

*CPSC 2720: Game Improvement Report*

*Created by:*

Brett Regnier <[brett.regnier@uleth.ca](mailto:brett.regnier@uleth.ca)>

Daylend de Grasse <[daylend.degrasse@uleth.ca](mailto:daylend.degrasse@uleth.ca)>

Brad Lindsay <[brad.lindsay@uleth.ca](mailto:brad.lindsay@uleth.ca)>

Michael Wynnychuk <[m.wynnychuk@uleth.ca](mailto:m.wynnychuk@uleth.ca)>

***Introduction***

This document describes in one section the testing approaches that were used to verify the existence of bugs and other non-bug issues. In the second section it shows the improvements made to the software that was partly made by those bug fixes.

***Testing Approach*** – As this was our software, we were able to test the software from multiple perspectives. We performed user interface testing, some edge case testing, and semi-grey white testing. As we had access to the debugger, and unit tests, this allowed us to get a more in depth analysis of problems we faced.

Interface testing was performed by repeated attempts to open the application, and simply play the game. We tried to throw invalid inputs to the application in an attempt to cause errors to occur, but we found that our initial menu design was well suited for sanitizing input.

Each game was repeatedly tested to ensure that the game logic was working correctly. Each developer would test and debug their own application as they were the most familiar with the rulesets of the game. While this was beneficial in regards to time constraint, it was not beneficial in the fact that it’s biased towards producing positive results.

We have limited tests for our application. The unit tests cover some of the larger more commonly used parts of the application that contain public interfaces, we decided not to unit test most pieces of the application, for example the concrete implementations of the games due to the fact that they contained a lot of private members and functions that would not be interfaces with.

***Improvements***

Improvements to the application were done in a limited fashion. We did do some improvements to memory management, the save load functionality, and the help functionality of the game. A lot of bugs were found in the Crazy Eights game, and they were squashed through an iterative process implemented by Michael.

We did not get much feedback from other groups on our application. To date, I believe there was only one bug report ever filed, and with that bug report we pushed to resolve quickly. The bug report revolved around problems with the compilation process, and we updated the appropriate documentation to reflect the proper process more actively.

It would have been nice to have all groups test all projects in some manner to help identify edge case bugs in our application, but again, time constraint would have been the biggest problem here.

Most game initially did not have any help functionality. This was a feature that was added later by Brett. He implemented a very slick little display feature in the GameMenu class that allowed for the help to be shown in a succinct way.

As our initial design leveraged design patterns already, we did not feel the need to do large scale refactoring to implement more patterns. If we were to visit the project again, we would probably leverage design patterns in a much larger way.