
Final Write-up

for

Dungeon Quest

Version 1.0 approved

Prepared by

Trevor Evans

5/2/2006

Table of Contents

1. Introduction.....	1
1.1 Abstract.....	1
1.2 Problem Statement	1
2. Planning Process.....	1
3. Requirements.....	1
3.1 Functional requirements	1
3.1.1 “Must-have” System Functional Requirements	1
3.1.2 “Would-like” System Functional Requirements.....	5
4. Design.....	5
4.1 Algorithms	5
4.1.1 Prim’s Algorithm: Random Maze Generation.....	5
4.2 User Interface	5
4.2.1 Interface Requirements.....	5
4.2.2 Interface Screenshots	8
4.3 Modules.....	12
4.3.1 DQFrame.....	13
4.3.2 GameCharacter.....	13
4.3.3 Item.....	13
4.3.4 Spell.....	14
4.3.5 Map.....	14
4.3.6 Tile.....	14
4.3.7 TreasureChest.....	14
4.4 Architecture	14
5. Software Development Process	15
5.1 Development Lifecycle	15
5.2 Testing and Debugging.....	15
5.3 Challenges: Where Time was spent.....	15
5.3.1 Game Design	15
5.3.2 Swing’s Layout Managers.....	15
5.3.3 Maze Generator Debugging.....	15
5.4 Actual Work Schedule.....	16
5.4.1 Pie Chart	16
5.4.2 Gantt Chart.....	16
6. Analysis, Results, and Discussion.....	17
6.1 Did the Program Work?	17
6.2 What Worked Well?.....	17
6.2.1 Random Maze Generator.....	17
6.2.2 Java’s Swing Package	17
6.3 What Needs Improvement?.....	17
6.3.1 Animation.....	17
6.3.2 Interface Polish.....	18
6.3.3 Story Line.....	18
6.3.4 Boss Monsters and End Game.....	18
6.3.5 Quick Maze Traversal Issue.....	18
6.3.6 Number Balancing	18
7. Lessons Learned and Conclusions	19
7.1 Lessons Learned.....	19
7.2 Conclusion.....	19
8. References.....	20

1. Introduction

1.1 Abstract

My goal was to implement a low-scale, role-playing game with the at least the bare-minimum features of a classic, Japanese-style RPG. Such features include, but are not limited to: 2-D overhead maps, random-encounter battles with a variety of monsters, a turn-based battle system, a character development system, an assortment of armor, weapons, and other items for the player to equip and use, and the ability to save game progress and continue a previously saved game.

1.2 Problem Statement

Since my project was to develop a game, there was no real problem addressed by the software, except maybe trying to curb boredom. You could say the problem was to develop a role-playing game with a classic feel that is both addictive and fun to play.

2. Planning Process

The idea for this game came straight out of my own head, but was inspired by the many hours of my youth spent playing role-playing games. I had no other client but myself. I began the brainstorming process over the Christmas break between fall semester 2005 and spring semester 2006. I thought about all the features of a bare-bones role-playing game and how I would implement each feature. I decided it would be best to use Java as my programming language, since it is the language I am most familiar with. Once I was confident that I could implement all these features in Java, I knew I had my senior project. I spent the rest of the break making a list of formal requirements and creating skeletons for each module. After submitting the pre-proposal and proposal documents and presenting my proposal, I began the formal design process.

3. Requirements

3.1 Functional requirements

The following is a list of the original functional requirements for the system. I have used following color coding scheme: implemented as originally proposed, **modified from original proposal**, and **dropped**.

3.1.1 “Must-have” System Functional Requirements

3.1.1.1 Upon starting the game, the title screen appears with a prompt for the player to start a new game, or load an existing one.

- 3.1.1.1.1 Choosing to start a new game prompts the player to enter a name for the main character (henceforth referred to as “the hero”). The player then takes control of the hero, which he/she will use to navigate the game world.
- 3.1.1.1.2 Choosing to load a saved game opens a file chooser, filtered to display only Dungeon Quest Save (.dq) files. Once the player selects the file and clicks the Open button, the saved game is loaded and the player will take control of the hero.
- 3.1.1.2 The game world consists of 2-D maps that the player can navigate using the arrow keys on the keyboard. The maps will be made up of 32 x 32 pixel tiles. The tile’s type determines whether or not the hero can walk over it (i.e. walls, water, etc. are impassable).
- 3.1.1.3 There is a main menu the player can access at anytime while navigating a map. The menu has tabs, with a status panel where the player can view the hero’s stats and allocate skill points, an inventory panel where the player can equip the hero with weapons and armor and use certain items, and a spell panel where the player can cast certain spells.
- 3.1.1.4 The hero’s status is made up of twelve stats:
 - 3.1.1.4.1 Experience (EXP) – The hero earns EXP for every victorious battle. When the hero reaches certain EXP milestones, he gains a level.
 - 3.1.1.4.2 Level (LVL) – The hero starts at LVL 1 and maxes out at LVL 50. When the hero gains a level, he is awarded 10 skill points.
 - 3.1.1.4.3 Skill points (SP) – The player raises the hero’s other stats (excluding level and experience) by allocating skill points to them.
 - 3.1.1.4.4 Health Points (HP) – The hero loses HP when hit with a monster’s melee attack or spell, or as the result of some negative status effects. The hero can restore HP with healing items, healing spells, or by staying at the inn. When the hero’s HP reaches zero, the game will end.
 - 3.1.1.4.5 Magic Points (MP) – The hero needs MP to cast magic spells. The more powerful the spell, the more MP it costs to cast. The hero can restore MP with magic restoration items or by staying at the inn.
 - 3.1.1.4.6 Attack (ATK) – This determines how much damage the hero inflicts on a monster when attacking with a weapon. Higher ATK means more damage.
 - 3.1.1.4.7 Accuracy (ACC) – This determines how likely the hero is to land a successful hit on a monster when attacking with his weapon. Higher ACC means a better chance of landing a hit.
 - 3.1.1.4.8 Defense (DEF) – This determines how much HP the hero loses when hit by a monster’s melee attack. Higher DEF means less HP lost.
 - 3.1.1.4.9 Wisdom (WIS) – This determines how powerful the hero’s spells are when cast. Higher WIS means more HP restored with healing spells, more damage inflicted with attack spells, etc. Higher WIS also increases the chance of the spell being cast successfully.

- 3.1.1.4.10 Magic Defense (MDEF) – This determines how much HP the hero loses when hit by a monster’s attack spell and how much his stats are lowered when hit by a status lowering spell. Higher MDEF means less HP lost.
- 3.1.1.4.11 Speed (SPD) – This determines whose move is executed first each turn in battle. If the hero’s SPD is higher than the monster’s, his move is executed first, otherwise the monster’s move is executed first.
- 3.1.1.4.12 Evasion (EVA) – This determines how likely the hero is to dodge a monster’s melee attack. Higher EVA means a better chance of dodging the attack.
- 3.1.1.5 The hero has an inventory with a maximum capacity of ten items, including those he has equipped. All items stack; duplicates of items (up to 99) take up only one spot in the inventory.
- 3.1.1.6 There are two types of items: usable items and equipment. Usable items, such as medicinal herbs and magic scrolls, disappear from the inventory when used from either the main menu or while in battle. Equipment, such as swords and shields, are equipped to provide the hero with stat boosts in battle.
- 3.1.1.7 Equipment comes in five types:
- 3.1.1.7.1 Weapons – These include swords, daggers, staves, etc. These items boost the hero’s ATK rating. Some weapons also provide other stat boosts and special effects in battle as well.
- 3.1.1.7.2 Shields – These items boost the hero’s DEF rating, but lower EVA and SPD. Some shields also provide other stat boosts and special effects in battle as well.
- 3.1.1.7.3 Head Gear – These include helmets, hats, masks, etc. These items boost the hero’s DEF rating. Some pieces of head gear also provide other stat boosts and special effects in battle as well.
- 3.1.1.7.4 Body Gear – These include tunics, robes, suits of armor, etc. These items boost the hero’s DEF rating. Some pieces of body gear also provide other stat boosts and special effects in battle as well.
- 3.1.1.7.5 Accessories – These include rings, boots, gauntlets, etc. These items provide stat boosts to any of the hero’s stats (except EXP, LVL, and SP) and provide special effects in battle as well.
- 3.1.1.8 The inventory panel of the main menu displays which equipment the hero has equipped. The hero has five slots in which he can allocate equipment:
- 3.1.1.8.1 Weapon – This is the slot where the hero equips items typed as weapons.
- 3.1.1.8.2 Shield – This is the slot where the hero equips items typed as shields.
- 3.1.1.8.3 Head – This is the slot where the hero equips items typed as head gear.
- 3.1.1.8.4 Body – This is the slot where the hero equips items typed as body gear.

- 3.1.1.8.5 Accessory – This is the slot where the hero equips items typed as accessories.
- 3.1.1.9 The hero is able to learn a maximum of ten spells. Once this limit has been reached, the hero will have to forget an old spell in order to learn a new spell. The hero learns spells from magic scrolls.
- 3.1.1.10 The game world consists of three areas: a world map, a town, and a dungeon.
- 3.1.1.11 The world map and the town consist of one map each and are safe havens for the hero, meaning there are no random-encounter battles on these maps.
- 3.1.1.12 The town map contains four buildings of importance to the player:
- 3.1.1.12.1 an item shop where the player can purchase usable items by interacting with the shopkeeper NPC
- 3.1.1.12.2 a weapon and armor shop where the player can buy and sell equipment by interacting with the shopkeeper NPC
- 3.1.1.12.3 an inn where the player can rest the hero for a fee, fully restoring HP and MP, by interacting with the innkeeper NPC
- 3.1.1.12.4 a church where the player can save his/her game progress by interacting with the nun NPC
- 3.1.1.13 The dungeon consists of multiple floors of randomly generated mazes, with each floor being a separate map. It is a “magical” dungeon, thus the mazes change every time the player leaves a floor. The mazes grow in size as the hero travels deeper into the dungeon.
- 3.1.1.14 The exit of each floor is guarded by a boss monster which the hero must defeat in battle in order to advance deeper into the dungeon. Once that boss has been defeated, it will reappear if the hero leaves that floor and returns at a later time.
- 3.1.1.15 Treasure chests are placed randomly throughout each maze of the dungeon. The number of chests on each floor is relative to the size of the maze; the larger the maze, the more chests. Each chest contains either a random amount of gold, or a random item. The value of a chest’s contents is also relative to the depth of the floor; the deeper the hero is in the dungeon, the greater the value.
- 3.1.1.16 While exploring the dungeon, random-encounter battles will occur. Upon entering the dungeon, a randomly generated value is assigned to an encounter count-down. The count-down will decrement by one with every step the hero takes. When the count-down reaches zero, the battle window pops-up and a battle ensues. If the hero comes out of the battle alive, a new value is generated for the count-down. This value will always be hidden from the player. As the hero travels deeper into the dungeon, the strength of the monsters he encounters will increase, as will the rate of random-encounter battles.
- 3.1.1.17 Each monster will have the same types of stats as the hero, with the exception of SP, as well as a list of spells and special attacks. This information will always be hidden from the player.

3.1.2 “Would-like” System Functional Requirements

I planned to implement the following features only if time permitted:

- 3.1.2.1 background music and sound effects
- 3.1.2.2 a party of heroes for the player to control
- 3.1.2.3 multiple monsters to fight, concurrently, during battles
- 3.1.2.4 a larger game world with multiple continents, towns, and dungeons

4. Design

4.1 Algorithms

4.1.1 Prim’s Algorithm: Random Maze Generation

For the random maze generator, I ended up implementing a randomized version of Prim’s algorithm. The algorithm works as follows:

- 4.1.1.1 Pick a cell and mark it as part of the maze. Add the walls of the cell to the wall list.
- 4.1.1.2 While there are walls in the list:
 - 4.1.1.2.1 Pick a random wall from the list, and make it a passage.
 - 4.1.1.2.2 Mark the cell on the opposite side as part of the maze.
 - 4.1.1.2.3 Add the neighboring walls of the cell to the wall list.

4.2 User Interface

The following is a list of the original interface requirements for the system. I have used following color coding scheme: implemented as originally proposed, modified from original proposal, and dropped.

4.2.1 Interface Requirements

- 4.2.1.1 While navigating a map, the player can open up the main menu window by pressing the space bar or right-clicking the mouse. The player can close the main menu window by clicking the X in the top-right corner, or by pressing the escape key on the keyboard.

- 4.2.1.1.1 From the status panel of the main menu, the player can allocate SP to any of the other stats **with editable text boxes** (all except EXP and LVL), **by typing the desired value into the appropriate text box, then left-clicking the APPLY button with the mouse. If the hero has enough SP, the points will be allocated and the hero's SP will be decremented by the appropriate amount. Otherwise, an "insufficient SP" error will be displayed.** The player can left-click the decrement and increment buttons next to the appropriate text box to de-allocate and allocate SP, then left-click the APPLY button to make the changes permanent.
- 4.2.1.1.2 The inventory panel of the main menu displays a list of the hero's inventory of items, **as well as the hero's equipment slots.** The player can use an item from the main menu by **left-clicking its name in the list and then clicking the USE button, or by scrolling the list with the arrow keys and pressing "enter" on the keyboard when the desired item is highlighted.** Weapons and armor are equipped in a similar fashion, **except the USE button toggles to the EQUIP button when the selected item is a piece of equipment.** Items and equipment are removed by **left-clicking the item's name in the inventory, or the equipment's name in the equipment slot, and then clicking the REMOVE button.**
- 4.2.1.1.3 The spell panel of the main menu displays a list of the hero's spells. The player can cast a spell from the main menu by **left-clicking its name in the list, then clicking the CAST button, or by scrolling the list with the arrow keys and pressing enter key on the keyboard when the desired item is highlighted.** The player can remove a spell from the list in a similar fashion, by **left-clicking its name in the list and clicking the REMOVE button, or by scrolling the list with the arrow keys and pressing the delete key on the keyboard when the desired item is highlighted.**
- 4.2.1.2 While navigating a map, the player can interact with non-player characters (NPCs) and certain object tiles adjacent to the hero by facing towards it and pressing the enter key on the keyboard **or left-clicking the mouse.**
- 4.2.1.3 Interacting with a shopkeeper NPC brings up a shopping menu window. The menu has tabs, with a buy panel where the player can select items from the shop's inventory that he/she would like to buy, and a sell panel where the player can select items from the hero's inventory that he/she would like to sell. The player can close this window by clicking the X in the top right corner, **or by pressing "esc" on the keyboard.**
- 4.2.1.3.1 The buy panel of the shopping menu displays a list of the shop's inventory. The player can purchase an item by **left-clicking its name in the list and then left-clicking the BUY button, or by scrolling the list with the arrow keys and pressing the enter key on the keyboard when the desired item is highlighted.** The sell panel of the shopping menu works in a similar fashion, with a SELL button rather than a BUY button.
- 4.2.1.3.1.1 If the player has enough gold when making a purchase, the item bought is **removed from the shop's inventory and** added to the hero's inventory and the amount of the purchase will be removed from the player's gold total. If the player does not have enough gold to make the purchase, **the BUY button will be disabled.**
- 4.2.1.3.1.2 Selling items to the shopkeeper will remove the item from the hero's inventory, then **add them to the shop's inventory and** add the appropriate amount of gold to the player's gold total. The shopkeeper has an infinite amount of gold with which to buy items at half the price he sells the item for.

- 4.2.1.4** When exploring the dungeon, if the hero is positioned on top of an exit or entrance tile, **the player must press the enter key on the keyboard for the hero to take the stairs down to the next floor or up to the previous floor.**
- 4.2.1.5** When exploring the dungeon, the player can open a chest by interacting with it like he/she would with an NPC. Upon opening a chest, a pop-up message displays the contents of the chest, the contents are transferred to the hero's inventory, and the chest disappears.
- 4.2.1.6** The battle window consists of four sections: the monster's image, the command buttons, the hero's HP and MP status, and the battle log.
- 4.2.1.6.1** The battles are turned based, with the hero and the monster taking one turn each per round. At the start of each round, the player must issue one of four commands, by left-clicking the appropriate command button, **or pressing the corresponding key (1 – 4) on the keyboard:**
- 4.2.1.6.1.1** ATTACK – Choosing this command orders the hero to attack the monster with whichever weapon he currently has equipped during his turn.
- 4.2.1.6.1.2** MAGIC – Choosing this command brings up a pop-up menu with a list of the hero's spells. The player must then select a spell for the hero to cast by either left-clicking, on the name of the spell in the list, or by scrolling through the list with the arrow keys and pressing the enter key on the keyboard when the desired spell is highlighted. If the hero has sufficient MP, he will cast the spell during his turn. **If the hero does not have sufficient MP, the spell's menu item will be disabled.**
- 4.2.1.6.1.3** ITEM – Choosing this command brings up a pop-up menu with a list of the hero's usable items in his inventory. The player must then select an item for the hero to use by either left-clicking on the name of the item in the list, or by scrolling through the list with the arrow keys and pressing the enter key on the keyboard when the desired item is highlighted. **If the item selected is a piece of equipment, it will be equipped to the appropriate slot, the previously equipped item will be unequipped, and player will be able to issue another command.** If the item selected is a usable item, the hero will use that item during his turn and the item will be removed from the inventory.
- 4.2.1.6.1.4** FLEE – Choosing this command orders the hero to attempt to flee from battle. If the hero is successful, the battle ends with the hero receiving no EXP or gold. If the hero is unsuccessful, he will forfeit his turn for that round. The odds of the hero successfully fleeing from battle is based on the difference between the hero's level and the level of the monster (i.e. if they are equal, 50% odds; if the hero is five levels higher, 55% odds, etc.). **The hero will not be allowed to flee from battles with boss monsters.**
- 4.2.1.6.2** After the player inputs a command for his turn, the monster randomly chooses an action from its list of spells and attacks. The odds for each action varies (i.e. a monster is more likely to choose a normal attack that does lesser damage than a powerful attack spell that does critical damage). The character that has the highest SPD rating will execute its action first. The results of each character's action are displayed in the battle log and the hero's HP and MP status are updated when necessary. If both characters have more than zero HP remaining at the end of the round, the player is prompted to choose his next command.

4.2.1.6.3 If the monster's HP drops to zero, the battle comes to an end and the hero is victorious. The hero is then awarded EXP and gold, the amount of which is displayed in the battle log. If the hero levels up as a result, it is displayed in the battle log as well. [The battle window closes automatically after a few seconds.](#)

4.2.1.6.4 If the hero's HP drops to zero, the appropriate message is displayed in the battle log and the battle will come to an end. [The battle window closes automatically after a few seconds.](#) The game will then return to the title screen when the player clicks any button on the keyboard.

4.2.2 Interface Screenshots

4.2.2.1 Title Screen



4.2.2.2 *Navigating the World Map*



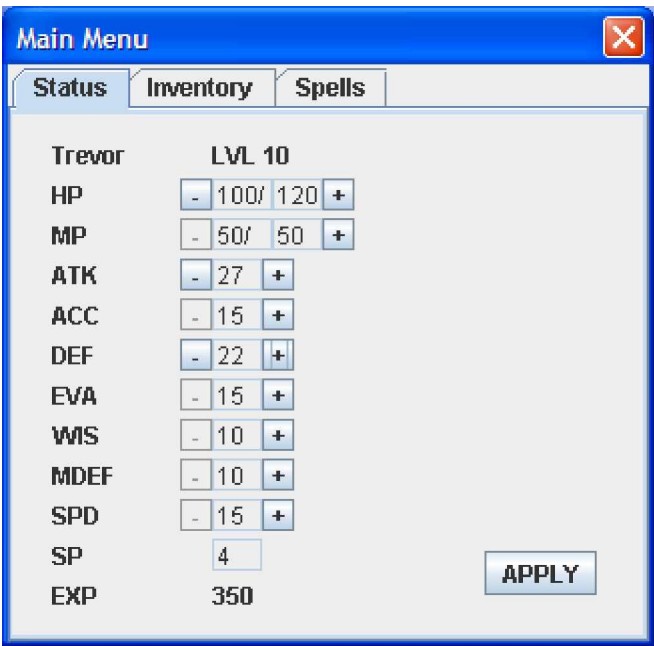
4.2.2.3 *Navigating the Town Map*



4.2.2.4 Navigating the Dungeon Map



4.2.2.5 Status Panel



4.2.2.6 *Inventory Panel*

The screenshot shows the 'Main Menu' window with the 'Inventory' tab selected. The inventory is organized into categories: WEAPON, SHIELD, HEAD, BODY, ACCESSORY, and GOLD. Each category has a corresponding input field. The 'DESCRIPTION' box shows the details for the selected 'herb' item.

Item	Quantity
bronze sword	x1
bronze shield	x1
magic ring	x1
bandit bandana	x1
bandit tunic	x1
herb	x10
potion	x5
minor strength	x1

DESCRIPTION
Restores 25 HP when used.

WEAPON
bronze sword

SHIELD
bronze shield

HEAD
bandit bandana

BODY
bandit tunic

ACCESSORY
magic ring

GOLD
71

DROP **USE**

4.2.2.7 *Spells Panel*

The screenshot shows the 'Main Menu' window with the 'Spells' tab selected. The spells list includes 'minor heal', 'minor blast', 'minor shield', 'minor weakness', and 'minor strength'. The 'DESCRIPTION' box shows the details for the selected 'minor heal' spell. The 'MP REMAINING' is 50.

minor heal
minor blast
minor shield
minor weakness
minor strength

DESCRIPTION
Recovers a small amount of HP.

MP REMAINING 50

REMOVE **CAST**

4.2.2.8 Shopping Menu



4.2.2.9 Battle Window



4.3 Modules

The system is comprised of several modules. The following is a list of some of the more important ones, along with a short description of what they do.

4.3.1 DQFrame

This is the largest module in the application. It contains all of the interface related modules.

4.3.1.1 MapPanel

This is the panel where the maps are displayed.

4.3.1.2 MainMenu

This is the main menu window. It contains 3 tabbed panels: Status, Inventory, and Spells.

4.3.1.2.1 StatusPanel

This is the status panel within the main menu. This is where the player can view the hero's stats and allocate SP to them.

4.3.1.2.2 InventoryPanel

This is the inventory panel within the main menu. This is where the player can use/remove items and equip/remove weapons and armor.

4.3.1.2.3 SpellsPanel

This is the spells panel within the main menu. This is where the player can cast/remove spells.

4.3.1.3 WeaponAndArmorShopMenu & ItemShopMenu

These are the weapon and armor and item shop menus. They contain 2 tabbed panels: Buy and Sell.

4.3.1.3.1 BuyPanel

Displays the shop's inventory and lets the player purchase items.

4.3.1.3.2 SellPanel

Displays the hero's inventory and lets the player sell items.

4.3.1.4 BattleWindow

It contains four sections: the monster's image, the command buttons, the hero's HP and MP status, and the battle log.

4.3.2 GameCharacter

This module creates a new game character. There are two types of game characters: heroes and monsters. Heroes are controlled directly by the player, while monsters are the player's adversaries controlled by the system.

4.3.3 Item

This module creates a new item. There are two categories of items: usable and equipment. Usable items disappear from inventory after one use. Items of type HEALING and MAGIC_SCROLL fall under this category. Equipment is allocated to equipment slots to give the user the stat boosts they provide and stay in inventory until they are dropped or sold. Items of type WEAPON, SHIELD, BODY, HEAD, and ACCESSORY fall under this category.

4.3.4 Spell

This module creates a new spell. Each spell requires a set amount of MP to cast. Spells come in four types: healing spells that recover HP, buffing spells that boost stats, de-buffing spells that lower stats, and offensive spells that deal out damage.

4.3.5 Map

This module creates a new map, which is then displayed in the map panel. The game world is made up of these maps, which the player navigates using the arrow keys.

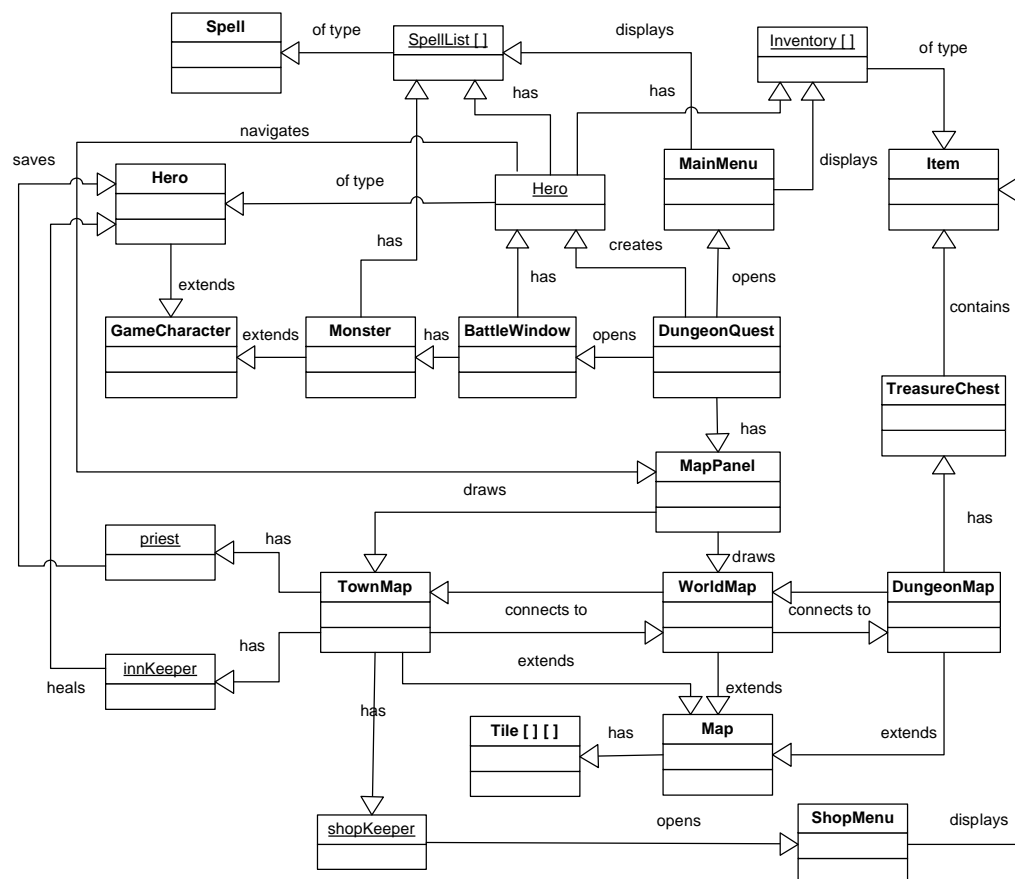
4.3.6 Tile

This module creates a new tile. Each map is made up of an array of these 32 X 32 pixel tiles.

4.3.7 TreasureChest

This module creates a new treasure chest. Treasure chests appear randomly in the dungeon and contain random items. The value of items increase the deeper the hero journeys into the dungeon.

4.4 Architecture



5. Software Development Process

5.1 Development Lifecycle

I decided to use the prototyping development methodology for this project. Once I had an initial working prototype, I played the game for a while, looked for modifications to be made, and decided which features to implement next. I repeated this process until I had completed all the requirements I planned to implement this semester, along with a few modifications.

5.2 Testing and Debugging

I spent quite a bit of time testing and debugging. I decided to perform incremental testing: create a module, integrate it into the system, then test it and debug it if necessary. After I had implemented enough features to consider the project completed, I spent a week playing through the game, squashing any last minute bugs and making adjustments to monster, item, and spell stats.

5.3 Challenges: Where Time was spent

5.3.1 Game Design

I probably spent the most time designing the game: maps, monsters, items, spells, and the interface. I designed all of these things before I started coding them. Designing the monsters, items, and spells took quite a while because of the sheer number of each that I created and because of all the number crunching it took to keep everything in balance.

5.3.2 Swing's Layout Managers

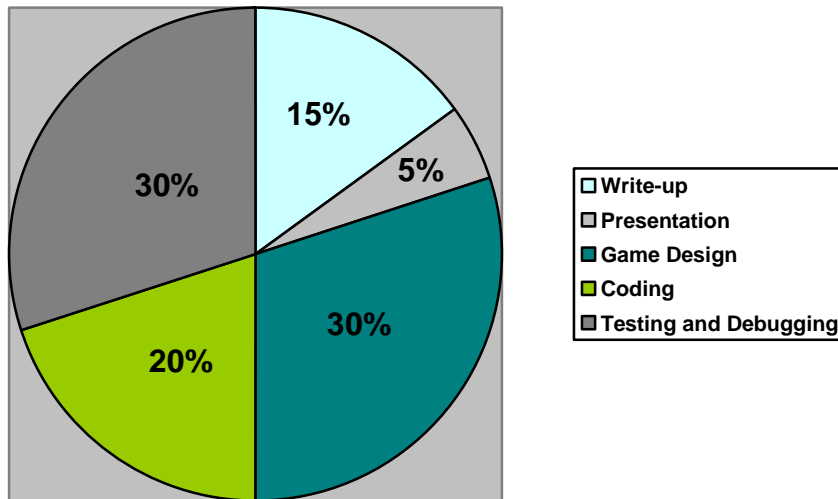
The actual implementation of the interface took lots of time because of the trouble I had initially using Swing's layout managers. I tried them all, but none seemed to suit my needs. After a few days of futile effort, I decided that I would be better off using no layout manager. It took several hours to size and position each component individually, but I at least made progress and eventually got the look I wanted. Once I got over the hurdles, the rest of the interface implementation was smooth sailing.

5.3.3 Maze Generator Debugging

The original algorithm I planned to use for random maze generation was quite simple to implement, however, it did not work as expected. Therefore, I was forced to use a different algorithm. I eventually settled on Prim's algorithm. As I implemented the algorithm, the code became quite complex and I managed to introduce quite a few bugs in the process of writing it. I spent lots of time debugging this module: fixing one bug would introduce a slew of other bugs. After several hours, I managed to squash the last bug and I couldn't be happier with the results.

5.4 Actual Work Schedule

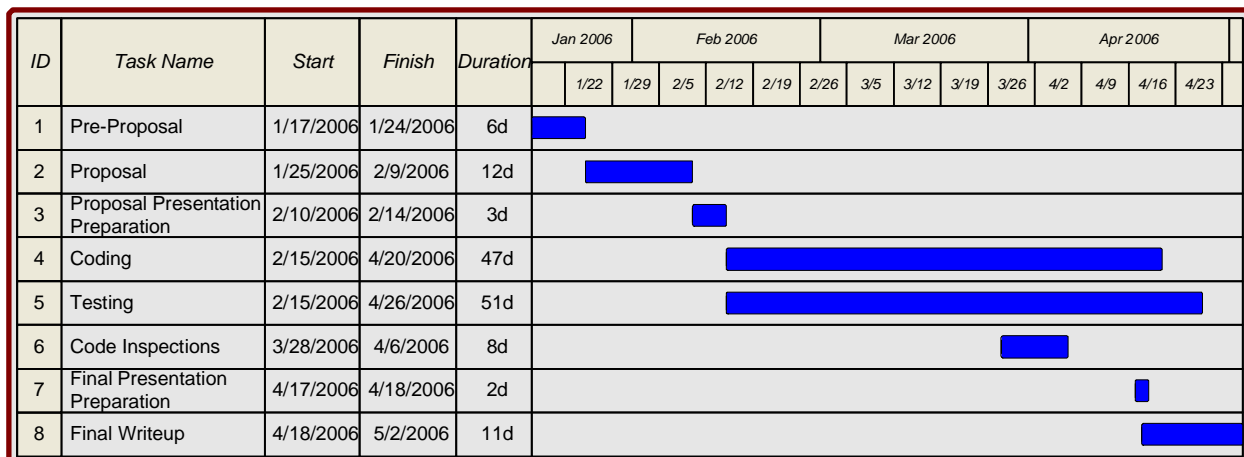
5.4.1 Pie Chart



Estimated Time Spent: 300 hours

Write-up: 45 hours
 Presentation: 15 hours
 Game Design: 90 hours
 Coding: 60 hours
 Testing and Debugging: 90 hours

5.4.2 Gantt Chart



6. Analysis, Results, and Discussion

6.1 Did the Program Work?

After extensive testing, I have determined that the system works according to the minimal requirements specified. There does not appear to be any lingering bugs, however I do not feel like the game has reached its final stage of development. There remain many improvements I would like to make and many additional features I would like to add before I will feel like the game is truly complete.

6.2 What Worked Well?

I feel like everything I was able to implement worked well, but here are a couple of things that worked even better than I had hoped.

6.2.1 Random Maze Generator

Using Prim's algorithm to randomly generate mazes for the dungeon design worked out really well. I was unsure how it would turn out using wall and floor tiles, but it turned out just as I envisioned it. One of my biggest concerns was that it would take a while to generate and draw the larger mazes, but the algorithm does it all in a flash. As soon as you take your finger off the enter key, there it is.

6.2.2 Java's Swing Package

Using Java's Swing Package for the interface design also worked better than expected. There were some problems with the layout managers at the beginning, but once I got past that, I was able to use Swing to implement all the interface features I had planned for. I found that when using Swing, you can pretty much do anything you want with your interface.

6.3 What Needs Improvement?

There is a whole laundry of improvements that I would like to make to the game. Here is a list of a few of the more important ones.

6.3.1 Animation

Probably the most glaring hole in the game right now is the lack of any animation. When I thought about all the important features of a role-playing game, I felt that flashy visual effects and high end graphics were not high on the list. After all, there are many RPG's out there that lack superior visuals, yet are regarded as some of the greatest games of all time. I'll concede that basic animation is not a "flashy visual effect" by any means, but to someone working alone on a very strict time limit, with little experience in animation implementation using Java, it is. That being said, I have found some great tutorials for creating animation in Java and I plan to put them to use on this game in my spare time, once the semester is over.

6.3.2 Interface Polish

Although I had much success using Swing to implement the interface, there are still a few additions I would like to make. For starters, I would like to make the interface more keyboard-friendly by adding key listeners to many of the components, that way the user could play the game without the need for a point-and-click device. I should probably also add roll-over features to the components, like changing the cursor from an arrow to a hand when it is rolled-over a clickable region. There are also some component focus issues (very minor bugs) I would like to hammer out. Finally, I should add pop-up warning messages that appear when, for instance, the player tries to drop or sell an item, or when they click the close icon in the top-right corner, that way they have a chance to cancel the resulting action if they clicked the button by mistake.

6.3.3 Story Line

As of now, the game has absolutely no story line what so ever. One of the most important elements to any great role-playing game is the story line. An RPG usually gets terrible reviews when it lacks a good plot. Some modern RPG's, like the entire "Final Fantasy" series of the past decade, have raised the bar so high that RPG story lines are now more like movie scripts. However, from a purely computer science perspective, a good plot is of no relevance, so I simply chose to exclude it for now.

6.3.4 Boss Monsters and End Game

Another seemingly gaping hole is the lack of boss monsters. This also means there is no end game situation either. Conceptually, you could just play until the end of time and never battle the final baddie, therefore "beating the game." There is a cap at level 50 and 50 floors to the dungeon, so I suppose you could say you've "beat the game" once you attain the maximum level or reached the end of the final maze, but most single-player RPG's require more than that. It's not that I couldn't have implemented this, or that I didn't want to, I just simply ran out of time.

6.3.5 Quick Maze Traversal Issue

Another issue I would like to fix is the player's ability to take advantage of the randomly placed entrance and exit of each maze by repeatedly exiting and re-entering a floor until the exit is very close to the entrance. This makes dungeon traversal too easy. They could also use the same technique to rack up lots of treasure, since chests are randomly placed as well and reappear when you exit and re-enter a maze. One solution I have considered is adding boss monsters at each exit, which was my original plan to begin with. These bosses would level-up when the player does and reappear each time you re-enter the maze, providing the player with a challenging obstacle every time they try to delve deeper into the dungeon.

6.3.6 Number Balancing

Role-playing games are all about numbers. One of the more challenging tasks of creating a good RPG is balancing those numbers so that the game is not too easy, yet not too hard. For instance, maybe an item is too expensive and needs for its price to be reduced, or a certain monster is way too weak for its level and needs its stats upgraded to provide more of a challenge, etc. It usually requires hours of testing by large groups of testers to get these kinds of numbers balanced just right. With the relatively limited time I have had for testing, I feel pretty comfortable with all the numbers at the moment. However, I have not had a chance to test the later stages of the game, so I'm sure there is still some fine tuning to be made. One solution would be to post the game on the web for download and run a beta test. People would download the game, play it for a while, and then hopefully send me their feedback, which I could then use to make the appropriate adjustments.

7. Lessons Learned and Conclusions

7.1 Lessons Learned

7.1.1 Swing layout managers are more trouble than they are worth

I had lots of success using Swing to implement the interface, but I got hung up for hours messing around with the layout managers. I should have just decided to place the components manually after the first hour or so, even though it took a lot more coding, which is what I was trying to avoid. After all, the fastest route from point A to point B isn't always the straightest.

7.1.2 Don't bite off more than you can chew

Although this is a fairly low-scale game, there are a huge number of items, spells, and monsters, and I spent a lot of time designing them. Considering the strict time limit, I would have been better off creating about a third of the items, spells, and monsters, and using the extra time to add more features and improve on the existing ones.

7.1.3 Don't waste time dwelling on a module that doesn't work perfectly

When programming, I sometimes get tunnel vision while working on a module. When it comes time to test it, if the module doesn't work exactly how I want it, even if the bugs are very minor, I tend to focus on fixing it before working on anything else. Rather than wasting time racking my brain, trying to figure out why it isn't working as it should, I need to just move on to something else and come back to it another time. I also discovered that rather than staying up all night trying to fix a bug, sometimes it's better to just sleep on it. On more than one occasion, I woke up the next morning, my mind refreshed, and while I was eating breakfast or brushing my teeth, a light bulb would turn on in my head and I would realize what the problem was.

7.2 Conclusion

In conclusion, developing the game was a very fun and rewarding experience. Unlike other programming assignments and projects, I found myself eager to get to work, rather than procrastinating until the last possible moment. Not only did I get to use all the software development skills I have already acquired while enrolled in the CS program, but I got to hone those skills even more. And most of all, I proved to myself that video game design really is a career I would enjoy. If I can secure the funding, I would like to work on my master's degree at one of the graduate level video game design schools that have been popping up all over the country as of late. Developing games for a living would be an absolute dream come true.

8. References

- 8.1 Birlew, D. (2006). Dragon Quest VIII: Journey of the Cursed King – Official Strategy Guide. Indianapolis: Pearson Education.
- 8.2 Hollinger, E. (2001). Dragon Warrior III: Prima's Official Strategy Guide. Roseville, CA: Random House.
- 8.3 Hollinger, E. (2001). Dragon Warrior VII: Prima's Official Strategy Guide. Roseville, CA: Random House.
- 8.4 Horstmann, C., & Cornell, G. (2003). Core Java 2: Volume I – Fundamentals. Palo Alto: Sun Microsystems Press.
- 8.5 Gray, J. (2001). Images: Tile Sets. Dragon-Warrior.com. Retrieved April 30, 2006, from the World Wide Web: http://www.dragon-warrior.com/Images/Tile_Sets.shtml
- 8.6 Gray, J. (2001). Treasure Horde: WAVs. Dragon-Warrior.com. Retrieved April 30, 2006, from the World Wide Web: http://www.dragon-warrior.com/Fan-Made_Games/WAVs
- 8.7 Muller, H., & Walrath, K. (2006). Using Timers in Swing Applications. Sun Developer Network. Retrieved April 30, 2006, from the World Wide Web: <http://java.sun.com/products/jfc/tsc/articles/timer>
- 8.8 Sun Microsystems, Inc. (2004). Java 2 Platform Standard Edition 5.0 API Specification. Java Technology. Retrieved April 30, 2006, from the World Wide Web: <http://java.sun.com/j2se/1.5.0/docs/api/index.html>
- 8.9 Sun Microsystems, Inc. (2005). The Java Tutorial: Playing Sounds. Java Technology. Retrieved April 30, 2006, from the World Wide Web: <http://java.sun.com/docs/books/tutorial/sound/playing.html>
- 8.10 Wikimedia Foundation, Inc. (2006). Maze generation algorithm. Wikipedia. Retrieved April 30, 2006, from the World Wide Web: http://en.wikipedia.org/wiki/Maze_generation_algorithm
- 8.11 Woodus (2002). Music & Sounds. Dragon's Den. Retrieved April 30, 2006, from the World Wide Web: <http://www.woodus.com/den/music/midi.php>