

# 몬스터 헌터 월드 모작

N A M E : 권 성 호

T E L : 010-8874-6452

P A R T : PROGRAMING

**Efforts make all doors open**



# 목 차



I . 포트폴리오 개요 .....	1
1. 코드 기반 게임 제작.....	2
2. 서버 기반 게임 개발.....	2
3. 게임 지원 툴 제작.....	3
4. GoogleDocs를 이용한 문서와 일정 관리.....	3
5. 코드, 커밋 컨벤션.....	4
6. 데이터 기반의 게임 개발.....	4
7. 코드, 리소스에 대한 형상 관리.....	5
II . 기술 요약 및 UML .....	6

# I. 포트폴리오 개요

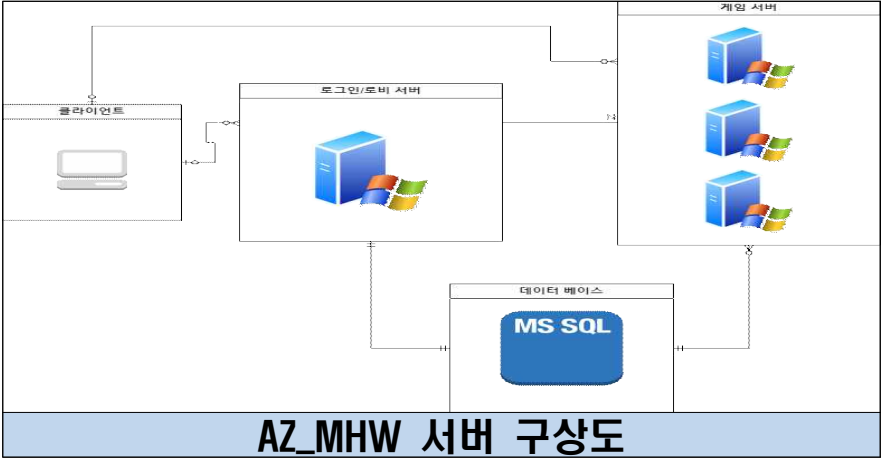
프로젝트 명	Project MHW(Monster Hunter World)
플랫폼	PC(Windows)
프로젝트 개요	
	
프로젝트 소개	헌팅액션 인기대작 '몬스터 헌터 월드'를 Unreal 5.1로 모작하자.
개발 기간	2023.4.03 ~ 2023.6.02
개발 환경	unreal 5.1.1, IOCP, MsSQL, NetCore3.1, GoogleDocs
게임 장르	MORPG
개발 인원	5명
설명	<p>언리얼 프로젝트에 대한 현업을 체험하고자 게임 제작에 대한 각 사항을 세분화하였다.</p> <ul style="list-style-type: none"> <li>- 코드 기반 게임 제작</li> <li>- 서버 기반 게임 개발</li> <li>- 게임 지원 툴 제작</li> <li>- GoogleDocs를 이용한 문서와 일정 관리</li> <li>- 코드, 커밋 컨벤션</li> <li>- 데이터 기반의 게임 개발</li> <li>- 코드, 리소스에 대한 형상 관리</li> </ul>
게임 개요	<a href="https://docs.google.com/presentation/d/1Ck2LKFmS7MVZE-wpCr3I_cNiaq2c7fjXOUH5tTaSF2k/edit?usp=sharing">https://docs.google.com/presentation/d/1Ck2LKFmS7MVZE-wpCr3I_cNiaq2c7fjXOUH5tTaSF2k/edit?usp=sharing</a>
GitHub URL	<a href="https://github.com/AZ-KGCA/AZ_MHW.git">https://github.com/AZ-KGCA/AZ_MHW.git</a>
동영상 링크	<a href="https://youtu.be/TL0zLTUghtk">https://youtu.be/TL0zLTUghtk</a>

# 코드 기반 게임 제작



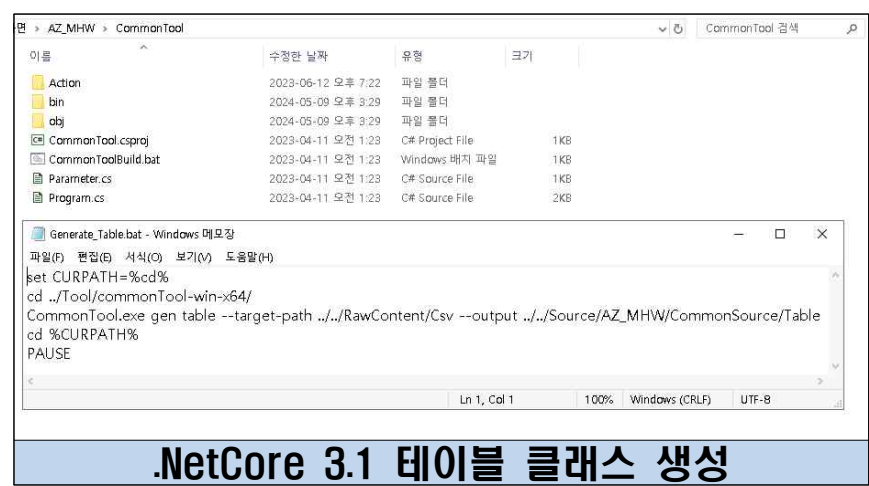
해당 프로젝트는 코드 기반으로 제작되었으며, 단순한 게임 개발을 넘어 실제 현업에서 이루어지는 프로세스를 체험하였습니다. 프로젝트를 철저히 모듈화하여 각자가 맡은 파트를 독립적으로 개발한 뒤, 이를 유기적으로 결합하는 방식으로 작업을 진행했습니다. 이러한 접근 방식은 팀원 개개인의 전문성을 최대한 발휘할 수 있도록 하였으며, 동시에 프로젝트의 유지보수성과 확장성을 크게 향상했습니다.

# 서버 기반 게임 개발



네트워크 이해도를 향상하고 서버 기반 게임 개발에서 발생하는 다양한 문제를 해결하고자, 우리는 서버 사이드 프로젝트를 진행했습니다. 이 과정에서 언리얼 엔진의 데디케이트 서버 방식이 아닌, 클라이언트를 네트워크 라이브러리로 랩핑한 후, 몬스터와 플레이어의 상태를 서버에서 처리하고 이를 클라이언트에 브로드캐스팅하는 방식을 채택했습니다. 이러한 접근은 언리얼 엔진에 국한되지 않고, 다양한 프로젝트에서 서버를 이해하고 구현할 수 있는 능력을 기르는 데 중점을 두었습니다. 이 경험을 통해 서버 사이드 개발의 핵심 원리와 실전적인 적용 방법을 깊이 이해할 수 있었습니다.

# 게임 지원 툴 제작



.NET Core 3.1을 활용하여 데이터 테이블 생성을 위한 게임 지원 툴을 개발했습니다. 이 프로젝트는 게임 제작과는 별도로 진행되었으며, 이를 통해 언리얼 C++과는 다른 C#이라는 매니지드 언어에 대한 깊은 이해를 쌓을 수 있었습니다. 특히, 테이블 클래스의 자동 생성 기능을 구현하여 작업 효율성을 크게 향상했습니다. 이 경험은 개발자로서의 역량을 다각적으로 확장하고, 다양한 프로그래밍 언어와 도구를 활용하여 게임 개발에 필요한 툴을 효과적으로 제작할 수 있는 능력을 키우는 계기가 되었습니다.

# GoogleDocs를 이용한 문서와 일정 관리



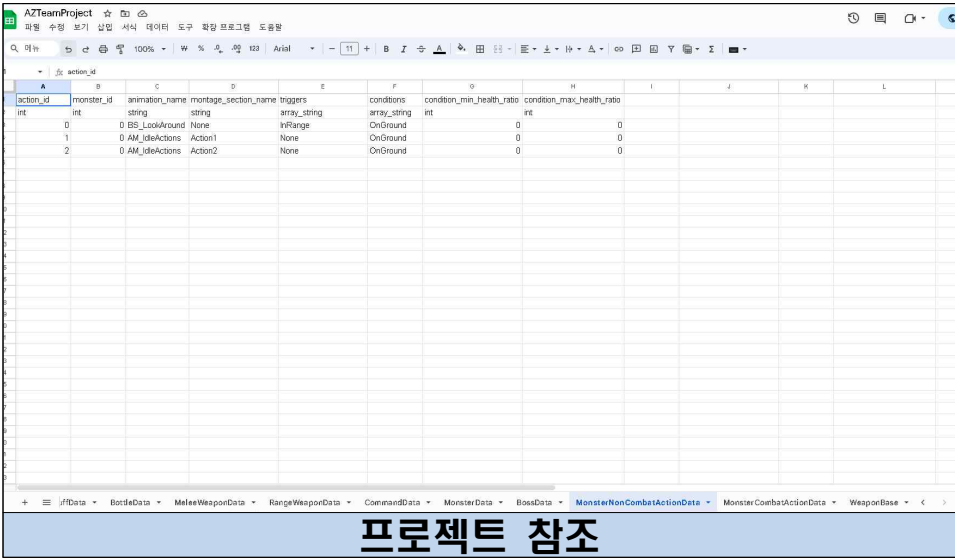
Google Docs와 Google Drive를 활용하여 공동 문서 작업 및 문서 공유를 진행했습니다. 이 도구들을 사용함으로써, 팀원 간의 실시간 협업이 가능해졌고, 프로젝트 진행 상황을 투명하게 공유할 수 있었습니다. 또한, 문서의 버전 관리와 변경 사항 추적이 용이해져, 의사소통이 원활하게 이루어졌으며, 이를 통해 개발 과정에서 발생할 수 있는 오류와 혼선을 최소화할 수 있었습니다. 이는 팀워크와 프로젝트 관리 능력을 한층 강화하는 데 도움이 되었습니다.

# 코드, 커밋 컨벤션

구글 C++ 스타일	<a href="https://jongwook.kim/google-styleguide/trunk/cppguide.xml">https://jongwook.kim/google-styleguide/trunk/cppguide.xml</a>
구글 C# 스타일	<a href="https://kaki104.tistory.com/720">https://kaki104.tistory.com/720</a>
Git Commit 스타일	<a href="https://karma-runner.github.io/latest/dev/git-commit-msg.html">https://karma-runner.github.io/latest/dev/git-commit-msg.html</a>
Git 브랜치 전략	<a href="https://gmlwjd9405.github.io/2018/05/11/types-of-git-branch.html">https://gmlwjd9405.github.io/2018/05/11/types-of-git-branch.html</a>
코드, 커밋, 브랜치 컨벤션	

코드 및 커밋 컨벤션을 프로젝트에 적용하면서 코드의 일관성과 가독성을 크게 향상할 수 있었습니다. 이를 통해 팀원 간의 협업이 원활해지고, 코드 리뷰 과정에서 발생할 수 있는 오류와 혼선을 최소화할 수 있었습니다. 또한, 체계적인 커밋 메시지 작성으로 변경 사항을 명확하게 기록하여, 프로젝트의 버전 관리와 문제 추적이 용이해졌습니다. 이러한 경험은 개발자로서의 역량을 다각도로 확장하는 데 기여했으며, 향후 다양한 프로젝트에서 효과적으로 협업하고 코드를 관리할 수 있는 능력을 키우는 계기가 되었습니다.

# 데이터 기반의 게임 개발

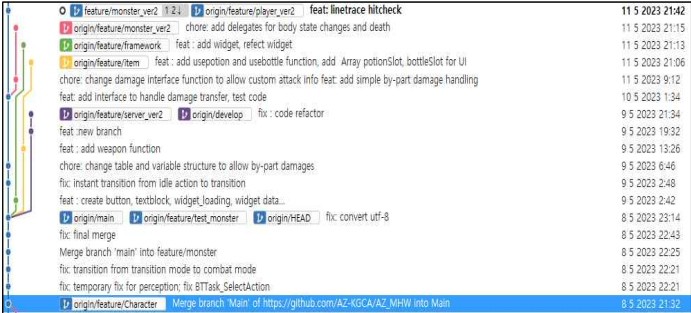


action_id	monster_id	animation_name	montage_section_name	triggers	conditions	condition_min_health_ratio	condition_max_health_ratio
0	0 BS_LockAround	None	None	InvRange	OnGround	0	0
1	0 AM_IdleActions	Action1	None	None	OnGround	0	0
2	0 AM_IdleActions	Action2	None	None	OnGround	0	0

프로젝트 참조


데이터 테이블을 기반으로 게임 개발을 진행하면서, 코드의 재사용성과 모듈화를 극대화하고, 코드의 간결성을 유지하는 데 중점을 두었습니다. 이 접근 방식을 통해 반복적인 작업을 줄이고, 다양한 게임 요소를 효율적으로 관리할 수 있었습니다. 모듈화된 구조 덕분에 각 기능을 독립적으로 개발하고 유지 보수할 수 있었으며, 코드의 간결성은 가독성을 높여 협업을 원활하게 했습니다. 이러한 경험은 게임 개발의 효율성을 높이고, 향후 다양한 프로젝트에서 재사용할 수 있는 구조를 설계하는 데 중요한 기반이 되었습니다.

# 코드, 리소스에 대한 형상 관리



11 5 2023 21:42  
11 5 2023 21:15  
11 5 2023 21:13  
11 5 2023 21:06  
11 5 2023 9:12  
10 5 2023 1:34  
9 5 2023 21:34  
9 5 2023 19:32  
9 5 2023 13:26  
9 5 2023 6:46  
9 5 2023 2:48  
9 5 2023 2:42  
8 5 2023 23:14  
8 5 2023 22:43  
8 5 2023 22:29  
8 5 2023 22:21  
8 5 2023 22:21  
8 5 2023 21:32

GitHub 트리



이름	수정된 날짜	유형	크기
.svn	2023-06-12 오후 7:27	파일 폴더	
AZ	2023-06-12 오후 7:30	파일 폴더	
Collections	2023-06-12 오후 6:39	파일 폴더	
Developers	2023-06-12 오후 7:30	파일 폴더	
EnvironmentPack2	2023-06-12 오후 7:30	파일 폴더	
InfinityBladeFireLands	2023-06-12 오후 7:30	파일 폴더	
InfinityBladeGrassLands	2023-06-12 오후 7:30	파일 폴더	
InGameMap	2023-06-12 오후 7:30	파일 폴더	
UltraDynamicSky	2023-06-12 오후 7:30	파일 폴더	

리소스 SVN 관리

언리얼 프로젝트에서 GitHub와 SVN을 병행 사용한 결과, 코드와 리소스를 각각의 특성에 맞게 최적화된 방식으로 관리할 수 있었습니다. GitHub는 코드의 효율적인 버전 관리와 협업을, SVN은 리소스의 안정적 관리를 가능하게 하여, 프로젝트의 전체적인 효율성과 안정성을 크게 향상했습니다. 이러한 관리 방식은 프로젝트 진행 중에 발생할 수 있는 다양한 문제를 사전에 방지하고, 팀원 간의 협업을 극대화하는 데 중요한 역할을 했습니다.

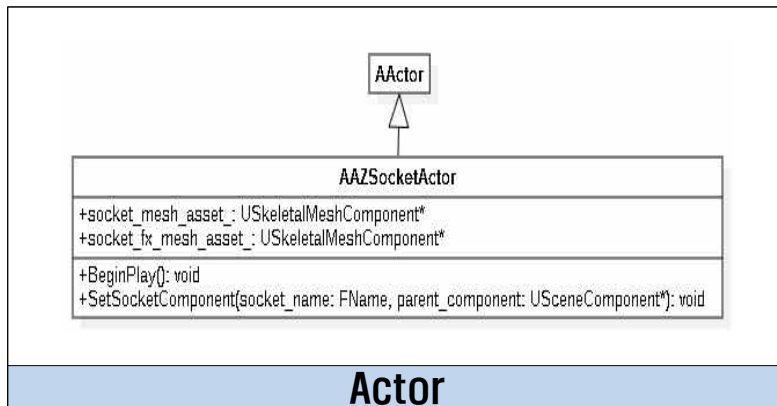
## II. 기술요약 및 UML

### 1. 코드 기반 프로젝트

해당 프로젝트는 코드 기반으로 제작되었다. 이를 통해 언리얼 C++의 이해와 구조를 깊게 학습할 수 있었다. 팀 프로젝트로 진행하면서 각자의 역할을 세분화하고 블루프린트 및 리소스 충돌과 병합 문제를 해결할 수 있었다. 또한, 코드 기반으로 작성하여 디버깅과 버전 관리가 용이해졌고, 확장성도 확보할 수 있었다. 비록 단기적으로 어려움이 있었지만, 언리얼 엔진을 더 잘 이해할 좋은 기회였다.

#### 1. Actor

게임 레벨에 배치될 수 있는 기본 클래스. 월드에 있는 가장 기본 클래스이다. 게임 오브젝트의 속성과 행동을 정의할 수 있다.

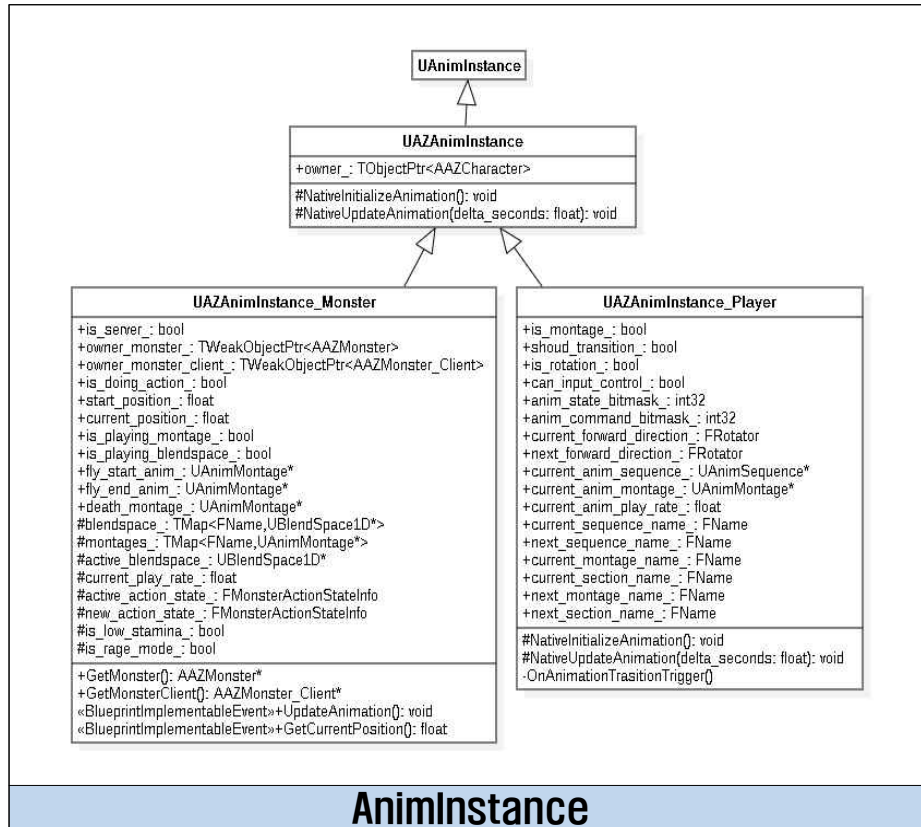


항목	내용	비고
AAZSocketActor	캐릭터의 소켓에 부착되는 Actor이다. 무기와 이펙트의 위치 및 Mesh를 담당하고 있다.	



## 2. AnimInstance

애니메이션을 만들기 위해 필요한 데이터나 기능들을 모아둔 클래스. 애니메이션 상태와 전환, 변수 제어 및 애니메이션 로직 구현을 관리하는 데 사용된다.

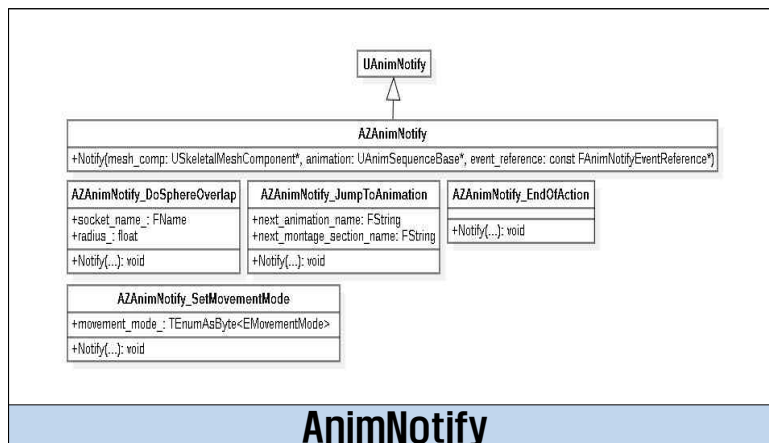


**AnimInstance**

항목	내용	비고
AZAnimInstance	UAnimInstance를 상속받은 기본 클래스	
AZAnimInstance_Monster	몬스터의 공통 애니메이션 데이터를 저장, 체크하기 위한 클래스	
AZAnimInstance_Player	플레이어의 애니메이션 데이터를 저장, 체크, 변경하기 위한 클래스	

### 3. AnimNotify

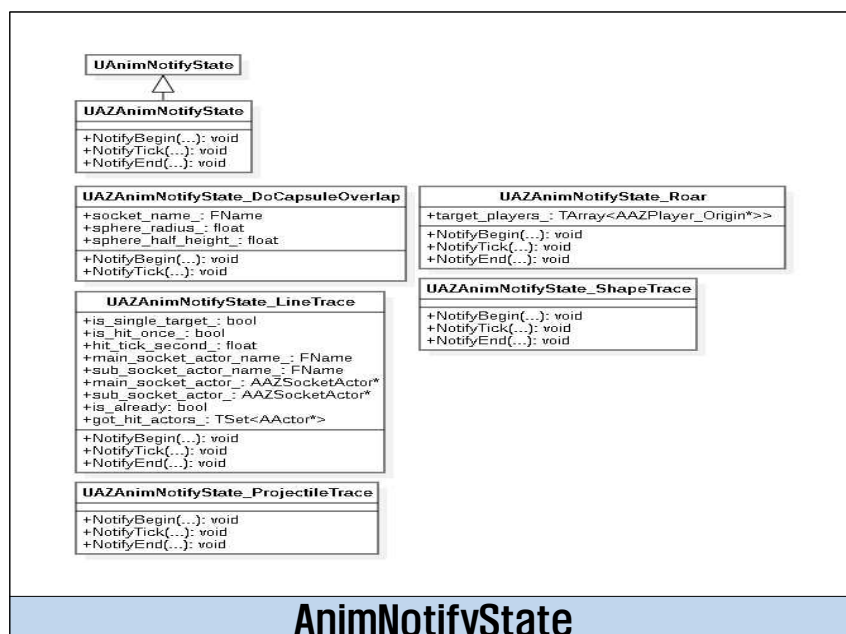
애니메이션 특정 시점에 트리거되는 알림 클래스. 애니메이션의 진행 상황에 따라 다양한 작업을 수행할 수 있음.



항목	내용	비고
AZAnimNotify	UAnimNotify를 상속받는 기본 클래스	
AZAnimNotify_DoSphereOverlap	AZAnimNotify를 상속받는 Notify클래스	해당 클래스는 블루프린트를 통해 새로 만들어진다. 각 액션에 맞게 애니메이션이나 몽타주(세션)에서 노티파이가 실행되어 진다.
AZAnimNotify_EndOfAction		
AZAnimNotify_JumpToAnimation		
AZAnimNotify_SetMovementMode		

### 4. AnimNotifyState

AnimNotify의 확장형으로, 애니메이션의 시작과 끝 시점 사이에 걸쳐서 어떤 동작을 수행할 수 있다. 애니메이션 상태표현, 시작과 끝처리, 홀드, 캐스팅 같은 지속적인 상태 확인에 이용



항목	내용	비고
AZAnimNotifyState	UAnimNotifyState를 상속받는 기본 클래스	
AZAnimNotifyState_DoCapsuleOverlap	공격 애니메이션 중 캡슐 충돌을 체크 하기 위한 클래스	
AZAnimNotifyState_LineTrace	공격 애니메이션 중 라인 충돌을 체크 하기 위한 클래스	
AZAnimNotifyState_Roar	포효 애니메이션의 액션을 체크하기 위한 클래스	

## 5. BehaviorTree

NPC의 AI를 담당. 블랙보드와 비헤비어 트리 두 가지 유형의 애셋의 조합으로 AI가 만들어짐

### 5.1 Composite

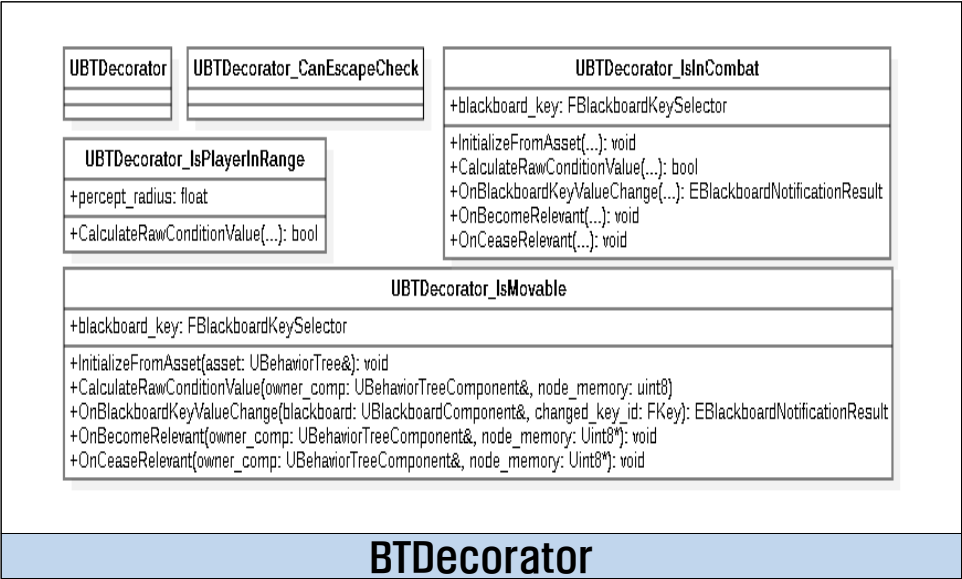
Selector-하나만 성공이여도 성공으로 끝남(하나라도 성공하면 성공)

Sequence-하나라도 실패하면 끝남(모두 성공해야 성공)

Simple Parallel-동시에 진행됨(움직임 + 사격) = 움직이면서 사격

### 5.2 BTDecorator

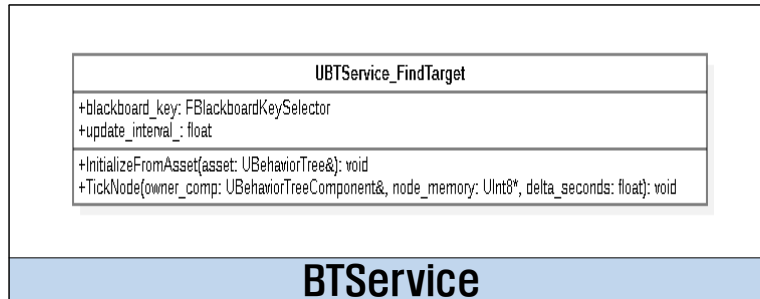
노드의 조건절로 이용. 해당 노드가 실행 여부를 판단.



항목	내용	비고
BTDecorator_IsInCombat	결투 상태인지 블랙보드를 체크	
BTDecorator_IsMovable	움직일 수 있는 상태인지 블랙보드를 체크	
BTDecorator_IsPlayerInRange	Player와의 거리를 체크한다.	

## 5.3 BService

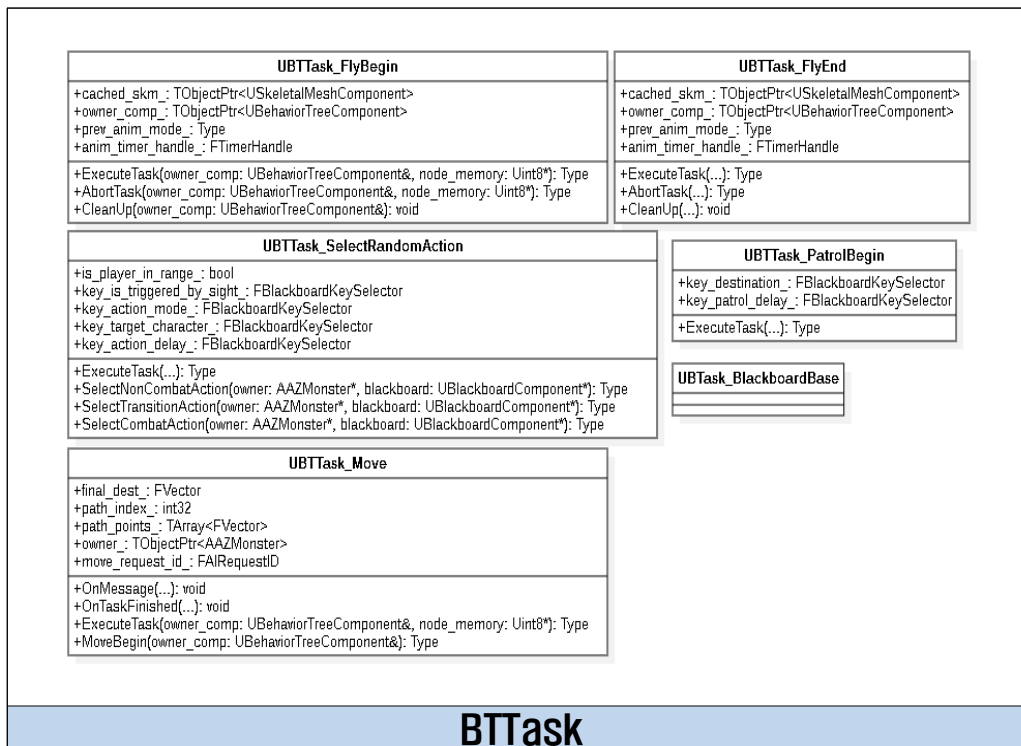
일정 주기동안 실행되고 블랙보드의 내용을 업데이트 한다.



항목	내용	비고
BService_FindTarget	일정 주기로 타겟 캐릭터를 블랙보드에 셋팅한다.	

## 5.4 BTTask

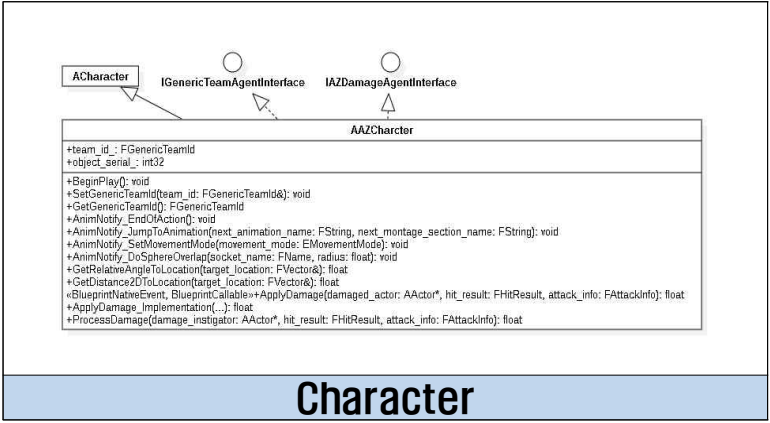
하나의 액션 이나 블랙보드의 값 조정과 같은 작업을 한다.



항목	내용	비고
BTTask_FlyBegin	몬스터의 상태를 변경 및 전파하고 애니메이션을 업데이트한다.	
BTTask_FlyEnd	몬스터의 상태를 마무리 및 전파하고 애니메이션을 업데이트한다.	
BTTask_Move	이동 경로를 업데이트하고 움직인다.	
BTTask_PatrolBegin	일정 거리를 순찰한다. 일정 조건에 만족하지 않으면 실패한다.	
BTTask_SelectRandomAction	idle, 액션 전의, 전투 돌입 중 랜덤으로 상태를 지정한다.	

# 6. Character

Pawn에서 더 복잡한 행위를 위한 클래스. 플레이어 캐릭터나 NPC를 구현을 하는데 사용됨.  
Mesh, Movement, 충돌처리 등 이 추가 되었다.

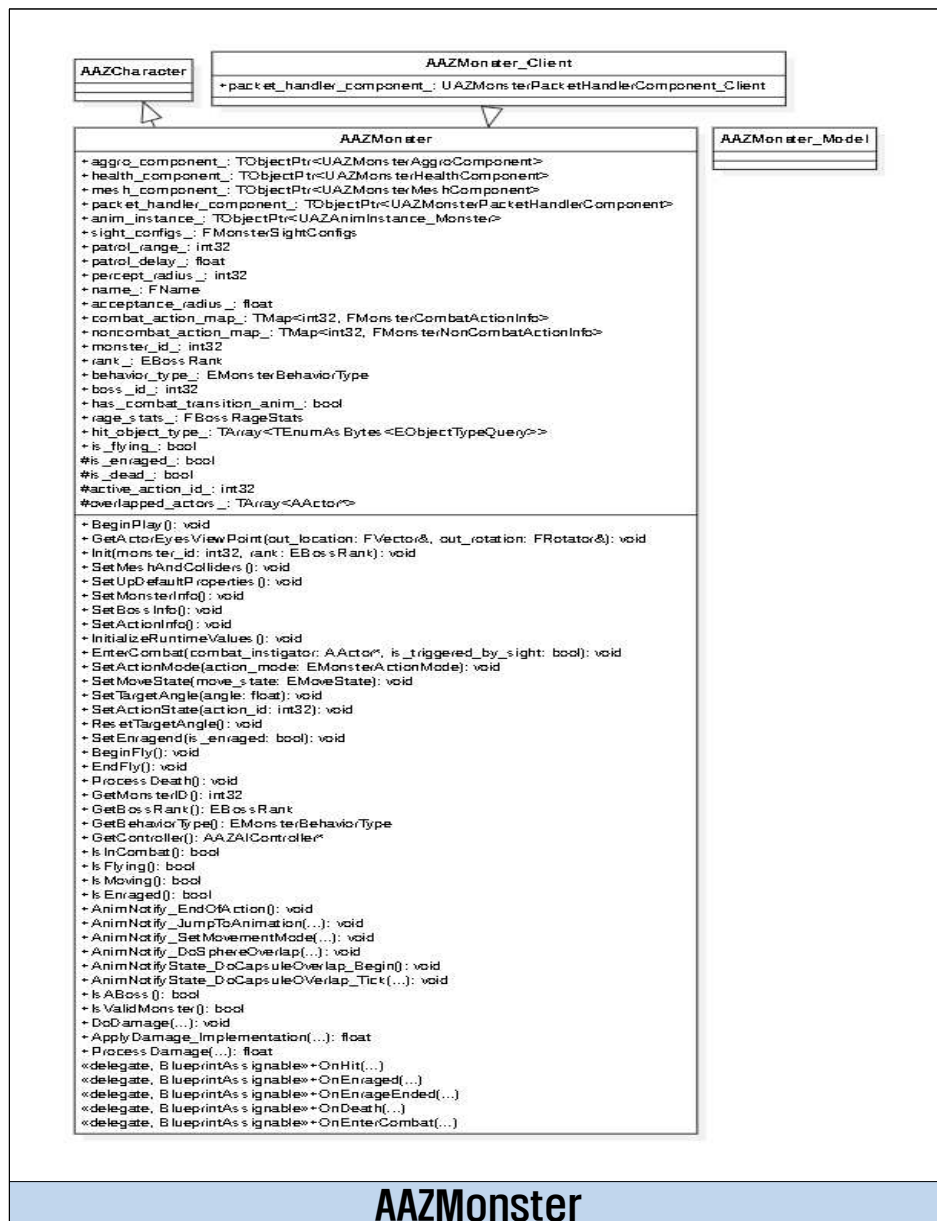


Character

항목	내용	비고
AAZCharacter	플레이어와 몬스터의 기본 클래스. ACharacter를 상속받는다.	
IGenericTeamAgentInterface	적군과 아군을 구분하기 위한 Team관련 인터페이스	
IAZDamageAgentInterface	캐릭터의 데미지 처리를 위한 커스텀 인터페이스	

## 6.1 Monster

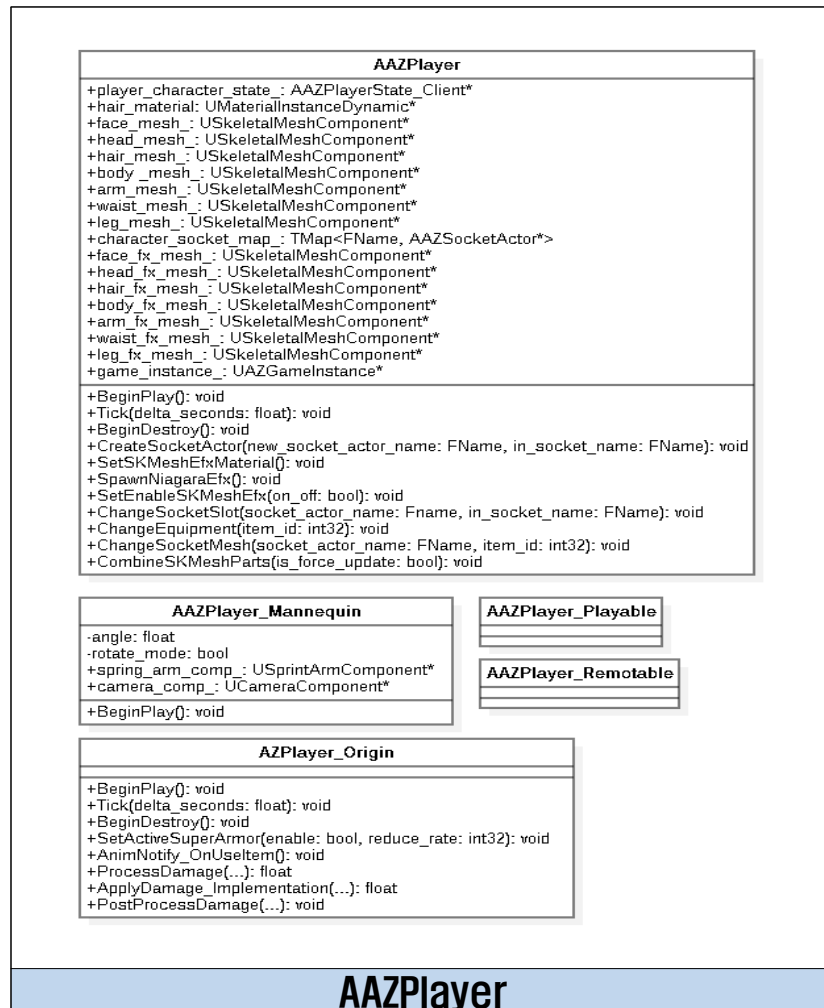
AAZCharacter를 상속받은 몬스터 클래스.



항목	내용	비고
AZMonster	몬스터를 담당하는 기본 클래스. 서버를 담당하고 있다.	
AZMonster_Client	몬스터를 담당하는 기본 클래스. 클라이언트를 담당하고 있다. AZMonster를 상속받고 있다.	
AZMonster_Model	몬스터의 기본 모델을 담당하는 클래스	

## 6.2 Player

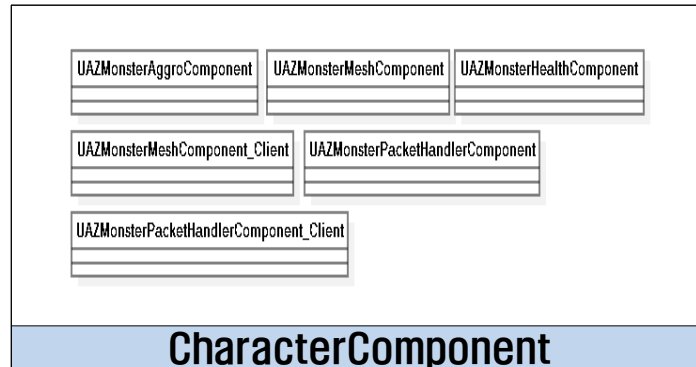
AAZCharacter를 상속받은 플레이어 클래스.



항목	내용	비고
AZPlayer	플레이어를 담당하는 기본 클래스.	
AZPlayer_Mannequin	장비 창을 담당하는 플레이어 클래스.	
AZPlayer-Origin	서버에서 플레이어의 상태를 담당하는 클래스.	
AZPlayer_Playable	유저가 조작하는 플레이어 클래스.	
AZPlayer_Remotable	서버가 움직이는 플레이어의 클래스.	

## 7. CharacterComponent

독립적으로 존재가 불가능한 액터에 포함시켜 하나의 기능으로 구현되는 모듈. AActor의 기능을 확장하거나, 특정 동작을 수행하게 사용됨

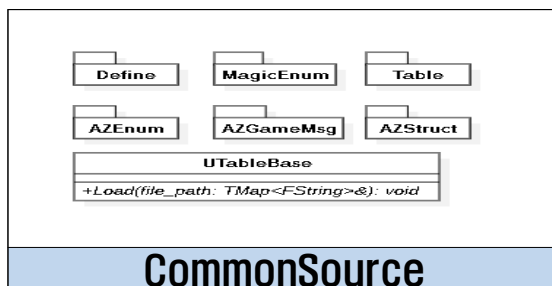


항목	내용	비고
AZMonsterAggroComponent	몬스터에 부착되어 타겟 플레이어를 체크하는 기능을 수행하는 컴포넌트.	
AZMonsterHealthComponent	몬스터의 부위별 데미지를 체크하기 위한 기능을 수행하는 컴포넌트	
AZMonsterMeshComponent	몬스터의 부위별 상태를 조정하기 위한 컴포넌트	
AZMonsterMeshComponent_Client	서버에서 수신된 몬스터의 부위별 상태를 조정하기 위한 컴포넌트	
AZMonsterPacketHandlerComponent	몬스터의 상태를 각 클라이언트에게 전송하기 위한 컴포넌트	
AZMonsterPacketHandlerComponent_Client	서버에서 수신된 몬스터의 상태를 업데이트하기 위한 컴포넌트	



## 8. CommonSource

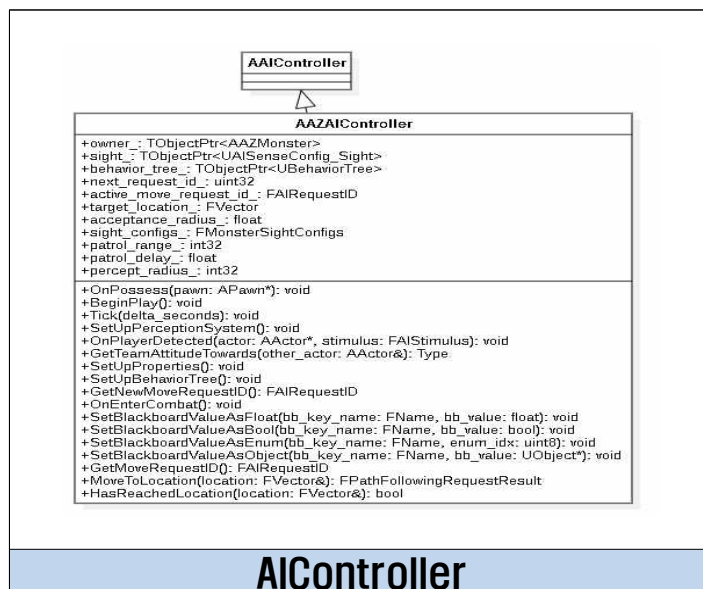
공용적으로 사용되는 라이브러리, 열거형, 테이블, 구조체등의 모음



항목	내용	비고
AZEnum	프로젝트에서 사용되는 열거형	
AZLog	프로젝트에서 사용되는 매크로 로그 함수 언리얼 로그 시스템을 래핑	
AZStruct	언리얼 포맷, 일반 포맷의 데이터 구조체 모음	
TableBase	프로젝트에서 사용되는 데이터 테이블의 추상클래스. 각 데이터 테이블은 해당 테이블을 상속받게 된다.	
MagicEnum	enum 라이브러리. Enum을 String으로 변환하기 위해 사용.	
Table	CSV를 로드하기 위한 데이터 테이블 클래스 CommonTool로 해당 테이블을 생성한다.	

## 9. AIController

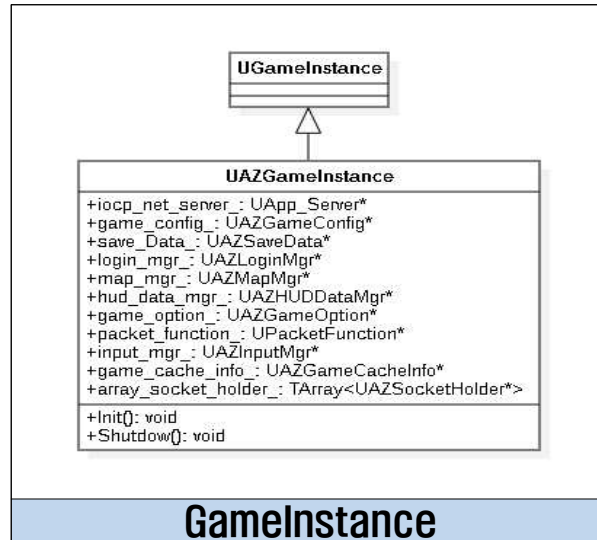
AI를 제어하기 위한 컨트롤러 기본 클래스이다.



항목	내용	비고
AZAIController	AAIController를 상속받는 공용 몬스터 AIController 공용 AIController로 공용화된 데이터를 세팅한 후 세팅된 값을 토대로 몬스터의 움직임을 제어한다.	

## 10. GameInstance

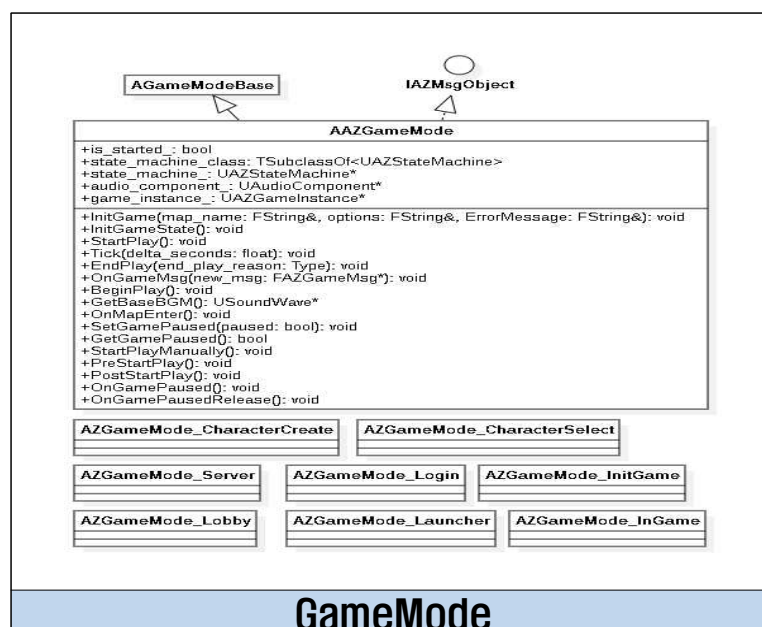
게임(에디터)이 시작할 때 만들어져 게임이 종료될 때까지 메모리에 존재하는 클래스. 게임의 전반적인 상태와 데이터를 관리한다.



항목	내용	비고
UAZGameInstance	UGameInstance를 상속받는 커스텀 클래스. 인풋, 네트워크, 맵정보, 게임시퀀스, 게임설정등, 레벨의 이동에 상관없이 사용되어야 할 Actor를 생성한다.	

## 11. GameMode

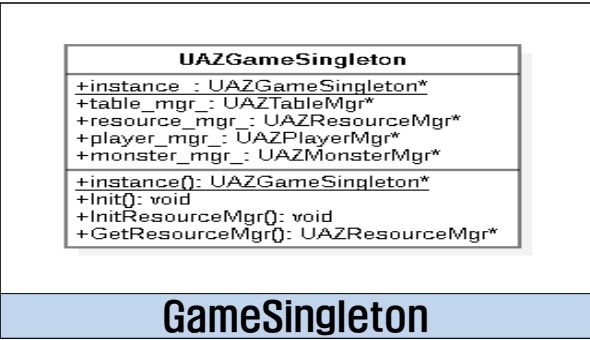
레벨당 월드 셋팅 되어 인스턴스가 생성되어 한 번에 하나씩만 존재. 게임의 규칙과 흐름을 정의 및 제어



항목	내용	비고
AZGameMode	기본 게임모드 공통으로 레벨에서 사용되는 기능을 정의한다.	
AZGameMode_CharacterCreate	캐릭터 생성창 레벨에서 사용되는 게임모드	
AZGameMode_CharacterSelect	캐릭터 선택창 레벨에서 사용되는 게임모드	
AZGameMode_InGame	인게임에서 사용되는 게임모드	
AZGameMode_InitGame	게임 가장 먼저 진입 시 서버인지 클라인지 확인하는 부분에서 사용되는 게임모드	
AZGameMode_Launcher	게임 시작 게임모드	
AZGameMode_Login	로그인 창 게임모드	
AZGameMOde_Server	게임 서버에서 사용되는 게임모드	

## 12. GameSingleton

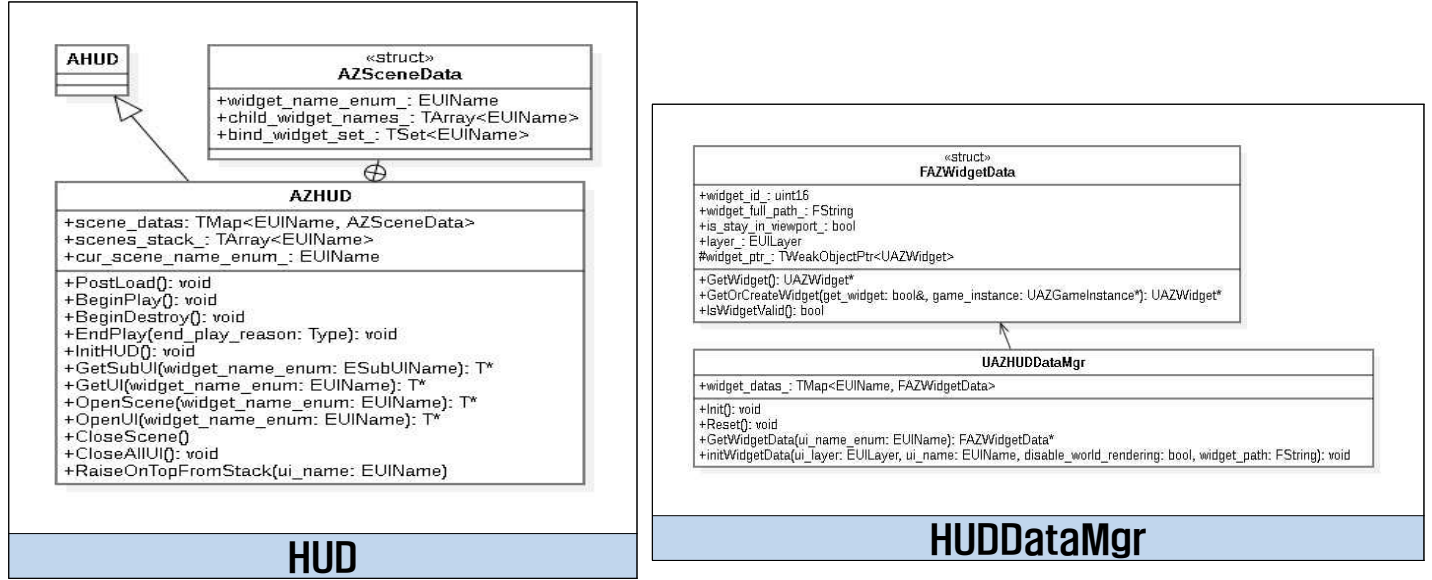
엔진 단위로 (앱단)에서 보관할 데이터나 설정에 필요한 데이터를 저장



항목	내용	비고
UAZGameSingleton	각 리소스의 관리, 데이터 테이블 업로드 등 앱의 상태 데이터등을 관리한다.	

## 13. HUD

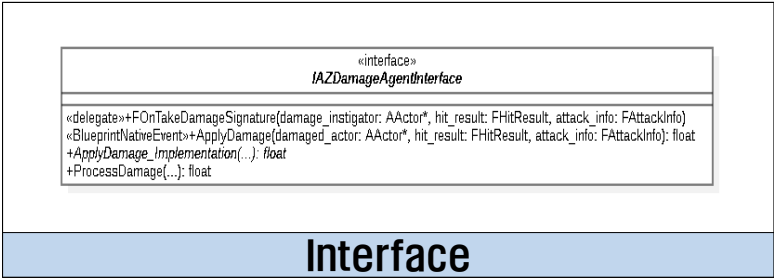
게임의 디스플레이를 담당한다. Widget의 생성 및 관리를 담당하는 클래스



항목	내용	비고
AZHUD	AHUD를 상속받는 커스텀 클래스 Widget하나를 하나의 화면으로 셋팅하며 이후 커스텀 Widget의 상태에 따라 스택 형식으로 UI를 관리하게 된다. 하나의 HUD는 GameMode에서 하나의 레벨을 담당하게 된다. 예)AZHUD_CharacterCreate, AZHUD_CharacterSelect, AZHUD_InGame, AZHUD_Launcher, AZHUD_Lobby, AZHUD_Login, AZHUD_StartGame	
AZSceneData	Scene에서 Widget의 자식과 스택을 관리. AZHUDDataMgr과 결합하여 위젯을 관리하게 된다.	
UAZHUDDataMgr	Widget리소스를 초기화하고 Widget의 생성과 관리를 담당한다.	
FAZWidgetData	AZHUDDataMgr와 결합하여 Widget의 정보를 담고 있다. AZHUDDataMgr는 해당 정보를 가지고 Widget을 관리한다.	

## 17. Interface

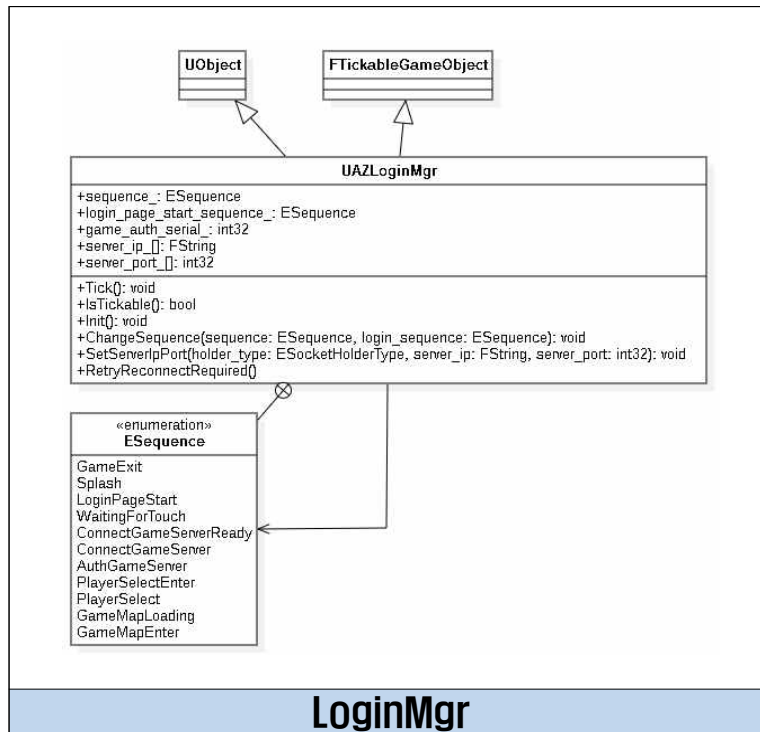
클래스 간에 공통된 기능을 정의하고 구현하는데 사용. 클래스 상속 계층과는 무관하게 여러 클래스에서 동일한 메서드를 구현할 수 있도록 한다.



항목	내용	비고
AZDamageAgentInterface	캐릭터의 데미지를 처리하기 위한 인터페이스. AZCharacter에 상속되어 Damage메소드를 처리하게 된다.	

## 18. Login

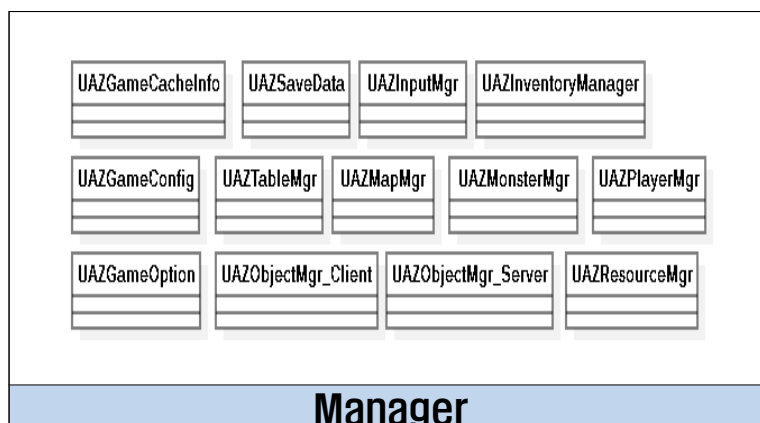
전체적인 게임시퀀스를 담당한다. 게임 진행의 절차적인 로직을 스텝에 맞춰 진행한다.



항목	내용	비고
AZLoginMgr	서버와의 접속 및 게임의 스텝을 담당하는 매니저 클래스 각 스텝에 따라 절차에 맞는 서버와의 접속, Widget 생성, 로그인 절차, 게임 레벨 전환을 담당한다. 각 매니저 클래스와 상호작용을 하여 게임을 진행한다.	

## 19. Manager

GameInstance, GameSingleton에서 사용되는 Manager의 모음, 프로젝트에서 전역적으로 사용되어야 할 기능을 담당한다.



## 20. PlayerController

Pawn과 그것을 제어하려는 사람 사이의 인터페이스. 플레이어가 캐릭터나 다른 폰을 조작할 수 있도록 중간 관리 역할을 한다.

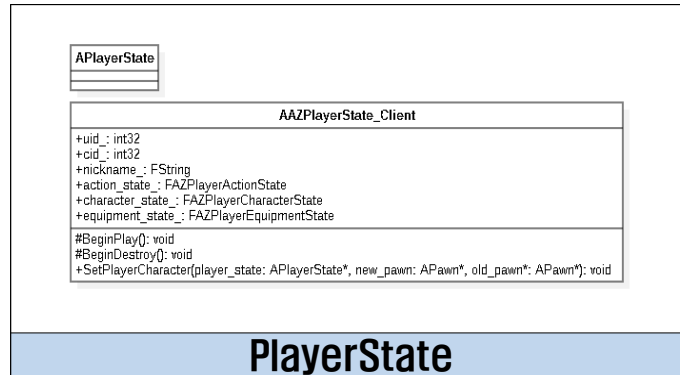


### PlayerController

항목	내용	비고
AZPlayerController	APlayerController를 상속받는 커스텀 클래스. 마우스 커서 표시 여부와 같이 공통으로 사용되는 데이터, 메소드를 정의한다.	
AZPlayerController_InGame	AZPlayerController를 상속받는 인게임 컨트롤러 클래스 실제 클라이언트가 플레이어를 움직이고 서버에서 플레이어들의 상태를 업데이트 받아 반영하는 클래스	
AZPlayerController_Server	AZPlayerController를 상속받는 서버 컨트롤러 클래스 클라이언트의 입력값을 서버 캐릭터로 전달하는 역할 및 입력의 결과를 각 클라이언트에게 전파하는 역할 수행	

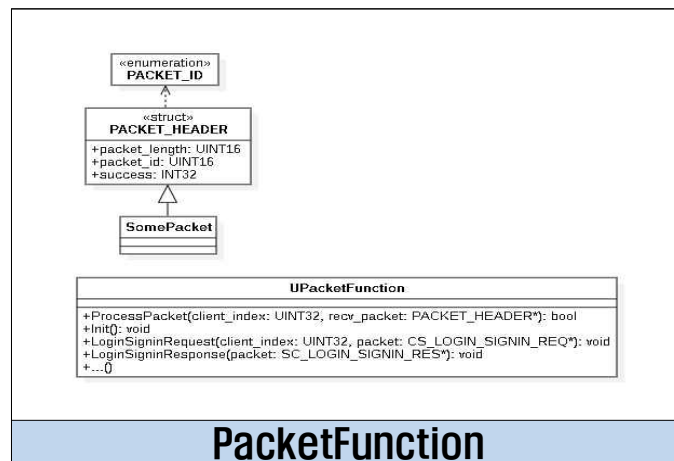
## 21. PlayerState

플레이어의 상태를 관리.



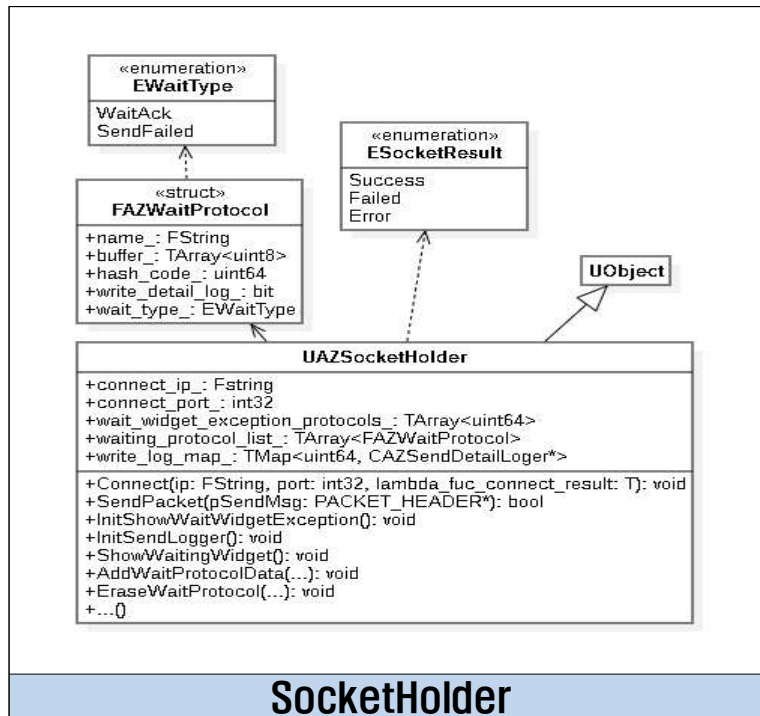
## 22. PacketFunction

패킷의 공용된 처리와 송수신 작업을 위한 코드 모음. 모든 송수신 패킷의 처리를 한곳으로 네트워크 처리를 한눈에 처리할 수 있게 하였다.



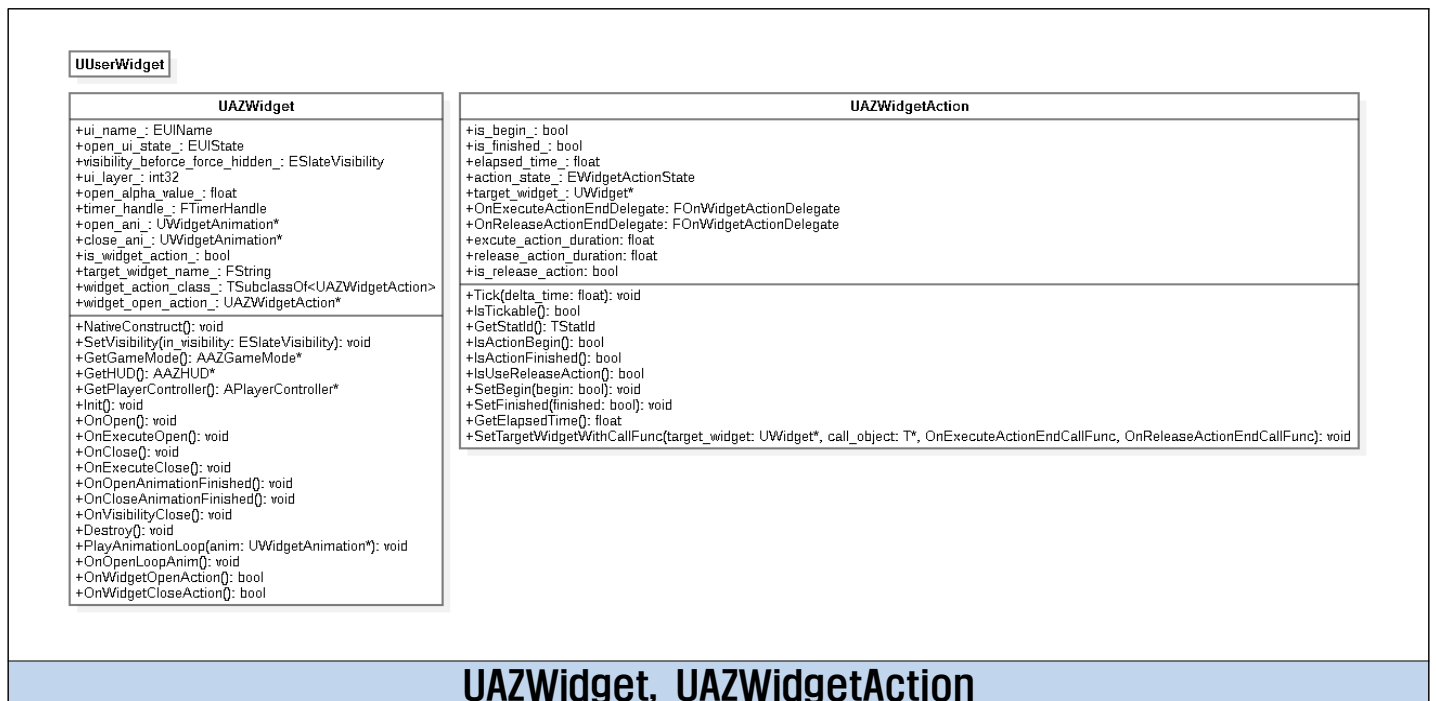
## 23. SocketHolder

네트워크 라이브러리를 랩 핑하여 패킷의 송수신 인풋 제어, 로그 처리를 담당하고 있다.

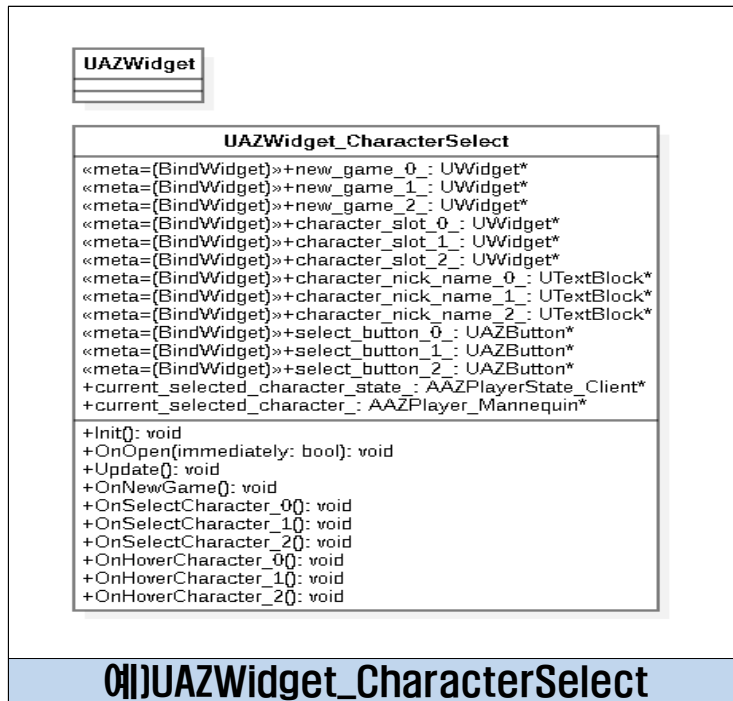


## 24. Widget

얼리얼 엔진에서 사용자 인터페이스를 구성하는 클래스. 플레이어와 게임 간의 상호작용을 관리할 수 있다.







항목	내용	비고
UAZWidget	UUserWidget을 상속받는 커스텀 클래스. 유저는 해당 클래스를 상속받아 위젯을 만든다. 공통적인 UI애니메이션, UI상태에 따른 액션을 바인딩한다.	
UAZWidgetAction	위젯의 액션을 정의한다.	
UAZWidget_CharacterSelect	UAZWidget 상속받아 사용되는 클래스. 각 멤버는 Widget블루프린트의 리소스에 1:1 매칭되어 각 액션에 따른 행위를 정의하게 된다.	