**CyberVandals’ Coding Standards**

1. **Naming Variables**

**Variables**

* Variable names should all be lowercase with white space represented by underscore ‘\_’.
* Prefixes should be used for Boolean variables; is\_ and has\_
* Postfix should be used for counting variable; \_count
* Variables should be named according to the value or object it represents.

Example : health

play\_button

is\_play\_button\_pushed

banana\_count

**Constant**

* Constant variable names should all be uppercase with white space represented by underscore ‘\_’.
* Variables should be named according to the value or object it represents.

Example : SCREEN\_HEIGHT

SCREEN\_WIDTH

**Type – class, struct, enum, etc**

* UpperCamelCase style should be used. The type names should start with an uppercase letter and every beginning of each word with an uppercase as well.
* Types should be named so that you can use the same name for genetic variables

Example : MainWindow \* main\_window;

SceneManager \* scene\_manager;

MainPlayer \* main\_player;

1. **Naming Functions**

**Functions**

* The same naming convention for variables should be used for naming functions. Function names should be all lowercase and underscores for spaces, if any.
* Functions should be named according to its job. A function needs to do what the name says it does.
* Since functions “do” something, a function name should include a verb.
* When there are parameters in a function, you should put space right after open parenthesis and right before close parenthesis.

Example : do\_something( with, theses, parameters );

create\_gameover\_screen( BTN\_HEIGHT, BTN\_WIDTH,

main\_window, view );

1. **Program Flow**

**Conditional Statements**

* For if statements, follow the example below.

if( condition ) {

statements;

}

else if( condition ) {

statements;

}

else {

statements;

}

if, else if are followed immediately by ‘(‘, and a space is used right after open parenthesis and right before close parenthesis. Open curly brace is placed in the same line with condition after a space.

* For loops, follow the same convention used in if statements

// while loop

while( condition ) {

statements

}

// for loop

for( int i = 0; i < MAX\_ELEMENTS; i++ ){

statements;

}

In for loops, variable names such as i, j, k, m should be used for iterating over numbers and name ‘it’ for iterating over objects.

* If conditions are longer than a line, then put open curly braces below the condition phrase

if( condition1

|| condition2

|| condition3 )

{

statements;

}

**Indentation**

* 4 spaces should be used for indentation

Example :

int main() {

while( condition ) {

statements;

if( condition ) {

statements;

}

statements;

}

}

**Comments**

* In every header file, comment should be placed containing the name of the file, name of the author and any additional information regarding its header file.

Example : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MainWindow.h

\* Chaeun Kim

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

* Comments should be left explaning what is being done where the purpose of the segment of your code is not intuitive or clear.
* For comments within code, double slash ‘//’ comment should be used.

Example : statements;

// Iterate through array and print

for ( int i = 0; i < MAX\_ELEMENT; i++ ) {

qDebug() << array[i];

}

statements;

1. **Error Handling**

* Use C++ function assert() to handle illegal inputs

Example :

create\_scene(int scene\_mode, QGraphicsScene \* scene){

// value of scene\_mode has to be between 0 and 2

assert( 0 <= scene\_mode <= 2);

// scene cannot be NULL

assert(scene != NULL);

statements;

}