

COMP130: Game Architecture

2: Epic Coding Standards

Learning outcomes

- ▶ **Understand** Epic coding standards
- ▶ **Compare** different coding standards
- ▶ **Apply** Epic coding standard to your own Unreal Projects

Why Coding Standards?

- ▶ Aids in software maintenance
- ▶ Improves readability and understandability
- ▶ Adds to the documentation of the project

Naming Conventions 1

- ▶ First letter of each word in name is capitalised, and no underscores between words
 - ▶ E.g. Health and UPrimitiveComponent, not lastMouseCoordinates or delta_coordinates
- ▶ Type and variable names are nouns
- ▶ Method names are verbs that describe the effects or describe return value
- ▶ Names should be clear unambiguous, and discriptive. Avoid over-abbreviation.

Naming Conventions 2

- ▶ All variables should be declared one at a time to allow comments
- ▶ All functions that return bool should ask a true/false questions
- ▶ A procedure(a function with no return) should use strong verb followed by an Object
- ▶ Prefix function parameters passed by reference with **Out**

Naming Conventions 3

- ▶ Type names are prefixed with an additional upper-case letter, to distinguish them from variable names. E.g. **FSkin** for type, and **Skin** is an instance of **FSkin**
 - ▶ Template classes are prefixed by T
 - ▶ Classes that inherit from UObject are prefixed by U
 - ▶ Classes that inherit from AActor are prefixed by A
 - ▶ Classes that inherit from SWidget are prefixed by S
 - ▶ Classes that are Interfaces are prefixed by I
 - ▶ Enums are prefixed by E
 - ▶ Boolean variables must be prefixed by b (e.g. bIsDead or bHasFallen)
 - ▶ Most other classes are prefixed by F

Naming Convention Examples

```
float TeaWeight;  
  
int32 TeaCount;  
  
bool bDoesTeaStink;  
  
FName TeaName;  
  
FString TeaFriendlyName;  
  
UClass* TeaClass;  
  
USoundCue* TeaSound;  
  
UTexture* TeaTexture;  
  
bool IsTeaFresh(UTea Tea)
```

Portable Aliases for C++ Types

Unreal Type	C++ Type	Size
bool	bool or BOOL	Never assume size
TCHAR	TCHAR or char	Never assume size
uint8	unsigned bytes	1 byte
int8	bytes	1 byte
uint16	unsigned short	2 bytes
int16	short	2 bytes
uint32	unsigned int	4 bytes
int32	int	4 bytes
uint64	unsigned long	8 bytes
int64	long	8 bytes
float	float	4 bytes
double	double	8 bytes
PTRINT	void*	Never assume size

Comments - Write self documenting code

```
// Bad:
```

```
t = s + 1 - b;
```

```
// Good:
```

```
TotalLeaves=SmallLeaves+LargeLeaves- ↵  
    SmallAndLargeLeaves;
```

Comments - Write Useful Comments

```
// Bad:  
// Increment Leaves  
++Leaves;
```

```
// Good:  
// we know there is another tea leaf  
++Leaves;
```

Comments - Do not comment bad code - rewrite it

```
// Bad:  
// total number of leaves is sum of  
// small and large leaves less the  
// number of leaves that are both  
t = s + 1 - b;  
  
// Good:  
TotalLeaves=SmallLeaves+LargeLeaves- ↵  
    SmallAndLargeLeaves;
```

Comments - Do not contradict code

```
// Bad:  
// never increment Leaves!  
++Leaves;
```

```
//Good:  
// we know there is another leaf  
++Leaves;
```

Comments - Formatting

- ▶ Unreal uses a system based on JavaDocs to extract comments to build documentation
- ▶ You have to use a specific format in order for this tool to run
- ▶ More info -
[urlhttp://www.oracle.com/technetwork/articles/java/index137868.html](http://www.oracle.com/technetwork/articles/java/index137868.html)

Const Correctness

- ▶ Const is documentation as much as a compiler directive
- ▶ If function arguments aren't modified by a function, ensure they are passed with const keyword
- ▶ Flag methods as const if they don't modify an object
- ▶ Use const iteration over containers if the loop doesn't modify container
- ▶ Const should be preferred on by-value parameters and locals

C++11 and Modern C++

- ▶ **nullptr** - Used instead of NULL except on XboxOne use **TYPE_OF_NULLPTR**)
- ▶ **auto** - You should not use auto except in the following cases
 - ▶ Binding to lambda types
 - ▶ Iteration variables
 - ▶ Template code (advance case)
- ▶ **Range Based For** - Preferred to regular for loop
- ▶ **Lambdas** - Can be used, but be careful (see docs)
- ▶ **Enums** - Always use strongly type enums which inherit from **uint8**
- ▶ **Move Semantics** - All main container types have move constructors and move assignment

References

[https://docs.unrealengine.com/latest/INT/
Programming/Development/CodingStandard/](https://docs.unrealengine.com/latest/INT/Programming/Development/CodingStandard/)