University of North Alabama

CS 455

Software Engineering Project

Spring 2013

General

All documentation shall be created and written using Google Docs.

Git and GitHub shall be used for source code revision control.

A README shall accommodate all source code, shall be written in either text or Markdown format.

Documentation released to customers shall be released in PDF format.

Documentation

The default font for text shall be Times New Roman size 12.

Source code place in document shall be in Courier New font size 11.

Coding

General

A tab shall equal 4 spaces.

No line shall be longer than 80 characters.

All variables shall be declared before used.

All functions shall be declared before they are used.

Unless there is a language constraint, functions and variables shall start with a lowercase letter.

All globals, constants, and defined names shall be in all caps. Multiple words shall be separated with an underscore.

```
#define SIDE_FRONT 0
```

Trailing braces shall be used everywhere (if, else, functions, structures, typedefs, class definitions, etc.)

```
if ( x ) {

}
```

The else statement shall start on the same line as the last closing brace.

```
if ( x ) {
} else {
}
```

Pad parenthesized expressions with spaces

```
if ( x ) {
}
```

Instead of

```
if (x) {
}
```

And

```
x = ( y * 0.5f );
```

Instead of

```
x = (y * 0.5f);
```

Or

```
x=(y*0.5f);
```

Java / C++ / JavaScript / PHP

In multi-word function names, each word shall start with an upper case:

```
    void thisFunctionDoesSomething( void );
```

eval shall not be used

Non-bounds checking C functions such as strcpy(), strcat(), sprintf() shall not be used.

File names in C++ and JavaScript shall be in all lowercase letters.

Precision specification for floating point values shall be used unless there is an explicit need for a double.

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

Instead of

```
        float f = 0.5;
```

And

```
        float f = 1.0f;
```

Instead of

```
        float f = 1.f;
```

The standard header for functions shall be:

```
/*

====================

functionName

Description

====================

*/
```

The first word of multi-word variable names shall start with a lower case character and each successive word starts with an upper case.

```
              float maxDistanceFromPlane;
```

Typedef names shall use the same naming convention as variables, however they always end with "_t".

```
               typedef int fileHandle_t;
```

Struct names shall use the same naming convention as variables, however they always end with "_t".

```
               struct renderEntity_t;
```

Enum names shall use the same naming convention as variables, however they always end with "_t". The enum constants use all upper case characters. Multiple words are separated with an underscore.

```
             enum contact_t {

                   CONTACT_NONE
             };
```

Names of recursive functions shall end with "_r"

```
            void walkBSP_r( int node );
```

The standard header for a class shall be:

```
/*
===============================================================================
Description
===============================================================================
*/
```

Class variables shall have the same naming convention as variables.

Class methods shall have the same naming convention as functions.

Ordering of class variables and methods shall be as follows:

1. public variables
2. public methods
3. protected variables
4. protected methods
5. private variables
6. private methods

Each class shall be in a separate source file unless it makes sense to group several smaller classes. The file name shall be the same as the name of the class.

Ruby / Perl / Python

In multi-word function or variable names, each word shall start with a lowercase letter and each word shall be separated by an underscore:

```
        def this_function_does_something
```

This sample standard is a revision of one used by Tuft University.

Possible Tools:

- FogBugz
- Gliffy
- Git and GitHub
- Pivotal Tracker
- Veracode Static Analysis