Unreal Engine 4 Portfolio

FPS CO-OP GAME

곽현진

목차

Project 설명

- 3인칭 슈팅 게임으로써, 총 4명의 플레이어가 플레이를 할 수 있습니다.
- 각 플레이어들은 라운드마다 2의 배수만큼 생성되는 TrackerBot or Advanced AI를 처치해야 합니다.
- 생성된 적들을 제거할 때마다 20점의 점수를 얻습니다.
- 서로 힘을 합쳐 최대한 버티면 됩니다.

Player : Fire 함수

```cpp
void ASWeapon::Fire()
{
    // 무기의 소유자가 누구인지, 누가 무기를 통제하는지
    AActor* MyOwner = GetOwner();
    if (MyOwner)
    {
        FVector EyeLocation;
        FRotator EyeRotation;
        MyOwner->GetActorEyesViewPoint(EyeLocation, EyeRotation);

        FCollisionQueryParams QueryParams;
        QueryParams.AddIgnoredActor(MyOwner);
        QueryParams.AddIgnoredActor(this);
        QueryParams.bTraceComplex = true;

        FVector TracerEndPoint = TraceEnd;

        // 적중 결과 저장 변수(어떤 사물이 맞았는지, 얼마나 멀리 있는지, 방향을 어떻게 잡았는지)
        FHitResult Hit;
        if (GetWorld()->LineTraceSingleByChannel(Hit, EyeLocation, TraceEnd, COLLISION_WEAPON, QueryParams))
        {
            // Trace가 막히면 데미지를 입히십시오
        }
        DrawDebugLine(GetWorld(), EyeLocation, TraceEnd, FColor::White, false, 1.0f, 0, 1.0f);
    }
}
```

- 충돌 쿼리 매개변수
충돌 대상이 소유자(플레이어)이거나 총이면 충돌을 무시
bTraceComplex : 맞은 부위가 어떤 부위인지 알기 위해 True로 설정 시 각 부위에 대해 Target 배치 가능

Player : 피해 입히기

```
UGameplayStatics::ApplyPointDamage(HitActor, 20.0f, ShotDirection, Hit, MyOwner->GetInstigatorController(), this, DamageType);
```

- Parameter 1 : Trace가 막혔을 때, 막히게 된 원인(Actor) `AActor* HitActor = Hit.GetActor();`
- Parameter 2 : 피해량
- Parameter 3 : 발사한 방향 `FVector ShotDirection = EyeRotation.Vector();`
- Parameter 4 : FHitRusult 타입, Hit 했을때의 정보가 들어있음
- Parameter 5 : 누가 데미지를 입혔는가
- Parameter 6 : DamageCauser, 무엇이 데미지를 입혔는가
- Parameter 7 : 데미지의 유형(TSubclassOf Type)

AI : TrackerBot

```
FVector ASTrackerBot::GetNextPathPoint()
{
    // player의 위치를 얻기
    ACharacter* PlayerPawn = UGameplayStatiacs::GetPlayerCharacter(this, 0);

    UNavigationPath* NavPath = UNavigationSystemV1::
        FindPathToActorSynchronously(this, GetActorLocation(), PlayerPawn);

    GetWorldTimerManager().ClearTimer(TimerHandle_RefreshPath);
    GetWorldTimerManager().SetTimer(TimerHandle_RefreshPath, this, &ASTrackerBot::RefreshPath, 5.0f, false);

    if (NavPath && NavPath->PathPoints.Num() > 1)
    {
        // 다음 경로 포인트 가져오기
        return NavPath->PathPoints[1];
    }
    // 경로 찾기에 실패
    return GetActorLocation();
}
```

(WorldContextObject, PathStart, GoalActor)

Navigation을 구현하려는 Level(?)

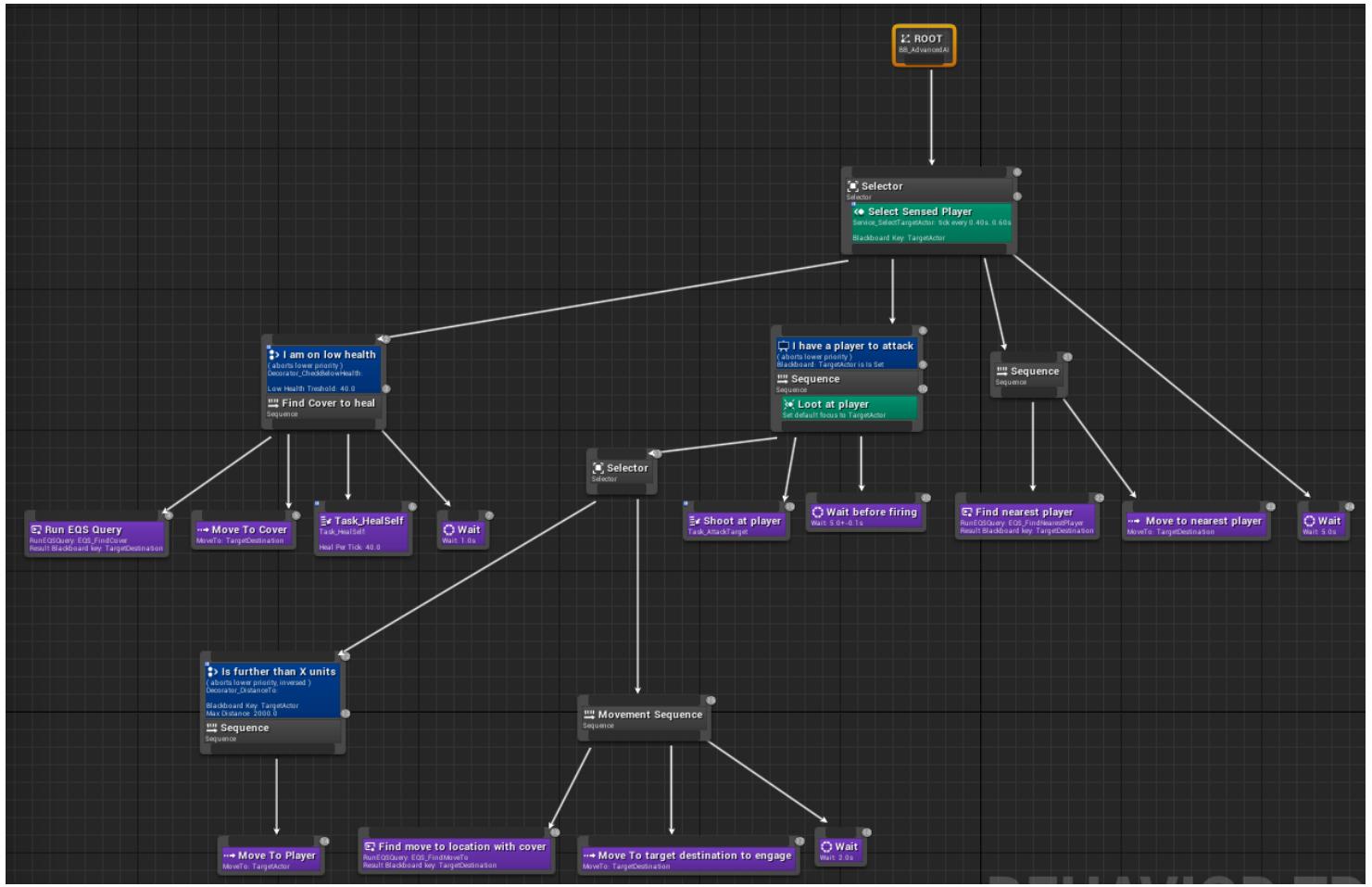
```
void ASTrackerBot::RefreshPath()
{
    NextPathPoint = GetNextPathPoint();
}
```

AI : TrackerBot

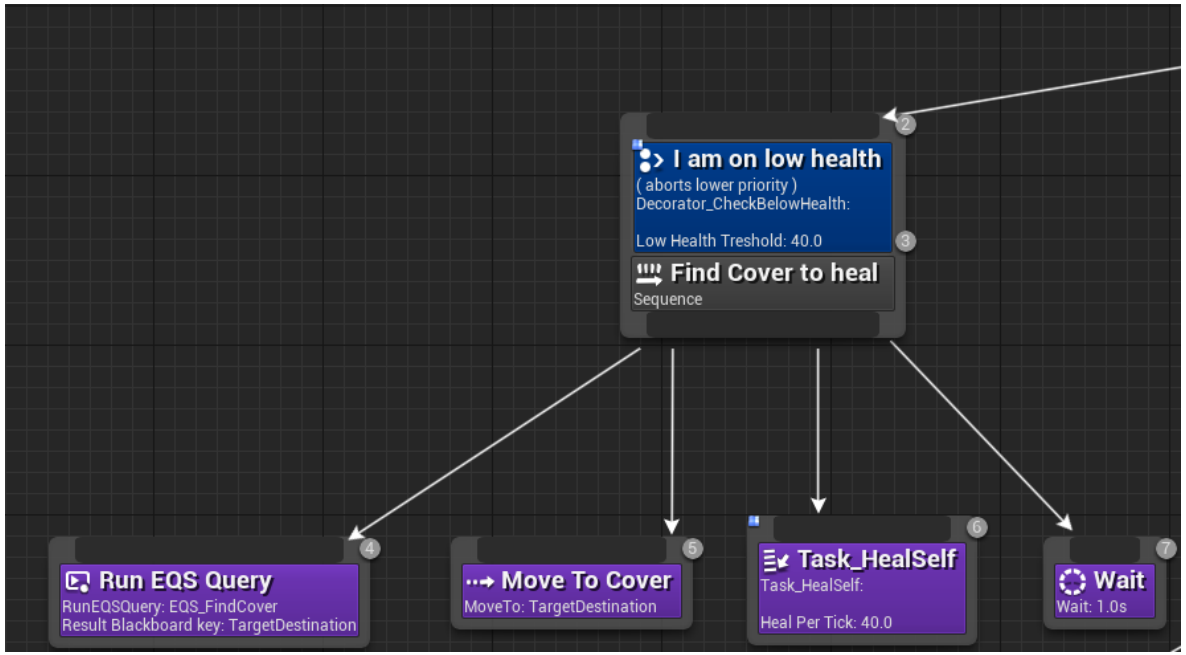
```cpp
void ASTrackerBot::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

    float DistanceToTarget = (GetActorLocation() - NextPathPoint).size();

    if (DistanceToTarget <= RequiredDistanceToTarget)
    {                                    Default  Value : 100
        NextPathPoint = GetNextPathPoint();
    }
    else
    {
        // 탐색 경로의 다음 지점
        FVector ForceDirection = NextPathPoint - GetActorLocation();
        ForceDirection.Normalize();

        ForceDirection *= MovementForce;

        MeshComp->AddForce(ForceDirection, NAME_None, bUseVelocityChange);
    }
}
```

- 대상(player)와의 거리가 100보다 작거나 같은 경우 GetNextPathPoint() 함수로 구해진 다음 경로로 이동
- 100보다 큰 경우 다음 point를 찾지 않고 플레이어가 있는 일직선 방향으로 움직임

AI : Advanced AI

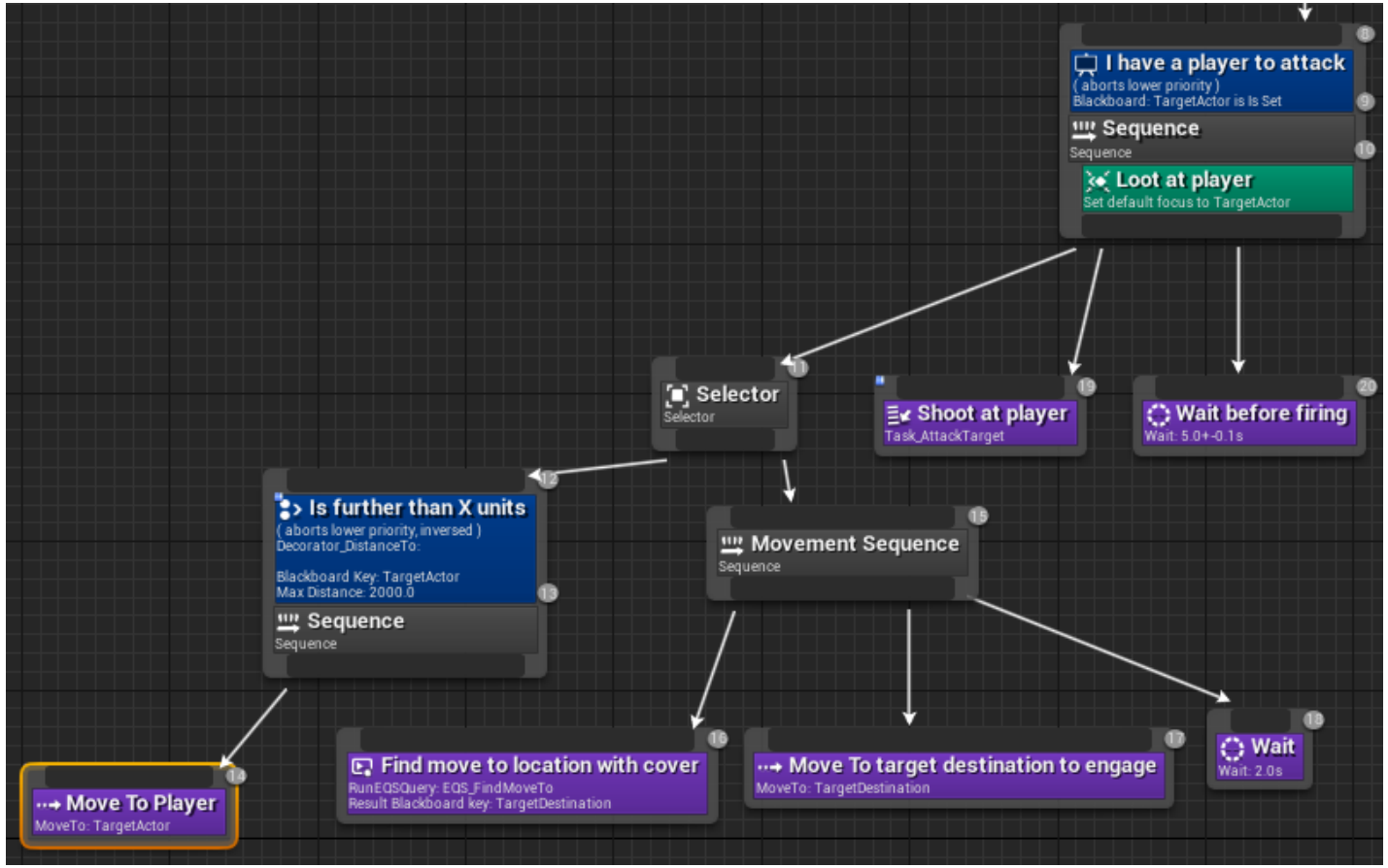


AI : Advanced AI



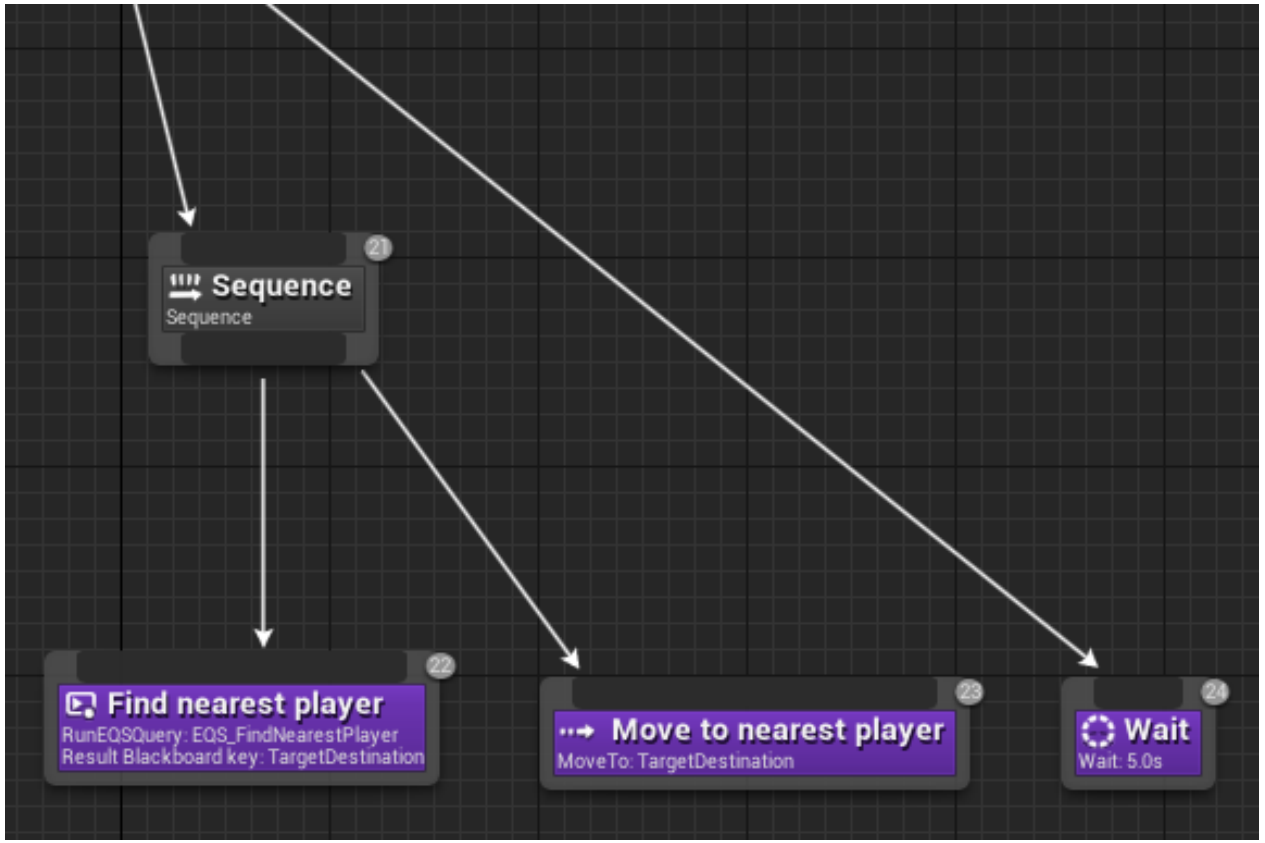
- 첫번째로 실행되는 Behavior Tree Node
- 우선적으로 Advanced AI가 체력이 없는 경우 숨을 곳을 찾아 움직여 체력을 회복함

AI : Advanced AI



- 두번째로 실행되는 Behavior Tree Node
- 플레이어를 공격하기 위해 움직이지만 플레이어와의 거리가 2000이상 떨어져 있으면 플레이어와 가까워지도록 함
- 가까워지면 엄폐물을 찾아 움직이고 Wait 상태가 됨
- 2초 후 플레이어를 공격
- 공격 후 5초 대기

AI : Advanced AI



- 세번째로 실행되는 Behavior Tree Node
- 플레이어를 공격 후 근처의 플레이어를 찾아서 가장 가까운 플에이어에게 이동

추가한 기능 : Crouch()

- Crouch의 버튼을 Pressed <-> Released가 아닌 Pressed일 때마다 상태가 바뀌길 원함
- Crouch 함수가 있는 Characther.h에서 찾아본 결과 CanCrouch()를 찾음
- CanCrouch()는 이 캐릭터가 현재 Crouch 할수 있는 경우 true를 반환

```
void APCharacter::BeginCrouch()
{
        Crouch();
}

void APCharacter::EndCrouch()
{
        UnCrouch();
}
```

```
void APCharacter::CrouchToggle()
{
    if (CanCrouch())
    {
        Crouch();
    }
    else
    {
        UnCrouch();
    }
}
```

```
PlayerInputComponent->BindAction("Crouch", IE_Pressed, this, &APCharacter::BeginCrouch);
PlayerInputComponent->BindAction("Crouch", IE_Pressed, this, &APCharacter::EndCrouch);
```

```
PlayerInputComponent->BindAction("Crouch", IE_Pressed, this, &APCharacter::CrouchToggle);
```

추가한 기능 : Sprint

- float Type의 SprintingSpeed를 생성하고 생성자에 2.0f의 값을 초기화(Sprint시 2배의 속도를 내기 위해)
- StartSprinting() 함수와 StopSprinting() 함수를 생성
- 애니메이션 블루 프린트를 이용하여 애니메이션 추가

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Movement")
float SprintingSpeed;

void StartSprinting();

void StopSprinting();
```

```
void ASCharacter::StartSprinting()
{
    GetCharacterMovement()->MaxWalkSpeed *= SprintingSpeed;
}

void ASCharacter::StopSprinting()
{
    GetCharacterMovement()->MaxWalkSpeed /= SprintingSpeed;
}
```

참고자료

- 전체적인 강의 : https://www.udemy.com/unrealengine-cpp/learn/v4/content
- UPROPERTY 설명 : https://www.youtube.com/watch?v=ZPJtFa9srXw
- UFUNCTION 설명 : https://www.youtube.com/watch?v=yNco6wM5ElI
- UE4에서의 데미지 : https://www.unrealengine.com/ko/blog/damage-in-ue4?sessionInvalidated=true