AI Puzzle Framework

Coding Standards Document

Brian Shaver
March 21, 2005
# Table of Contents

Table of Contents ............................................................................................................................. 2

Introduction ...................................................................................................................................... 3

Variables .......................................................................................................................................... 3
  Rules ......................................................................................................................................... 3
  Conversion Table .................................................................................................................. 3

Constants and Definitions ................................................................................................................ 4

Files .................................................................................................................................................. 4
  Comment Block ........................................................................................................................ 4
  Rules ......................................................................................................................................... 4

Classes .............................................................................................................................................. 4
  Comment Block ........................................................................................................................ 4
  Rules ......................................................................................................................................... 5

Structures ......................................................................................................................................... 5
  Comment Block ........................................................................................................................ 5
  Rules ......................................................................................................................................... 6

Functions .......................................................................................................................................... 6
  Comment Block ........................................................................................................................ 6
  Rules ......................................................................................................................................... 6

Miscellaneous .................................................................................................................................. 7
  Indentation ................................................................................................................................ 7
  Braces ........................................................................................................................................ 7
Introduction
This is a coding standards document intended for use with the C++ language. These standards should be followed for all source code developed for the AI Puzzle Framework. The use of these coding standards is to promote reusable, reliable, extensible, easily maintainable, cross-platform code which is well commented and documented. The commenting standards presented here are intended to be compatible with the Doxygen documentation tool.

Variables

Rules
1. All variable names should be in title case, with the exception of the first letter of the name, which should be in lower case. Ex: minVelocity, maxRecordedVelocity, speed, etc.
2. Variable names should be logical and descriptive.
3. All global variables must be contained within a namespace.
4. There should be no static variables in class member functions.
5. Built-in variable types should not be used. Instead use the typedef’s for the types, following the conversion table provided below.

Conversion Table

<table>
<thead>
<tr>
<th>Built-in Type</th>
<th>Standard Type</th>
<th>Size (bytes)</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>bool</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>int8</td>
<td>1</td>
<td>-127</td>
<td>128</td>
</tr>
<tr>
<td>unsigned char</td>
<td>uint8</td>
<td>1</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>short</td>
<td>int16</td>
<td>2</td>
<td>-32,767</td>
<td>32,768</td>
</tr>
<tr>
<td>unsigned short</td>
<td>uint16</td>
<td>2</td>
<td>0</td>
<td>65,535</td>
</tr>
<tr>
<td>int</td>
<td>int32</td>
<td>4</td>
<td>-2,147,483,647</td>
<td>2,147,483,648</td>
</tr>
<tr>
<td>unsigned int</td>
<td>uint32</td>
<td>4</td>
<td>0</td>
<td>4,294,967,296</td>
</tr>
<tr>
<td>float</td>
<td>real32</td>
<td>4</td>
<td>1.5e-45</td>
<td>3.4e+38</td>
</tr>
<tr>
<td>long</td>
<td>int64</td>
<td>8</td>
<td>-9.2e18</td>
<td>9.2e18</td>
</tr>
<tr>
<td>unsigned long</td>
<td>uint64</td>
<td>8</td>
<td>0</td>
<td>1.8e19</td>
</tr>
<tr>
<td>double</td>
<td>real64</td>
<td>8</td>
<td>5.0e-324</td>
<td>1.7e+308</td>
</tr>
</tbody>
</table>
Constants and Definitions

1. All constants should be uppercase
2. Never use \#define except in header file \#ifndef statements. Always define constants.
   Ex: `const [type] name = [value]`

Files

Comment Block

Every file should contain the following header comment block, which contains CVS expansion keywords used to place log information into the comment block:

```c
//////////////////////////////////////////////////////////
/// @file         Header.h
///
/// @author       Your name here
///
/// @compiler     C++
///
/// @project      Puzzle Framework
///
/// @description  A brief description of the file
///
//////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////
/// VERSION CONTROL HISTORY
/// $Revision:$
///
/// $Log:$
///
//////////////////////////////////////////////////////////
```

Rules

1. Files should be no more than 1000 lines long.
2. Each line of code should not exceed 80 characters.
3. File names should be in title case.
   Ex: `BoardState.cpp`

Classes

Comment Block

Classes should always be preceded by the following comment block:
Rules

4. Class names should always begin with a ‘C’.
   Ex: CBook
5. Interface class names should begin with an ‘I’.
   Ex: ISocket
6. Class names should be in title case.
7. Class member variable names should always be prefixed with “m_”.
   Ex: m_minValue
8. Accessor functions names should always be preceded with either Get or Set.
   Ex: SetMaxLimit(...) and GetMaxLimit().
9. A “Get” accessor should not take any arguments.
10. A “Set” accessor should take a single argument and return a void type.
11. All public attributes should be const.
12. All classes that use heap memory must have a copy constructor defined.
13. All dynamic memory allocated in a class must be deleted by the same class.
14. Public class functions should never return a pointer to dynamically created memory. The exception to this is Singleton or Factory classes.

Structures

Comment Block

Structures should always be preceded by the following comment block:

```cpp
/****************************************************************************
/// PutStructNameHere
///
/// @description A brief description of the struct
///
/// @limitations Any limitations for the struct, or None
///
/****************************************************************************
```
Rules

1. Structure names should be in all uppercase.
2. Structure member variable names should always be prefixed by “s_”.
   Ex: s_minValue
3. Structures should not contain functions. If a function is needed, the structure should be
   made into a class.

Functions

Comment Block

Functions should be preceded by the following comment block:

```c
/************************************************************
/// GetArea
///
/// @description A brief description of the function
///
/// @param w: The width of a rectangle
/// @param h: The height of a rectangle
///
/// @return uint32: The area of the rectangle
///
/// @limitations Only works for rectangles
///
uint32 GetArea(uint32 w, uint32 h)
{
    ........
}
```

1. If no parameters are present, omit param lines entirely.
2. If the function returns void, or has no return (such as for a constructor), omit the
   return line entirely
3. If the function has no limitations, simply enter None after limitations.

Rules

1. Function names should always follow variable naming conventions, with the exception
   that the first letter of the function name should be title case.
   Ex: GetArea(...) 
2. Functions should be no more than 400 lines of code and comment.
3. Functions should be const whenever possible.
4. Function parameters should be made const whenever possible.
Miscellaneous

**Indentation**

1. Tabs should be set to 4 spaces.
2. The code within a scope should always be indented one tab.
3. Code should not be nested any deeper than five levels of indentation.
4. Code files should not contain any tab characters; use spaces (detabify document).
5. The `else` statement in an if-else block should be placed on its own line, as shown here:

```plaintext
if(condition)
{
    // body
}
else
{
    // body
}
```

**Braces**

1. Opening French braces should be on their own line.
2. Closing braces should have the same indentation as the associated opening brace.