프로젝트 기술서 - Project Squad

기간	1-개월	제작 환경	Unreal 4, Visual Studio 2017	언어	C++, Blueprint
개요	1. 계속해서 나타나는 로봇을 처치하며 목표를 사수하는 웨이브형 게임. 제작 2. 조작 캐릭터 는 RTS 와 같이 여러 캐릭터를 번갈아 가며 선택 조작. 선택 되지 않는 캐릭터는 자체 AI 를 통해 로봇 처치.				
구현 목적	1. UE4 에서 제공하는 대부분의 기능을 사용, UE4 를 전반적으로 공부 및 습득 2. 시스템의 구조 및 대부분의 구현은 C++ 로 제작 할 것 3. 참고 가능 한 것은 결코 복사 붙여넣기 하지 않을 것 이해 한 것 만 직접 타이핑 혹은 응용 해서 구현 할 것				
이 프로젝트는 계속해서 제작되고 있으며 Github 를 통해 최신 소스코드를 확인 할 수 있습니다.(Github 링크) 혹 전체 프로젝트를 확인하고자 한다면 별도의 메일문의 부탁드립니다.					

주요 클래스 기술	**Core Class** **CoreCameraControl - Pawn 상속** - 실제 플레이어가 조작하는 Pawn - RTS 카메라 조작 내용을 그대로 구현. **CoreActCharacter – Character 상속** - 레벨 상에 구현될 Character 에 대한 기본 구조 정의 - FSM (상태 패턴) 구현을 위한 기초 함수 정의 - 적 체크, 시야 체크, HP Bar 등 설정 **CoreActState – Uobject 상속** - 상태패턴을 구현하기 위한 기본적인 상태 함수 정의 - 클래스 내에서 사용하기 용이하게 GameInstance 설정 및 GetWorld 재정의 **CoreActSightSphereComponent – SphereComponent 상속** - 기존 Actor 를 상속한 CoreActSight 를 대신하여 사용 (CoreActSight 사용 X) - 구(Sphere) 에 들어온 Actor 들을 적 혹은 아군이 선별 및 TArray 로 관리 - 적(Enemy) 일 경우 거리 와 전방각도를 통해 공격 할 수 있는 대상 액터인지 체크 및 관리 - 기본적으로 Collider Overlap 이벤트를 통해 위의 기능을 구현 **CoreWindowUWidget – UuserWidget 상속** - UI 상에 출력 될 UMG Widget 에 대한 기분 구조 정의. - Hud 클래스를 사용하지 않았으며 모든 UI 는 Widget 상속받아 사용 함. (이 부분 에서는 Hud 와 Widget 를 굳이 분리해야 하는가? 에 대한 궁금 증에서 시작 현재 결론은 Widget 만 사용하기로 결정. 실제로 Hud 클래스 를 유사한 기능을 정의하고자 했던 CoreRootUWidget 도 사용하지 않음)

주요 클래스 기술	**Hero / Robot Class** **공통사항.** - Hero/Robot ActCharacter, ActState 를 각각 Core Class 를 상속받아 제작. - Robot 액터 는 생성 즉시 비헤이비어 트리에 의해 자동 수행 - 각각 공격모션 은 Montage 를 통해 재생 , 블랜딩은 AnimBP 를 이용 - (문제점) 현재 FSM 구조는 1개의 단일 상태관 관리 가능. Hero 같은 경우 움직임과 공격 상태가 동시에 동작할 수 있음에도 복수상태 작동 관리 작업 이 미흡함. 현재 개선 노력중 **HeroActControl - CoreCameraControl 상속** - 레벨에 생성 된 Hero 들을 컨트롤 함수 등 정의 (Tap 를 통해 레벨 에 생성 된 Hero 들을 순차적으로 선택, 선택 된 Hero 의 이동 혹 공격 등 조작 함수 정의) - 레벨에 생성 된 Hero 개별 간 조작 모드 를 설정 (선택 되지 않았으면 AIControl 를 통해 자동 이동, 공격 등 수행 (구현 중) 선택 된 이후에는 TPS 혹은 RTS 처럼 조작할 지 여부 설정) **HeroGunWeapon / Projectile – Actor 상속** - Hero 가 사용할 무기(라이플) 및 발사체 정의 - Weapon - 무기의 발사체계 조작 함수 정의 (단발, 점사, 연발) - Projectile 은 ProjectileMovementComponent 를 통해 구현. **HeroBTTask_PushToPatrolPoint** - Hero 가 Behavior 를 이용하여 행동시 순찰 해야 할 TargetPoint 설정 해줌. - 이동 해야 할 TargetPoint 는 PSFixedGoalToProtect 에서 바로 얻어옴 **RoboAIController – AIController 상속** - UE4 에디터 상에서 제작한 BlackBoard / BehaviorTree 와 동기화. - BlackBoard 에 설정된 변수 를 설정 할 수 있는 함수 정의. - (문제점) 실제 BehaviorTree 에서는 가고자 하는 목표 점만 관여. 실제 로봇의 이동, 공격 타겟 이동, 공격, 다시 돌아오기 등은 RobotActState 에서 모두 정의 **RobotActAnimAttackNotifyState – AnimNotifyState 상속** - 로봇의 근접 공격 모션 시 공격 애니의 특정 타이밍에 공격 한 대상 들에게 데미지를 주는 등 공격 행위에 대한 클래스 정의 - (문제점) NotifyTick 에서 SkeletalMesh 를 통해 접근한 오너(로봇) 액터 를 통해서 CoreActSightSphereComponent 에 접근, 공격 가능한 액터 를 검색 도중 CoreActSightSphereComponent 에서 관리되는 TArray 들 자체 가 메모리에 제대로 올라오지 않는 경우가 간혈적으로 생김. 현재 수정 중

주요 클래스 기술	**Manager Class** **PSBaseManager – UObject 상속** - 기본적으노 GameInstance 에서 관리될 것을 전제로 제작 - 각종 매니저 객체가 필요할 내용 함수 정의 (GetWorld 함수 재정의 등) **PS[Actor, Data, Widget]Manager – PSBaseManager 상속** - **Actor Manager** 는 CoreActCharater를 상속 받은 캐릭터 들이 레벨 상에 생성 된다면 자동으로 ActorManager 의 TArray 에 리스트로 관리 됨. - Charater 들의 UniqueID 를 부여함. - Character 들에게 BoradCast 와 비슷한 형식의 상태 변경 함수 등 관리 중인 객체들에게 동시에 호출할 수 있는 함수 등 제작 - **Widget Manager** 는 호출/생성 되는 UMG 의 생성 과정을 간소화(메모리에 올릴 UMG 객체들 별도의 폴더에서 관리 등) 함수 제공 - 생성, 출력 될 Widget 을 Stack 형태로 관리, (Push / Pop 함수 제공) - **Data Manager** 는 게임상에서 적용된 로컬데이터(CSV) 를 Load 및 관리 - 현재는 Data Manager 가 초기화 될 때 모든 로컬데이터를 Load 함.
	UMG Class - **Window** 폴더 내에 정의 된 Widget 들은 CoreWindowUWidget 화면 전체에 출력 되는 Widget 을 정의 - **Item** 폴더 내에 정의 된 Widget 들은 Window 를 부모 혹은 그 안에서 출력되는 개별적인 Widget 정의 - 특히 HP / 데미지 / 무기 상태(발사체계 등) 등 각각 필요한 Actor 들을 계속 Tick 에서 검사 출력하지 않고. Core/Hero 등 액터에서 정의하고 등록한 Delegate 함수에 함수를 바인딩 하여 액터에서 정보(HP, 데미지 등) 변경 시 별도의 조작없이 Widget 에도 정보 변경에 따른 출력이 되겠금 정의
	GameMode Class **UE4PSGameModeBase – GameModeBase 상속** - 기본적으로 필요한 변수(GameInstance , 매니저 객체 등) Load 및 정의 **PSFieldGameMode - UE4PSGameModeBase 상속** - 레벨 로딩 후 Game 시작시 출력 될 UI 설정, 로봇이 처치해야 하는 GoalActor 에 대한 HP UI 출력 등 설정. - 총 3개 의 웨이브 구간에 대한 시간 설정 및 Tick 통해 관리. - Game 성공, 실패 게임 전체 흐름 및 그에 따른 필요 함수 호출 정의. **PSLobbyGameMode - UE4PSGameModeBase 상속** - 로비 제작시 사용 예정

주요 클래스 기술	**그 외 Class** **PSFixedGoalToProtect – Actor 상속** - Robot 이 자폭하여 파괴해야 할 목표 액터 를 정의 - Delegate 함수를 이용하여 객체의 HP 가 줄어들 때마다 바인딩 된 함수를 통해 정보를 현재 이 액터의 정보를 알려줌 - Hero 가 Behavior 를 이용할 시 패트롤 이동 포인트 제공 **PSFieldRobotCreatePoint – TargetPoint 상속** - 로봇이 생성 되는 위치를 정의 . - 웨이브 번호에 따라 생성된 되는 로봇의 수 등을 조절 되겠금 정의
	Common Class . **PSDataSchema – UObject 상속**. - 각종 enum , 구조체 등 정의, enum 을 스트링으로 변경할 유틸리티 함수 정의 **PSGameInstance - GameInstance 상속** - 각종 매니저 객체 및 목표(로봇이 가서 처치(자폭)해야 할 객체) 등록 및 관리 - 웨이브 진행 시 각 로봇 생성 포인트에 생성 함수를 Call 하기 위한 함수 정의

외부 Assets	Assets 이름	Asset 구분
1	Robot Pack	Model / Animation
2	Rifle Animset Pro	Animation
3	Prototype Weapons	Model / Animation / Particle / Sound
4	Action Adventure Male	Model
해당 에셋 들 내부에서 구현 된 AnimBP 등 Blueprint 등 일체 사용하지 않았습니다.		

질문..	1. AnimNotifyState 에서 인자로 넘어오는 SkeletalMesh 를 통해 Owner Actor 을 가져와 Actor 안에 정의된 특정 객체(TArray 등) 를 참조시 제대로 메모리에 올라가지 않아 참조가 불가능한 경우가 있습니다. AnimNotifyState 질문 을 카페에 올렸고 답변에 따라 현재 수정 중이나 정확한 원인 설명을 찾지 못해 현재까지도 찾고 있습니다. 혹 이와 같은 문제점 에 대한 원인과 해결방법을 알려 주실 수 있으신지요? 2. Projectile 객체 같은 경우 Projectile 생성 후 Projectile 를 주관하는 Character 객체를 셋팅 합니다. 문제는 SpawnActor 함수를 통해 Projectile 생성 후 그 다음 구문에서 Projectile ->SetCharacter 함수를 호출 하는데 그 전에 Projectile 의 Collision 이벤트 가 먼저 호출 됩니다. 즉 생성자 자체에서 Charater 지정을 한다면 크게 문제가 없으나 현재 UE4 에서는 그러한 기능 지원을 찾지를 못했습니다. 혹 이럴 경우 해결방법을 알려주실 수 있으신지요?