BSc Computer Games Programming: programming naming conventions

Introduction

This document contains the naming standards that you must adhere to.

Basics

Variables, constants and functions must have sensible, informative names. Looping variables may be called i, j, k, etc. Other variables should not use these names (and certainly not functions).

Function names must use lowerCamelCase or UpperCamelCase. It doesn't matter which is selected (although the former is more common for Java) as long as it used consistently throughout a project.

Variable names must UpperCamelCase if they have a prefix and lowerCamelCase if not.

User defined type names (e.g. classes) must use UpperCamelCase.

Prefixes

The vast majority of variables should have prefixes. These are characters that appear at the start of a name to provide type information about a variable. There are some exceptions which are given below.

The general format of variable names should be as follows:

[scope][array][pointer][const][type]SensibleName;

The [] brackets mean that this character might be optional. As stated above, if no prefix is to be used the variable name must use lowerCamelCase.

There might be occasions where the order is varied or where multiple instances of each are required (e.g. const pointers to const variables). Discuss with your tutor if you feel that might be necessary.

The characters to be used for these prefixes are as follows:

Type

Type	Prefix character
Integer	i (signed) ui (unsigned)
Char	c (signed) uc (unsigned)
float and double	F
bool	b
String	s

For user defined types some sensible single character can be used or the type somehow named or implied in the variable name, e.g. vPlayerVelocity or playerVelocityVec (both vectors). Note that, of course, you shouldn't use single characters that have another use in the naming standards.

Arrays:

prefix = a

Consts

prefix = k

Pointers

prefix = p

Scope

Scope	Prefix
global	g_
static	s_
member	m_

Examples

```
int iNumber;
unsigned int uiNeverNegative;
string sName;
float fValue1;
double fValue2;
char cChar;
static const int s_kiArraySize = 16;
char acName[s_kiArraySize]; // array of char
char* pcName; // pointer to character;
char* apcNames[s_kiArraySize]; // array of pointers to char
const int kiCannotChange = 32; // const int
const int akiIndices[] = {0,1,2,3,4}; // array of const int
int g_iGlobal;
static int s_iClassMember;
float m_apPlayers; // array of pointers to Player objects (a user defined type)
```