# CSC 581 Homework 5 Writeup

Jae Jimmy Wong

## 1 Introduction

The final assignment is to implement a script management system, some scripted functionalities, improved input handling, and two new games to demonstrate the reusability of the game engine.

## 2 Scripting

In this design, I wrapped all scripting related code into a single `script_manager` class. The class has three main functionalities. First, it manage the life cycle of the scripting engine including the initializing and disposing of the V8 platform and the JavaScript virtual machine. The second function of the class is providing a convenient interface for the game server to run scripts. Last one is adding the binding of C++ objects to the context of JavaScript runtime.

Since the V8 environment can only be initialized once in the same process, I implemented the class based on the Singleton pattern. In the pattern, all instances of the class are referencing to the one same instance. C++ itself does not have any language feature that supports this design pattern, so I use static members and a special constructor to achieve the pattern.

To create a direct mapping between C++ memory space and the JavaScript space, I find out that the most simple and efficient way is to use a only one JavaScript virtual machine with a long-living global context. The arrangement not only avoids the cost to construct and destruct a virtual machine each time the manager runs a script, but also prevents inconsistencies between different contexts. Since long-living contexts are not the default usage of V8, it requires the implementation to use the special "Eternal" handle to prevent the garbage collector from deleting the context.

For each attribute of game objects that needs to be accessible from the scripting environment, a getter function and a setter function are added to the static scope of the `script_manager` class. Then, the two functions are referenced by the `SetAccessor` method of the handle of the global object. Though this takes a lot of boilerplate code, it gets the jobs done without any complex structure.

## 3 A Second Game

The second game is a simplified replica of the “NS-SHAFT” game. The players of the game need to jump down lifting platforms orderly to increase their score. If a player drop off the bottom of the game screen, the game is over. By calculating the number of changed line, I found that 16.5% of the code is different from the first game.

## 4 A Third Game

The third game is a version of the “flappy bird” game. The players has to try go through the gap of two vertical pipes with a single jump button. If the character touches any pipes or the ground, the game is over. By calculating the number of changed line, I found that 20.1% of the code is different from the first game.