# Music Box Games Programming Style Guide

Ryan Hanson

2019-09-25

# Contents

# Overview

This document outlines the required programming style for development at Music Box Games. This is an internal document and is not to be shared.

# 1 Source Control

## 1.1 Git

All code should be regularly checked into the development git repository on a branch given an appropriate name (If you're working on graphics, the branch should be named graphics). Working in master is only allowed with permission and final merges of code should be done by the project's Technical Director after sufficient testing.

# 2 Commenting

## 2.1 C++

Documentation for C++ source code should be done with Doxygen style comments. Every source and header file must contain a header comment, and each function should have their own header comment.

A good file header comment should explain the purpose of code in the file in as much detail as necessary:

```
1  /******************************************************************************/
2  /*!
3  \file   hdrcmt.cpp
4  \author Ryan Hanson
5  \par    email: iovita\@musicboxgames.net
6  \par    Project: Full Hearts
7  \date   9/25/2019
8  \brief
9      This file is an example of a file header comment.
10
11 \copyright    All content (c) 2019 Music Box Games, all rights reserved.
12 */
13 /******************************************************************************/
```

Function header comments should give a more detailed explanation than the file header, as it is specific to only one function. Details on the inputs, outputs, and functionality of function.

```
/*****************************************************************************/
      /*!
        \param val
            The value to be checked.
        \return
            Returns if given value is even or not.
        \brief
            Checks that a given value is even.
      */
/*****************************************************************************/
bool is_even(int val)
{
    return val % 2 == 0;
}
```

Additional info such as run-time complexity can also be added at your discretion.
For more information of Doxygen style commenting, see here

# 3 Language

You should always program using modern, but reliable conventions.

## 3.1 C++

The current approved standard for C++ is C++ 17, this will be updated when a newer standard is supported on all operating systems. Please use as much modern C++ as you can.

# 4 Build Systems

Please use the build systems provided in the development repositories. For more specific details on build systems, see the Project Primer document present in the repository.