# Title

Reengineering Original Space Invaders Arcade Game to Modern C# Desktop Application

# Team

Nicholas Mueller

Richard Bergemann

Jared Charneski

Caylan Laundrie

Cole Reuter

# Team Summary

## Time Log

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Time**  **( in minutes )** | **Activities (description)** |
| Nicholas Mueller | 1094 | Hand drew the sprite for the arcade machine and joystick, and bullets. Wrote game logic code. Hit detection on entities, score updating and writing to file, animations to give arcade box feel, and state management for what to display. |
| Richard Bergemann | 602 | Focused on all logic and functionality of things outside of the game itself. This included the process of drawing the coins and coin pile, as well as the logic behind grabbing coins, dropping coins, depositing coins, coin type error connection, as well as coin returning. |
| Jared Charneski | 554 | Hand drew the sprites for the quarter, dime, and coin piles. Wrote code to make the entire start screen |
| Caylan Laundrie | 664 | Wrote formal requirements, coded aliens shooting back at the player, made bullets bidirectional. |
| Cole Reuter | 440 | Worked on shooting, collision detection, moving left and right with arrow keys, health on shields, class diagram, and PowerPoint. |
| **TEAM TOTAL** | **3354** | Total time needed to produce this deliverable. |

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

|  |  |  |
| --- | --- | --- |
| **Who** | **Date** | **Problem Documentation** |
| *Caylan* | *10/15* | *I did not use a computer in this meeting* |
| *Caylan* | *10/21* | *These were breaks between classes* |
| *Caylan* | *10/31* | *I should have started CSSE hub in the morning but got caught up in the problem.* |
| *Caylan* | *10/31 evening* | *Started the timer after getting to the meeting and adjusted start time* |
| *Cole* | *10/30* | *Forgot to start the log time* |
| *Jared* | *10/22* | *Forgot to start the log time* |
| *Jared* | *10/30* | *Forgot to start the log time* |
| *Jared* | *10/30* | *Forgot to start the log time* |
| *Jared* | *10/31* | *Forgot to start the log time* |
| *Nick* | *10/20* | *Left lab without stopping* |
| *Nick* | *10/21* | *Updated small file during class* |
| *Nick* | *10/21* | *Forgot to start the log time* |
| *Nick* | *10/21* | *Forgot to start the log time* |
| *Nick* | *10/21* | *Breaks between club meeting* |
| *Nick* | *10/22* | *Work with Jared and Richard – was in dorm meeting room and I wasn’t on my laptop (Verbal help)* |
| *Nick* | *10/30* | *Forgot to start the log time – group meeting this day* |
| *Nick* | *10/30* | *Forgot to start the log time* |
| *Nick* | *10/31* | *Forgot to punch in during meeting* |
| *Nick* | *10/31* | *Worked on after getting back from lab with no internet connection* |
| *Richie* | *10/15* | *Forgot to start the log time* |
| *Richie* | *10/30* | *Forgot to start the log time* |
| *Richie* | *10/30* | *Forgot to start the log time* |
| *Richie* | *10/31* | *Forgot to start the log time* |

## Individual Contribution

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **LOC written** | **#tests written** | **Description of contribution** |
| Nicholas Mueller | 1285 | 11 | Game Logic, Game states and displaying and saving scores, hit detection for entities, health, level progression, arcade box image (drawn by hand) |
| Richard Bergemann | 280 | 2 | Did all coin logic, and drawing for coins in gameboxform |
| Jared Charneski | 127 |  | Worked on start screen, coin, created sprites for coins (hand drawn) |
| Caylan Laundrie | 94 |  | Modified the Bullet class, programmed aliens shooting and targeting the player |
| Cole Reuter | 217 |  | Worked on shooting, collision detection, arrow key movement, and health on shields. |
| **TEAM TOTAL** | 2003 |  |  |

# Project files for this Deliverable

|  |  |  |  |
| --- | --- | --- | --- |
| **File Name** | **Path** | **Purpose** | **Final Version**  **& Date** |
| Re-Engineering Legacy system Document.docx | Reengineering > docs | Write up Nick did for understanding how the legacy system worked in the Assembly code structure. | 10/21 |
| Reengineering Final ProjectReport.docx | Reengineering > docs | This project report. | 11/1 |
| Readme.txt | Reengineering > docs | File to explain how to run it | 11/1 |
| SpaceInvaders.asm | Reengineering > src > original | Assembly code file of legacy SpaceInvaders game |  |
| All image files in this file location | Reengineering > src > new > Image Assets | Folder that holds some images that were used for development | (see sln date) |
| SpaceInvadersButBetter.sln | Reengineering > src > new > SpaceInvadersButBetter | Solution file for Visual Studio | 11/1 |
| All files in this location | Reengineering > src > new > SpaceInvadersButBetter  > SpaceInvadersButBetter | Project files for the program split into folders.  Core (main classes), Views (screen views) | (see sln date) |
| SpaceInvadersButBetter.exe | Reengineering > src > new > SpaceInvadersButBetter  > SpaceInvadersButBetter  > bin > Debug | Executable file for the game (debug folder) | 11/1 |

# Process and Methods

The first thing we did was sit down and read the whole documentation. We created a context diagram and a diagram 0 for the system which helped us see what processes happened. Nick volunteered to read through the entire Assembly code and documentation to get a grasp on what was going on and then help explain it to the group while they started other tasks. We used this information to model out system off of but making changes are we are trying to improve it by making it in a new language. See the document titled “Re-Engineering Legacy System Document” for details in explaining the inner workings of the old legacy system. This describes the game description, general structure, game objectives, and other features of the assembly code. This information in this document was used as a guideline for some basic necessities for the new system.

# Reengineered System Summary

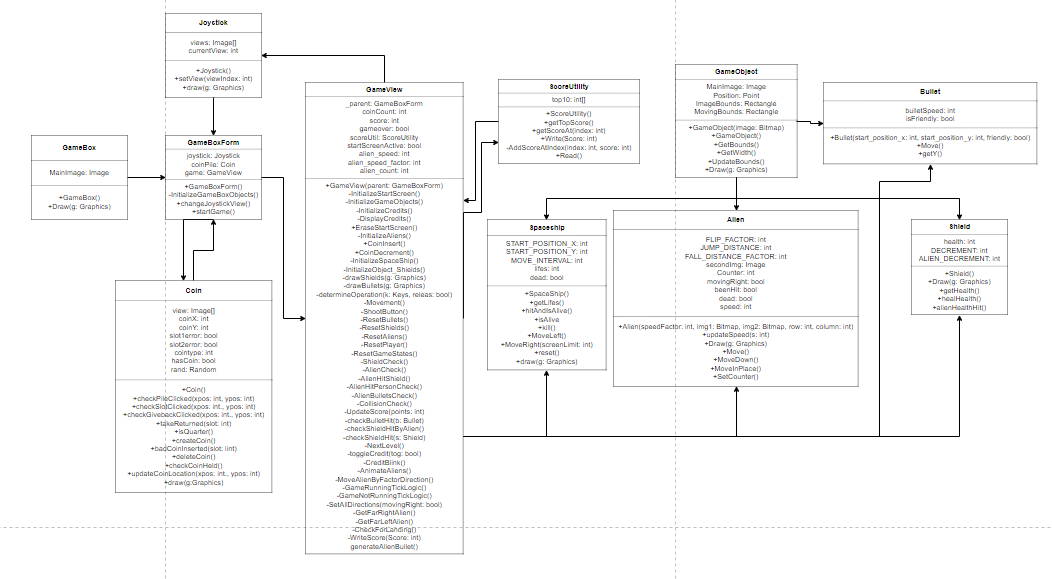
## Old vs. New Summary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Name** | **Release date** | **Platform** | **Language** | **LOC** | **#Functions** |
| **Old** | Space Invaders (Arcade) | 1978 | 8080 CPU | Assembly | 4639 | 169 named jumps  (numerus number of random jumps not named) |
| **New** | Space Invaders but Better | 2019 | Windows | C# | 2019 | 109 functions including constructors |

## Major Changes

* Now allows for faster shooting
* Infinite levels
* Works on modern computers running windows, you don’t need a dedicated arcade machine

## Reengineered System Design



## Program Metrics

Lines of Code: 2019 (includes solution and tests)

Maintainability Indices by Class: (out of 100, higher is better)

GameBox: 92

GameBoxForm: 67

GameForm: 70

GameObject: 77

GameView: 59

Joystick: 82

Program(start driver): 81

Alien: 73

Bullet: 77

Coin: 72

ScoreUtil: 67

Shield: 92

Spaceship: 78

Cyclomatic Complexity (code paths, lower is better) by Class:

GameBox: 2

GameBoxForm: 19

GameForm: 17

GameObject: 6

GameView: 166

Joystick: 3

Program (start driver): 1

Alien: 10

Bullet: 5

Coin: 40

ScoreUtil: 14

Shield: 5

Spaceship: 13

Depth of Inheritance (different classes that inherit from one another, lower better) by Class:

GameBox: 1

GameBoxForm: 7

GameForm: 7

GameObject: 1

GameView: 7

Joystick: 1

Program (start driver): 1

Alien: 2

Bullet: 2

Coin: 1

ScoreUtil: 1

Shield: 2

Spaceship: 2

Class Coupling (number of classes referenced, lower is better) by Class:

GameBox: 3

GameBoxForm: 20

GameForm: 21

GameObject: 5

GameView: 40

Joystick: 3

Program (start driver): 3

Alien: 6

Bullet: 4

Coin: 8

ScoreUtil: 6

Shield: 3

Spaceship: 5

After evaluating our program with these metrics, we notice that we could have split the Views into individual controllers for different game elements. This would decrease class coupling as our GameView and GameBoxForm had relatively high coupling compared to other classes. It would also decrease the cyclomatic complexity and increase maintainability of the project.

# Lessons Learned

**Cole** – From this program I learned how to properly reengineer a software system. One major challenge I faced was trying to get the shields to get holes in the edges when hit because it isn’t possible using the images we have. To fix this we eliminated the need for the shields to physically show damage and instead added a health score on top of the shields to display how much longer they would last. The most important lesson I personally learned was the importance of going through the whole process of reengineering and not trying to skip to the end and just start coding right away.

**Caylan** – Overall I think the most challenging parts of this project include identifying complex patterns in the original Assembly code, planning the new software, and setting up the base for the rest of the game. The original code was interpreted by those of us who had the most recent experience with CAOS, the logic and functionality of the game were broken into somewhat organized chunks, and it was decided that a C# game-view would be used as the basis of the new game while visually imitating an arcade case. The most important lesson I learned is that the planned structure of a project may not be static and that I cannot dwell negatively on changes to the structural plan. I also miss Team Foundation Server dearly.

**Jared** – For this program I learned how to reengineer a software system. However, the biggest thing I learned is how to work with visual objects in C#. In C#, usually I worked with the model-view controller pattern and use JavaScript for visual effects. I made the start screen, one of the issues I ran into when making the start screen was how to not make the game objects present while the start screen is active. I was able to solve this issue by displaying some of the objects and stopping the game timer. The most difficult non-programming thing was getting source tree to work. It would not work on my laptop, so I had to start emailing my code changes to Nick to have him push it.

**Nick** – This program was a challenge for me mainly because it was the first game I have created. In all my course work and outside class projects, I have never done a game. It was a learning experience for sure. Some challenges were managing were everything was by as efficiently as possible. Given more time, the efficiency would have been much better as there are lots of looping threw the arrays for the aliens that sometimes bogs the system down on a low spec PC. Another hard thing was understanding the Assembly code for the old system. I spend a long time reading each line and all the documentation that went with it to understand how it worked. I can confidently say that while I am not the best at writing Assembly code, I could explain much of the in’s and out’s of how the code works in the old system. It was very neat to see how they managed to do all the things they did and the weird things they did as well. I think the most important thing I learned was that having a design first helps as we started to code right away and made things challenging to bring everything together.

**Richie –** This project was the first time I used C# in a game setting. This caused chalenges for me with drawing objects by pixel, as well as understanding how all of the components fit together. We also had troubles with SourceTree throughout the project, which caused us to think quickly on fixes. We solved these issues by doing most of the coding on the project together. Some of the members of the group were much more experienced than me in some aspects of C#/SourceTree as well as game knowledge, and well as vice versa. These get togethers allowed us to share this knowledge with each other, allowing for a ton of new skill development. The most important lesson this project taught me was that going to your teammates for help can be extremely beneficial. I could not have done what I did without the help of them and am much more well versed in the language as well as how testing works in C#.