2단계_11. 행동트리 모델의 이해

수강일	@2023/06/21
이름	전영재
멘토	Min-Kang Song,현웅 최
작성 상태	Done
강의 시청 여부	☑

Contents

행동트리 모델의 이해

💡 **강의 목표**

- 행동트리 모델의 개요와 장점을 이해
- 행동트리 모델의 구성 요소와 이를 활용한 설계 방법 학습

---

[1] 행동 트리 모델 개요

~~와! 맛스타 치프!~~

2004년도, 헤일로2 AI의 설계에 사용되었다.

'우선순위'와 '트리'구조를 사용한 BehaviorTree는, 그당시 일반적이였던 FSM의 고질적인 문제를 해결하며 등장했다.

BehaviorTree의 장점

- 모듈화가 잘 되어 확장이 자유로움
- 트리를 기반으로 계층화가 잘 되어 있어, 복잡한 모델을 쉽게 설계할 수 있다.
- 직관적이기에, 현재 모델의 동작이 어떻게 작동하는지를 쉽게 파악할 수 있다.

행동트리 모델의 구성 요소

행동노드를 중심으로 설계되는 BehaviorTree는 하나의 부모 노드에서 다수의 행동을 컨트롤 한다.

행동 노드는 수행한 결과에 따라 **성공, 실패, 중지, 진행 중** 이라는 행동에 대한 결과를 컴포짓 노드에게 반환하며, 부모 노드는 이에 따라 전체 행동의 작동 여부를 결정하는데, 이를 **컴포짓(Composite)**이라고 한다.

셀렉터	여러 행동 중 하나의 행동을 지정 하나의 노드라도 성공을 리턴하면 하위 행동을 종료
시퀀스	여러 행동을 순서대로 모두 수행 모든 행동을 수행해야 하기에 하나라도 실패가 나온다면 하위 행동을 종료
패러렐	여러 행동을 동시에 병렬적으로 수행

또한 컴포짓 노드에는 여러가지 부가 기능을 부착할 수 있는데, 이를 통해서 복잡한 BT구조를 설계할 수 있으므로, 이들을 이해하는 것이 굉장히 중요하다.

데코레이터	컴포짓 노드가 실행되는 조건을 지정
서비스	컴포짓 노드가 활성화될 때, 주기적으로 실행하는 부가 명령
관찰자 중단(Abort)	데코레이터 조건에 부합되면, 컴포짓 내 활동을 모두 중단하고 Root로 회귀 (행동노드 작동 도중에 중단해야 되는 경우에 사용된다)

[2] NPC 전투를 위한 행동 트리 모델 예시

우리가 플레이어 캐릭터를 PlayerController를 통해 조종하는 것처럼, AI도 AIController를 통해 조종해줘야 한다.

```
AABCharacterNonPlayer.cpp
```

```
AABCharacterNonPlayer::AABCharacterNonPlayer()
{
  GetMesh()->SetHiddenInGame(true);

  AIControllerClass = AABAIController::StaticClass();
  AutoPossessAI = EAutoPossessAI::PlacedInWorldOrSpawned();
  /**
  AutoPossessAI : AI Controller가 해당 Pawn을 언제 Possess할지에 대한 설정이다.
  AI Controller가 할당되지 않는다면 아예 NPC가 조종되지 않을테니 Placed In World Or Spawned로 해놓는게 무난할듯하다.
  */
}
```

또한 AI Controller는 AI의 두뇌와 행동로직이 되어줄 BlackBoard와 BehaviorTree를 소지하고 이를 작동시켜야 한다.

```
SAAIController.cpp //AB예제가 아닌 언씬 프로젝트에 적용시킨 코드를 가져왔다.

#include "AI/SAAIController.h"
#include "BehaviorTree/BlackboardData.h"
#include "BehaviorTree/BehaviorTree.h"
#include "BehaviorTree/BlackboardComponent.h"

ASAAIController::ASAAIController()
{
  static ConstructorHelpers::FObjectFinder<UBlackboardData> BBAssetRef(TEXT(" 경로명 ")); //만든 BB경로를 넣어줘야한다. 싫으면, Ref말고 직접 BP에
  if (nullptr != BBAssetRef.Object)
  {
    BBAsset = BBAssetRef.Object;
  }

  static ConstructorHelpers::FObjectFinder<UBehaviorTree> BTAssetRef(TEXT(" 경로명 "));
  if (nullptr != BTAssetRef.Object)
  {
    BTAsset = BTAssetRef.Object;
  }
}

void ASAAIController::RunAI()
{
  UBlackboardComponent* BlackboardPtr = Blackboard.Get(); //AIController에서 기본적으로 보유하는 Blackboard 속성의 ptr를 가져와 할당해주는 방식인듯
  if (UseBlackboard(BBAsset, BlackboardPtr))
  {
    bool RunResult = RunBehaviorTree(BTAsset);
    ensure(RunResult);
  }
}

void ASAAIController::StopAI()
{
  UBehaviorTreeComponent* BTComponent = Cast<UBehaviorTreeComponent>(BrainComponent); //왜 BehaviorTree라 안하고 BrainComponent라는 이름을
  if (BTComponent)
  {
    BTComponent->StopTree();
  }
}

void ASAAIController::OnPossess(APawn* InPawn)
{
  Super::OnPossess(InPawn);

  RunAI();
}
```

AI Controller에서 이미 소유하고 있는 `Blackboard`와 `BrainComponent`라는 프로퍼티에 우리의 애셋을 연결시켜주는 방식을 사용하는 것에 유의하자.

📌 **Summary**

- 행동 트리 모델의 구조인 컴포짓, 행동노드, 데코레이터 등을 이해한다.
- 후에 사용될 Blackboard와 BehaviorTree를 사용하기 위해 Ai Controller에서 이를 소유하고 작동시키는 방법을 이해한다.

0n. 행동트리 모델의 이해

❓ Q1. 가상의 NPC 행동 시나리오를 만들고, 이를 구현하기 위한 행동트리 모델을 설계하시오.

몬스터는 스폰과 동시에 플레이어에 대한 참조를 얻게되며, MoveTo 태스크를 통해 플레이어를 추적한다.

플레이어에게 도달하게 되면 공격 태스크를 수행하게 되고 전방으로 짧은 거리의 라인트레이스를 발사하여, 범위 내에 목표물이 존재한다면 공격성공으로 판정 되고 데미지를 입힌다.

단순하게 위 동작을 반복하다 체력이 0이 되어 사망상태가 된다면 데코레이터를 통해 이를 파악하여 그 자리에서 사망 애니메이션을 출력하고 관련 정보를 삭제한다.