Final Written Report

5/14/2021

Ronni Kurtzhals

Exploring AI and Deployment of a Java Game

Over the course of the spring semester, I conducted research into three topics: artificial intelligence, package deployment, and multiplayer servers in Java. This research came together to form my project presentation on the implementation of these topics, which I felt accurately demonstrated the various things I have learned from my courses at Moorhead State University. Several resources were consulted throughout the project, including the work of W3Schools and StackOverflow as well as relevant assignments and textbooks from previous classes. I found this project relevant to computer science and information systems for several reasons, such as the AI component and use of SQL data tables; but it was also relevant because the project involved a deeper look into how app development and deployment works. I feel that these topics can hold high importance for potential computer scientists, and it seemed beneficial to learn more about them. Leveraging off knowledge from several courses at MSUM, I was able to accomplish most of my goals for this project and learn quite a bit about those goals that ultimately were not realized.

The project I intended to make was a card game application. It was going to include an AI bot that would function as the opposing player, the ability to download an installer package for the game, and the makings of a server allowing for multiplayer between two people. The installer would be found on a webpage written in HTML and PHP, and after downloading it would assist with the installation of the user interface for the game. The game itself would be written entirely in Java, and I wanted the user to be able to connect to other players instead of needing to play against the AI. I had started working on the user interface several months before deciding on this project, and thus it was working fine without all these additions. Using knowledge I had obtained on the Agile process, I thought that I would be able to efficiently add features to the already-built GUI: I chose to treat the project as a step-by-step process in order to finish what I had set out to do.

The first topic I dove into was artificial intelligence. Having never actually researched it before, I originally feared that it might be very difficult to implement the AI portion of the project. As it turned out, there were many complex forms of AI that were beyond my current understanding; however, it did not need to be complicated. AI can be done in a straightforward and simple manner when complexity is unnecessary, and the implementation came together quickly once I was able to see that. In this project, the AI is found within the game itself. As mentioned previously, it takes the form of the other player (or COM, as they are known within the game). The COM player is controlled by a Java class that determines the best card to play based on the cards currently in their hand. This can be demonstrated with a tree: first, the player starts with three random cards from the deck. Then, depending on the score and if the player has good cards or not, the tree traverses down the corresponding branch. Those branches also have one or two branches that will finally tell the COM player which card they should play. The tree begins again at the start of each of COM's turns, and it ends when the turn is over. It is not advanced by any means, but it does pick the card that a skilled player might pick. Currently, if no special cards are required to continue playing, the tree does not distinguish between regular cards. It simply selects a random card that fulfills all criteria to continue, and this could be made more efficient in the future. There are several more features that could be included in this AI class, such as the implementation of difficulty levels or the previously mentioned distinguishing between cards that are not special, but I felt these were better suited to later sprints. Of all the concepts touched on in the final project, the AI took the least amount of time and functions the most efficiently.

The area I spent the most time working with was the process of creating an executable file for the game. It seemed easy enough, but several issues arose during this endeavor. Due to these same issues, I ended up learning a lot about Java; differences between JREs (Java Runtime Environments) and JDKs (Java Development Kits), using installer packages, and compressing the files correctly were all topics that came up while researching the problems. The main issue I ran into while attempting to do this was that the version of software I chose to use, the IntelliJ IDE, did not have the capability to bundle together the necessary JAR file. After unsuccessful attempts in IntelliJ, I moved everything over to Eclipse, a different environment. This was also wildly unsuccessful for a time, but eventually I was able to determine the problem. While Eclipse was able to package the JAR I needed to create the executable, the IDE had several ways to go about it. Unsure of which to use, I had to rely on trial and error until I discovered the correct setting that enabled the project goal to be reached.

By the end of that sprint, I was able to figure out an exact workable process for the creation of a Java executable. To begin, an executable JAR file was needed. This was created in Eclipse by exporting all project files to what is called a 'runnable JAR file' in that IDE. Once this was finished, simply double clicking the new JAR should run the game. However, this was not enough to get the game running on others' computers; the JAR would also need to be packaged as a formal executable with a .exe extension. To accomplish this, I utilized a software called Exe4J found online. This software allowed me to select the runnable JAR file and convert it and anything else needing to be packaged into a single exe. The fresh executable would work on others' PCs, in theory. I soon realized that the process was not over, and that was when I learned about creating installer packages. As it turned out, anyone wanting to use the game would need to either have Java installed on their machine, or it would have to be packaged along with the executable file. The installer package needed for this particular bit was created with the InstallForge software; its purpose was to install the previously made exe onto the computer directly along with any Java libraries the game needs to run. InstallForge would not properly bundle the actual JRE with the game, and as a result users will still need to personally install Java. To rectify this temporarily, I added a page with instructions for downloading the JRE on the webpage designed for this presentation. In conclusion, this sprint ends with a neat desktop icon that launches the playable game when clicked.

The third and final sprint concerned the development of a multiplayer server that could be used for the game. The idea was to implement a fully functional server such that a player could link up with others who had the game installed and play against them instead of the COM. While this goal was not entirely realized, significant progress was made toward it. The server-client connection proved to be more of a challenge than I had anticipated; even though the client and server appear to be functioning as expected, the real problem arose when trying to connect the server to the GUI application of the game. The turn-based system works in theory, but the interface has yet to be completely synced with the system. In other words, the GUI does not necessarily reflect the choices made by the other players, but the server does indeed receive data from incoming clients and add them to the connection as appropriate. In the end, I was able to create a server and client class that utilized a TCP type connection, which is ideal for turn-based games. The server is initialized, and then clients can connect to it remotely. In its current version, two clients exactly are required for the server to allow a multiplayer game to begin.

Throughout this experience, the opportunities to learn something new were endless. I had taken classes in Java and GUI applications, and I knew the basics of connecting clients to servers, but some of the finer ins and outs had escaped me in previous semesters. Aside from the small details, I was happy to learn about how AI works and how simple it can start out as. This is something I look forward to researching more in the future as interest in and need for AI grows. I also thought that researching executables and deployment processes was extremely beneficial to my education; understanding how to do that could open many doors to future projects and research opportunities. Being as it isn’t something taught in my classes at MSUM, I wanted to explore the concept so that I could make games of my own and maybe show others how to do that, too. I also got to experience the Agile process in action because I treated each topic as a different sprint and focused more on what I expected out of each sprint rather than the final product. Utilizing this process seemed to make my work more efficient and less taxing on me. I believe the most important lesson I learned from the experiences gained is that it is most important to focus on expanding what you know and improving skills instead of delivering a perfect product on the first sprint. High-quality deployable projects take time, sometimes group efforts, and multiple sprints. New updated versions will bring more features and a better-quality GUI, but those are things to be thought about in later sprints. Taking small portions of a project at a time to perfect seemed to work well for me, but naturally the Agile process is not for everyone.

In summation, the project was a success. I may not have finished every goal I set out to achieve, but I was able to accomplish most of it and learn something significant about the rest. The final product produced during this run is a working GUI card game, where the installer is downloadable from a template webpage I created specifically for this project. The game can be accessed from the player's desktop or search bar. It works properly in Windows operating systems but has not been tested on any others such as Linux or Mac. Upon opening the game, the user will be asked to either log in or create an account. This information is sent or received from an SQL database through the school, and once logged in the user can select to play single or multiplayer. If single player is selected, then the game functions completely as expected with the AI correctly playing as the COM player and the user playing their hands themselves. When multiplayer is selected, the server starts and waits for two players to connect; however, this is not reflected whatsoever by the GUI at the application's current stage. In future sprints, I would continue working on the server's functionality before perfecting the other elements, but I am eternally glad to have accomplished the things I did over the semester.

References

DeGregorio, Amy. "How to Fix Java.lang.UnsupportedClassVersionError." Baeldung, March 17, 2021. https://www.baeldung.com/java-lang-unsupportedclassversion.

"Jar2Exe #1: Creating Custom Runtime Image." IT looks so easy, August 31, 2019. https://itlookssoeasy.com/java/creating-custom-runtime-image/#jlink-tool.

Sanmay Joshi. "Add Javafx Jmods to Java --List-Modules," June 1, 1968. https://stackoverflow.com/questions/57458954/add-javafx-jmods-to-java-list-modules.

"Where Developers Learn, Share, & Build Careers." Accessed February 3, 2021. https://stackoverflow.com/.

Martin, Robert C. *Clean Agile: Back to Basics*. Pearson Education, 2020.

"HTML." W3Schools Online Web Tutorials. Accessed February 3, 2021. https://www.w3schools.com/.

"A Computer Science Portal for Geeks." GeeksforGeeks. Accessed February 3, 2021. https://www.geeksforgeeks.org/.

wikiHow. "How to Make a File Downloadable from Your Website." wikiHow. wikiHow, January 31, 2021. https://www.wikihow.com/Make-a-File-Downloadable-from-Your-Website#:~:text=Open%20the%20page%20you%20want,the%20built%2Din%20page%20editor.

"Java Multiplayer Tutorial Episode 1 : Creating the Server." Youtube, n.d. https://www.youtube.com/watch?v=9vz-Dcdl8JA.

Leist, Rachel. "An Introductory SQL Tutorial: How to Write Simple Queries." HubSpot Blog. Accessed February 3, 2021. https://blog.hubspot.com/marketing/sql-tutorial-introduction.

"Creating an Executable jar File." Accessed February 10, 2021. http://www.skylit.com/javamethods/faqs/createjar.html.

"JavaFX - CSS." Tutorialspoint. Accessed February 10, 2021. https://www.tutorialspoint.com/javafx/javafx_css.htm.

"AI Algorithm Design: Card Game." Stack Overflow, April 1, 1960. https://stackoverflow.com/questions/6020969/ai-algorithm-design-card-game.

"Develop a Basic JavaFX Application - Help: IntelliJ IDEA." IntelliJ IDEA Help. Accessed February 10, 2021. https://www.jetbrains.com/help/idea/developing-a-javafx-application-examples.html.

“HTML - Email Links.” Tutorialspoint. Accessed February 10, 2021. https://www.tutorialspoint.com/html/html_email_links.htm.