Rock Chalk Rendezvous	Version: 0.1
Source Code Standards	Date: 3/23/2024
code_stds1	

Source Code Standards

Revision History

Date	Version	Description	Author
03/21/2024	0.1	Initial structure and some standards included.	Ben Phillips

Rock Chalk Rendezvous	Version: 0.1
Source Code Standards	Date: 3/23/2024
code_stds1	

Table of Contents

Rock Chalk Rendezvous	Version: 0.1
Source Code Standards	Date: 3/23/2024
code_stds1	

1. Introduction

<describe the purposes of this document>

2. Practical

2.1. Comments

Comments should only be used to clarify code whose intent is not already clear and not able to be made clear. Names that follow the guidelines in section 3.1 should make comments unnecessary in many places. Signposting comments, combined with proper spacing, also help when locating different sections of a longer file.

2.2. Auto

The “auto” keyword should not be used unless the type has a long name and either the type is unimportant and a more descriptive name is used or the type is obvious from the initialization.

2.3. Types

2.4. Structs

Functions should be methods of a struct only if their functionality is inherent to the data being stored by the struct.

2.5. Inheritance

No struct should extend any other struct. Other methods of code reuse are just as effective and are much less ambiguous when methods are used.

3. Formatting

3.1. Naming

Names for values should suitably describe how their purpose differentiates them from other values of the similar types. Longer names, especially function names, improve the ability of code to accurately explain what it is doing without the need for comments. Single letter names should be avoided unless the type or context gives enough information about it.

Abbreviations should never be used in names in order to avoid ambiguity. For example, “function” should never be shortened to “funct”, “func”, “fn”, or “f”, it should be spelled out entirely. If a name is used in a frequent and widespread manner, like the “int” keyword, abbreviations are acceptable.

Camel case, snake case, when

3.2. Indentation

Spaces vs tabs, size

3.3. Curly Braces

When to use / not use where optional

3.4. Number representations

When numeric literals are the result of combining multiple values together, they should always be expressed in the terms of those operations. For example, five hours in minutes should be written as 5 * 60 rather than 300.

When the same numeric value is used in many places whose meaning is not otherwise clear, it should be represented with a macro using a #define directive in order to label and potentially facilitate modifying this value.

4. Testing

4.1.