Code Standard

C++

This document exists to give developers a coherent image of the syntax style. Camel casing should be used and used properly! A more elaborate explanation of this can be found at: http://en.wikipedia.org/wiki/CamelCase .

Dead code

Code that is not used should not be in the source code. The End!

Code comments

Comments should be placed adjacent to the code it belongs to. English language should be used and used correctly! Google is your friend! Sentences should begin with capital letter. Comments should be descriptive and compact!

Variables

Variable name should always be descriptive and well defined.

ex. `int variableName = 0;`

Variables representing number of elements

Variables representing number of elements should always be descriptive and begin with 'nrOf'.

ex. `int nrOfElements = 5;`

Constants

Constant name should always be descriptive and written in capital letters. Words are divided by an underscore.

ex. `const int CONSTANT_VALUE = 1;`

Functions

Function name should always be descriptive and begin with a capital letter. A blank space should be left in the beginning as well as the end of the function parameter list. If function does not have a parameter list, then a blank space should **not** be used.

Default parameters are placed last in the parameter list as it is demanded by the compiler. Functions that is read-only should always be declared with the keyword const.

ex.
```
void Foo( int index, string name = "default" );

void Bar();

int GetIndex() const;
```

Classes

Members

Member name should always be descriptive and begin with 'm'.

ex.
```
int mHealth;
```

Templates

Template names should always be descriptive.

ex.
```
template <typename T>
void PrintValue( T* value );
```

Functions

See description below the 'Function' section above.

Initialize function

Initialize functions should always be named 'Initialize' and return a HRESULT.

ex.
```
HRESULT Initialize( int value );
```

Release function

Release functions should always be named 'Release' with no parameter list.

ex.
```
void Release();
```

Constructors

Default Constructor should always be implemented in class and initialize members

Structure

The structure in class should as follow:

```
class B : public A
{
        //Members
        private:
        protected:
        public:

        //Templates
        private:
        protected:
        public:

        //Functions
        private:
        protected:
        public:
                HRESULT Update( float deltaTime );
                HRESULT Render( float deltaTime );
                HRESULT Initialize();
                void Release();
                B();
                ~B();
};
```

Mathematical expressions

Mathematical expressions should be defined with a blank space between the operand and the operator.

ex. `int sum = 5 + 5;` **Note:** Parenthesis can be used for clarification **BUT** be aware that they can alter the expression.

Control structures

Control structures should always be defined with blank spaces within the parenthesis.

ex. `if ( value < 5 )`

Object subscripting

Object subscripting should not contain any blank space directly within the square brackets or parenthesis.

ex. `vertices[i] = vertex;` **or** `vertices.at(i) = vertex;`

Object subscripting with mathematical expression

Object subscripting with mathematical expressions should not contain any blank space directly within the square brackets or parenthesis. There should be a blank space between the operand and the operator.

ex. `vertices[i + 2] = vertex;` **or** `vertices.at(i + 2) = vertex;`

Type casting

Blank space should **not** be included within the parenthesis at type casting.

ex. `int sum = (int)floatSum;`

Questions and/or additions to this document?

Contant SvinSimpe!