코딩 스탠다드
msdn or unity document

코딩 스탠다드 - 원칙

1. 가독성을 우선시 합니다.(대부분의 경우, 코드는 문서 역활을 합니다)
2. 특별한 이유가 없다면, IDE 자동 포맷 스타일을 사용합니다.
3. 기존 코드로부터 학습합니다.

명명 규칙 및 스타일 **1**

- 클래스와 구조체는 파스칼 표기법(단어의 첫문자를 대문자)을 사용합니다.

```
class PlayerManager;
struct PlayerData;
```

명명 규칙 및 스타일 **2**

- 지역 변수와 함수 매개변수에는 카멜(낙타) 표기법을 사용합니다.

```
public void SomeMethod(int someParameter) //함수 이름도 파스칼 표기법
{
      int someNumber;
      int id;
}
```

명명 규칙 및 스타일 **3**

- 기본적으로, 메소드 이름은 동사원형-목적어 형식으로 작성합니다.

```
public uint GetAge() //uint == unsigned int
{
      // function implementation...
}
```

명명 규칙 및 스타일 **4**

- 그러나, 메소드가 불리언 타입을 반환한다면, 동사의 앞에 Is, Can, Should가 와야 합니다. 만약 이렇게 해도 이름이 자연스럽지 않다면, 다른 3인칭 단수형 동사를 사용합니다.

```
public bool IsAlive(Person person);
public bool Has(Person person);
public bool CanAccept(Person person);
public bool ShouldDelete(Person person);
public bool Exists(Person person); //자연스럽지 않는 경우
```

명명 규칙 및 스타일 **5**

- 모든 메소드의 이름은 파스칼 표기법을 사용합니다. (하이픈이 뭐지?)

```
public uint GetAge()
{
    // function implementation...
}
```

명명 규칙 및 스타일 **6**

- 상수에 대해서는 단어마다 언도바(_)으로 구분하여 모두 대문자로 작성합니다.

```
const int SOME_CONSTANT = 1; //define이 없기 때문에 이런 식으로 사용
```

명명 규칙 및 스타일 **7**

- 객체가 상수인 경우, static readonly(read only)를 사용합니다.

```
public static readonly MY_CONST_OBJECT = new MyConstClass();
```

- static readonly 변수들은, 단어마다 언도바(_)으로 구분하여 모두 대문자로 작성합니다.

명명 규칙 및 스타일 **8**

- 한번만 할당되는 변수들은, readonly를 사용합니다.

```
public class Account
{
      private readonly string mPassword;
      public Account(string password)
      {
            mPassword = password;
      }
}
```

명명 규칙 및 스타일 **9**

- 네임스페이스는 파스칼 표기법을 사용합니다.

```
namespace System.Graphics
```

명명 규칙 및 스타일 **10**

- 불리언 타입 프로퍼티 변수인 경우, 변수 접두사로 Is, Can, Should, Has를 사용합니다.

```
public bool IsFired { get; private set; }
public bool HasChild { get; private set; }
public bool CanModal { get; private set; }
public bool ShouldRedirect { get; private set; }
```

명명 규칙 및 스타일 **11**

- 인터페이스의 경우 접두사로 I를 붙여서 사용합니다.

```
interface ISomeInterface;
```

명명 규칙 및 스타일 **12**

- 프로퍼티는 파스칼 표기법을 사용합니다.
- public, protected 멤버 변수는 카멜 표기법을 사용합니다.
- private 멤버 변수인 경우, 접두사로 m을 붙여, 카멜표기법을 사용합니다.

```
public class Employee
{
      public int DepartmentID { get; set; }
      public int year;
      protected int day;
      private int mAge;
}
```

명명 규칙 및 스타일 **13**

- 값을 반환하는 메소드인 경우, 반환 값을 설명하는 이름을 가져야 한다.

```
public uint GetAge();
```

명명 규칙 및 스타일 **14**

- for 문이 아닌 경우에는, i 또는 e 같은 변수명 대신, 변수를 설명하는 이름을 사용합니다.

명명 규칙 및 스타일 **15**

- 만약 약어 뒤에 단어가 없다면, 약어를 모두 대문자로 표기합니다.

```
public int OrderID { get; private set; }
public string HttpAddress { get; private set; }
public bool GdeExists { get; set; }
private int mID;
```

명명 규칙 및 스타일 **16**

- Getter와 Setter 함수를 사용하는 대신에, 프로퍼티를 사용합니다.

```
use: //사용 O
public class Employee
{
      public string Name { get; set; }
}

Instead of: //사용 X
public class Employee
{
      private string mName;
      public string GetName();
      public string SetName(string name);
}
```

명명 규칙 및 스타일 **17**

- 웹 간격의 경우, 비주얼 스튜디오의 기본 값을 사용합니다. 만약 다른 IDE를 사용하는 경우, 실제 탭 대신 **4**번의 스페이스 간격을 사용합니다.

- 지역 변수를 선언 할 때, 해당 지역 변수를 사용하는 라인과 최대한 가까운 위치에 선언합니다.

명명 규칙 및 스타일 **18**

- 여는 중괄호는 항상 새로운 줄에 위치합니다.

```
Use:
if (bSomething)
{

}

Instead of:
if (bSomething) {

}
```

- if문안에 코드 한줄만 존재하는 경우, 중괄호를 사용하지 않습니다.

```
if (bSomething)
      return;
```

명명 규칙 및 스타일 **19**

- 명시적으로 double형 타입을 사용하는 것이 아니라면, float 형을 표현할 때, 고정 소수점 표식자 기호(f)를 사용합니다.

```
float f = 0.5f;
```

명명 규칙 및 스타일 **20**

- Switch문의 경우, 항상 default:를 가져야 합니다.
-

```
switch (number)
{
	case 0:
		break;
	default:
		break;
}
```

- 만약 Switch문에서 default: 에 도달해서는 안된다면, 항상 Debug.Assert(false) 또는 Debug.Fail()을 추가합니다.

```
switch (type)
{
	case 1:
		break;
	default:
		Debug.Fail("Unknown Type");
		break;
}
```

- 유니티에서는 Debug.LogError(“Unknown Type”);

명명 규칙 및 스타일 **21**

- 재귀 함수의 이름끝에 Recursive라고 표기합니다.

```
public void FibonacciRecursive();
```

명명 규칙 및 스타일 **22**

- 클래스 변수와 함수는 다음의 순서를 따라 위치합니다.
    a. public variables/properties
    b. internal variables/properties
    c. protected variables/properties
    d. private variables/properties
        Exception if a private variable is accessed by a property, it should appear right before the mapped property
    e. constructors
    f. unity messages (Reset, OnEnabled, OnDisabled, Awake, Start, Update…)
    g. public methods
    h. Internal methods
    i. protected methods
    j. private methods
    k. event handlers

명명 규칙 및 스타일 **23**

- 대부분의 경우, 함수 오버로딩은 가능하면 사용하지 않습니다.

```
Use:
public Anim GetAnimByIndex(int index);
public Anim GetAnimByName(string name);

GetAnimByIndex(5);
GetAnimByName(idle);

Instead of;
public Anim GetAnim(int index);
public Anim GetAnim(string name);

GetAnim(5);
GetAnim(index); //시각적으로 타입 확인 어려움.
GetAnim(idle);
```

명명 규칙 및 스타일 **24**

- 여러개의 클래스를 그룹화하는게 적절하지 않다면, 각각의 클래스는 별도의 파일로 존재해야 합니다. (클래스 하나에 다 만들지 마라)

- 파일 이름과 클래스 이름은 대소문자를 구분하여 이름이 같아야 합니다.

```
public class PlayerAnimation {}

PlayerAnimation.cs
```

명명 규칙 및 스타일 **25**

- 디폴트 매개변수를 사용하는 대신 오버로딩을 사용하세요.(FadeIn : 연출기법)

```
// 디폴트 매개변수를 사용한 경우
void FadeIn(float duration = 1.5f)
{
	...
}
FadeIn();
FadeIn(1.5f);

//디폴트 매개변수를 사용하지 않고, 오버로딩을 하는 경우
void FadeIn()
{
	float duration = GetDefaultValue();
	FadeIn(duration);
}

void FadeIn(float duration)
{
	...
}

FadeIn();
Fadein(1.5f);
```

```
// 디폴트 매개변수를 사용해서 컴파일 에러가 발생하는 경우
void FadeIn(float duration = 1.5f)
{
	...
}

void FadeIn(float duration, float delay = 1f)
{
	...
}

FadeIn(1f); //어떤 FadeIn() 함수 호출될지 모호해져, 컴파일 에러가 발생함.
```

명명 규칙 및 스타일 **26**

- 기본 매개변수를 사용할 때, null, false, 0처럼 변경할 수 없는 상수는 사용하지 않습니다.

```
// 상황에 따라서 0이 아닌 값이 디폴트 매개변수로 사용될 수 있다.
void FadeIn(float duration = 1.5f)
{
      ...
}
```

명명 규칙 및 스타일 **27**

- 할당될 값의 타입이 모호하거나, 타입이 중요한 경우, var를 사용하는 대신 실제 타입을 사용하세요. IEnumerable 혹은 new 키워드처럼, 해당 오브젝트의 타입이 명확하게 알 수 있는 경우, var이 사용 가능합니다.

```
var text = "string obviously";
var age = 20;
var employee = new Employee();
// 기존 사용 예) 타입이름이 긴 경우에 var 사용함

// 코드를 쳤을 때, 리턴 값의 타입 확인에 어려움이 있음
string accountNumber = GetAccountNumber();
```

명명 규칙 및 스타일 **28**

- 외부의 데이터를 사용할 때, 실제 사용할 함수에 적용하기 전에 데이터가 유효한지 검사하세요. 검사를 통과하면, 이 시점부터는 모든 데이터는 유효하다는 것을 의미합니다.

- 그러므로, 내부에서는 예외를 던질 필요가 없습니다. 또한 예외는 경계부분에서 처리되어야 합니다.

명명 규칙 및 스타일 **29**

- 예외적으로, switch문의 default에서 처리되지 않은 enum에 대한 예외를 던지는 것을 허용합니다. 하지만, 이에 대한 예외를 처리하지는 않습니다.

```
switch (accountType)
{
      case AccountType.Personal;
            return something;
      case AccountType.Business;
            return somethingElse;
      default:
            throw new ArgumentOutOfRangeException(nameof(AccountType));
}
```

명명 규칙 및 스타일 **30**

- null 매개변수를 사용하지 않습니다. (특히 public 함수에서)

- 만약 null 매개변수를 사용한다면, 매개변수 이름에 접미사로 OrNull을 붙여서 사용합니다.

```
public Anim GetAnim(string nameOrNull)
{

}
```

명명 규칙 및 스타일 **31**

- 함수의 리턴 값으로 null을 사용하지 않습니다. (특히 public 함수에서)

- 만약, 함수의 리턴 값으로 null을 사용한다면, 함수 이름에 접미사로 OrNull을 붙여서 사용합니다.

```
public string GetNameOrNull();
// 다른 대안
public bool TryGetName(out string name);
```

명명 규칙 및 스타일 **32**

- 한 라인에, 오직 하나의 변수만 사용합니다.

```
BAD:
      int counter = 0, index = 0;

GOOD:
      int counter = 0;
      int index = 0;
```