Final Report
Runescape Mini-game

Brenden Davey

Advisor: Dr. Brian Ladd

Semester: Spring 2020

May 15, 2020

To conclude my education at SUNY Potsdam, I undertook a project that used the skills I learned during my years of education and built on them, both cementing my existing understanding, and learning new skills along the way. There is a game called *Runescape* developed by the company *Jagex* LTD that has been a part of my life since I was about eight years old. It has impacted my life in many ways from teaching me about British culture since *Jagex* is based in the United Kingdom, as well as developing lifelong friendships. Due to its impact on my life, I have wanted to develop a part of the game myself for a long time. There are many mechanics in the game that always made me curious about how the code for the game is written, and due to these curiosities, I was further pushed to find my own version of the game and immerse myself in it. After finding a version of the source, I began looking through the code figuring out how things work, and I quickly would lose track of hours. At that time, I had already decided to do my senior project on something else, but it quickly became apparent that I needed to change. I'd easily lose eight or more hours per day working with the *Runescape* code, and any time I was not doing something that required me to think, my mind would wander, thinking about how to do things with the source. This was at the same time that I had not begun touching the code for what I was planning on doing for my senior project. It had been a long time since I had this much passion to work on something and learn from it, so I proposed the idea to make my own mini-game inside of *Runescape*. I decided on a survival type minigame where the player fights waves of enemies that progress in difficulty. Beating waves awards points to the player that allows them to buy gear to get even stronger. What I proposed was exactly what I produced. There were minor changes to things here and there, primarily for balancing reasons, or due to unforeseen issues, but I produced my own survival mini-game that has 3 difficulties, with each difficulty having ten waves of enemies that progressively get more difficult. You are awarded points for each wave you make it through, and these points can be used to buy gear from a shop.

There were multiple factors that went into me being prepared to undertake this project. The main factor being the basic skills. *Runescape* is written in Java, so Computer Science I, Computer Science II, and Analysis of Algorithms were essential. Every class that I have taken in the Computer Science major helped prepare me, even if the class wasn't based around writing code. They all taught me how to be patient, and that solutions do not always come easy, but with enough time you can figure it out. I did not learn how to do proofs in foundations overnight, but by the end of the semester I knew my way around them. A very important part of my project was to be able to look at code that I had no part in writing, be able to understand what the code is doing without being able to ask the original author, and add to it. In many Computer Science courses, the students are provided some source code that they must take and build upon. The only way to build on code is to first understand what the code is doing, and how it is doing it. This combined with the patience is what made it possible for me to understand what the client code does because that code is from a decompiler, which means that there are useless variable names and class names, and no comments to help you understand. Outside of academics, I have a friend that completely wrote a server to go with a client for a different *Jagex* game, and I had been exposed to this before I touched my *Runescape* source, so I already had an idea of what to expect. Without that experience, it would have been a substantially slower start to the development of the mini-game. This same friend, who happens to be one of the lifelong friends that I gained through playing *Runescape*, has also helped me throughout my college career by being a source of supplemental knowledge. Any time I encountered problems he was essentially a tutor that would go through the problem with me and act as a guide without giving me the answer. He made sure I learned the material instead of just helping me get higher grades than what I otherwise would have.

Developing my own *Runescape* mini-game has taken the skills that I developed over the years of being a Computer Science major and went beyond in many ways. The most important thing I learned while completing the *Runescape* mini-game was just how valuable documentation is. Before I started

this project, I thought I knew the value of good documentation, whether it be having useful comments, variable names, function names, or class names. I recognized their importance, but sitting in front of a code base that is huge compared to anything I had ever worked with and finding that there are almost no comments anywhere on the server side, the idea was put into a whole new light. This is the first real project I have worked on that was not for one of my Computer Science courses where you write the code, hand it in, it is looked at once or twice, and goes into the archives. While I had been told many times that in the real-world code is read many more times than it is written, I never had first-hand experience with it. I found myself looking through many of the same classes every time I worked on the project without adding anything to them. Not having the documentation there meant that I would have to waste more time looking at how a section of code is working than I would have to if there were comments. In class we were always told we need to have comments, and in some cases if there are no comments, we would get a zero on the assignment. This combined with many assignments being so simple that it would take little effort to understand what is going on always made commenting seem tedious. This project took what every professor taught me to do for documentation and provided the real-world experience that is required to understand how important it truly is. After documentation, the next important thing that this project built upon from my formal education was how to scale things into classes. We learn early on that having a main class that is doing everything our entire program is designed to do is less than desirable. With a project of this scale it makes you plan further ahead before you start writing any code. Classes generally need to be smaller, and methods are made to do one thing. In many of the assignments for my classes, I would tend to have very large methods that had a lot of different things going on, but for this project almost all of my methods had one purpose. If something else needed to be done, it was made into a separate method. This process also made me realize what it would be like when I get a job in the real world. Due to this project, I now have experience with what a work-week would be like in this field. I never had experience with writing code for eight or more hours

each day for a week, and while making the mini-game I frequently would put at least that much time into it, and I really enjoyed it. Most of the assignments that we had over the years would have some testing that needed to be done, but many assignments were so small that there was very little to actually test. Before the mini-game I knew testing was incredibly important, but during the process I spent a fair portion of my time testing that things worked as intended, and when the mini-game was functional, there was a lot of time play-testing. When I took Game Development and Design I also had to do some play testing to balance the combat, but that game was much smaller, and a lot less involved, so my project took the skills that I began developing in Game Development and expanded on them to the point where I am confident in how the mini-game is balanced into the game around it. There are many things that I have learned throughout my education, and having the opportunity to complete this project took many of those concepts and either built upon them, or reinforced their importance.

This project was written completely in Java with nothing added to it. I originally started development using Eclipse because that is what I used for most of my Java programs for classes, but the deeper into the project I got, the more I looked at Visual Studio Code and eventually decided to make the swap. There are some features that make the process much quicker, like being able to have multiple terminals open at once inside the program. Both the client and the server must be running at all times in order to be able to login to the game. Being able to have those run in the same place that I am writing the code saved a lot of time, especially when compiling. Another feature that steered me toward Visual Studio code was how easily I could access version control also in the same place that I was writing the code. Having all the tools available to you in one location makes it much more immersive, and harder to get distracted. I mostly used the department's Gitea server for the development of the project because when I first started, I had issues getting my code onto Github. I eventually learned how to do what I wanted on Github and switched to that instead. I did not have to use Git to go back to previous versions when something broke, or I lost something, but I did have to use it for the code distribution. Seems I did

not always write code from the same computer, being able to push changes to a location off of my devices made it so that if I changed something in one location, I could almost instantly access the updated version when using a different machine.

The design and implementation process were largely decided for me by how the systems in place works. The existing server code was written in ways that are unlike anything I had ever seen before because it is largely not object oriented. Overall, I believe the system that is in place is terribly inefficient, but I decided to continue it like this anyway because it was a learning experience, even if what I learned was how I should never write code. There are no classes that I have taken that taught how switch statements work, but the people that wrote the server use them everywhere, so I decided to use this opportunity to learn about them, and use them myself. Even though I could have implemented the mini-game however I wanted, I decided to do it in the way that both fit the system, and was unfamiliar to me. The people that made the server have a file containing any object that you can click to interact with inside of a switch statement. This file contains an 1800-line switch statement which is where I started the work for the mini-game. When you click an object, the switch statement gets the ID of the object, so I made a portal for easy, medium, and hard. I decided from there that when you click it, it should move the character inside the area the mini-game takes place, but then the game has to actually start. I made a class for the survival mini-game that contains most of the methods used across the files. The main one is what actually starts the event. It houses all the information about the waves and spawns them. Due to how things are structured, there is a different location that handles when you have finished the wave, to call the spawn method again with the next wave. Figuring out the structure of things was a bit of a challenge without comments, but after looking through some of the other mini-games that exist, I eventually found where things needed to be.

The methodology that I followed was a loose Agile Methodology. I started off with a trello board that had the things I needed to complete in order to call the project done, but with being the only

person working on the project, I soon stopped having an actual visual backlog of the tasks that needed to be completed because I could keep it all in my head. I would take an element of the mini-game and work through that until it was completed which was essentially my sprint. Once the feature was done, I would re-evaluate what was left to complete, and based on what seemed to be the next most important part would be in the backlog for the next sprint. I call it a loose interpretation of the Agile methodology because while I follow the principals, sprints did not really have a set length. What I considered a sprint was essentially the amount of time that it took to finish a specific major feature.

The coding standard I followed was for the most part the department's Computer Science coding standards. Functions have pre-conditions, post-conditions, and return values. I also took a slight personal twist on it and if I felt like parts needed some more information around them, they received a comment as well, like in the switch statement that had the stats, there are comments off to the side of where the NPCs for the different difficulties start.

What this project ended up producing is a mini-game inside of an old decompiled Runescape client. The mini-game is a survival type minigame that has the player trying to survive waves of enemies. The user can choose between easy, medium, or hard difficulties. Each difficulty has 10 waves of enemies, where each wave has four normal enemies, and one boss enemy that is indicated by a flashing arrow over its head. The boss has higher offensive stats and health than the standard enemy, but less defense. When a player kills all of the enemies in a wave, the player is awarded points based both on the difficulty, and on which round they are on. Higher rounds give more points, as well as harder difficulties. The next wave spawns soon after the elimination of a wave. Once the player has finished all ten waves, they are teleported back out to the main area and awarded bonus points. In the main area there are a few things the player can do. There is a bank chest on the right side of the picture that contains all of the players items which they can use to swap out gear, there is an altar on the left side which is used to restore prayer points, and a monk that is right above the altar, that can heal the player

to full health every 10 minutes, and restores special attack. If the player is full health, it does not start the timer for when you can heal next. One of the most important things in the waiting room is the shop that must be accessed through a right-click, and click trade. This shop is where you can spend your survival points that you have earned on new gear that I also made. The shop keeper is right above the monk, and the portals are easy, medium, and hard respectively.

This gear used the models of existing gear that I recolored and gave new stats to. The second-tier sword I made from a two-handed sword that I turned into a one-handed sword, changed the attack speed of it, and how much energy it takes to use the special attack.

On the left is the first-tier armor and weapon with the stats, and on the right is the second-tier armor and weapon with its stats. The helmets are a bit glitched, but that's a client issue. If you go into a portal, you get transported into the survival area, and the enemies start spawning. When they spawn, they will immediately start coming after you, and keep following you as long as you are in the area. You fight until you either die, or finish all 10 waves. There is a portal in the room that allows you to leave whenever you want. A big part of how I chose to implement this was around a bug that would make you unable to interact with an enemy if it spawned too far away from where the player was. I eventually figured out the issue, and it was that the server would delete any record of the enemy because it was too far away from the player, but it still existed in the client. I found the bug way too far into development to change how the waves spawned, but it at least wont break if you run too far in a direction.

Once you have points, there is the shop.

You can left click an item and it will show you the cost, or you can right click and item to purchase it. Once it is purchased, you can equip it from your inventory if you have the stat requirements. All of the enemies are in a list that makes it easy to swap out what enemies spawn for each wave. All you need is the new enemy ID and to swap the new ID in for the old one. I can also increase the amount of enemies per wave, and number of waves by adding to the dimensions of the array. The only issue is if you try and

spawn more enemies than there are coordinates for.

```
// First dimension is number of waves, second dimension is IDs that will spawn in the wave
private final int[][] easyWAVES = { { 1265, 1265, 1265, 1265, 1265 }, { 74, 74, 74, 74, 74 },
        { 75, 75, 75, 75, 75 }, { 1153, 1153, 1153, 1153, 1153 }, { 76, 76, 76, 76, 76 },
        { 63, 63, 63, 63, 63 }, { 1218, 1218, 1218, 1218, 1218 }, { 1633, 1633, 1633, 1633, 1633 },
        { 1587, 1587, 1587, 1587, 1587 }, { 111, 111, 111, 111, 111 } };
// First dimension is number of waves, second dimension is IDs that will spawn in the wave
private final int[][] mediumWAVES = { { 1587, 1587, 1587, 1587, 1587 }, { 93, 93, 93, 93, 93 },
        { 111, 111, 111, 111, 111}, { 1616, 1616, 1616, 1616, 1616 }, { 1637, 1637, 1637, 1637, 1637 },
        { 1582, 1582, 1582, 1582, 1582 }, { 1624, 1624, 1624, 1624, 1624 }, { 1558, 1558, 1558, 1558, 1558 },
        { 1961, 1961, 1961, 1961, 1961 }, { 1565, 1565, 1565, 1565, 1565 } };
// First dimension is number of waves, second dimension is IDs that will spawn in the wave
private final int[][] hardWAVES = { { 2455, 2455, 2455, 2455, 2455 }, { 1615, 1615, 1615, 1615, 1615 },
        {2591, 2591, 2591, 2591, 2591}, {2604, 2604, 2604, 2604, 2604 }, {2614, 2614, 2614, 2614, 2614 },
        {2030, 2030, 2030, 2030, 2030 }, {2026, 2026, 2026, 2026, 2026 }, {1460, 1460, 1460, 1460, 1460 },
        {1157, 1157, 1157, 1157, 1157 }, {84, 84, 84, 84, 84 } };
// First dimension is number of spawn locations there are to be. THERE MUST BE AT LEAST AS MANY SPAWN LOCATIONS
// AS THERE ARE MOBS IN A WAVE
// second dimension has x,y coordinates for world location
private int[][] coordinates = { { 2772, 9630 }, { 2762, 9637 }, { 2767, 9638 }, { 2763, 9629 }, { 2772, 9634 } };
```

Coordinates can also be changed in a similar way to add more locations, or to change the existing ones.

Overall, this project took what I learned in the classroom and went above and beyond it, using both the skills I had already learned, and skills I had to learn to complete the project. This project was so much bigger than anything else I had worked on, so it made me really understand how classes interact with each other so much better. From using git throughout the stages of development, I am now much more confident with its functionality both in command line, and in the GUI. With this being my first experience with decompiled code, it took the skill of being able to read code and figure out what it is doing to a whole new level. By having to read code that has no documentation, and useless variable and class names, it really pushed me to use what I learned. Because of this, every assignment that I have had to do since the completion of this project has had much better documentation, and better variable names. Visual Studio Code is now a part of any code I write. It has become a great tool for so many tasks whether it be using the terminal, using version control, or just writing code. Being able to put all of my attention into one program has made me a better developer, and I would advise anyone else to find an editor that does everything you want. Your editor is your life as a programmer, so don't settle on one. Also, find something you are passionate about. A senior project is something that you spend months

with, so if it isn't something you are passionate about, you are not going to want to work on it, or enjoy it. This project is something that I was not expecting to get so attached to, but I fell in love with it even though the version of the client I found has a lot of bugs. I used the bugs as a learning experience, and I found ways around all of them to complete my vision. It provided more challenge which motivated me further. Overcoming the challenges of this project was incredibly gratifying, and reminded me that I am in the correct major for me. It was a great learning experience, and I'm glad it got to be a part of my education.