Paku Paku Reengineering Project

Fire-Breathing Rubber Duckies: Eric Weber, Evan Krug, Henry Yang, Michael Luedtke

# Team Summary

## Time Log

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Time**  **( in minutes )** | **Activities (description)** |
| Eric | **2 days 7 hours = 3300 mins** | 9/27 - 10/9 Reverse engineering. Making diagrams, reading old code.  10/09-10/20 Building UI and underlying logic  10/20-11/01 More UI building and debugging. War with IDE. |
| Evan | 100  48  68  32  53  300  257  405  157  340  226  140  303  460  120  **2 days 6 hours = 3240 mins** | 9/26 Committing original source code to repository; setting up C# classes, adding game variables  9/29 edit reengineering notes, start to plan how to convert pascal code to c# program  9/30 learning pascal, adding class files  10/07 converted c# files to Java  10/11 Ghost functionality  10/15 Proofread requirements, ghost flee/eated activity diagrams, JSON research, made Paku class a singleton  10/16 Player update diagram, game controller, jsonObjects  10/18 context diagram; JSON data, character locations, ghost classe  10/23 DFD  10/27 UI-backend communication; fixing naming conventions,  10/28 consolidating gameData class; JSON forming, score tracking  10/29 Mvc refactoring  10/30 Paku move method, wall collision check method, JSON, JUnit tests  10/31: JUnit tests , bug fixing regarding Paku and game Controller, project report, project presentation, making class diagram  11/01: Organized repository, finished unit tests for GameController, counted LOC, Project report, presentation |
| Michael | 106  55  146  57  34  19  28  122  147  82  84  47  138  46  89  188  657  107  63  166  435  **1 day 11 hours = 2100 mins** | 9/27: Creating a document of information learned from the pascal code.  9/30: Working on a mock up class diagram for the old code  10/4: Created a high level data flow diagram and lower level activity flow diagrams  10/11: Toying with classes to better understand the design.  10/14: Worked on requirements documentation  10/15: Continued requirements documentation  10/17: Finished requirements documentation  10/18: Basic class diagram mockup created and ghost movement coding started.  10/19: Main movement methods for ghost were made along with Stinky’s movement pattern. GhostState was updated to match what the program needed.  10/21: Ghost movements were completed to the map-specific variables  10/22: Experemented with a new style for the ghost’s map.  10/23: Movement prototype for the ghosts is completed.  10/27: Working on tying up loose ends in the code, added the reset methods in GameController and processed the collision detection in the GameController.  10/28: Worked on fixing bugs in the code.  10/29: adding Fruit as a valid variable in the mix, and wrapping up dot munching methods. Gave the ghosts a map instance so they could move properly.  10/30: Rewriting code with the new GameData layout. Created a few tests for the ghosts.  10/31: Finalized the Ghost class and finished the testing for each ghost, documentation.  11/02: Perfecting flee so that it matched the previous code. Rewrote some comments to match better.  11/03: Debugging the code and narrowed down one of the bugs but couldn’t exactly find it without UI help.  11/04: Bug fixing and other tasks. Fixed bug that map was copying itself too much, fixed the copier for the map, fixed the jail.  11/05: Created base implementation clientside for fruit spawning, fixed the issues with ghosts getting stuck or getting to places they cannot be, UI communication assistance, and general debugging  11/06: On call debugger, assisted Eric with finishing the rest of the project. |
| Henry | **1 day 17 hours = 2460 mins** | **9/27-10/04** We made plan for how to implement things. But there were roadblocks, so we had to ditch the old one. I was not familiar with the new solution at all, so I spent a lot of time learning from Eric.  **10/04-10/09** Reverse Engineering. Read old code, make and understand diagram. Making notes of the design details in order to replicate it in the new game.  **10/09-10/21** Worked on backend logic. Making sure we follow the OO design for our code.  **10/21-10/31** Debugging. Backend is ready to send JSON to frontend. Backend can receive certain control from frontend. Writing and running testing cases. Writing up documents. |
| **TEAM TOTAL** | **11,100 mins** | Total time needed to produce this deliverable. |

## Time Reporting Problems (all manual entry of time Shall be noted here)

|  |  |  |
| --- | --- | --- |
| **Who** | **Date** | **Problem Documentation** |
| *Michael* | *10/17* | *Missed initial punch* |
| *Michael* | *10/29* | *CSSE Hub was down so I could not punch in* |
| *Michael* | *11/02* | *Forgot Initial punch* |
| *Henry* | *10/22* | *My brain burned too much calories that I forgot to clock in. Fun Fact: Chess player burns as much calories as Marathon runner.* |

## Individual Contribution

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **LOC written** | **#tests written** | **Description of contribution** |
| Henry | 600 | 1 | Worked on Ghost, GameMovingObject, paku, gameData, Servlet.Debugged on all. |
| Michael | 1574 | 29 | Worked on Ghost, Kinky, Stinky, Hinky, Blaine, Fruit, and collision functionality of GameController. Wrote tests for Ghost, Kinky, Stinky, Hinky, Blaine, and one in GameController. |
| Evan | 800 | 10 | 2 activity diagrams, MVC architecture, Paku class, GameData class creation, Score class, Paku unit tests, GameController unit tests, Jira managing |
| Eric | 2000 | 20 | All front end works. CSS Styling, JS scripts HTML design. |
| **TEAM TOTAL** | **4746** | 41 |  |

# Project files for this Deliverable

|  |  |  |  |
| --- | --- | --- | --- |
| **File Name** | **Path** | **Purpose** | **Final Version**  **& Date** |
| readme | reengineering/docs/readme | Serves as user manual | 11/01/2019 |
| PakuJava | reengineering/src/new/PakuJava | Contains source code for new rengineered game | 11/01/2019 |
| PAKU.PAS | Reeengineering/src/original/PAKUPAKU/PAKU.PAS | Original game source code that we reengineered | 11/01/2019 |
|  |  |  |  |
|  |  |  |  |

# Process and Methods

To get everything started, we did reverse engineering with old Pascal code. We generated Contest Diagram, Data Flow Diagram, Activity Diagram. Also, we did study the old source code in order to fully understand it. After each member had gain enough knowledge about the old game, we converted the old game logic to the new Object-Oriented style design. After the design for the new system is done, we began to implement the system. The team was divided into two parts. One for frontend, one for backend. We decided the backend to be in Java and will talk to the frontend via JSON object. The frontend is in JS and can be deployed via internet if hosted on server.

We have fully utilized the OO feature of Java when we code the game. The new system is designed according to the MVC pattern. By doing this, each component can have a clearer task assignment that lowers the coupling within the system. We implemented Singleton pattern to gameData, Paku and ghosts to ensure that only one instance is created. We designed our class according to the inheritance standard to increase the code reuse ability that lowers the code cohesion inside the system.

The system was coded with descriptive-coding concepts in mind to make the code as readable as possible.

For now, the code is hosted locally on TomCat server.

# Reengineered System Summary

## Old vs. New Summary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Name** | **Release date** | **Platform** | **Language** | **LOC** | **#Functions** |
| **Old** | PakuPaku | 11/9/2011 | MS-DOS | Pascal | 2263 | 40 |
| **New** | PakuJava | 11/01/2019 | Win10 | Java | 3700 | 218 |

## Major Changes

PakuPaku was recreated as a Java program using the Tomcat9 system for UI.

PakuPaku can no longer receive input from a joystick.

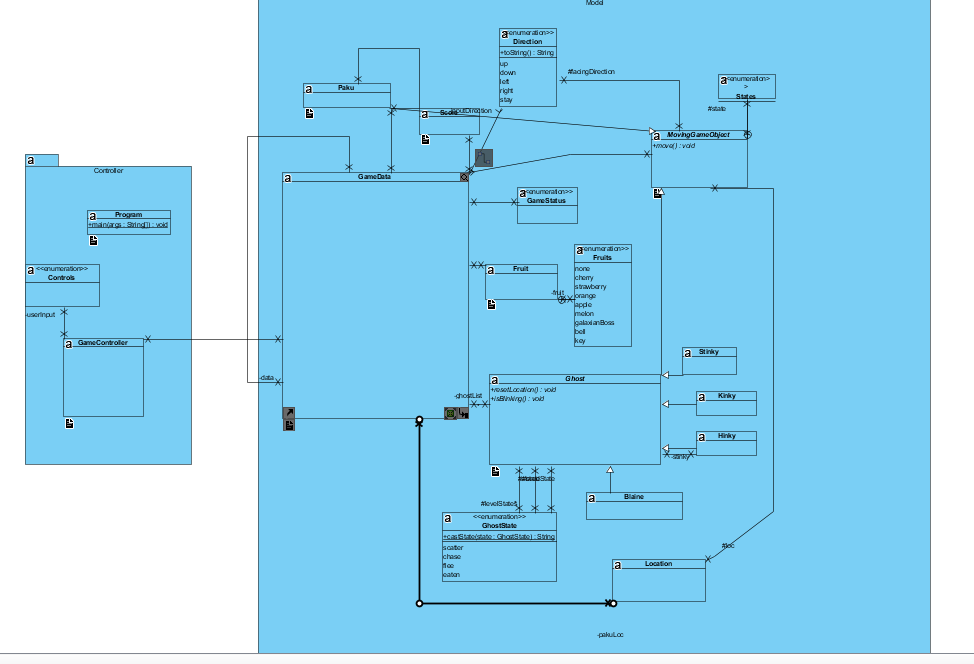
PakuPaku no longer uses command line codes.

PakuPaku would break at high lever due to memory shortage. That’s no longer the case.

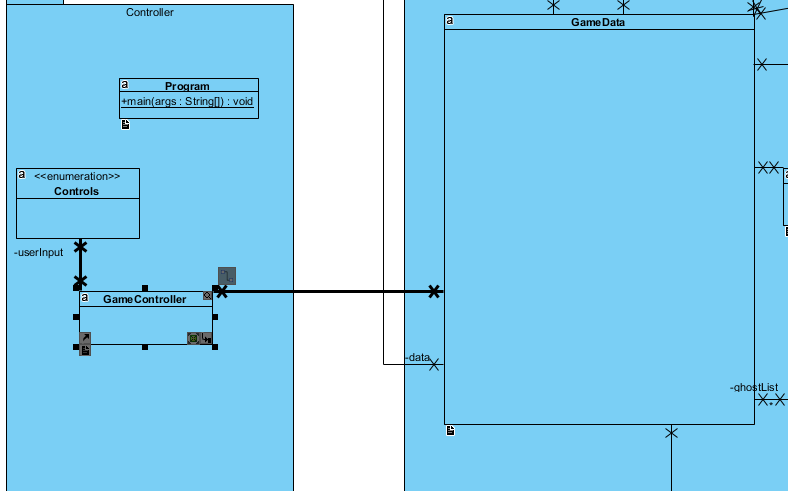
PakuPaku now runs on anything that has a compatible web browser.

## Reengineered System Design

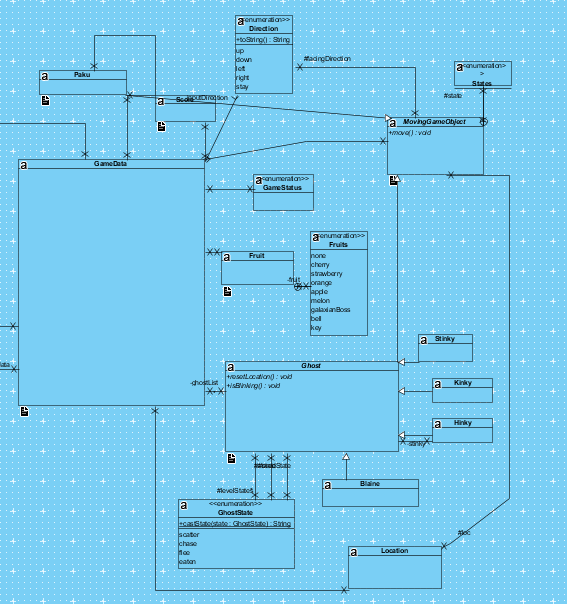
System Overview



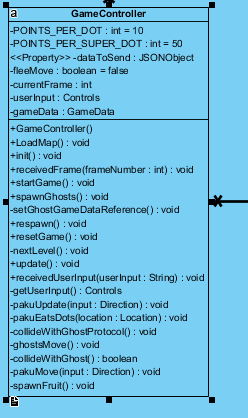
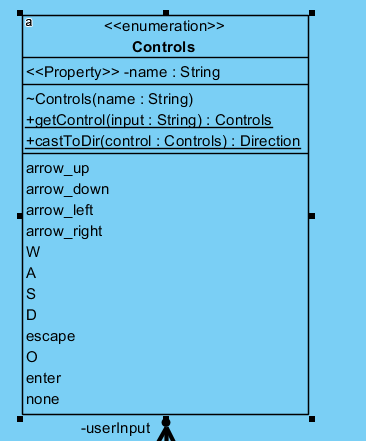
Controller Package, with reference to GameData

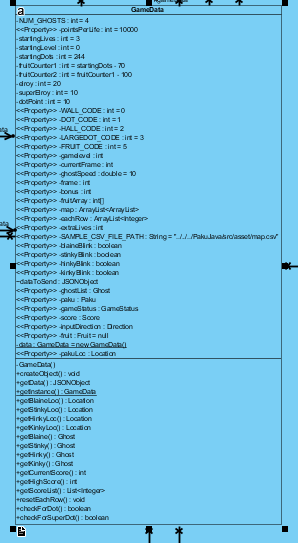


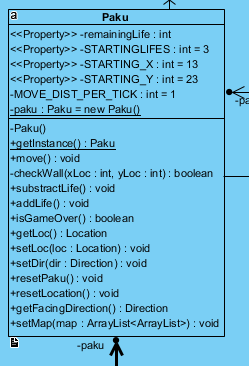
Model Package

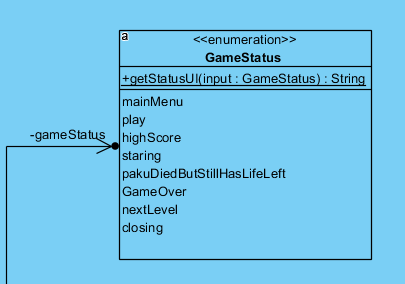
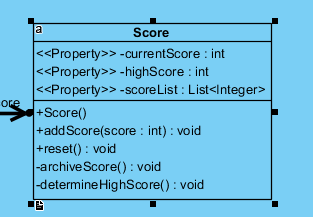


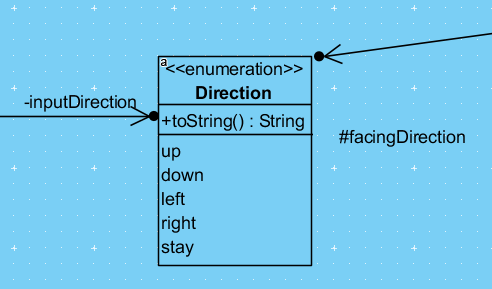
Individual Classes

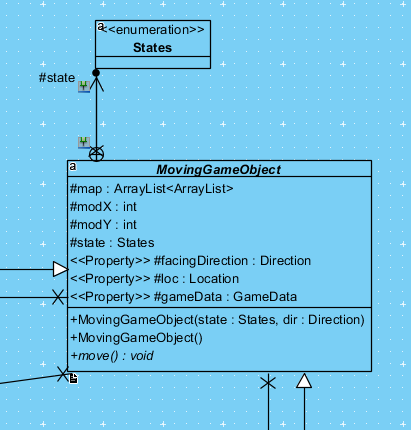
 

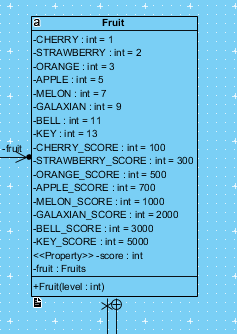


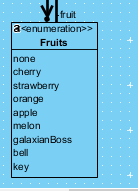


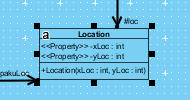


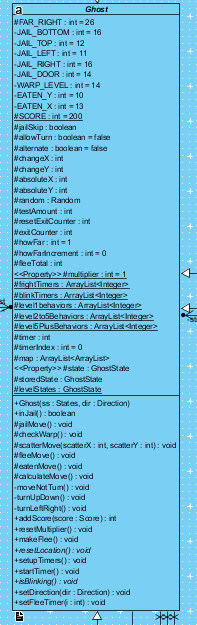


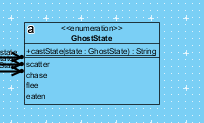


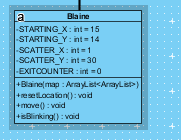


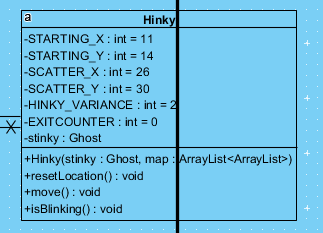


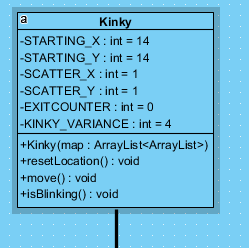


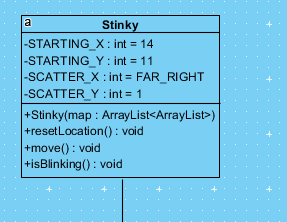












## Program Metrics

Backend LOC ~ 2450

CSS: ~ 900

JS ~ 350

Total LOC ~ 3700

5.80 Average Efferent Couplings  
8.40 Average Afferent Couplings

2.36 Average Cyclomatic Complexity

1.97 Average Design Complexity

75.88% Attribute Hiding Factor

# Lessons Learned

Evan: The major challenge of this project was finding meeting times and planning out tasks and subtask due dates. We tried to keep in contact via Microsoft Teams. Another challenge was interpreting PASCAL code and modeling it with activity diagrams. Finally, front end work was a major challenge because Eric was the only one with experience with it. We should have chosen a project plan that was more compatible with all our capabilities. The most important lesson I learned was the importance of time management and task planning. It would have helped greatly if we had created a detailed class diagram ahead of time so that we were all on the same page regarding design.

Eric:

Michael: The major challenges for me were the meeting times and time planning. We tried to use Microsoft teams, but there were some issues early on with notifications not being sent that did hurt a bit. Pascal was also a challenge, as the code was very odd at times. I had to review it multiple times until I got my parts as close as I could, as the lack of documentation wasn’t super helpful. The most Important lesson for me would be how to time out the reengineering flow. The longer reengineering time did not help us too much.

Henry: I say we studied too much about the old program. The reverse engineering should start at requirement level for this project since the game is too old. We designed the new system too good that it took more work to complete than a class project can handle. We only have one people working on frontend so that’s a lot of work for Eric. Also, I am appreciated that him worked 12 hours last night until 7 AM tried to get things to work.

For this reengineering project, the big bang solution does not fit. I propose that next time we can try partial method.

For now, I feel extremely confident coding in Java in a descriptive way. I got to learn set up a frontend with backend and was able to get them to talk to each other. I have learned more about how JavaScript works. My knowledge from the Mobile App class came in to help. Coding this project has helped me gain more experience with dealing with JSON object. Building the project that can be put online easily with a migration to the actual server feels amazing.

For the project itself, it was not completed but I still satisfied with all the things I learned from each process.