[PORTFOLIO]

이름 : 권오현

번호 : 010.2719.5330

이메일 : ohhyun5442@gmail.com

< 목차 >

1. 이사만루 스크린 야구..... 3

-

2. 이사만루3 10

이사만루 스크린 야구

1. 게임 소개

- 게임 : 이사만루 스크린야구
- 개발 환경
    - C# 기반의 Unity3D .Net 2.X, .Net 4.X
    - PC, Mac & Linux Standalone 플랫폼
    - TortoiseSVN
    - FTP를 사용한 리소스 관리
    - MySQL 기반의 DB
- 담당 파트 : 스크린 야구 메인 클라이언트
- 담당 업무 : 콘텐츠 개발 및 코드 리펙토링
- 개발 기간 : (2018. 07 ~ 2020. 03)

<하이라이트>

2. 세부 목차

NGUI 5

런처 6

가상 키보드 7

게임 모드 8

그래픽 환경 툴 9

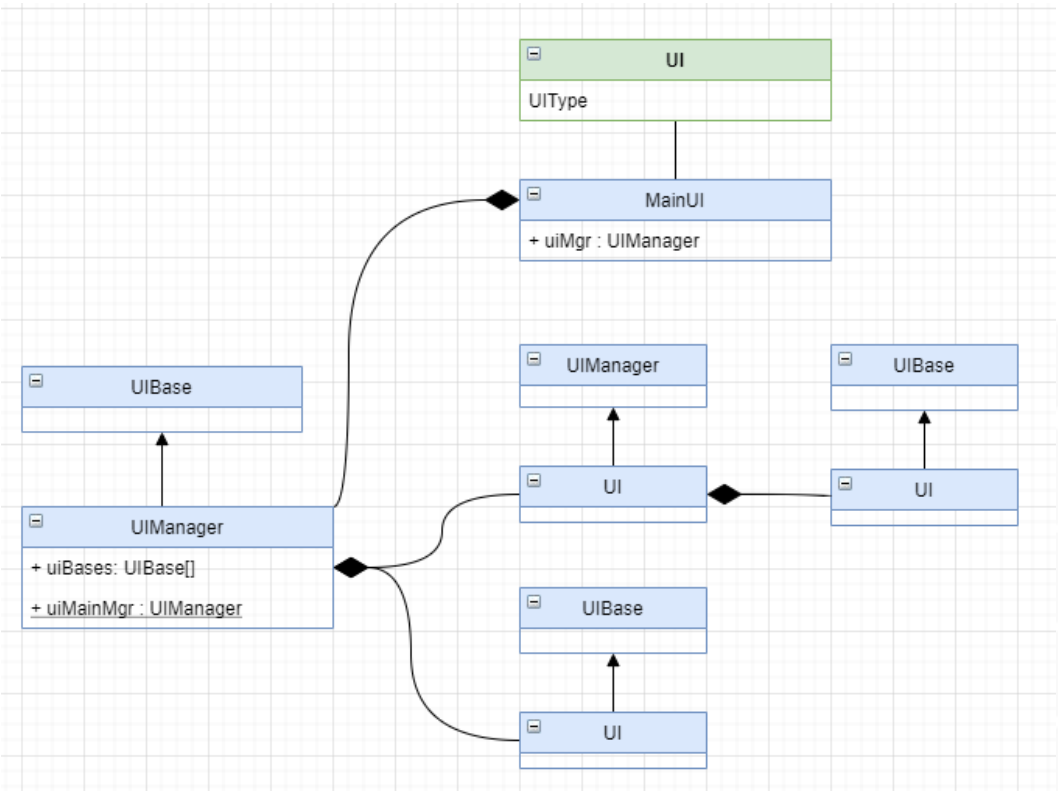
3. 콘텐츠 소개

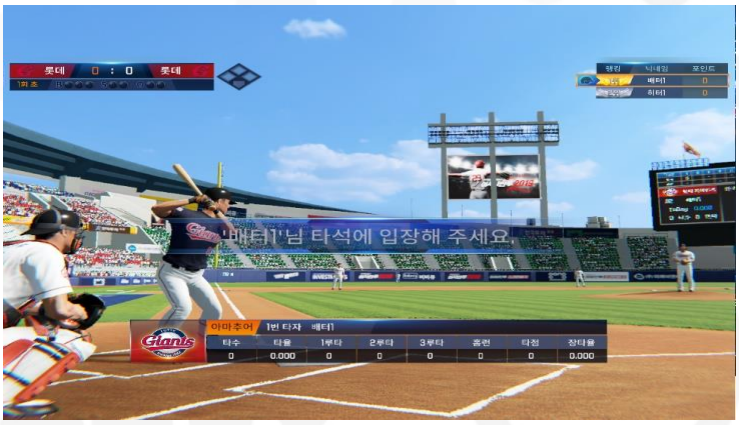
런처

1. NGUI

- 사용 목적 : 유니티에서 지원하는 UI로 게임에서 유저들이 게임 내 정보 등을 볼 수 있도록 하기 위해 사용했습니다.

- 콘텐츠 설명 : 유니티에서 지원하는 UI중 하나로 MainUI에서 UI를 생성했으며 Text, Image, Button, Toggle, Scroll 등을 사용했습니다.
Depth관리는 UI에서 Panel을 통해 했습니다.

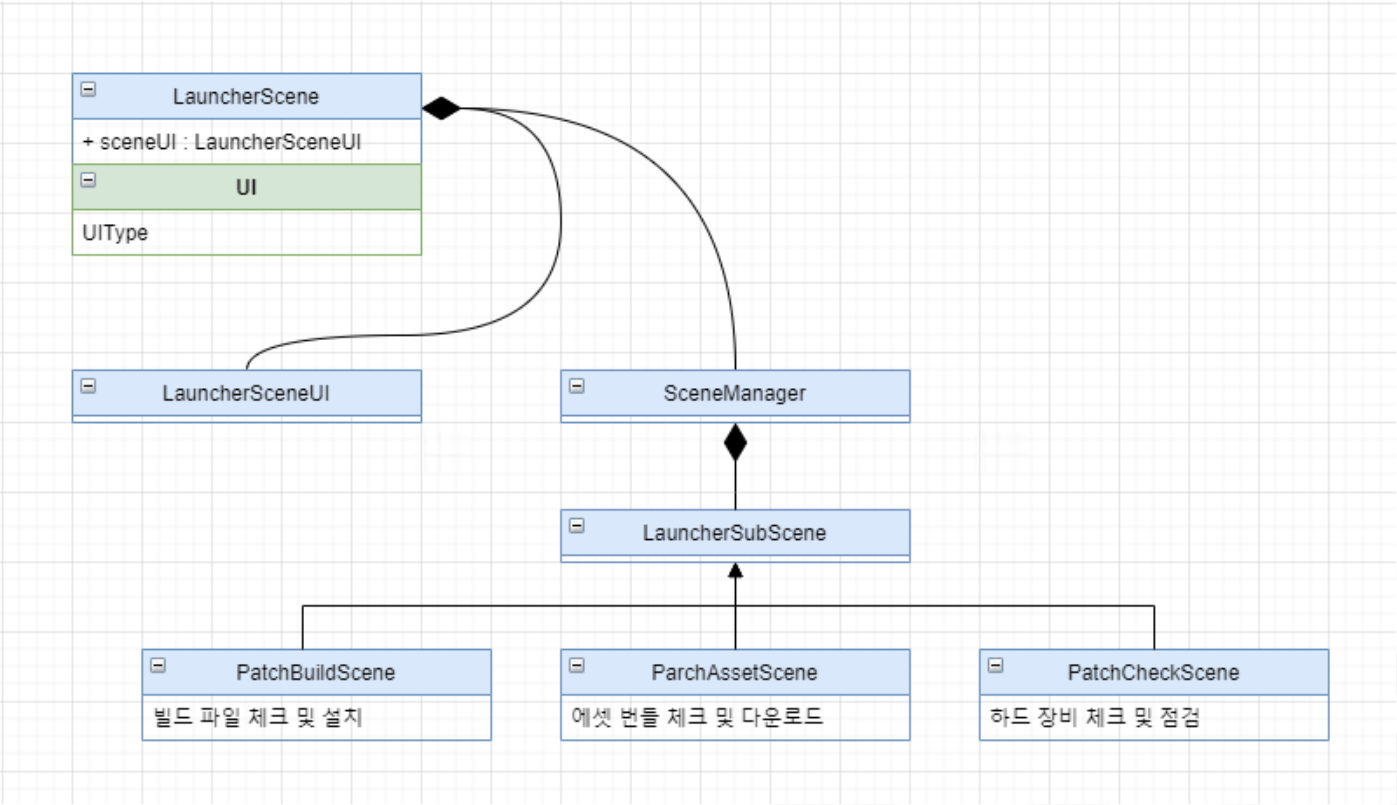




2. 런처

- 사용 목적 : 매장에서 런처를 통해 버전에 따른 실행파일, 리소스를 받고 자체 점검을 가능하게 하기위해 개발했습니다..

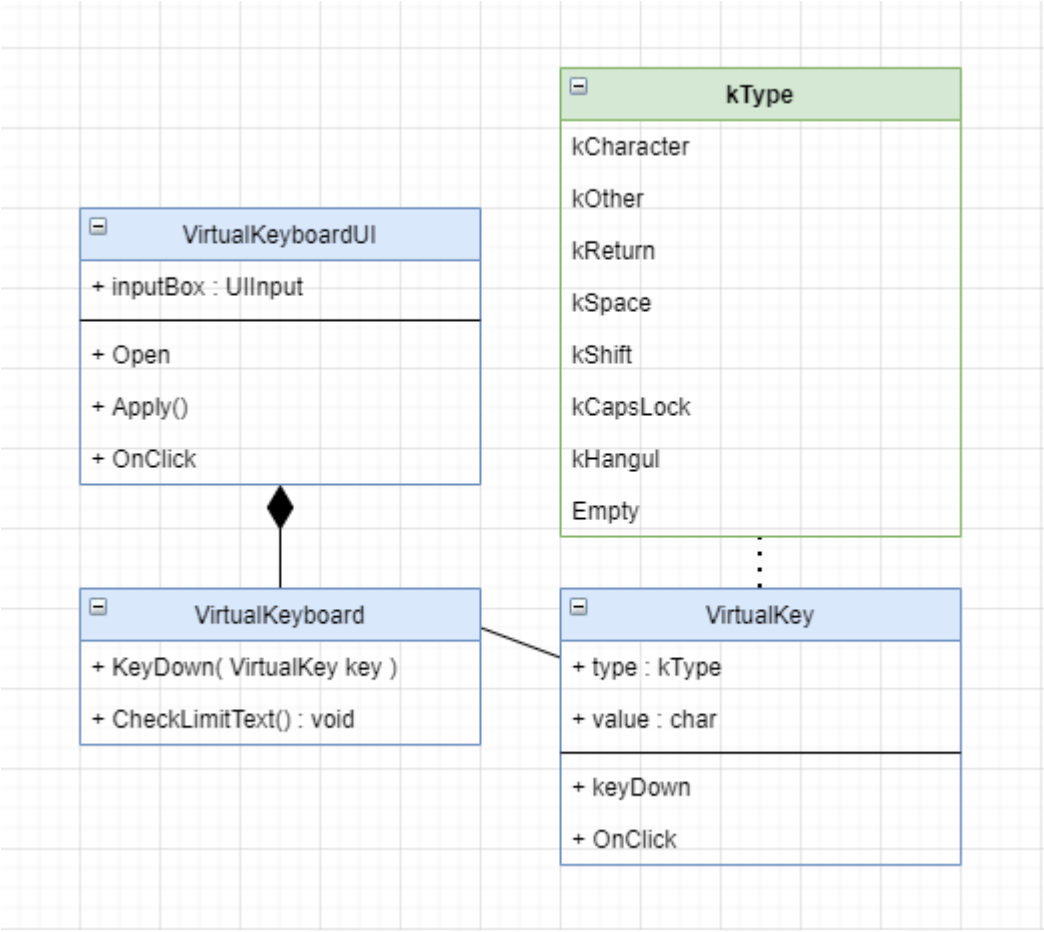
- 콘텐츠 설명 : 빌드 파일, 리소스, 장비 체크를 하는 단계를 만들어 해당 씬 에서 각각의 데이터를 체크하고 다운로드가 필요하면 FTP에 올라가 있는 파일들을 다운로드합니다.



2. 가상 키보드

- 사용 목적 : 키오스크를 통해서 원하는 텍스트, 숫자를 입력을 하기 위해 개발했습니다.

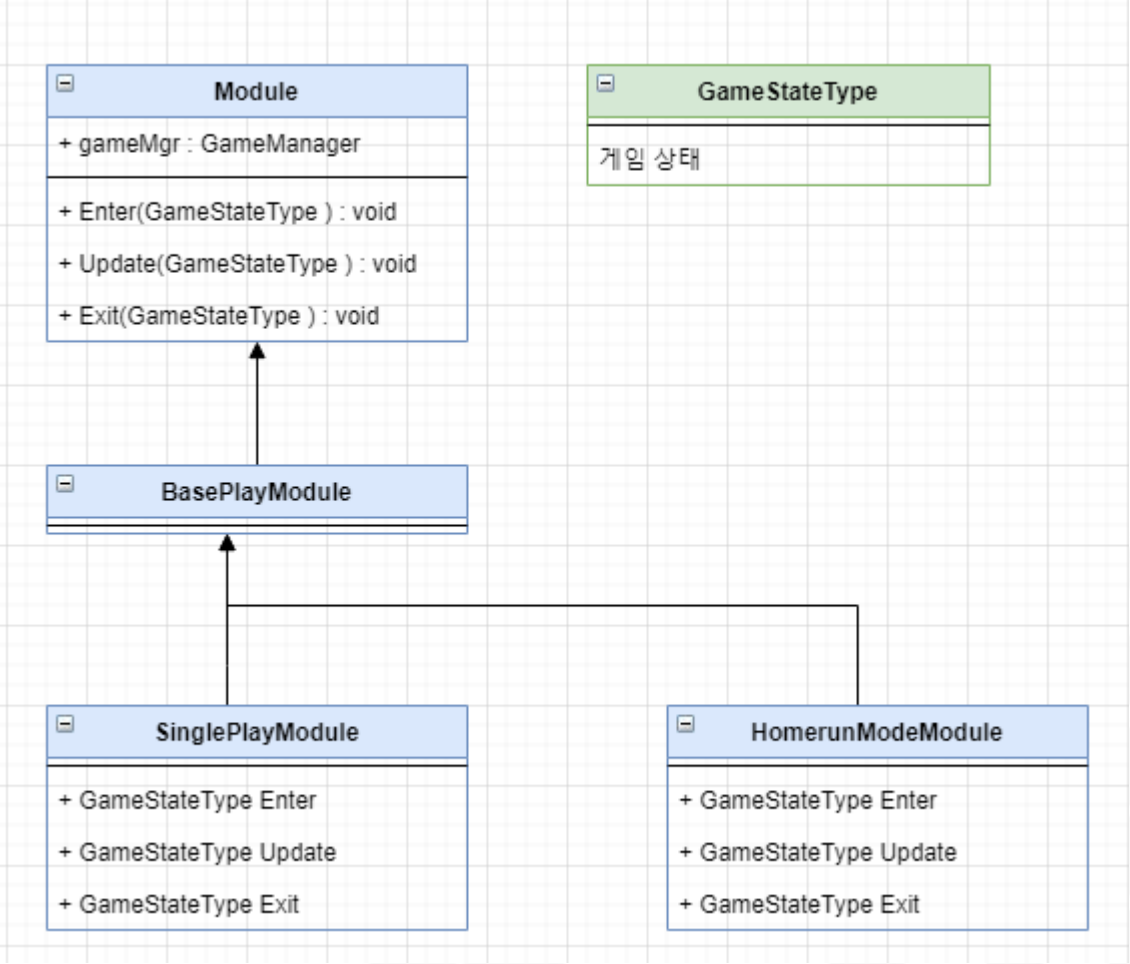
- 콘텐츠 설명 : VirtualKeyBoardUI를 만들어 해당 키를 클릭하면 이벤트를 통해서 해당 키의 이벤트가 발생하여 키입력이 되도록 했습니다.

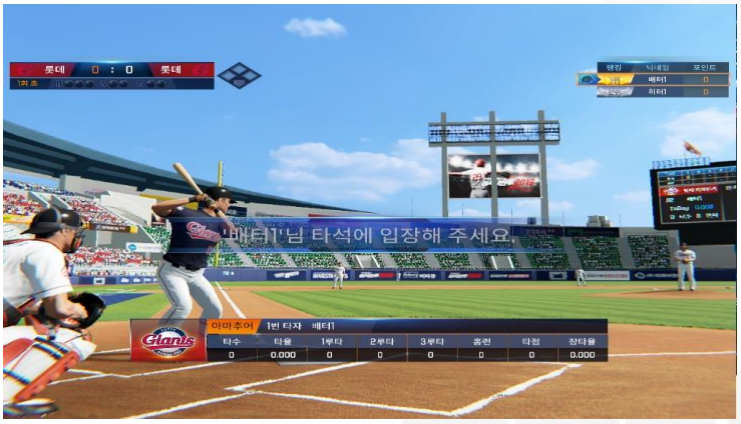


3. 게임 모드(1 vs 1, 팀 대전, 홈런 모드)

- 사용 목적 : 스크린 야구를 통해서 진행 할 수 있는 게임 모드를 플레이 할 수 있는 것을 목표로 개발했습니다.

- 콘텐츠 설명 : 인 게임 모듈을 만들어 BasePlayModule에서는 기본적으로 하는 작업들을 작업을 하며 각 모드마다 필요한 상태에 따라 작업을 하도록 했습니다.

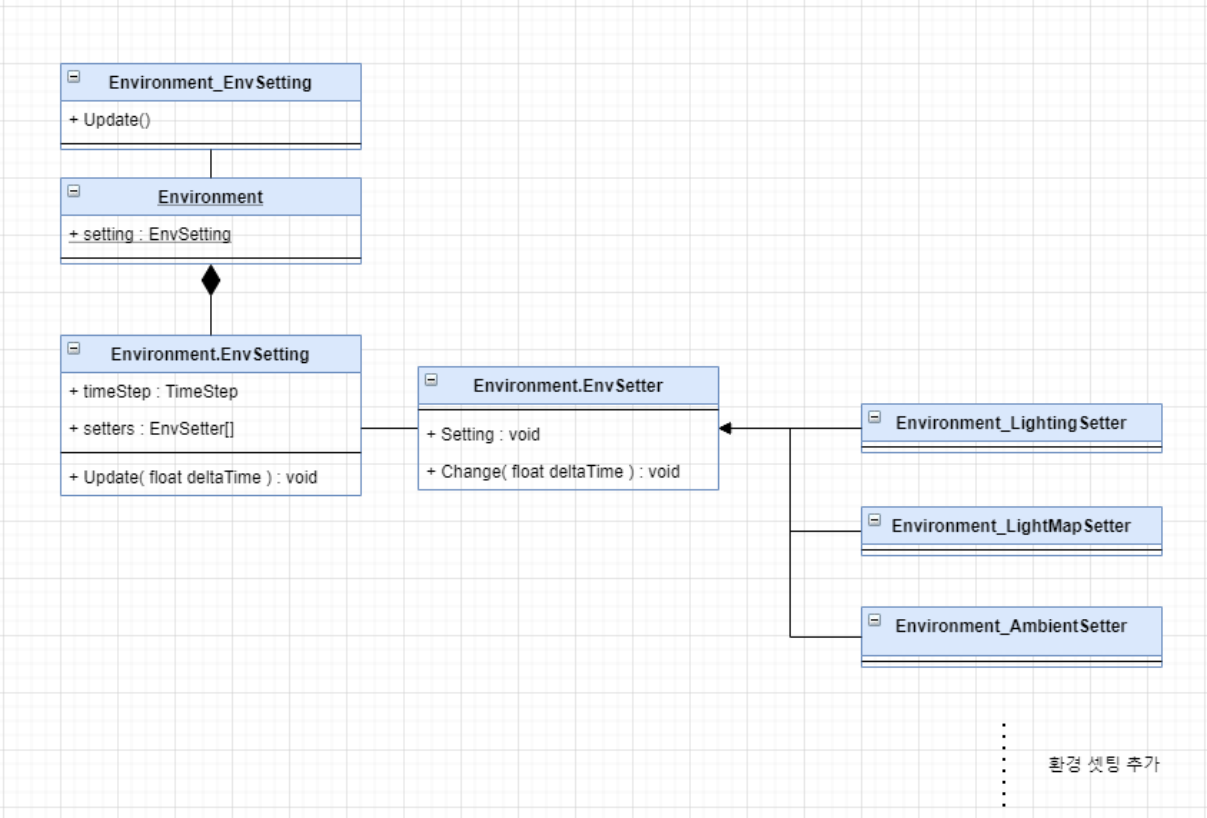


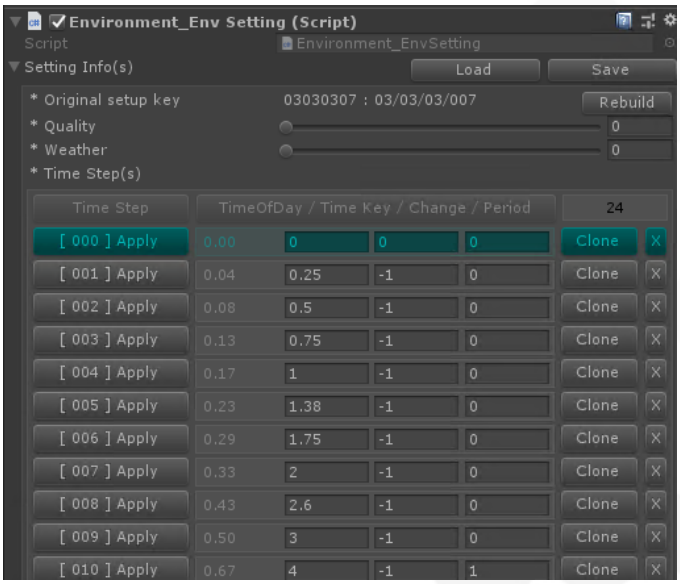


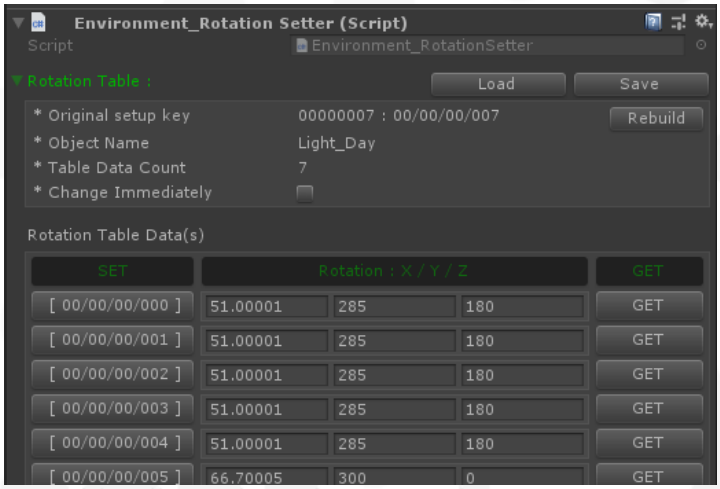
4. 그래픽 환경 툴

- 사용 목적 : 시간에 따라 그래픽 환경을 설정할 수 있는 툴을 목표로 개발했습니다.

- 콘텐츠 설명 : 유니티에서 제공하는 커스텀 에디터를 사용해 시간의 흐름에 따라 Light, Shadow, Ambient, Emission, Bloom 등이 원하는 정보로 게임 환경을 구성하도록 했습니다.



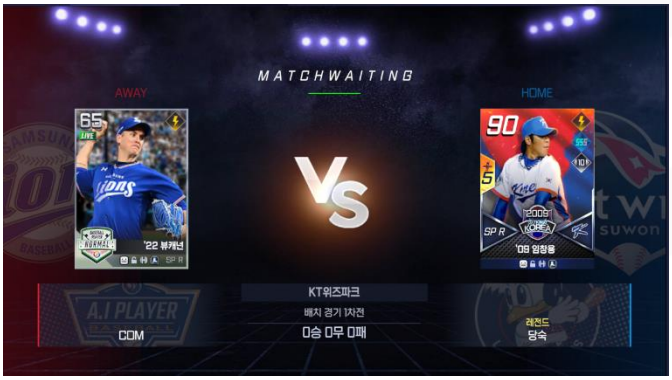




이사만루3

1. 게임 소개

- 게임 : 이사만루3
- 개발 환경
    - C# 기반의 Unity3D, .Net 4.X
    - AOS(Android Studio), IOS(XCode) 플랫폼
    - TortoiseSVN
    - CDN/FTP를 이용한 AssetBundle 관리
    - MySQL 기반의 DB
- 담당 파트 : 이사만루3 콘텐츠 개발 클라이언트
- 담당 업무 : 콘텐츠 개발 및 코드 리펙토링, 유지보수
- 개발 기간 : (2020. 03 ~)



2. 세부 목차

UGUI 13

사운드 다양화 14

상점 및 상품 15

게임 모드(관전 모드, 시뮬레이션 모드, 즉시 완료) 16

룰렛 17

플레이 패스 18

ToolTip 19

팀 컬러 20

프리셋 21

선수강화 및 스탯 22

데이터 뷰어툴 23

에셋번들 24

사운드 25

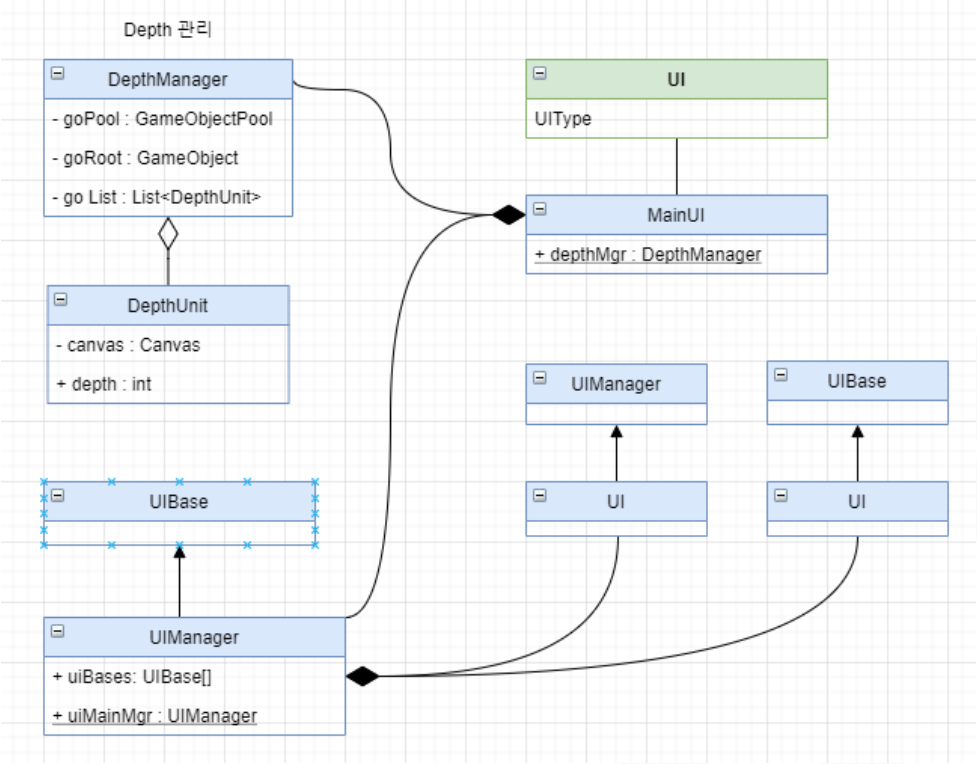
World UI 26

타이머 27

1. UGUI

- 사용 목적 : 유니티에서 지원하는 UI로 게임에서 유저들이 게임 내 정보 등을 볼 수 있도록 개발했습니다.

- 콘텐츠 설명 : MainUI에서 UIManager를 통해 UI를 관리하며 UI를 생성하면 Depth를 관리하는 DepthManager를 통해 셋팅 하려고 하는 DepthUnit을 만들어서 Depth 관리를 했습니다.

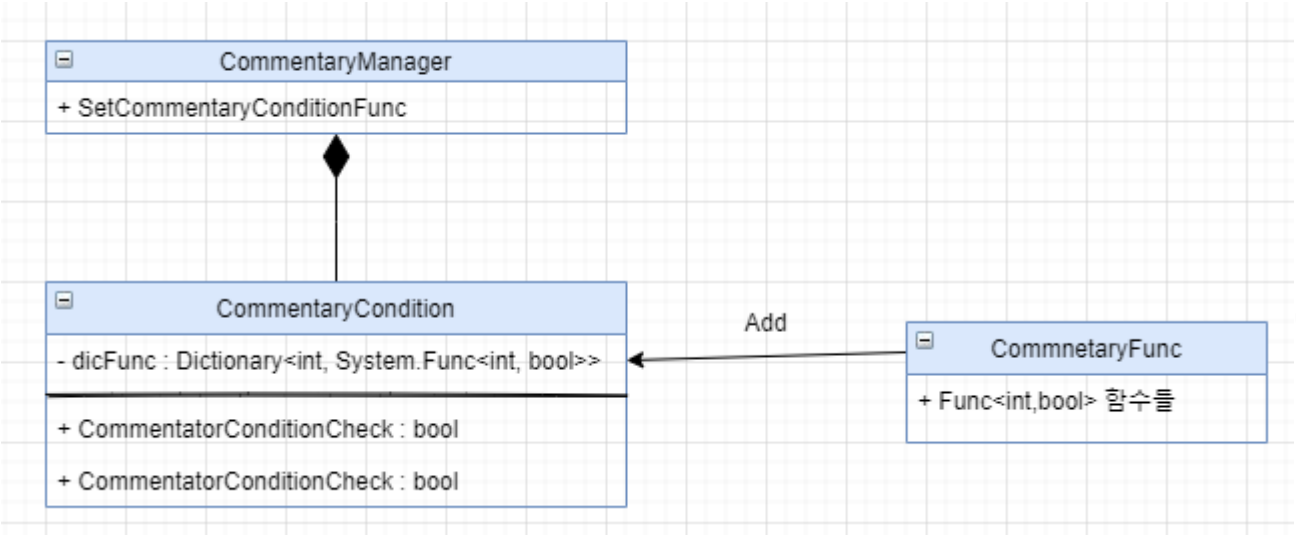


2. 사운드 다양화

- 사용 목적 : 사운드에서 상황별로 체크를 해서 사운드가 나오도록 개발했습니다.

.

- 콘텐츠 설명 : 야구에서 사용되는 코멘터리를 상황에 따라 타입을 만들어 해당 기능들을 Dictionary에 저장을 했습니다. 그리고 사운드 테이블에서 해당 타입을 가진 데이터를 셋팅 하면 현재 상황에서 조건을 체크하여 사운드가 재생이 되도록 했습니다.




```
public virtual void SetCommentaryConditionFunc()
{
    condition.SetConditionFunc( (int)CommentatorFuncType.Random,            CommentaryFunc.ConditionRandom );
    condition.SetConditionFunc( (int)CommentatorFuncType.MatchType,         CommentaryFunc.ConditionMatchType );
    condition.SetConditionFunc( (int)CommentatorFuncType.PreInningScored,   CommentaryFunc.ConditionPreInningScored );
    condition.SetConditionFunc( (int)CommentatorFuncType.OverTime,          CommentaryFunc.ConditionOverTime );
    condition.SetConditionFunc( (int)CommentatorFuncType.GameResult,        CommentaryFunc.ConditionGameResult );
    condition.SetConditionFunc( (int)CommentatorFuncType.OutCount,          CommentaryFunc.ConditionOutCount );
    condition.SetConditionFunc( (int)CommentatorFuncType.Runner,            CommentaryFunc.ConditionRunner );
```

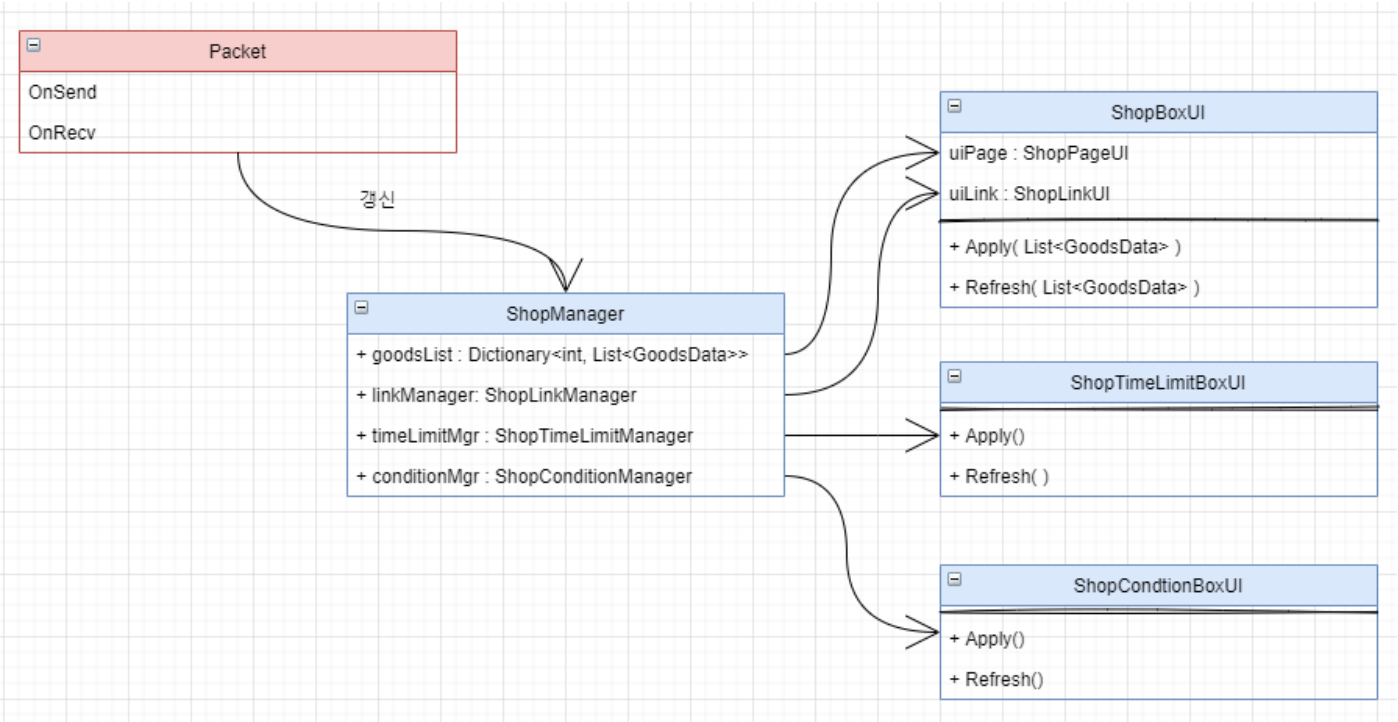
```
public partial class CommentaryFunc
{
    static public bool ConditionRandom( int param )
    {
        return MatchUtility.CheckRandomPer100( param );
    }
    static private bool RoundScore( int a, int b )...
    static public bool ConditionMatchType( int param )...
    static public bool ConditionStadium( int param )...
    static public bool ConditionPreInningScored( int param )...
    static public bool ConditionOverTime( int param )...
    static public bool ConditionGameResult( int param )...
    static public bool ConditionOutCount( int param )...
    static public bool ConditionRunner( int param )...
    static public bool ConditionBatterAdmission( int param )...
    static public bool ConditionThreeBA_OrMore( int param )...
```

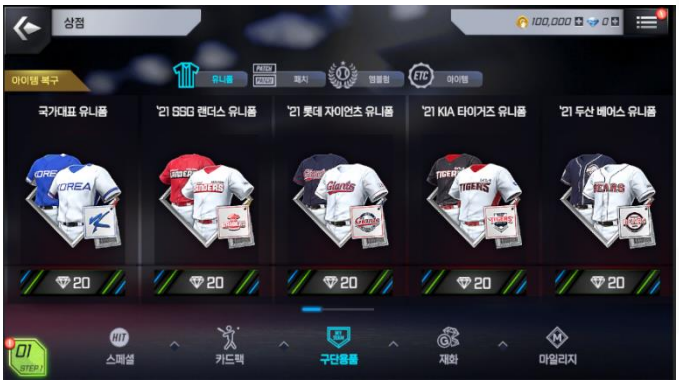
3. 상점 및 상품

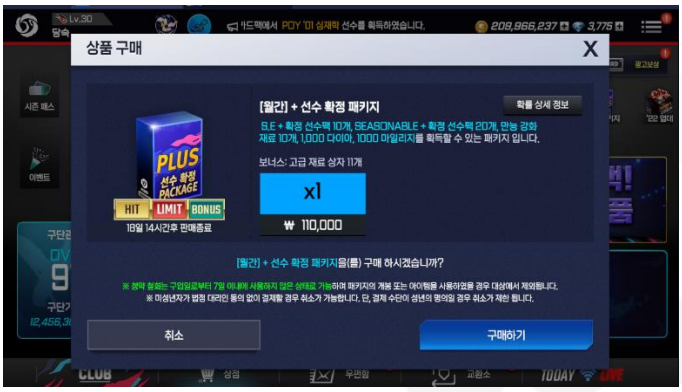
- 사용 목적 : 게임에서 게임재화나 결제를 통해 게임 아이템이나 카드팩을 구매 할 수 있도록 개발했습니다.

.

- 콘텐츠 설명 : 패킷을 통해 상점 및 상품 데이터 정보를 갱신하며 해당 정보들을 UI에서 표현했습니다..



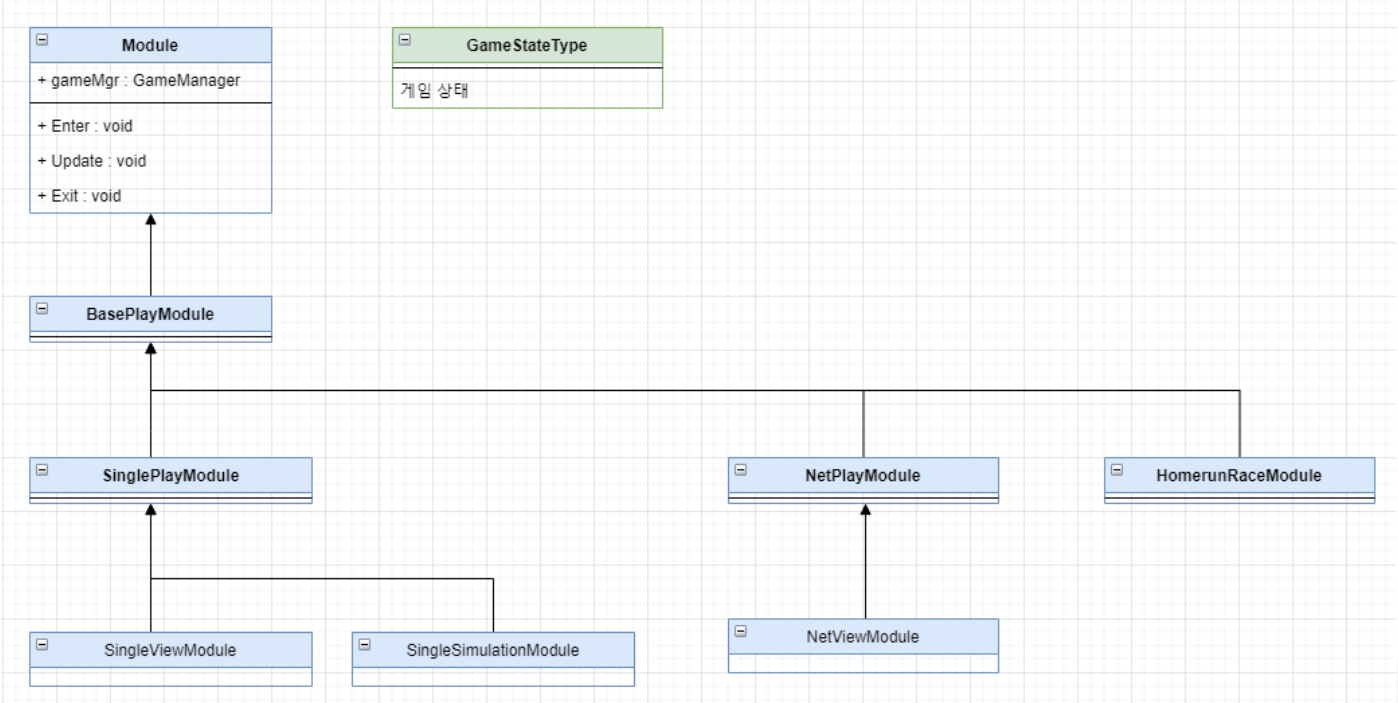




4. 게임 모드(관전 모드, 시뮬레이션 모드, 즉시 완료)

- 사용 목적 : 이사만루3에서 야구로 플레이 할 수 있는 게임 모드를 개발했습니다.

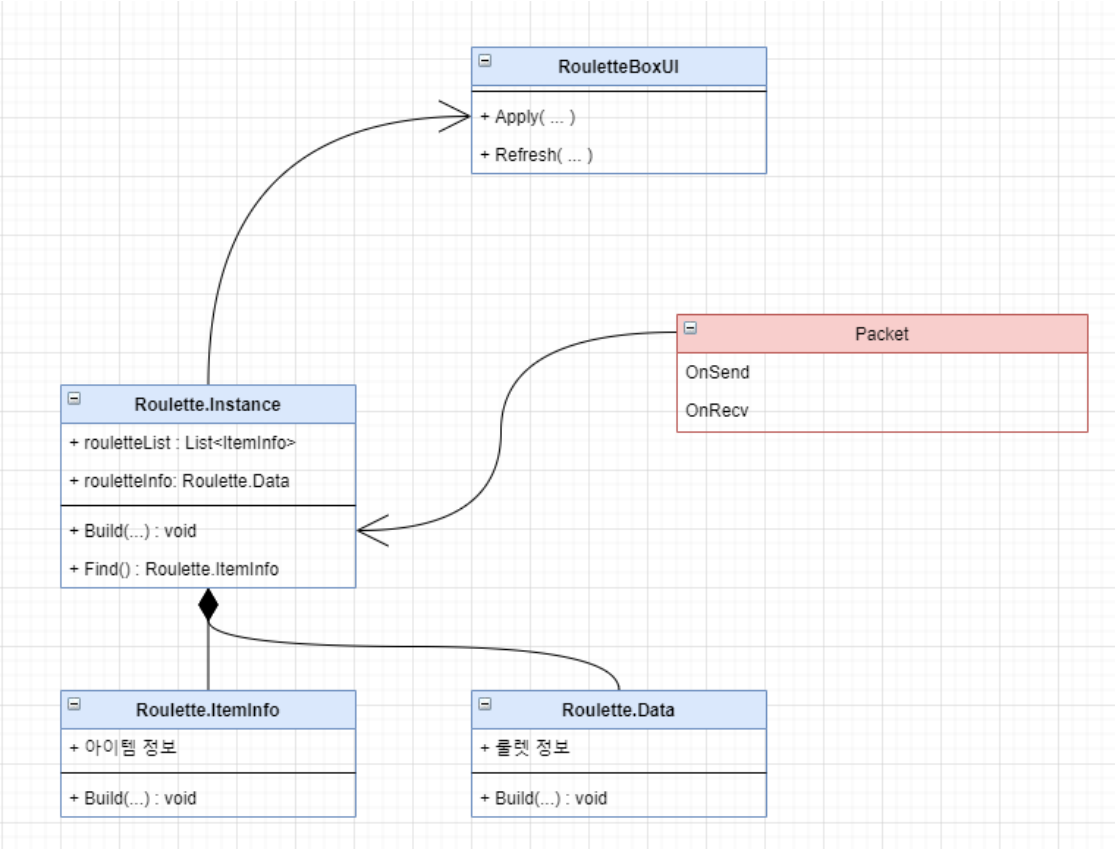
- 콘텐츠 설명 : 인 게임 모듈을 만들어 BasePlayModule에서는 기본적으로 하는 작업들을 작업을 하며 각 모드마다 필요한 상태에 따라 작업을 하도록 했습니다. 즉시 완료의 경우에는 팀의 능력치로 보정을 해서 게임 결과가 나오도록 개발했습니다.



5. 룰렛

- 사용 목적 : 게임 내 콘텐츠로 재화를 소모하여 아이템을 얻을 수 있는 콘텐츠를 개발했습니다.

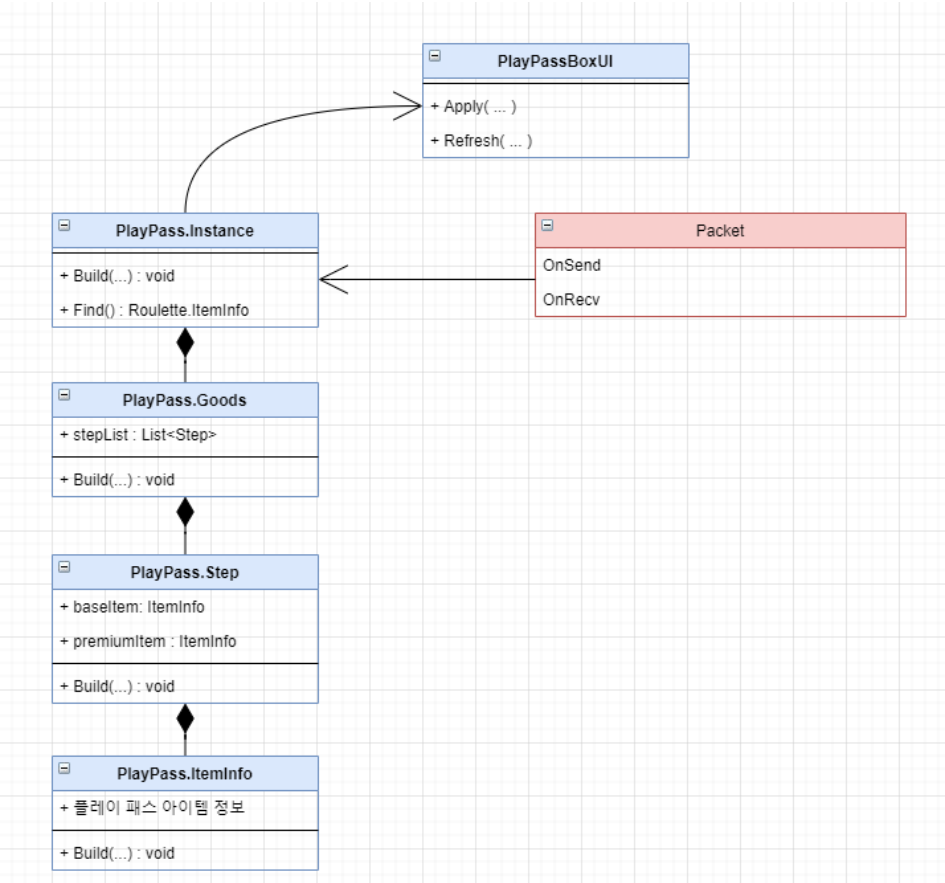
- 콘텐츠 설명 : 패킷을 통해서 룰렛의 정보들을 Roulette.Instance에서 생성하고 해당 정보를 RouletteBoxUI에서 해당 정보들을 표현했습니다.



6. 플레이 패스

- 사용 목적 : 게임 내 콘텐츠로 재화를 소모하여 아이템을 얻을 수 있는 콘텐츠를 개발했습니다.

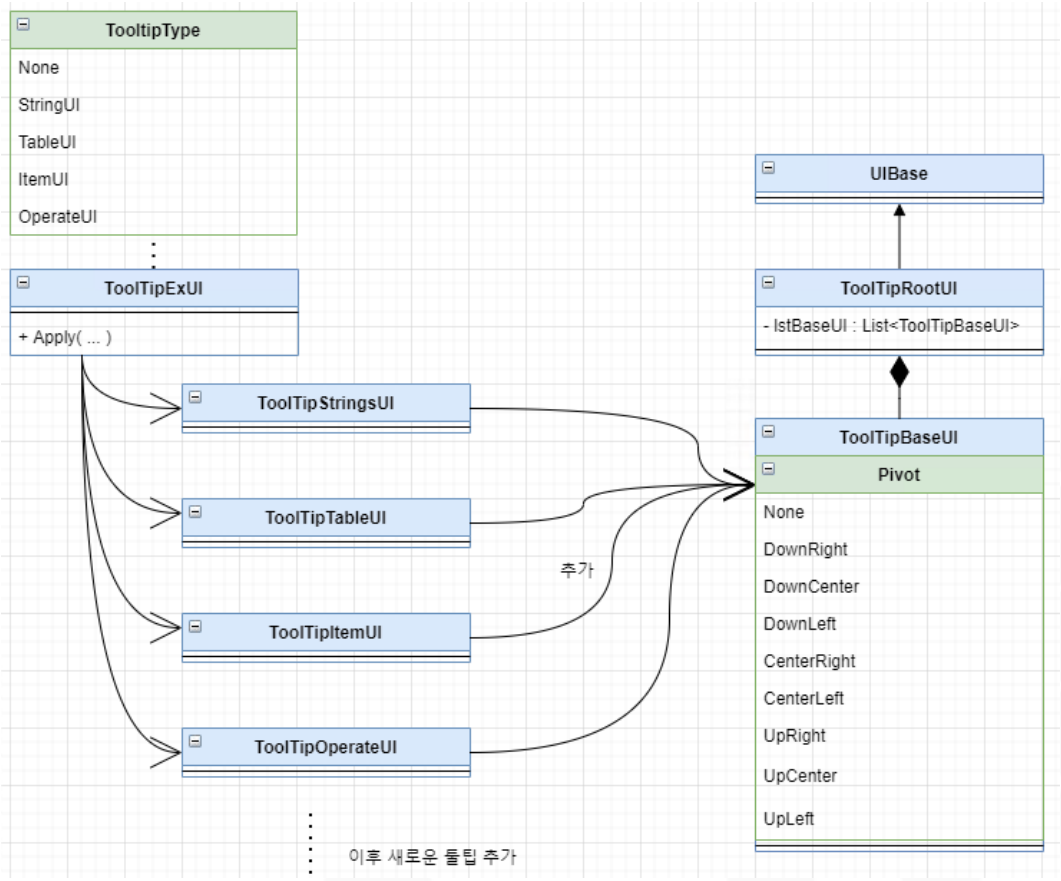
- 콘텐츠 설명 : 패킷을 통해서 플레이패스 정보들을 PlayPass.Instance에서 생성하고 해당 정보를 PlayPassBoxUI에서 해당 정보들을 표현했습니다.

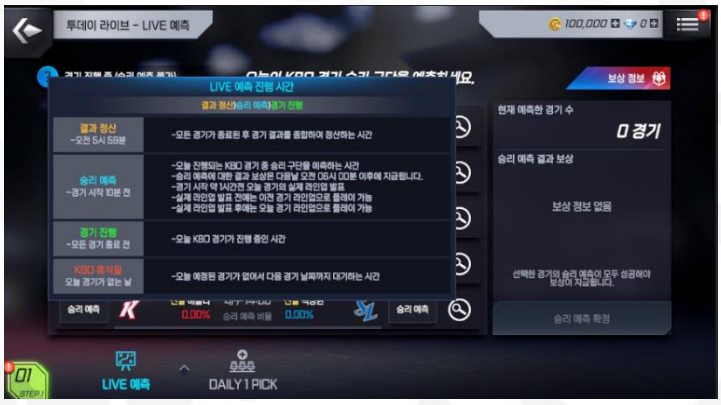


7. ToolTip

- 사용 목적 : 게임에서 사용되는 ToolTip을 다양하게 표현을 할 수 있도록 컴포넌트화 했습니다.

- 콘텐츠 설명 : ToolTipExUI 컴포넌트를 사용하면 원하는 타입의 ToolTip을 만들어 상단의 Depth에 있는 ToolTipRootUI에 ToolTipBaseUI를 생성하여 ToolTip을 원하는 Pivot으로 셋팅 하여 생성하도록 했습니다.

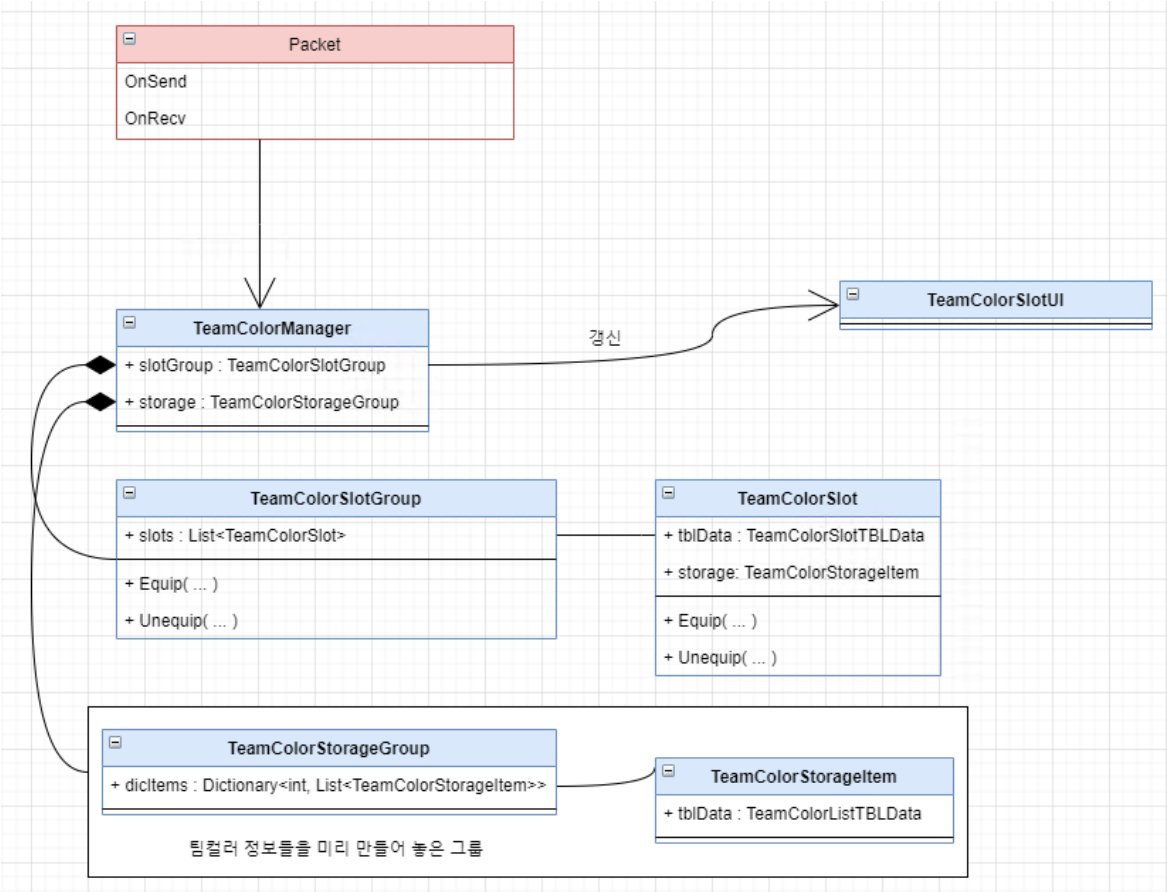




8. 팀 컬러

- 사용 목적 : 선수들을 모으면 조건에 따라 장착을 하고 스탯을 올려주는 콘텐츠를 개발했습니다.

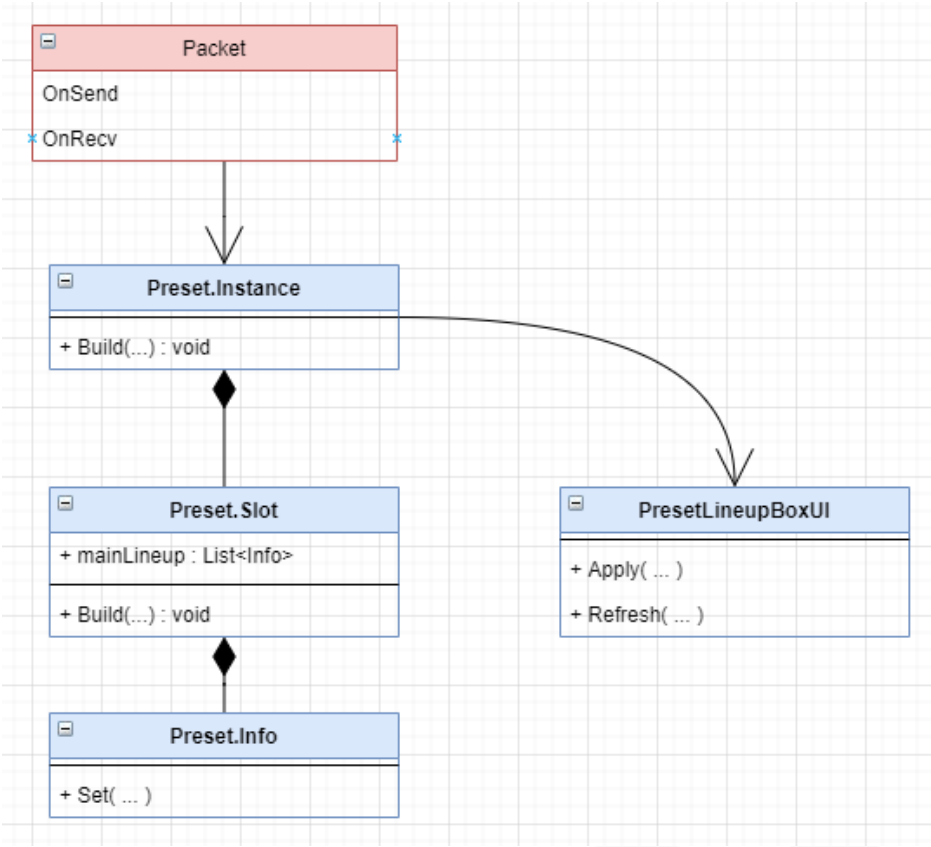
- 콘텐츠 설명 : TeamColorManager에서 TeamColorSlotGroup에서는 장착하는 슬롯을 관리하며 TeamColorStorageGroup에서는 TeamColor의 장착 가능여부를 관리하기 위해 테이블을 통해 생성했습니다. 패킷으로 자신의 팀 컬러 정보를 셋팅하면 UI에서 표현하도록 했습니다.

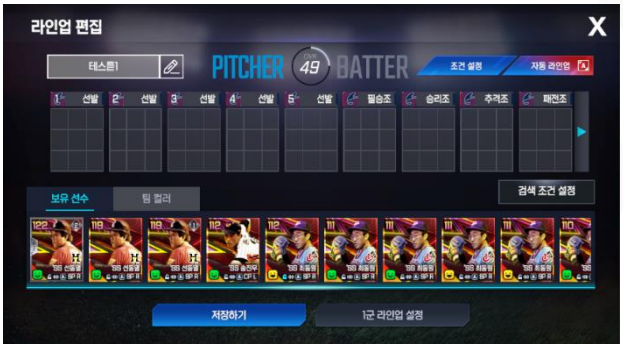


9. 프리셋

- 사용 목적 : 라인업과 팀컬러를 미리 저장하고 불러올 수 있도록 개발했습니다.

- 콘텐츠 설명 : 패킷을 통해 Preset.Instance에서 현재 가지고 있는 프리셋 정보를 빌드를 하며 해당 정보들로 PresetLineupBoxUI에서 표현하도록 했습니다.

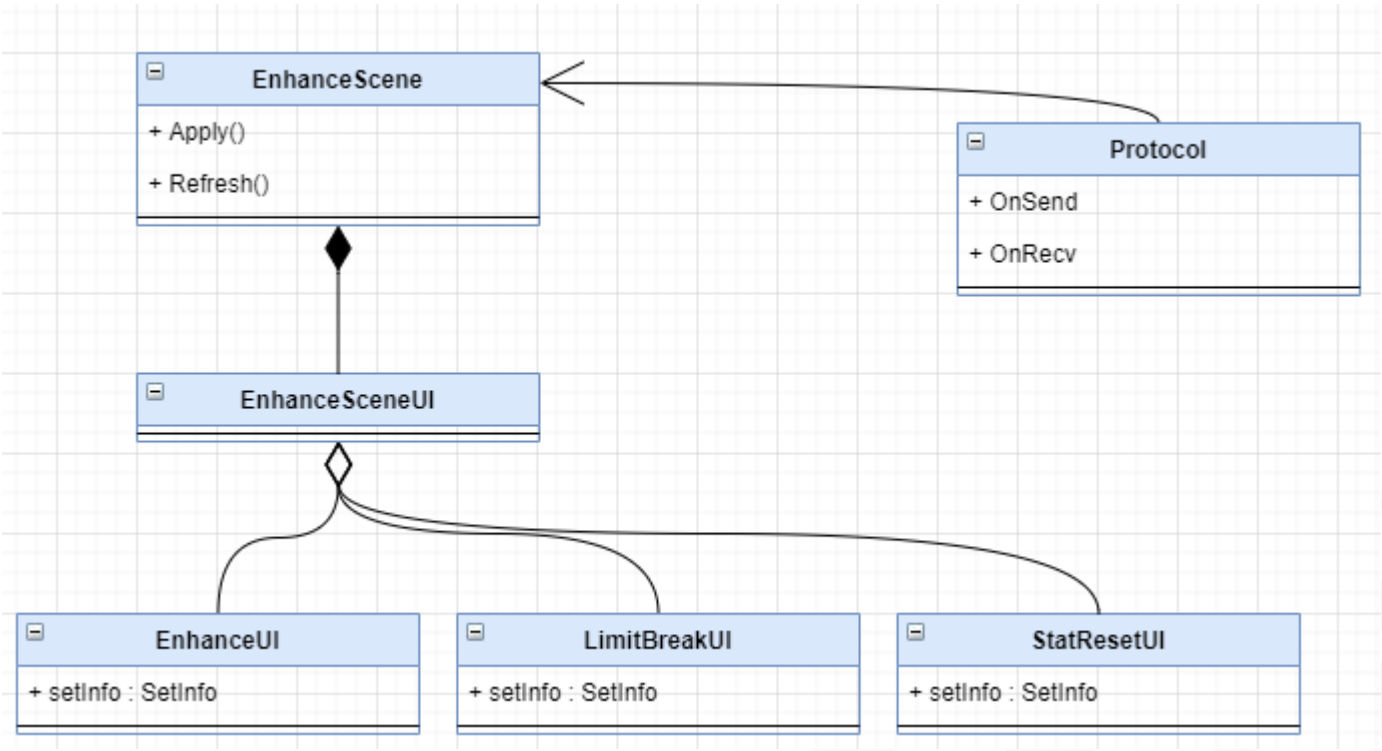




10. 선수 강화 및 스탯

- 사용 목적 : 선수를 강화 시키고 추가 스탯을 받아서 선수들의 스탯을 올리고 올렸던 스탯들을 초기화 가능하도록 개발했습니다.

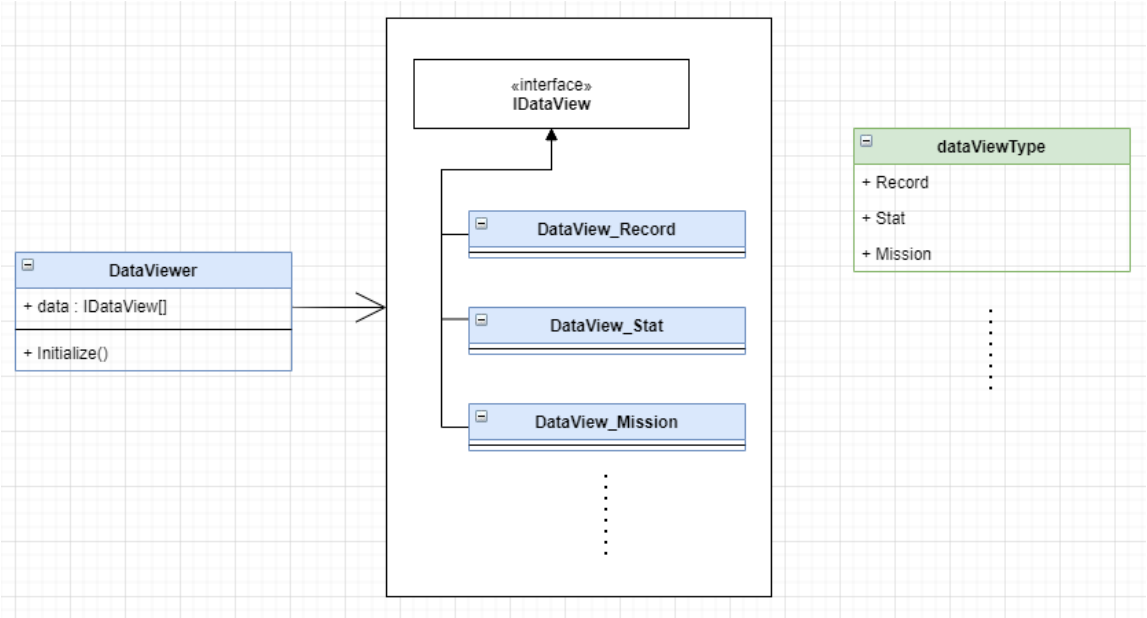
- 콘텐츠 설명 : 강화 Scene이 있으며 SceneUI안에 SubSceneUI들로 구성했습니다. 각 SceneUI들은 SetInfo들을 가지고 있어 Scene을 구성할 때 필요한 설정이나 데이터를 가지고 있습니다. 해당 정보들로 각각의 UI들의 Active가 켜질 때 UI가 갱신되도록 했습니다.

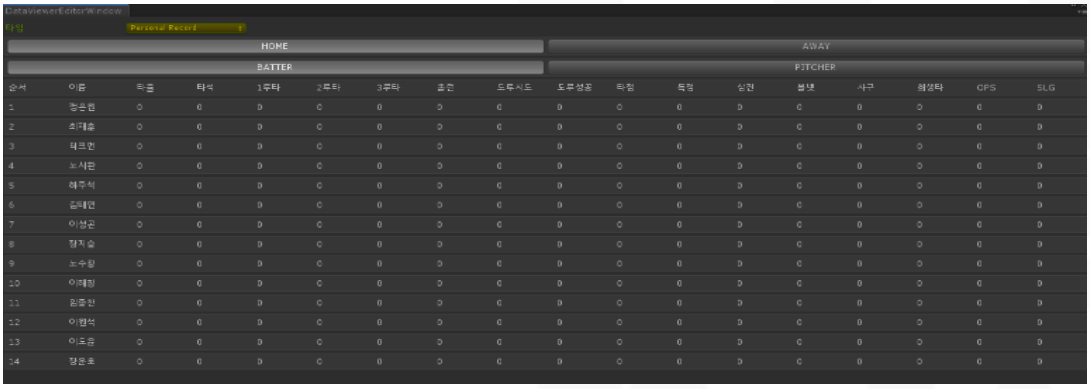


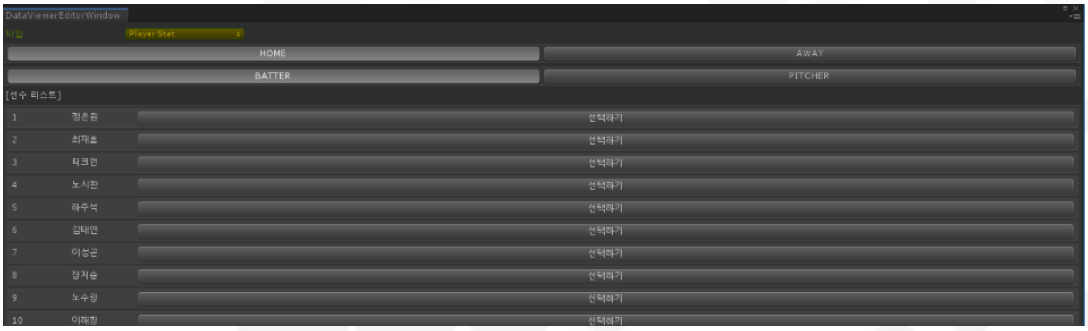
11. 데이터 뷰어툴

- 사용 목적 : 게임 정보들을 툴을 통해 상세히 볼 수 있도록 개발했습니다.

- 콘텐츠 설명 : CustomEditor를 사용해 DataVeiwer를 만들었으며 그 안에 IDataView를 상속받는 각각의 Viewer클래스를 그리도록 했습니다.



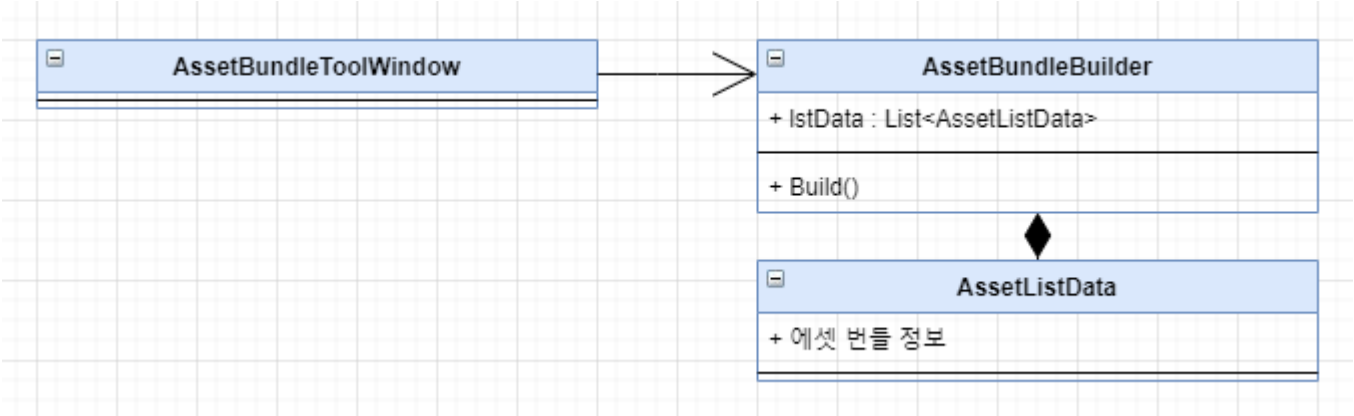


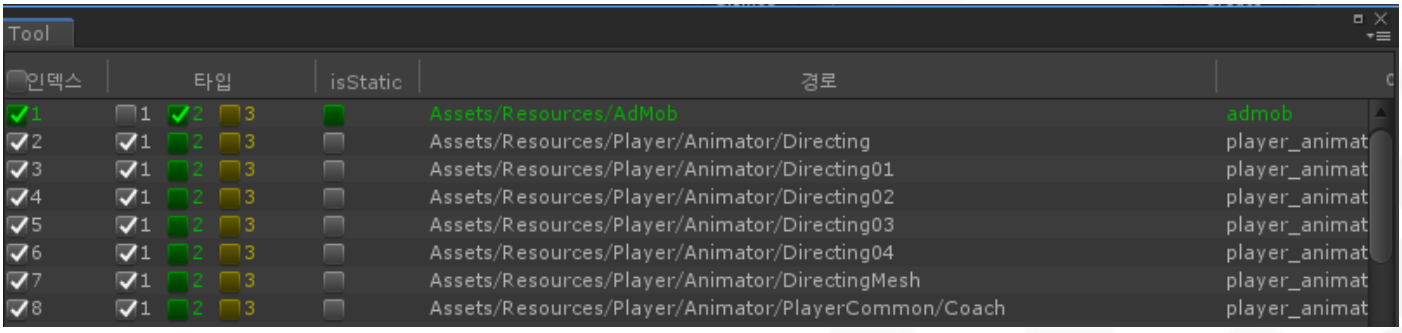


12. 에셋 번들

- 사용 목적 : 지정한 폴더내에 있는 리소스 파일들을 에셋번들로 변경하고 해당 에셋번들을을 FTP/CDN에서 다운로드하여 캐싱하도록 개발했습니다.

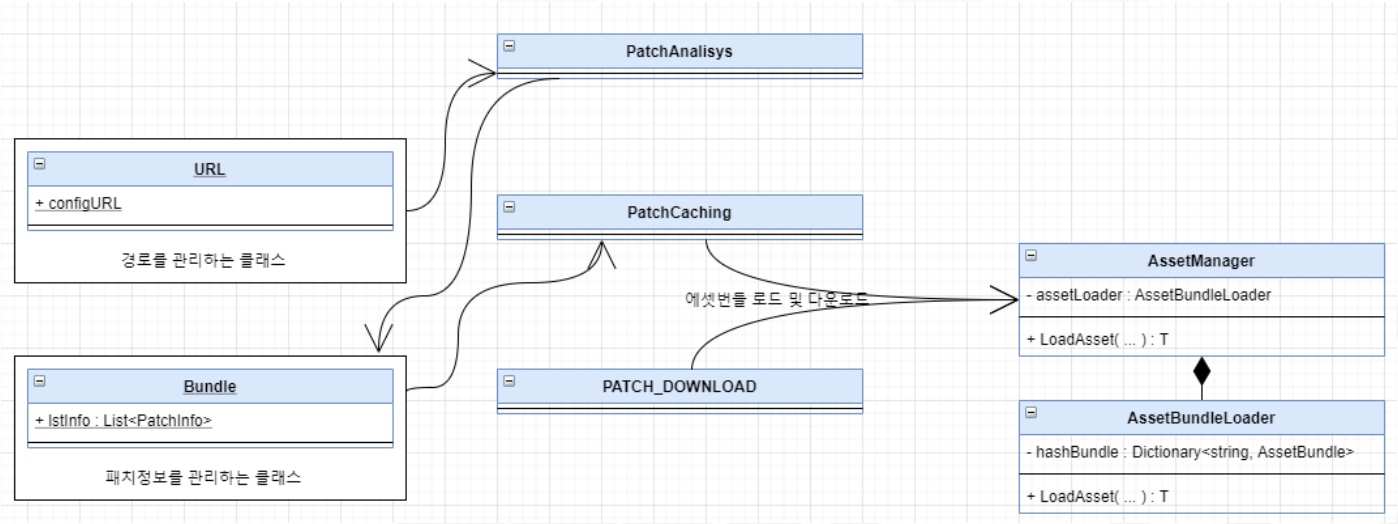
- 콘텐츠 설명 : 에셋 번들을 원하는 경로에 있는 리소스들을 AssetBundleBuilder에 AssetListData로 정보를 저장해두고 Build이벤트를 통해 빌드를 진행했습니다. 그 정보들을 로그인 시에 현재 에셋 번들을 분석하고 이후 FTP/CDN에 있는 에셋 번들을 다운로드합니다.





Tool

인덱스	타입	isStatic	경로	
1	1 2 3		Assets/Resources/AdMob	admob
2	1 2 3		Assets/Resources/Player/Animator/Directing	player_animat
3	1 2 3		Assets/Resources/Player/Animator/Directing01	player_animat
4	1 2 3		Assets/Resources/Player/Animator/Directing02	player_animat
5	1 2 3		Assets/Resources/Player/Animator/Directing03	player_animat
6	1 2 3		Assets/Resources/Player/Animator/Directing04	player_animat
7	1 2 3		Assets/Resources/Player/Animator/DirectingMesh	player_animat
8	1 2 3		Assets/Resources/Player/Animator/PlayerCommon/Coach	player_animat

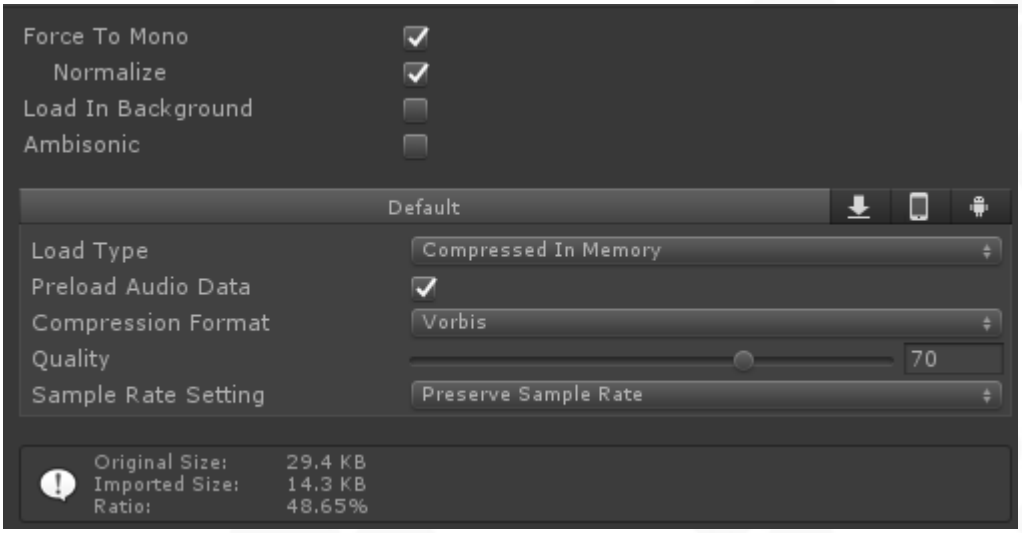


<로드과정>

13. 사운드

- 사용 목적 : 사운드 파일의 메모리 적재 및 최적화되도록 설정했습니다..

- 콘텐츠 설명 : Force To Mono의 경우에는 멀티 채널(Stereo)을 사용하지 않고 싱글 채널(Mono)로 강제로 사용하도록 했습니다. 유니티에서는 기본적으로 싱글 채널 기본이고 만약 사운드 파일에서 좌우 사운드가 나눠져 있지 않는 이상 Mono를 사용하여 최적화를 했습니다.
  Load Type은 파일을 압축한 채로 메모리에 적재할지에 대한 값입니다. Decompress On Load의 경우에는 압축하지 않고 메모리에 적재를 해서 CPU연산이 적지만 사이즈가 크다 보니 작은 용량의 사운드 파일에 설정했으며 Compressed In Memory의 경우에는 반대로 사이즈는 적지만 CPU연산이 필요하며 자주 사용되지 않는 사운드 파일에 설정했습니다.
  Compression Format은 압축형식을 설정하는 것으로 주로 사용한 방식은 PCM과 Vorbis 방식입니다.



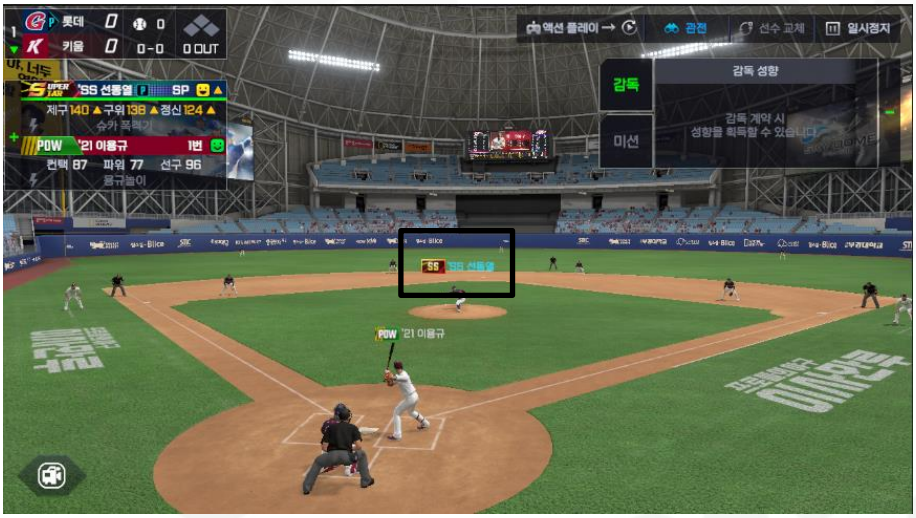
14. World UI

- 사용 목적 : 게임 내 World에 있는 오브젝트의 좌표를 가지고 UI 스크린 좌표에 표현되도록 개발했습니다.

- 콘텐츠 설명 : 원하는 오브젝트의 월드 좌표가 카메라 안에 있는지 체크하고 해당 좌표를 스크린 좌표로 변환하여 UI의 좌표에 셋팅 하도록 했습니다.

```
Vector3 position = target.playerObject.GetNameCardPosition();

if( false == Match.Instance.gameMgr.gameCamera.isActiveName
    || Match.GameUtil.IsInGameCamera( position ) )
{
    SetActive( false );
    return;
}

SetActive( true );

Vector3 uiPosition = Match.GameUtil.GameWorldToScreen( position );

obj.transform.position = uiPosition;
```

```
static public Vector2 GameWorldToScreen( Vector3 position )
{
    if( gameCam == null )
        return Vector2.zero;

    return gameCam.WorldToScreenPoint( position );
}
```



15. 타이머

- 사용 목적 : 게임 내에서 시간을 체크하는 경우 사용할 수 있도록 개발했습니다.

- 콘텐츠 설명 : 타이머 구조체를 만들어 남은 시간과 끝나는 시간을 넣으면
오늘 날짜로부터 남은 시간과 끝나는 시간을 체크하도록 했습니다.

```
public struct ExpireTimer
{
    private DateTime    _expireTime;
    public DateTime     expireTime      { get { return _expireTime; } }
    public TimeSpan     remainTime      { get { return ( _expireTime - DateTime.Now ); } }
    public bool         isExpire        { get { return ( _expireTime <= DateTime.Now ) ? true : false; } }

    public int          days            { get { return ( isExpire == true ) ? 0 :remainTime.Days; } }
    public int          hours           { get { return ( isExpire == true ) ? 0 :remainTime.Hours; } }
    public int          minutes         { get { return ( isExpire == true ) ? 0 :remainTime.Minutes; } }
    public int          seconds         { get { return ( isExpire == true ) ? 0 :remainTime.Seconds; } }
    public double       totalDays       { get { return remainTime.TotalDays; } }
    public double       totalHours      { get { return remainTime.TotalHours; } }
    public double       totalMinutes    { get { return remainTime.TotalMinutes; } }
    public double       totalSeconds    { get { return remainTime.TotalSeconds; } }

    public ExpireTimer( int seconds )   { _expireTime = System.DateTime.Now + new TimeSpan( 0, 0, seconds ); }
    public ExpireTimer( float seconds ) { _expireTime = System.DateTime.Now + new TimeSpan( (long)( seconds * 10000000 ) ); }
    public ExpireTimer( TimeSpan span ) { _expireTime = System.DateTime.Now + span; }
    public ExpireTimer( double endDate ) { _expireTime = System.DateTime.FromOADate( endDate ); }
```

```
public class CustomTimer
{
    protected   bool    active      { set; get; }
    protected   float   time        { set; get; }

    public CustomTimer()...

    ~CustomTimer()...

    public bool Active...

    public float Time...

    public void AddDeltaTime()
    {
        time += UnityEngine.Time.deltaTime;
    }
}
```

감사합니다.

이름 : 권오현

번호 : 010.2719.5330

이메일 : ohhyun5442@gmail.com