팀 드림코스트

코딩 스타일 가이드

Team DreamCoast Coding Style Guide

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** |
| 0207 | 15/02/07 | HD | Initial Document |
| 0208 | 15/02/08 | HD | Modify integer part after meeting |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 서론

## 본 문서의 목적(Purpose of this Document).

본 문서는 팀 프로젝트 작업 시, 팀원 간 코딩 스타일의 다름으로 생기는 문제를 예방하기 위해 미리 코딩스타일의 가이드 라인을 정함으로 불필요한 갈등을 예방하는 데 그 주 목적을 둠.

(This Documents is the guide line that prevent unnecessary conflict between teammates, which is from coding style difference.)

## 적용 범위

현 Direct3D프로젝트 포트폴리오에 적용된 모든 범위에 해당되며, 차후 구두약속의 기준 역시 본 문서의 적용, 혹은 수정을 중심으로 적용된다.

## 정의, 유의어, 약어 목록

차후 추가

# 변수, 속성, 함수, 클래스, 구조체 명명

## 함수, 클래스, 구조체, 속성, 그리고 열거형(enumerated type) (Public member methods, classes, structs, and enums)

* + 타입(소문자), 해당명

|  |  |
| --- | --- |
| c | Class |
|  | Function, Methods |
| st | Structs |
| e | Enums |
| f | Float | vec | Vector(STL) |
| n | Integer (의미가 있을 경우) | set | Set(STL) |
| d | Double | map | Map(STL) |
| ui | unsigned int | dw | Dword |

* + - 예시
      * 총알과 관련된 것들을 관리하는 클래스: cBulletManager
      * 업데이트 함수: void Update(float fDelta)
      * 포인트 구조체: stPoint
      * 상태를 관리하는 열거형: eState{}
    - 함수의 단어 선택
      * 함수명만 보더라도 그 함수가 무슨 역할을 하는 지 명시적으로 표현이 되어야 한다.
        + 예시  
          플레이어의 공격력을 인자로 받아와, 특정 계수에 따라 계산해서 float형태로 리턴 해주는 함수  
          float CalculateBulletDMGByStat(float fPlayerAttackPower)  
          (함수명 해석: 스탯에 의한 총알 데미지계산)
* 인자(Parameters) 및 변수
  + 위 법칙과 동일한 법칙을 이용한다.
* #define
  + 띄어쓰기 대신 밑줄을 이용하고 해당 매크로에 대한 적절한 설명이 되는 단어의 조합으로 생성한다.
  + 예시: 열의 최대치 #define MAX\_COL 15

# 클래스 내부 명명

## Private, Protected

* m 밑줄 변수설명(소문자) 변수명(PascalCasing)
  + Pascal Casing
    - 각 의미 있는 단어 맨 앞 대문자
  + 예시(Examples)
    - private 멤버변수로서 남은 총알 수를 관리하는 변수
      * private:  
         int m\_nNumBulletLeft;

## Public

* 변수인 경우에만 한해서 private, protect 내 법칙 적용
* 이외에는 일반 법칙과 동일하게 적용

# 코딩 스타일

## 싱글톤 패턴의 이용

* 싱글톤 패턴은 팀원들과 상의 하에 추가하도록 한다.
* 최대한 싱글톤 패턴의 개인적인 차원에서의 사용은 지양하는 선에서 마무리한다.   
  (게으른 코드가 되기 쉽상이다)

## 함수는 최대한 쪼갠다.

* Nesting(중괄호)가 많아지면 코드가 읽기가 어려워진다. 중괄호가 필요이상 늘어날 경우, 읽기가 복잡해진다.

## 한줄 짜리 if state문, switch case문이라도 Nesting을 꼭 실시한다.

* 코딩 가독성을 훨씬 높혀준다.

## 주석

* 당연히 코드를 통해 금방 확인할 수 있는 것은 달지 않는다.
* 나쁜 명명법에 주석을 달아 설명으로 처리하지 않는다. (차라리 길게 처리)
* 주석은 생각을 기록한다.
  + 주석에 있어서 몇 가지 법칙

|  |  |
| --- | --- |
| @TODO: | 아직 안 한 것 |
| @FIXME: | 오동 작이 알려진 코드 |
| @HACK: | 더 나은 해결책이 있을 것 같은 코드 |
| @XXX: | 이 코드는 위험한 코드 |

* 주석에는 코드의 의도를 설명하는 수준에서 마친다.

# 참조(Reference)

* Cwalina, K., & Abrams, B. (2006). *Framework design guidelines: Conventions, idioms, and patterns for Reusable .NET libraries*. Upper Saddle River, NJ: Addison-Wesley.
* Cox, L. (2015, December 15). Naming of Variables, Properties, Functions, Classes and Structs. Retrieved February 7, 2015, from http://www.gamasutra.com/blogs/LindsayCox/20141215/232280/C\_Coding\_Guidelines\_for\_my\_Personal\_Projects\_\_The\_2014\_edition.php
* Park, C. (Director) (2015, February 6). Game Programming Expert Training Lecture. *Game Programming Expert Training*. Lecture conducted from Seoul Game Academy, Seoul.
* Lee, W. (2014, April 21). 읽기 좋은 코드가 좋은 코드다. Retrieved February 7, 2015, from http://www.slideshare.net/e2goon/ss-33769330