BuildSystem

프로젝트 기간	@2023년 5월 17일 → 2023년 5월 24일
태그	개발
생성자	
완료	Done
멘토 그룹	Min-Kang Song,현웅 최
상태	Not started
날짜 1	
날짜	
프로젝트명	
완료 여부	
유형	
다중 선택	
교육생 전체 학습 진도율	전영재

<구현해야할것>

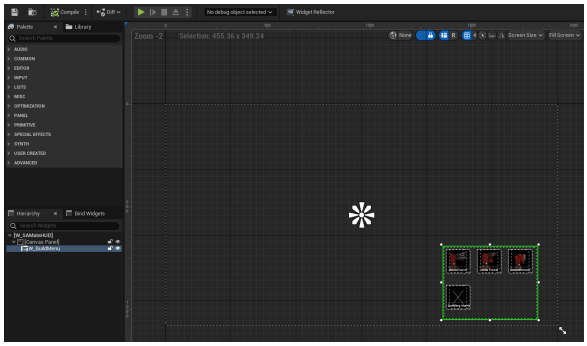
1. 빌드UI 버튼을 누를시, 건물 생성 및 마우스 커서에 위치
2. 건물 생성단계에서 타Actor와 오버랩시 초록색 혹은 빨간색 Material로 변경
3. 건설 상태에서 마우스 클릭 시 Build완성, 해당 커서 위치에 Location고정과 원래Material로 변경

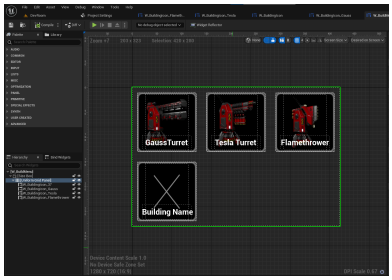
빌드UI와 Actor생성

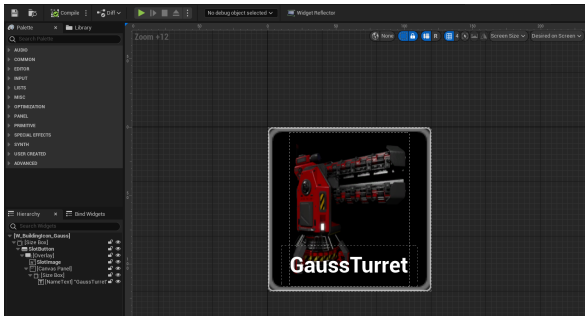
UI는 3중 구조로 만들었다. (MainHUD→BuildMenu→Build Icon)

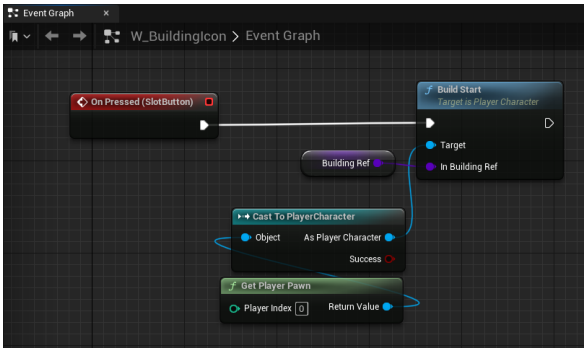
PlayerCharacter의 `BeginPlay()`에 MainHUD가 부착되고, 플레이어의 버튼입력에 의해 BuildMenu UI가 오픈된다. 그 후, 메뉴 내의 Icon을 누르면 PlayerCharacter의 `BuildStart()`를 호출해 캐릭터에서 해당 건물의 생성을 다룬다.

차라리 BuildComponent를 만들어서 UI와 구독/발행 관계로 만들어보면 어떨까? 빠르게 강의를 듣고 실행해보자.









위 Icon을 통해 UI는 캐릭터에게 특정 건물에 대한 UClass정보를 보내고, 캐릭터는 해당 정보를 수신함과 동시에 `SpawnActor()`를 호출하고, `InputMappingContext`를 BuildType으로 변환한다.

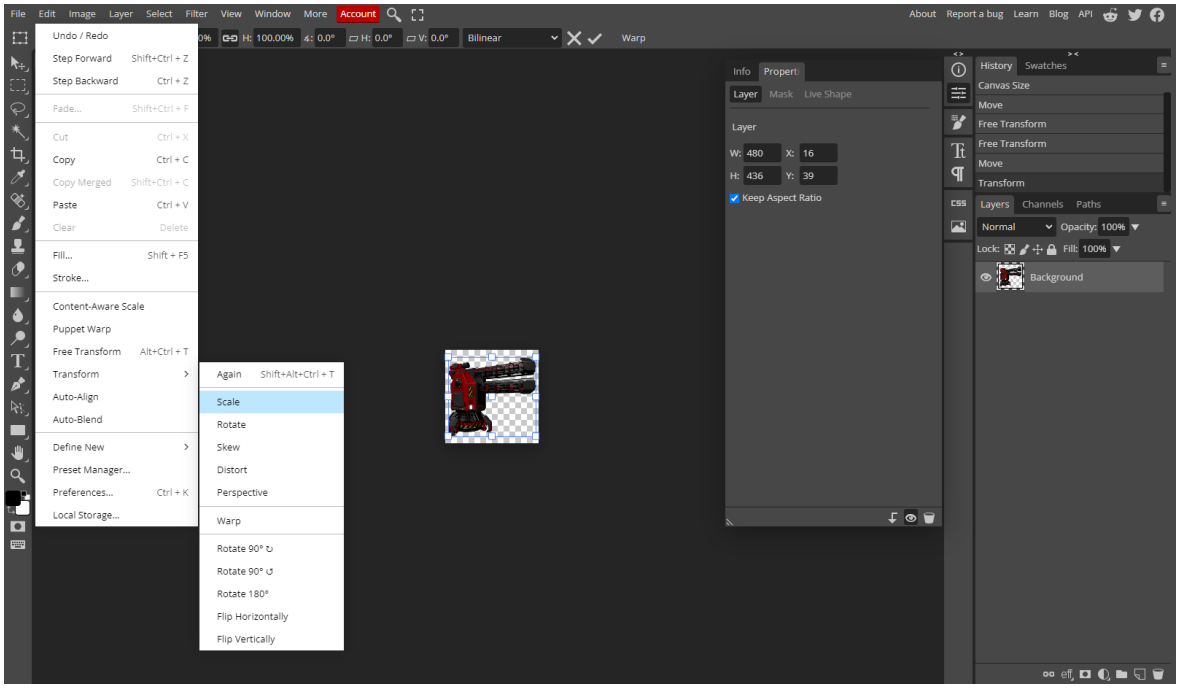
```
void APlayerCharacter::BuildStart(UClass* InBuildingRef)
{
  if (!InBuildingRef)
  {
    return;
  }

  if (!IsValid(HoldingActor))
  {
    HoldingActor = GetWorld()->SpawnActor<AActor>(InBuildingRef, FTransform());
  }

  if (UEnhancedInputLocalPlayerSubsystem* Subsystem = ULocalPlayer::GetSubsystem<UEnhancedInputLocalPlayerSubsystem>(PlayerController-
  {
    Subsystem->ClearAllMappings();
    Subsystem->AddMappingContext(IMC_PlayerBuildMenu, 0);
  }
}

void APlayerCharacter::BuildComplete()
{
  if (!IsValid(HoldingActor))
  {
    return;
  }
  UE_LOG(LogTemp, Warning, TEXT("BuildComplete"));


  ABaseTurret* HoldingTurret = Cast<ABaseTurret>(HoldingActor);
  HoldingTurret->BuildCompleted();

  HoldingTurret = nullptr;
  HoldingActor = nullptr;

  if (UEnhancedInputLocalPlayerSubsystem* Subsystem = ULocalPlayer::GetSubsystem<UEnhancedInputLocalPlayerSubsystem>(PlayerController-
  {
    Subsystem->ClearAllMappings();
    Subsystem->AddMappingContext(IMC_PlayerCombat, 0);
  }
}
```

생성된 Actor의 위치는 해당 Actor내부로직에서 작동되기에, 캐릭터에서는 그저 Spawn만 담당하기로 구조지었다.

▼ UI Icon생성

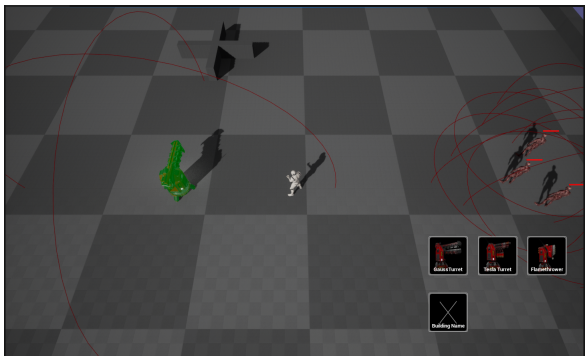
카메라 Perspective는 Orthographic으로 액터의 RenderCustomDepthPass는 true로 바꿔준다. 그리하여 거리에 상관없이, 겹침에 상관없이 이미지를 찍을수있다.

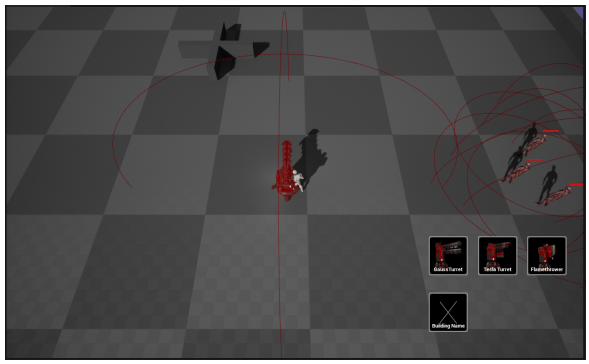
찍은 스크린샷은 포토샵을 통해 적당한 크기로 가공.(Photopea를 통해 포토샵없이 진행할 수 있다.)



오버랩 시 Material 변경

마치 게임 '스타크래프트' 와 같이, 건물 생성단계에서 특정 요인으로 인해 해당 위치에 건설이 불가능하다면, 건물을 붉게 만들어 불가능함을, 가능하다면 녹색으로 만들어 시각적으로 알기 쉽게 구현.





```
void ABaseBuildingActor::SetAllOverlayMaterials(UMaterialInterface* InMaterial)
{
  TArray<UActorComponent*> ActorComponentArray = GetComponentsByClass(UStaticMeshComponent::StaticClass());
  for (UActorComponent* EachComponent : ActorComponentArray)
  {
    UStaticMeshComponent* EachStaticMeshComponent = Cast<UStaticMeshComponent>(EachComponent);
    EachStaticMeshComponent->SetOverlayMaterial(InMaterial);
  }
}

void ABaseBuildingActor::BuildCollisionBeginOverlap(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimitiveComponent*
{
  SetAllOverlayMaterials(RedMaterial);
}
void ABaseBuildingActor::BuildCollisionEndOverlap(UPrimitiveComponent* OverlappedComp, AActor* OtherActor, UPrimitiveComponent* OtherC
{
  SetAllOverlayMaterials(GreenMaterial);
}
void ABaseBuildingActor::BuildCompleted()
```

```
{
  BuildCollision->SetGenerateOverlapEvents(false);
  SetAllOverlayMaterials(nullptr);
}
```

▼ 폐기된 코드

Actor의 모든 StaticMeshComponent에 대해 Material을 하나씩 바꿔주는 형식으로 구현되었다.

그러나 후에 `MaterialOverlay()` 의 존재를 알게되면서 폐기하도록 결정했다.

```
void ABaseBuildingActor::ChangeMeshMaterialToGreen(UStaticMeshComponent* InMesh)
{
  checkf(GreenMaterial, TEXT("[%s] RedMaterial is not Set."), *GetName());

  UStaticMeshComponent* TargetMesh = InMesh;
  TArray<UMaterialInterface*>MaterialArray = TargetMesh->GetMaterials();

  for (int cnt = 0; cnt < MaterialArray.Num(); ++cnt)
  {
    TargetMesh->SetMaterial(cnt,GreenMaterial);
  }
}

void ABaseBuildingActor::ChangeMeshMaterialToRed(UStaticMeshComponent* InMesh)
{
  checkf(RedMaterial, TEXT("[%s] RedMaterial is not Set."), *GetName());

  UStaticMeshComponent* TargetMesh = InMesh;
  TArray<UMaterialInterface*>MaterialArray = TargetMesh->GetMaterials();

  for (int cnt = 0; cnt < MaterialArray.Num(); ++cnt)
  {
    TargetMesh->SetMaterial(cnt, RedMaterial);
  }

  void ABaseBuildingActor::BuildCompleted()
  {
    BuildCollision->SetGenerateOverlapEvents(false);
//ETurretState::Searching변경은 BaseBuilding이 아닌 BaseTurret에서 override해 적용
  }
}
```

▼ 문제였던 부분

건설이 완료되기 전까지는 해당 Actor는 모든 물체를 뚫고 지나갈 수 있어야 한다.

매번 새로운 Class를 제작할때마다 CollisionProfile을 일일히 수정해 줄 수는 없기에 이것을 스마트하게 다룰 방법이 필요했고, 최상위 부모에서 후에 자식Class에서 추가될 컴포넌트에 대해서도 다룰 방법이 없나 생각하다 다음과 같은 방법을 시도했다.

```
void ABaseBuildingActor::BeginPlay()
{
  Super::BeginPlay();

  BuildCollision->OnComponentBeginOverlap.AddDynamic(this, &ABaseBuildingActor::BuildCollisionBeginOverlap); //constructor에 넣으면 
  BuildCollision->OnComponentEndOverlap.AddDynamic(this, &ABaseBuildingActor::BuildCollisionEndOverlap);

  /** Collision이 존재하는 모든 컴포넌트 추출 후 CollisionProfile 변경 */
  TArray<UActorComponent*> CollisionArray = GetComponentsByClass(UPrimitiveComponent::StaticClass());
  for (UActorComponent* EachComponent : CollisionArray)
  {
    UPrimitiveComponent* EachPrimitiveComponent = Cast<UPrimitiveComponent>(EachComponent);
    EachPrimitiveComponent->SetCollisionProfileName(TEXT("BuildingPreset"));
  }
  BuildCollision->SetCollisionProfileName(TEXT("OverlapAllDynamic")); //나중에 채널파줘야될듯. 일단 임시방편

  /** build시작 시, 초록색 material로 변경*/
  SetAllOverlayMaterials(GreenMaterial);
  // Build 가능한 위치인지를 표시하는 property 하나 만들어줘야겠다.
}
```

클래스 생성자 부분이 아닌 BeginPlay에서 자신이 소유하는 모든 Component를 검색하는 방법이다. 이를 통해 기능은 같지만 검색해야 될 대상이 매 class마다 다른 경우를 어떻게 다룰지에 대해 해결할 수 있었다.

건설 완료 시

해당 포탑이 적을 인식하고 공격하는 동작을 취하기 위해서는 Tick을 사용해야 한다.

아예 Tick의 사용을 배제할 수 없다면 Tick을 최대한 효과적으로 사용하려고 생각했고, 건물의 생성시 움직임을 switch문에 구현했다.

BuildCompleted()를 통해 switch문 중 이동상태에서 Search상태로 변하기에, 자연스럽게 건물의 정착을 표현할 수 있다.

```
void ABaseTurret::BuildCompleted()
{
  Super::BuildCompleted();

  TurretState = ETurretState::ETS_Searching;
  FireField->SetCollisionProfileName(TEXT("OverlapAllDynamic"));
}

void ABaseTurret::TurretBehaviorStateMachine(float DeltaTime)
{
  switch (TurretState)
  {
    case ETurretState::ETS_OnBuild:
    {
      FHitResult CurorHitResult;
      UGameplayStatics::GetPlayerController(this, 0)->GetHitResultUnderCursor(ECollisionChannel::ECC_Camera, true, CurorHitResult);

      float InGridSize = 50.f; //후에 타일사이즈를 GameState에서 설정하여 관리할 것
      FVector BuildPosition = UKismetMathLibrary::Vector_SnappedToGrid(CurorHitResult.Location, InGridSize);

      SetActorLocation(BuildPosition);
      break;
    }
    case ETurretState::ETS_Searching:
    {
      RotateTurret();
      if (EnemyArray.Num() > 0)
      {
        SetTurretState(ETurretState::ETS_InCombat);
      }
      break;
    }
    ...
}
```

❓ traceComplex란 무엇인가?

❓ 포탑의 행동로직을 tick말고 다른 곳에 넣을수도 있을까?