Tic-Tac-Toe Game – Final Report

Note: This report will primarily focus more on the GUI aspects of design rather than other aspects that have already been implemented in the previous project, such as AI.

Section 1: Team Information:

Team Members:

- Grant Hruzek (Team Lead)
- Chris Padilla
- Ethan Wright
- Taylor Incorvia

Team Member:	% Contribution:
Grant Hruzek	25%
Chris Padilla	25%
Ethan Wright	25%
Taylor Incorvia	25%

Section 2: Problem Statement and Significance:

The purpose of this project is to work and further improve on our version of a 3-dimensional 4x4x4 game of tic-tac-toe and thoroughly test its functionality as an application. Our team of developers has been working endlessly to ensure the user gains the best experience possible when playing this game.

While this project is very closely related to the previous, they do differ in their goals. The goal of the previous project was to develop a working game and AI that could effectively and efficiently keep the game moving. This project, however, is focused on developing a convenient and easy-to-use environment in which to play said game. As a group, we were able to develop, test and deliver a working product while managing a tight deadline and quickly gaining the skills necessary to be successful. The goal was to manage many different moving parts of project development.

Not only was this good practice in program development and keeping in mind the concept of computer human interaction, it is now also a fun game that anyone can run and enjoy, and whether they can open a terminal or not, the intent is to provide players with a sense of pride and accomplishment for executing and playing a fun game of tic-tac-toe.

Section 3: Restrictions and Limitations:

- Memory issues are the largest concern, since hard limiting the AI to a certain depth based on our test with the TAMU compute servers, we are invariably allowing for issues to occur when other users attempt to run our program on more devices that don't share the same level of available memory (ex: raspberry pi).
- There are a few known combination of moves that can currently beat the AI, such as if you get into a situation where you have two possible winning lines, the AI will have to pick to block one of these and leave the other open for you to win. For this project we only emphasized effort on the GUI portion of the project as required by rubric.
- Windows Forms was to be used as our GUI interface, but limited us to its style abilities and restrictions. For example, Windows Forms does not support CSS, and does not support transparent GUI elements on-top of background images, a major part of what made our GUI-1 look so clean and aesthetic, so we had to compromise our more aesthetic design in JavaFX (GUI-1) for an easier to navigate and easier to use design in C# (WinForms).

Section 4: Overall Development Approach:

During the first lab meeting for the first GUI, the team considered and analyzed its options and decided to assign smaller parts to individual members. We decided to code the first part in JavaFX because it gives users a nice, modern feel of the game. The risky but ultimate consensus was to let each person do their own part for the time allotted and meet up a couple of days before each of the pre-determined deadlines set by the team lead. The team began by dividing the project into pieces that each member could complete separately. For the first deadline, Grant created the board and game flow while Taylor managed score-keeping logic. Christopher and Ethan during this time worked on testing the GUI elements, and implementing small, but important, visual improvements to the GUI. It was important to start development early for JavaFX because we had minimal knowledge of JavaFX thereof. During the development process, we kept in mind to try to write highly portable code so that the development of GUI-2 would be easier in the future and the porting of code from Java to C# would be more straightforward.

The final few days of development on GUI-1 were spent on integrating all of the parts with one another. The front-end and back-end coders carefully kept their code agile, object-oriented and mostly open-ended for assimilation with the user interface. Functions and methods such as convert_to_vector(), which converts a numeric cell to vector notation, allowed for smoother integration with the UI programmer's code. The score-keeping and splash screen programmer also tightly conferred with the UI coder to ensure proper linkage as we developed towards a final product.

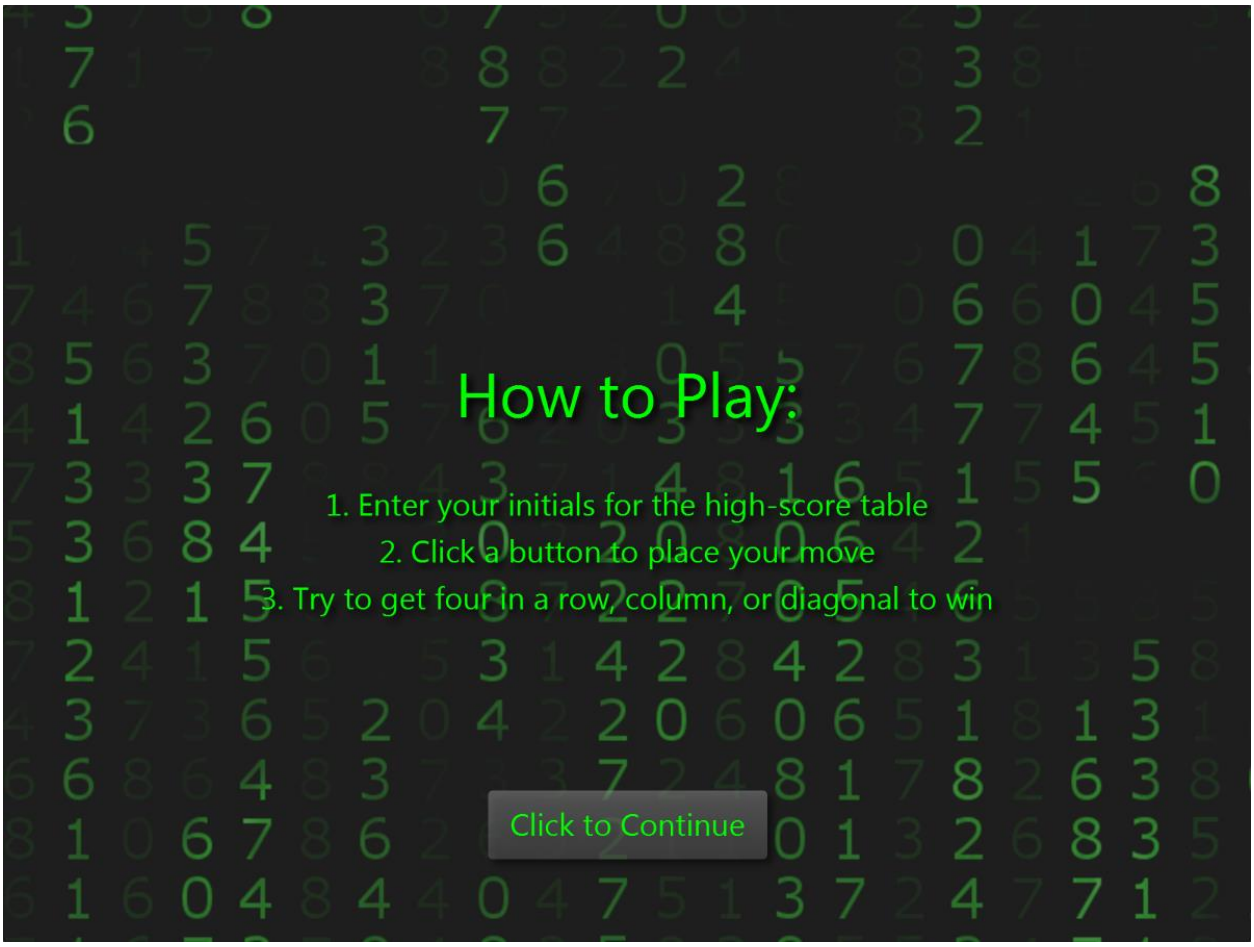
The second important lab meeting was where we were assigned the task to port over our project's current iteration in Java to the C# programming language using Winforms. Ethan was our most experienced member using the language, and he therefore handled most of the font-end duties. The rest of the group worked on the back-end, which was essentially porting over Java code to C# code. We set aside several meeting times through-out the week in order to plan development strategies and talk about the overall design and any improvements that could be made to the current iteration. Thanks to our highly portable design, the porting process of the back-end from Java to C# only took a few hours, leaving us with learning to implement the C# GUI in WinForms. From there the GUI/front-end was created, and linked to the back-end. Testing was done by the entire team to ensure a smooth and seamless game-play experience. From there we set up the project for delivery along with the final report, burn-down charts, and any other deliverables regarding the project's documentation.

Section 5: Sample Run:

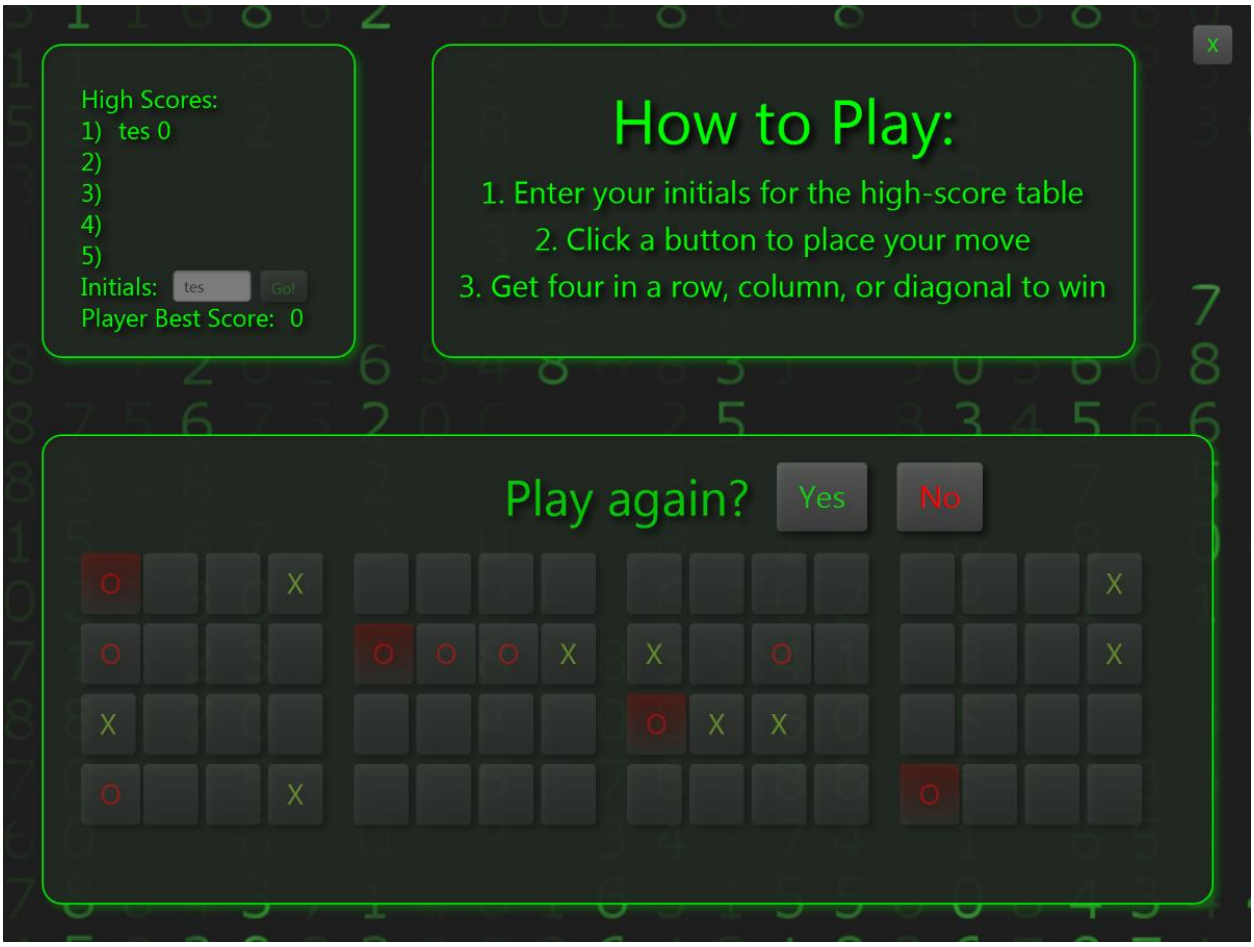
Below is a Sample Run of GUI-1:

Splash Screen:

Instructions Page:

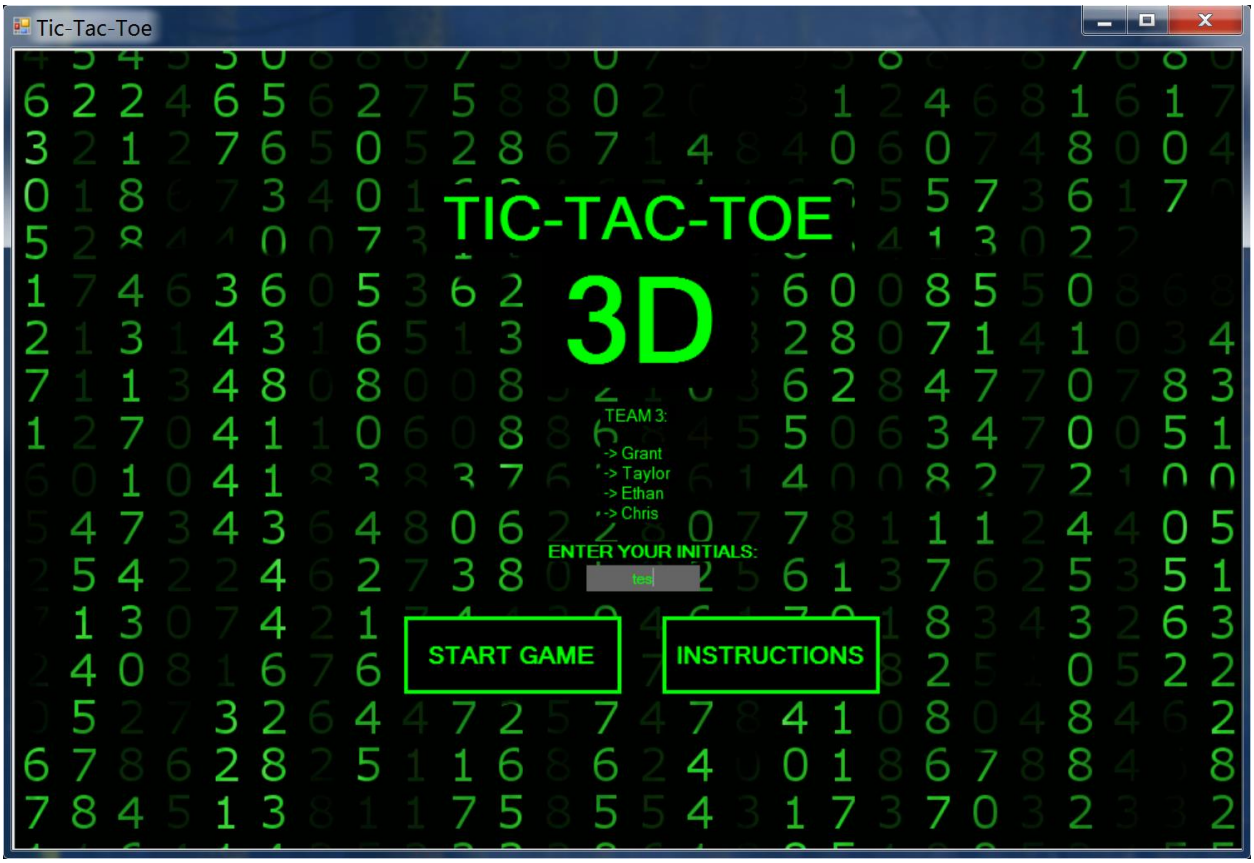


Game Page:

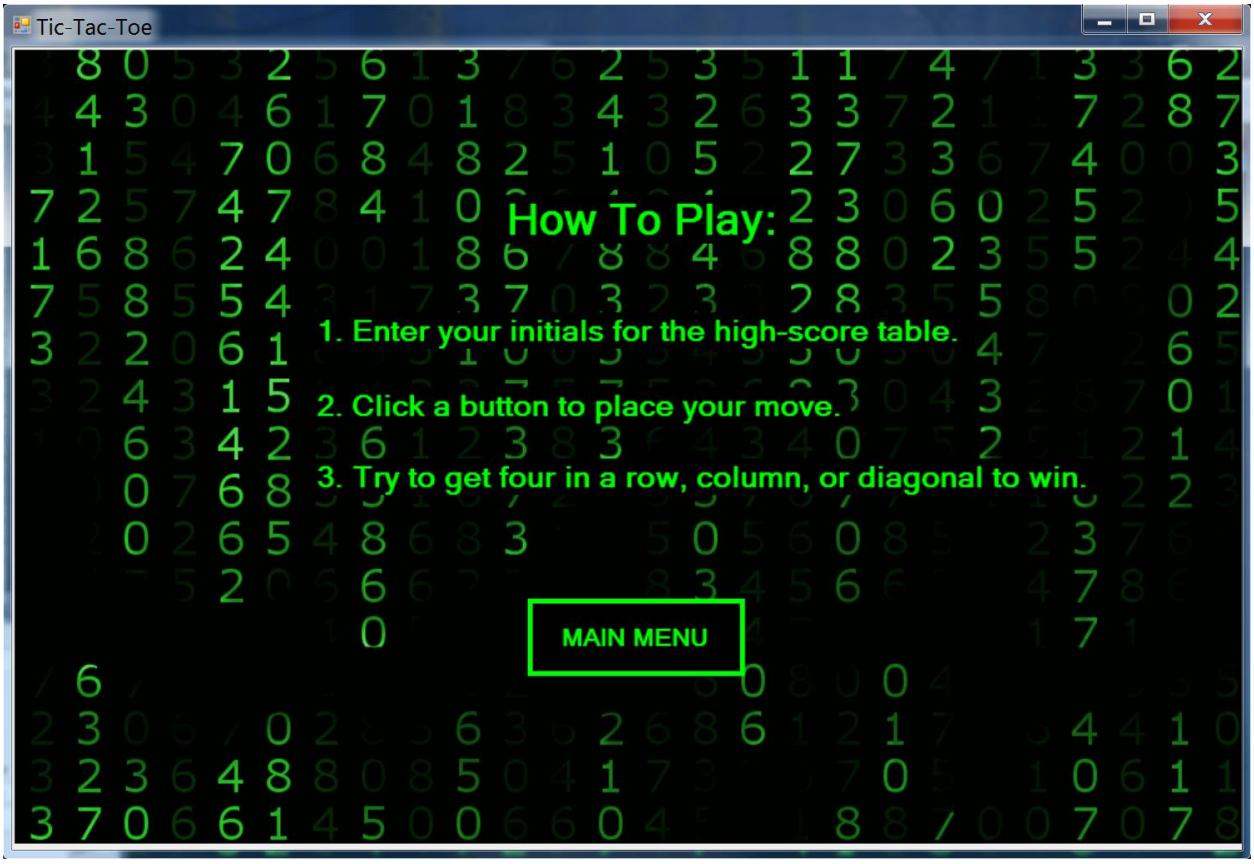

High Scores:
1) tes 0
2)
3)
4)
5)
Initials: tes Go!
Player Best Score: 0
How to Play:
1. Enter your initials for the high-score table
2. Click a button to place your move
3. Get four in a row, column, or diagonal to win
X
Play again?
Yes
No

Below is a Sample Run of GUI-2:

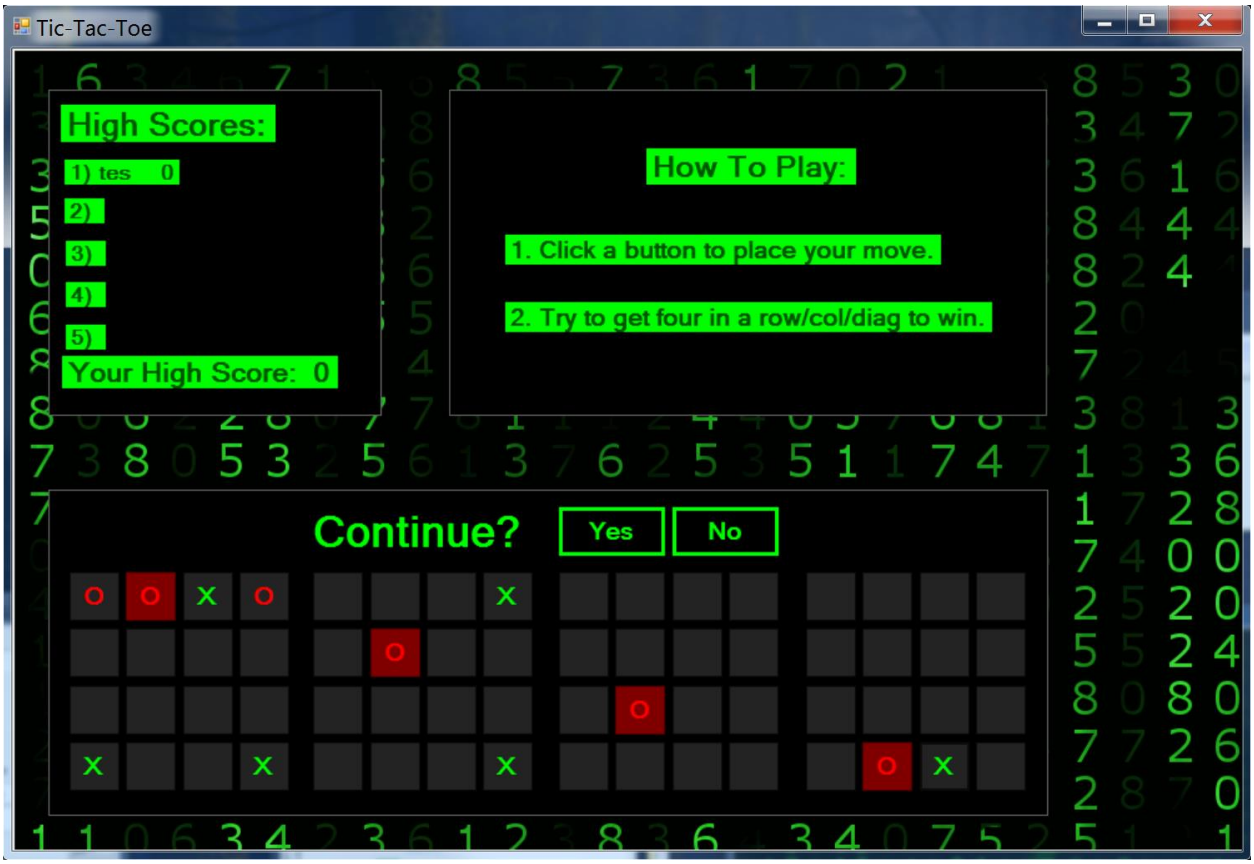
Splash Screen:



Instructions Page:



Game Page:



Section 6: Results, Analysis, Design Changes, Difficulties and Solutions:

Results

The result of our project is a fully working 3D Tic-Tac-Toe game, providing an easy to read and visually pleasing GUI environment, incorporated with the suggested improvements provided by our users.

Analysis

We had a group of diverse non Computer-Science students analyze the initial JavaFX implementation. While most of the feedback was generally positive (especially after

having them use the less-than-ideal ncurses), we were still made aware of some improvements that could be made. These included, but were not limited to, having the player and AI moves stand out more, and minimizing external, unnecessary distractions to make actual game-play contrast with the background better. These peer-evaluations are located a few sub-sections below.

Design Changes

After having non-programmer critics play and analyze the first JavaFX game (GUI-1), we decided to make a few changes to our design before porting over the project to C#, such as making the AI moves more easily readable by the human eye so that the player does not lose the game solely because they didn't see a move placed somewhere. This and having fewer distractions was crucial for the player to have the best experience when executing and playing this game, thus we decided to go with opaque backdrops for the high-score and game-board elements so that it was less distracting for the user and easier to play. These were the major improvements we gathered from our peer reviews from GUI-1 and implemented in the current iteration of our project (GUI-2).

Difficulties

We encountered some difficulties at first for the preliminary JavaFX GUI when setting up the different scenes and being able to develop smooth transitions between them. We kept this in mind before starting our project in C#, so we could implement all of the functionality without ending up with confusing spaghetti code.

Solutions

We broke both the Java and C# the code into smaller helper functions that did a lot of the messy, repetitive work for us; this not only simplified our implementation but also helped us make more sense of each specific task we were trying to accomplish and helped keep the code base manageable.

Evaluations to consider for designing GUI-1:

For the Ncurses evaluation we used a basic numeric scale shown in lecture to objectify qualitative feedback regarding the GUI's navigation, overall ease of understanding, and play-ability. A high score of 5/5 denotes "of high quality or ease" whereas a low score of 1/5 denotes "of poor quality or difficulty."

Peer Review #1:

- Ncurses GUI made it easy to navigate the game screen: ***4/5***
- Ncurses GUI made it easy to understand the instructions and play the game: ***5/5***
- Ncurses GUI overall aesthetic: ***4/5***
- Problems overall:
    - It was confusing to see where the other player had moved and I kept losing track of my own moves as well as his (the AI's).

Peer Review #2:

- Ncurses GUI made it easy to navigate the game screen: ***4/5***
- Ncurses GUI made it easy to understand the instructions and play the game: ***3/5***
- Ncurses GUI overall aesthetic: ***2/5***
- Problems overall:
    - The game screen was very hard to read and understand, it was confusing to tell which move went where because of the lack of color. I did not understand at first how the game worked but I eventually caught on.

Peer Review #3:

- Ncurses GUI made it easy to navigate the game screen: ***3/5***
- Ncurses GUI made it easy to understand the instructions and play the game: ***3/5***
- Ncurses GUI overall aesthetic: ***5/5***
- Problems overall:
    - It was different than any other games that I had played so it was confusing at first, the game board did not help in clarifying what was going on.

Remarks

With these three Peer Reviews, it is apparent that the Ncurses GUI suffers from a lack of aesthetic, clear navigability, and clearness of instruction. GUI-1 paid respects to these insights by taking a step forward in the right direction by making a pleasing interactive aesthetic with the streamlined color layout and sleek buttons while offering straightforward navigability through a few screen-transitions and direct access to the game instructions at all times during game-play.

Evaluations to consider for designing GUI-2:

For the GUI-1 evaluation we used a basic numeric scale shown in lecture to objectify qualitative feedback regarding the GUI's navigation, overall ease of understanding, and play-ability. A high score of 5/5 denotes "of high quality or ease" whereas a low score of 1/5 denotes "of poor quality or difficulty."

Peer Review #1:

- GUI-1 made it easy to navigate the game screen: ***5/5***
- GUI-1 made it easy to understand the instructions and play the game: ***5/5***
- GUI-1 overall aesthetic: ***4/5***
- Problems overall:
    - "The text-field is too small and the player/AI pieces need to standout more" - Peer #1

Peer Review #2:

- GUI-1 made it easy to navigate the game screen: ***4/5***
- GUI-1 made it easy to understand the instructions and play the game: ***3/5***
- GUI-1 overall aesthetic: ***3/5***
- Problems overall:
    - "The player/AI pieces definitely need to stand out more either by color or by depth. The most important part of a visual element should always stand out the strongest to the user and should be the first thing to catch their eye." - Peer #2

Peer Review #3:

- GUI-1 made it easy to navigate the game screen: ***4/5***
- GUI-1 made it easy to understand the instructions and play the game: ***5/5***
- GUI-1 overall aesthetic: ***5/5***
- Problems overall:
    - "The animated background can be slightly distracting and uncomfortable to my eyes when I am focusing on the game board trying to beat the AI. I am prone to seizures so my input and evaluation might be slightly biased or non-representative of the general population." - Peer #3

Remarks

With these three Peer Reviews, it is apparent that the GUI-1 offers easy navigability and straightforward usability while offering a clean aesthetic. GUI-2 paid respects to these insights by taking a step forward in the right direction by giving the player/AI pieces a bright, bold, color so that they clearly stand-out for the user to see. The background container elements behind the game-board, highscore-board, and also the instruction-board, were also all set to opaque so the animated background would not be visible through the GUI elements so the user would not be distracted during game-play and there would be less strain on the player's eyes.

Section 7: Conclusions:

Over the course of the past several weeks, we were able to work together to reach the common goal of having a more presentable implementation of the game (superior to ncurses). The development of more sophisticated user interfaces is a very important toolset to posses in the discipline of Computer Science. We implemented the first GUI in Java while keeping in mind Agile methodologies and later were able to smoothly transition into and create the same product in C#.

The use of different packages meant we had to learn quickly and adapt, which therefore has made us better programmers who consider the needs of the consumer. Even if not all of us are going specifically into game development, we can use what we've learned from this project to contribute to the development of a wide variety of user interfaces, and this was still a very good exercise the way we may have to end up working in the real world. It's always good to have all kinds of individual knowledge, but the important thing is that we were able to make use of this knowledge by committing to something as a group and learned that success is made up of the bringing of ideas and work initiatives to achieve something together. In the end, we are proud to present our project to the class, and the faculty here at Texas A&M.

Section 8: Future Improvements

- In terms of future improvements, we could implement a 3-D model of the game board. This would help the user in their efforts to understand the positions of each individual move on the board as well as better viewing of possible winning lines.
- In the future we could implement different levels of difficulty for the AI gameplay. This would allow users to have a more pleasant experience of playing the game if they were still trying to understand the concept of the different winning lines but could not get the hang of it, as the AI would quickly beat them.
- Adding multiplayer gameplay would allow two people to play against each other and can in turn make the game more fun (either online or on the same window).

Section 9: How to Run the Game

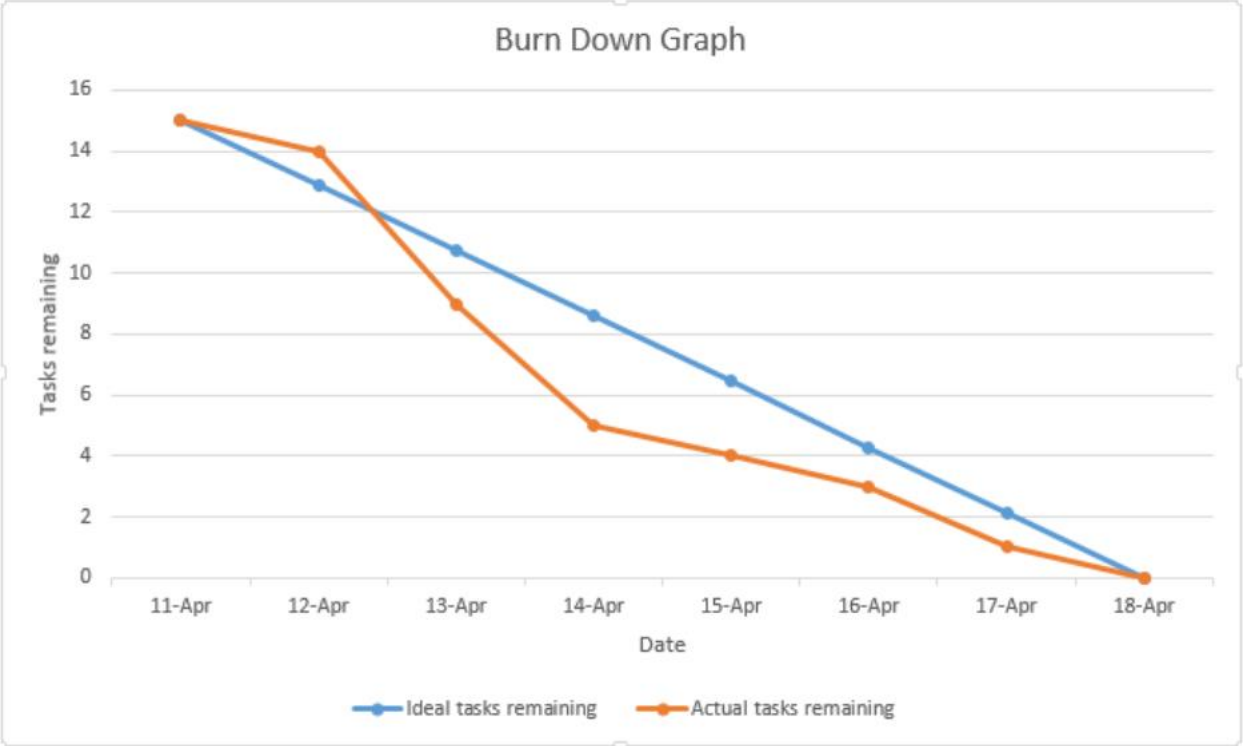
In order to run the game, the user can simply double click on the provided “GUI-2” executable file.

Section 10: Burn-Down Charts, Burn-Down Lists, & Dev-Logs

GUI-1 Burn-list, Burn-chart, and Devlog:

GUI-1:

Task:	Expected Time Needed:	Time Used:	Daily Progress:	Overall Progress:
Discussed GUI Platform	2 hours	1 hour	100%	2.5%
Laidout code structure	1 hour	0.5 hour	85%	15%
Implement GUI	4 hours	2 hours	100%	5%
Implement Backend	2 hours	1 hour	100%	10%
Tweak & finish GUI	1 hours	5 hours	75%	18%
Get Peer Reviews	2 hours	1 hour	60%	23%
Write Design Doc	2 hours	3 hours	100%	25%
Prepare for Submission	1 hour	1 hour	100%	35%



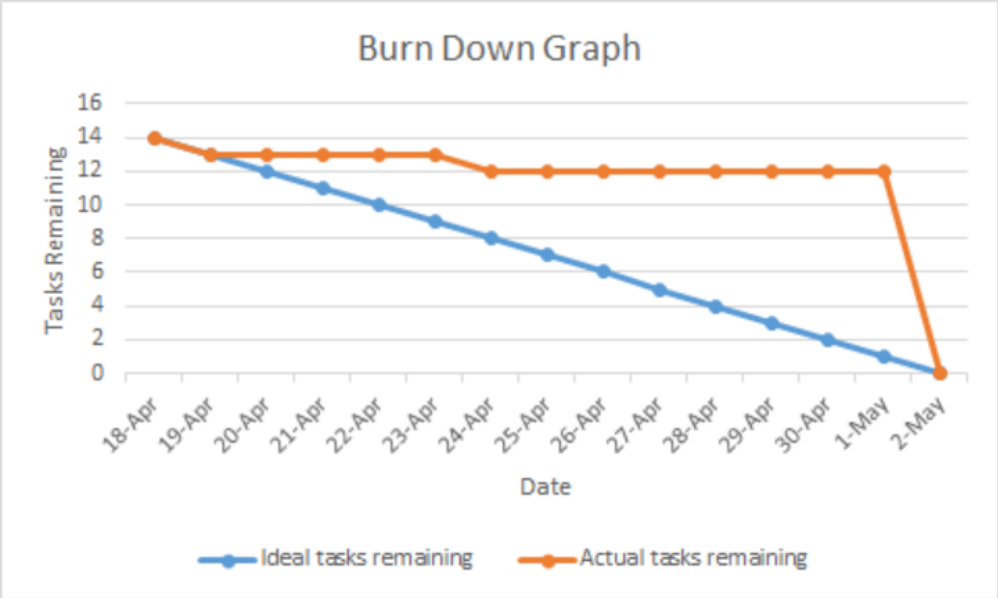
Development Log for GUI-1 (JavaFX, Java):

Type:	Feature/Requirement:	STATUS:	Date:	Name / Details:
ALL	GUI	PASS	04/11/18	Grant/Chris
ALL	Back End	PASS	04/13/18	Grant/Taylor
ALL	AI	PASS	04/13/18	Grant
ALL	Get testimonials	PASS	04/16/18	Ethan
ALL	Design Document	PASS	04/17/18	Team
ALL	All functions <24 lines?	PASS	04/16/18	Grant/Taylor
GUI	Continue (y/n) buttons	PASS	04/13/18	Grant
GUI	Handle Initials button	PASS	04/12/18	Taylor
GUI	Validate Player Move	PASS	04/12/18	Grant
GUI	Optional: Show blinking victory	PASS	04/14/18	Grant
BACKEND	Victory Check	PASS	04/12/18	Grant
BACKEND	Validate Move	PASS	04/12/18	Grant
BACKEND	Store Game Board	PASS	04/12/18	Grant
BACKEND	Handle High-scores	PASS	04/16/18	Taylor
BACKEND	Import AI	PASS	04/13/18	Grant

GUI-2 Burn-list, Burn-chart, and Devlog:

GUI-2:

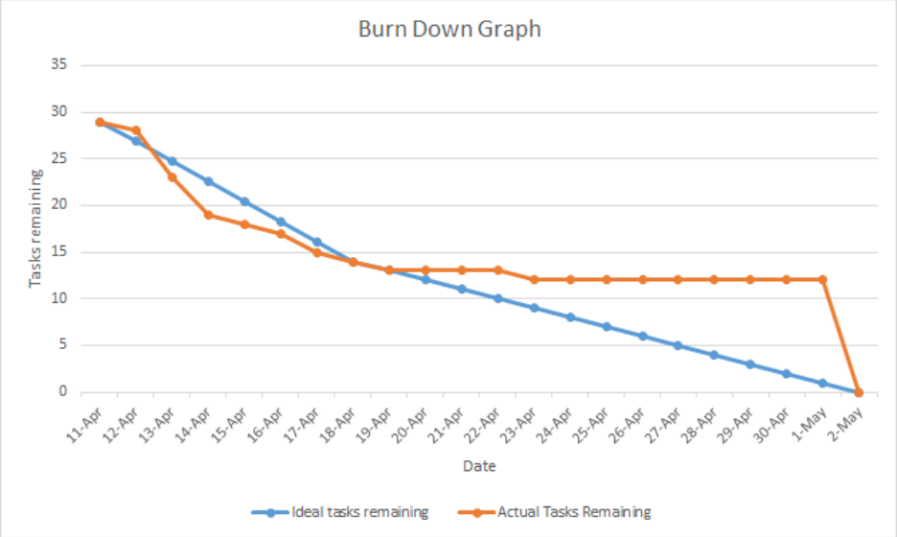
Task:	Expected Time Needed:	Time Used:	Daily Progress:	Overall Progress:
Collect reviews GUI-1	1 hour(s)	1 hour	100%	40%
Implement GUI Elememts	7 hour(s)	5 hour	100%	45%
Port over Backend	3 hour(s)	2 hour	100%	68%
Polish & finish game	2 hour(s)	1 hour	75%	63%
Collect reviews GUI-2	1 hour(s)	1 hour	60%	23%
Write Design Doc	2 hour(s)	3 hour	100%	74%
Presentation/slides	3 hour(s)	3 hour	100%	89%
Final Report	4 hour(s)	3 hour	100%	100%



Development Log for GUI-2 (WinForms, C#):

Type:	Feature/Requirement:	STATUS:	Date:	Name / Details:
ALL	GUI	PASS	05/1/18	Full Team
ALL	Back End	PASS	05/1/18	Grant/Taylor
ALL	AI	PASS	05/1/18	Grant
ALL	Get testimonials for GUI-1	PASS	05/1/18	Ethan/Grant
ALL	All functions < 24 lines?	PASS	05/1/18	Grant/Taylor
ALL	Final Report	FAIL	05/1/18	Full Team
ALL	Presentation Slides/Video	FAIL	05/11/18	Full Team
GUI	Splash Screen	PASS	05/1/18	Ethan
GUI	Instruction Screen	PASS	05/1/18	Chris
GUI	Game Screen	PASS	05/1/18	Grant
BACKEND	Victory Check	PASS	05/1/18	Grant
BACKEND	Validate Move	PASS	05/1/18	Grant
BACKEND	Store Game Board	PASS	05/1/18	Grant
BACKEND	Import AI	PASS	05/1/18	Grant
BACKEND	Handle High-scores	PASS	04/25/18	Taylor
BACKEND	Link High scores to GUI	PASS	05/1/18	Taylor
Optional	Show blinking victory	PASS	05/1/18	Grant
Optional	Add Player move sound			

Overall Project burn-down chart:



Section 12: References

- Microsoft .NET Documentation
  - https://docs.microsoft.com/en-us/dotnet/index
- Microsoft C# Guide
  - https://docs.microsoft.com/en-us/dotnet/csharp/