Software Requirements Specification (SRS)

Dungeon Quest

Team: Group 7

Authors: Greg Kaplowitz, Nicholas Miceli, Derrick Lor, Wyatt LaRose, Alberto Chavez

Customer: 6th Graders

Instructor: Dr. James Daly

**1 Introduction**

This is a software requirements specification document (SRS) outlining the necessary details about our proposed educational game. In this document, we will describe our development process including tools we are using and the requirements of our product.

## Purpose

The purpose of this document is to establish an understanding of requirements between the clients, users, stakeholders, and developers of our proposed educational game. All participants involved in this software project must agree to this document, and hereby follow the established outline to the creation, documentation, and distribution of the proposed educational game. It’s intended audience are the developers, customers, and any other stakeholders in Dungeon Quest’s production.

## Scope

The software product being developed is titled “Dungeon Quest”. Dungeon Quest is aimed at the target audience of 6th grade students. It is an educational game meant to be a learning aid, not replacement, of core curriculum taught at public schools regulated by the Massachusetts Department of Elementary and Secondary Education.

The benefits of Dungeon Quest include the ability to have students be asked a - variety of questions depending on the desires of the educator. Players will have the option to decide which subject out of Math, History, or English they would like their questions to be based in.

The objective of Dungeon Quest is to provide a simple yet engaging experience for students to be able to reinforce the topics that are covered in their classroom. This includes reasonably paced gameplay for students in the 6th grade age range as well as questions drawn from the core curriculum provided by the Massachusetts Department of Elementary and Secondary Education.

Dungeon quest will include game sessions that can reasonably be completed within a standard class block. Game sessions are meant to be played during the class and progress is not to be carried over throughout multiple class sessions. The game will give players the option to select what subject they would like to answer questions for and will provide more difficult questions the deeper the player is into their current session. Dungeon quest will keep track of player’s score throughout the session and display it upon completion (game over or game won) so users can aim to better their scores during future sessions.

## Definitions, acronyms, and abbreviations

Definition list of all the terms, acronyms, and abbreviations in this document.

\*Map = the selection screen where players choose which dungeon to play.

\*Dungeon = the level and the specific genre covered in that level.

\*Path = the direction the player can move to.

\*Encounter = the battle between players and enemies.

\*User = the person interacting with the software

\*Enemy = the entity that the user is fighting against

\*SRS = Software requirement specification

\*UI = User Interface

\*JSON = JavaScript Object Notation

\*SW = Software

\*HW = Hardware

\*MDoE = Massachusetts Department of Education

\*OS = Operating System

\*CPU = Central Processing Unit

\*RAM = Random Access Memory

\*GPU = Graphics Processing Unit

## Organization

Going forward, the rest of the SRS will cover topics pertaining to Dungeon Quest’s interface design, functionality, user expectations, possible constraints, requirements, models, diagrams, and finally points of contact.

2 Overall Description: Describes the content of section 2.

2.1 Product Perspective: Describes our product and its requirements for making a fun game that is also educational.

2.2 Product Functions: Describes how the game plays.

2.3 User Characteristics: Features that users are expected to have.

2.4 Constraints: Limiting factors on the development of the game.

2.5 Dependencies: Hardware and software that the development of and/or ability to play are reliant on.

2.6 Apportioning Requirements: Features of the product that will not make it into the initial release.

3 Specific Requirements: List of all necessary abilities of the software.

4 Modeling Requirements: Collection of development diagrams.

4.1 Use-Case Diagram: displays different use cases of the game.

4.2 Class Diagram: Displays attributes and methods of classes, as well as their relations.

4.3 Sequence Diagram: Illustrates the possible sequences that can be taken when interacted with the software.

4.4 State Diagram: Shows the state transitions of the game.

5 Prototype: Describes the layout of the developed prototype

5.1 How to Run Prototype: hardware requirements and instructions on where/how to download and play the game

5.2 Sample Scenarios: Describes an example run-through of the game,

6 References: citations

7 Contact information

**2. Overall Description**

This section will cover the context of the product, the major functions it will perform, the characteristics of the expected users, constraints on the software’s design, assumptions during production, and apportioning of requirements.

**2.1 Product Perspective**

Dungeon Quest is designed to provide a fun environment where students can test their knowledge while playing an interactive game. Research has been done to support the fact that “using games in teaching can help increase student participation, foster social and emotional learning, and motivate students to take risks” [2] and Dungeon Quest will be no exception. By including questions based on 6th grade core curriculum set by the Massachusetts Department of Education and gameplay that incorporates fantasy elements kids are known to love, Dungeon Quest is the perfect game for students to improve their skills.

The interface needs to be kept simple so that children will have an easy time interacting with it. Dungeon Quest’s creation is constrained by Unity as that is the platform that has been chosen to develop with, but it offers many different tools that can be used to help the product reach its full potential. Unity Collaborate is being used to facilitate production which supplies 25GB of cloud storage space to develop collaboratively which constrains the size of the game; however, Dungeon Quest will fit within these bounds. Finally, the current build of Dungeon Quest is being produced to run on Windows, however future iterations will include support for other platforms.

Graphical user interface, application, website

Description automatically generated

2.2 Product Functions

The player will start Dungeon Quest and choose the subject they would like to have their questions be about. From there, they will be given a map that consists of branching paths. Starting from the root of this tree-like structure they will pick one of the child paths. Choosing a path will take the user into the battle screen of the game where the educational aspect takes place.

During each battle, the user will be asked questions relating to the 6th grade core curriculum from MDoE. By answering correctly, the user’s character will attack the enemy and cause them to lose health. On the other hand, if the user answers incorrectly, the enemy will attack the user’s character causing the user to lose health. The encounter continues until either the user’s character or the boss is out of health. In the former case, “you lose” will be displayed to indicate that the user has lost, in the latter case the “you win” will appear with a button to return to the map to select their next encounter. If the user beats the encounter, the next encounter will be displayed indicating that the user has access to the next pathway.

User’s scores will be tracked throughout a gameplay session and will increase for each question the user answers correctly. Once the user reaches the game over screen, the score will be reset, and progression in each of the dungeons will be reset.

This Diagram displays Dungeon Quest’s High-level goals:

Diagram

Description automatically generated

The three main goals of testing student’s knowledge, supplementing teacher’s lessons, and encouraging students to replay combine into our main goal of creating an intuitive and enjoyable gameplay experience.

**2.3 Assumptions and Dependencies**

The expected grade level of the users of Dungeon Quest are 6th graders (approximately 11 years old). The users are expected to have enough foundational knowledge through typical teaching styles to have the ability to answer the questions asked of them in Dungeon Quest. The user ideally should have enough experience with multiple choice style questions to rule out incorrect answers and arrive at a reasonable answer to the questions provided. Users will be provided with different subject options so they will be able to choose the subject that best suits them.

Users are expected to have some experience interacting with computers to be able to play the game. They will require the motor functions needed in order to use a mouse or trackpad and will also need to be sighted in order to successfully interact with Dungeon Quest’s graphical interface. Finally, users should have the decision-making abilities necessary to navigate Dungeon Quest’s maps to seek out the end of the dungeon.

2.4 Constraints

One constraint on the project is the memory allotted. Unity teams provide 25 GB of cloud storage to be used for Dungeon Quest which constrains how large the game can be. That said, Dungeon Quest will fit within these bounds. Also, only 5 members are allowed on a Unity team which constrains how many members can work on Dungeon Quest.

Another constraint is the hardware that the game will run on. The current build of Dungeon Quest is intended to be run on a Windows machine. Future iterations may provide builds for other platforms.

Finally, Dungeon Quest’s production is constrained by time. Given that only a couple of months are given to complete the project, there are certain features that will need to be trimmed to meet deadlines.

2.5 Assumptions and Dependencies

One assumption being made is that the computer running Dungeon Quest will have a mouse or trackpad to interact with the game. Also, this computer will have Windows operating system to run the game. The device must have the hardware specifications detailed in section 5.1 of this SRS document to successfully run the game.

The production of Dungeon Quest is dependent on Unity’s game development environment. Unity was selected because of its collaborative abilities using Unity teams as well as it being free to use.

Users of the game will not need to have access to Unity, as the game will be packaged as an executable.

2.6 Apportioning of Requirements

Our biggest requirement that we probably won’t be able to do because of time is implementing a powerup system to alter how combat works. Currently our combat system is just answering questions, we might have both multiple choice and short answer but currently the requirement is for multiple choice. In future iterations of this game, we could make combat more interesting for users so that it isn’t simply based on answering questions.

Another requirement that is beyond the scope of the current iteration of our project is the ability for teachers to input their own questions for their students to be asked when playing through Dungeon Quest. This functionality would allow educators to fine tune the game to ensure that the questions being asked fit the scope of their lesson plan and would allow teachers of other grade levels (other than 6th) to utilize Dungeon Quest in their classrooms as well.

# Specific Requirements

1. There shall be a main menu from which the game, “Dungeon Quest”, will begin when the user launches it
2. The menu will be dungeon themed with the name of the game as well as a way for users to navigate to other screens.
3. There will be an “about” page that users can navigate to for credits.
4. There will be a way for users to navigate from the main menu to a new game playthrough.
5. The menu will have a way for users to quit the game.
6. The page at the start of a new playthrough will display different options for different subject.
7. The user must then select which subject they would like to play (English, History, Math)
8. After the user selects their subject, the game will begin.
9. There will be a bank of questions for each subject
10. Questions will be four option multiple choice.
11. Questions are for 6th graders
12. The game will begin with the user’s character in the dungeon selection screen and a tree-like structure, branching out towards the dungeon’s end.
13. The user will begin with 10 lives (maximum)
14. The user will choose which path in the dungeon to take
    1. The selected paths will determine how the user navigates towards the end of the dungeon
    2. If a level can be reached by two paths, the user must complete the levels from both incoming paths before progressing.
15. Once a level has been selected, the user will enter an encounter with an enemy.
    1. Within the encounter, the user will be asked a question from the subject they selected.
       1. If the user answers correctly, the boss will be defeated, and the user will be returned to the dungeon.
       2. If the user answers incorrectly, the player will lose health. Three incorrect answers will result in the player’s death. Then, the player is returned to the map with one less life.
16. When an encounter is successfully completed, the next level(s) on the map will be able to be selected by the user.
17. If the user runs out of lives at any point, they will be sent to the start of each of the dungeons. Progression in all dungeons will be reset.
18. The score will be based on how many encounters a user successfully completes (Max 10/10).
19. The “about” page will feature information about the creation of the game.
20. The about page will describe the reason behind the development of the game.
21. The about page will display pictures of each team member and describe their role in Dungeon Quest’s development.
22. Credits assets used, and links to find those assets.
23. Link to Dungeon Quest website.

**4 Modelling Requirements**

Use Case Diagram

The diagram shows the options the player actor has in the game. The player can quit which takes them out of the game, view the about page, or start the game which calls the other cases which set up and start the game. Within the game the user encounters enemies and answers questions.

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Use Case Name: | View About Page |
| Actors: | Player |
| Description: | The Player clicks on the “About” button on the main menu and is directed to the project’s website |
| Type: | Primary |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 5 |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Start Game Session |
| Actors: | Player |
| Description: | The Player clicks on the “Start” button on the main menu and begins a session of the game |
| Type: | Primary and Essential |
| Includes: | Select Difficulty/Dungeon |
| Extends: | None |
| Cross-refs: | Requirement 2 |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Exit Game |
| Actors: | Player |
| Description: | The Player clicks on the “Quit” button on the main menu and the game is exited |
| Type: | Primary |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 1b |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Select Subject/Dungeon |
| Actors: | Player |
| Description: | The Player selects which subject they would like to have their questions pulled from (Math, History, English) |
| Type: | Primary and Essential |
| Includes: | Encounter Enemy/Answer Question |
| Extends: | None |
| Cross-refs: | Requirement 2a |
| Uses cases: | Player must first click the start button to be able to select their subject |

|  |  |
| --- | --- |
| Use Case Name: | Encounter Enemy/Answer Question |
| Actors: | Player |
| Description: | The Player battles an enemy by answering questions |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 4c |
| Uses cases: | The player must have started a run of the game and selected their subject and a level to play |

|  |  |
| --- | --- |
| Use Case Name: | Answer Correctly |
| Actors: | Player |
| Description: | The Player chooses the correct answer to a question asked in an enemy encounter and then adjusts enemy health accordingly |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | Encounter Enemy/Answer Question |
| Cross-refs: | Requirement 4.c.i.a and 4.c.ii.a |
| Uses cases: | The player must have started a run of the game and selected their subject |

|  |  |
| --- | --- |
| Use Case Name: | Answer incorrectly |
| Actors: | Player |
| Description: | The Player chooses the incorrect answer to a question asked in an enemy encounter and then adjusts player health accordingly |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | Encounter Enemy/Answer Question |
| Cross-refs: | Requirement 4.c.i.b and 4.c.ii.b |
| Uses cases: | The player must have started a run of the game and selected their subject |

Class Diagram:

The class diagram shows what objects and scenes hold what attributes. It also shows how each class interacts with each other. For development it gives us an idea of what we know each scene is going to need and how we are going to transition between them.

Graphical user interface, application

Description automatically generated

Data Dictionary:

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| MainMenu | | This class represents the main title screen that will be displayed upon launch of the game |
| Attributes | background : Image | The image that will be displayed behind the text and buttons on the Main Menu |
|  | title : String | The title of the game “Dungeon Quest” that will be displayed on the Main Menu |
|  | startGame : Button | A button on the Main Menu that will begin the actual gameplay |
|  | about : Button | A button on the Main Menu that will open the website for our project |
|  | quit : Button | A button on the Main Menu that will exit the game altogether |
| Operations | startGameClicked() : void | Called when the StartGame button is clicked, begins the gameplay |
|  | aboutClicked() : void | Called when the About button is clicked, displays project’s website |
|  | quitClicked() : void | Called when the Quit button is clicked, exits the game |
| Relationships | The main menu begins an instance of “Player” when the game is started. It is also displayed when a player quits during/after their run of the game. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Player | | This class represents the player in the game and contains attributes that would carry with them throughout the encounters. |
| Attributes | numLives : int | The current amount of lives a player has, how many more times they can be hit by the boss before game over |
|  | currentNode : int | The current node that a player is at |
|  | currentDungeon : string | The current dungeon the player is at |
|  | gameData ; int[] | Holds player data variables in the form of integers |
| Operations | numLivesZero() : Bool | Checks to see if the current value of NumLives is 0 |
|  | gameOver() : void | Displays game over screen if num lives reaches 0 |
| Relationships | Is created when start game is selected from the main menu. Views the map of the levels and displays the game over screen when necessary. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Map | | This object represents the map of levels that the player will go through during the game |
| Attributes | mapBackground : Image | The image that will be displayed behind the levels and other map icons |
|  | currNode : LevelNode | The node of the map that the player is currently standing on |
|  | winNode: LevelNode | The final node of the map that once beaten indicated the player has won |
|  | nodeList: list | Structure to hold all level objects of the map |
| Operations | playCurrNode() : void | Begins the level associated with the node that the player is currently standing on |
|  | moveLeft() : Bool | Moves the player to the node to the left of them in the dungeon, returns true if move was successful |
|  | moveRight() : Bool | Moves the player to the node to the right of them in the dungeon, returns true if move was successful |
|  | atWinNode() : Bool | Checks if the current node is the win node |
|  | currNodeBeated() : Bool | Checks if the current node’s “Beaten” flag is set to true. |
| Relationships | The map is composed of a graph of multiple LevelNodes. It is viewed by the Player and plays the levels associated with its nodes. Finally, it displays the GameWon screen if the current node is the win node and it has been beaten | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| GameOver | | This object represents the game over screen that will be displayed if the player runs out of lives |
| Attributes | gameOverText : String | The main text that is displayed on the GameOver screen |
|  | score : int | The player’s score at the time of GameOver being displayed |
|  | background : Image | The background image displayed behind the other attributes on the GameOver screen |
|  | playAgain : Button | A button that begins a new gameplay loop |
| Operations | playAgainClicked() : void | Begins a new gameplay loop by creating a new Player object |
| Relationships | The GameOver screen is displayed by the Player object. It has the option to exit to the main menu or begin a new player instance and restart the game. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| GameWon | | This object represents the game won screen that will be displayed if the player successfully completes the WinNode level |
| Attributes | gameWonText : String | The main text that is displayed on the GameWon screen |
|  | score : int | The player’s score at the time of GameWon being displayed |
|  | background : Image | The background image displayed behind the other attributes on the GameWon screen |
|  | playAgain : Button | A button that begins a new gameplay loop |
| Operations | playAgainClicked() : void | Begins a new gameplay loop by creating a new Player object |
| Relationships | The GameWon screen is displayed by the map object. It has the option to begin a new player instance. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| LevelNode | | This object represents a single node on the tree that makes up the map of the dungeon |
| Attributes | parent : LevelNode | The LevelNode that leads to this node |
|  | left : LevelNode | The left child of this LevelNode |
|  | right : LevelNode | The right child of this LevelNode |
|  | beaten : Bool | Indicates if the level associated with this node has been successfully completed by the player |
| Relationships | The map has a tree of LevelNodes that the player can traverse throughout the gameplay. Each LevelNode is associated with a Level object that can be accessed and played from the node. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Level | | This object represents encounter where the player will be asked questions in order to defeat a boss |
| Attributes | bossHealth : int | The current health level of the boss within the encounter |
|  | player : Sprite | The sprite displayed on screen representing the player |
|  | boss : Sprite | The sprite displayed on screen representing the boss |
|  | background : Image | The background image displayed behind the sprites and other on screen attributes. |
| Operations | askQuestion() : void | Asks a question and displays it on the screen for the player to respond |
|  | answerCheck() : Bool | Called once an answer is provided, returns true is the answer is correct |
| Relationships | Each Level is associated with a specific LevelNode on the Map. The Map plays a Level when the player clicks on the LevelNode that it is associated with. The Level asks questions from the bank of questions to ask. | |

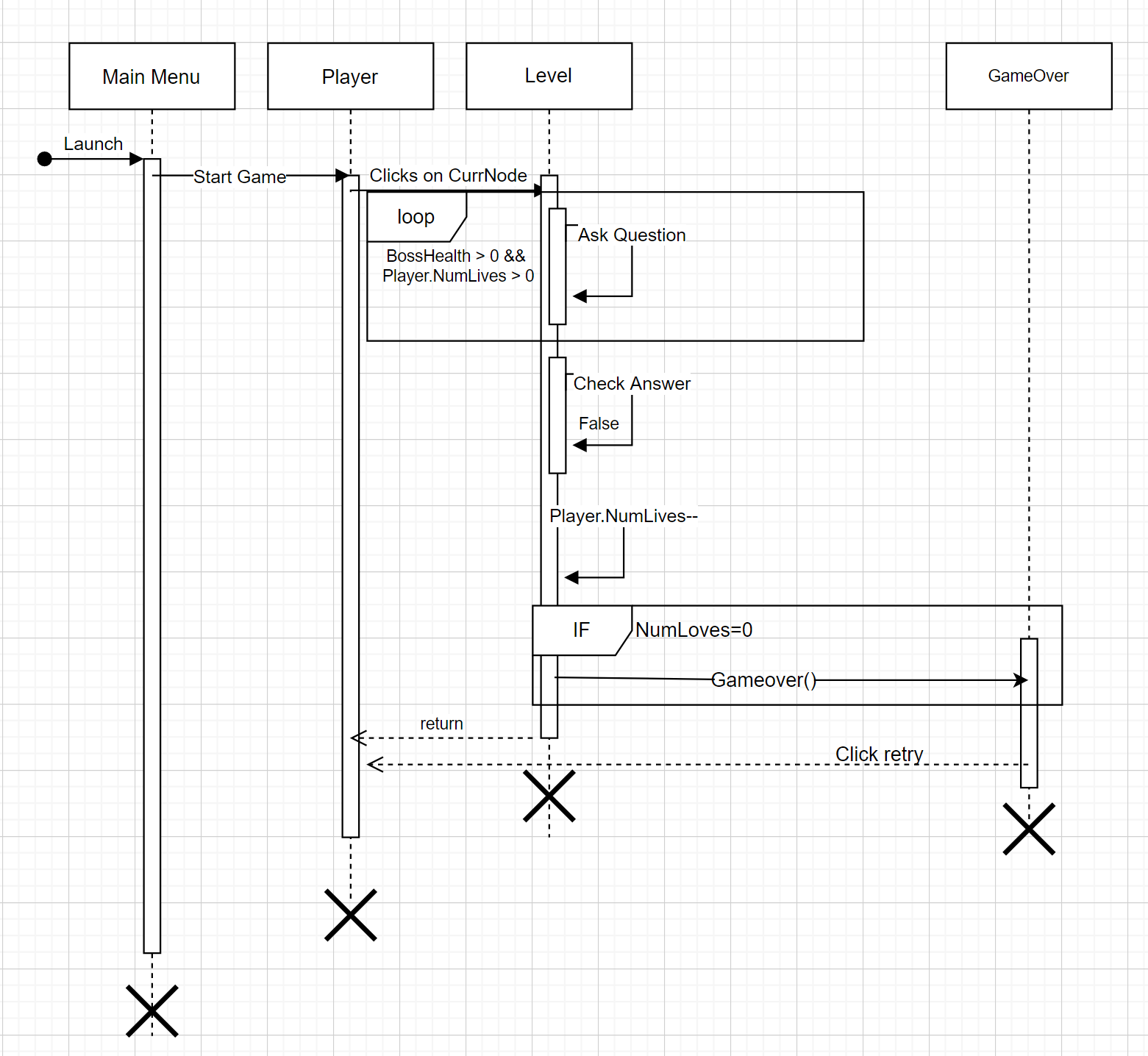
|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Question | | This object represents one of the questions that a Level can ask |
| Attributes | class : string | The class subject that this question is pertaining to |
|  | difficulty : enum | The difficulty level of this question |
|  | text : String | The actual text to display on screen in order to ask the question |
|  | answer : String | The correct answer for the question |
| Relationships | Question objects are asked by Level objects. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| MultipleChoice | | This object represents one of the questions that a Level can ask |
| Attributes | Array:  optionA : String  optionB : String  optionC : String  optionD : String | The possible answers that can be selected |
| Relationships | MultipleChoice is a subclass of the Question class. | |

Sequence Diagrams:

The user starts the game on the main menu screen. User is then able to select which dungeon to play. Upon selecting a dungeon, they are moved to the dungeon scene. The user then clicks on the LevelNode they are currently at (CurrNode) and the encounter for that node begins. Within the encounter, the user is asked questions if the boss’s health and the player’s number of lives remaining are above 0. Each answer provided by the user is then checked. If answer choice is wrong, the player’s health decrements. Once NumLives is 0, the level exits and returns. Upon return, the Player checks their life count to see if it is 0. Because it is 0, the Player then displays the GameOver screen, and the user clicks quit which returns to the Main Menu.

Sequence 1: Player starts game by entering encounter with enemy. Player loses the encounter resulting in player losing all lives, sending them to the game over screen.



Sequence 2: User views about page and then quits game.

User Launches the game and then on the Main Menu they click on the “About” button. This button opens the projects webpage in their browser. Once they return to the game, they click the quit button which exits the game.

Graphical user interface, text

Description automatically generated

State Diagram:

The state diagram shows how the system moves based on the input of the user. The system starts at the start arrow which goes into the main menu and has three possible state changes based on which of the three buttons the user clicks. If the user clicks start, then the system flows into the dungeon select and then into the map where the user picks encounters and fights enemies. This is the standard gameplay loop the system exists in until the game ends and sends the user back to the main menu.

Diagram

Description automatically generated

5 Prototype

The first prototype is a test of user interface design (UI) of the project. It will show an early build focusing on the basic start game loop and select dungeon/difficulty screen, encounters, and quit game functionality. The only hardware component used is the mouse.

Upon loading the game, it will direct the user to the main menu page of the game. The main menu page consists of the start game button, which leads to a scene changer subroutine that directs to the gameplay loop of the game, the quit game button, which exits out of the application, and finally the about button, which directs the user to the about page of our project.

Upon clicking the start game button with the left mouse click, a scene changer event listener will activate the next scene to be loaded. The map select scene is loaded, where the user will select the topic of the dungeon to play. To enter the dungeon, all the users must do is press the left mouse click while hovering over the icon of the dungeon. This will load the scene specific to the dungeon.

Included on every page except for the main menu and encounter is the back button. The back button will allow users to revert back to the previous scene. This will come in useful if the user ever finds themselves miss clicking on accident and needing to revert to a previous page.

The encounters scene is where the main gameplay loop is held. It accesses question data from JSON files associated with that dungeon. Each dungeon has their own set of JSON files that hold data such as questions, correct answers, and four options to choose from. A question is chosen based on the node the player chooses. More difficult questions will appear the further the player progresses. Once a player chooses an option and submits it, the scene will respond accordingly if the player choice is correct. On correct answer choice, the enemy is defeated, and a win screen is displayed, returning the player back to the dungeon. On wrong answer choice, the player takes damage equivalent to 33% of health bar. Once the player submits three wrong answer choices, a lose screen is displayed, decrementing the life counter, and returning player back to dungeon.

The quit game button, as it suggests, will close the game by calling scene changer event listener to exit the application. The quit game button saves any progress the user makes in the dungeons. Each game starts up builds on the last save.

5.1 How to Run Prototype

Minimum system requirements:

OS = Windows 7 or higher

CPU = Single core 2.0 Gigahertz

RAM = 2 Gigabytes

GPU = Intel Ultra High-Definition graphics or equivalent

Size = 102 Megabytes

Hardware = Mouse

Prototypes require a mouse as the minimum hardware component required to play.

Version one and two of the prototypes will have one executable (Project Dungeon Game.exe) that only runs on Windows operating systems. All the necessary files associated with the game will be packaged into a zip file (.zip) to allow for compression and easy distribution. The zip file will be listed on the project website, available for download.

To run prototype version one or two, download the associated zip file from the project website. Unzip the zip file into the current directory. Popular unzipping software tools are WinRar and 7zip, which are free to download online and use. Run the game by double clicking the .exe application.

Download version one and two prototypes at:

URL : https://derricklor.github.io/SWEProject/Website/documents.html

5.2 Sample Scenarios

Graphical user interface, website

Description automatically generated

## When the game started the users will see this page where they can start by pressing start, or exit by pressing exit, in this case let’s assume the users presses the start button.

A picture containing text, indoor

Description automatically generated

The user will then see this screen where they can choose between 3 separate dungeon options. The user can press any button which will determine what types of questions they will receive in the dungeon. Let’s assume the user selects the English dungeon.

Map

Description automatically generated

The user is then brought to this screen where they see the map. The whole map is visible so the user can see all the enemies.0Tthe goal is to defeat the boss without running out of lives. Since the user is at the start there are two possible encounter options so they will click one to start an encounter.

Graphical user interface

Description automatically generated

This is an example encounter screen with a cursed sword enemy. The user answers the questions by selecting their answer and submitting. This continues until either the enemy is defeated or the player gets three questions wrong.

Map

Description automatically generated

The player can continue fighting as long as their lives are greater than zero

**6 References**

1. D. Thakore and S. Biswas, “Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks,” Proceedings of IEEE Military Communication, Atlantic City, October 2005.
2. G. Kaplowitz, N. Miceli, D. Lor, W. LaRose, A. Chavez, “Dungeon Quest” (2021). https://derricklor.github.io/SWEProject/Website/index.html
3. H. Nguyen, “How to Use Gameplay to Enhance Classroom Learning,” Edutopia.org, March 2021.
4. Massachusetts Department of Elementary and Secondary Education. (2017). “Massachusetts Mathematics Curriculum Framework — 2017”. https://www.doe.mass.edu/frameworks/math/2017-06.pdf
5. Squire, Kurt. "Video games in education." *Int. J. Intell. Games & Simulation* 2.1 (2003): 49-62. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.543.5729&rep=rep1&type=pdf>

**7 Point of Contact**

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james\_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.