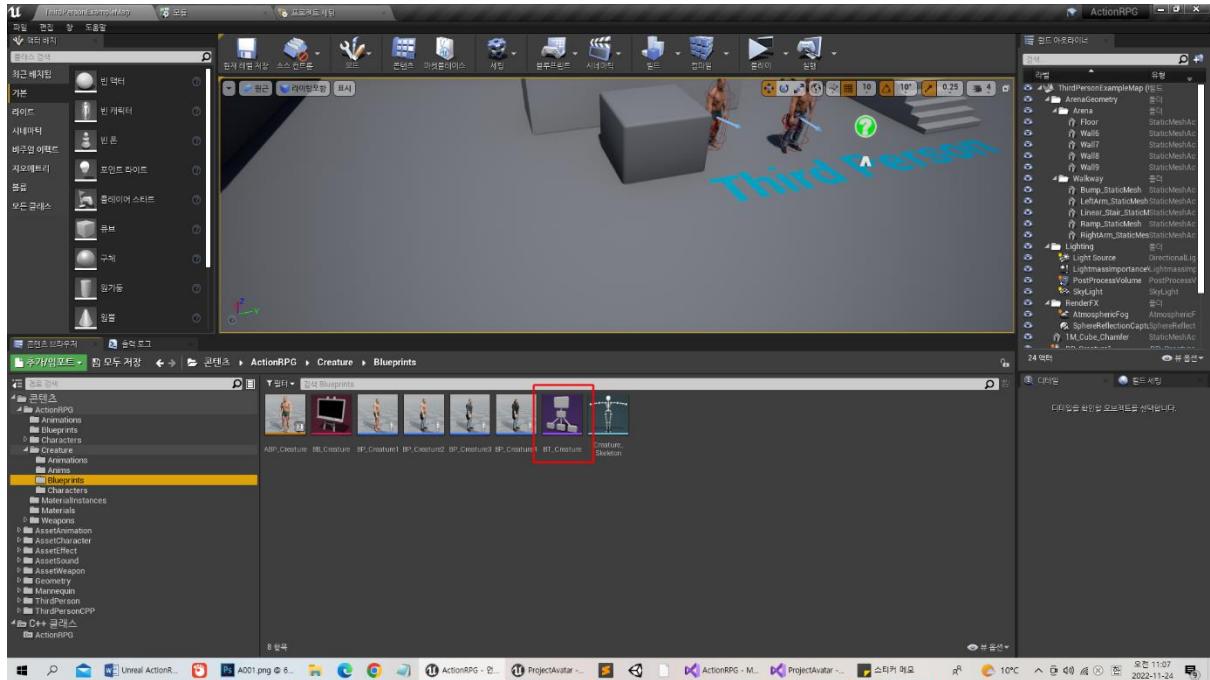
이제 AI의 두뇌에 해당하는 블랙보드를 정의해 주도록 합니다. 이름을 BB_Creature라고 지정해 줍니다.



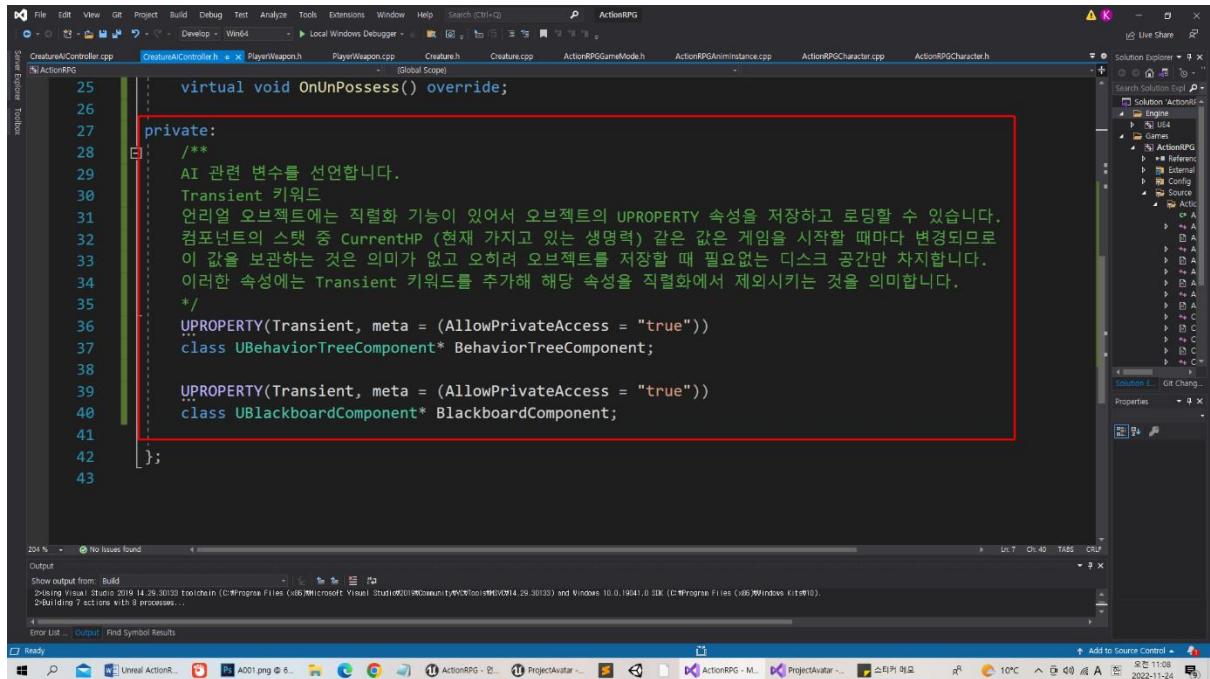


AI를 트리 구조로 관리해주는 BehaviorTree를 정의해 주도록 합니다. 이름을 BT_Creature라고 지정해 줍니다.

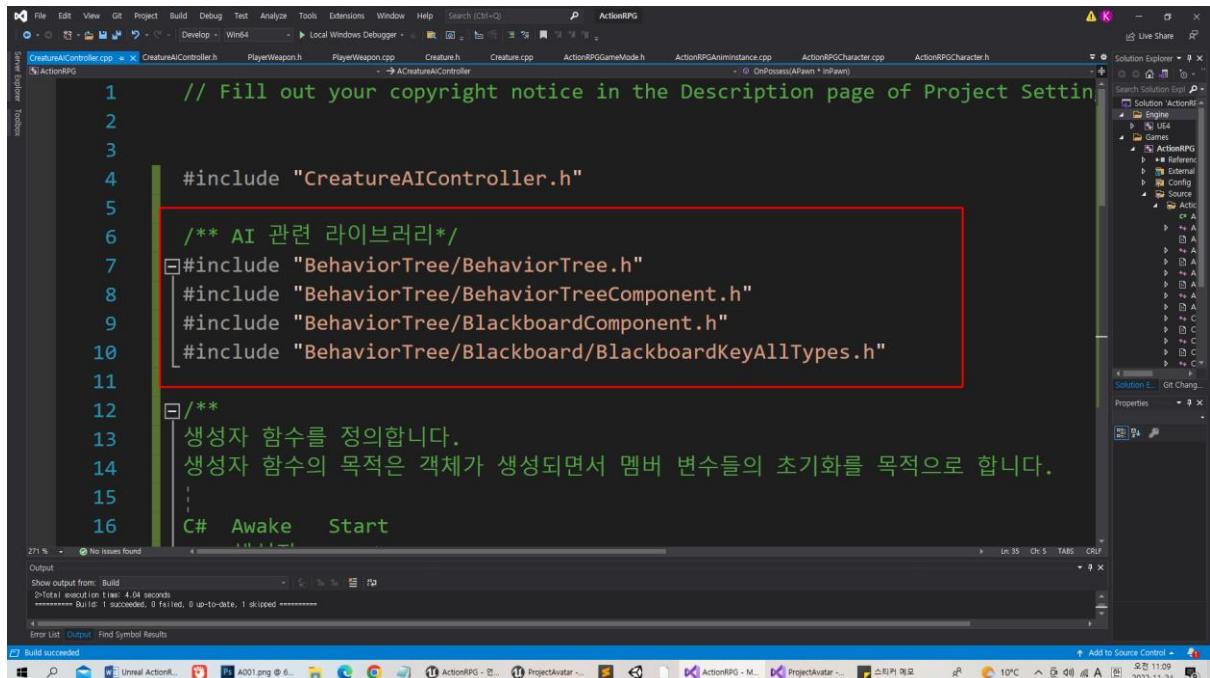




AIController 헤더에서 BehaviorTree와 BlackBoard를 저장할 변수를 선언합니다.



생성자 함수에서 생성해 주도록 합니다.



```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3
4 #include "CreatureAIController.h"
5
6 /**
7  * AI 관련 라이브러리/
8 #include "BehaviorTree/BehaviorTree.h"
9 #include "BehaviorTree/BehaviorTreeComponent.h"
10 #include "BehaviorTree/BlackboardComponent.h"
11 #include "BehaviorTree/Blackboard/BlackboardKeyAllTypes.h"
12
13 /**
14  * 생성자 함수를 정의합니다.
15  * 생성자 함수의 목적은 객체가 생성되면서 멤버 변수들의 초기화를 목적으로 합니다.
16
17 C# Awake Start
18
19
20
21
22
23 소멸자 : 객체가 메모리에서 해제되면 소멸자 함수가 호출됩니다. 객체가 해제될 때 해야 할 일들을
24 정의합니다.
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

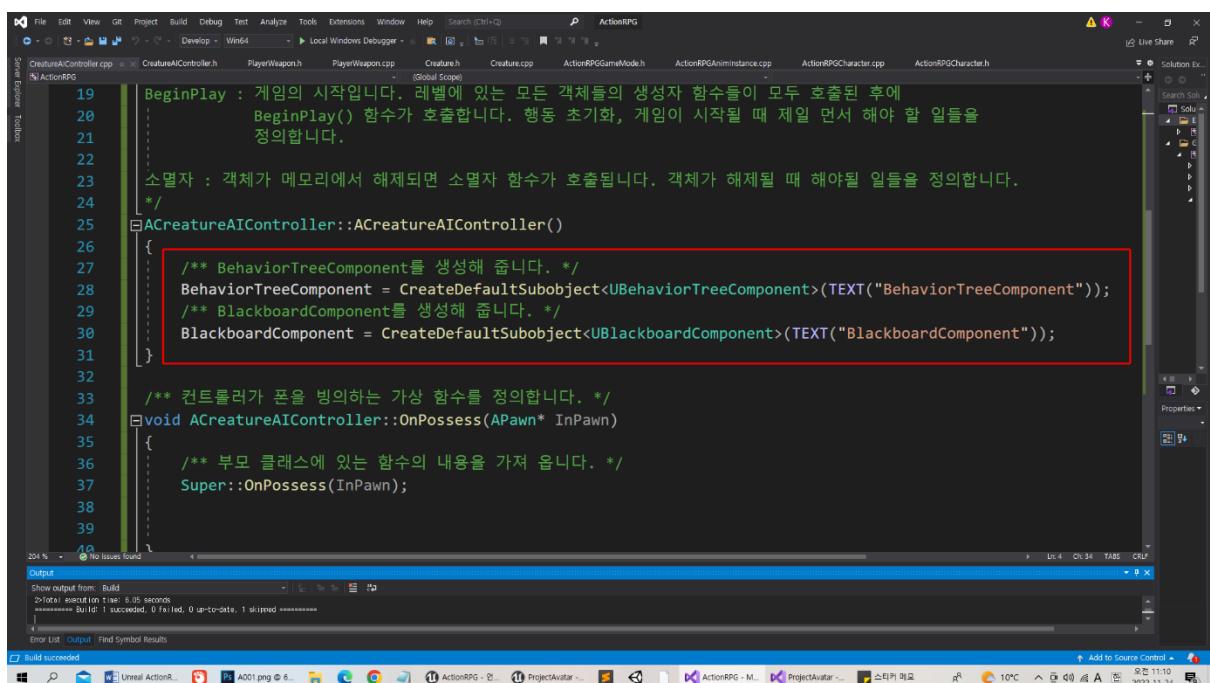
No issues found

Output

Show output from: Build

2023-11-24 11:09 10°C 2023-11-24

Build succeeded



```
19 BeginPlay : 게임의 시작입니다. 레벨에 있는 모든 객체들의 생성자 함수들이 모두 호출된 후에
20 BeginPlay() 함수가 호출합니다. 행동 초기화, 게임이 시작될 때 제일 먼저 해야 할 일들을
21 정의합니다.
22
23 소멸자 : 객체가 메모리에서 해제되면 소멸자 함수가 호출됩니다. 객체가 해제될 때 해야 할 일들을 정의합니다.
24
25 ACreatureAIController::ACreatureAIController()
26 {
27     /**
28      * BehaviorTreeComponent를 생성해 줍니다.
29      * BehaviorTreeComponent = CreateDefaultSubobject<UBehaviorTreeComponent>(TEXT("BehaviorTreeComponent"));
30      * BlackboardComponent를 생성해 줍니다.
31      * BlackboardComponent = CreateDefaultSubobject<UBlackboardComponent>(TEXT("BlackboardComponent"));
32
33      /**
34      * 컨트롤러가 폰을 빙의하는 가상 함수를 정의합니다.
35      * void ACreatureAIController::OnPossess(APawn* InPawn)
36      {
37          /**
38          * 부모 클래스에 있는 함수의 내용을 가져 옵니다.
39          * Super::OnPossess(InPawn);
40
41
```

No issues found

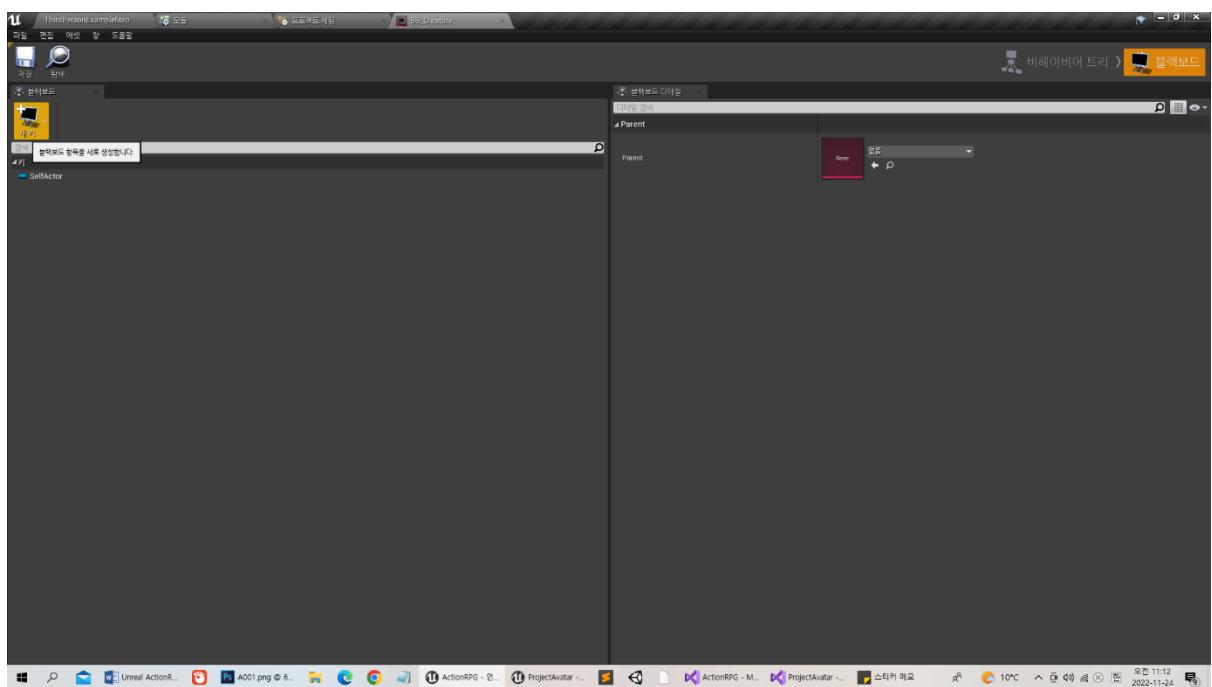
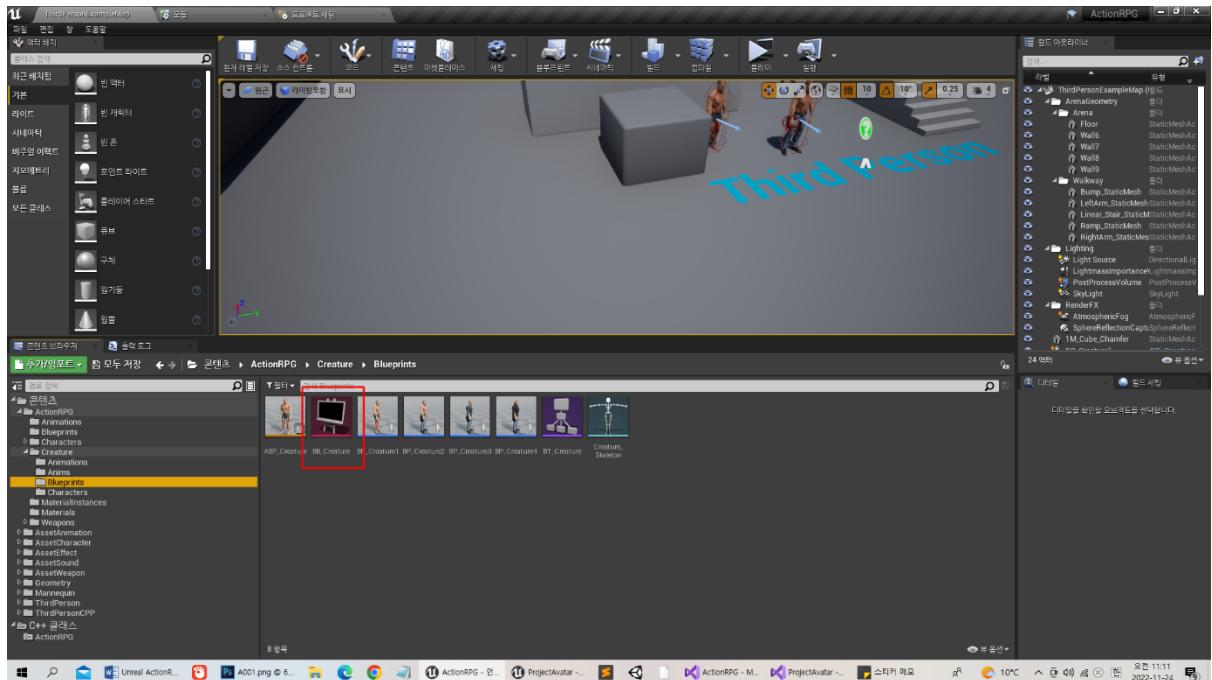
Output

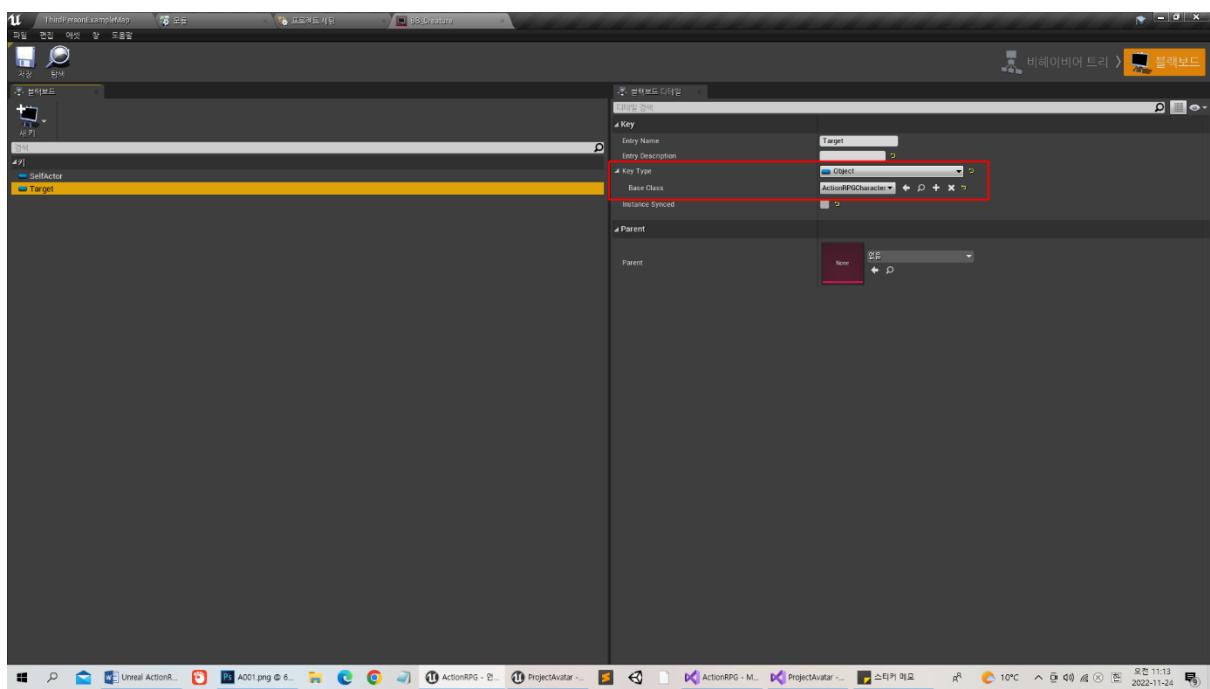
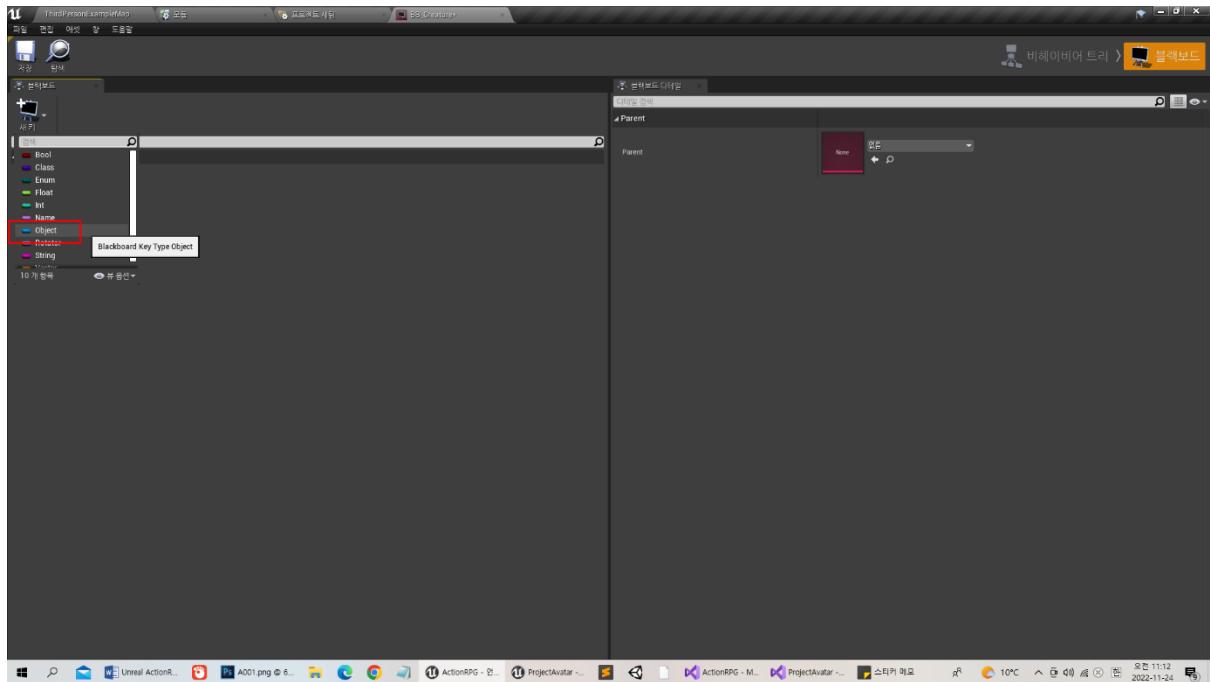
Show output from: Build

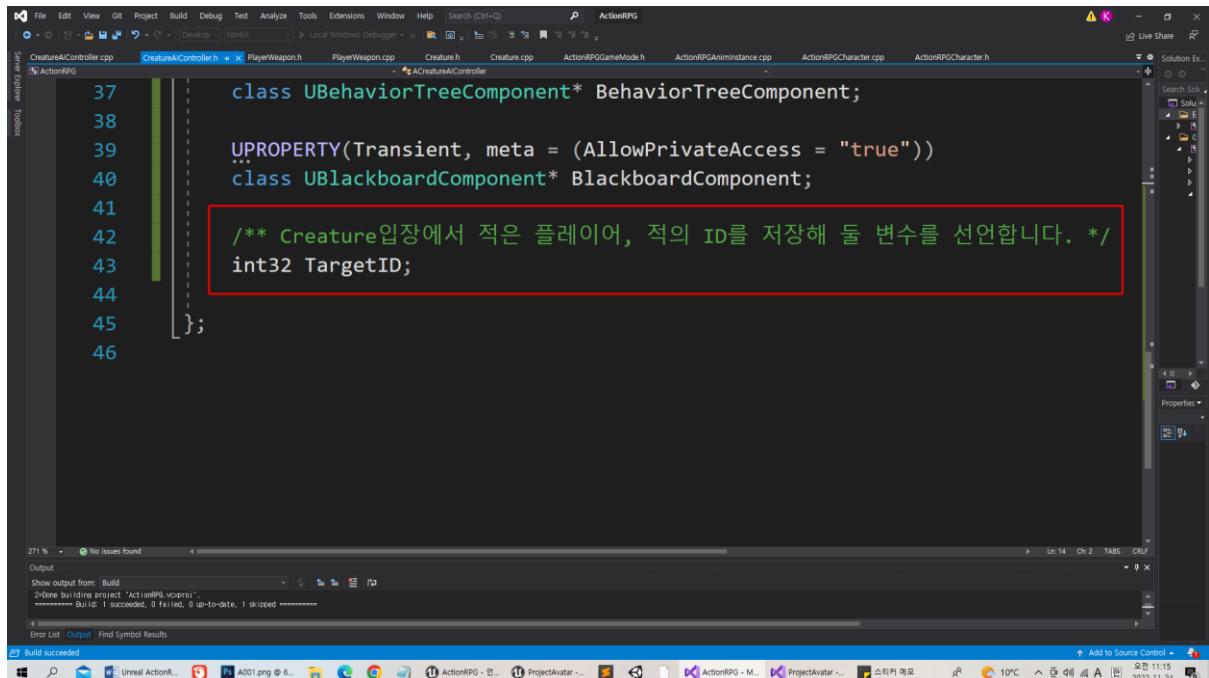
2023-11-24 11:10 10°C 2023-11-24

Build succeeded

블랙보드로 가서 타겟을 정의해 줍니다. Enemy 입장에서 타겟은 플레이어 입니다.

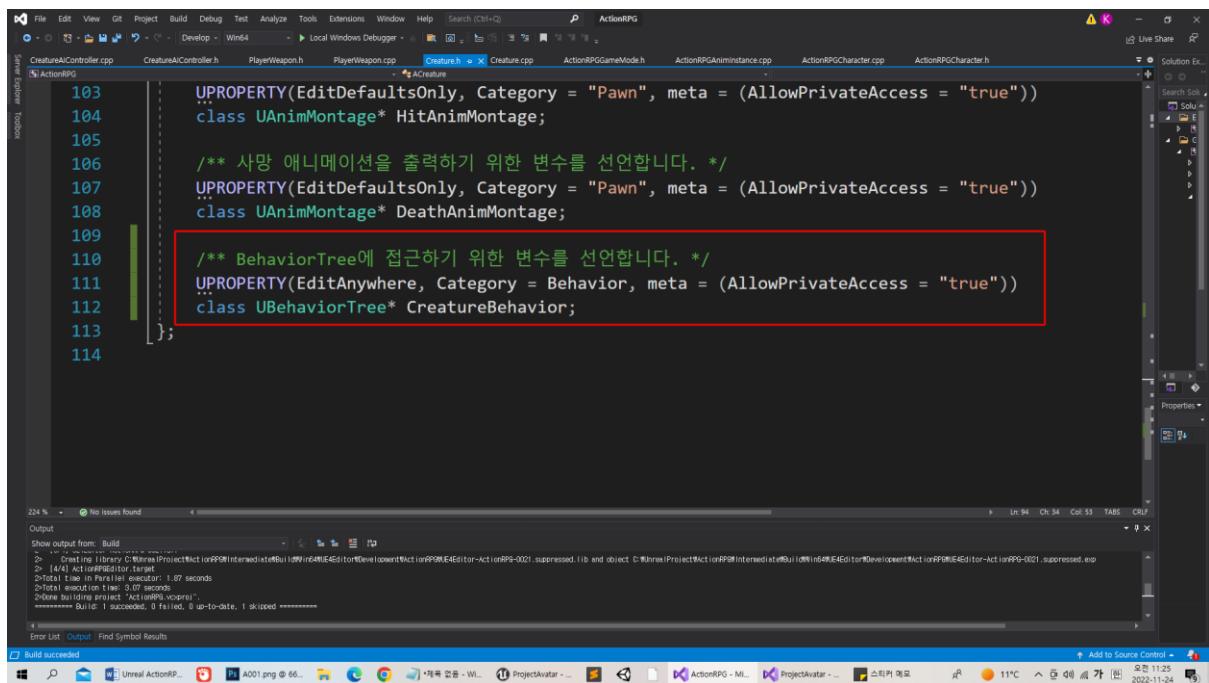






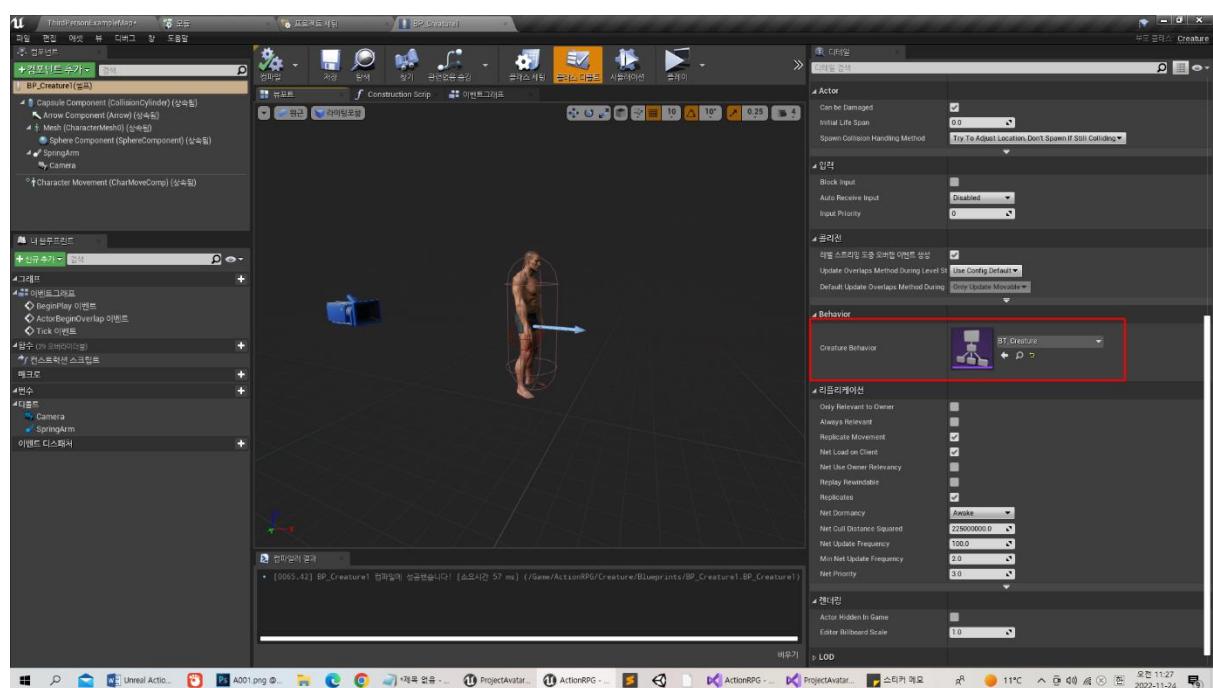
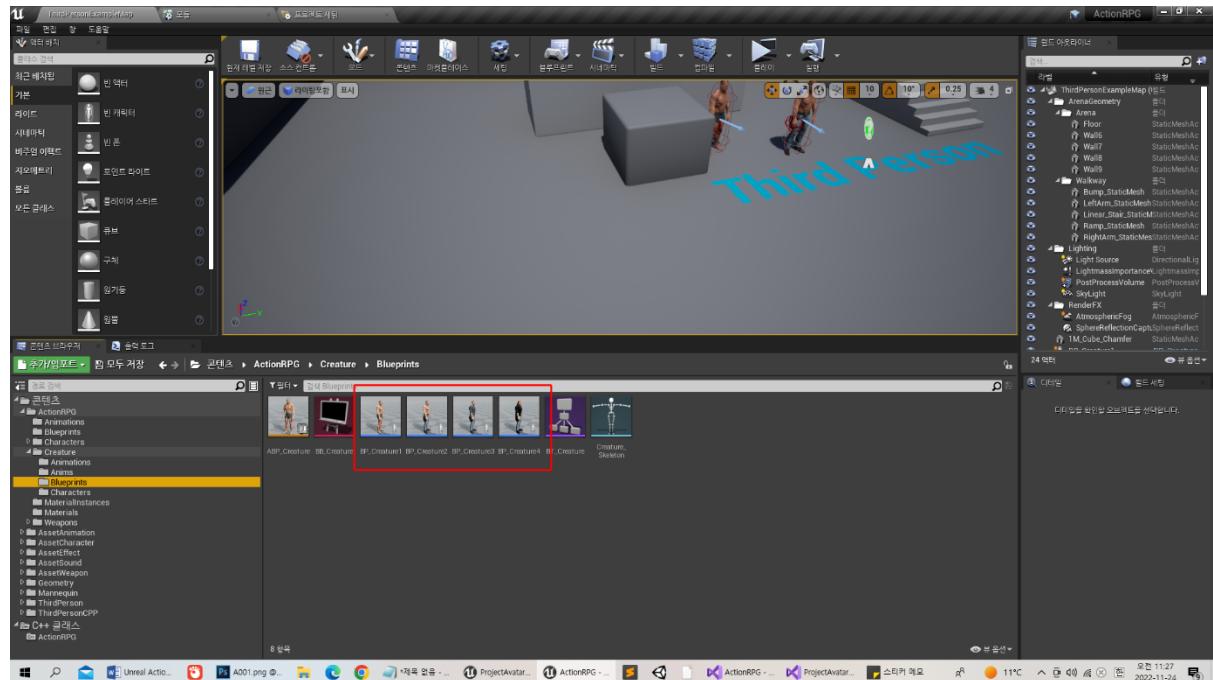
```
37     class UBehaviorTreeComponent* BehaviorTreeComponent;
38
39     UPROPERTY(Transient, meta = (AllowPrivateAccess = "true"))
40     class UBlackboardComponent* BlackboardComponent;
41
42     /* Creature 입장에서 적은 플레이어, 적의 ID를 저장해 둘 변수를 선언합니다. */
43     int32 TargetID;
44
45 };
46
```

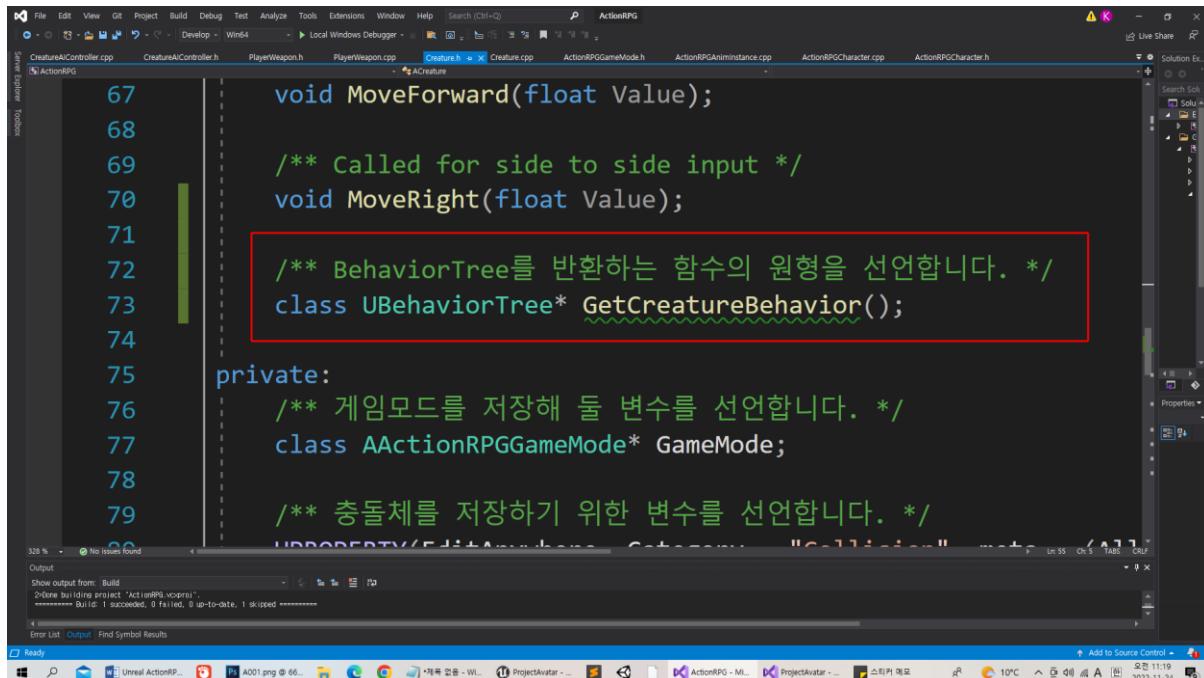
Creature헤더에서 BehaviorTree를 반환하는 함수의 원형을 선언합니다.



```
103     UPROPERTY(EditDefaultsOnly, Category = "Pawn", meta = (AllowPrivateAccess = "true"))
104     class UAnimMontage* HitAnimMontage;
105
106     /* 사망 애니메이션을 출력하기 위한 변수를 선언합니다. */
107     UPROPERTY(EditDefaultsOnly, Category = "Pawn", meta = (AllowPrivateAccess = "true"))
108     class UAnimMontage* DeathAnimMontage;
109
110     /* BehaviorTree에 접근하기 위한 변수를 선언합니다. */
111     UPROPERTY(EditAnywhere, Category = Behavior, meta = (AllowPrivateAccess = "true"))
112     class UBehaviorTree* CreatureBehavior;
113
114 }
```

블루프린트의 디테일 패널에서 적용해 줍니다.



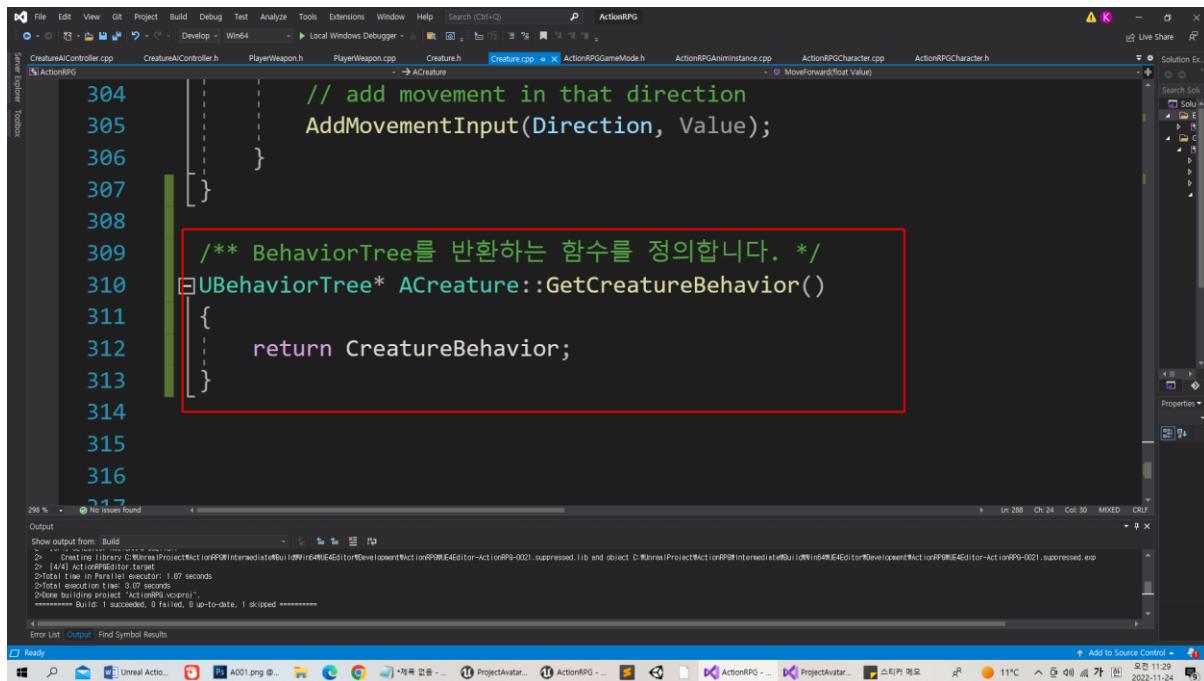


```
67     void MoveForward(float Value);
68
69     /** Called for side to side input */
70     void MoveRight(float Value);
71
72     /** BehaviorTree를 반환하는 함수의 원형을 선언합니다. */
73     class UBehaviorTree* GetCreatureBehavior();
74
75     private:
76     /** 게임모드를 저장해 둘 변수를 선언합니다. */
77     class AActionRPGGameMode* GameMode;
78
79     /** 충돌체를 저장하기 위한 변수를 선언합니다. */
80 }
```

Output
Show output from: Build
2022 building project "ActionRPG.vcxproj".
----- Build 1: succeeded, 0 failed, 0 up-to-date, 1 skipped -----

Error List Output Find Symbol Results

구현해 줍니다.



```
304     // add movement in that direction
305     AddMovementInput(Direction, Value);
306 }
307 }
308
309     /** BehaviorTree를 반환하는 함수를 정의합니다. */
310     UBehaviorTree* ACreature::GetCreatureBehavior()
311     {
312     return CreatureBehavior;
313     }
314
315
316
317 }
```

Output
Show output from: Build
2> Creating library C:\Users\Project\Actor\ActorRPG\Intermediate\Build\Win64\Editor\Development\ActionRPG-0021\suppressed.lib and object C:\Users\Project\Actor\ActorRPG\Intermediate\Build\Win64\Editor\Development\ActionRPG-0021\suppressed.exp
2> Total 1 file(s) in Parallel executor: 1.07 seconds
2> Total execution time: 3.07 seconds
2> Done building project "ActionRPG.vcxproj".
----- Build 1: succeeded, 0 failed, 0 up-to-date, 1 skipped -----

Error List Output Find Symbol Results

OnPossess() 함수를 구현해 주도록 합니다.

```

7 #include "BehaviorTree/BehaviorTree.h"
8 #include "BehaviorTree/BehaviorTreeComponent.h"
9 #include "BehaviorTree/BlackboardComponent.h"
10 #include "BehaviorTree/Blackboard/BlackboardKeyType.h"
11
12 /** Creature 클래스에 접근하기 위한 */
13 #include "Creature.h"
14
15 /**
16  * 생성자 함수를 정의합니다.
17  * 생성자 함수의 목적은 객체가 생성되면서 멤버 변수들의 초기화를
18

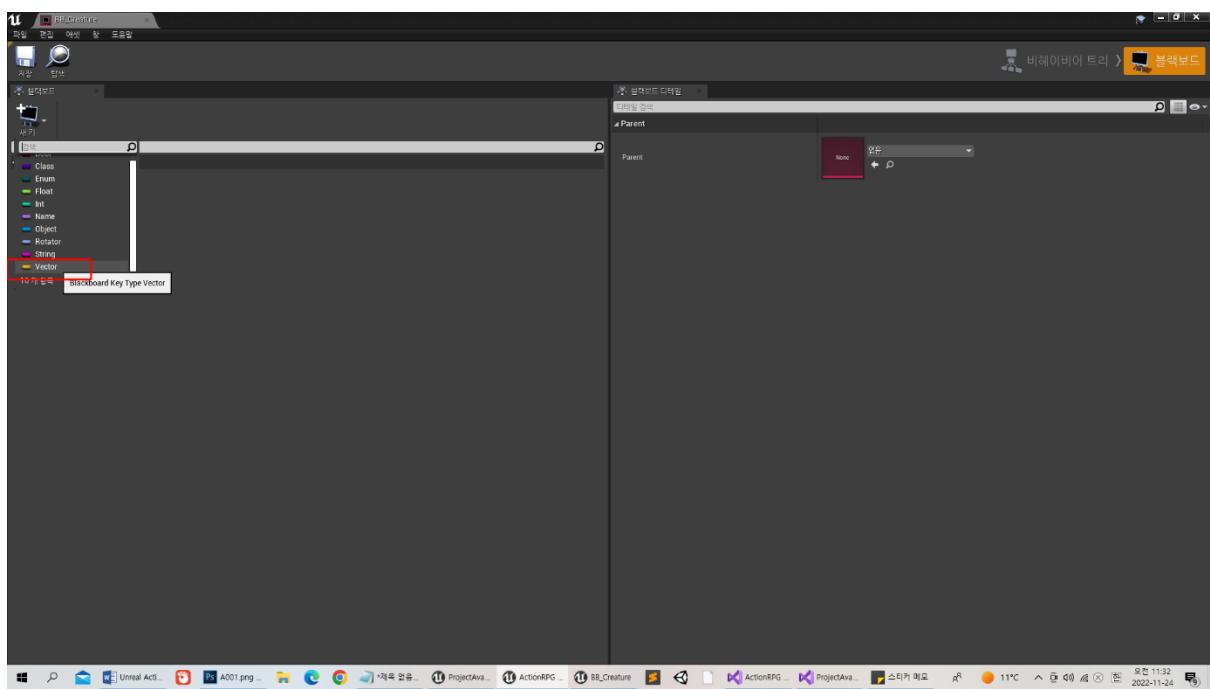
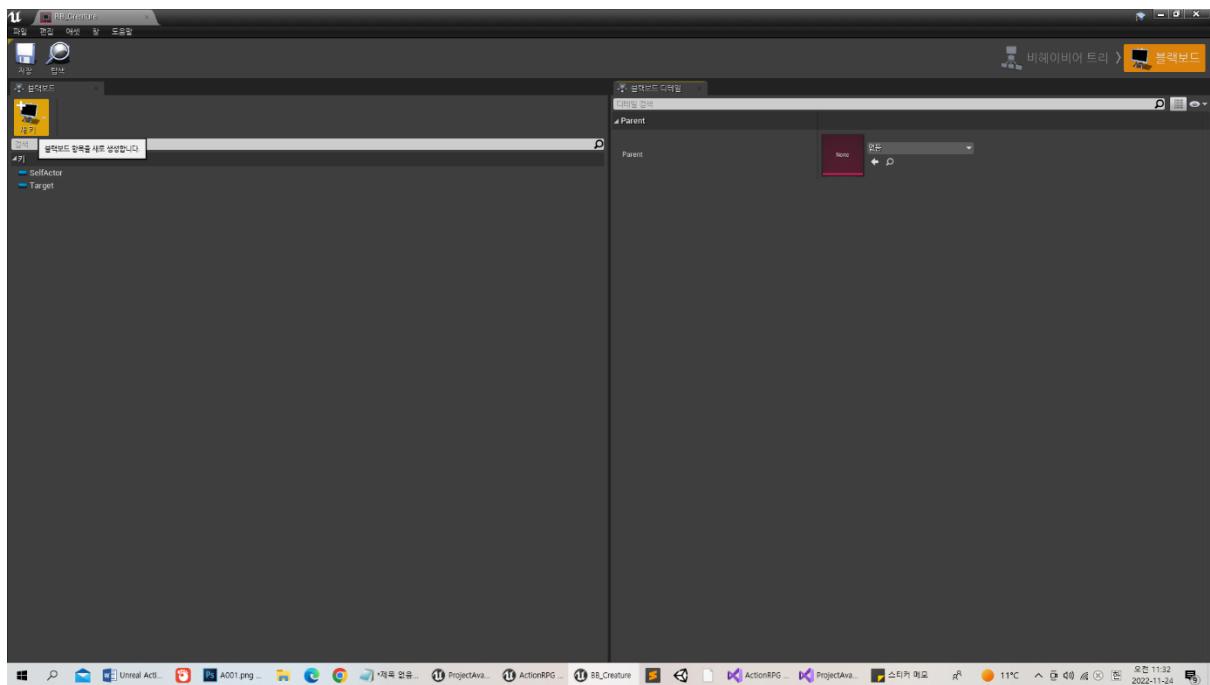
```

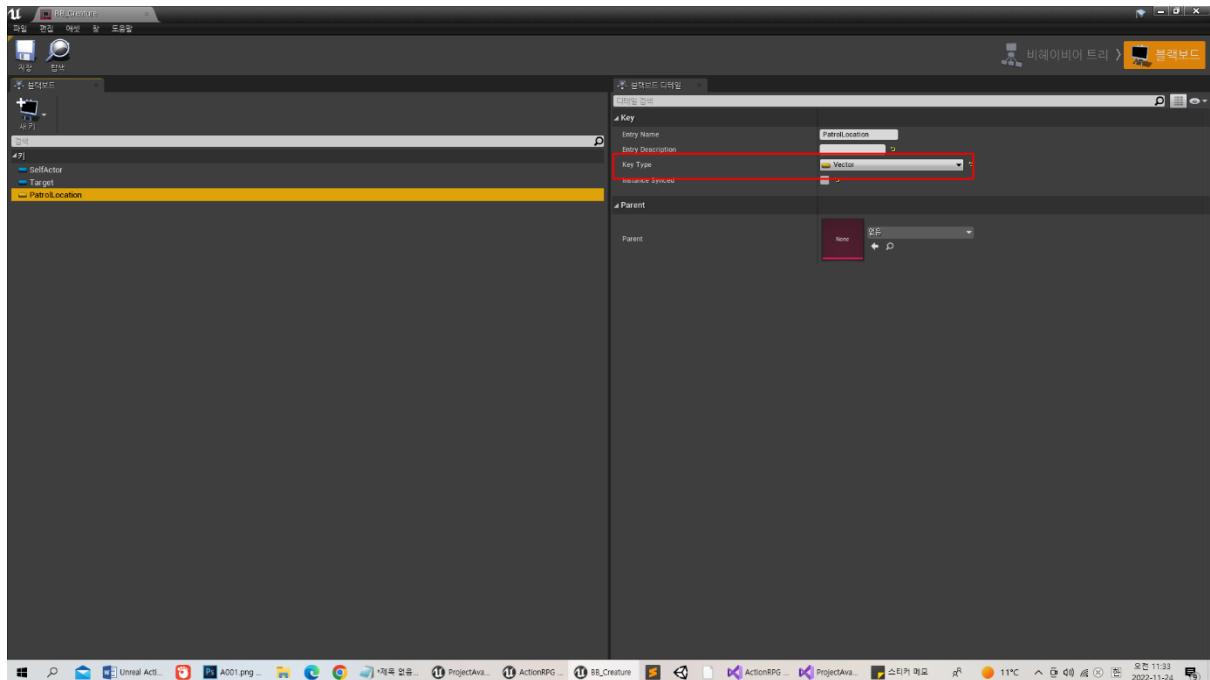
```

34 /**
35  * 컨트롤러가 폰을 빙의하는 가장 함수를 정의합니다.
36 */
37 void ACreatureAIController::OnPossess(APawn* InPawn)
38 {
39     /**
40      * 부모 클래스에 있는 함수의 내용을 가져 옵니다.
41     */
42     Super::OnPossess(InPawn);
43
44     UE_LOG(LogTemp, Warning, TEXT("OnPossess"));
45
46     /**
47      * Pawn을 Creature로 형변환 합니다.
48      */
49     ACreature* Creature = Cast<ACreature>(InPawn);
50
51     /**
52      * 만약 creature가 있고, Creature의 BehaviorTree가 있다면.
53      */
54     if (Creature && Creature->GetCreatureBehavior())
55     {
56         /**
57          * 만약 creature의 BehaviorTree에 BlackboardAsset이 있다면.
58          */
59         if (Creature->GetCreatureBehavior()->BlackboardAsset)
60         {
61             /**
62              * BlackboardAsset을 초기화 해 줍니다.
63              */
64             BlackboardComponent->InitializeBlackboard(*Creature->GetCreatureBehavior()->BlackboardAsset);
65             /**
66               * 적의 ID를 가져옵니다.
67               */
68             TargetID = BlackboardComponent->GetKeyID("Target");
69             /**
70               * 이제 캐릭터의 AI를 시작합니다.
71               */
72             BehaviorTreeComponent->StartTree(*(Creature->GetCreatureBehavior()));
73
74             UE_LOG(LogTemp, Warning, TEXT("BehaviorTreeComponent->StartTree"));
75         }
76     }
77
78     /**
79      * 컨트롤러가 폰을 빙의해제하는 가장 함수를 정의합니다.
80      */
81     void ACreatureAIController::OnUnPossess()
82 {
83
84

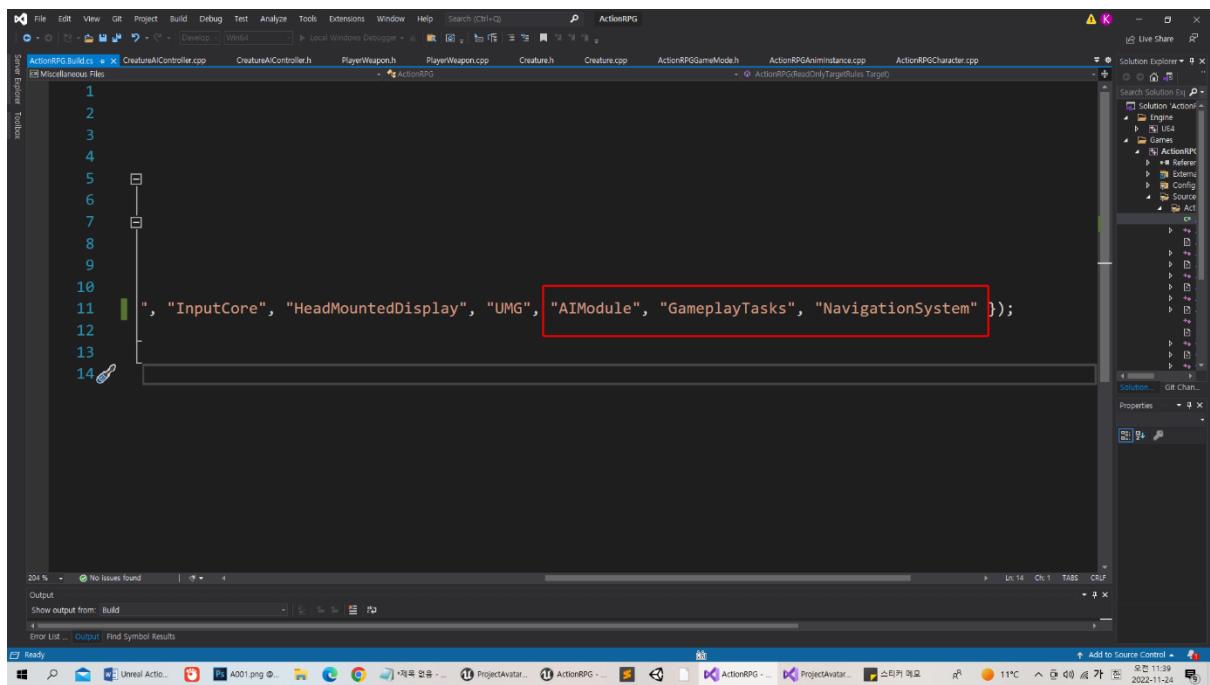
```

몬스터가 플레이어를 발견하지 못했을 경우는 임의의 위치로 패트를 합니다. 블랙보드에서 정의해 줍니다.

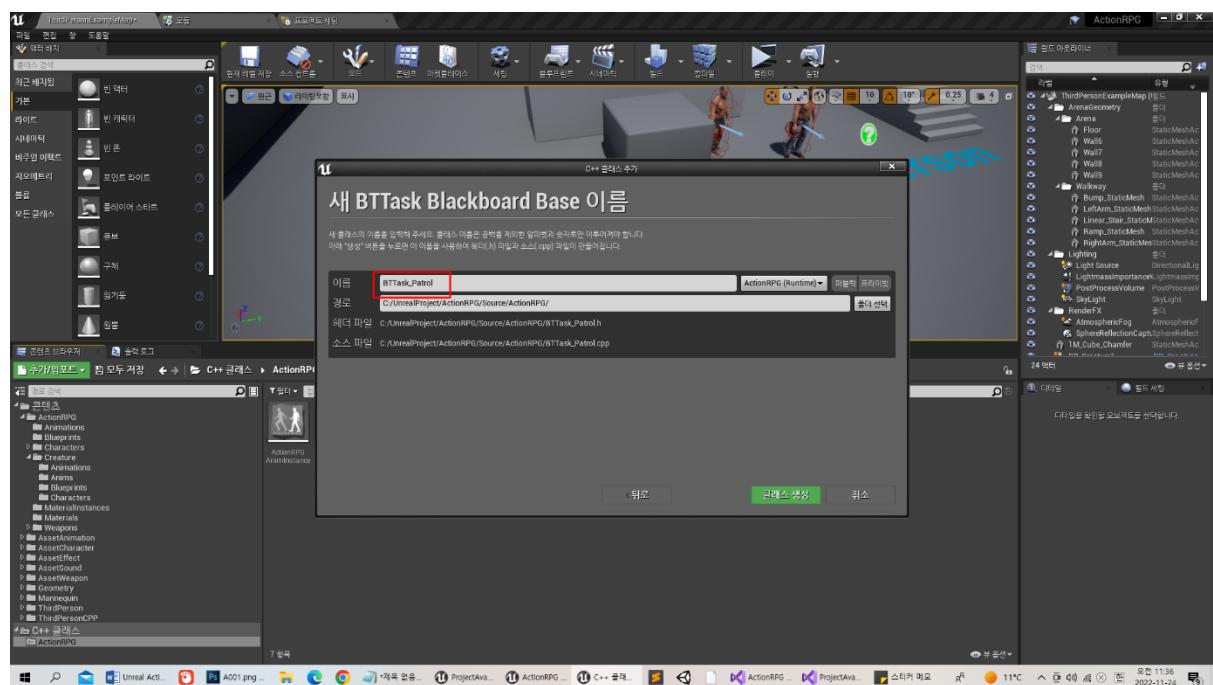
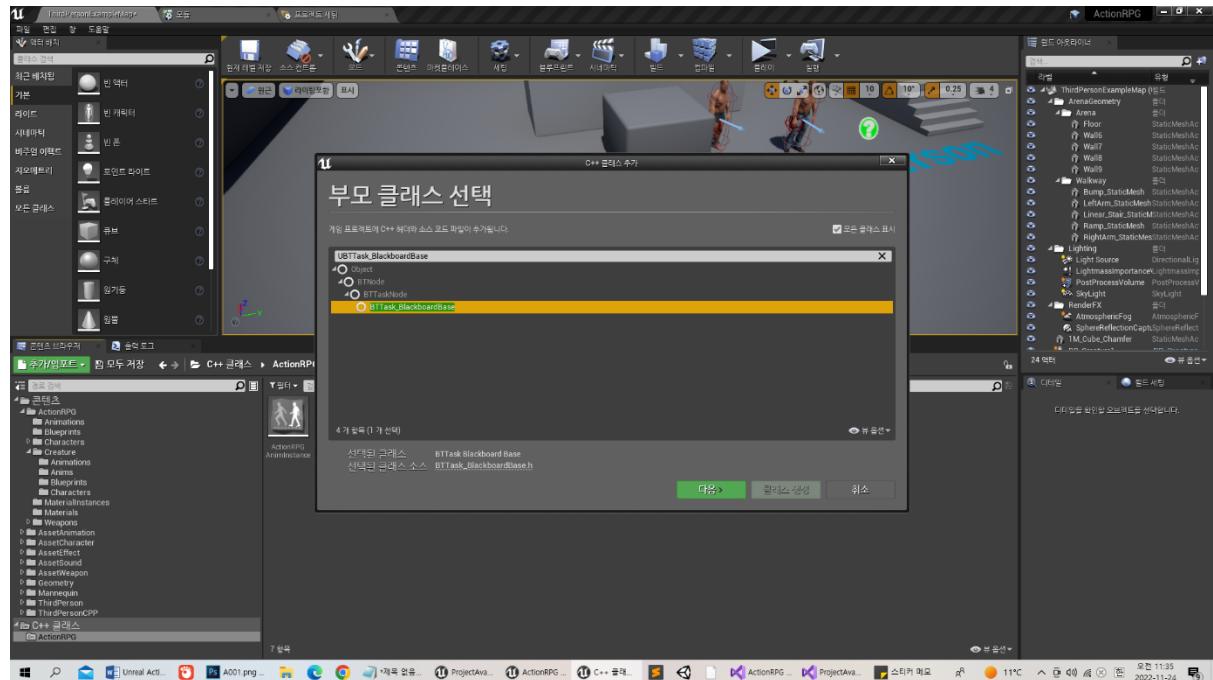


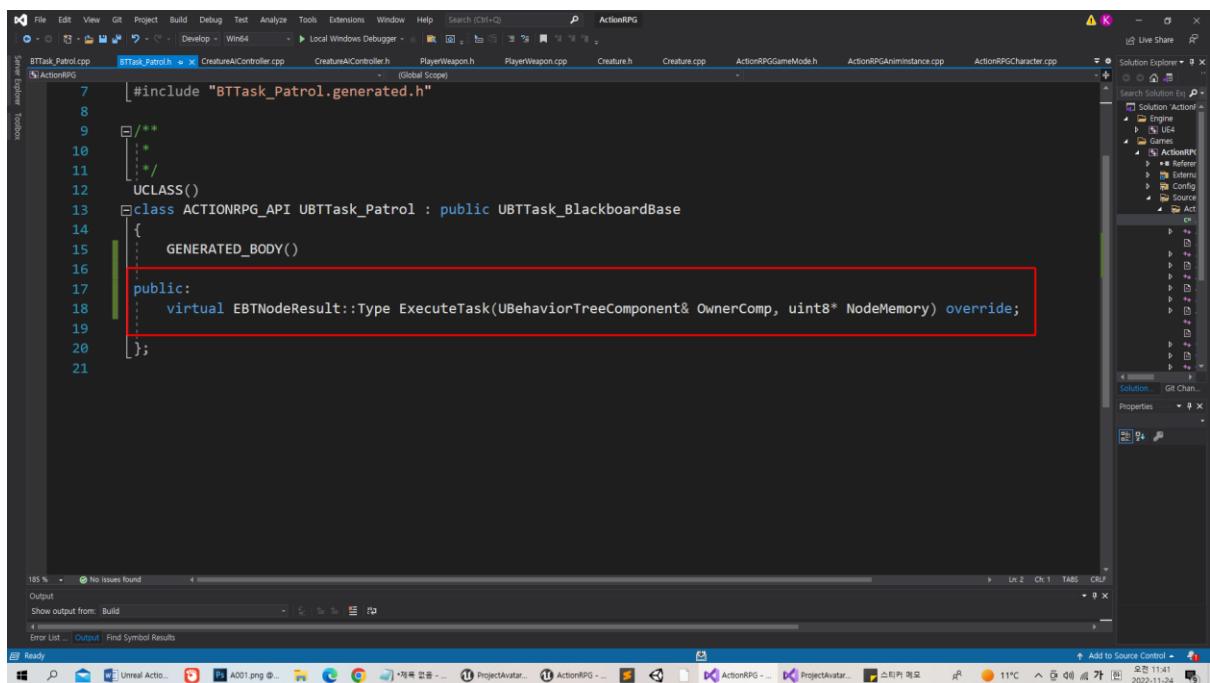
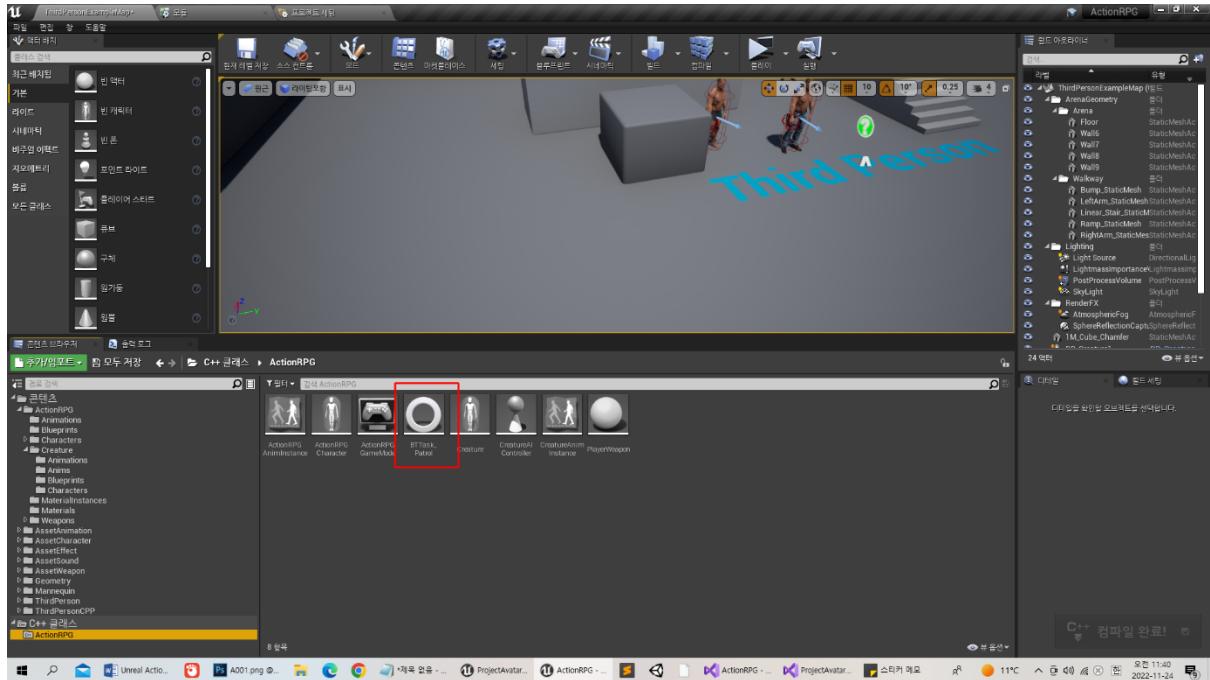


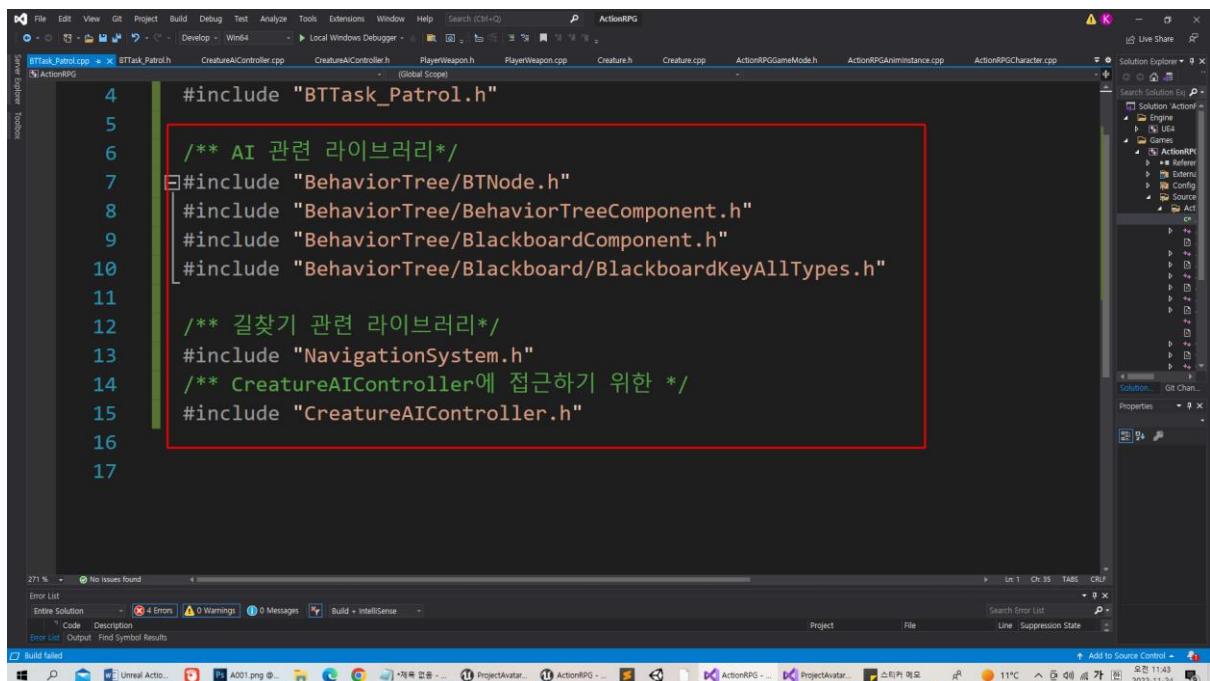
AI 모듈을 추가해 줍니다.



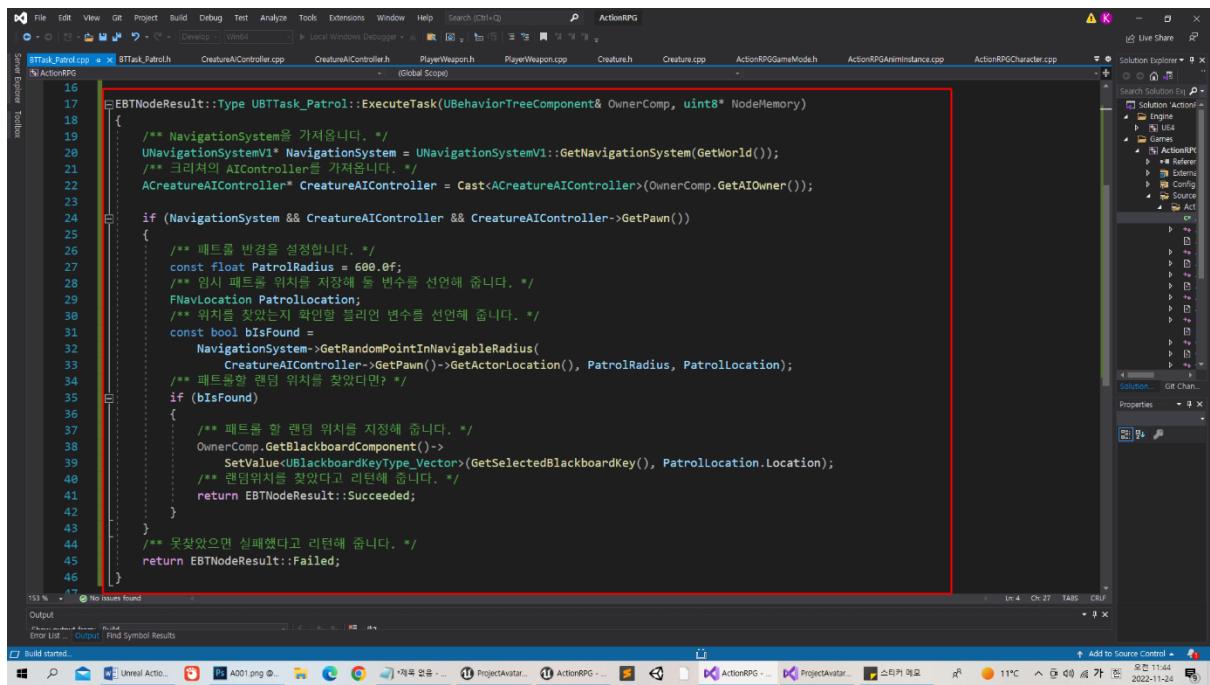
BehaviorTree에서 패트롤할 위치를 찾은 Task를 정의해 줍니다.





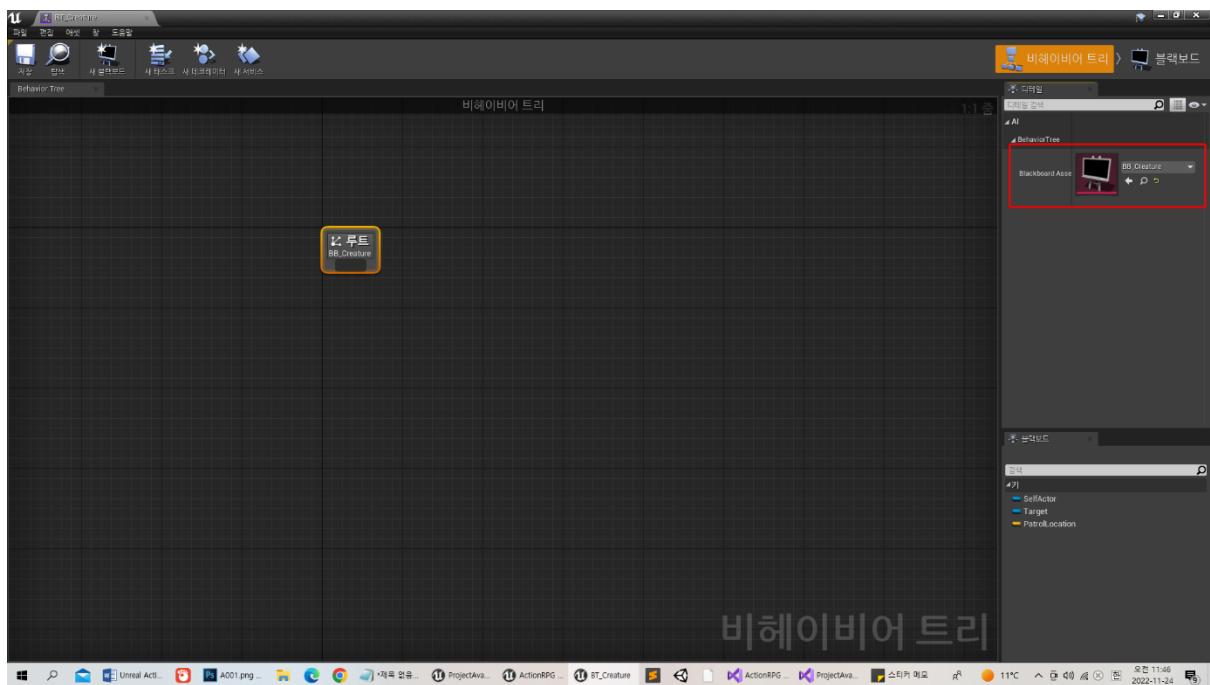
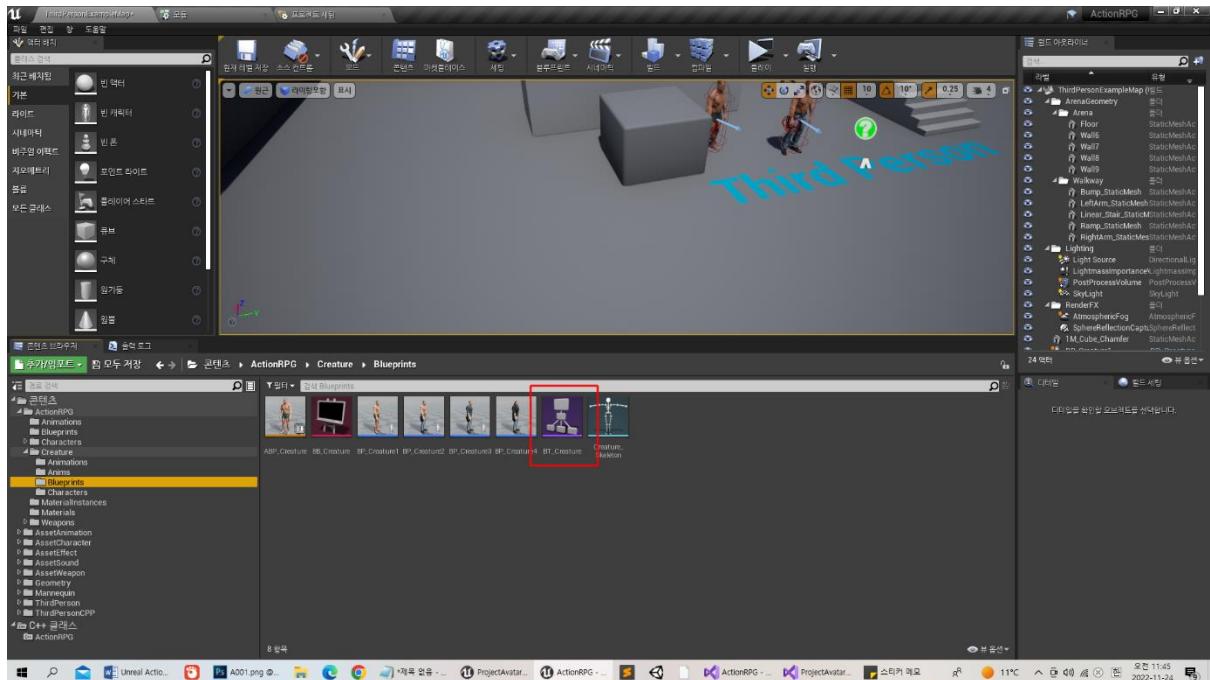


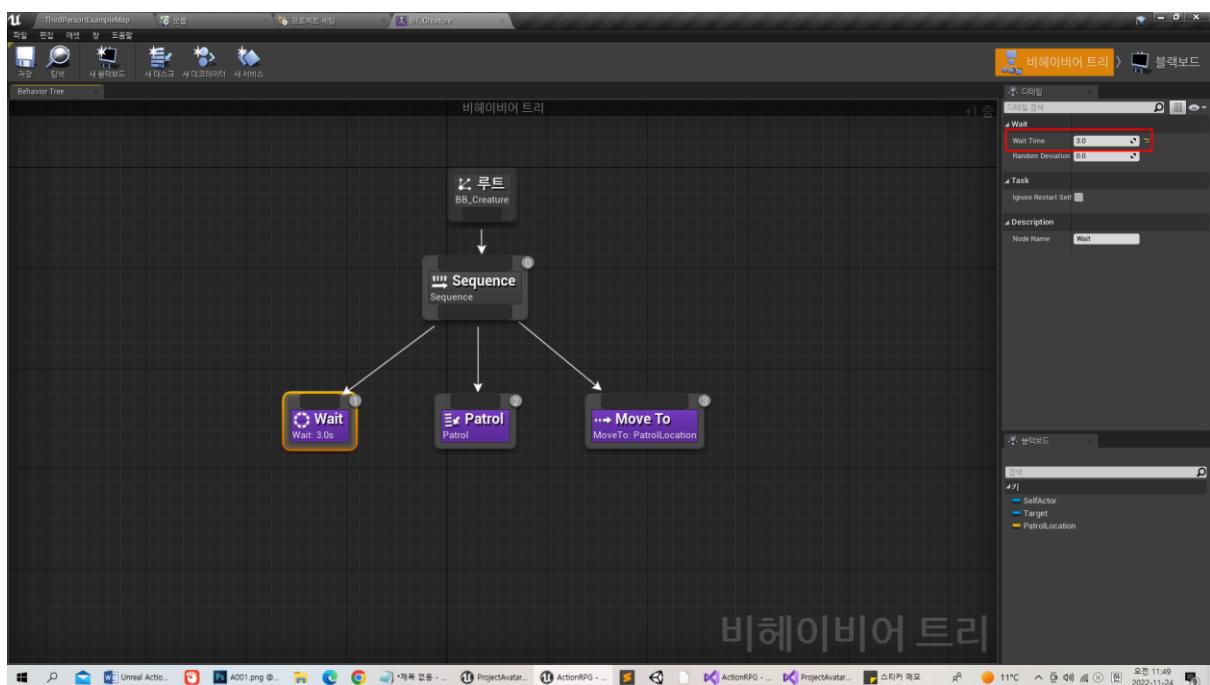
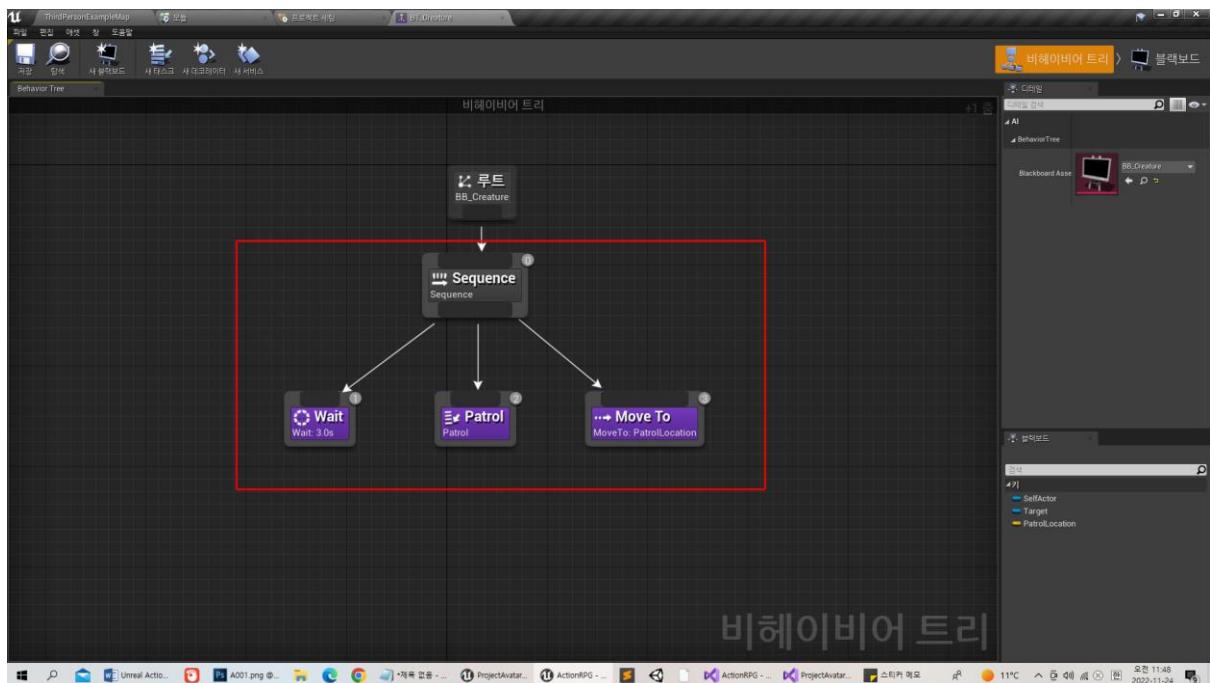
```
4 #include "BTTask_Patrol.h"
5
6 /* AI 관련 라이브러리*/
7 #include "BehaviorTree/BTNode.h"
8 #include "BehaviorTree/BehaviorTreeComponent.h"
9 #include "BehaviorTree/BlackboardComponent.h"
10 #include "BehaviorTree/Blackboard/BlackboardKeyAllTypes.h"
11
12 /* 길찾기 관련 라이브러리*/
13 #include "NavigationSystem.h"
14 /* CreatureAIController에 접근하기 위한 */
15 #include "CreatureAIController.h"
16
17
```

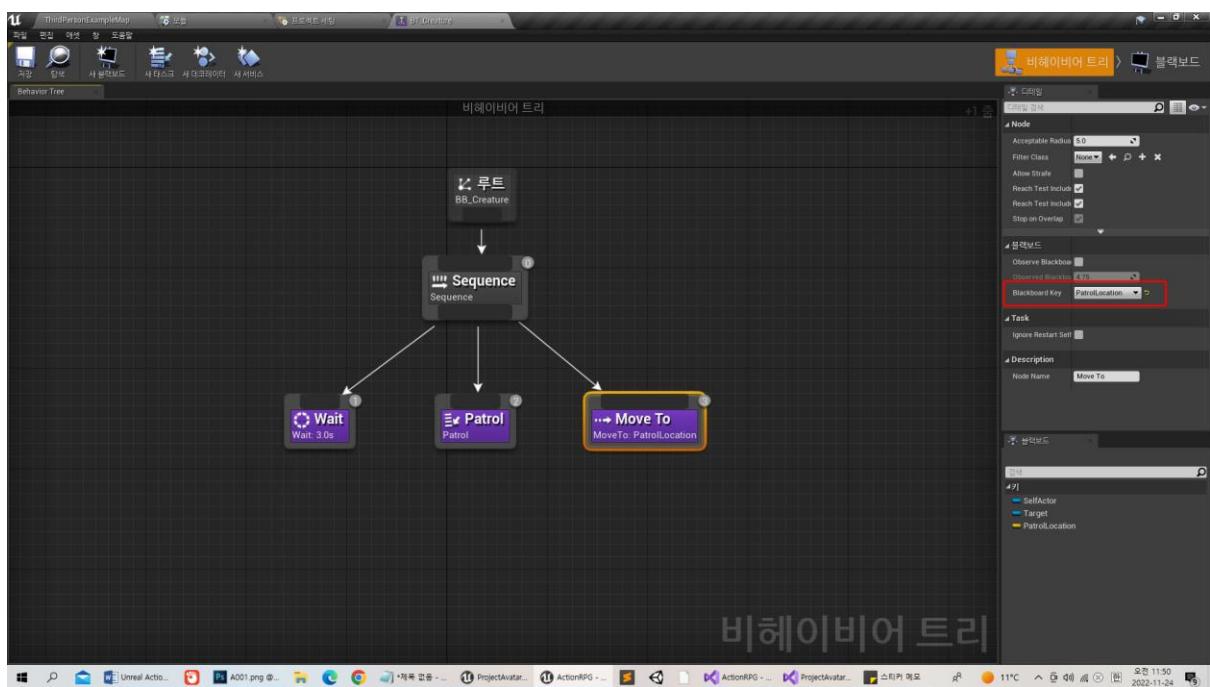
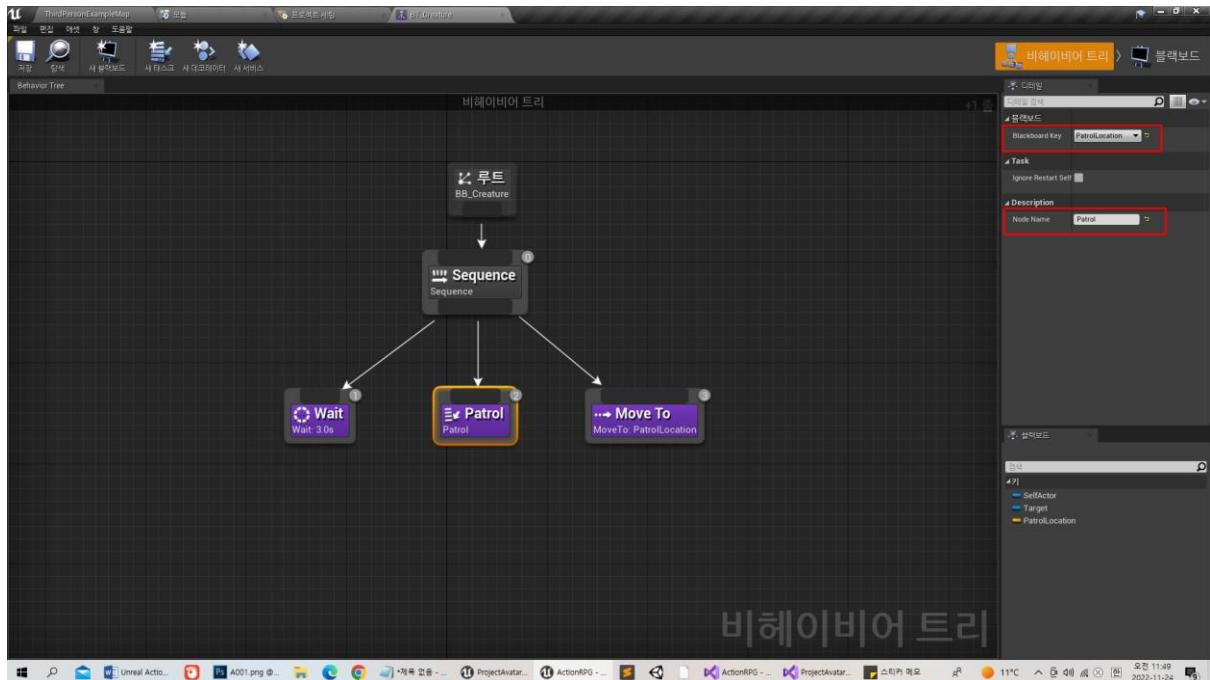


```
16 EBTNodeResult::Type UBTTask_Patrol::ExecuteTask(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory)
17 {
18     /* NavigationSystem을 가져옵니다. */
19     UNavigationSystemV1* NavigationSystem = UNavigationSystemV1::GetNavigationSystem(GetWorld());
20     /* 크리처의 AIController를 가져옵니다. */
21     ACreatureAIController* CreatureAIController = Cast<ACreatureAIController>(OwnerComp.GetAIOwner());
22
23     if (NavigationSystem && CreatureAIController && CreatureAIController->GetPawn())
24     {
25         /* 패트를 반경을 설정합니다. */
26         const float PatrolRadius = 600.0f;
27         /* 임시 패트를 위치를 지정해 둘 변수를 선언해 줍니다. */
28         FNavLocation PatrolLocation;
29         /* 위치를 찾았는지 확인할 플리언 변수를 선언해 줍니다. */
30         const bool bIsFound =
31             NavigationSystem->GetRandomPointInNavigableRadius(
32                 CreatureAIController->GetPawn()->GetActorLocation(), PatrolRadius, PatrolLocation);
33         /* 패트를 할 랜덤 위치를 찾았다면? */
34         if (bIsFound)
35         {
36             /* 패트를 할 랜덤 위치를 지정해 줍니다. */
37             OwnerComp.GetBlackboardComponent()->
38                 SetValue<UBlackboardKeyType_Vector>(GetSelectedBlackboardKey(), PatrolLocation.Location);
39             /* 랜덤위치를 찾았다고 리턴해 줍니다. */
40             return EBTNodeResult::Succeeded;
41         }
42     }
43     /* 못찾았으면 실패했다고 리턴해 줍니다. */
44     return EBTNodeResult::Failed;
45 }
46
```

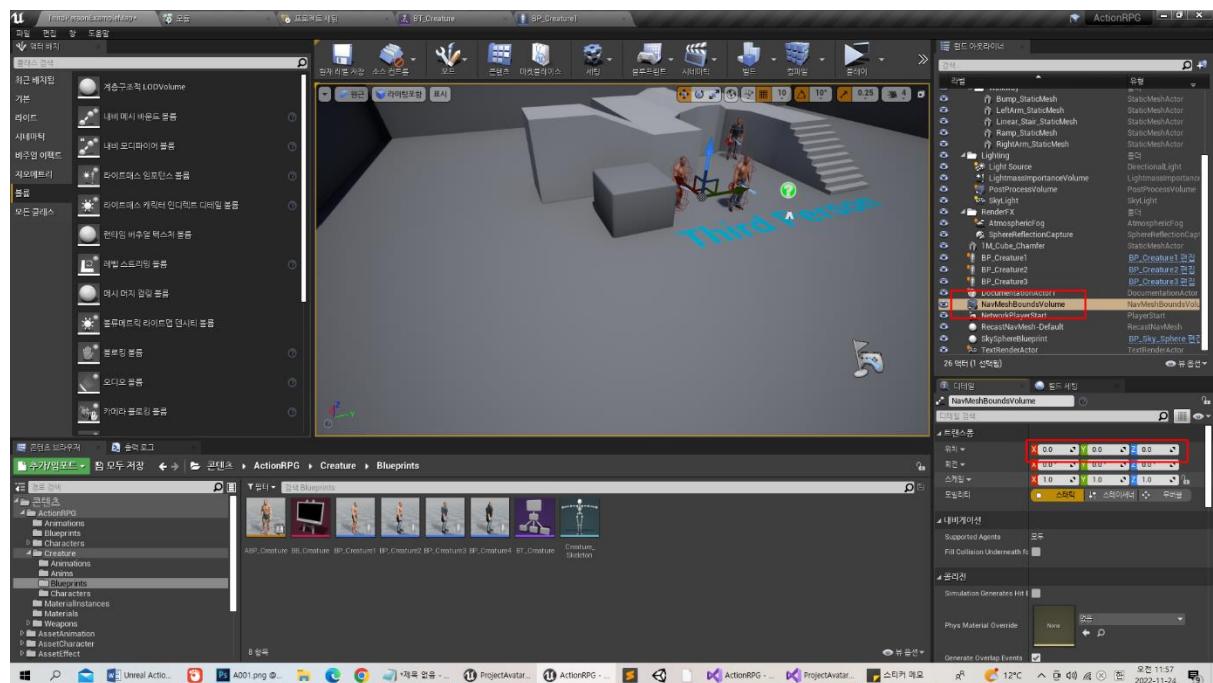
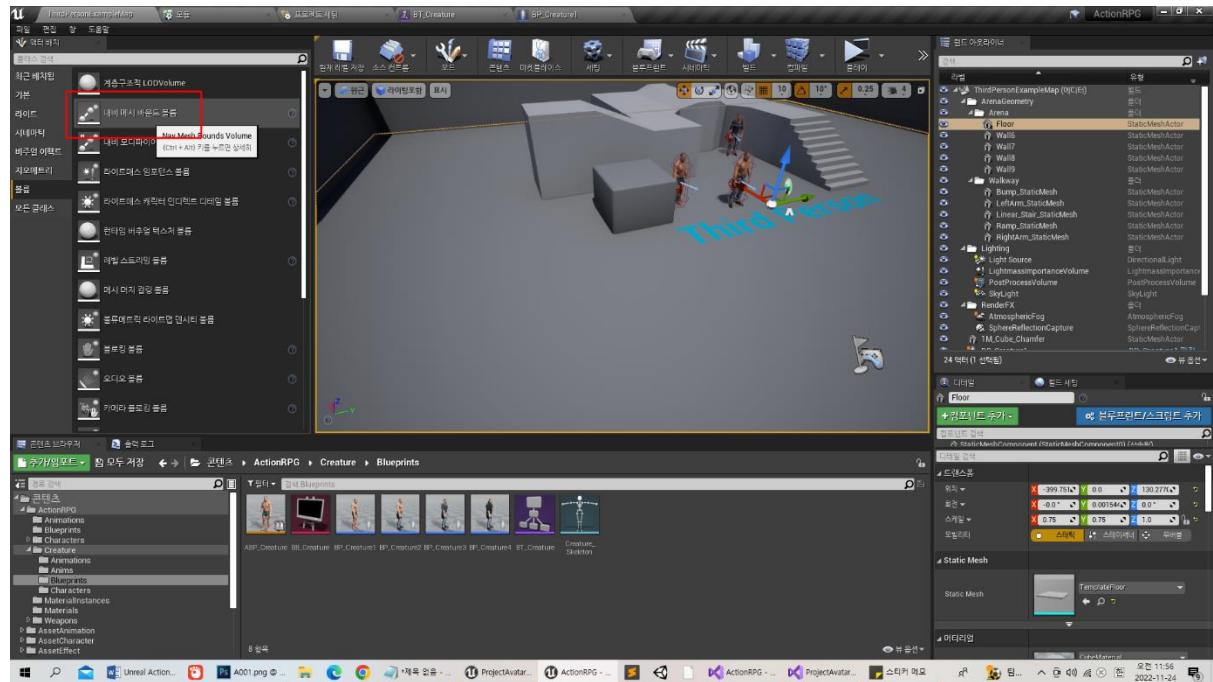
BehaviorTree에서 구성해 줍니다.

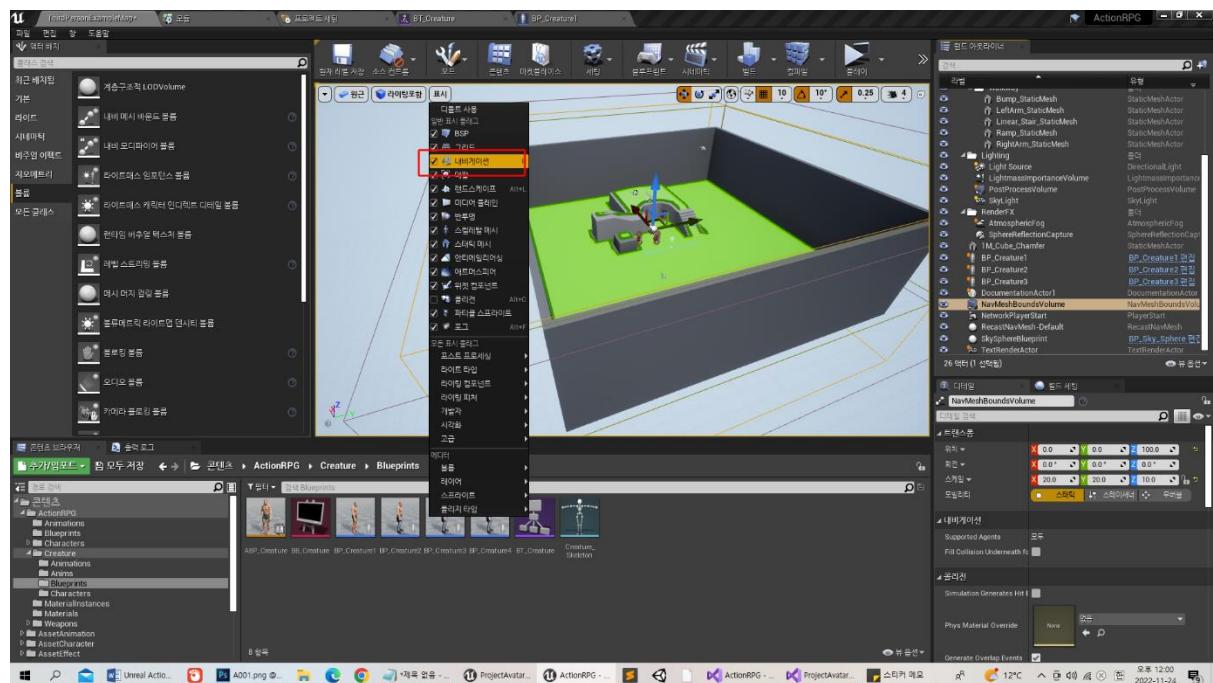
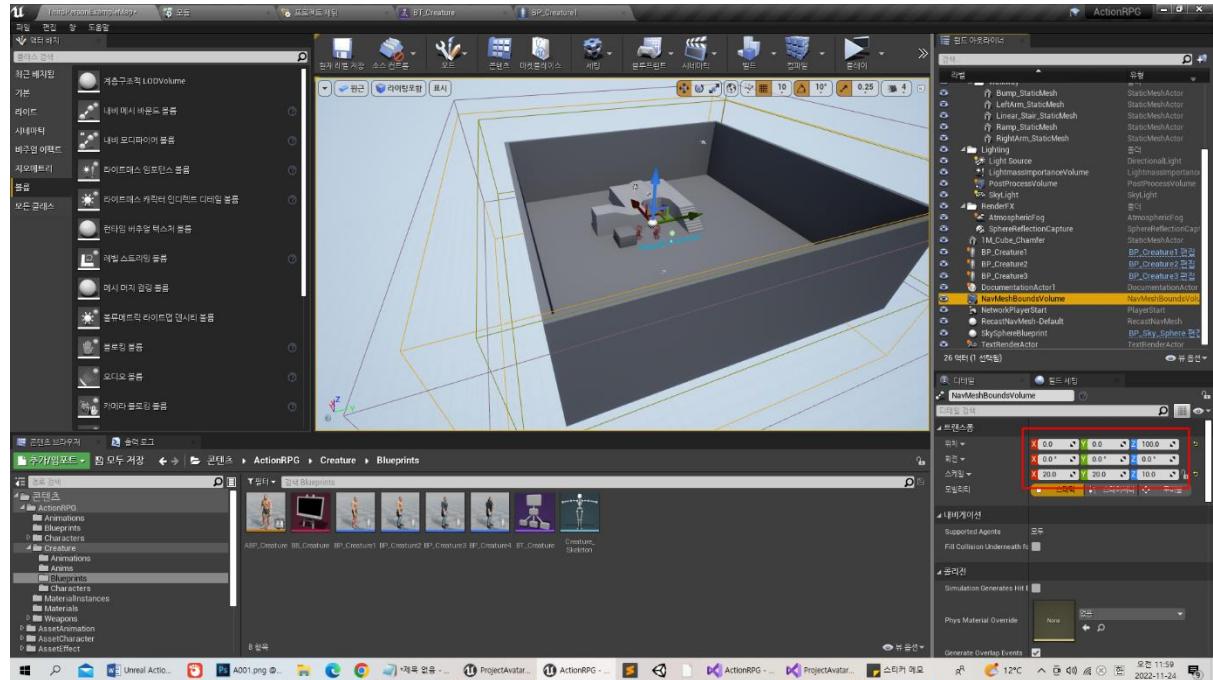




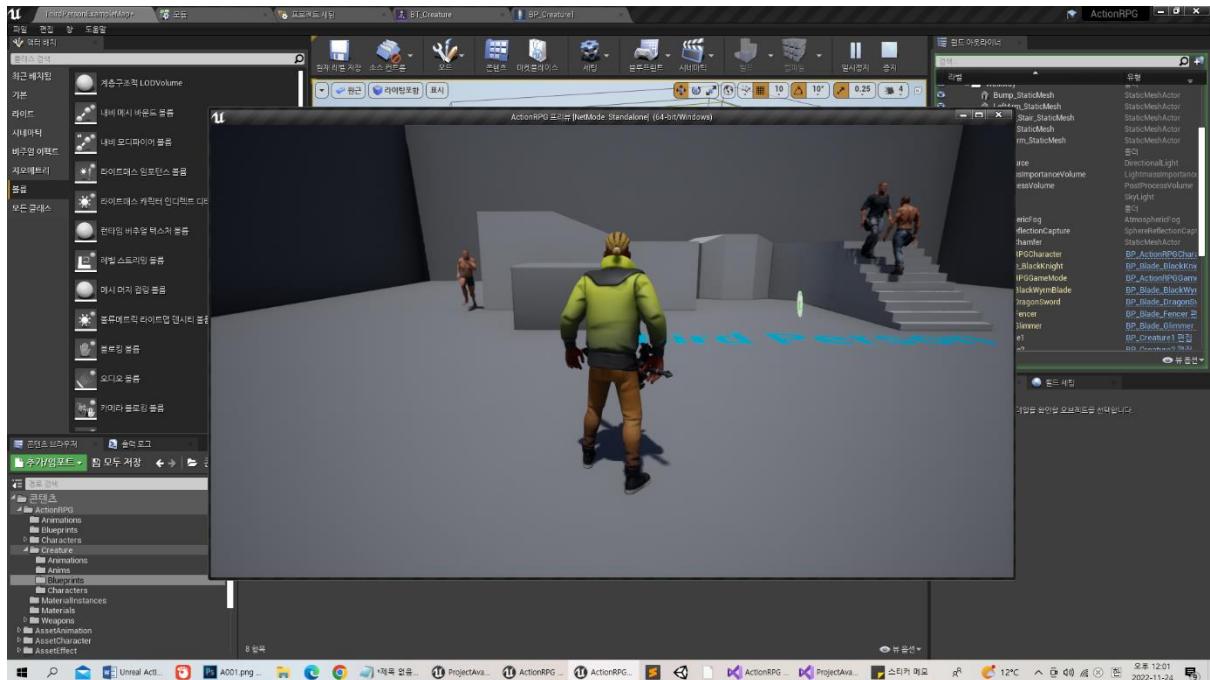


네비게이션 메쉬를 레벨에 설치해 줍니다.



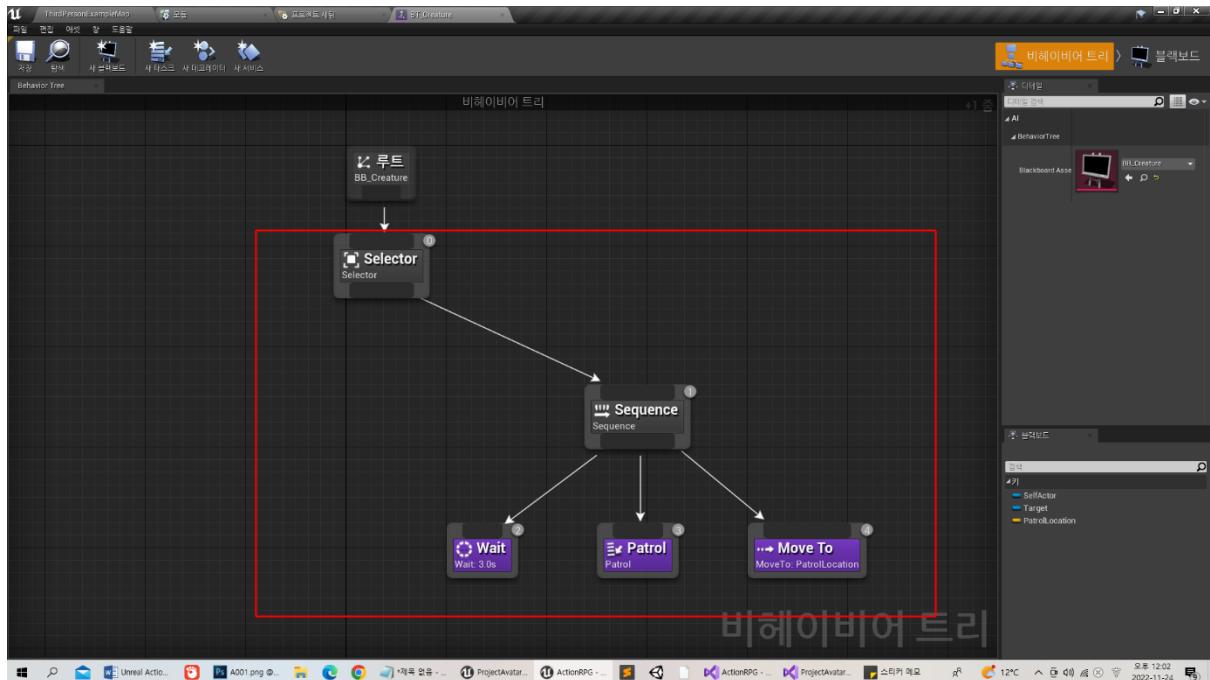


플레이를 해 봅니다.

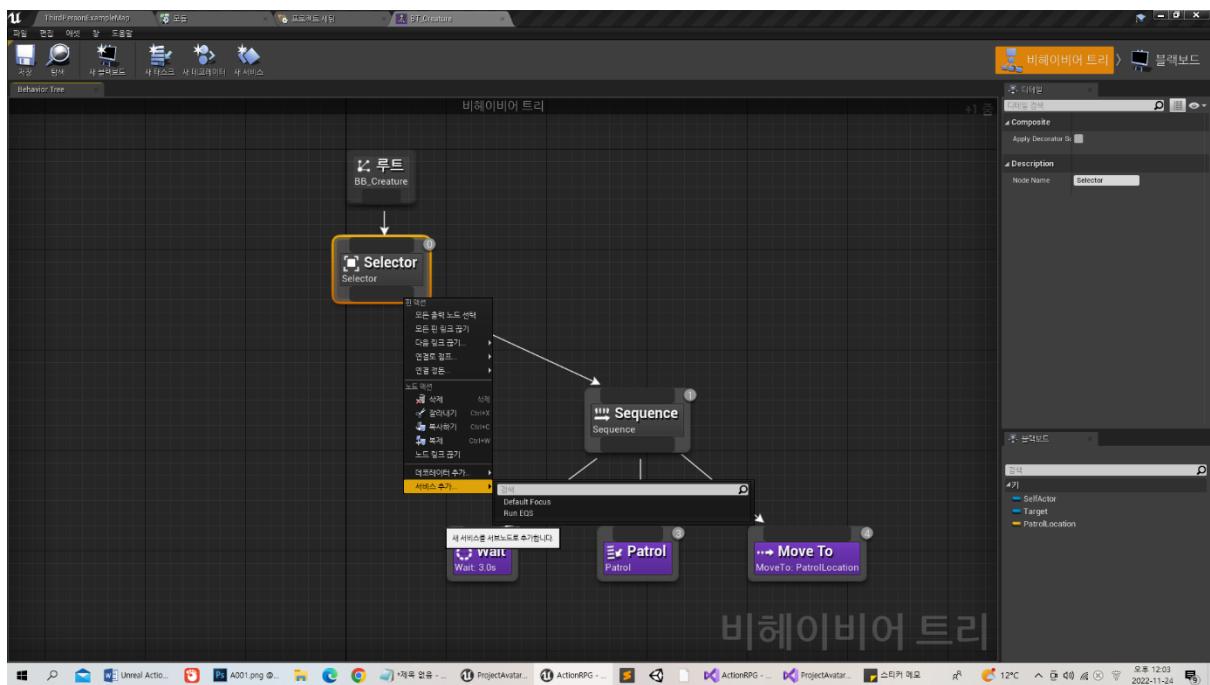


플레이를 해서 결과를 확인해 봅니다. 3초 쉬고 패트를 할 위치를 찾고 패트를 할 위치로 가는 것을 볼 수 있습니다.

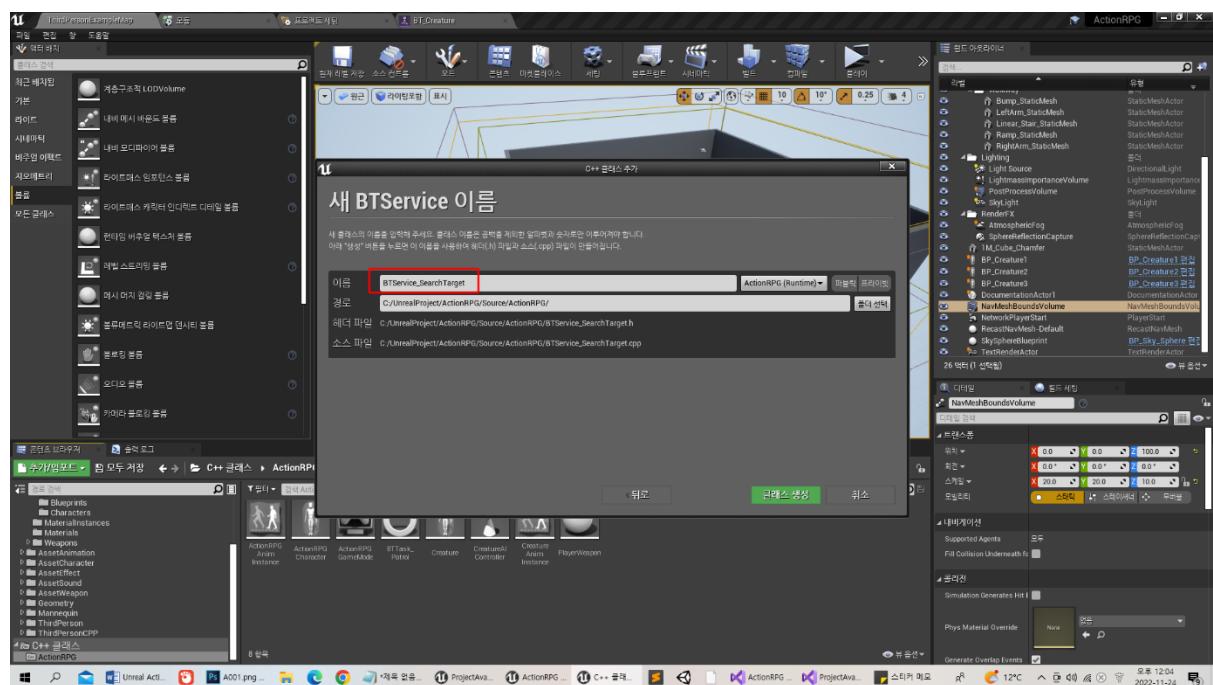
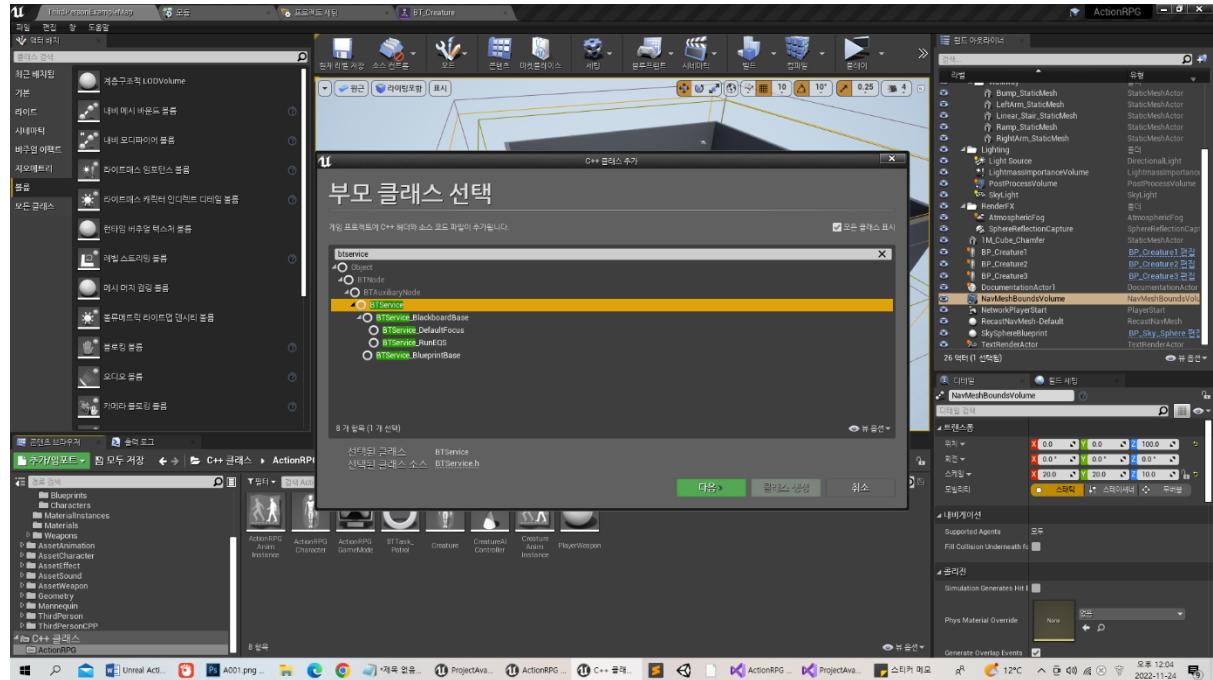
이제 플레이어와 일정거리 안에 있으면 플레이어에게 다가와 공격하고 그렇지 않으면 패트를 하도록 해 줍니다.

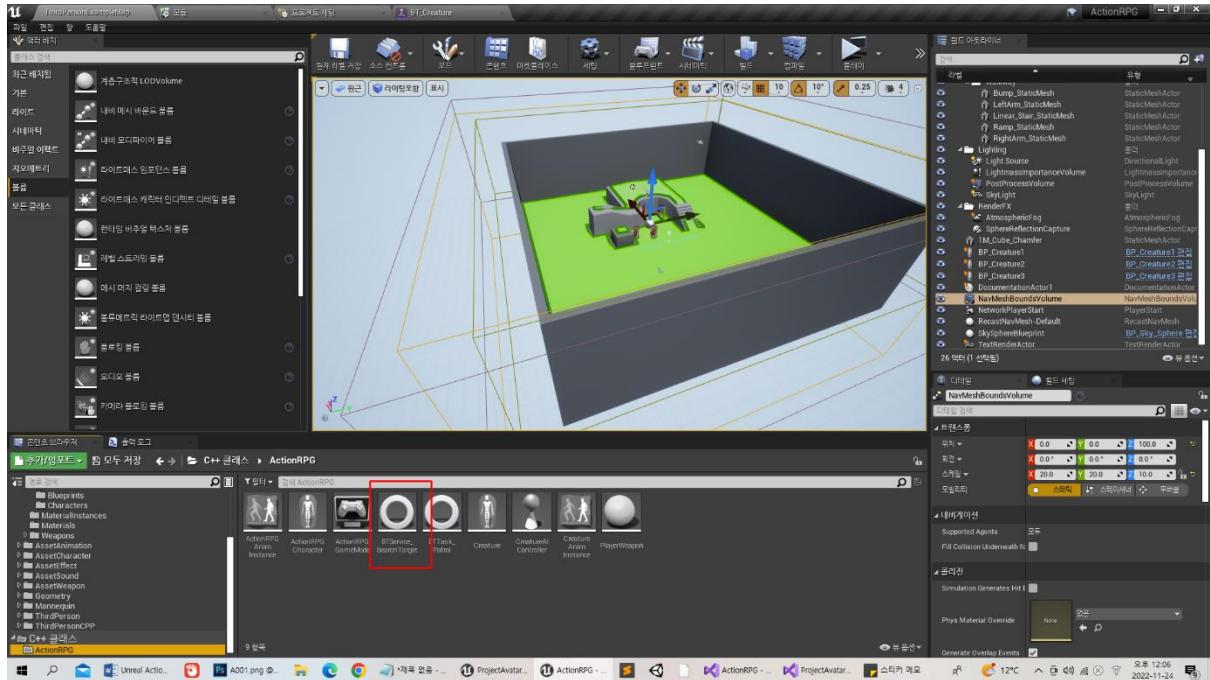


매프레임 타겟을 찾도록 하면 부하가 많이 걸릴 수 있으니 주기적으로 타겟을 찾도록 해 줍니다. 서비스로 해 줄 수가 있습니다.

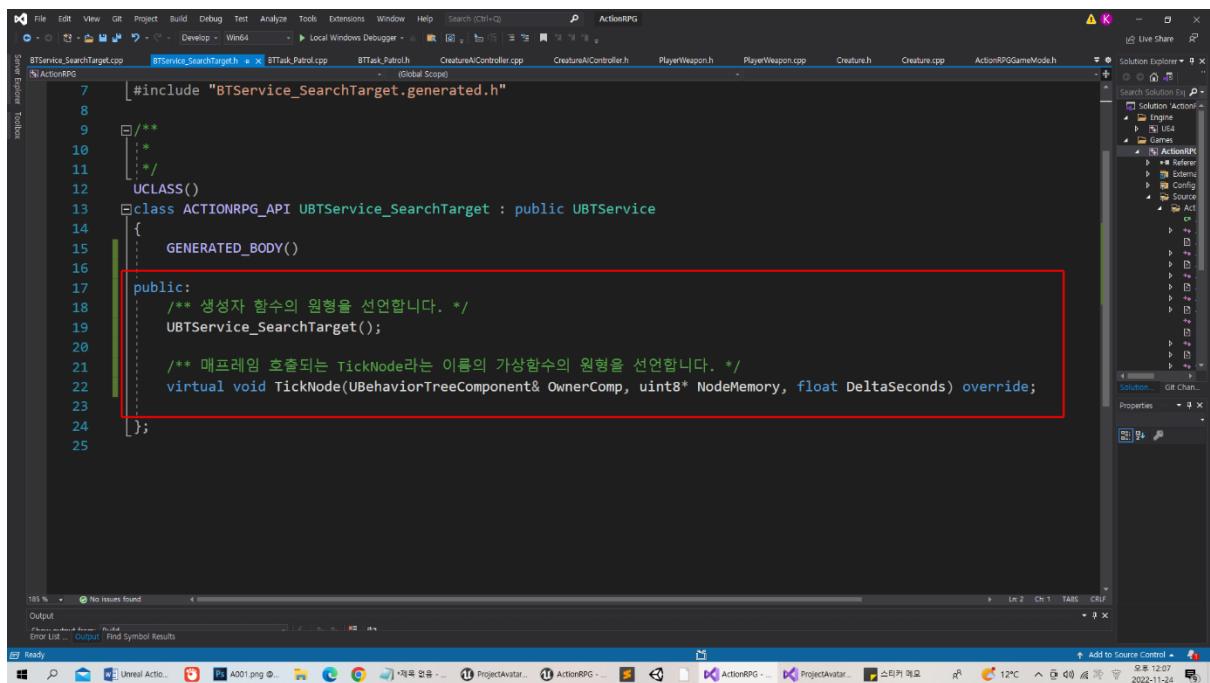


BTService를 부모 클래스로 상속받는 BTService_SearchTarget라는 이름의 클래스를 정의해 주도록 합니다.

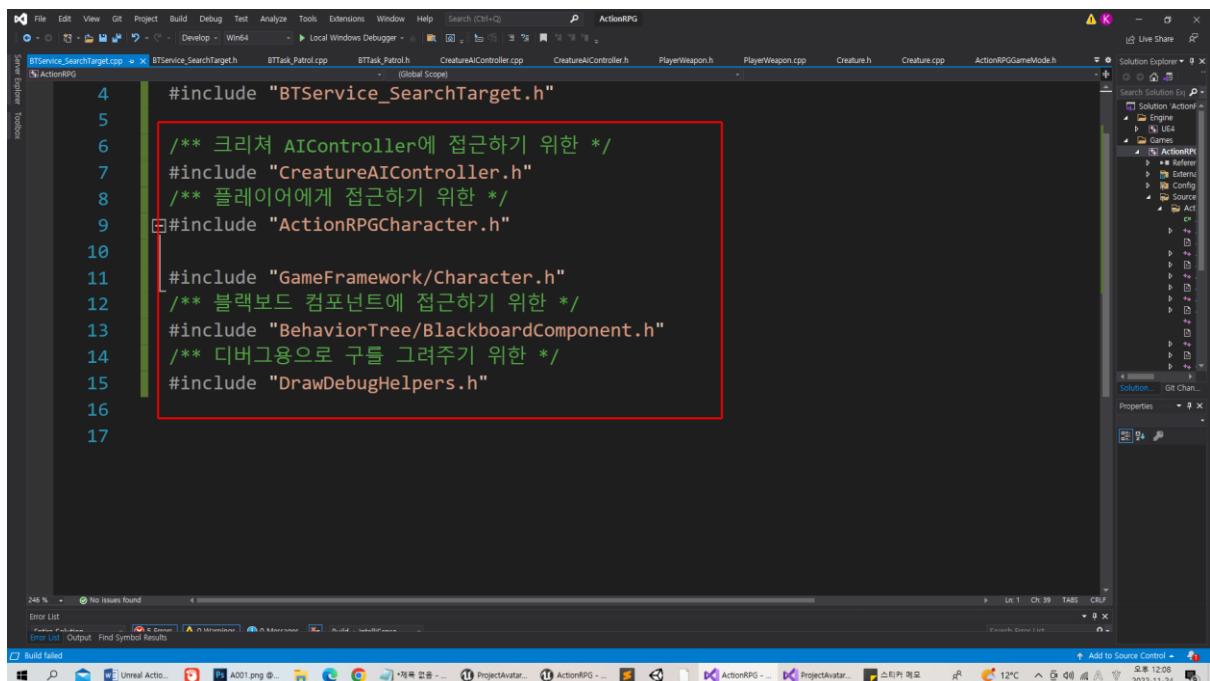




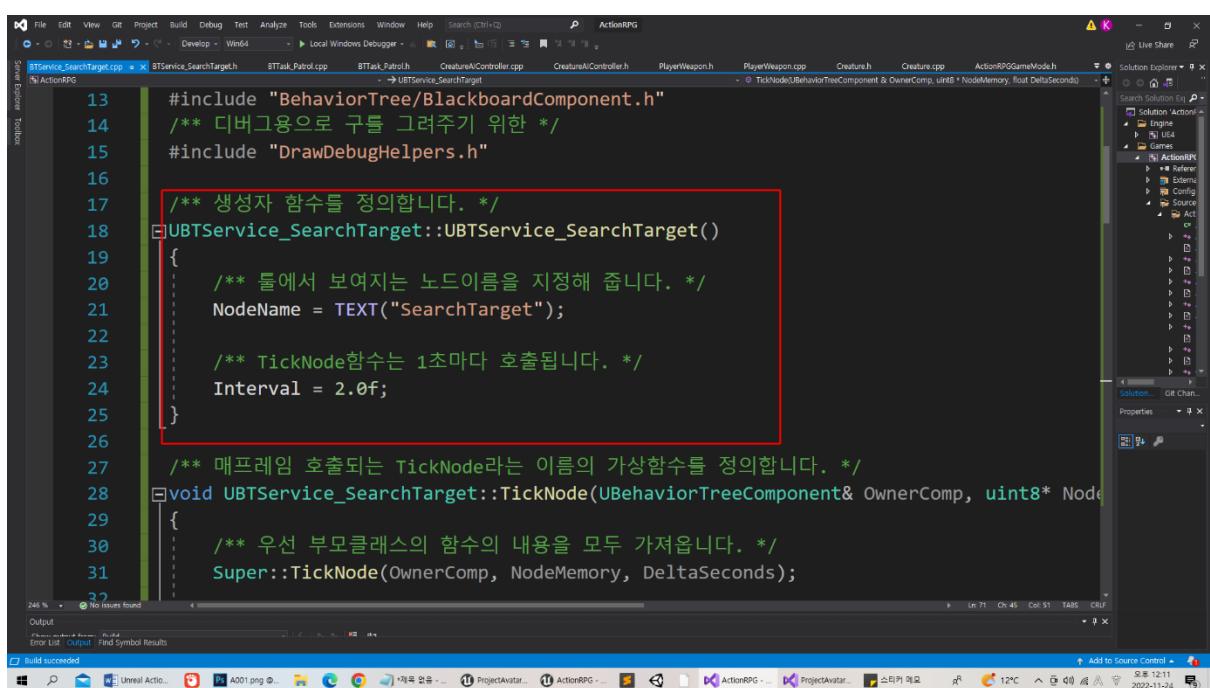
필요한 함수의 원형을 선언합니다.



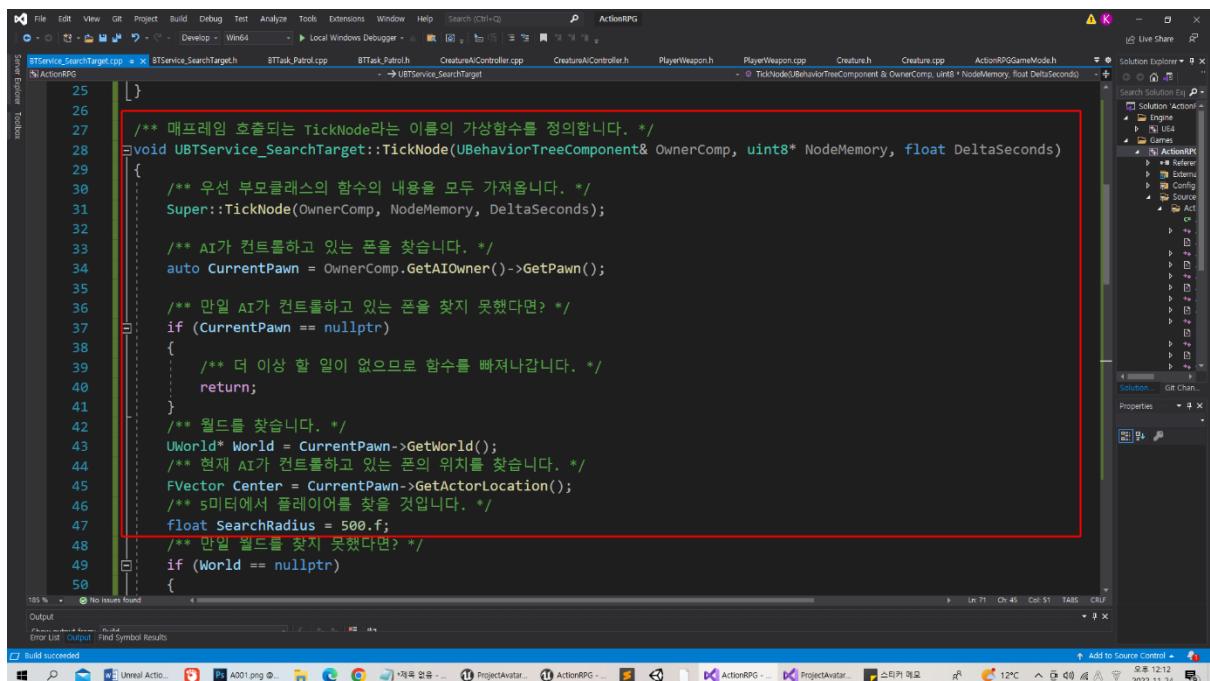
구현해 주도록 합니다.



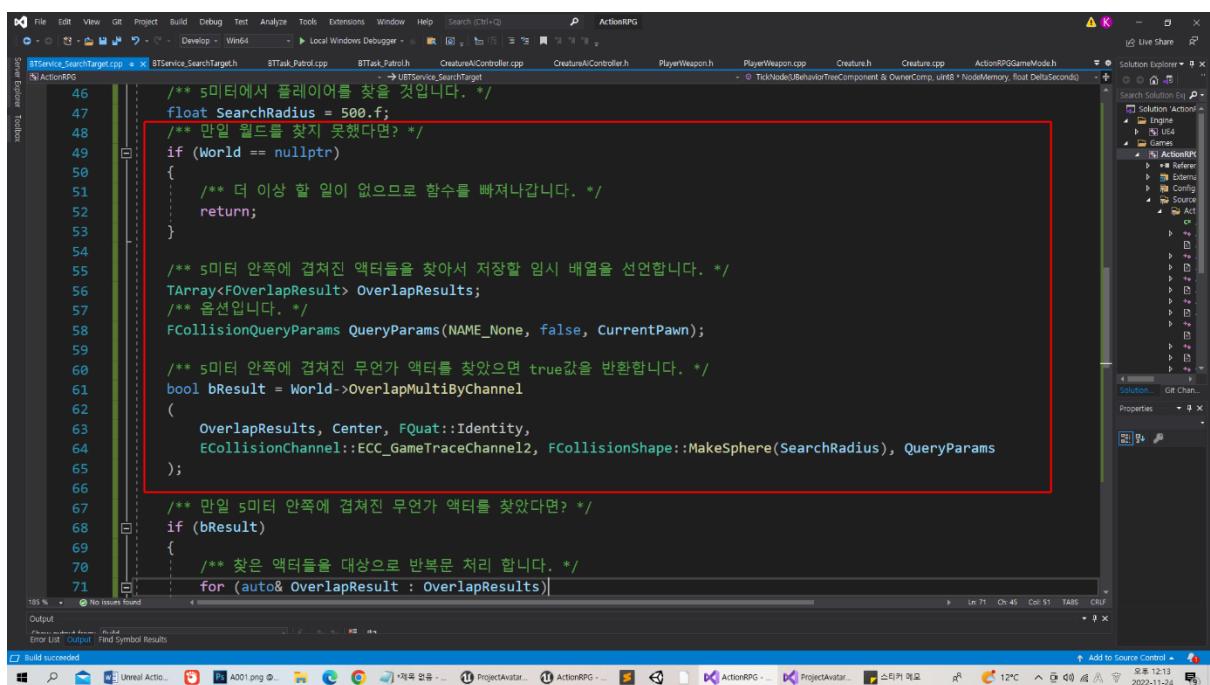
```
4 #include "BTService_SearchTarget.h"
5
6 /** 크리쳐 AIController에 접근하기 위한 */
7 #include "CreatureAIController.h"
8 /** 플레이어에게 접근하기 위한 */
9 #include "ActionRPGCharacter.h"
10
11 #include "GameFramework/Character.h"
12 /** 블랙보드 컴포넌트에 접근하기 위한 */
13 #include "BehaviorTree/BlackboardComponent.h"
14 /** 디버그용으로 구를 그려주기 위한 */
15 #include "DrawDebugHelpers.h"
16
17
```



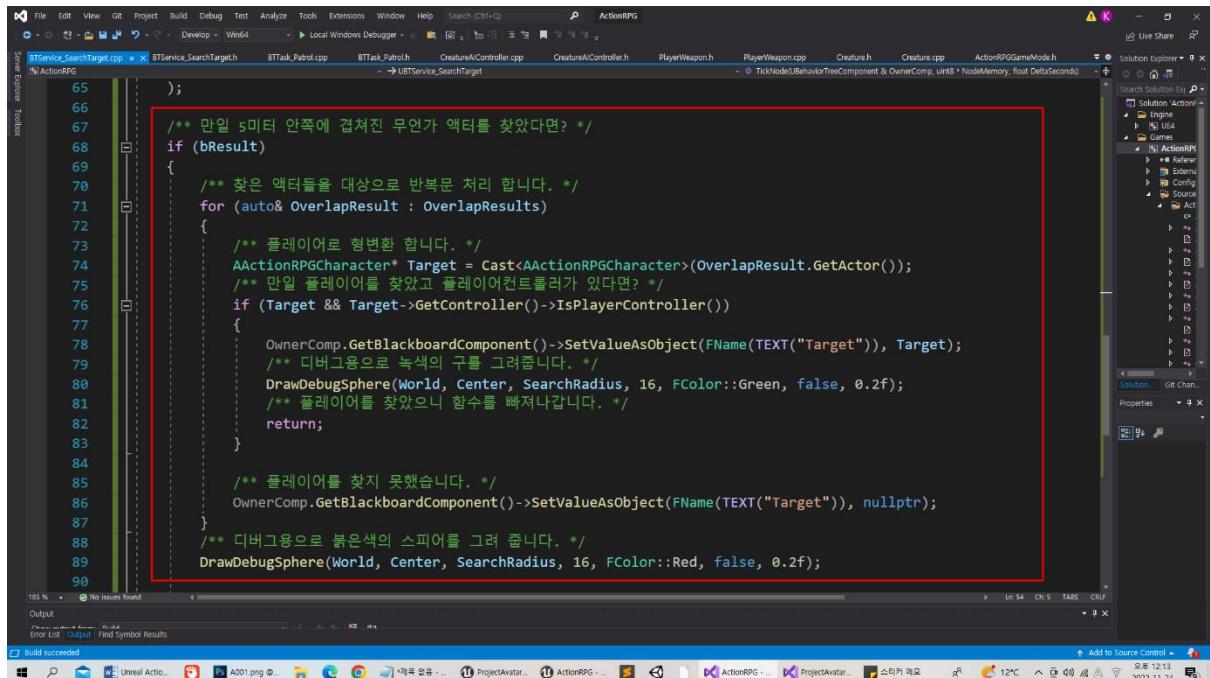
```
13 #include "BehaviorTree/BlackboardComponent.h"
14 /** 디버그용으로 구를 그려주기 위한 */
15 #include "DrawDebugHelpers.h"
16
17 /** 생성자 함수를 정의합니다. */
18 UBTService_SearchTarget::UBTService_SearchTarget()
19 {
20     /** 둘에서 보여지는 노드이름을 지정해 줍니다. */
21     NodeName = TEXT("SearchTarget");
22
23     /** TickNode함수는 1초마다 호출됩니다. */
24     Interval = 2.0f;
25 }
26
27 /** 매프레임 호출되는 TickNode라는 이름의 가상함수를 정의합니다. */
28 void UBTService_SearchTarget::TickNode(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory, float DeltaSeconds)
29 {
30     /** 우선 부모클래스의 함수의 내용을 모두 가져옵니다. */
31     Super::TickNode(OwnerComp, NodeMemory, DeltaSeconds);
32 }
```



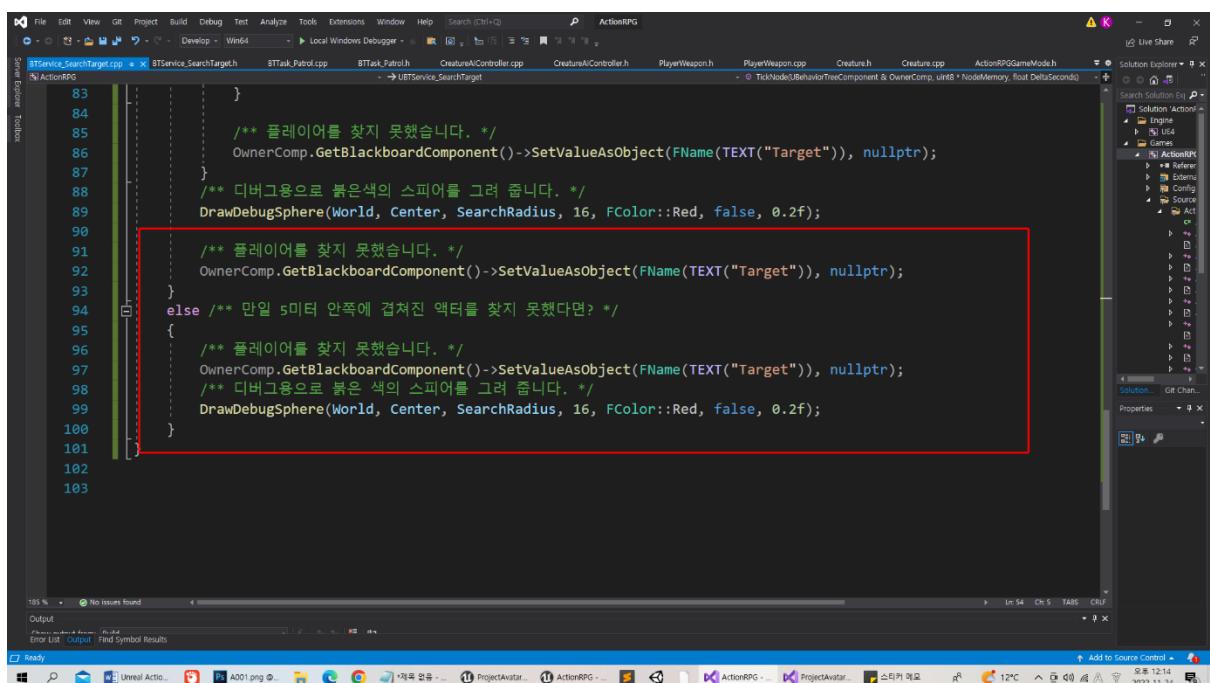
```
25
26
27 /*** 매프레이 흐출되는 TickNode라는 이름의 가상함수를 정의합니다. */
28 void UBTService_SearchTarget::TickNode(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory, float DeltaSeconds)
29 {
30     /* 우선 부모클래스의 함수의 내용을 모두 가져옵니다. */
31     Super::TickNode(OwnerComp, NodeMemory, DeltaSeconds);
32
33     /* AI가 컨트롤하고 있는 폰을 찾습니다. */
34     auto CurrentPawn = OwnerComp.GetAIOwner()->GetPawn();
35
36     /* 만일 AI가 컨트롤하고 있는 폰을 찾지 못했다면? */
37     if (CurrentPawn == nullptr)
38     {
39         /* 더 이상 할 일이 없으므로 함수를 빠져나갑니다. */
40         return;
41     }
42
43     /* 월드를 찾습니다. */
44     Uworld* World = CurrentPawn->GetWorld();
45     /* 현재 AI가 컨트롤하고 있는 폰의 위치를 찾습니다. */
46     FVector Center = CurrentPawn->GetActorLocation();
47     /* 5미터에서 플레이어를 찾을 것입니다. */
48     float SearchRadius = 500.f;
49     /* 만일 월드를 찾지 못했다면? */
50     if (World == nullptr)
51     {
52         /* 더 이상 할 일이 없으므로 함수를 빠져나갑니다. */
53         return;
54     }
55
56     /* 5미터 안쪽에 겹쳐진 액터들을 찾아서 저장할 임시 배열을 선언합니다. */
57     TArray<FOverlapResult> OverlapResults;
58     /* 옵션입니다. */
59     FCollisionQueryParams QueryParams(NAME_None, false, CurrentPawn);
60
61     /* 5미터 안쪽에 겹쳐진 무언가 액터를 찾았으면 true값을 반환합니다. */
62     bool bResult = World->OverlapMultiByChannel
63     (
64         OverlapResults, Center, FQuat::Identity,
65         ECollisionChannel::ECC_GameTraceChannel2, FCollisionShape::MakeSphere(SearchRadius), QueryParams
66     );
67
68     /* 만일 5미터 안쪽에 겹쳐진 무언가 액터를 찾았다면? */
69     if (bResult)
70     {
71         /* 찾은 액터들을 대상으로 반복문 처리 합니다. */
72         for (auto& OverlapResult : OverlapResults)
73         {
74             /* 폰과 겹친 액터의 위치를 찾습니다. */
75             FVector TargetLocation = OverlapResult.Origin;
76
77             /* 폰과 겹친 액터의 위치를 폰의 위치로 업데이트합니다. */
78             CurrentPawn->SetActorLocation(TargetLocation);
79
80             /* 폰과 겹친 액터를 찾았습니다. */
81             *NodeMemory = 1;
82         }
83     }
84 }
```



```
46     /* 5미터에서 플레이어를 찾을 것입니다. */
47     float SearchRadius = 500.f;
48     /* 만일 월드를 찾지 못했다면? */
49     if (World == nullptr)
50     {
51         /* 더 이상 할 일이 없으므로 함수를 빠져나갑니다. */
52         return;
53     }
54
55     /* 5미터 안쪽에 겹쳐진 액터들을 찾아서 저장할 임시 배열을 선언합니다. */
56     TArray<FOverlapResult> OverlapResults;
57     /* 옵션입니다. */
58     FCollisionQueryParams QueryParams(NAME_None, false, CurrentPawn);
59
60     /* 5미터 안쪽에 겹쳐진 무언가 액터를 찾았으면 true값을 반환합니다. */
61     bool bResult = World->OverlapMultiByChannel
62     (
63         OverlapResults, Center, FQuat::Identity,
64         ECollisionChannel::ECC_GameTraceChannel2, FCollisionShape::MakeSphere(SearchRadius), QueryParams
65     );
66
67     /* 만일 5미터 안쪽에 겹쳐진 무언가 액터를 찾았다면? */
68     if (bResult)
69     {
70         /* 찾은 액터들을 대상으로 반복문 처리 합니다. */
71         for (auto& OverlapResult : OverlapResults)
72         {
73             /* 폰과 겹친 액터의 위치를 찾습니다. */
74             FVector TargetLocation = OverlapResult.Origin;
75
76             /* 폰과 겹친 액터의 위치를 폰의 위치로 업데이트합니다. */
77             CurrentPawn->SetActorLocation(TargetLocation);
78
79             /* 폰과 겹친 액터를 찾았습니다. */
80             *NodeMemory = 1;
81         }
82     }
83 }
```

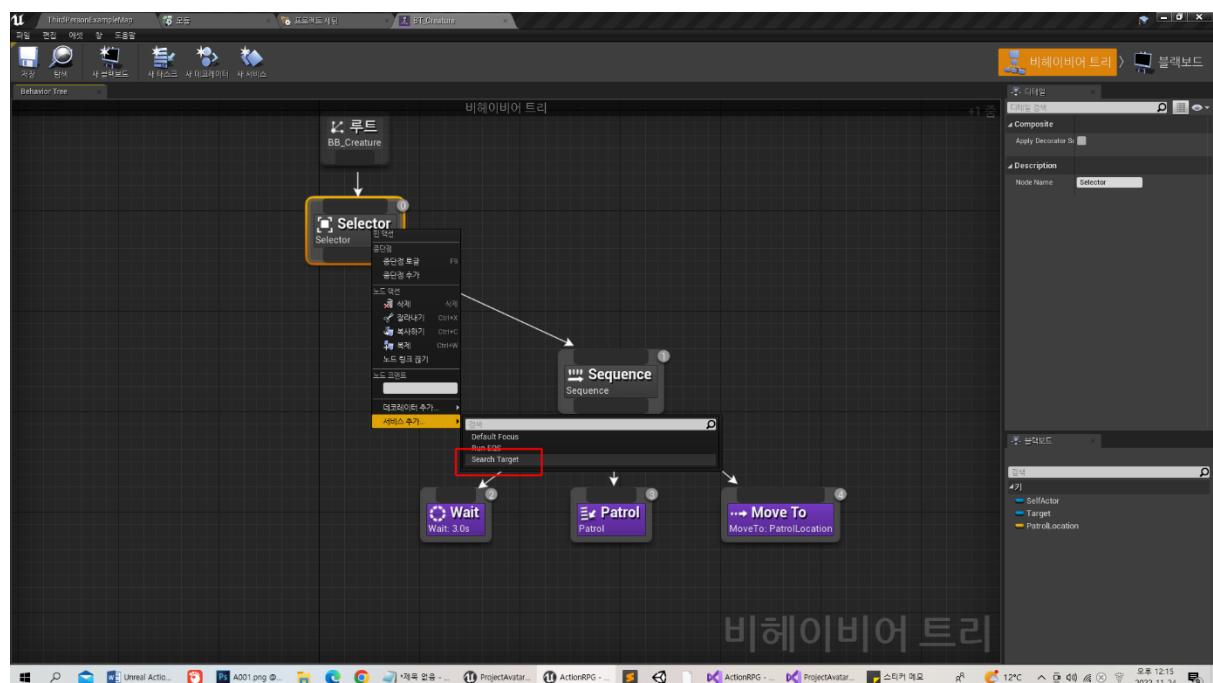
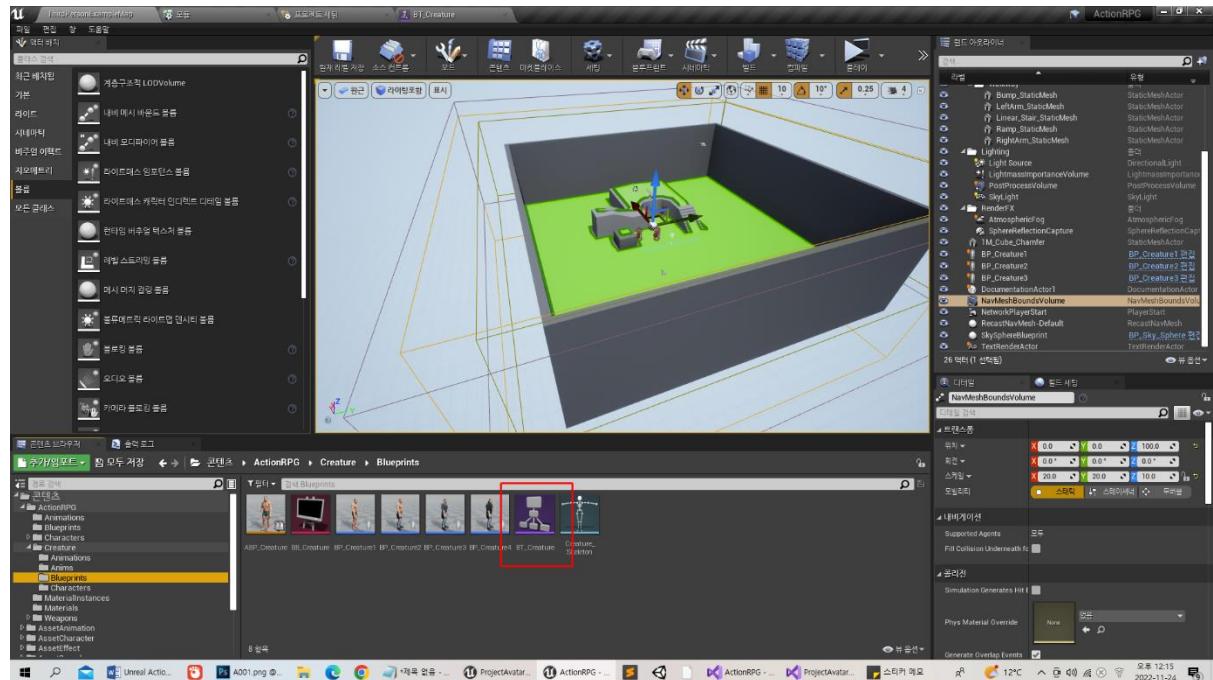


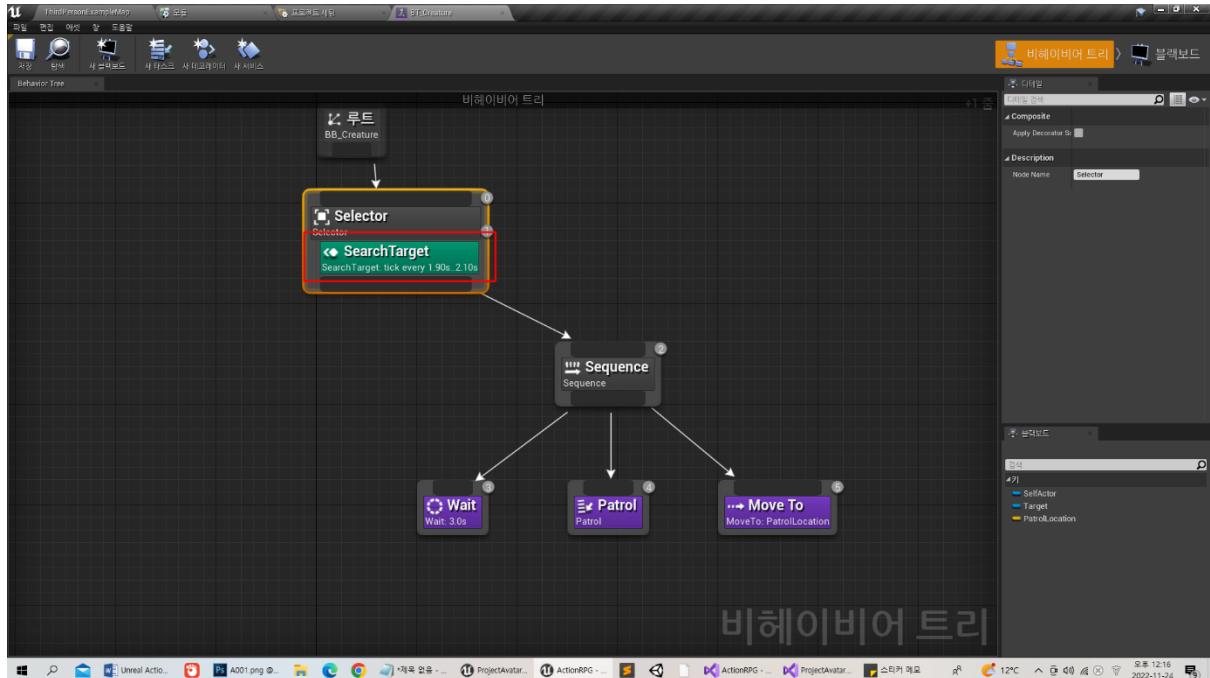
```
55     );
56
57     /* 만일 5미터 안쪽에 겹쳐진 무언가 액터를 찾았다면? */
58     if (bResult)
59     {
60         /* 찾은 액터들을 대상으로 반복문 처리 합니다. */
61         for (auto& OverlapResult : OverlapResults)
62         {
63             /* 플레이어로 형변환 합니다. */
64             AActionRPGCharacter* Target = Cast<AActionRPGCharacter>(OverlapResult.GetActor());
65             /* 만일 플레이어를 찾았고 플레이어컨트를려가 있다면? */
66             if (Target && Target->GetController()->IsPlayerController())
67             {
68                 OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), Target);
69                 /* 디버그용으로 녹색의 구를 그려줍니다. */
70                 DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Green, false, 0.2f);
71                 /* 플레이어를 찾았으니 할수를 빼져나갑니다. */
72                 return;
73             }
74
75             /* 플레이어를 찾지 못했습니다. */
76             OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
77         }
78
79         /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
80         DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
81     }
82
83     /* 플레이어를 찾지 못했습니다. */
84     OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
85
86     /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
87     DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
88
89     /* 플레이어를 찾지 못했습니다. */
90     OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
91
92     /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
93     DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
94
95     /* 만일 5미터 안쪽에 겹쳐진 액터를 찾지 못했다면? */
96     else
97     {
98         /* 플레이어를 찾지 못했습니다. */
99         OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
100        /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
101        DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
102    }
103
104
105 }
```



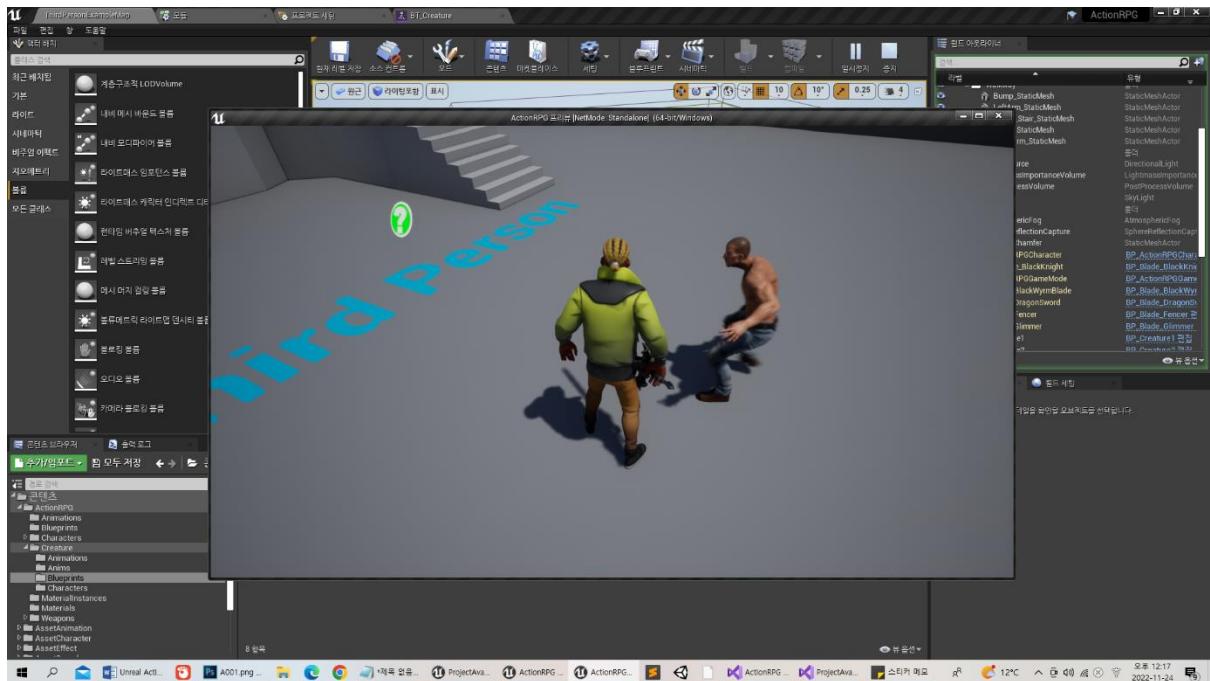
```
83     }
84
85     /* 플레이어를 찾지 못했습니다. */
86     OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
87
88     /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
89     DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
90
91     /* 플레이어를 찾지 못했습니다. */
92     OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
93
94     /* 만일 5미터 안쪽에 겹쳐진 액터를 찾지 못했다면? */
95     else
96     {
97         /* 플레이어를 찾지 못했습니다. */
98         OwnerComp.GetBlackboardComponent()->SetValueAsObject(FName(TEXT("Target")), nullptr);
99         /* 디버그용으로 붉은색의 스피어를 그려 줍니다. */
100        DrawDebugSphere(World, Center, SearchRadius, 16, FColor::Red, false, 0.2f);
101    }
102
103
104
105 }
```

비헤이비어 트리에서 서비스로 등록해 줍니다. 1초마다 플레이어를 찾습니다.

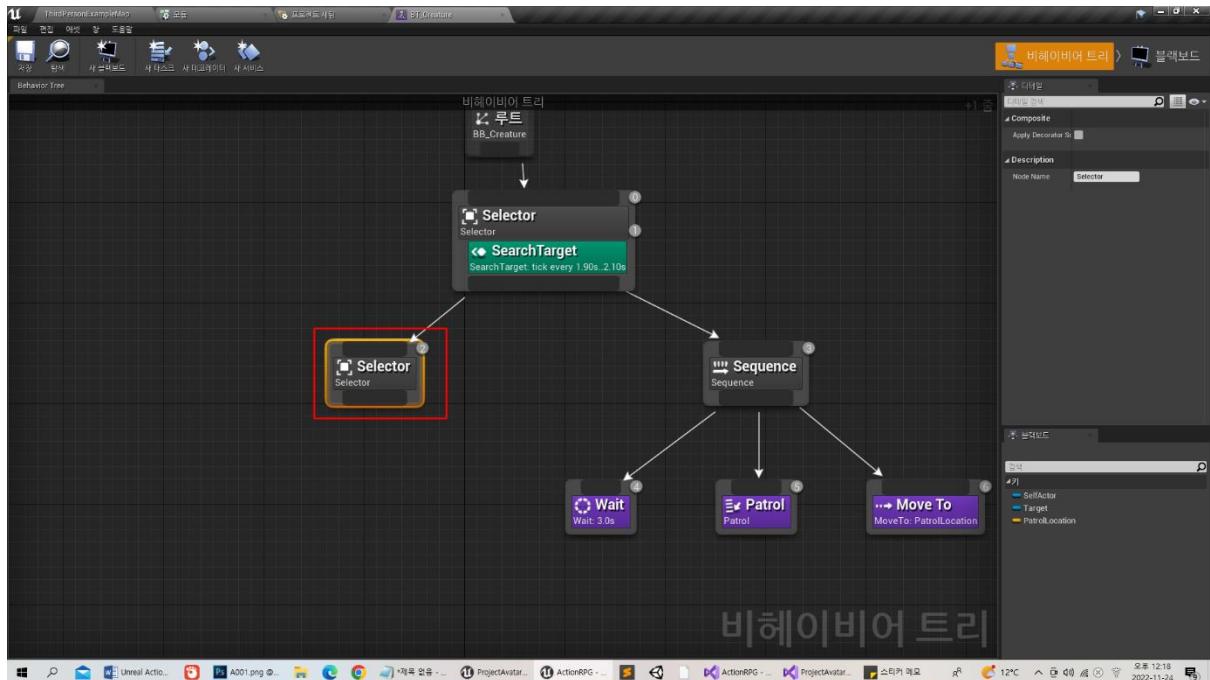




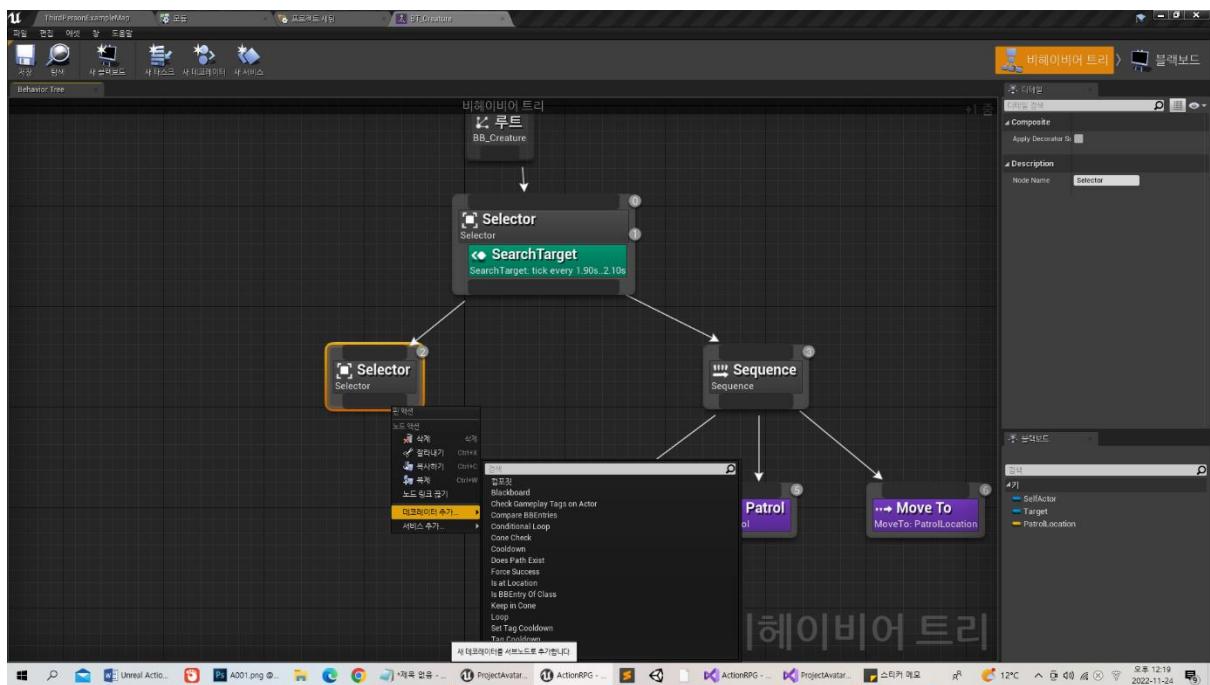
플레이를 해서 결과를 확인합니다.

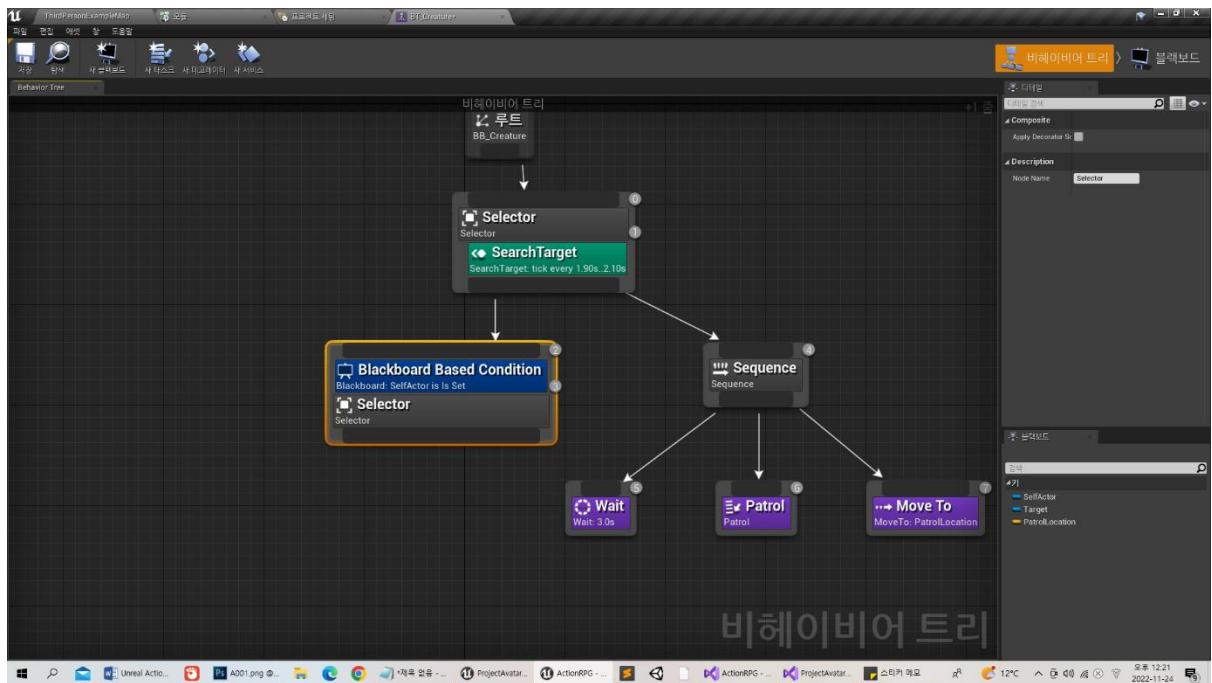
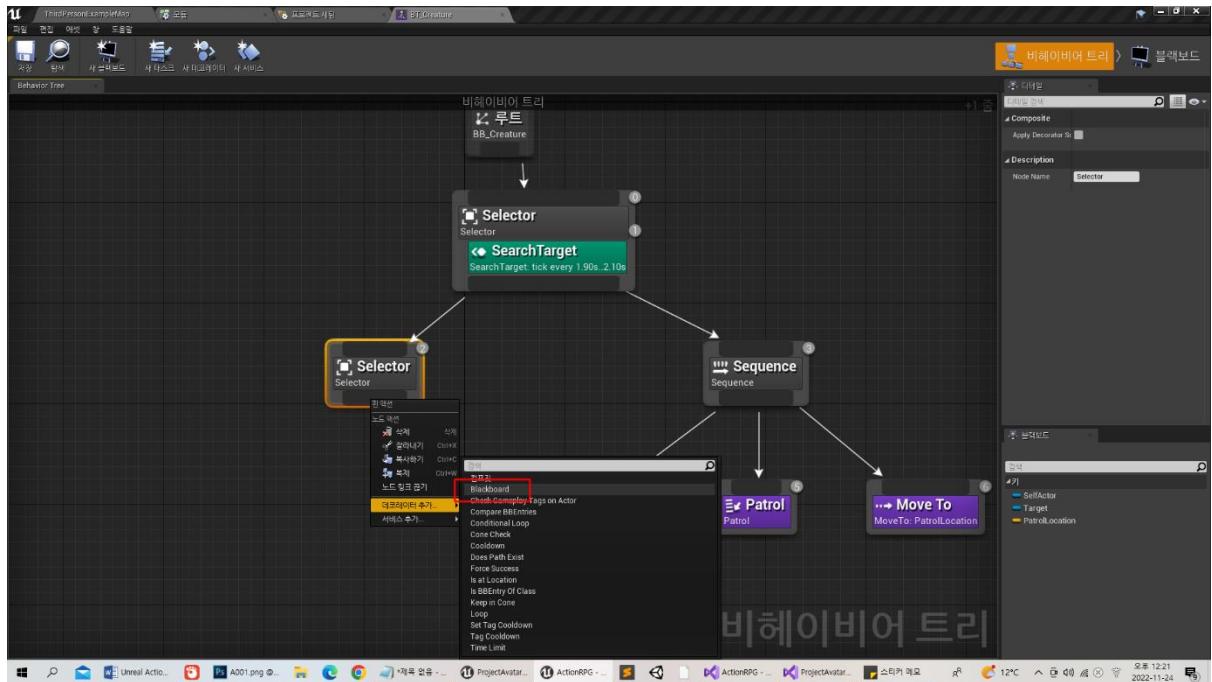


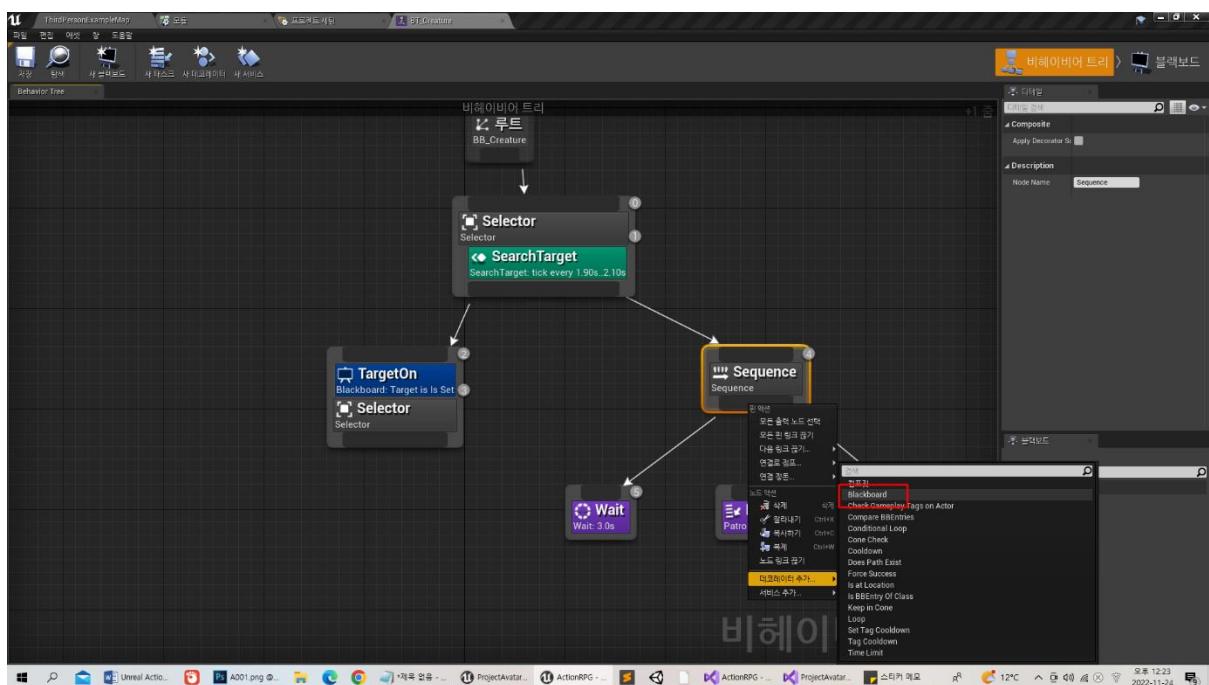
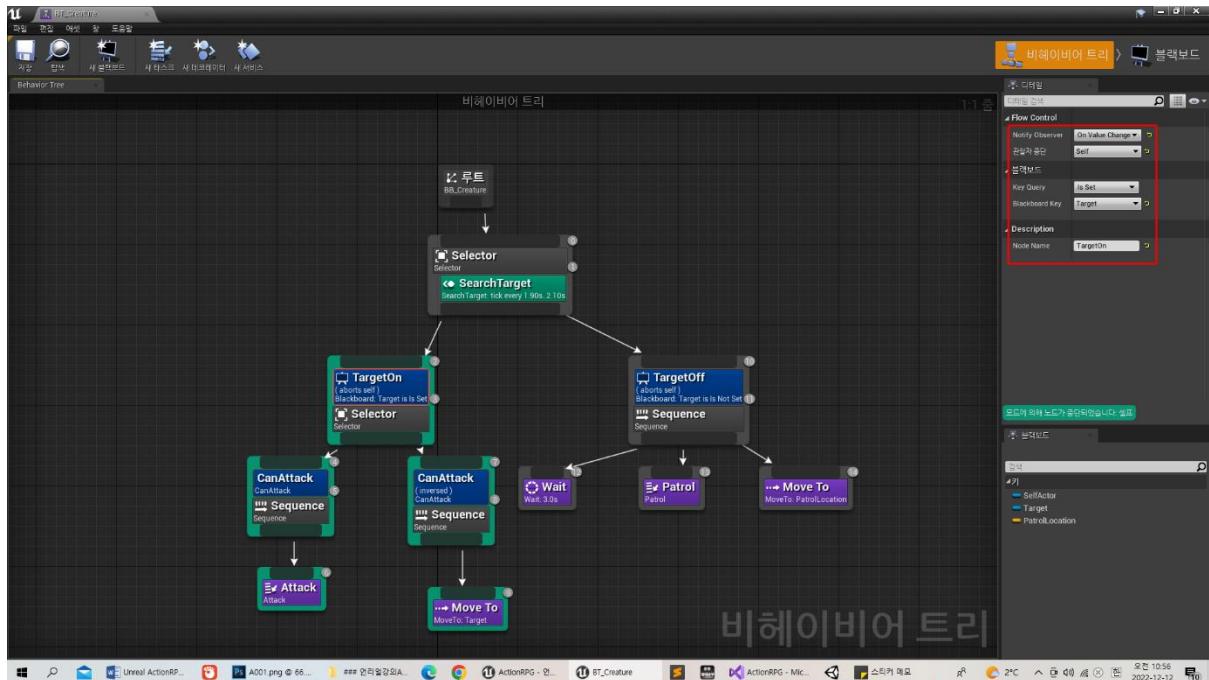
플레이어를 발견했을 때의 처리를 위해서 selector노드를 추가해 줍니다.

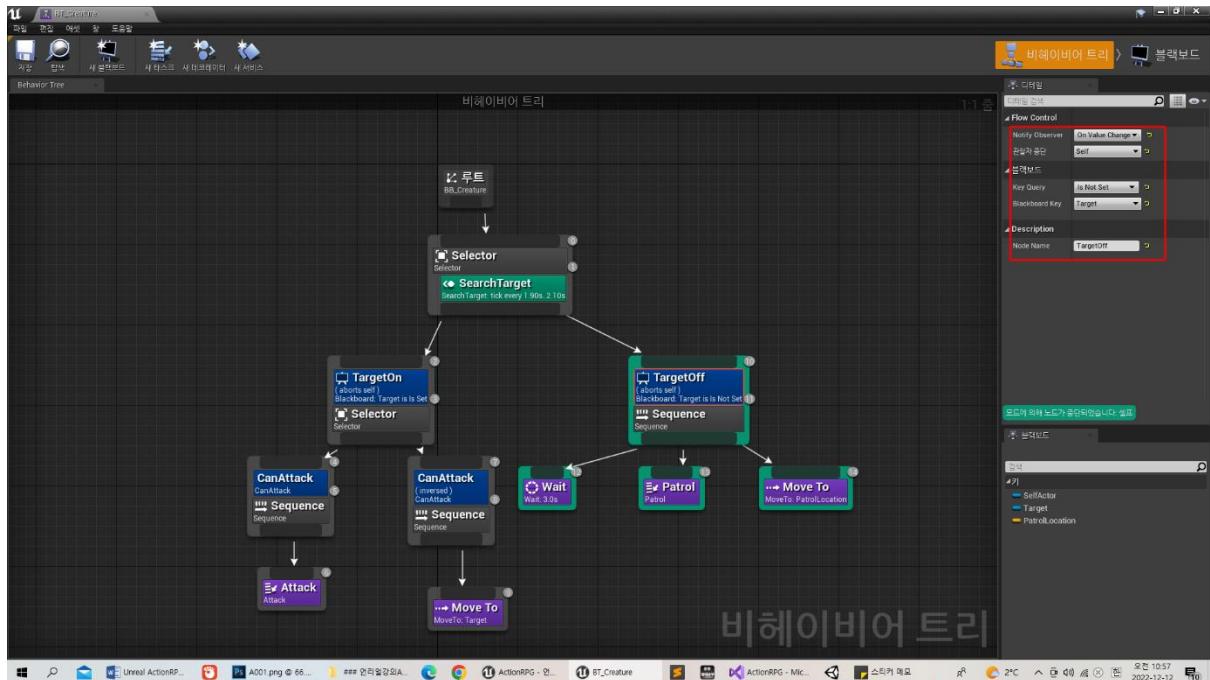


플레이어를 발견했는지, 발견하지 못했는지를 판별하기 위해서 특정 조건을 판별하기 위해서 데코레이터를 추가합니다.

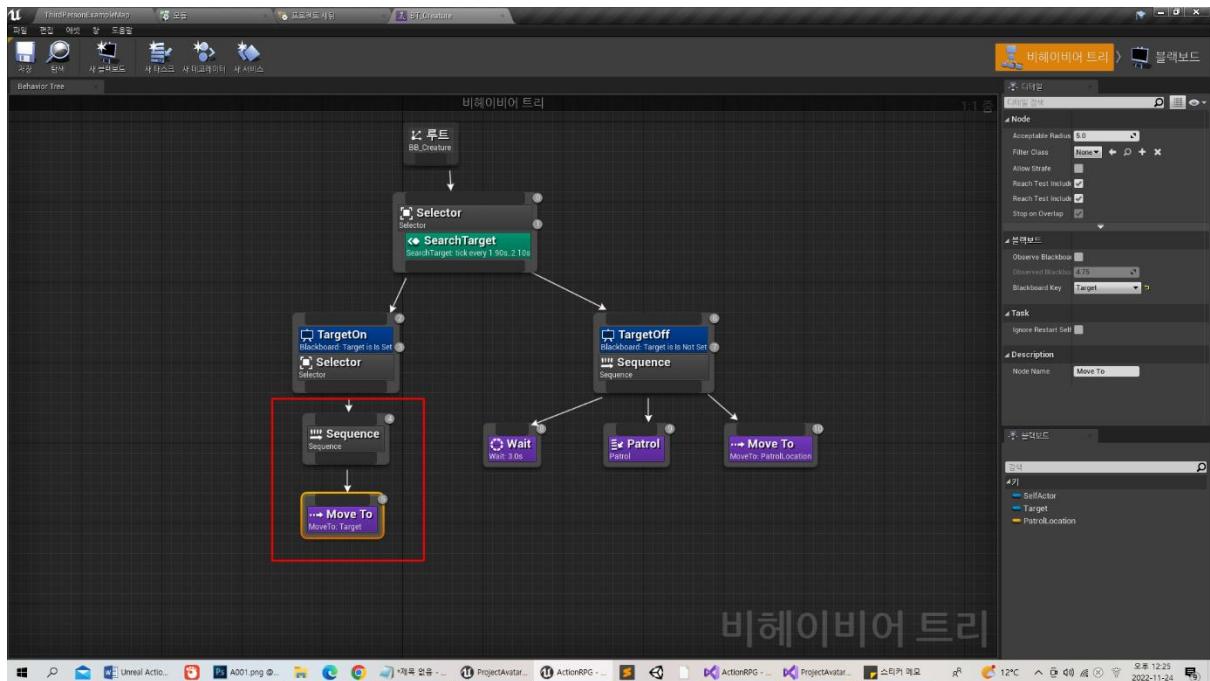




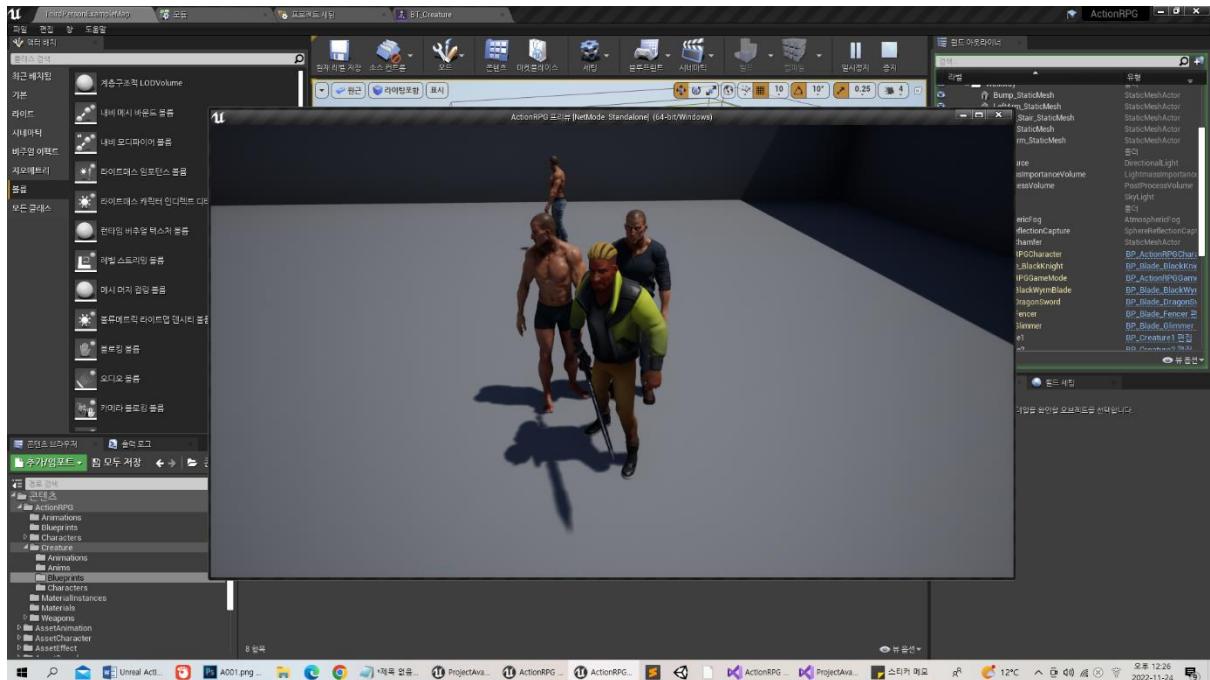




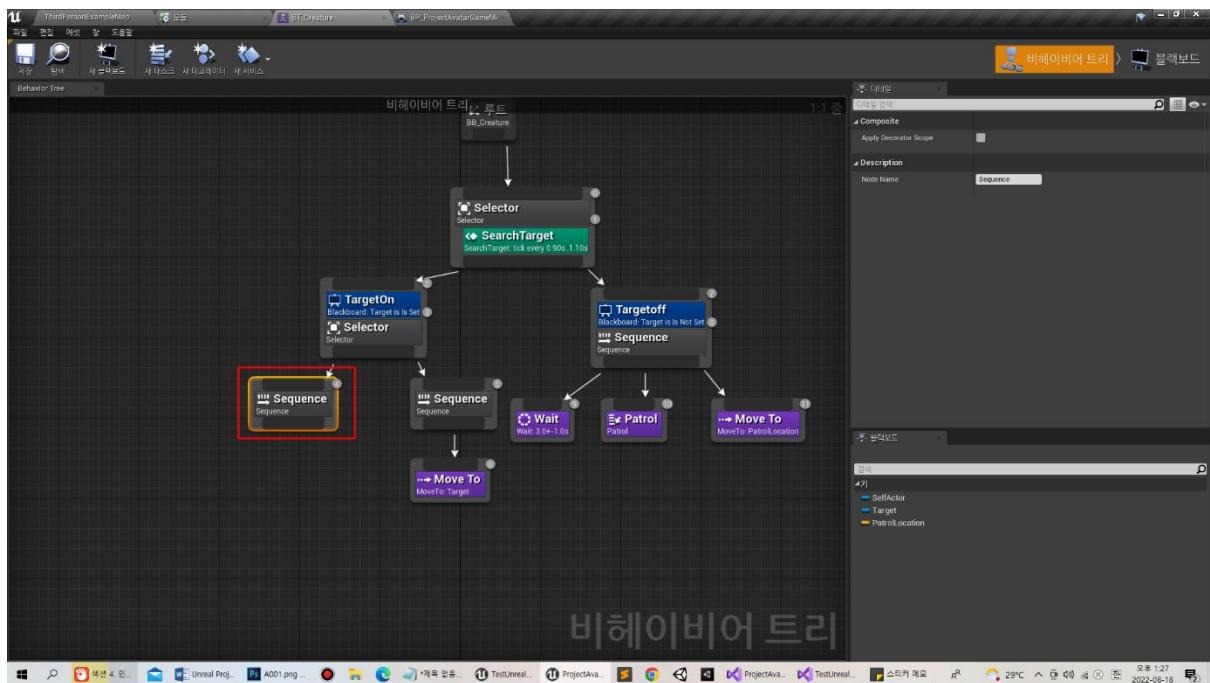
타겟을 향해 이동하도록 해 줍니다.



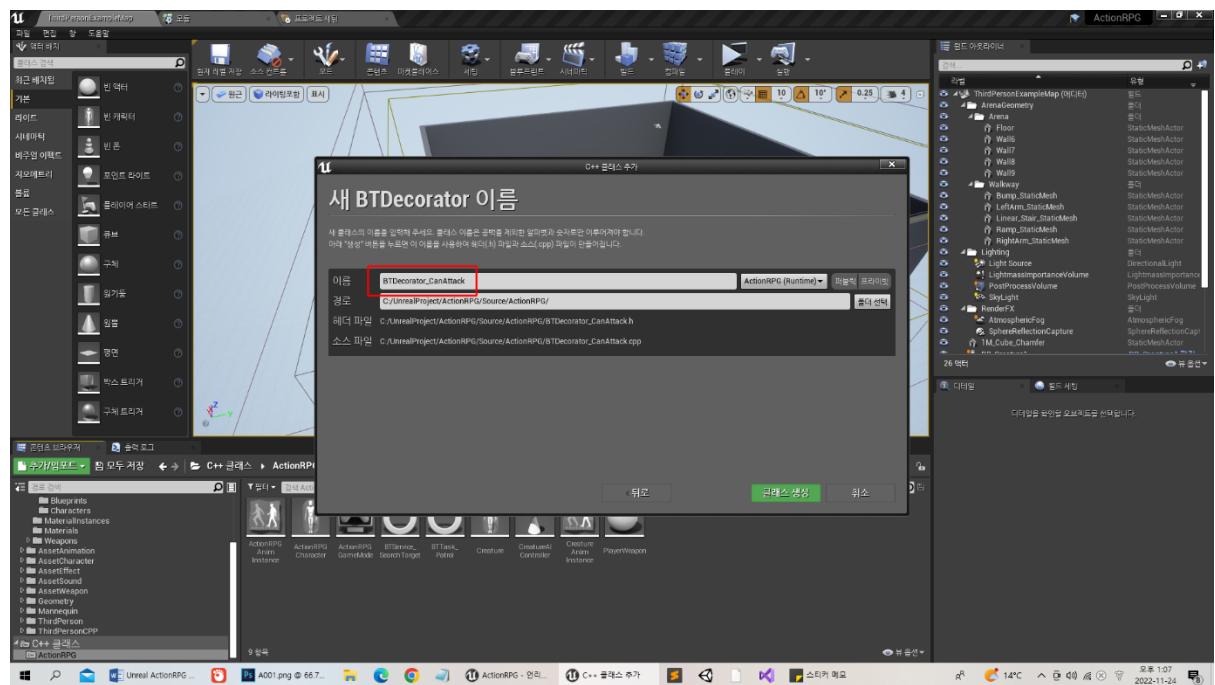
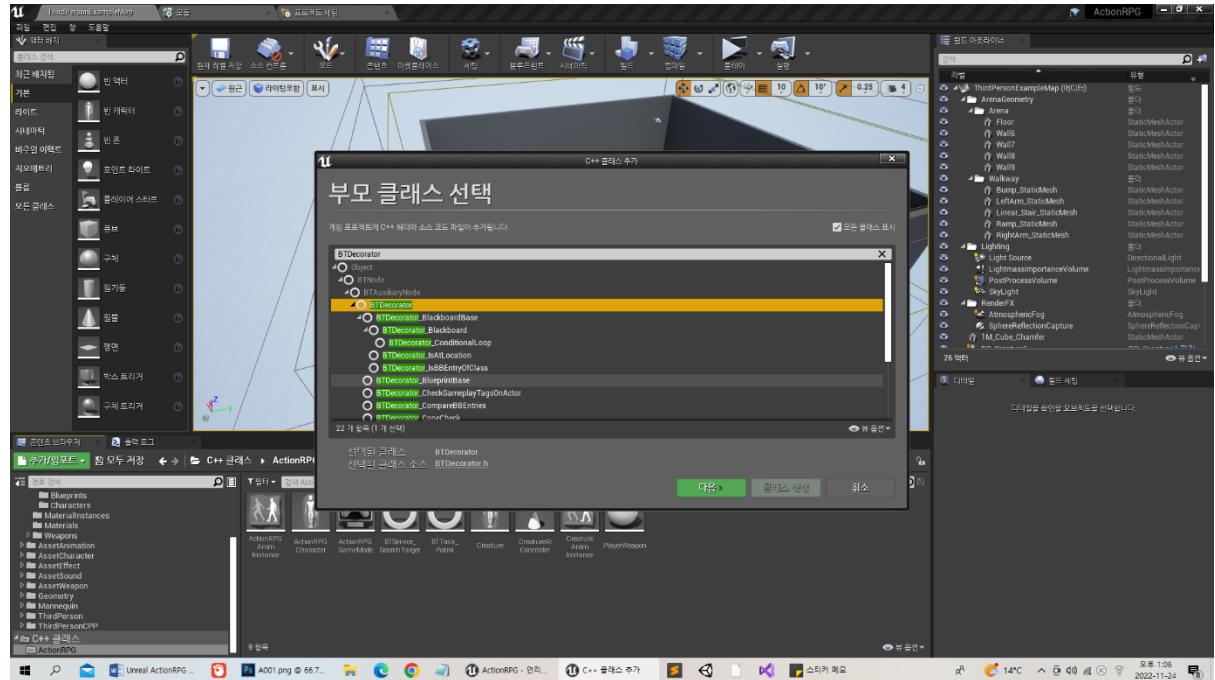
플레이를 해서 결과를 확인합니다.

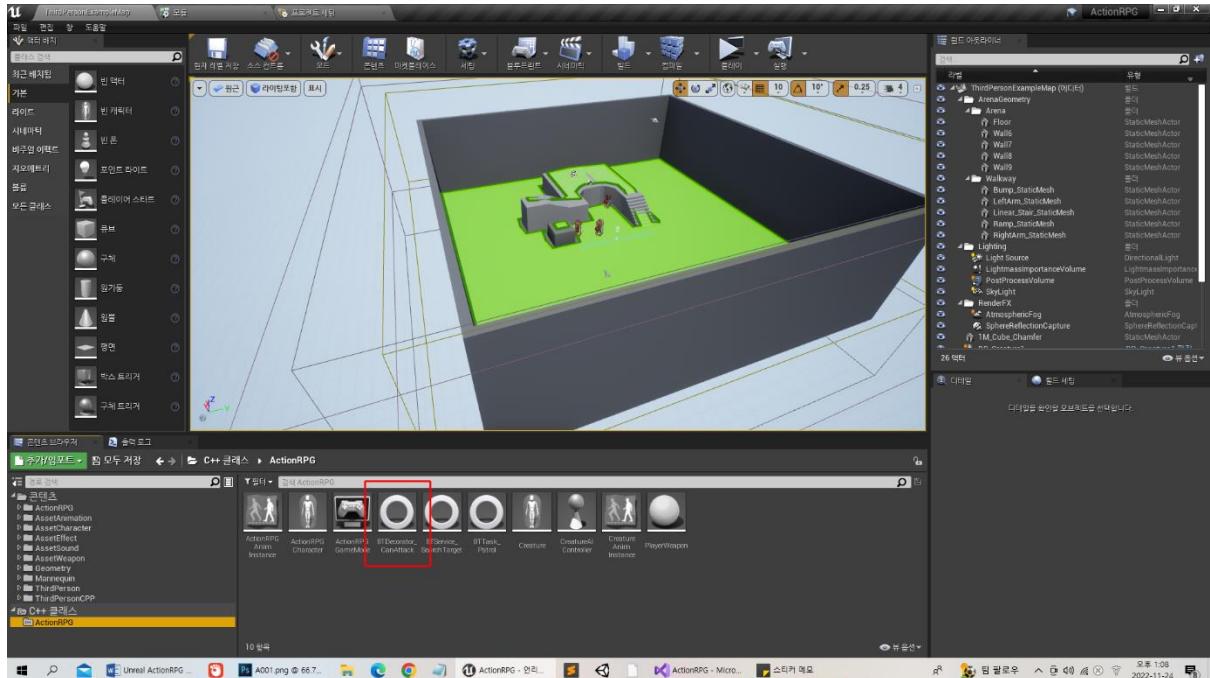


공격범위 안에 들어왔을 때 플레이어를 따라올지, 따라오지 않을 때 분기를 해 줍니다.

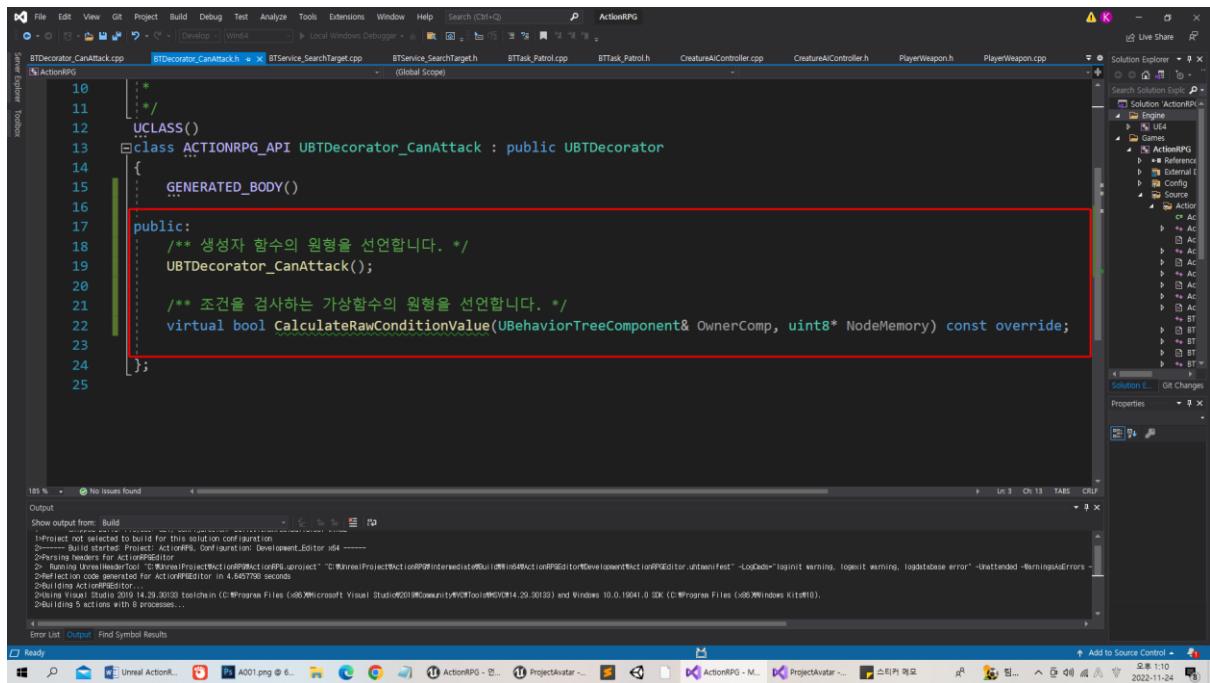


공격을 할 수 있는지, 없는지에 해당하는 데코레이션을 정의해 주도록 합니다. BTDecorator를 상속받는 BTDecoratorCanAttack이라는 클래스를 정의해 줍니다.

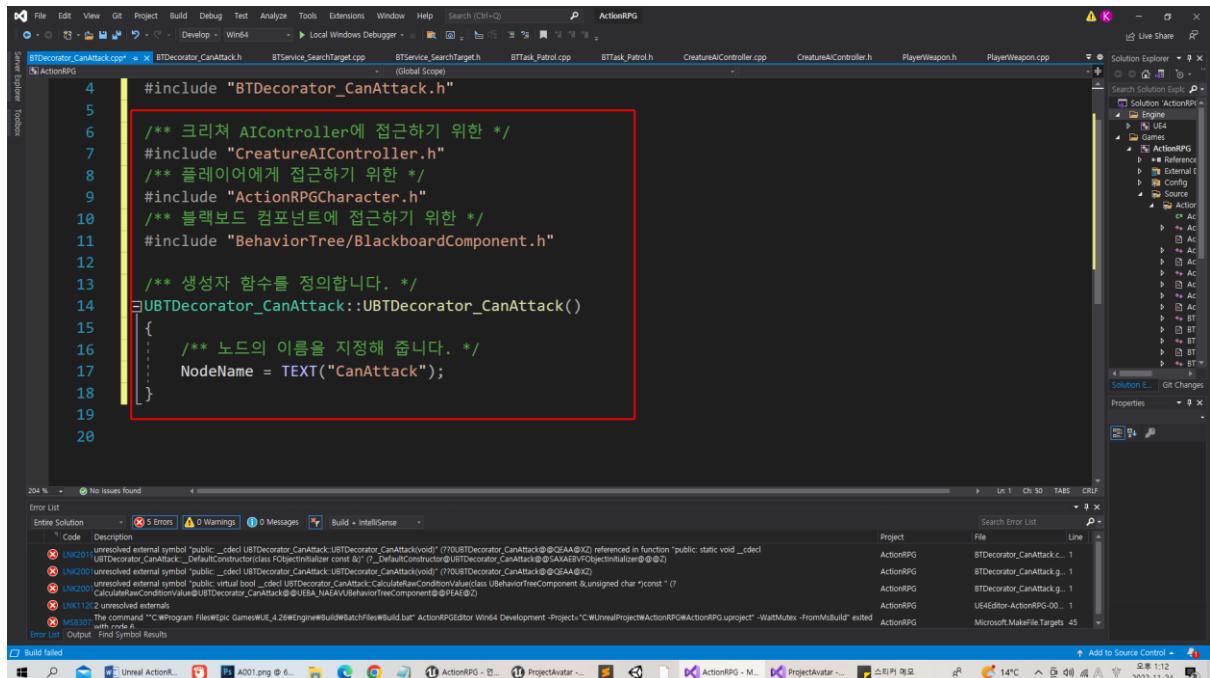




필요한 함수의 원형을 선언합니다.



구현해 주도록 합니다.



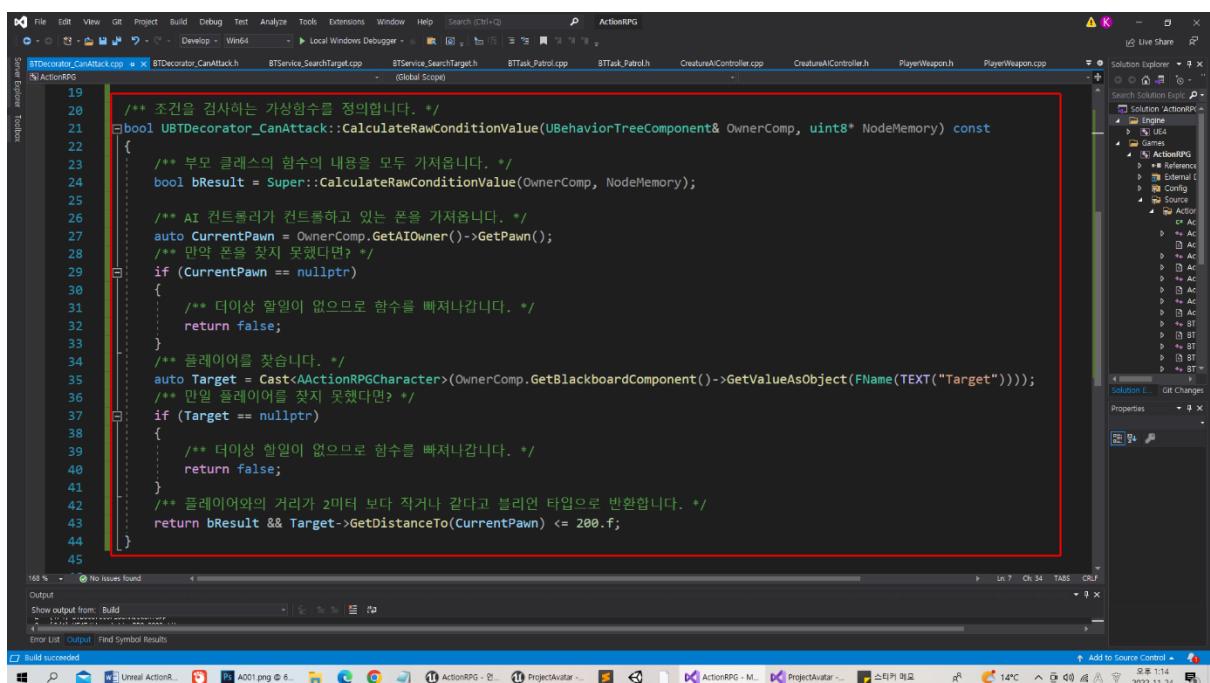
```
4 #include "BTDecorator_CanAttack.h"
5
6     /** 크리쳐 AIController에 접근하기 위한 */
7     #include "CreatureAIController.h"
8     /** 플레이어에게 접근하기 위한 */
9     #include "ActionRPGCharacter.h"
10    /** 블랙보드 컴포넌트에 접근하기 위한 */
11    #include "BehaviorTree/BlackboardComponent.h"
12
13    /** 생성자 함수를 정의합니다. */
14    UBTDecorator_CanAttack::UBTDecorator_CanAttack()
15    {
16        /** 노드의 이름을 지정해 줍니다. */
17        NodeName = TEXT("CanAttack");
18    }
```

Unresolved external symbol "public __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ) referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ)

Unresolved external symbol "public: virtual bool __cdecl UBTDecorator_CanAttack::CalculateRawConditionValue(class UBehaviorTreeComponent &OwnerComp, unsigned char *const)" (??_CUBTDecorator_CanAttack@UBTDecorator_CanAttack@@UEA8UBehaviorTreeComponent@0@PAC@Z)

Unresolved external symbol "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ) referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ)

The command "C:\Program Files\Epic Games\UE_4.26\Engine\BuildBatchFiles\Build.bat" ActionRPGEditor Win64 Development -fProject="..\UnrealProject\ActionRPG\ActionRPG.uproject" -WaitMutex -fFromMsBuild" exited with code 1.



```
19
20
21     /** 조건을 검사하는 가상함수를 정의합니다. */
22     bool UBTDecorator_CanAttack::CalculateRawConditionValue(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory) const
23     {
24         /** 부모 클래스의 함수 내용을 모두 가져옵니다. */
25         bool bResult = Super::CalculateRawConditionValue(OwnerComp, NodeMemory);
26
27         /** AI 컨트롤러가 컨트롤하고 있는 폰을 가져옵니다. */
28         auto CurrentPawn = OwnerComp.GetAIController()->GetPawn();
29
30         /** 만약 폰을 찾지 못했다면? */
31         if (CurrentPawn == nullptr)
32         {
33             /** 티이상 할일이 없으므로 함수를 빠져나갑니다. */
34             return false;
35         }
36
37         /** 플레이어를 찾습니다. */
38         auto Target = Cast<ActionRPGCharacter>(OwnerComp.GetBlackboardComponent()->GetValueAsObject(FName(TEXT("Target"))));
39
40         /** 만약 플레이어를 찾지 못했다면? */
41         if (Target == nullptr)
42         {
43             /** 더이상 할일이 없으므로 함수를 빠져나갑니다. */
44             return false;
45         }
46
47         /** 플레이어와의 거리가 2미터 보다 작거나 같다고 블리언 타입으로 반환합니다. */
48         return bResult && Target->GetDistanceTo(CurrentPawn) <= 200.f;
49     }
```

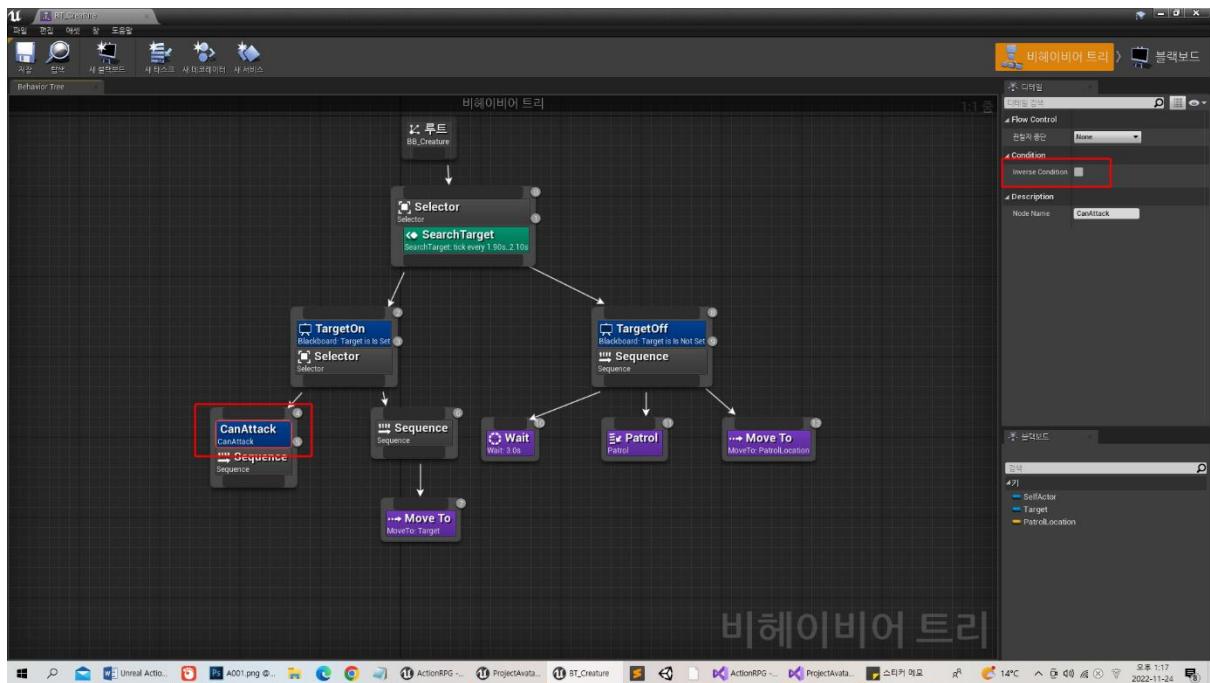
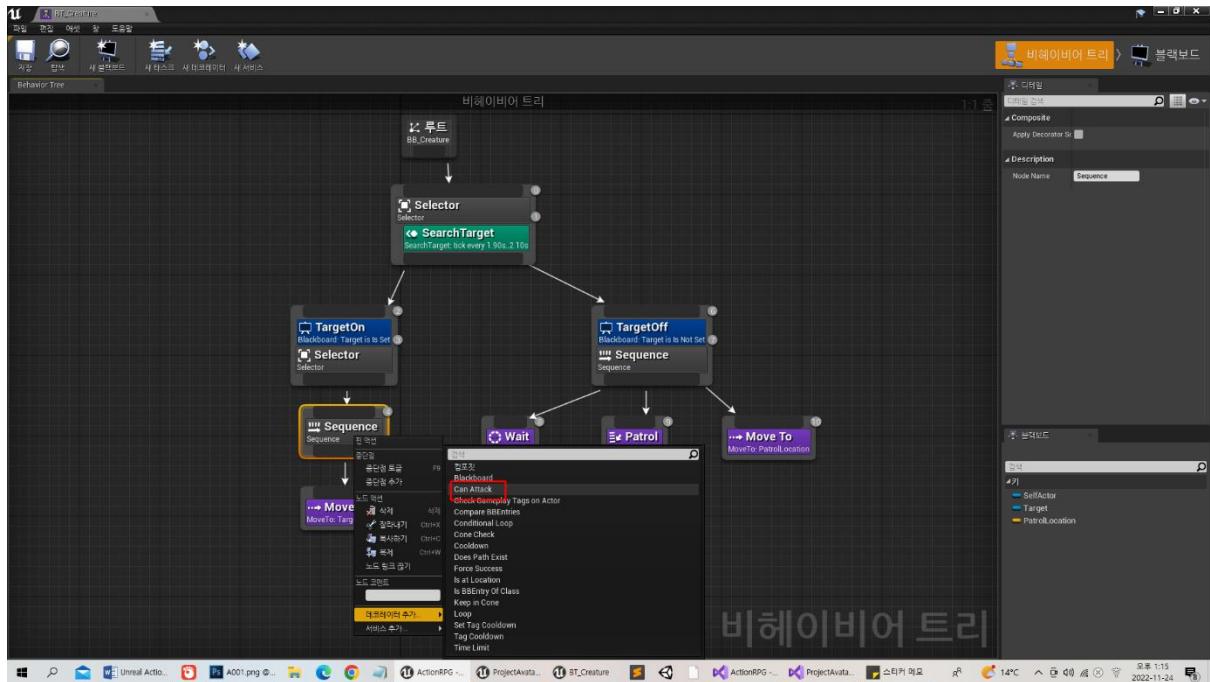
Unresolved external symbol "public __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ) referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ)

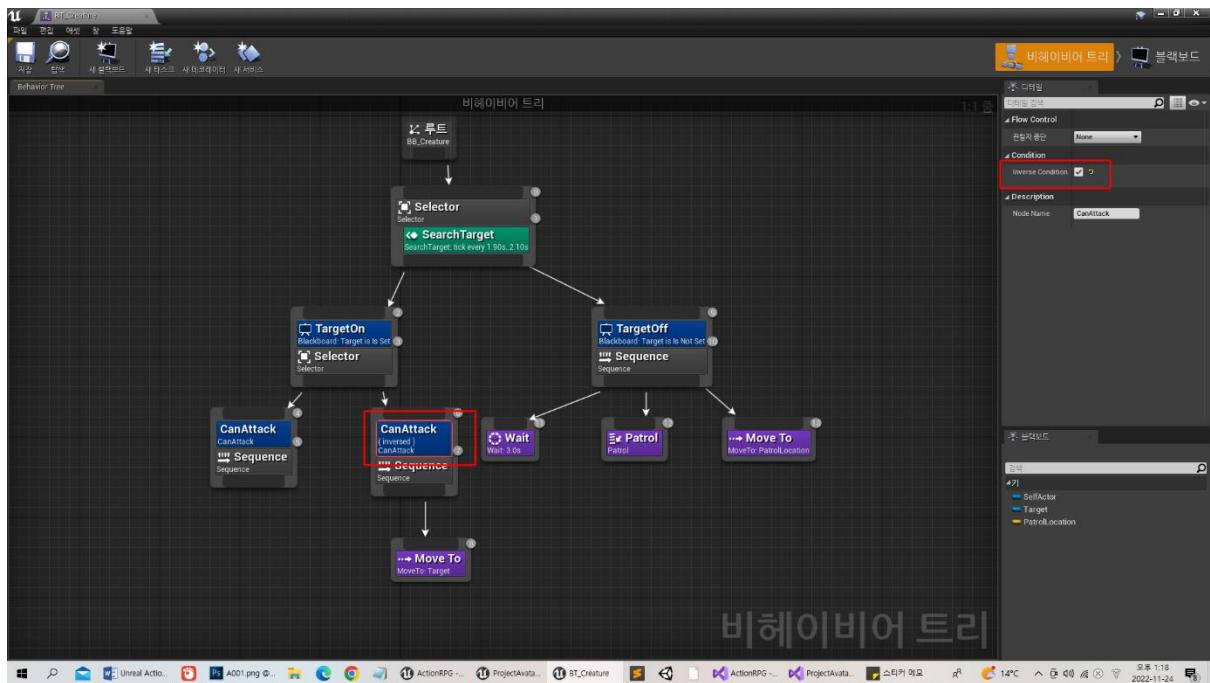
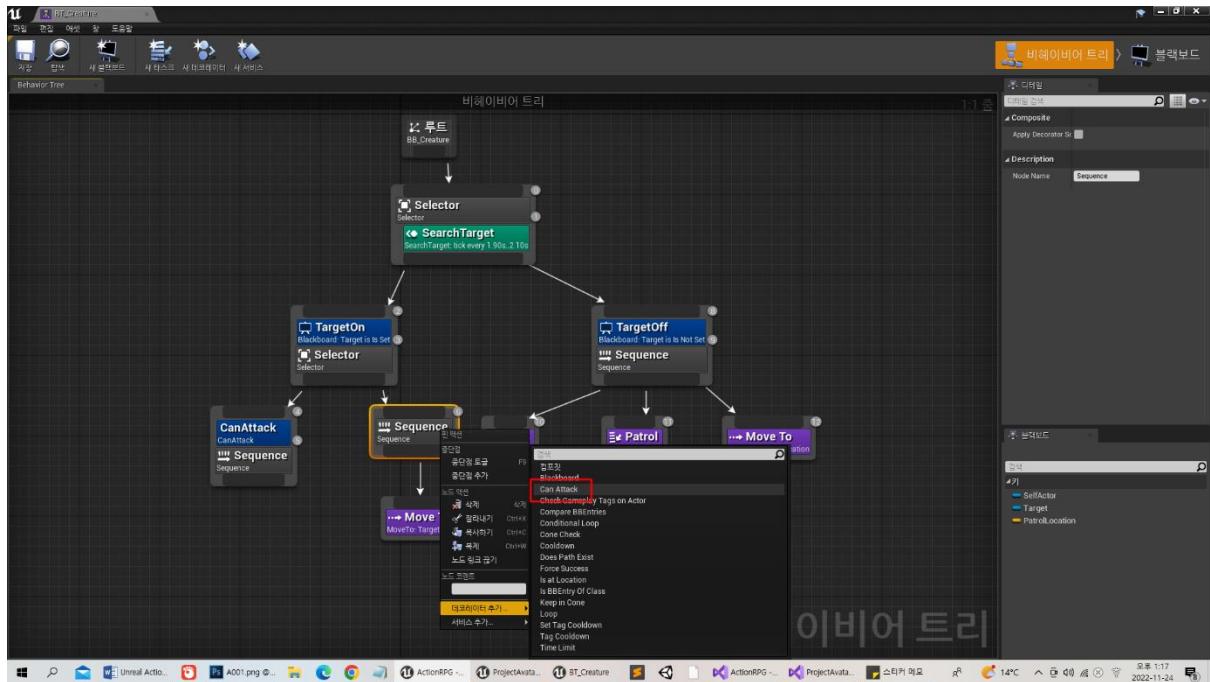
Unresolved external symbol "public: virtual bool __cdecl UBTDecorator_CanAttack::CalculateRawConditionValue(class UBehaviorTreeComponent &OwnerComp, unsigned char *const)" (??_CUBTDecorator_CanAttack@UBTDecorator_CanAttack@@UEA8UBehaviorTreeComponent@0@PAC@Z)

Unresolved external symbol "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ) referenced in function "public: static void __cdecl UBTDecorator_CanAttack::`##__dtor'() [BTDecorator_CanAttack]" (??0UBTDecorator_CanAttack@@@QEAA@XZ)

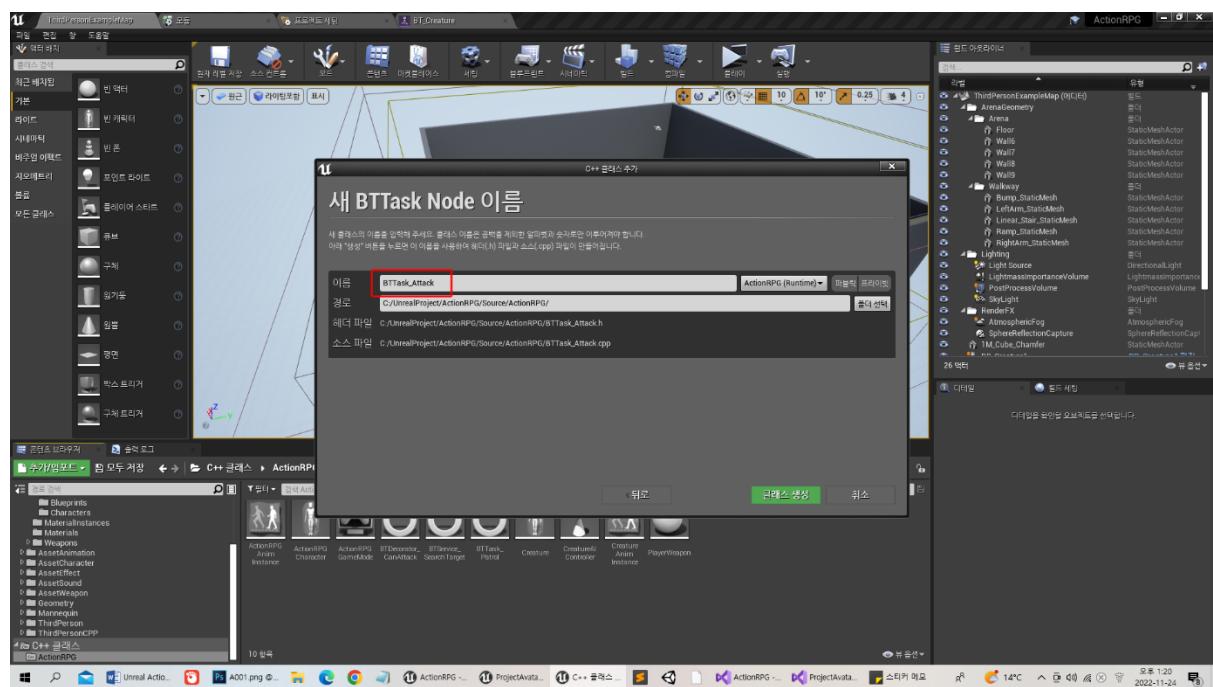
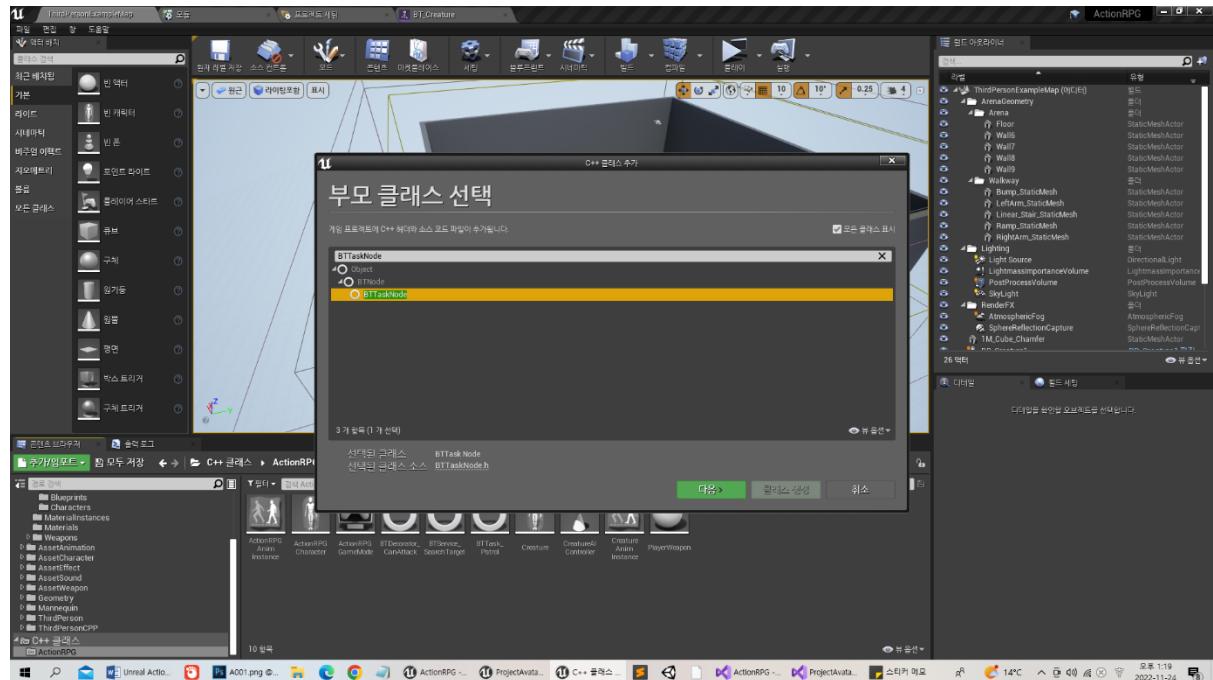
The command "C:\Program Files\Epic Games\UE_4.26\Engine\BuildBatchFiles\Build.bat" ActionRPGEditor Win64 Development -fProject="..\UnrealProject\ActionRPG\ActionRPG.uproject" -WaitMutex -fFromMsBuild" exited with code 1.

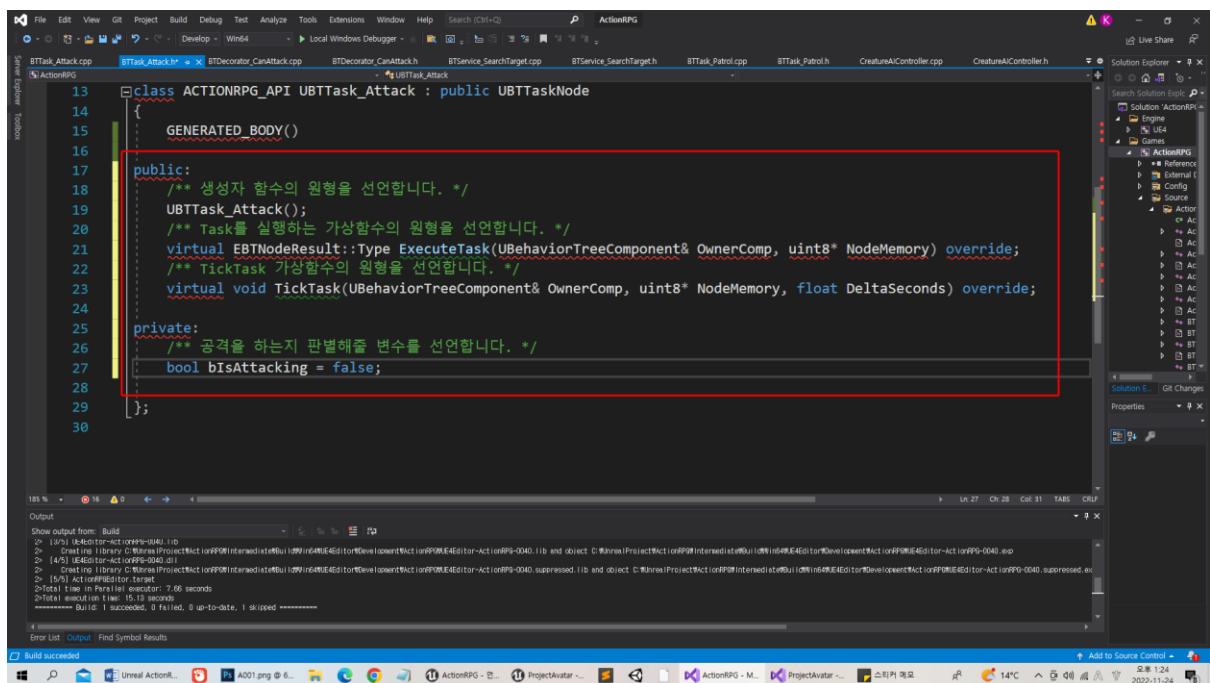
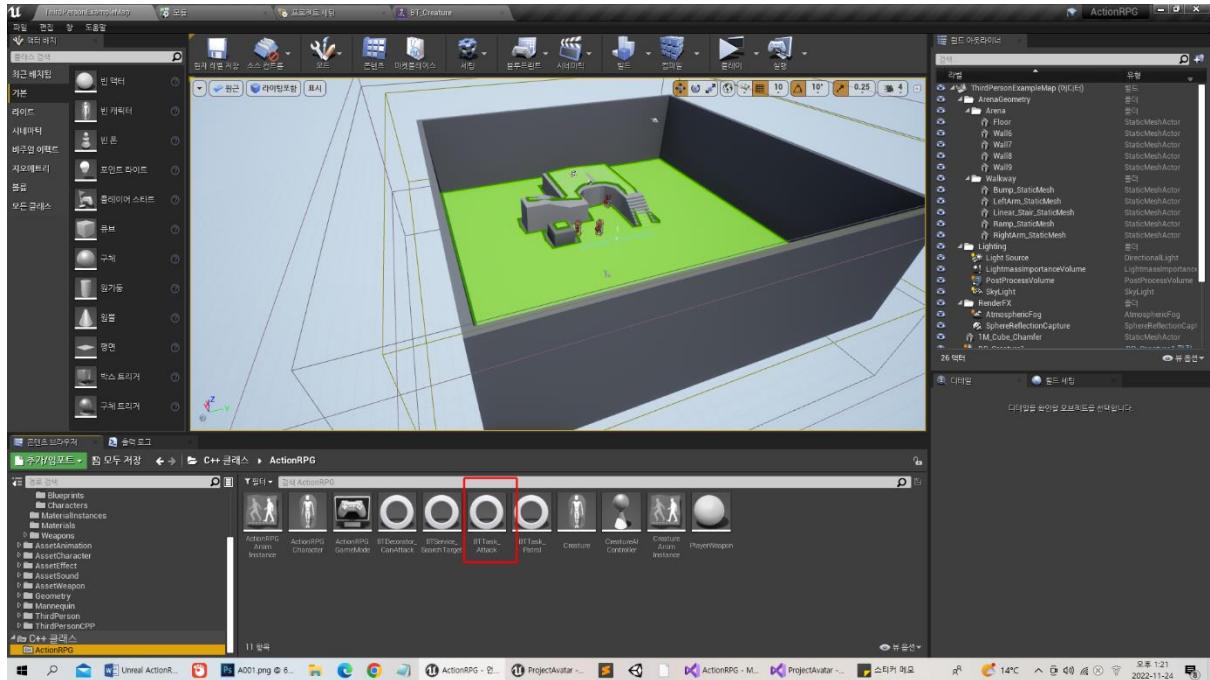
이제 조건검사를 해 주도록 합니다.

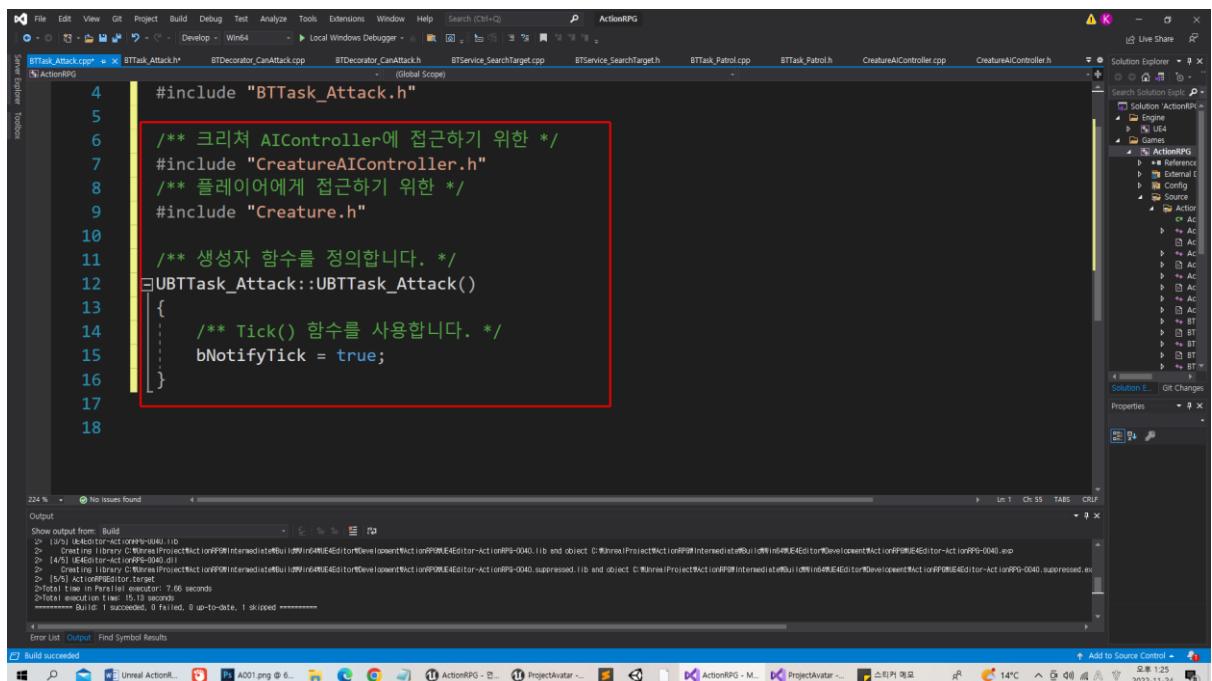




BTTaskNode를 부모클래스로 상속받는 BTTask_Attack이라는 클래스를 정의해 주도록 합니다.





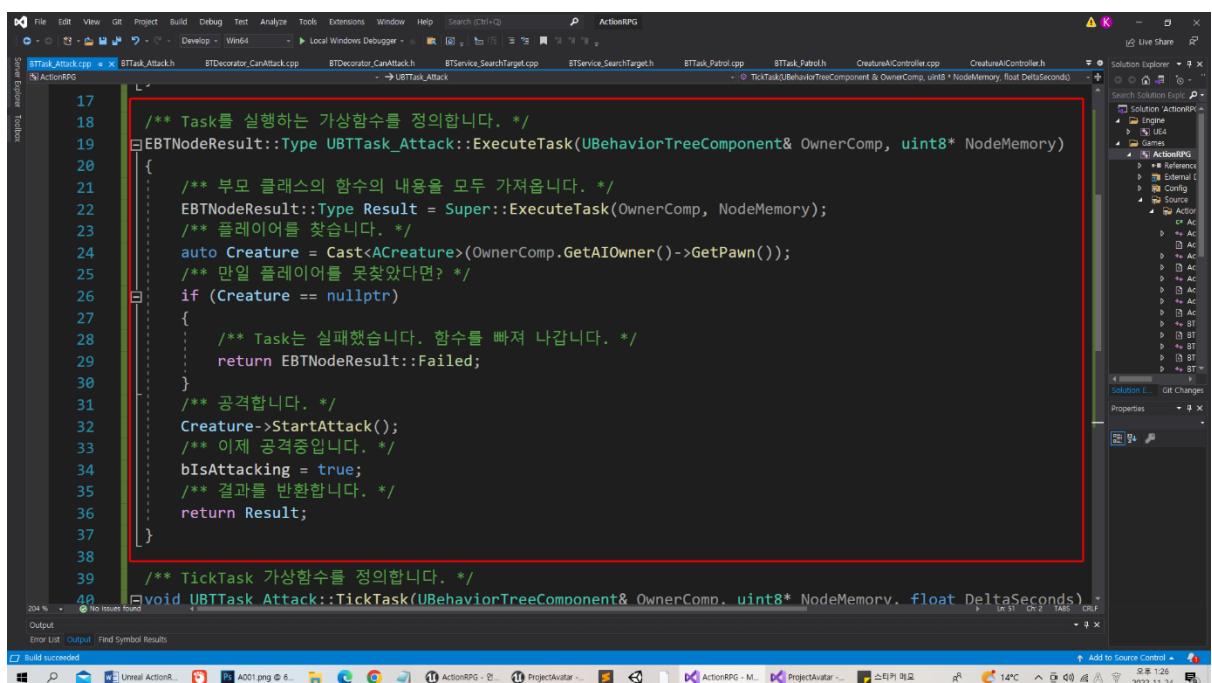


```
4 #include "BTTask_Attack.h"
5
6 /** 크리쳐 AIController에 접근하기 위한 */
7 #include "CreatureAIController.h"
8 /** 플레이어에게 접근하기 위한 */
9 #include "Creature.h"
10
11 /** 생성자 함수를 정의합니다. */
12 UBTTTask_Attack::UBTTTask_Attack()
13 {
14     /** Tick() 함수를 사용합니다. */
15     bNotifyTick = true;
16 }
```

Output window content:

```
Show output from: Build
> 1/14 UBTTask_Attack.cpp -> D:\UnrealEngine\4.24\Intermediate\Build\Win64\Editor\Actions\ActionRPG\Editor\Actions\ActionRPG-0040.lib and object C:\Users\Project\Actions\ActionRPG\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.exp
> 2/45 UEditor-ActionRPG-0040.dll
> 3/45 UBTTask_Attack -> D:\UnrealEngine\4.24\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.suppressed.lib and object C:\Users\Project\Actions\ActionRPG\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.suppressed.exp
> 4/51 ActionRPGEditor.Target
> Total 1 line in Parallel executor: 7.66 seconds
> Total 1 line in Serial executor: 15.19 seconds
> ===== Build 1 succeeded, 0 failed, 0 up-to-date, 1 skipped ======
```

Build succeeded

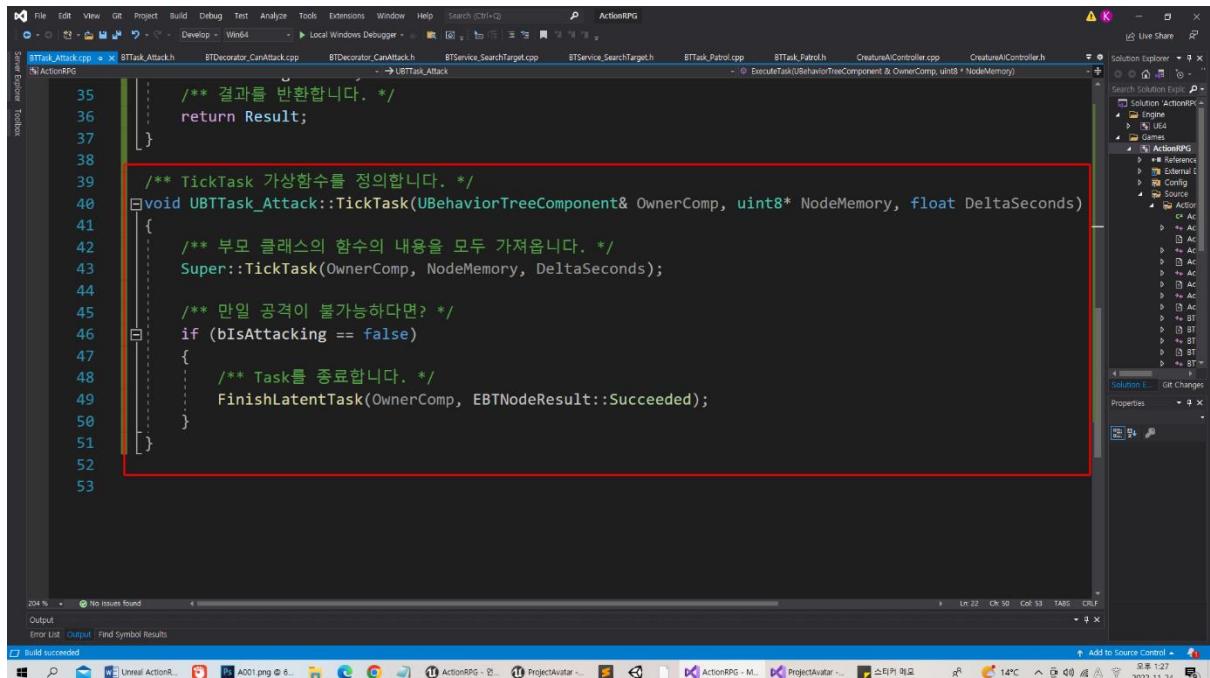


```
17 /**
18  * Task를 실행하는 가상함수를 정의합니다.
19  */
20 EBTNodeResult::Type UBTTTask_Attack::ExecuteTask(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory)
21 {
22     /** 부모 클래스의 내용을 모두 가져옵니다. */
23     EBTNodeResult::Type Result = Super::ExecuteTask(OwnerComp, NodeMemory);
24     /** 플레이어를 찾습니다. */
25     auto Creature = Cast<ACreature>(OwnerComp.GetAIOwner()->GetPawn());
26     /** 만일 플레이어를 못찾았다면? */
27     if (Creature == nullptr)
28     {
29         /** Task는 실패했습니다. 함수를 빠져 나갑니다. */
30         return EBTNodeResult::Failed;
31     }
32     /** 공격합니다. */
33     Creature->StartAttack();
34     /** 이제 공격중입니다. */
35     bIsAttacking = true;
36     /** 결과를 반환합니다. */
37     return Result;
38 }
39
40 /**
41  * void UBTTTask_Attack::TickTask(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory, float DeltaSeconds)
42 */
```

Output window content:

```
Show output from: Build
> 1/14 UBTTask_Attack.cpp -> D:\UnrealEngine\4.24\Intermediate\Build\Win64\Editor\Actions\ActionRPG\Editor\Actions\ActionRPG-0040.lib and object C:\Users\Project\Actions\ActionRPG\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.exp
> 2/45 UEditor-ActionRPG-0040.dll
> 3/45 UBTTask_Attack -> D:\UnrealEngine\4.24\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.suppressed.lib and object C:\Users\Project\Actions\ActionRPG\Intermediate\Build\Win64\Editor\Actions\ActionRPG-0040.suppressed.exp
> 4/51 ActionRPGEditor.Target
> Total 1 line in Parallel executor: 7.66 seconds
> Total 1 line in Serial executor: 15.19 seconds
> ===== Build 1 succeeded, 0 failed, 0 up-to-date, 1 skipped ======
```

Build succeeded

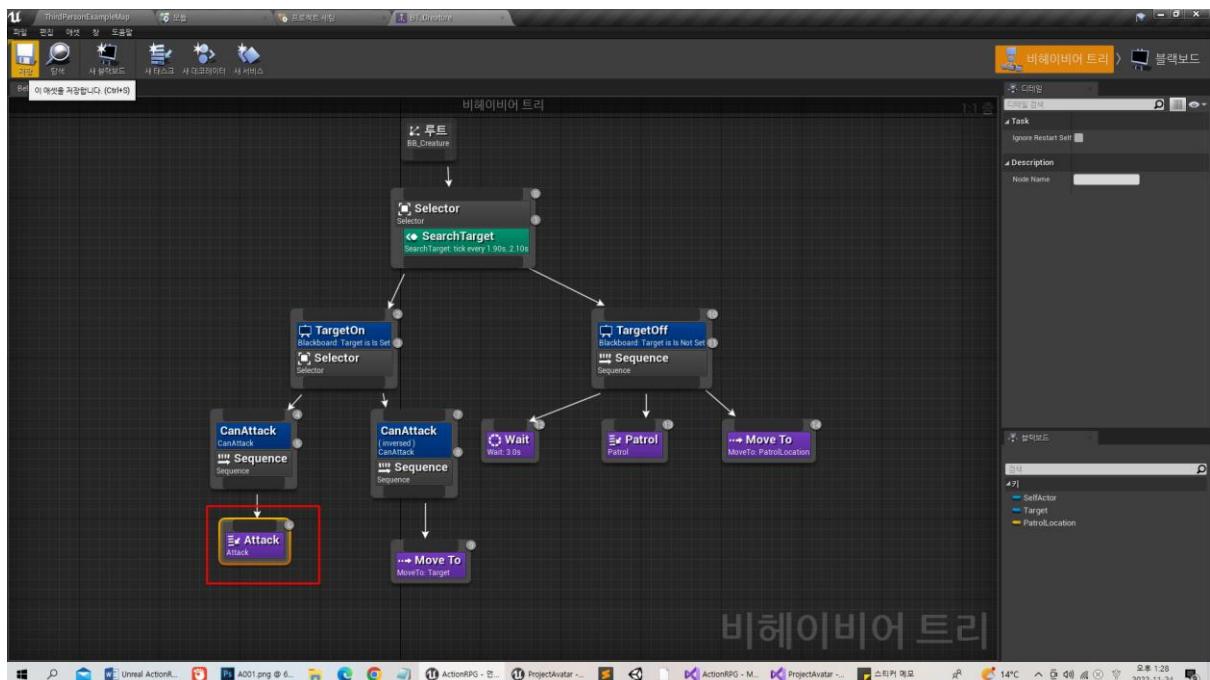


```

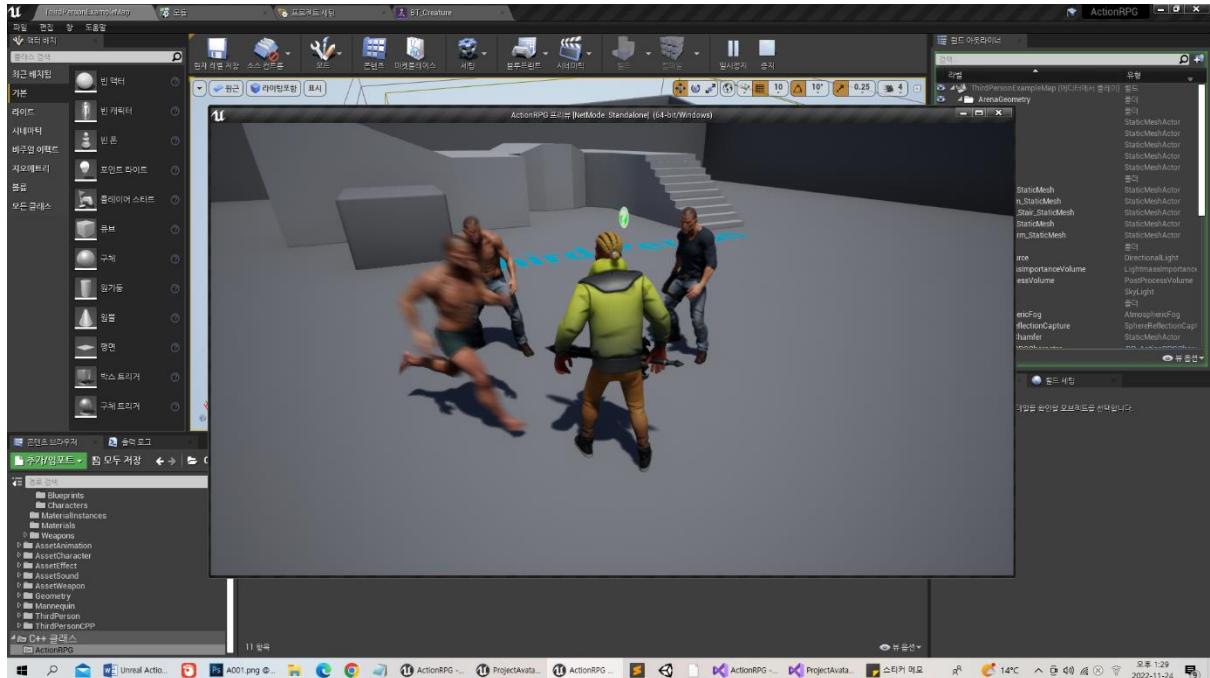
35 	/** 결과를 반환합니다. */
36 	return Result;
37 }
38
39 /**
40  * void UBTTask_Attack::TickTask(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory, float DeltaSeconds)
41 {
42 	/** 부모 클래스의 함수의 내용을 모두 가져옵니다. */
43 	Super::TickTask(OwnerComp, NodeMemory, DeltaSeconds);
44
45 	/** 만일 공격이 불가능하다면? */
46 	if (bIsAttacking == false)
47 	{
48 		/** Task를 종료합니다. */
49 		FinishLatentTask(OwnerComp, EBTNodeResult::Succeeded);
50 	}
51 }
52
53

```

BehaviorTree에서 추가해 줍니다.



플레이를 해서 결과를 확인합니다.



크리쳐가 플레이어를 공격할 때 플레이어를 향하지 않을 때가 있습니다.

생성자에서 값을 주도록 합니다.

```

File Edit View G Engine Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+F) ActionRPG
Creature.cpp Creature.h
ActionRPGAnimInstance.cpp ActionRPGAnimInstance.h ActionRPGCharacter.cpp ActionRPGCharacter.h BTService_SearchTarget.h BTService_SearchTarget.cpp CreatureAIController.cpp CreatureAIController.h
Creature.h
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

GetCharacterMovement() -> bOrientRotationToMovement = true;

/** AI 컨트롤러를 지정해 줍니다. */
AIControllerClass = ACreatureAIController::StaticClass();
/** AI가 이미 레벨에 있거나 스폰되었을 때도 적용되도록 지정해 줍니다. */
AutoPossessAI = EAutoPossessAI::PlacedInWorldOrSpawned;

/** 컨트롤러가 크리쳐폰을 회전할 수 있도록 해 줍니다. */
bUseControllerRotationYaw = true;

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

```