**Reengineering Project Final Report – Nibbles.bas**

1337 haXX - John Polus, Alex Jacobs, Erik Rasmussen, Logan Brincks, Steven Karrmann

1. **Building Instruction**

* **Group Report.docx:** Group project report
* **Final Report.docx:** Final project report
* **BoundingBox.java:** Represents the location and area of an entity
* **CollisionLookup.java:** Collision detection object
* **EnumDirection.java:** Enumeration of keyboard directions
* **ICollidable.java:** Collision interface
* **IDrawable.java:** Drawn object interface
* **Nibbles.java:** Primary logical driver of program
* **NibblesGUI.java:** GUI interface of program
* **Snake.java:** Parent of snake objects
* **SnakeHead.java:** Head component of snake
* **SnakePart.java:** Body component of snake
* **Wall.java:** Level wall object
* **Level.java:** Level object
* **Entity.java:** Generic object drawable
* **Edibles.java:** Snake edible object
* **Pair.java:** (x,y) pair location object
* **Player.java:** Player class for multiplayer
* **Scoreboard.java:** Scoreboard for game
* **SoundManager.java:** Manages sound output
* **Audio/Levels/Fonts:** Various asset files for game text, font, level design, and audio files

1. Make sure you have the latest Java version installed
2. Build project using Netbeans or another Java IDE
3. Run project from NibblesGUI.java
4. Game should now be running.
5. **Testing Report**
   1. **Testing Strategy**

Our initial approach was to test the most basic functionality and build off that. Move the snake, and check that it is in fact in a new position. Eat a nibble, check that the snake grew. This is what we began with, but as we started to piece together more and more of the game, we quickly realized our basic approach wouldn’t quite cut it. In the end, we prioritized implementation and left TDD behind. This brought about its own set of issues, which we’ll expand upon in the conclusion.

* 1. **Bug Report**
* Edibles can still spawn on snakes occasionally.
* The window dimensions are static, causing problems in new environments.
* A specific sequence of inputs in the menu can cause the snake to die when unpausing the game.
* Edibles can spawn partially overlapping walls. They are still able to be eaten, but this is unintentional.
* In multiplayer, two players cannot hit a movement key at the same time.
* Holding down the cheat button to skip levels rapidly may cause the game to end randomly.
* Using the cheat button to skip levels while the game is paused will cause Sammy to die.

1. **Conclusion**

This project was broken up in five main phases: Structured Analysis, OO-Design, Test Design, Beta Testing, and Alpha Testing. By utilizing these process phases as guideposts for this project, we managed to successfully complete our reengineering of Microsoft’s Nibbles from QBasic to Java. However, despite meeting all primary design goals, there were a few issues encountered along the way and aspects of the process that could have been done better.

Regarding the process of porting Nibbles, we began with a structured analysis of the QBasic program and developed a list of requirements necessary for the port to be a success. During this phase, we spent most our time understanding the “what-and-the-why” of the program in order to ensure that the port we were going to create was as close to the original as possible. By developing a data dictionary and P-Spec table, we were able to piece together the various components of the software to use in our OO-design. Our OO-design was more-or-less focused on high-level ideas. We spent almost all our time trying to take what we obtained from our analysis and convert that into an OO architecture. By the end of this phase, we developed a general class diagram for our reengineering process to follow. There was one problem that we encountered during this phase and that was the fact our class diagram we developed was too vague. There was not enough detail in our classes and this led to problems in the implementation phase.

The test design phase was probably the most difficult as it required a massive amount of effort to develop test cases. Test-driven development was a challenge for us as it required going against many of the habits we have picked up during our years of programming. As such, this phase led to the creation of many test cases, however, we have discovered many of them were too simple or vague to be effective. Therefore, many of the unit tests and regression tests we created were not detailed enough to properly test our program.

Finally, the last two phases were beta and alpha testing. Both phases were primarily implementation phases in which we tried to match our port’s functionality to the requirements we had created originally. The only major difference between both of these phases was that the alpha phase was not feature complete and was used primarily was a core functionality test demo, and the beta phase was a playable feature complete version, albeit it has a few bugs. By the end of beta testing, we had a working, feature-complete version of Nibbles in Java.

Overall, we feel the project was a success, although there were problems. Our test cases and designs were far too simple and many were of mediocre quality, and we were not nearly rigorous enough in the development of our OO-design. We also had issues with project planning as we often worked too close to deadlines. These mistakes have put into perspective many of the potential pitfalls of managing a reengineering project. Without a good OO-design, it is very easy to end up with software that is full of highly-coupled classes, duplicate classes/methods, and bloated classes. It also makes it difficult to implement as many design decisions may not fully be represented and therefore must change late in development. Not only this, but proper test design is vital to ensure a project ends up meeting requirements. Poorly designed tests make bug testing difficult and make it extremely easy to end up with functionality that doesn’t meet desired standards. Even worse, it could result in a project that simply doesn’t work. Lastly, we feel that the biggest lesson learned is the importance of proper planning, teamwork, and communication. Without these things, it is very easy to end up with a poorly made, disjointed project that fails to meet the original design goals.