The Zeven Trials

The Beta Boyz Of C-Block

Team Chronos

Ross Visser, Luke Leontowich & Shannon Abeda

Due: Feb 17, 2020

TABLE OF CONTENTS

Introduction

Our software will run our text based adventure game The Zeven Trials

The Zeven Trials

The Zeven Trials is a single player semi-randomly generated dungeon crawler game. The game will consist of 7 floors and the player wins the game upon the completion of the 7th floor

- There is no scoring system for this game or experience or leveling up
- Instead of leveling you get increased armour and weapon strength

The Seven Floors

Each floor will consist of an enemy fight as well as a puzzle to move onto the next floor

Fight Phase

- Each fight will be randomly selected between one of three enemies for that have been predetermined for that floor
- The enemy's will have a random weapon assigned that is appropriate for the floor level
- After each fight the player will be given a choice between 3 items (weapon, armour or shield) that will replace the current weapon, armour or shield they currently have equipped
- If the player dies (HP <= 0) then the game is over and the player must restart and try again
- The fight starts with the game declaring how much damage the enemy intends to attack for (randomly generated based on their enemy's weapon), that way the enemy doesn't deal the same amount of damage every turn and adds an element of strategy. The player can then choose to try to block the damage with their shield and armour stats, or let their armour alone negate some of the damage and attack the enemy.

Puzzle Phase

- There will be a different style of puzzle for each floor with randomly generated aspects, such as a word scramble from a selected bank of words or picking the directions through a maze with a random solution. The solution take in input from the user, and the puzzle will tell the user their choices or how to solve the puzzle (entering the right phrase, entering directions in the right order),
- The puzzle could have a clue attached to it like a riddle or be a trial and error puzzle (maze)
- The puzzle must be completed to move onto the next floor

PROJECT MANAGEMENT

Team Organization:
Shannon Abeda: Developer, Tester & Document Control
Ross Visser: Team Lead, Developer & Main Designer
Luke Leontowich: Developer & Quality Assurance Lead

Team Roles

Team Member	Design	Implementation	Testing Maintenance
Ross	Phase Lead	Librarian	QA Lead
Luke	QA Lead	Phase Lead	Librarian
Shannon	Librarian	QA Lead	Phase Lead

Risk Management

Our team had prior experience in software development with an intro project given by Dr. Anvik. As a team, we all collected our thoughts and shared our experiences as to what went well and could be better for this project. One of the biggest concerns that was raised was boundaries and direction with the previous group. Given our first meeting, we discussed and established boundaries and issues that may arise. These are not limited to:

Communication

At our first meeting, we established a clear communication guideline and understanding each group members expectation and different personality traits that might conflict with other members.

Work Habits

Our group has created min work time for each day/week where each member has to contribute a certain amount of time towards the project. They're expected to log their time and document their work. During the reading break, our group members are expected to contribute 2-3hrs per day for testing & implementation

Expectations

Each member is expected to contribute to each stage of the development. There is no metric established as each member has different schedules however contribution is expected during each group members free time.

Scheduling

We have created an excel sheet with weekly sheets where each member is expected to log their free time and when we can meet up as a group to work. This will

Conflicts

Our group has discussed possible conflicts that may arise given our experiences, unexpected events and our demanding schedules (eg: work, training & other classes). If a communication problem were to arise (eg: lack of contribution, member conflicts, etc) as a team we would sit down and understand what the issues are and come to a resolution. If one is not met, then we escalate and talk to Dr. Anvik.

Unexpected Events/Changes

Our team has established protocols in case of unexpected events or changes (eg: group member drops the course). During the reading break, our team will complete a majority of our implementation to lessen the load for further development of our game. This would offset the risk if a group member cannot contribute to the project and/or drops the course

Lack of Experiences

All of our group members are proficient in C++. At this point, there are no outstanding issues. As we progress through the course and learn new development tools, there is potential for issues pertaining to application of these tools. If a group member is struggling with their assigned tasks, as a group we would assist them if needed but also expect them to put their own effort into research.

Timeline

Our project will be divided into three stages

- **Implementation**
- **Testing**
- **Maintenance**

The expected allotted times for each phase are:
Implementation: 1-2 weeks
Testing: 1 Week
Maintenance: 1 Week

Given our current timeline, we're project to have a majority of the project completed 1-2 weeks prior to the given deadlines for each phase. This will allow us the flexibility if any issues were to arise provide us with as much code coverage as possible before developing the game class.

Development Process

Code Review Process

- Team members proposed changes will go through the phase lead for each phase, addressing how they think their idea or change will improve the project
- The phase lead will then decide if their idea will improve the project or add unnecessary confusion to the project and the idea will either be implemented, revised and changed to be implemented or scrapped
- Any large ideas or major changes will need to be discussed in team meetings and everyone will have a say in the status of the idea

Communication Tools

- Our group has created an iPhone text message group that will be used for the bulk of team communications
- We will also be using our slack group for other modes of communications in the event that someone's phone becomes broken or other unforeseen events

Change Management

- The QA lead will be responsible for all bug reports and assignment of who will fix the bug, typically the person who wrote the code will be in charge of fixing the bug, unless there are unforeseen circumstances in which someone else will be tasked with fixing it
- If the bug occurs from two people working on the same section of code then both of the people responsible will be forced to work together to come up with a solution to the problem
- The individual that handles and fixes the bug report will then comment on the status of the bug and give a brief description of what they did to solve the issue

Software Design

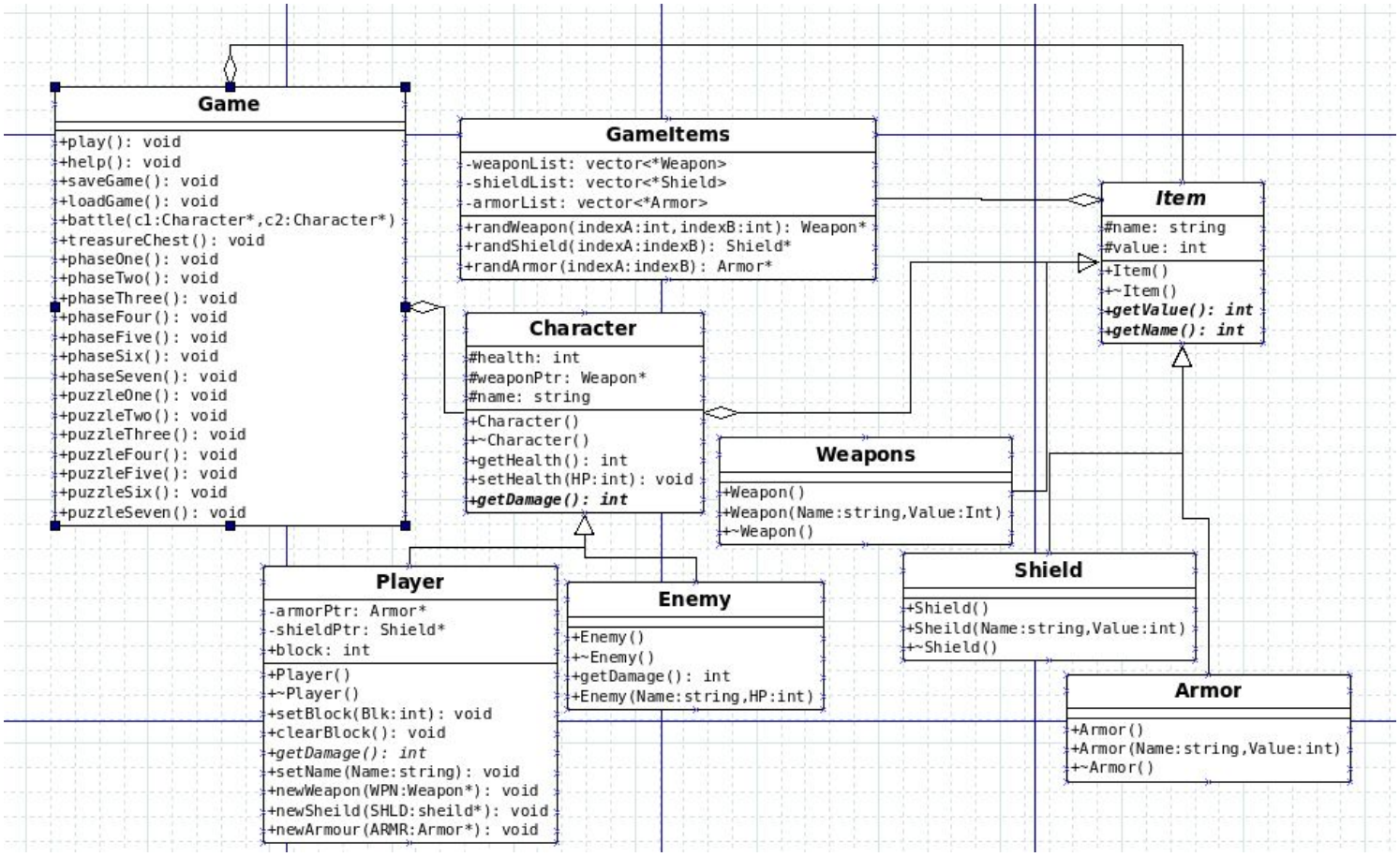
Starting for each of the games
Every game will start off the same way having the player on the first floor, having the basic items (weapon, shield armour) and will have to build their run from there.

A turn for each of the games
The player will only have turns during the fighting phase of the game where they can choose to block for added defence or attack the enemy. Otherwise the player will just input directions or commands based on the puzzles they encounter.

Saving The Game
The game will be saved by generating a file containing the players current floor (the game will not save where the player is on a floor, if you save in the middle of a fight you will have to start the floor all over again), their items (armour, weapon, shield), the players name and their current health.

Loading The Game
Loading the game the program will take in the generated file (as long as the player has a save file) and create the character with all of their items, health, name and the last floor they cleared

APPENDIX


Game
+play(): void
+help(): void
+saveGame(): void
+loadGame(): void
+battle(c1:Character*,c2:Character*)
+treasureChest(): void
+phaseOne(): void
+phaseTwo(): void
+phaseThree(): void
+phaseFour(): void
+phaseFive(): void
+phaseSix(): void
+phaseSeven(): void
+puzzleOne(): void
+puzzleTwo(): void
+puzzleThree(): void
+puzzleFour(): void
+puzzleFive(): void
+puzzleSix(): void
+puzzleSeven(): void
GameItems
-weaponList: vector<*Weapon>
-shieldList: vector<*Shield>
-armorList: vector<*Armor>
+randWeapon(indexA:int,indexB:int): Weapon*
+randShield(indexA:indexB): Shield*
+randArmor(indexA:indexB): Armor*
Item
#name: string
#value: int
+Item()
+~Item()
+getValue(): int
+getName(): int
Character
#health: int
#weaponPtr: Weapon*
#name: string
+Character()
+~Character()
+getHealth(): int
+setHealth(HP:int): void
+getDamage(): int
Weapons
+Weapon()
+Weapon(Name:string,Value:Int)
+~Weapon()
Player
-armorPtr: Armor*
-shieldPtr: Shield*
+block: int
+Player()
+~Player()
+setBlock(Blk:int): void
+clearBlock(): void
+getDamage(): int
+setName(Name:string): void
+newWeapon(WPN:Weapon*): void
+newSheild(SHLD:sheild*): void
+newArmour(ARMR:Armor*): void
Enemy
+Enemy()
+~Enemy()
+getDamage(): int
+Enemy(Name:string,HP:int)
Shield
+Shield()
+Sheild(Name:string,Value:int)
+~Shield()
Armor
+Armor()
+Armor(Name:string,Value:int)
+~Armor()