**코드 작성 규정집**

코드 작성에 대한 규칙을 정해 정리한 것이다.

단, 기존까지 작성하던 코드는 작업 진행 상황을 고려하여 임시적으로 수정하지 않고 진행하는 것을 허용한다.

**소스코드**

하나의 소스코드의 코드수는 1000줄 이내 규모로 작성한다.

하나의 소스코드 길이는 80자 이내로 작성한다.

함수의 내용은 70줄 이내로 작성한다.

하나의 문장을 2줄 이상으로 작성할 경우, 적절한 길이를 넘은 쉼표(,)를 기준으로 새 줄을 시작한다.

하나의 문장을 2줄 이상으로 작성할 경우, 두 줄의 가독성을 고려하여 정리한다.

거의 모든 파일에서 사용하는 변수나 #include 전처리문, 함수는 “stdafx” 파일에 작성한다.

파일의 #include 전처리문 위치는 #include “stdafx.h” 보다 아래에 위치한다.

**작명 규칙**

명칭은 31자 이내로 정한다.

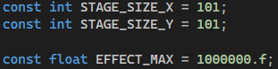
변수 명

변수 명은 camelCase(카멜 케이스) 네이밍 룰로 진행한다.

(첫 단어는 소문자로 표기, 이후 다른 단어의 첫 글자는 대문자로 표기)



매크로와 상수는 대문자와 언더스코어(\_)로 작성한다.



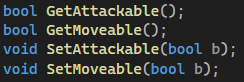
Enum class와 그 자료형의 상수들은 “E\_”를 붙인다.



**함수 명**

함수 명과 class는 PascalCase(파스칼 케이스) 네이밍 룰로 진행한다.

(단어의 첫 글자를 대문자로 표기)



함수의 인자 설명이 필요하지 않을 땐 자료형의 첫 글자를 따서 작성한다.



함수의 인자가 단순하지 않아 설명이 필요할 땐 변수명으로 이해가 되도록 작성한다.



말단 자식 클래스의 가상 함수는 virtual 키워드를 빼고 final을 붙여 사용한다.



**주석 처리**

복잡한 함수, 알기 어려운 함수 인자, 복잡한 자료구조, 복잡한 논리식은 주석을 작성하여 설명을 남긴다.

주석이 필요한 코드엔 원시 코드 바로 윗줄에 주석을 작성한다.

주석을 작성/수정할 땐 변경자, 변경 일자, 변경 내용을 작성한다.

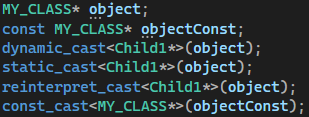
**자료형**

생성자 호출은 특별한 이유가 아니면 Uniform Initialization 문법을 우선으로 한다.



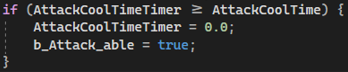
**캐스팅**

모든 캐스팅은 명시적으로 표시한다.

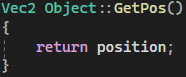


**문법**

if, for, while, switch 등 분기를 나타내는 문장은 중괄호({})를 문장 시작 옆에 바로 작성한다.



함수의 중괄호({})는 문장 시작 아래 줄에 작성한다.



Switch 문엔 break를 꼭 표기하되, 다음 case로 넘어가는 것을 의도하는 경우는 주석 처리로 표기한다.

Switch 문은 항상 default 문을 넣는다.

Goto 문을 사용하지 않는다.

For 문을 제어하는 수식에 실수 값을 사용하지 않는다.

For 문을 제어하는 수치 변수를 계산에 사용하지 않고, 반복 횟수를 목적으로만 사용한다.

For 문에 Iterator를 사용하여 반복하지 않는다.

If-else 문의 else를 제거할 수 있는 경우 else를 사용하지 않는다.

If 문이 한 줄일 경우 중괄호({})를 제거한다.

임의로 지역을 만들어 사용할 땐 같은 레벨의 문장은 같은 레벨의 지역으로 둔다.

들여쓰기는 항상 tab으로 한다.

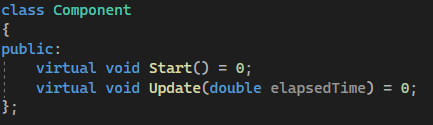
함수 내에서 중괄호({})를 사용하는 문장은 가독성을 위해 줄 바꿈을 한다.

코드의 구분을 위한 줄 간격은 1~2줄로 한다.

**그 외**

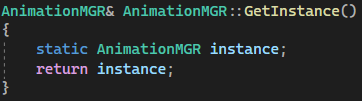
객체 기능 추가

객체의 기능은 미리 만든 Component를 상속하여 만들어 Object에 추가하는 방식으로 구현한다.



Singleton 패턴

Singleton 구현은 Static Local Singleton 패턴으로 진행한다.



#define 키워드를 사용하지 말고 const나 enum 키워드를 사용한다.

더 필요한 규칙이 필요할 경우 그 때 추가로 작성한다.