**Project Skeever Coding Standards**

by Winston Oswald-Drummond

**Indentation**

For our project, we’ll be using Allman style indentation. Each indent will be equivalent to 4 spaces. Curly brackets will be on their own line, but they won’t be further indented. The content within the curly brackets will be indented. Functions shall be separated by 1 blank line.

Example:

void ExampleFunction()

{

//function code

}

void ExampleFunction2()

{

//function code

}

**Naming**

Unique and descriptive (but not redundant) names will be used throughout. The name should tell us something about the thing being named, without being too long or too short.

**Functions/Methods:**

For function names, our team will be using PascalCase. This means function names will start with a capital letter, and new words will be distinguished using capital letters (but not spaces) at the start of each new word.

Example:

void ExampleFunction();

int Example();

**Variables:**

Regular variable names are to be written in camelCase, which is similar to PascalCase but the beginning starts with a lowercase letter.

Example:

int exampleInt = 5;

string thisIsAnExampleString;

int example;

Constants will be in all CAPS and words will be separated by underscores.

Example:

const int THIS\_IS\_A\_CONSTANT\_VARIABLE;

**Classes:**

Our class and type names will use camelCase.

Example:

class exampleClass

{

//class code

}

**Files:**

File names shall use PascalCase, beginning with a capital letter.

Example:

ExampleFile.cs

**Commenting**

Comments will be written before the line/chunk of code it’s describing, or if it’s short enough it may be written on the same line. Also, header comments summarizing the use of the file are allowed and may be pushed for later in our development.

Single line comments example:

//allows the player to open their inventory

void OpenInventory()

{

code… // inline comment example

}

Multi-line comments example:

/\*

This function makes the enemy go after the player.

The enemy will always run in the player’s direction, unless something is in the way.

\*/

void EnemyMovement()

{

//code

}

**Error Handling**

The assert function will be used to deal with error handling.

Example:

void assert(expression);