**Reengineering Project Final Report - Title of the software project**

*Steven Hartnet, Jonah Tolefson, Kyle Hoffheim, Kyle Kwasniewski*

1. **Building Instruction**

* SystemFrame.java – Main system UI
* BasePanel.java – Parent class of all panels
* MainMenu.java – Starting panel that takes in inputs
* Game\_Panel.java – Game panel uses the game controller
* GameController.java – Main control class that handles almost all calculations and processing
* IntroPanel.java – Temporary panel that shows up right before game start
* Entity.java – Parent class of all entity object
* Gorilla.java – entity class that holds gorilla information
* Banana.java – entity class that holds banana information
* Sun.java – entity class that holds sun information
* Explosion.java – entity class that holds explosion information
* Arrow.java – entity class that holds explosion information
* Building.java – entity class that holds building information
* InputRequest.java – deals with all inputs and labels shown in the game
* GameOver\_Panel.java – end panel shown when game is over.

It is a NetBeans project, so you can just open the project and hit run/build to play the game.

1. **Testing Report**
   1. **Testing Strategy**

We created unit tests for all of the critical classes:

GameController

Banana

Entity

Building

Gorilla

Sun

In the tests, we created unit tests for the methods that algorithmically solve cases. We used normal Equivalence class partitioning to tests our classes that use algorithmns and robust equivalence class partitioning on methods that involve inputs and variables that could be out of bounce, banana collision etc.

* 1. **Bug Report**
* Very low probability that a building would be created to small all the way on the right side of the screen.

1. **Conclusion**

For this project we successfully completed the porting process of IBM’s version of Gorillas. The original version was in QBasic and we decided to port it into Java. Some tools that we used for documentation to understand the original program is Microsoft Word and Excel, and SciTool’s Understand. To actually do our porting process we used NetBeans IDE and planned our schedule with Microsoft Project. Our main purpose was to use object oriented programming for this process and a method used for this was Test Driven Development.

A main goal of ours was to port the game almost exactly how it was originally, this included certain algorithms used in the original game. This is where we started having issues as we had trouble capturing a solid understanding of some of these algorithms. We also had trouble understanding the original source code as none of us had every worked with QBasic before, although the great amount of commenting did help. Analysis of the original program was also tough due to the source code given was not the same as the testable game given to us on D2L, this caused a decent amount of confusion.

Some lessons learned through the reengineering process of this project are the difficulties of using test-driven development, working as a team with people of different coding levels, and the difficulties of changing from functionally transformed code into a structured code.