[IR] 클라이언트 코드 컨벤션 [v. 0.2]

# 데이터 타입

기본형 데이터 타입, 자료구조는 UE4에서 정의해 놓은 타입을 쓴다. 크로스 플랫폼 이식에 용이. Platform.h에 정의 되어 있다.

STL은 되도록 쓰지 않는다. 서버와 연계하는 부분 제외.

# 명명 규칙 (Naming)

기본적으로 UE4 명명 규칙을 따른다

## Pascal Case & Upper Case

첫 글자는 대문자로 시작하며, 대문자로 단어를 구분한다.

예) GuildMark, EntityView 등

* 파일명
* 클래스명
  + 명사형으로 정의
* 함수명
  + 동사로 시작. 일부 예외 사항 허용 (컨텐츠 분류, On과 같은 전치사로 시작)
  + 매개변수가 변경되지 않을 경우 const를 명시한다. 리턴 값도 마찬가지.
  + 매개변수가 사용자 정의 타입일 경우 &(참조)타입으로 선언한다.

예) void SetUnit(const AUnit& Unit);

* + 오버라이드 한 함수는 override 키워드를 명시한다.
* 타입
  + Enum은 enum class를 사용
* 변수명
  + 헝가리식 표기법 금지, 단 bool형은 앞에 b를 명시 예) bLoop
  + 컨테이너(Tarray, TMap 등)는 이름 끝에 s를 붙인다.

예) Tarray<FString> MemberNames;

TMap<FName, UPaperSprite\*> SpriteNames;

## Namespace

일단 쓰지 않는 방향으로 결정. 추후에 바뀔 수 있음

# 주석

클래스는 에디터에서 생성할 경우 만들어지는 기본 양식 사용

/\*\*

\* 클래스 내용

\*/

함수는 한 줄 주석 사용. 함수 선언부 위쪽

변수는 뒤에 적는다. 같은 맥락에 속성은 Tab 키를 두 번 눌러 변수끼리 Allign을 맞춘다.

FString GuildName; /// 길드 이름

Image\* GuildMark; /// 길드 마크

# 클래스 레이아웃

함수, 변수 별로 public, protected, private 섹션을 가지며 함수, 변수 각각 선언한다.

함수는 변수 보다 먼저 선언한다.

* 선언 순서

1. Inner Class, Inner Struct
2. typedef
3. 생성자, 소멸자
4. BP에 공개할 변수 및 함수
5. 일반 로직 변수 및 함수

# 헤더 예시

#pragma once

#include “CodeConvention.generated.h”

class Image;

/\*\*

\* 클래스 주석

\*/

class CodeConvention : public Convention

{

public:

Class InnerCovention

{

public:

int32 InnerID;

}

typedef uint8 Flag;

typedef float ConventionDataType;

public:

CodingConvention();

~CodingConvention();

/// 컨벤션 BP 함수

UFUNCTION(BlueprintImplementableEvent, Category = “Convention”)

void OnStartConvention();

UFUNCTION(BlueprintImplementableEvent, Category = “Convention”)

void OnEndConvention();

UPROPERTY(EditAnywhere, Category = “Convention”)

bool bActive; /// 활성화 유무

UPROPERTY(EditAnywhere, Category = “Convention”)

FString ConventionName; /// 컨벤션 이름

public::

void BeginPlay() override;

void Tick() override;

void EndPlay() override;

private:

void CalculateConvertionNumber();

public:

int32 ID;

private:

int NumberOfConventions;

Image\* ConvetionImage;

};

# 구현 레이아웃

* 함수 원형에는 파라미터 이름에 Out 접투어를 붙이고, 함수 호출 구문에는 해당 파라미를 /\*out\*/ 주석으로 명시한다.

예)

\* 함수 원형

Void ExtractStringNumber(const FString& AniName, int32& OutNameNumber); /// 파라미터 이름에 Out 접두어

\* 호출 구문

FString AniName = “Attack”;

Int32 NameNumber;

ExtractStringNumber(AniName, /\*out\*/ NameNumber); /// /\*out\*/ 이라고 주석 명시

* { 는 항상 다음 라인에서 시작.
* if, for 문을 쓸 때, 코드가 한 줄이라도 { }로 묶는다.
* switch 문을 쓸 때는, 두 줄 이상일 경우 { }로 묶는다. 그리고 { } 다음 라인에 break;를 붙인다.
* .cpp에서 .h Include 순서

1. project 헤더
2. 해당 class 헤더
3. UE4 라이브러리 헤더
4. 시스템 헤더
5. 컨텐츠 헤더

팀원들이 알아 보기 쉬운 코드로 작성해 주세요.