Computer Science Project - Final Report

Team 5 Members: Maike Keune, Nicholas Paquette, Maren Sandner, Brett Raible

Abstract

For our computer science project, our team's assignment was to create a game based on Japanese Role Playing Games (JRPG), the most well-known of which being the Final Fantasy series. Our program would allow players to play a game similar to early installments of the Final Fantasy franchise, which contained features like two-dimensional graphics, the ability to travel a tile-based world, explore dungeons, battle monsters, and collect money and items. We chose this idea for our computer science project as we have each had experience with playing games of this genre, some of us already with programming small games, and we were interested in creating a game similar to those we had played ourselves. Some members of our team also had access to a commercially available product, RPG-Maker VX-Ace, from which we were able to extract graphical and audio resources to use with our game.

For our technical innovation, our team decided to utilize the Java Spring framework as well as the JavaFX library for our graphics and to create a randomized overworld and randomized dungeons. In the progress of our project we also created an editor for creating fields.

Introduction

The idea of programming a game without an engine such as the Unity engine appealed to our group, as it seemed like a challenge as well as an opportunity to increase our programming skills while creating our own world for the game. As we came to find out, this was quite challenging, not just with programming but also with the inevitable project organization.

In the first weeks we gathered together and created a product backlog, a plan regarding all the features we would want our game to have. These included:

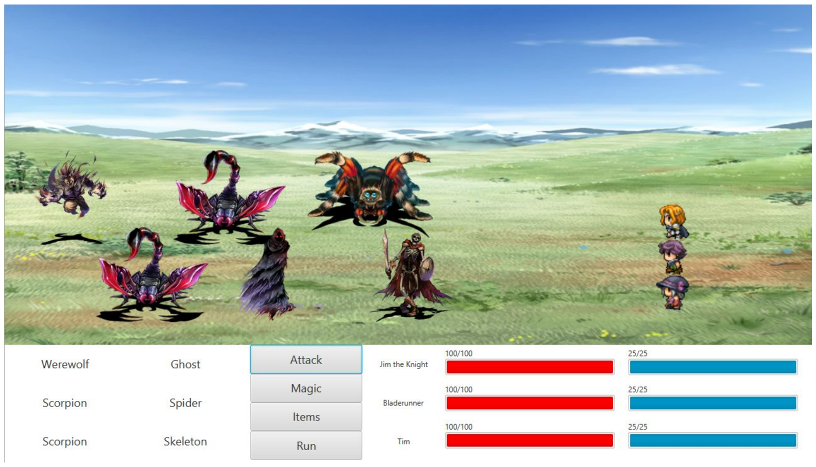
- A random overworld and dungeon generator.
- One moveable character on the screen .
- A battle system with random monsters, experience system and leveling up.
- Dungeons with boss monsters and rewards at the end.
- Items and a shop, as well as a currency system.
- A static town for the character to shop in.
- Quests for the player, given by a non-player character. (NPC)

As for every project we attempted to estimate the time we needed for the tasks and miscalculated in a few small parts. This means that not all features mentioned in the backlog could be implemented. In the end we worked more hours than we estimated in the beginning, (300 hours for all team members) yet because we underestimated the time it would take to complete some features we were unable to implement them all. However, we think we have created a good basic RPG with worlds and dungeons to explore and battles to win. We were also able to create our own field editor to help build maps for the world.

Body:

Actors and the Player Party
Every character in this game has an overclass called BattleActor, so in the end they all mainly function the same. The playable characters are set as a PlayerActor (extending BattleActor) and come together in a PlayerParty. This is set at the beginning of the game, based on input in the character selection screen. In the field you will walk around as the first PlayerActor saved in the PlayerParty. The Monsters class also extends from BattleActor and are different from the PlayerActors as they do not have mana points or individual spells. Monsters are saved in a MonsterParty, which can hold up to 6 monsters.

The Battle System



A battle is triggered by a five percent chance while walking in a field or a dungeon.
The game screen is changed to the battle screen, containing three player actors that you control and up to six random enemies. The enemies are randomly chosen from a switch/case structure and their levels are dynamically adjusted to those of the players. This way, they are either at the same level or one to two levels higher than the player's level. You can also see a menu at the bottom of the screen which has the buttons "Attack", "Magic", "Items" and "Run".
With "Attack" you use a basic attack and use no magic to damage an enemy. After pressing the button you get a list of the enemies at the top of the screen to choose which one you want to attack.
With the "Magic" button you first get a selection of the enemies and afterwards the list of available spells for the specific character. (Each character has their own set of spells) The spells for the characters are achieved after leveling up. At level one every character starts with just a basic attack and no spells.

Level	Knight	Mage	Thief
2	**"Shield Bash"** - has a sixty percent chance to disable the enemy for the next move while doing damage to him	**"Fireball"** - damages one enemy	**"Ambush"** - has a fifty percent chance to do double damage, otherwise normal damage
4	**"Whirlwind"** - damages the targeted enemy and a enemy next to it	**"Chain Lightning"** - damages two enemies next to each other at the beginning, with increasing attack the radius also gets increased	**"Mutilate"** - attacks the enemy with double damage
7	**"Berserk"** - does double damage to the enemy when the player is under 25% of his health	**"Heal"** - can heal a player in his party, when increasing the spell the amount of people getting healed also increases	**"Execute"** - kills a target if the health of the enemy is under 25%, otherwise does normal damage

11	**“Massive Sword Slash”** - makes enormous damage, but also consumes more mana than other spells	**“Frostbite”** - disables the enemy for one round	“**Shuriken Toss**” - causes low damage on all enemies on the battlefield

For all levels between those that grant the character a new spell, the character’s spells get stronger, increasing damage and the required mana. This is calculated with the mathematical equation **new_value = old_value + (4 * sqrt(current_level))**. After researching how to calculate damage, health and mana points, we came to know that equations with square roots are commonly used since they rise exponentially and are very practical for small games like ours. With leveling up, the health and mana points of the players also are increased, thus making them stronger. However, because enemies level up with the players, they also get increased health and attack damage. These values are calculated similar to the equation for the spells, only adjusting the magic number.
Furthermore, while in a battle, the player can choose to use a potion item available in their inventory. Potions can restore a character’s health or mana, or can restore the entire team’s health or mana.
The final button, run, allows the player to exit the battle without having to defeat all of the monsters, forfeiting any rewards they would have gained upon successfully completing the battle.

Items and the Shop
Items come in one of three different types: weapons, armor and potions. Weapons and armor are items that were designed to be able to be assigned or equipped to player actors to increase their attack or defense respectively. Furthermore, weapons and armor have class restrictions, limiting the types of player actors that can use those items. However, due to time constraints these features were not able to be implemented. These items still exist in the game, however they do not serve any purpose. Potions, on the other hand, are items that can be used during battle to restore health or magic to a player actor. All of these items can be found in a shop located in the town with shop field.
Upon entering the shop, the player may enter a shop screen by pressing the “B” key on their keyboard. Upon doing so, the shop screen appears, showing five main options to the player: weapons, armor, potions, sell and exit.The first three options bring up the shops inventories for those classes of items. Four items are shown at a time, and up and down buttons alongside the items allow the player to “scroll” through the inventory. To buy an item, the player need only press the button showing the item’s name and have enough currency to buy it. A final back button allows the player to return to the main options. Pressing sell brings up options similar to the previous three options, but this time instead of showing the shop’s inventories, it shows the party’s inventory. Pressing a button with an item name on this screen will sell the item, giving currency to the player and removing the item from their inventory. The final main option, exit, returns the player to the shop field to continue their game.

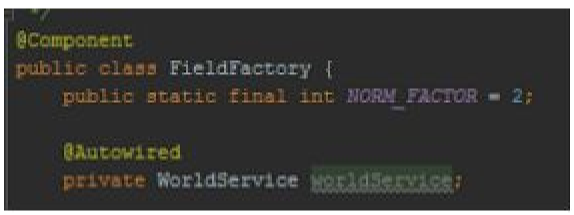
Field Editor
During the course of the first sprint, we realised that it would be too much work in data maintenance to code every single field by hand. Thus, the field editor was developed to simplify and shorten those ever recurring processes. The editor uses already existing algorithms from the main program, for example the render method to display the field in the main window of the editor. To edit the field, it was required to add click listeners to let the editor know how and when it has to manipulate the field. It was also necessary to add a save functionality next to the already existing load method.
Working with the editor made us realise that there is a misconception between our normal tiles and the tilesets we got from the RPG-Maker VX-Ace. The problem was that the tilesets were originally created for some method that can create quarter-tiles that take neighbour tiles somehow into account, to create smooth transitions between them. As we didn’t know how this was working, we had to create our own algorithm that could emulate

that. The Tile class got a new boolean named “complex” to tell the render method not to simply render one image but render four smaller quarter images depending on the neighbour tiles.

```
Private void render (Tile currentTile, Tile[][] neighbours, …) {
        …
        Group group = tileGroupMap.get(currentTile); // the javaFX scene graph
        …
        if(currentTile.isComplex()) {
                Tile upLeft = neighbours[0][0];
                … // add the four quarter images for one complex tile
                group.getChildren.add(getUpLeftPart(currentImage,currentTile, upLeft, ..));
                group.getChildren.add(getUpRightPart(currentImage,currentTile,upRight,..));
                group.getChildren.add(getDownLeftPart(currentImage,currentTile,..));
                group.getChildren.add(getDownRightPart(currentImage,currentTile,..));
        } else {
                ImageView imageView = new ImageView(currentImage);
                … // make this imageView look like the tile
                group.getChildren.add(imageView);
        }
}
```

This code snippet shows how the render method (found in class ScreenFactory.java, line 141) decides to render a complex tile with four smaller images or a normal tile with just one image.

Java Spring



```
@Component
public class FieldFactory {
    public static final int NORM_FACTOR = 2;

    @Autowired
    private WorldService worldService;
```

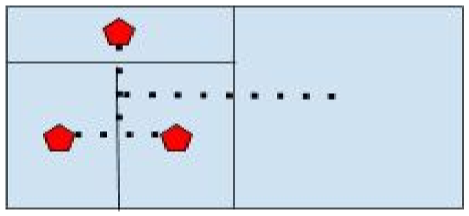
Spring is an open source framework for Java, the goal of this framework is to simplify the development process. We used Spring for our application-context to initialize the controller objects as Spring beans. A bean is an object that is instantiated and assembled by Spring. Spring automatically holds the controller-beans as singleton beans. One big advantage of Spring is that it makes sure that accesses to a bean are thread-safe (not like the classic Singleton pattern, where you have to take care of thread safety for yourself). As of version 2.5 of the Spring framework, the wiring of components are done by Spring automatically just by adding the @Autowired -annotation to members of a Spring component class.

Annotation	**Meaning of Annotation**
@Component	generic stereotype Annotation for Spring-managed component
@Autowired	declares methods or members (members don’t have to be public when marked with @Autowired)

Randomized Overworld and Dungeons

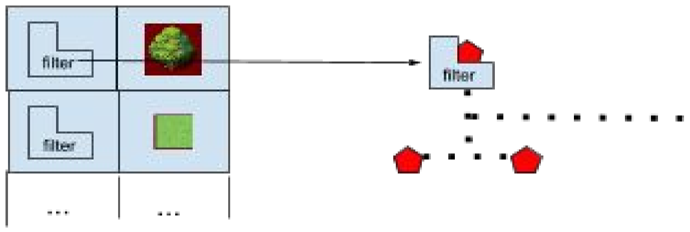
According to our Feasibility study one solution to get the randomly generated Overworld running was that we “..generate the random worlds through pre-made maps that get pieced together instead of just single tiles to reduce the odds of creating impractical worlds…” (p.1). The "pre-made maps" are technically fields, created with the editor and saved as .json files. Just like other static fields, these pre-made fields are loaded when first starting the game, and instead of setting the playable character on one of those subfields the generator creates a single large field and randomly arranges the loaded fields on it.

1.



recursive splitting, creates random layout

2.



left side: tile mapping, maps tile constructs to linear filters
right side: if a filter is recognised in the random layout, the mapped tile construct is placed at the recognised position.

The random dungeon generator was a more complex task. The single floors of a random dungeon each have their own unique layout of corridors and rooms. The rooms are made with the editor but get randomly arranged on the random dungeon layout. Conditions, such as height and width of a random floor, are given by parameters. First the layout is created by randomly splitting the whole area recursively until a specific threshold is reached. At this point we know where rooms will be placed later (by marking the middle of the subfield). Depending on the type of a dungeon a tile mapping can be used to fill the above created information, leading to the final field.

At the end all floors are linked together by teleporters so the player can progress from a source location to a target location (a town for example). This way a teleporter has the option to not directly teleport to a target location but let the player first cross a random dungeon before reaching the target.

Graphics & Audio Resources

All graphics come from a program called RPG Maker VX Ace. This program is designed to help game developers design their own RPGs. Though we of course did not use any code from the program to design our game, RPGM-VX:A allows people to export all of it's sprite, image, and audio files to their computer. Similarly, all but one audio file was exported from RPGM-VX:A. The one audio file that was not exported from RPGM-VX:A is known as GameBattleMusicv3.mp3, a song created in garageband by a friend of Brett's, Billy Baer.

ImageView Object

Because the visual for player movement in RPGMVXA is just one png file with 12 different character movement sprites, JavaFX made it really easy to simulate character movement in our program. An ImageView object shows the entire image, however, one can set the viewport of the ImageView object to show just a portion of the image. With our case, since all of the actor's movement sprites are in one picture, to show movement of the character, the viewport of the image can be moved from one position on the image to the next as the character walks.

Software Engineering Code of Ethics and Professional Practice Issues:

- 1.01. Accept full responsibility for their own work.
  - Everyone claimed their own code and when it needed to be changed, the rest of the team let the owner do the editing to it.
- 2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.
  - We worked together, as a team, on every document. Two of our group members did the corrections of the texts where needed.
- 3.01. Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.
  - We reached acceptable costs during some points by developing generic methods, so we didn't need to develop too many nested if-else cases. For reaching a high quality we researched classic rpgs to see how they worked. It was very important to find out how we could implement fields to save time, for example. Because of our inexperience, we were not able to anticipate how many features we would be able to implement. This led to either too late of a realization that features would have to be skipped (end boss, npcs, quests), or to working throughout the night to achieve those goals.
- 4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.
  - In situations of conflict we got together and talked it out over Skype. When this happened, we tried to include everyone so that the entire team knew what was going on.
- 7.03. Credit fully the work of others and refrain from taking undue credit.
  - We appreciated every step forward in our project. Also we tried to help when it was needed. We learned that appreciating everyone's work motivates every group member.
- 7.08. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.
  - In situations we experienced a loss of competence we asked each other for help. Also the internet was sometimes a big help, too (for Spring for example). We think it is important to face weaknesses and to be not too proud to ask for help.
- 8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance and testing of software and related documents, together with the management of the development process.
  - By the end of each sprint, the group did their best to fully understand exactly what was going on behind the scenes in our program. The sprint reports helped with this an incredible amount. One thing to mention is that for the first two sprints, everyone worked on the project blindly, they were given a generic task and made it into what they thought it was supposed to do. During the third sprint, the instructor, Prof. Adams suggested a spreadsheet with tasks for the sprint and to assign people these tasks. We tried it and it worked better than expected.

    Everyone knew what they had to try to get done and the tasks were specific which lead to an easier connection of the tasks.

- 8.03. Improve their ability to produce accurate, informative, and well-written documentation.
    - In the source code of our project, we quickly learned that comments are one of the most important things to include in a group project of any size. Along with this, the sprint reports, feasibility study, and project prospectus helped the team's documentation skills by providing papers that we all worked on together.

Teamwork Reflection:

- **Brett** - Working on a team is always a nice experience. When you need help with something you can just ask one of your teammates and see if they have an answer that satisfies the question. Alongside this, with multiple people working on a project, they often throw things into the project that I wouldn't have imagined putting in it. While sometimes it can be frustrating if one member does not pull their weight during a sprint, the point is that with the other group members the product got close to where you wanted it to be. This project taught me that when I am working on a group project as big as this one, communication is the key point that I would pass on to every other group to ever exist. Lack of communication will make a great project quickly fall into a failure. Overall, I had a good time working as a team on this project.
- **Nichloas** - Working in a group can be a challenge for me. When I don't have a good idea of what I'm supposed to be doing I can end up getting frustrated and not do a very good job. My performance in the first sprint was very lackluster, and our team wanted to make sure everyone knew what they had to do. During the second sprint, we once again failed to communicate as a group, and I only made some progress on some minor parts of the project. After the second sprint, we finally were able to work as a team and create a task sheet. With this, I was able to complete all of my tasks, though once again they were fairly minor. For the final sprint, I resolved to do some serious work on the project, and spearheaded the shop system. Though it took me a while, I was finally able to complete this portion of the project. In hindsight, I wish I had been a larger part of this project, especially seeing as how excited I was at the beginning of the semester. What I have learned is to dive head first into your projects, don't fall behind and be forced to take time figuring out what other people have done.
- **Maike** - The first two sprints on this project were very stressful for me, since i had to work till night on several days to finish tasks (some tasks had to be done on the last day, when another group member didn't finish them). I learned in this project that I will definitely try to help others out, if they ask, but I won't finish work for other people again just to get the sprint finished. In future projects I would always suggest a task sheet, so every group member knows what to do. Also I definitely have to say that devaluing the work of others in a group is not acceptable and in the future I won't allow anyone to do that to me or other group members again. We had some positive times and I think teamwork will be an important part in our professional life. Personally I think teamwork can be a really fun and nice experience.
- **Maren** - This project was perfect example that organisation and reliable team members are key to any successful project. When others don't do their assignments or do too much work it harms the whole group. Everyone should work the same amount of time so it is fair and everyone has evenly distributed to the end result. Since we didn't use a spreadsheet for the assignments in the first two sprints, I think we weren't as organized as we could have been. For my next projects I will definitely care for a good organisation right at the beginning and be stricter when assignments don't get done properly. Neitherless we worked hard and reached our goal and now have a fine working game that was fun to code and is now to play. I think the communication with the different time zones was sometimes a little bit difficult but never a real problem. The project was still fun and I would do it again.

Conclusions:
It's important to think abstract but not too complex (or little tasks may become big) and to make plans on how to develop specific tasks. It's useful to draw some kind of paper-prototype if a problem is very complex, in order to help figure out how to solve it. Sometimes it helps to talk with colleagues or friends about the project to get another sight of it (this could help you to develop an idea). Project management is a very important part of development and shouldn't be underestimated. We learned that it is necessary to manage how your project is designed, especially with something this large. As we didn't have any real management, this would be a part of the project that we would have done differently. GIT and the version control in Intellij IDEA really helped us to develop this project. Overall, we wish we could have done more with our project, but we are proud of what we have created in the amount of time given to us.