BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CS 319
Object Oriented Software Engineering Project
e-Bola
Analysis Report
Group 2B

Can AVCI
Deniz SİPAHİOĞLU
Ergün Batuhan KAYNAK
Esra Nur AYAZ

Course Instructor: Bora Güngören

October 7, 2017

Table of Contents

1. Introduction

1.1. **Purpose of the System**

e-Bola is a real time strategy - puzzle game, which consists of 4 levels. The main idea of the project is improving the main character's attributes with items found around the map, and by solving puzzles, and becoming powerful enough to defeat the boss, the deadly ebola virus.

The project hopes to create a game where the player will be tested in aspects such as quick decision making while facing with computer controlled enemies, and puzzle solving.

1.2. **Scope, objectives and success criteria of the project**

The created product of the project will be a video game called “e-Bola”.

The game will have small scale puzzles that the player will have to solve while fighting with enemies.

The project aims to deliver enough game features to achieve the desired strategy - puzzle environment. We deem the project successful if at least 1 feature is implemented in the items we specified in game features overview (sections 2.4.*).

1.3. **Outline**

This report will describe the system of the e-Bola project. First major section gives an overview of the game features. Here, the game elements are explained. The next major section is about the requirements. Functional requirements that the game has to achieve by combining the game features are demonstrated and nonfunctional requirements of the game are defined. Last major section consists of models of the game.

2. Overview

2.1. **Storyline**

The game starts with a cell membrane, that doesn't have any organelles in it. In the first level, the player wanders around to improve the cell accordingly with organelles, and proceeds to the second level. The cell solves some puzzles to get the

keys of locked treasure chests. This cell defeat others to gain nucleotides, which are the currency in the game. In the third level, the organism collects more nucleotides by battling small bacteria, and solving some puzzles. After a while, the organism gathers enough nucleotides to mutate itself, and to face the deadly ebola virus. In the last level, the organism has to defeat the virus by surviving a fierce battle, and after defeating it, the organism continues its life, peacefully.

2.2. **Gameplay and controls**

The player will use the up, down, right and left arrow keys to move around the map, access the menu, pause the game and resume the game. The player will have to use the space key to interact with the interactable entities and to attack the enemies. These features (including the map) will be explained deeply in the functional requirements. An example gameplay:

"You start the game as Celly, a little, young "cell", who only has a cell membrane! In order to defeat the evil bacteria and virus, he must evolve.

The game starts in a room, where there's nothing to be seen. There are portals to other rooms, but you don't know where they lead, of what will be waiting for you... You go to the room at the right first, and see that there are some organelles waiting for you. You quickly get a mitochondria, knowing that you will need it later on, for your stamina. You realise that getting some ribosomes wouldn't be bad at all for your vitality. You take these organelles, and continue to the room below. On that room, there is a giant treasure chest. You go near it, but it's locked. You realise that you need a key for the chest to open, and you realise that you need to find a key. You go to the room on the left, and find a key, waiting to be picked up. You pick the key, and go to the room with the treasure chest. The key fits the keyhole perfectly, and the chest opens. In the chest, there are 3 possibilities to choose from: a greater mitochondria, a ribosome, or some nucleotides. You will take the one you choose, and the other 2 will be destroyed. The ribosomes are the same with the ones you have, so you pass on the ribosomes. You already have a mitochondria, but this mitochondria is better than the one you have. Nucleotides are the currency of the game, and to improve yourself later on, you take the nucleotides. The mitochondria and ribosome are destroyed."

2.3. **Levelling**

There are 4 levels of e-Bola, and in each level, the game gets harder. In the first level, there are no enemies. The number of enemies increase throughout the game.

2.4. **List of Entities**

2.4.1. **Tiles**

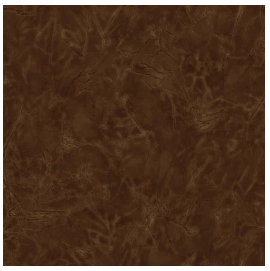
The game will take place at different environments, such as grass, rock and water. Each environment will have different effects on Celly.

2.4.1.1. **Grass**

When Celly moves in grass, his movement speed increases.

2.4.1.2. **Rock**

When Celly moves in rock, his movement and therefore dodging skills decrease.

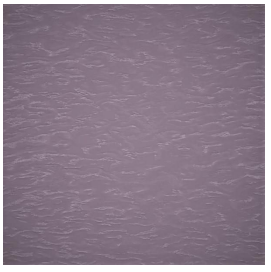


2.4.1.3. **Water**

When Celly is in a body of water, his vitality slowly increases.

2.4.1.4. **Neutral**

Neutral tiles are the default tiles in the game, and have no effect on Celly.



2.4.2. **Interactable**

There are 3 types of interactables in the game. The chests, keys and portals. The player can use keys to open treasure chests, and portals to proceed to the next room.

2.4.2.1. **Organelles**

Organelles are used to improve the attributes of Celly. Celly can only carry 2 organelles, so the player should choose between the best organelle that Celly can carry.

2.4.2.2. **Nucleotides**

Nucleotides are the currency of e-Bola. The nucleotides found throughout the game can be spent later on to improve Celly's attributes.

2.4.2.3. **Chest**

The treasure chests might contain nucleotides, organelles or even enemies! Some of them require a specific key, and some of them don't.

2.4.2.4. **Key**

There is a specific key for the each one of the treasure chests, that require a key.

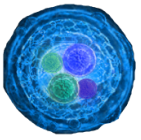
2.4.2.5. **Portal**

The portals are used to proceed to the next room. It is enough for Celly to run through the portals.

2.4.3. **Alive**

2.4.3.1. **Celly**

Celly is the name of the main character of e-Bola.



2.4.3.2. **Enemies**

There are 2 different virus that try to attack Celly. Some of them use ranged weapons and the others chase Celly, and try to kill him to steal his organelles.

2.4.3.3. **EBOLA**

EBOLA is also a virus, but is a special one. It is the final boss in the game, and uses ranged weapons to attack Celly.

3. Requirement Specification

3.1. Functional Requirements

3.1.1. Main Menu

The main menu is the first thing that the user will interact when he initializes the game. This menu will give him the access to the functions that the player needs. The following options can be found in main menu:

- Start: Initializes a new game.
- Settings: Player can customize voice level, brightness, and input keys using the settings panel.
- Load: Loads the previously saved game.
- Help: Player can reach a help panel to learn the basics of the game
- Credits: Credits panel has the necessary information to reach the developers.
- Exit: Exits the game.

3.1.2. In-Game Menu

While playing the game there is another menu that the player can use. This menu can be reached with ESC button on the keyboard if the game is running. While this menu is on the screen, the game is paused. The player can use this menu for:

- Pause: Pauses the game without saving. If player terminates the game without saving, current state of the game will be lost.
- Resume: Resumes the paused game.
- Save: Saves the current state of the game
- Load: Loads the previously saved game. This causes the loss of current game.
- Settings: Player can customize voice level, brightness, and input keys using the settings panel.
- Exit: Exits the game.

3.1.3. Map

The game will have a map that the player can open when she/he wants to check the state of the game. The map is a useful adjustment to make the game easier to play for the player. The player can see the position of the character and her surroundings with this functionality. The map shows the enemies as red dots, Celly as a blue dot and other entities as yellow dots. Map can be reached from the “map button”. Initially it is the ‘m’ key in the keyboard. It can be customized from the settings. When the user presses the key binded to the map, it automatically pauses the game. This one of the two possibilities that interrupts the game. The game continues when the player exists the map with the “esc” button or the map button.

3.1.4. Save/Load

The game will support saving and loading current state of the game. The player might want to save the game and play continue from where he left from later on. The player can save the game from the menu while the game is still running. Later on, that save can be reached when player loads the game.

3.1.5. Settings

The game should be customizable via a settings panel. Here, the player can customize the game with the following settings:

- Enable/Disable music
- Enable/Disable game voice
- Adjust voice levels
- Change input keys
- Change brightness

3.1.6. Input Devices

The game should accept inputs from the user. We decided that the easiest input device for playability is keyboard. From the keyboard, the player can use all the functionality in the game. The mouse can also be used throughout the menu.

3.1.7. View Help

The user should be able to get a small amount of help about fame features and controls. This function will help the new players to understand the basics of the game. It will consist of a few screenshots from the game and their explanations by words.

3.1.8. Credits

The credits option can be found in the menu. If clicked, the user can see the list of developers and their contacts. It can be useful for further development of the project and bug reports.

3.2. Non-Functional Requirements

3.2.1. Error Handling

The application will do its best to maintain the load caused by the current state of the game. But sometimes, errors are unavoidable. In such cases, the game should strive to save its current state before the system completely crashes, so that the user will not lose all his/her progress.

3.2.2. Smooth Gameplay

The user should be able to play the game without any problems in terms of performance. We will be doing our best to make the game playable at the same smoothness in every computer device being used nowadays, without lag. Our aim is having good graphics and animations while keeping the graphics running as smooth as possible. It is possible to maintain the smoothness in most computers being used today, since our program will use very little memory.

3.2.3. Extendibility

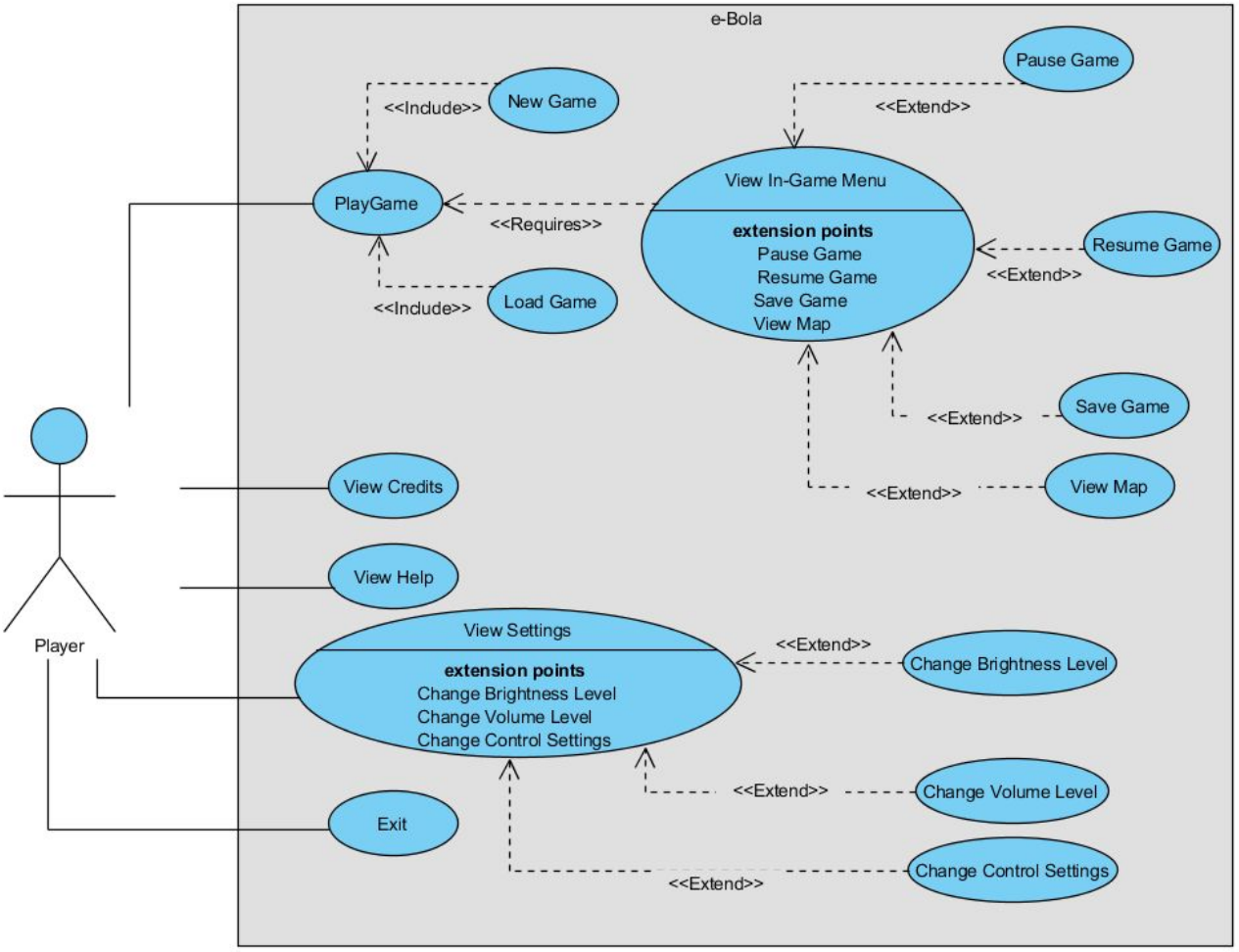
While designing the game, one of our aim was to make game as extendable as possible for future works. New levels, items, enemies, backgrounds and even player characters can be added to the game without effort thanks to our design. We also added the credits button to get user opinion about further updates.

3.2.4. User Friendly Interface

Our graphical interface will be user-friendly, so that the player will find it easy to navigate through the game menu. It is important for us to make the player comfortable while playing the game, and that's why we tried to keep everything as simple as possible. There won't be any unnecessary settings or buttons in the game that would make the game complicated and confusing for the player.

4. System Model

4.1. Use Case Diagram

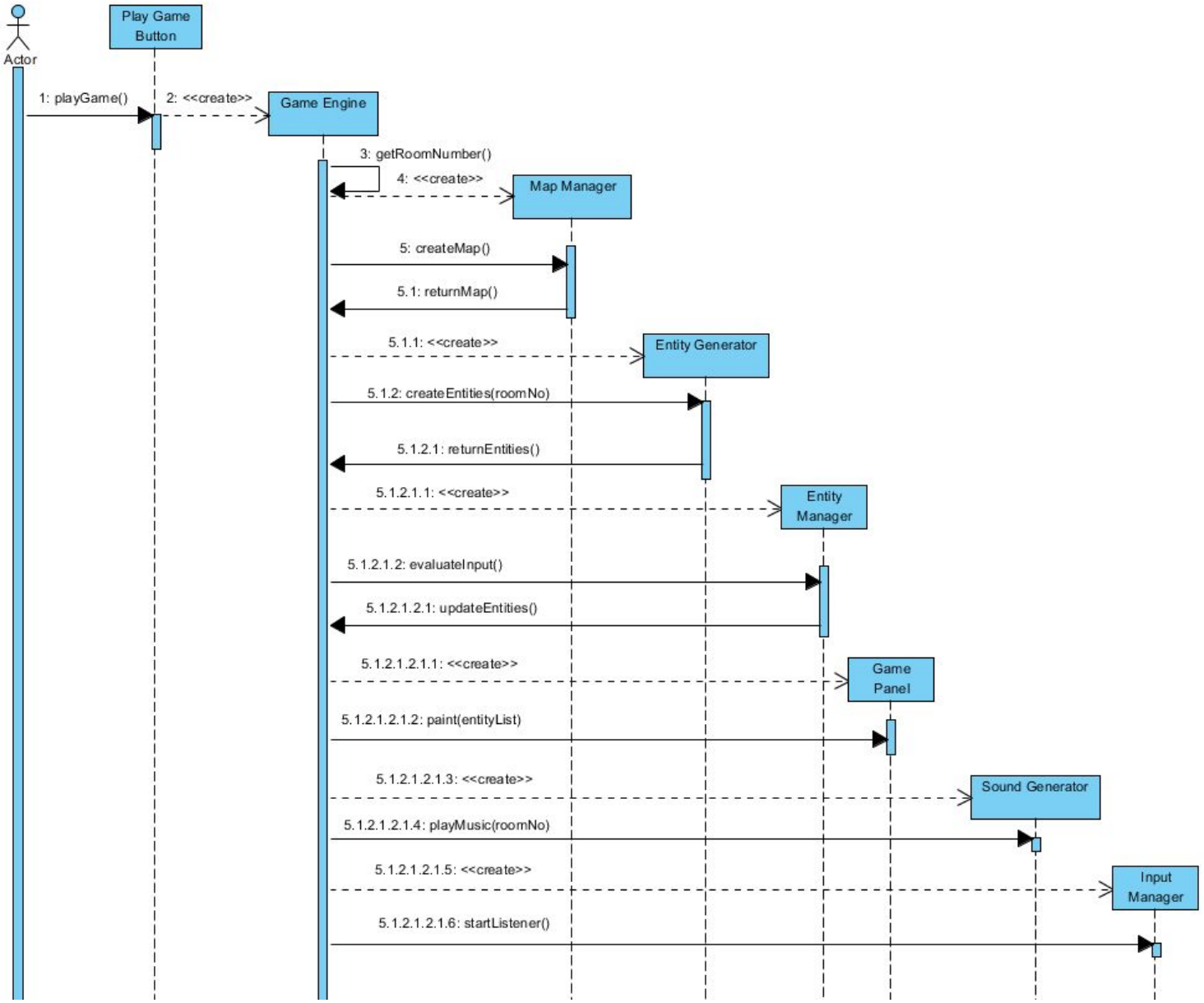


When the player desires to play a game, he can press the play game button which will lead him to 2 choices, new game or load game. Player has to choose one to initialize the game. When new game or load game is initialized, the player can now use the in-game menu. There are few conditions to exit from play game; player should use the in-game menu to go back to the main menu, player should lose the game or player should win the game. While in game, the player is able to use in game menu which gives him the chance to pause the game, resume the game, save the game and view the map. To exit the In-game menu, the player should use the in-game menu button once again and he will continue playing the game. The player can choose the view credits option without any precondition if the user desires to contact the developers. The player can exit this view with the back button which takes him back to the main menu. The player can use the view help option if he desires to learn the basics of the game. It can be exited with a back to main menu button. The View settings button is used when the player wants to make changes in the customizable settings. On the settings view, the

player can use the change brightness level, change volume level and change control settings functionalities options. This view can be closed by a back to main menu button.

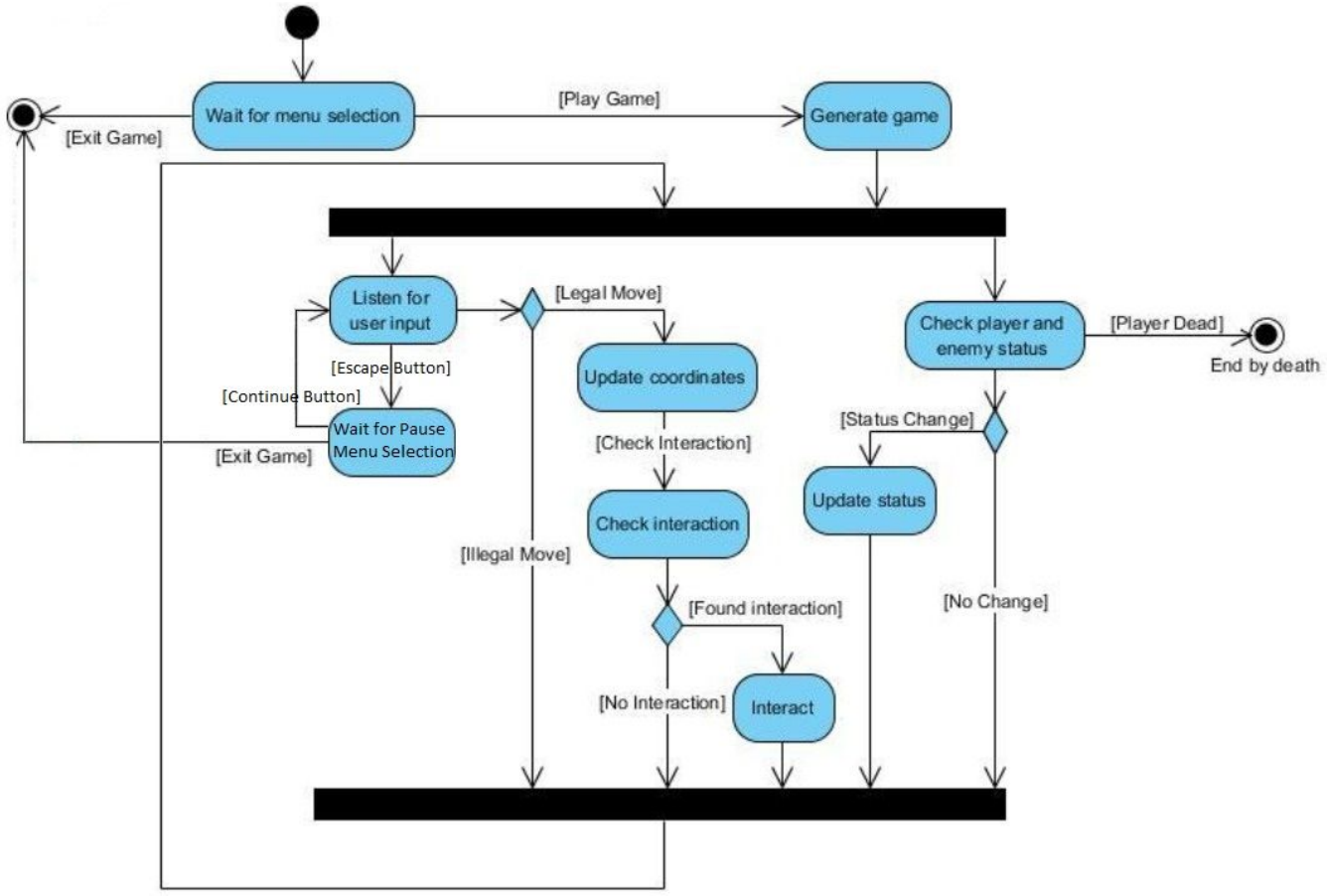
4.2. Dynamic Models

4.2.1. Sequence Diagram



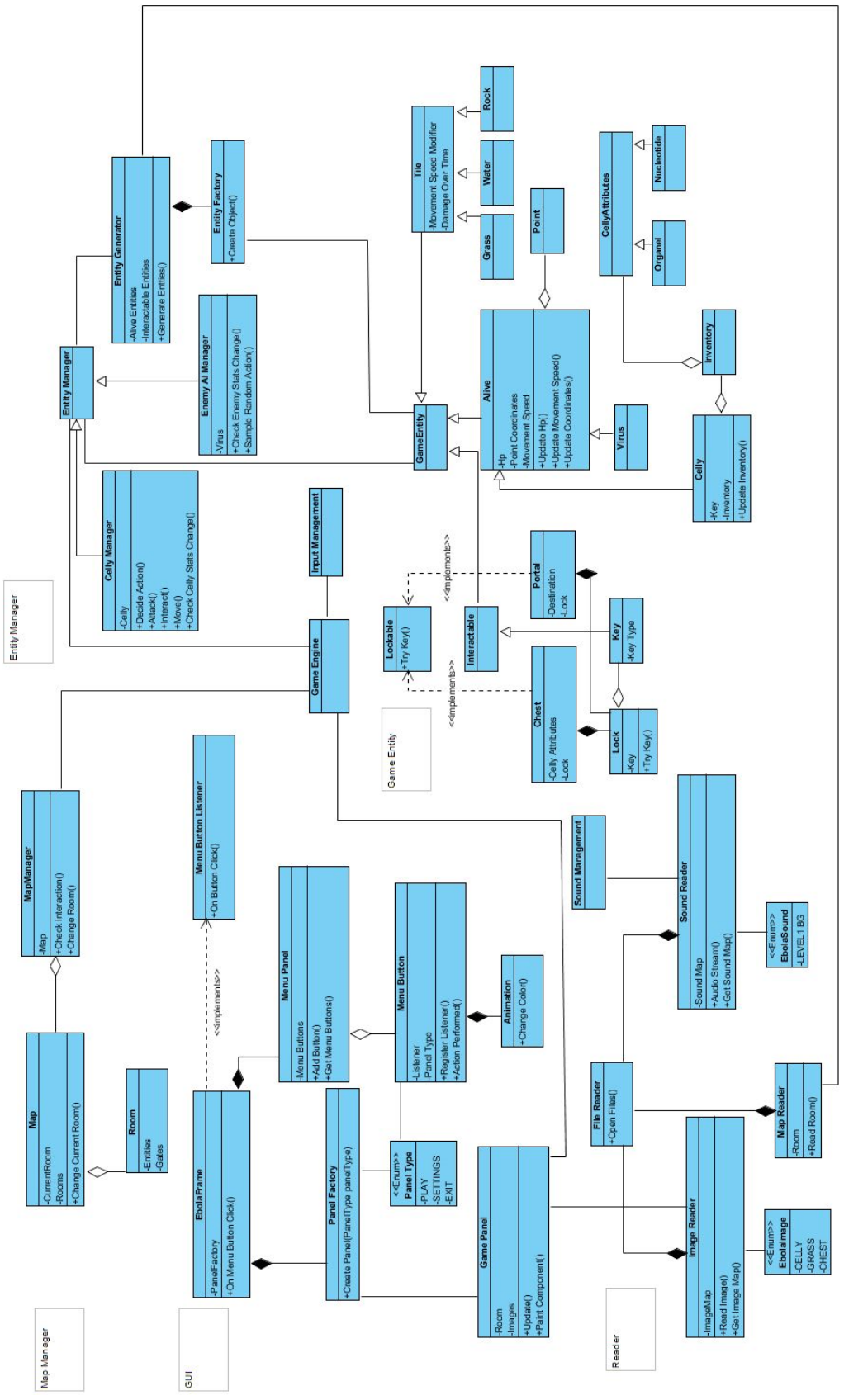
Scenario: The player starts the game by using the Play Game button. Upon the request, the Game Engine is created. The Game Engine creates the Map Manager which creates and returns the current map. Using the Room Number returned by the Map Generator, the Entity Generator is called to create Entities of the current room right after the creation of the Entity Generator. Then, the Entity Manager is created. The Entity Manager controls the movement of the Virus and Ebola and evaluates user input to manipulate Celly object. The Game Engine is responsible of updating the properties of all Entities. Game Engine sends commands to repaint the screen accordingly with the statuses of Entities. Meanwhile, the Sound Generator plays music specific to each room. Input Manager manages the interaction between the player's keystrokes and movement of Celly. The Game Engine constantly checks if the room is changed, and updates the corresponding objects.

4.2.2. Activity Diagram



After the actor (the player) selects the “Play Game” button, the main loop of the game starts. The game will concurrently listen for user inputs and check the status of the player and enemies. Status here refers to attributes that change as the game progresses, such as being dead or alive. When there are changes in statuses, the entity objects will be updated according to those changes. If these updates result in the player dying, the game ends. The status changes can occur even if the player does not give any inputs, so listening to user input and status checking have to be processed at the same time. If the user sends an input to the game, first the move has to be checked to see whether it is a legal move. An illegal move might be trying to go through an obstacle tile. If the move is legal, the coordinates of the player object is updated. At this point, we check if the target location has interactable objects, so the interaction is registered before the next game cycle. Now that all the updates and checks are made, these are shown to the player and another game cycle begins. While listening for user inputs, if the actor presses the escape button, the Pause Menu is shown. The Pauses Menu pauses and the player can go on playing by pressing continue button or exit the game using the exit button.

4.3. Class Model



Game Engine: The main controller of the game, it orders other control classes to do their tasks when they are needed in the game.

AI manager: Controls the actions of non-user controlled entities and requests the game engine to update these entities if needed.

Entity Generator: Creates entity objects of the room the player is currently in.

Map Manager: Reads the text files that contain the rooms of the game and creates a Map object with them, that contain Room objects.

Map: Map of the current level, contains Room objects.

Room: Represents the rooms of the map.

Game Entity: A generalization for all the entities in game they share attributes such as: all of these entities can be drawn to the game screen, have a certain location.

Alive: Objects that can move, take damage and die. Consists of our enemies and main character, Celly.

Interactable: Objects that the player interacts to advance further into the game. Consists of elements such as gates to change rooms, or keys to open doors and chests.

Celly: Entity which represents the cell the user plays as.

Virus: Entity which represents the enemies of the game.

Chest: Entity which represents chests that can be opened by by the user using a fitting Key object.

Key: Entity that the user collects to open chests.

Portal: Entity that is used to switch between rooms of the map.

Tile: Abstraction for objects that represent the floor of the game board.

Point: Represents the current coordinates of Alive entity objects.

Inventory: Holds objects of the CellyAttribute class.

Organelle: Represents objects with certain attributes that can be held in the Inventory of Celly.

Nucleotide: Represents the main currency in game.

Lock: Lock objects for Chests and Portals

Input Manager: Listens to user input and informs the game engine when the player causes an update to the game.

Sound Generator: Listens to events that could trigger any sound effects and plays them when needed.

Game Panel: Draws the entity objects to the screen on every game cycle.

File Reader: Helper for reading files

Image Reader: Reads all image files into the memory.

Ebola Image: Enumeration for images

Map Reader: Reads map data into memory

Sound Reader: Reads sound files into memory

Ebola Sound: Enumeration for sounds

MenuPanel: Panel for man menu. It contains MenuButton objects.

MenuButton: Buttons of the menu.

Panel Type: Enumeration for panels.

Menu Button Listener: Interface to listen for MenuButton clicks.

EbolaFrame: Main application frame of the game.

EntityManager: Creates and controls CellyManager and EnemyAIManager.

Celly Manager: Controls the lifecycle of Celly.

Enemy AI Manager: Controls lifecycle of Virus objects

5. Improvement Summary

Changes made until 18/11/2017

- Introduction was changed to show the purpose and scope of the project, content of the previous introduction has been moved to the storyline section (2.1)
- Section 2.4 was updated to include new features
- Functional and nonfunctional requirements are reworded, a new non functional requirement has been added (3.2.1)
- All diagrams and their explanations are updated to show the new changes
    - “Exit” use case in use case diagram
    - Several class and method names
    - “Wait for Pause Menu Selection” action to quit the game while in game
    - Numerous new classes