AALBORG UNIVERSITY
STUDENT REPORT
First Year of Study
P0 project, 1st semester
A. C. Meyers Vænget 15
2450 København SV.
http://tnb.aau.dk

Title: MortenKombat

Theme: Video Game Application Development
Project period: 02/10/18 – 19/12/18

Project group: ITCOM 5
Members:
Patrick Vibild
Demira Dimitrova
Nourjan Sido
Christian Holfelt
Mads Herlevsen
Máté Bataloni

Supervisor(s):
Morten Falch
Sokol Kosta

Synopsis:

This project was made to create an educational and entertaining game for students with a desire to learn.
Inspired by other popular games, we came up with the idea of a maze/fighting game with popup questions and with the aspect of the RPG genre.
For this project we analyzed the gaming industry and its market, and how popular the educational and RPG games are.
We made a game test and constructed a survey to see if the players of the game would learn from it, and to see if it was entertaining enough.
These data were then used to see how we could improve our game.

Copies:
Pages: 81
Annex number and kind: 11
Finished: 19/12/18

MORTEN KOMBAT

P1 - PROJECT
ITCOM 5
2018

Table of Contents

Introduction

For the P1 Project, the group had a wish to develop both a game as well as a useful application, that could have a greater impact on some of the aspects of life.
By discussing and brainstorming various ideas that would combine these wishes, we chose to develop a Java game that focuses on the educational aspect. Building upon this idea, we agreed to develop a role-playing game that takes place in a maze/labyrinth. The user would have the option to choose and control a character with the objective of going from the start of the maze to its end. As walking through the maze, the player will encounter various enemies that he/she will have to defeat through a turn-based battle.

The educational aspect of the game will be implemented in the form of randomized questions appearing during the battles throughout the maze. We want the game to reach as many people as possible so we want to implement a flexible game play that suits different people with different educational levels.
The players are able to change the questions of the game, increasing the replay-ability and the flexibility of the game in terms of educating. This allows for customizable questions to be targeted to a specific audience if desired.
The player will choose a team of characters at the beginning. For the role-playing aspect of the game, we will imply stats for those characters such as attack points and health points.

We wanted to make an educational game so we looked into how games help, especially kids and students, learn concepts by giving them an opportunity to experience problems and finding solutions in an entertaining way.[1] Games motivate and increase the desire to learn, by combining learning process with rewards, the student will seek out the knowledge by playing and winning the game. This feeling of being rewarded during the learning process can be difficult to achieve with more traditional learning, where the reward is often at the end of the process.

Games can also turn failure into motivation by giving the satisfaction of finally overcoming hard problems. It will also show the students that persistence will lead to success. Games giving an opportunity to keep trying an infinite amount of times to learn, makes students more likely to be more experimental and trying to solve solutions.[2] Some games might not even focus on teaching but just presenting concepts, time periods and other interesting facts. If the player is immersed and interested, the player will remember it and might seek out more information on the topic on his own.[3] By making the game free to play, it should reach out to more people and therefore help more students to learn.

[1] https://educator.mangahigh.com/2017/03/08/gbl-growthmindset-edutainment/
[2] https://www.frontiersin.org/articles/10.3389/fpsyg.2015.01056/full
[3] https://politiken.dk/kultur/art6760839/Falder-du-ogs%C3%A5-i-s%C3%B8vn-til-old%C3%A6vl-Avanceret-spil-g%C3%B8r-oldtidens-Gr%C3%A6kenland-sexet

1. Problem Formulation

Main Problem Formulation
-How can we develop a Java educational quiz game?
Sub - Questions
- How can we make our game flexible in terms of making it a learning tool capable of reaching a large audience with different educational levels?
-How can we make our quiz game more entertaining for the end user?

1.1 Problem Delimitations

We encounter some limitation during the project. First for academic purpose the project requirement was to develop a program in Java, and we had no alternative in choosing another programming language.

Second, we had around 10 weeks to realize the program and the report for the P1. That limited the time that we have to implement the game.

Last, the project should be focused on programming the game and not actually designing or making the artwork for the game. All the artwork was taken from open-source websites.

2. Methodology

A system of methods has been chosen to analyze our problem formulation, “How can we develop a Java educational quiz game?”

The main strategy to flow our work will be the waterfall model. It will allow us to have a stepped method where we can evaluate our project in different stages. The first stage will be to have the requirement for our project. To get the requirement we will analyze existing games that are similar to our ideas. Also, the game needs to be able to educate our players. We will study how educational games are developed and if there is any process to make sure that the game will fulfil our main purpose.

Once we have the requirement for our project, we will design our project and how we will code everything, before any implementation is being done. The first step will be to choose a Java game engine or framework. We will compare what is on the market and choose the one that suits the most. Last, to learn how to use the engine/framework we have chosen, we will perform some reverse engineering on existing codes to gain some knowledge before the implementation.

After having the requirements of our games and the system design, we will begin with the implementation of the project. We will follow the System Design and we will split the jobs into different sections. To be able to work individually we will be working with a version control method. The project might require graphic design and audio input for our game. To be able to spend most of

our time coding we will try to find some public repositories where we can find most of our graphics and audios elements.

Last, when the game has been developed, we will collect some data and analyze if the game fulfils the requirements. We will run some game testing followed with a survey, to get some feedback from the users. This will allow us to verify if our game accomplishes our first requirements. If not, we might consider making some changes to our game.

2.1 Research Onion

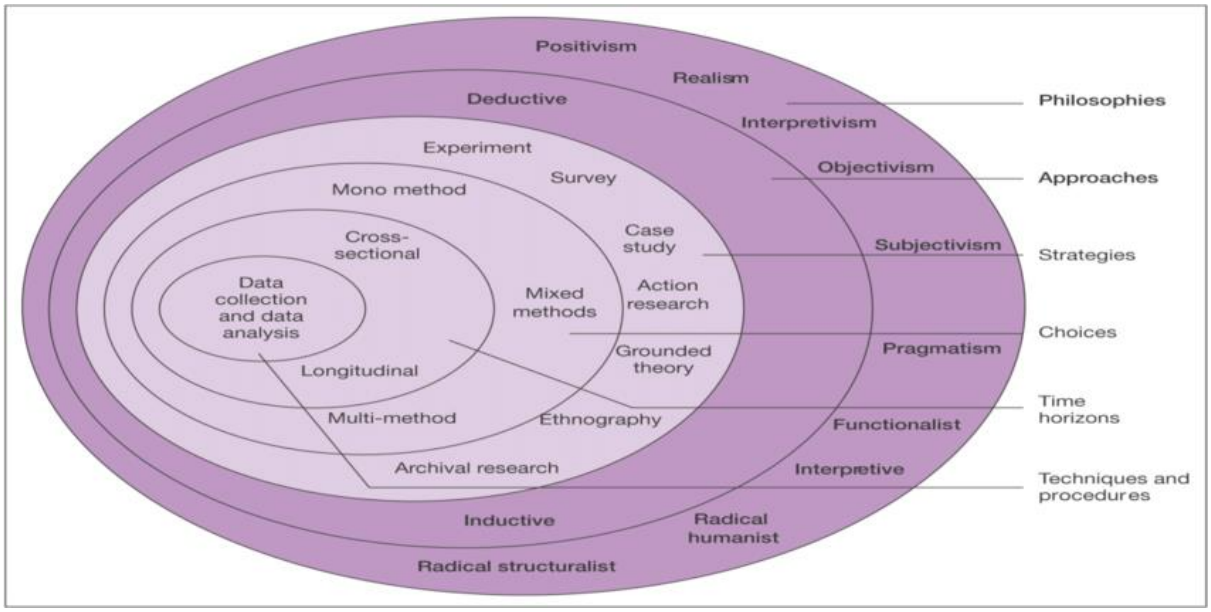


Figure 1: Research onion illustration. https://www.researchgate.net/figure/The-research-onion-Source-Saunders-et-al-2007_fig1_303674706

The research onion was developed by Saunders et al. (2007) in order to describe the stages through which the researcher must pass when formulating an effective methodology.

The research philosophy you adopt contains important assumptions about the way in which you view the world. These assumptions will underpin your research strategy and the methods you choose as part of that strategy.[4]

The outermost layer of the onion is about philosophies. There are three main branches such as ontology, epistemology and axiology. The research philosophy, ontology, used in the project is pragmatism as we focus on answering our problem formulation, while the epistemology that we follow is interpretivism since we have to research different point of view to understand how we should realize a educational game that fulfill our problem formulation. Axiology followed is realism, after all the research that we are carrying out is biased by cultural environment.

The first layer of the onion research model is about the philosophical stances adopted in the research. There we had the Pragmatism philosophy as we were researching for actual statistical data but at the

[4] https://eclass.teicrete.gr/modules/document/file.php/DLH105/Research%20Methods%20for%20Business%20Students%2C%205th%20Edition.pdf.

same time, we conducted a survey to better know the influence on the people, to see whether we had achieved our goal or not.

The second layer is about the approaches when making the research. The two words in that level are 'deductive' and 'inductive'. In our case, we adopted the inductive approach. We took an example or fact and then we generalized it.

The third layer is about the strategies used to achieve the research. Here we chose the survey strategy because we needed rich statistical data especially focused on our project. Also, we use archival research, we are using documentation for existing programming languages and libraries.

The fourth layer is about making the choice to use qualitative or quantitative data. For our project, we collected both quantitative and qualitative data but the outlook is rooted in only one of them. So, both types of data are analyzed from only one point-of-view.[5] This choice is named Multi-method.

The fifth layer is about Time horizons. Here we adopted the cross-sectional study, since we are researching different genres in the game design. Likewise, we use a short time horizon in our study case. The study is focused during a certain span of time, and the project does not research any data in a longer period of time.

The sixth layer is about Data collection and Data analysis.
The methods used to collect data in the report fall mainly under 2 categories, data we collect ourselves (primary data), and data found through research and manuals (already published secondary data).
The data presented in the report fall under both qualitative and quantitative data, as the data we collect will include conducting a survey and few interviews.

The purpose of the survey is to have feedback from actual players to improve the players satisfaction and find missing game instructions.

2.2 Waterfall model

Waterfall model: The waterfall model was first formally introduced in 1970 by Dr Winston W. Royce[6]. It's a software development process model, in which the development method is linear and sequential. The waterfall model consists of various phases and each phase has its unique objectives. Each phase must be completed in order to start the next one, there is no overlapping or going back in the phases of the waterfall model. Generally, the output of the previous phase can serve as input for the next phase in line.

The waterfall model is divided into several separate phases, where it progresses downwards in one direction- hence the name waterfall model. The progression of the different phases is shown in the diagram below.

[5] https://onion.derby.ac.uk/
[6] https://en.wikipedia.org/wiki/Waterfall_model

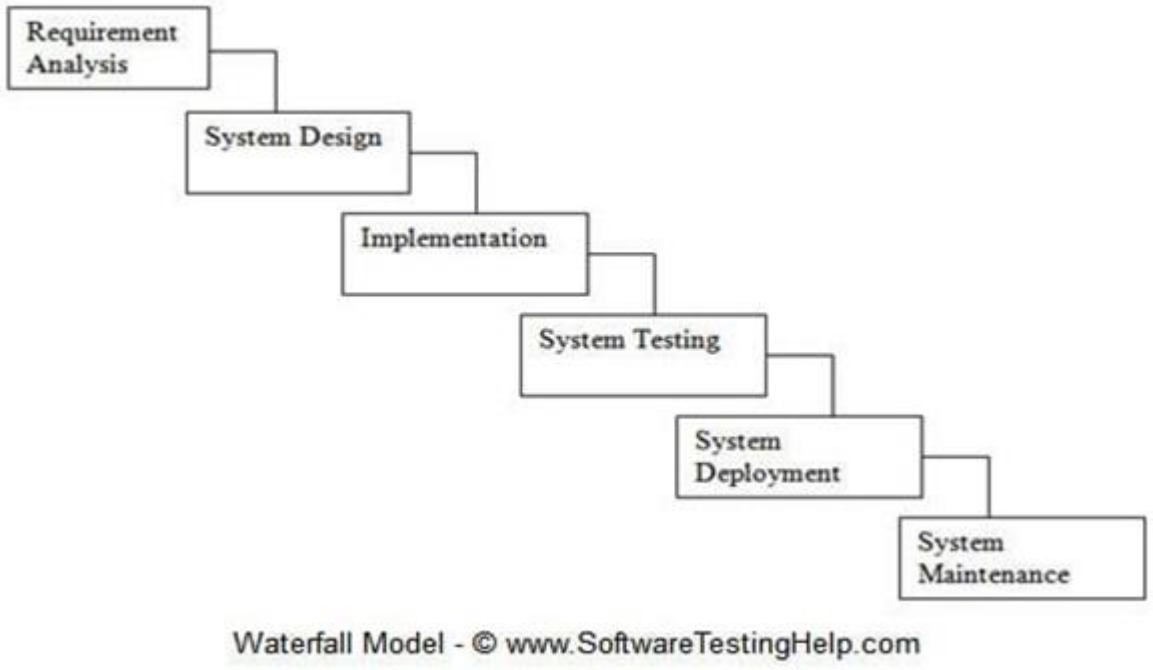


Figure 2: Waterfall model phases taken from: https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/

Requirement analysis: During the initial phase, the system requirements need to be documented. The requirements should specify what the application should do.

System design: In this phase, the design of the system should begin according to the specification made in the previous stage.

Implementation: Having created the design, the code will be the next thing to develop.

System testing: The testing phase begins; the code needs to be functional and meet the requirements.

System deployment: Here the application should be running and be ready to be released in its respective environment.

System maintenance: In the last phase, any user encountered defects need to be addressed and fixed.

2.3 Market Segmentation

In accordance with the Waterfall model, we start with the Requirement Analysis. We will make a market segmentation and a Stakeholder Analysis to target our customers and before continuing to the System Design.

A market segmentation is a process of dividing a market of potential customers into groups/segments, which is based on different characteristics. It can be used to prioritize the target audience and personalize the market strategies to ensure higher success rate. Since customer preferences are becoming wider along with the competitive options becoming more available, the market segmentation is an important part of any marketing plan.
The segmentation divides a population into variables.

There are four main types of Market Segmentations. These are based on the following:[7]

- **Geographic Segmentation**
  The geographic segmentation includes targeting people who lives in certain areas or visits the area. Customers have different needs based on their location.

- **Demographic Segmentation**
  The segmentation by demographics is more targeted to certain ages, genders, educations etc.

- **Behavioral Segmentation**
  The behavioral segmentation divides the population by their decision-making pattern, behavior and usage.

- **Psychographic Segmentation**
  This segmentation is focused on the lifestyle of people and their interests/activities to define a market segment. [8]

2.4 Stakeholder

A Stakeholder Analysis is important for identifying and analyzing the stakeholders needs. The stakeholder analysis aims to develop a strategic overview of the relationships between the stakeholders and the issues they care about the most.

The analysis is also a systematic way to analyze them by their power and interest. Here, the key players are the high power, high interest stakeholders, whereas the low power and low interest stakeholders are less important.

There are 3 important steps to create a Stakeholder Analysis. First, the stakeholders must be identified. Second, when the stakeholders have been identified, they should be mapped in terms of their power in the project. Finally, the strategies for communication with the stakeholder group will need to be linked to their interests.

2.5 Java Game Development with LibGDX

The book Java Game Development with LibGDX[9] written by Lee Stemkoski introduces both LibGDX basic to advance and Java advance documentation.

The book contains 14 chapter where each of them creates a small game in LibGDX, introducing new concept on each of them. The book has been helpful to understand the functionalities that LibGDX can

[7] https://www.disruptiveadvertising.com/marketing/market-segmentation/
[8] https://www.marketing91.com/4-types-market-segmentation-segment/
[9] https://www.apress.com/us/book/9781484215012

bring on Java, giving samples of each of those functionalities with a source code that could be downloaded at the owner's GitHub[10].

The book also contains an introduction on how you can write the requirement of a game following the appendix from the book called “Game Design Documentation”.

This method is part of the Requirement Analysis, so it should be done before continuing to the System Design.

To start the development of our game, following the waterfall model, we need to discuss and describe the game design beforehand. Moreover, even if not following the waterfall model, a game design documentation is needed in order to serve as a guide for the development process. This section describes step by step on how to make the design of a game, by answering a list of questions.

2.6 Framework- LibGDX

The second milestone in the development of our game will be the System Design. The waterfall model requires that we are not overlapping phases, but the outcome of the previous phase can be used in the following one.

In this project we decided to use LibGDX to develop our gaming application. In the analysis section, we will explain our choice and different engines and framework will be compared, to see which one provides the best solution for our game development. Complexity, utility and documentation will be compared among the Java frameworks and engines to choose the solution that best suits our game.

LibGDX is a java based cross platform and open source game development framework. It enables the development of desktop and mobile games with the same base code[11].

2.7 Tiled map editor

Tiled map editor is a 2D level editor that helps with editing tile maps of various forms and projections (i.e. isometric, hexagonal and straight rectangular tile layers). We used Tiled Editor to design the different maps of the game as it is a free, easy and flexible to use program[12].

The Tiled map editor has object layers, so it is easy to implement spawn points, solid objects, animations etc.

In addition to this, it is also possible to use image layers and just regular tile layers to create an in-depth view of a map. It is easy to import tile assets and scale it, so it fits the size of the preferred map.

[10] https://github.com/Apress/java-game-dev-LibGDX
[11] https://en.wikipedia.org/wiki/LibGDX
[12] https://www.mapeditor.org/

2.8 IntelliJ class diagrams

Following the Waterfall Model, after the System Design, the Implementation part is coming next. In this part it should be considered the actual code development after the design is already created.

In our project, we are going to use mainly the class diagram in order to depict attributes, methods and disparate relationships between classes. Practically, it is the blueprint of a software and used to get a better overview of the software`s system.

Usually, the diagrams are used in the planning phase of the software design to get a better understanding of the software`s structure in advance. By applying this tool in our work, we would also like to get an overall picture of the program, which can help us to organize more efficiently the actual software writing part of our project.

We can use IntelliJ to generate diagrams of our project after it is completed, these diagrams are shown in the report. Most of the diagrams show a small part of the project because the full picture won't be readable.

2.9 Reverse Engineering

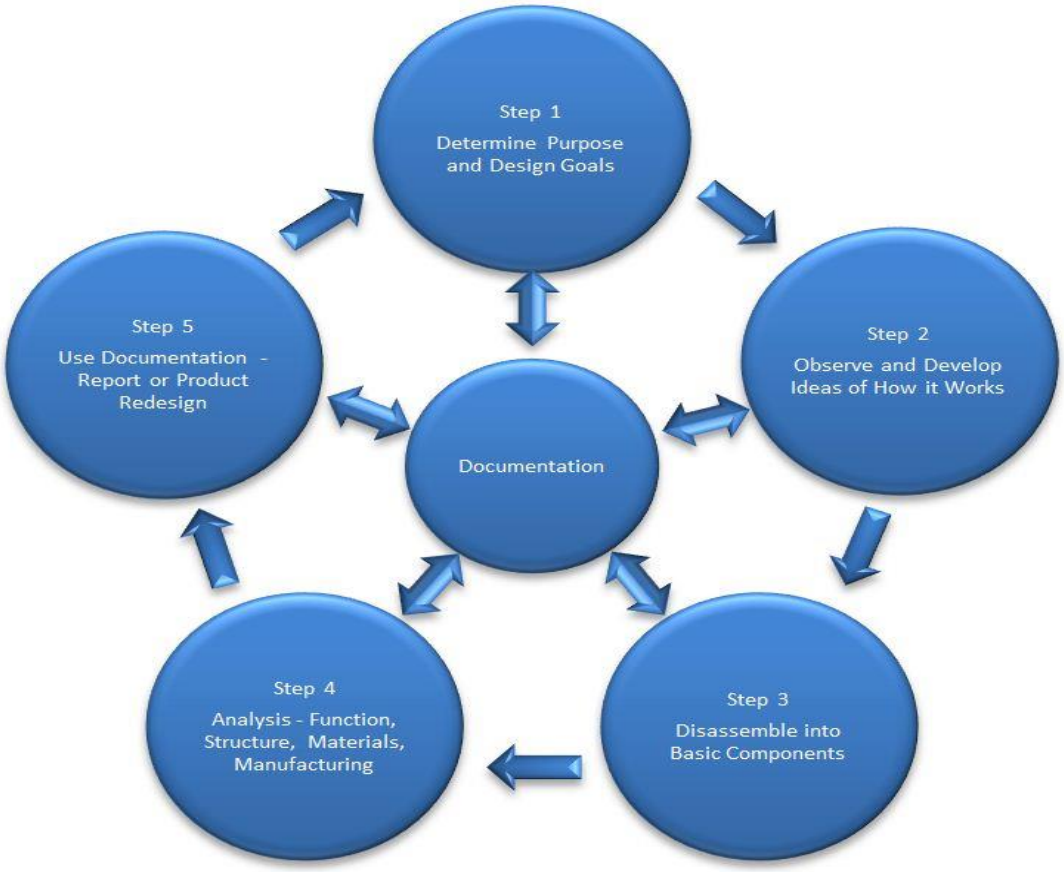


Figure 3: The five steps of Reverse Engineering:

http://legacy.lincolninteractive.org/html/CES%20Introduction%20to%20Engineering/Unit%203/u3l7.html

"Reverse engineering is the process by which a man-made object is deconstructed to reveal its designs, architecture, or to extract knowledge from the object. It is similar to scientific research, the only difference being that scientific research is about a natural phenomenon".

In our case, we talk about the reverse engineering of software. This means the analysis of the software system by identifying its components and their interrelationships in order to recreate the software in another form. The inversion of the waterfall model is another interpretation of reverse engineering.
An actual example for reverse engineering is class diagram. Class diagram tools analyze the source code, understands the interrelationships between the parts and finally generates the diagram itself.[13]

In our project, we apply this method because we would like to reduce the learning period due to lack of time. We choose some already existing Java game demos which use Libgdx and try to use reverse engineering on it. It is found a better approach by us than starting from scratch for the above-mentioned reason.

2.10 Version Control

For our game implementation, we will use version control with the help of Git-Hub. This will give the project the opportunity to have different members working in the same code on different computers.

Each of the members will have the freedom to code and decide when to share their changes in the master branch of the project. It will also allow having the same project shared with different computers. Using version control integrates work done simultaneously by different team members.

By using version control it will grant the advantage of having a historical record of each of the version of our project. This will give an insurance to the project if an accident occurs during the development of the project. It will provide also an opportunity to roll back to a previous version on the project and undo some changes in the code if needed.

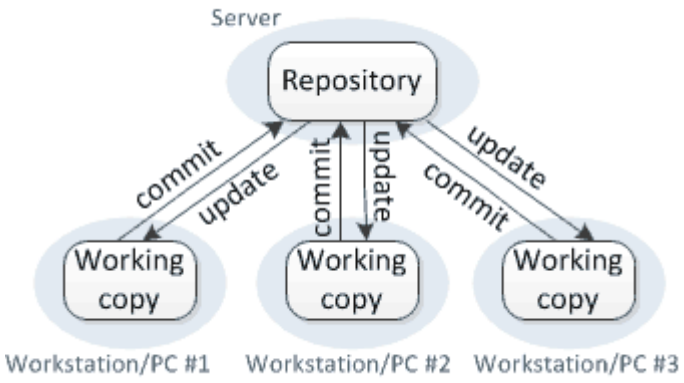


Figure 4: https://homes.cs.washington.edu/~mernst/advice/version-control.html

[13] https://en.wikipedia.org/wiki/Reverse_engineering

2.11 Graphic design and audio

To realize the idea we came up with, we would need to implement specific graphics and audio. In order to do this, we decided to use open libraries, because it wouldn't be possible to make all the graphics (background, characters, etc.) and audio files (sounds for action, fight, background, etc.) by ourselves. Moreover, using public libraries will benefit the user experience. Our team does not have the necessary abilities to make professional graphics or audios. Those things are part of the front-end programming, which is straight connected to the user experience.

2.12 Game Testing - Survey

Referring to the Waterfall model, the next step is System Testing. We will make a game testing, to test if the code actually fulfils the requirements.

After finalizing the code and having a running application, we plan on conducting a survey in which we gather quantitative data in order to asses and possibly improve our game. We will try to get as many people as possible from the university to play the game and then answer some questions regarding their experience during play time.

The purpose of the survey is mainly to test the players satisfactions, in terms of entertainment and the educational aspect of our game. The results will be collected and carefully analyzed, they will help us make any changes to the game and conclude on our problem formulation.

2.13 SWOT analysis

After making the game testing we plan to do a SWOT analysis, because we want to get a better understanding of the application and its potential advantages and disadvantages. SWOT stands for strength, weakness, opportunity and threats. We believe that a SWOT analysis is necessary in order to evaluate and get a general overview of the application from different viewpoints.

Figure 5: SWOT Analysis: http://guides.library.cornell.edu/HADM2720/SWOT

3. State of the Art

When speaking of educational games, there exist a broad variety of different games on the market. Some are very similar to each other and others are very innovative. We got some inspiration from different games both educational and entertaining.

The games that we studied were mostly RPG (role-playing game) and trivia games. We wanted to mix both genres trying to make something not many people have tried before, having the educational part of trivia game while having the player been involved and entertained in an RPG game.

The gaming industry has a significant market, and more people than ever before, now play video games[14] and some even have gaming as a lifestyle. "The ability to take gaming on the road and integrate it into our daily routines via powerful mobile devices has been significantly influential in bringing gaming into the mainstream over the past few years."[15] Today, more than half of the world's population in the industrialized countries, owns some form of a gaming device.

Educational games have a huge potential in immersive learning experiences. It can also increase the experience and the motivation to learn. Most educational games are based on simulation or adventure games, which is often about micromanaging, strategy or to take the player to a historical event.

Since failure is an option in games, people can learn from their mistakes and there is a risk for trying again to achieving success. "They learn to work collaboratively and to solve problems through highly complex mental challenges that games provide."[16]
Educational video games can provide some possibilities, which are similar to active learning. "Active learning is student participation in the learning and teaching process, where students themselves engage with and, to an extent, create their own learning experience."[17]

3.1 Darkest Dungeon

Darkest Dungeon is a gothic RPG developed by Red Hook studio released in 2016, it's the main source of inspiration for our game. The game is well received with good ratings on most major game publications[18] and it has sold over 2 million copies worldwide[19].

[14] https://www.statista.com/statistics/748044/number-video-gamers-world/
[15] https://www.nielsen.com/us/en/insights/news/2016/gaming-gone-global-keeping-tabs-on-worldwide-trends.html
[16] https://www.sagu.edu/thoughthub/the-hidden-value-of-gaming-in-education
[17] http://summit.sfu.ca/item/281
[18] https://www.metacritic.com/game/switch/darkest-dungeon
[19] https://en.wikipedia.org/wiki/Darkest_Dungeon

Figure 6: In-game screenshot of combat: https://store.steampowered.com/app/262060/Darkest_Dungeon/

In Darkest Dungeon you manage multiple heroes to explore dungeons under a mansion you inherited at the start of the game. The combat is a turn-based combat, and all heroes have a stress level that increases with further exploration and combat. When a high stress is reached for our heroes it might create a negative trait to our heroes or even kill them.

The hero navigates underground mazes-dungeons, walking in hallways between large rooms looking for treasure. The treasure is guarded by monsters such as skeletons and zombies which you come in combat with. During advanced dungeons there might be as well a boss at the last room that you have to defeat to reclaims bigger rewards.

During the game at the end of each dungeon the player get rewarded with items and experience point that increases the combat ability of his heroes, allowing the player to successfully pass content with higher difficulty.

Figure 7: In-game screenshot of boss: https://store.steampowered.com/app/262060/Darkest_Dungeon/

Features we include:
While our game is nowhere near as complex and eye pleasing as Darkest Dungeons, the concept of our game is similar, as we will also have the main characters navigating mazes, looking for new areas and coming in combat with multiple enemies. In our game, we also want to include the turn-based combat as we will have multiple heroes fighting multiple enemies. The heroes will also have different abilities to attack with and options to heal.

Just like Darkest Dungeon has boss fights, we also have different bosses the hero has to fight. Bosses are special character that are stronger and more complex than usual encountered enemies in the mazes. In our game the player instead of being rewarded with unique items when defeating bosses, he will get keys so he can open the treasure room and win the game.

3.2 Pokémon

Pokémon is an extremely popular game that also has given us some inspiration for the design of our game. The player controls fantasy creatures, that he has previously captured and trained. These are called Pokémon. During the game the player will be challenged to fight different Pokémon, increasing the difficulty the more the player advance in the maps.

Our main interest in this game is how the player explores the map, and moves within the world. In Pokémon game you control your player seen from birds-eye view, giving you a good overview of the map around you. By designing MortenKombat in a similar way we can implement a maze system making the game more complex, compared to for example a platformer game.

Like in Pokémon, in MortenKombat we collide with enemies in the maze map screen and start a fight with them, in Pokémon these fights are hidden and random in special tall grass. The fights are also turn based, but only 1 versus 1 in Pokémon, compared to our 3 versus 3.

Figure 8: Screenshot from the game: 'Pokemon: Emerald'

3.3 Final Fantasy IV

In Final Fantasy IV, the player chooses between a large variety of characters and complete quest to advance in the gameplay. There is a main character travelling within the world map trying to complete the quest to advance in the story game. The main characters will trigger in random intervals a fighting screen where all the crew has to defeat a certain amount of monsters.

Every member of the crew has different rolls, some of them are more focused on supporting their crew-mates, some other are dealing damage to the enemies, and last some characters will have the roll to receive and mitigate damage (this roll setup has been called as a holy-trinity between the gaming community). The status like mana points and health points will be saved between fights. Forcing the player to be conscious of the life of their characters in a long term, and having the option to return to the city to heal all their characters.

Figure 9: Source: https://boards.na.leagueoflegends.com/en/c/gameplay-balance/xtEnN5sA-abandonment-of-the-holy-trinity

Our game will have the same exploring mechanics where there will be one character moving through the exploring map. Before the game even start the player will be asked to choose between different characters. Each of those characters will represent a roll in the holy-trinity and is those characters that will be used to fight the enemies.

Figure 10: Source: https://store.steampowered.com/agecheck/app/346830/

Also, we want to give a long term thinking to our player, trying to minimize the damage taken for each fight so the decision made in the fight matters in the long term. That means the health point of our characters will be saved between screen, and we will create some healing spots during the exploring map that will allow the player heal their characters.

3.4 Quest of Dungeons

Quest of Dungeons is another dungeon crawl game. It's a 2d game featuring 16-bit graphics. You choose from 4 characters which are warriors, wizards, shamans or assassins, then you have to roam the dungeons, combating enemies and looting everything to stay alive[20].

This game has a top down design, the dungeons have limited vision as most of the screen is dark; similarly, in our game one of the dungeons has also a dark theme where there is very limited visibility. We took inspiration in the game on a design and artistic way. The game uses pixelated illustrations, and the maps are built in a tiled way, creating square shapes within the dungeon.

Figure 11: In-game screenshot of a dungeon taken from: http://www.playstationcountry.com/quest-dungeons-ps4-review/

3.5 Trivial pursuit

Since we wanted our game to be educational and serve a purpose that is more than entertainment. We looked around for some educational games and found Trivial pursuit, a board question game with six categories of questions, the categories are geography, history, science and nature, arts and literature, entertainment and finally sports and leisure.

[20] https://store.steampowered.com/app/270050/Quest_of_Dungeons/

3.6 Triviador

Triviador is an online multiplayer game, where strategy and knowledge are key to leveling up. You conquer territories and attack castles while answering trivia questions.[21] Answering correctly levels you up and you conquer new areas.

Figure 12: In-game screenshot of Triviador where you play against 2 other players: https://world.triviador.net/index.php

This is a multiplayer game where you are paired against two other people when answering the questions. The questions vary in theme and you are given a time limit to answer. Answering correctly lets you move/conquer a new territory while by giving a wrong answer, you lose your territory.

Our game will implement the same concept of been rewarded or punished when answering to questions. In MortenKombat during the fights with the enemies, each action has a chance to pop-up a question in the screen with four multiple-choice questions. If the player knows the answer will increase the damage their characters does to the enemy, giving a wrong answer will make the character deal less or no damage to the enemies. The same idea will be implemented when the enemy attack our hero, giving us a beneficial outcome for each turn in the fighting screen when we know the answer for the game.

3.7 Summary of the State of Art

After having looked at all these games, we now have some good ideas on what we want to include in our game and how we should implement them. We wanted to have multiple characters to choose between to increase replay-ability and complexity. We also wanted to have each character to play a specific role, so we made three roles equal to the holy trinity of tank, damage dealer and healer.
Most of the user interface and the idea behind the game comes from Darkest Dungeon, as the game we wanted to create was Darkest Dungeon with trivia questions.

[21] https://itunes.apple.com/dk/app/triviador-world/id1001043187?mt=8

We wanted the player to navigate a maze between the fights to make the game more interactive, instead of just stringing all the fights together in a long line. So, we took the navigation system from Pokémon, which remain almost unchanged and still popular after 18 years[22]. This allows good players to avoid fights in the maze if they want to, and gives the player the feeling of exploration.
By requiring to collide with an enemy to start a fight in another screen and not have any timings on buttons or time pressure on the player, we can make our game focus more on the educational part compared to having a good reaction time and good tactics. This should make it easier for non-gamers to play our game.

4. Analysis

In the analysis we will take a look on the game design and education through games. Furthermore, we make a game market analysis, do a market segmentation, and we will define our stakeholders. Lastly, we will analyze our game testing results and do a SWOT analysis.

4.1 Game Design

Morten Kombat is a maze based educational quiz game. The player controls three characters, who move as one when exploring the maze. In the game world there are four bosses, one of which is the final boss. The other three hold a key, which opens the door of the treasure room. The objective is to get the three keys after killing the bosses to open the treasure room, where the final boss is and kill the final boss to win the game. When searching for the bosses, the characters will come into combat with monsters. We have made seven maps and we made it very easy to include more maps, and make them connect with the rest of the maps.

During combat, questions will sometime be shown to the player who will have to answer the them. All the questions are general knowledge, multiple-choice. Each character will have a chance to get a question every turn, both the heroes and the enemies. If the question is answered correctly the effect is beneficial to the player, if the answer is wrong the effect will be a detriment to the player.
We wanted to have the questions where they mattered and could not be ignored or brute forced. Our default questions are taken from trivia games within a target of 16 years and above, but we have made it easy to update and change the questions to every player or a specific topic.

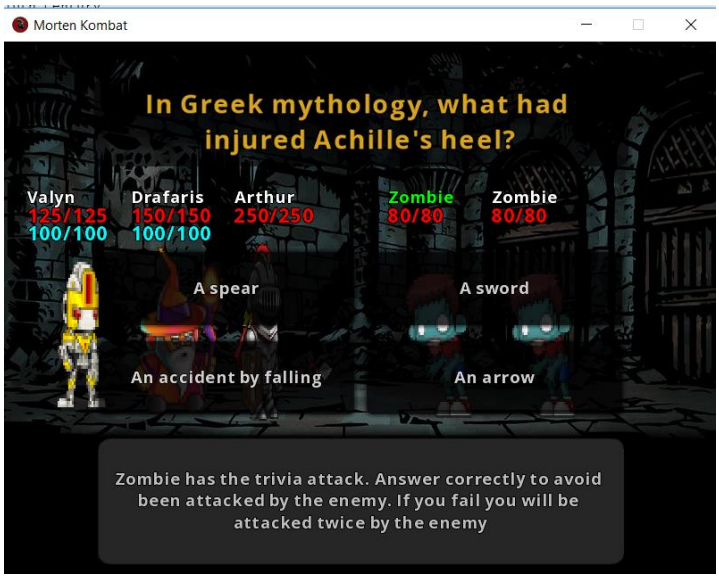


Figure 13: Screenshot of a trivia attack in our game.

[22] https://en.wikipedia.org/wiki/Pok%C3%A9mon_(video_game_series)

The questions can easily be changed in a text file, so it matches the players desires depending on what they want to learn. Simply write the question, followed by one right answer and 3 wrong answers. From here, the questions will be randomized during the fights.

This game is playable for most people who can navigate the maze, the game is easy to get into and simple to play. Because we use almost a copy of the Pokémon games movement system, and our combat is hardly more complex than Pokémon's. Kids should easily be able to play and learn with our game. The game does not use any complex graphic and should be able to run on almost any desktop or laptop. The only requirement is to have java installed in your computer. This makes the game have a very wide audience.

The presence of searching, finding, thinking and answering activities in the game makes it intriguing. However, a big gaming experience is not needed. The targeted game platform is desktop and the game are controlled with the keyboard and a mouse/touchpad.

The game is attractive because it keeps the player thinking outside of the box and anticipating following combats and questions. While at the same time searching for the exit of the current maze room and enemies to fight with. The player will be able to see only a part of the current maze, which is making the whole process of searching for the exit a lot harder. Although the game is educational, the player is losing the notion of that because of all the activities included and the combat. Different teams of characters would be formatted in order to increase the players involvement, replay-ability and interest in the game.

In the first room, there will be a treasure room, with the final boss inside. To access the room the player needs to navigate and fight three different bosses to be able to access the treasure room. Once the player goes back to the treasure room the player must fight the final boss that was not available before. The user will win the game when killing this last boss.

The short-term goals of the character will be to successfully fight the enemies, find the door to the next maze room, find health packs to get health refilled. The medium-term will be to get the less possible damage from combats and do as much as possible to the enemy, find the bosses in the maze. The long-term goals will be to collect all of the keys from the bosses, to open the final door and kill the final boss.

The enemies are moving back and forth in the maze, and when the character comes into contact with one of the enemies the combat starts. The big boss is staying in the treasure room, locked until the other three bosses are killed and the three keys are collected.

Figure 14: Picture of a health bar: https://boards.fireden.net/v/thread/409294289/

During combat, the characters can attack the enemy and the enemies attack the characters seemingly at random, but they will attack the front character the most. The player will have control over three different characters, each with its own health points and some will have mana points, which will be visible above each character. Each of our characters will be able to use one of three different abilities per turn. The name of the active character will be green to show it is able to act. Abilities will target and attack the enemies, sometimes costing mana points to use and our support character will be more focused on healing abilities to use on the player-controlled characters.

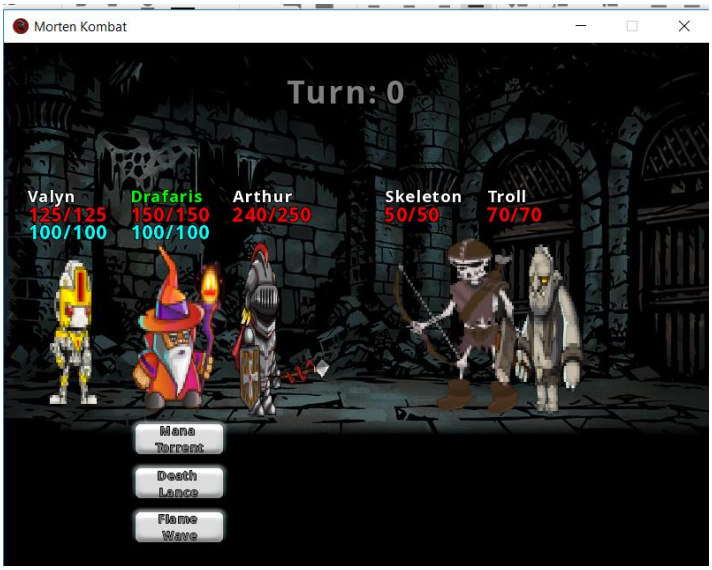


Figure 15: Screenshot of the FightingScreen

The combat will be over when the Health Points of the enemies or the player characters are set to zero. The player should have a tactic to defeat the enemies, but the main goal is to answer correctly and get rewarded by answering the questions correctly. Answering the questions should be better rewarded than having a good strategy, and not answering any questions should make you unable to complete the game.

After each fight the health will be saved for the next fight, but while exploring the player can find a medic pack that will restore the health of the player characters, but there is only one per maze so it is a limited resource.

Furthermore, we strive to make upgrades of the character and a loot system, as a major part of the game. The player will also be able to go up in levels by getting experience points during the fights. This will give it the feel of a real RPG. By doing so we combine an educational trivia game with an entertaining and vivid game-world.

The player will play with an 800 x 600-pixel window, and the maps are 2048 x 2048 pixel, having a ratio of 8.7 windows/map to explore on each of the seven maps. It is important that the player can't see the whole map at the beginning, and will have to move around the maze to find the exit.

All the maps are intended to have a maze layout that the player have to explore to find the entrance of the next map. The player should be challenged on finding the exit of the rooms. Exploration should be the main topic when moving between maps.

The maps are connected with doors that transport between them, and each map has the possibility to have two exits. This will create a non-linear maze that the player has to explore to finish the game. The structure of the maps should form a binary tree data structure[23]. During the dungeon exploration there won't be any interface that shows any information to the player nor any map of the maze or where the play has already been.

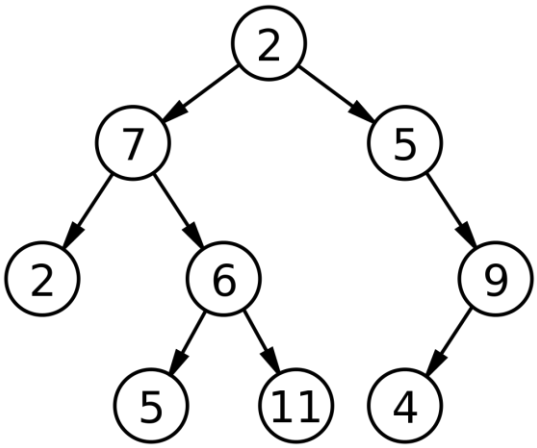


Figure 16: https://en.wikipedia.org/wiki/Binary_tree#/media/File:Binary_tree.svg

To be proficient at this game, the player will need to be able to answer correctly to the questions during the combat. Moreover, the player will need to be able to control the movement of the character and remember where they have been in the maze, to minimize walking in circles or the same path twice and find the fastest way out of the maze.

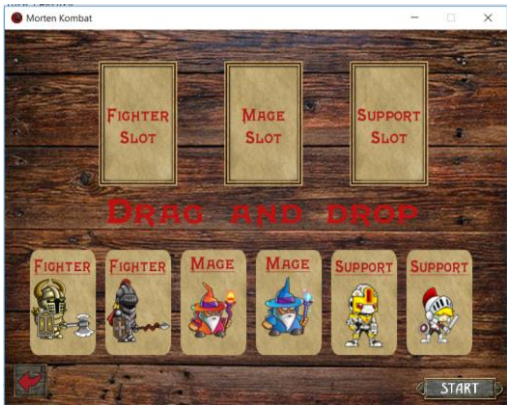


Figure 17: Screenshot of the Drag & Drop Screen

The player will be introduced to a menu screen where the player is able to choose their team. Once the player has decided his characters, he will be introduced to the game with the tutorial screen, before the game starts.

The game uses pixel art with varied themes. Some of the maps have a nature theme and will use brighter colors while the dungeon maps will use darker themes for the layout. The maps will have some animations to give the player a experience of being dynamic world where object have animations and enemies having also some animations and being move randomly.

The fight screen will have some basic animation to describe what character is getting affected by abilities.

There will be background music on for the exploring maps and another background for fighting maps. The exploring music will be chill giving the mood to move between the created maps. The fighting music will be more active trying to persuade the player to be more focused on the actions.

If the player loses all three characters in one of the battles, he will lose the game and will return to the main menu after a Game Over screen. If the player is able to obtain all 3 keys and defeat the last boss, the screen will be changed to ‘You Win’.

[23] https://en.wikipedia.org/wiki/Binary_tree

4.1.1 Story of the game

We created a small storyline to introduce the player to the gameplay. Creating a storyline helps the player to connect to the game and this is a strong aspect of Role-Playing Games.

"A long time ago in a far-away land, a small village was located in a valley between the mountains. Although the citizens were poor and didn't have much, they were happy and lived in harmony. Living off the land, this year had been good to them. They had a very great harvest and earned a lot of gold from it. They stored the gold in the local bank until they decided what they should spend it on to make life better in the village.
But then, on a dark, stormy night, a wizard came to town. He abused his powers and stole all the gold from the villagers and disappeared to his castle. The wizard was known as Morten and has been harassing the neighboring villages together with his evil companions lately and have been stealing their treasures too.
The villagers had now lost what little they had left and didn't know what to do until one local boy spoke up. He promised he would go to the wizard's castle and get all of their possessions back and make an end on the tyranny the wizard and his companions has been causing for so long. He would need companions to tag along on this dangerous task, but only a few dared to volunteer. The boy chose three companions to join him.
And so, they left the village with thunderous applause to begin their adventure…"

4.2 Education through games

With the fast development of new technologies infinite number of game types appeared. The main purpose of games is entertainment. However, the best job a person can do is the one he likes. So, when one enjoys the activity, he does not feel how fast the time is passing by. Taking the momentum of entertainment and targeting a specific goal, could be crucial to transforming the world we live in a better place. Therefore, games could be focused on educating broad masses of people with diverse background.

Educational games are games explicitly designed with educational purposes, or which have incidental or secondary educational value.[24] Those are more and more needed nowadays with the fast-growing technological world. The question to be asked before begin working on the project is if Game-Based Learning (GBL) has any major benefits compared to regular classroom learning.

Good education is one of the main factors for the transformation of a child to a responsible and acknowledged adult. Unfortunately, there is not one method of educating as there are different learning styles.[25]

[24] https://en.wikipedia.org/wiki/Educational_game
[25] https://teach.com/what/teachers-know/learning-styles/

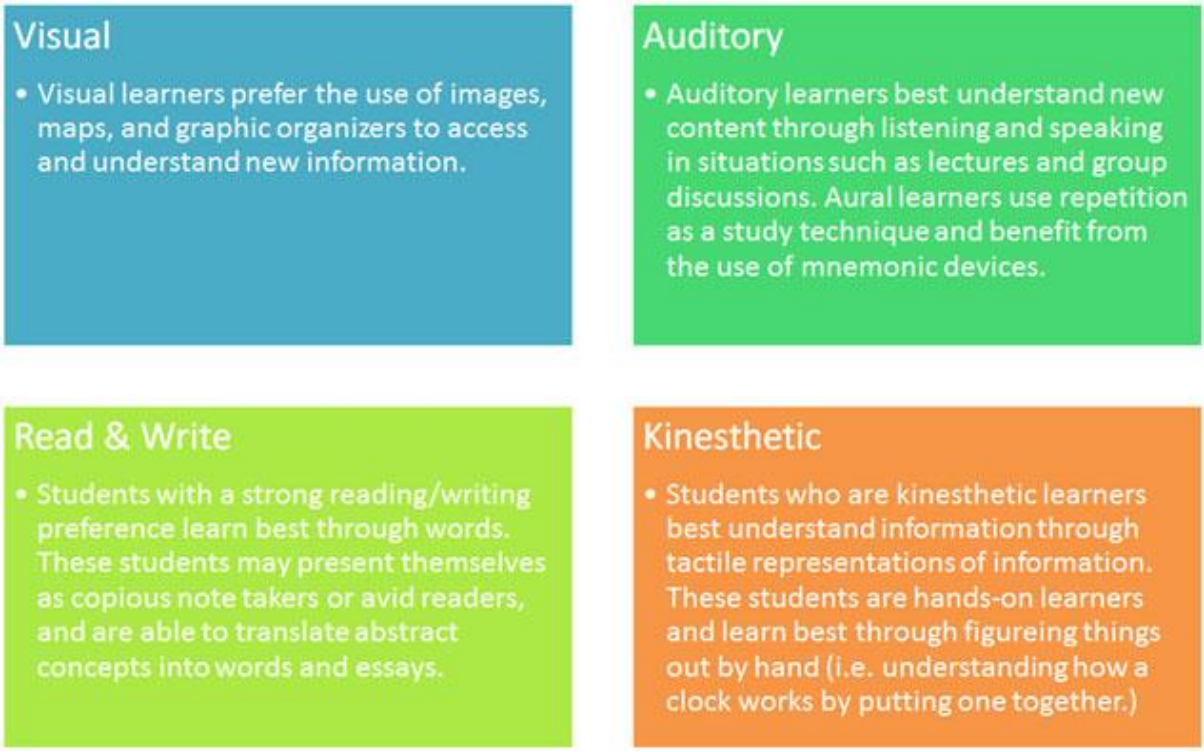


Figure 18:VARK model on learning styles: https://teach.com/what/teachers-know/learning-styles/.

This leads to the use of different teaching methods to suit the needs of the broad audience. In the typical classroom with 30 students in, the normal class is composed of lecture and maybe the assistance of slides. "This focuses mainly on only one aspect of learning, audio with some additional support from visual. According to Foreman, this is contrary to the very nature of a productive learning environment and is used throughout academia because it is the most cost-effective teaching method".[26]

On the other side, implementation of games in the classes brings many benefits compared to the standard class routine. One of them is the instructiveness and the combination of different learning styles at once. Another one is the accessibility via the World Wide Web, which is allowing the player to have access to the game from any location. Moreover, it is provoking and promoting the problem-based learning by active involvement in the process rather than just listening passively and taking notes.

4.2.1 Gamifying the education

Games make sense in a number of arenas of learning: military, finance, even urban planning. It is much easier to run a few games or simulations to see which strategies work than to spend time, money, and lives implementing bad ideas.[27]

The much-quoted work by Avedon and Sutton-Smith (1971) made the following observations about the benefits of using games as educational tools:

1. Games with simulated environments engender more student interest than the more conventional classroom activities.

[26] "Education through Video Games - West Point." https://www.usma.edu/cfe/Literature/Pennola_09.pdf.
[27] "Education through Video Games - West Point." https://www.usma.edu/cfe/Literature/Pennola_09.pdf.

2. By participating in games, students will learn more facts and principles than by studying in the conventional manner.
3. Students will retain information learned in games longer than information presented through conventional methods.
4. Students will acquire more critical thinking and decision-making skills by participating in games with simulated environments.
5. Student's attitudes will be significantly altered by taking part in games.[28]

Moreover, there is a survey made by MIT, examining 650 freshmen showed that 88 percent of them had played games before the age of 10, and more than 75 percent of them were still playing games at least once a month. Sixty percent of MIT students spend an hour or more a week playing computer games. By comparison, only 33 percent spend an hour or more a week watching television, and only 43 percent spend an hour or more per week reading anything other than assigned textbooks. On the one hand, one would expect these technologically advanced students to be early adopters and enthusiastic users of new media. On the other hand, given the bad reputation that gaming has in some circles, it may be news that so many students can play games and keep up the GPA needed to get into a place like MIT.[29]

As another proof of the stated above, we had a look at a Bulgarian educational website Ucha.se[30]. It aims students from 1st grade in elementary school up until 12th grade in high school. It covers a wide range of disciplines - everything included in the curriculum of the Bulgarian educational system plus crucial courses missing from the schools like finances, health and sport, entrepreneurship etc. They present the lessons with videos and have games and quests after each lesson. So, we questioned, why this website was so popular and how they were continuing growing? As an answer, we can deduct that students from all ages do have interest in learning and are not getting dumper, so is a matter of the teaching method used to keep the interest of the student and to nourish their curiosity.

We were inspired by this approach and we wanted to implement it in our game too.

We, therefore, see a potential in making an educational trivia-game, that strives to create a believable and entertaining universe for the user, while still keeping the aspect of teaching and educating.

4.3 Game market analysis

Video games have come a long way since the first games emerged in the 1970s. Today's video games offer photorealistic graphics and simulations of reality, to a degree which is astonishing in many cases.[31]

Nowadays market for video games is bigger than ever and its growth does not seem to be slowing down.[32] Its place on the market indicates a popularity that cannot be matched with even the movie

[28] "Learning through Games: Essential Features of an ... - SURFACE." https://surface.syr.edu/cgi/viewcontent.cgi?article=1055&context=idde_etd.
[29] http://plato.acadiau.ca/courses/engl/saklofske/download/digital%20gaming%20education.pdf
[30] https://ucha.se/
[31] https://www.statista.com/topics/868/video-games/
[32] https://www.cnbc.com/2018/07/18/video-game-industry-is-booming-with-continued-revenue.html

business.[33] This means the market for these games has many consumers making it a strong platform to be developing a product on. Anything ranging from a First-Person Shooter to a racing game, you can be sure that there is a group of people willing to buy and play it. With the insurance of a big market, creators are given the chance to experiment and 'twist' the genre by incorporating different aspects into the game. A such aspect could be the educational take on an entertainment product as a video game is. Which is what we have chosen to focus on for the P1 project.

There are several genres of video games, such as RPG (Role Playing Games), shooter games, sports games among many others. The chart below shows the most popular game genres sold in 2017 in the US. RPG games are fourth place with 11.3% share of units sold.

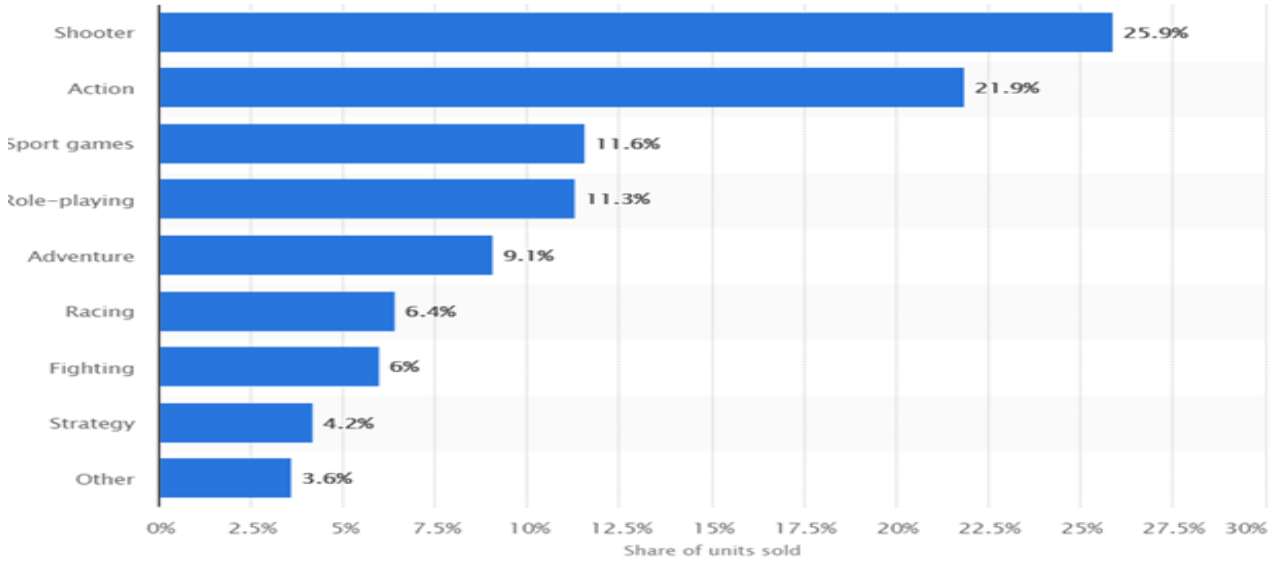


Figure 19: Sales of different game genres: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/

Our game genre choice is RPG games as our game is primarily based on Darkest Dungeons, which is also an RPG. Additionally, as mentioned before, consumers have huge interest in RPG games and it is gaining more and more popularity. 2018 has been a big year for the RPG genre. Titles as Red Dead Redemption 2, God of War, Monster Hunter World along with many other titles, has dominated the game market this year. By taking a look on this year's game awards, (Which is like the Oscar nominations for games) most of the nominated games belong in the RPG genre where God of War took the title as Game of the Year[34].

Role playing games involve the players in the story and allow them to exert control over the narrative of the game. Players are not usually focused on only one thing but are rather prompted to take part in various activities such as discovering the environment, conversing with other characters, solving problems etc. These components make for a great learning environment, making role playing games ideal for incorporating educational aspects into the game play[35].

[33] https://gamingbolt.com/

[34] -https://www.forbes.com/sites/insertcoin/2018/12/07/heres-the-full-list-of-winners-from-the-game-awards-2018-including-goty/#6fc4c8661cc4

[35] https://dspace.cuni.cz/bitstream/handle/20.500.11956/61652/BPTX_2013_1_11410_0_352950_0_145703.pdf?sequence=1

The market for educational games is on the rise.[36] Over the last decade, educational games has become more and more popular. Many games are specifically designed as educational, as well as a number of other entertainment games, have been successfully used for an educational purpose.[37]

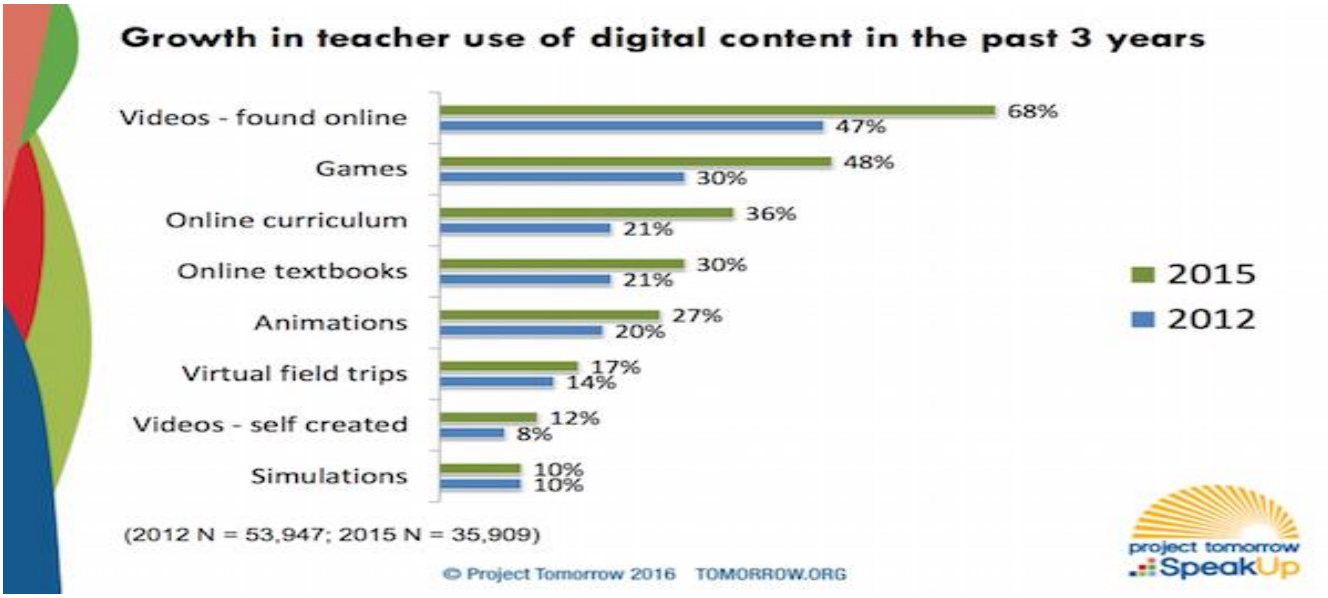


Figure 20: https://blog.kinems.com/educational-games-are-growing-in-popularity-with-us-teachers-and-students/

As seen in figure 20, almost half of the teachers in the U.S (48 percent) uses game-based learning in the classrooms. The figure shows the growth of the learning tools used in schools in the U.S from 2012 to 2015. It shows there is an increasing market for the educational game and it is gaining more popularity.

4.4 Market Segmentation

As seen in the game market analysis, there is an increasing interest of game-based learning. And with the already huge popularity of RPG games, the next step is to define our target audience.

Therefore, we want to focus on gamers who have interest in educational games, RPG games along with retro gaming. The market for the gaming industry is very broad and there is a lot of competition out there. We already covered the RPG genre and the game-based learning, but the popularity of retro gaming is also increasing.[38] Old games are being remade to the newer systems and PlayStation have launched a small, new version of the old PlayStation 1 (Called Sony PlayStation Classic) so you now can play the old nostalgic games. New modern games with a retro look are also becoming more and more popular for casual gaming.

Our game aims towards a target audience, suitable for the Demographic - and Psychographic segmentation.

---

[36] https://www.gamesindustry.biz/articles/2018-08-08-metaari-game-based-learning-market-will-reach-usd17-billion-by-2023

[37] https://www.researchgate.net/publication/261076302_Educational_Games_-_Are_They_Worth_the_Efforta_Literature_Survey_of_the_Effectiveness_of_Serious_Games (Download Pdf file)

[38] https://www.eurogamer.net/articles/2018-02-09-the-retro-gaming-industry-could-be-killing-video-game-preservation

It can be hard to make a Demographic segmentation since gaming is liked by all ages. But our game mostly focuses on students and in general, for the people who wants to learn in a different way. But it does require the player to be able to understand and read basic English.

For the Psychographic segmentation, we aim for the player base that are into retro gaming, since the game has a classic pixel style artwork. It is simple to play and at the same time, entertaining and helpful for learning new things.
If the game gets published, it will be free to download so everyone owning a computer, is able to play the game.

4.5 Stakeholders Analysis

To better understand and identify all interested parties and the influence they have over the project, we decided to do stakeholders analysis. The main stakeholder for us right now is the end user, as the game is educational. But it is not really targeting a specific topic, it is more general knowledge questions covering several themes. Anyone who enjoys trivia kind questions and/or looking to gain some new information will be interested in the game. While the type of game users can hugely vary, an important group is students, playing games while learning might interest students more than reading straight out of a textbook.
It is possible to customize the questions and choose any questions one wishes to include in the game. This makes the game desirable for many stakeholders, as it allows for great flexibility with the questions asked during gameplay. We will identify some of the more important stakeholders for our project, stakeholders will be listed below accompanied with clarifications for the choice. Stakeholder include:

Schools
Broadly speaking, schools of various educational stages because the questions can be easily adjusted to the pupils' level of knowledge. Schools want to better the performance and experience of their students and therefore schools are one of the main stakeholders for us who would be interested in introducing such a game in their system. Schools hold high authority regarding our project.

Teachers
Teachers are responsible for teaching their students, teachers can write questions appropriate to the topics they teach and students play the game and solve the questions.
Depending on how the teachers like the game and the concept could affect the use of the game. While schools might allow the use of the game in teaching process, if teachers do not like it and do not encourage the students to use it then the game will not reach its potential. Teachers are one of the main promoters of the game and they hold high authority.

Students
Students are the ones the game is designed for, they are the ones that are going to spend the most time playing the game as they need to constantly learn new things, thus this would be of interest to them. They, of course, need to enjoy playing the game while also getting something out of it, which is learning. Otherwise the purpose of the game is defied. Students hold low authority, but have high interest over the project.

The government
The government holds decisions regarding funding of educations, our game could be of interest as a way of introducing a new learning concept in school systems. Governments holds low authority because they have low to no influence on the deployment of our game.

Parents
Parents care about the games their children are spending their time on and thus will have an interest in our game. Parents need to trust and believe in what the game offers, they are concerned about the validity of the game and whether it is an appropriate tool to teach their children. The game needs to provide a safe learning environment and meet the parent's expectations. Parents hold low authority.

The group
The group members are the ones doing all the work regarding the project, they decide upon all details that should be included in the game. They are the creators and the main promoters. the group members hold high authority over the project.

The supervisors
Our supervisors are in charge of overseeing the project, the project has to be approved by them and we are often getting help from them as well. Supervisors are the ones that decide on the direction we are heading with the project as they are constantly advising us regarding our work. Supervisor hold high authority over the project.

In the graph below, all stakeholders are shown relative to their position over the project. they are placed according to their level of interest and power/authority regarding the game.

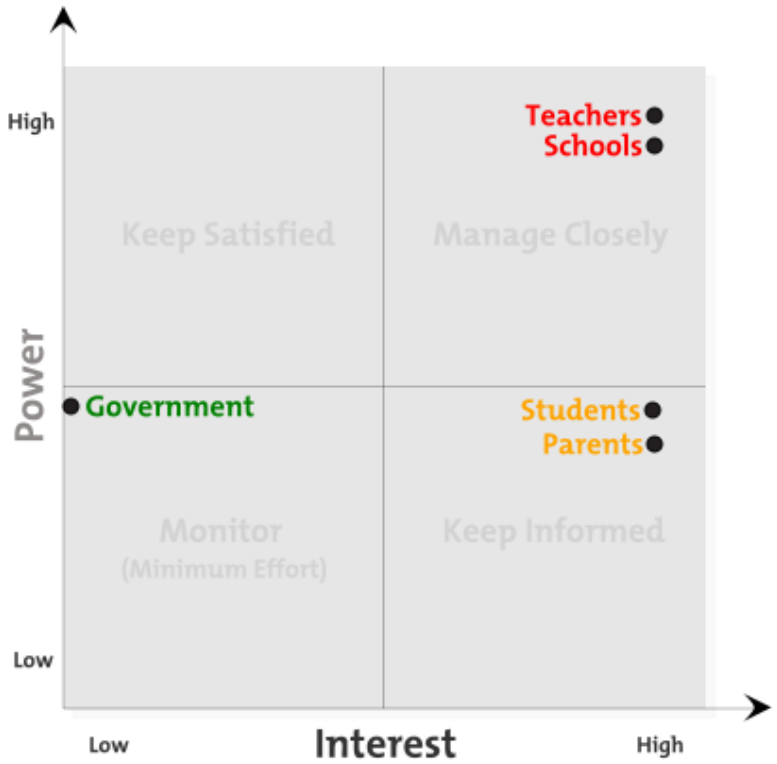


Figure 21: Stakeholders Graph: https://www.mindtools.com/pages/article/newPPM_07.htm

As for strategies for communicating with the stakeholders with high influence (schools and teachers), we would like to establish an agreement with a school to give us permission to conduct a study in a classroom environment, where we would like to introduce the teacher to the game and have them design his/her own questions. We monitor the process and carefully analyze the findings to support future advancement.

4.6 Game testing and survey

We wanted to have feedback on the game and see how people like it, in order to do so, we decided to conduct a survey. We have 2 modes in the game, quick and normal. One mode of the game focuses on map navigation (quick mode) and has very few enemies, the participants were told to complete the game. The second mode (normal mode) was the complete game which means it had enemies everywhere in the map. This version required the participants to play for a certain time rather than complete the game, the focus was the fights. we made two surveys to test both modes the game testing went smoothly and in we got 19 responses for both versions of game and survey. The surveys focused on both the game play and the players experience and the possibility to add comments at the end of the survey. We tried to make the surveys short and straight forward, we used the Likert scale to indicate the answers. The surveys were created using Google Surveys and can be seen in appendix 6 and 7. The respondents were handed an introduction to the game testing (see appendix 5) stating the purpose of the survey and that no personal information will be collected. Our sample group was students from Aalborg University as that is where we carried out the game testing. Most game testers did the survey, however, we still managed to talk with a couple of the respondents afterwards for more detailed feedback.
We got feedback and discovered some bugs, the results helped us make the following changes to the game. We got many suggestions, and all of them were considered, and thus most of them were fixed after the game testing.

Before the testing, we let two of our classmates play the game, they found it difficult to know what they had to do and got lost in the maps. Therefore, we implemented signs in the different maps to guide the player and make it easier. Most survey respondents found it easy to navigate the map with 69.3% finding it easy or very easy. They also found the tutorial extremely helpful as the majority of the respondents 84.3% strongly agreed or agreed on it being helpful (both results can be seen in appendix 8).

Furthermore, on the hero selection screen, people found it confusing as they were not sure how they should select the champions. We added instructions on the screen that they need to drag and drop the hero in order to select.

Another change we made to the game was to decelerate the hero's movement in the ice map, as some people found it too difficult to control his movement.

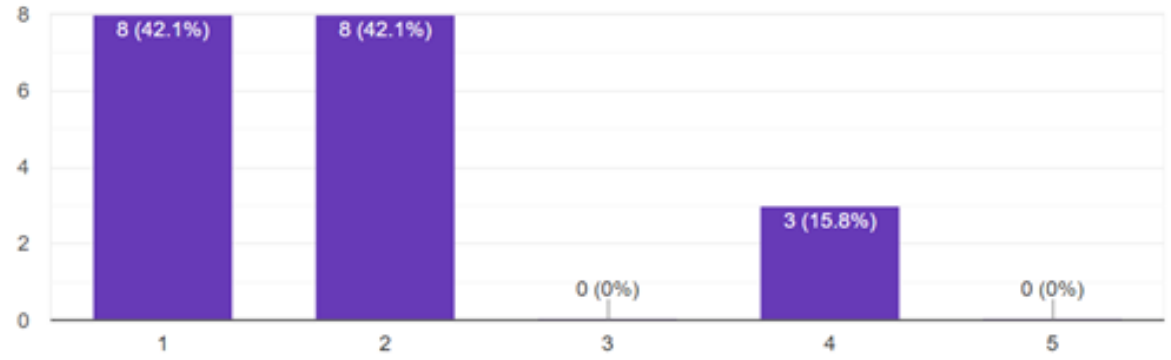
One last change we made, was to hide the inactive buttons of the other champions during the fighting screen, although we have a turn indicator some people still found it unclear, so we changed such as that the only active buttons are the champions who has the attack turn.

We found two bugs during the game testing, firstly, two of the game testers encountered a bug where they got stuck in the fighting screen with no possibility for further attack. This was fixed by using a thread safe java data structure. The second bug was that one of the attacks belonging to the one of the warriors, could heal him for more than his max health allowed, we fixed this after the game testing. As for the overall experience and enjoyment, we wanted to know if the game fulfilled the purpose of incorporating education in the game while still being entertaining.

Below we present the results, the charts are automatically generated by Google Survey, the y-axis show the number of people and x-axis shows the number of choices.

We asked the participants to rate their entertainment while playing, entertainment is crucial aspect of playing video games.

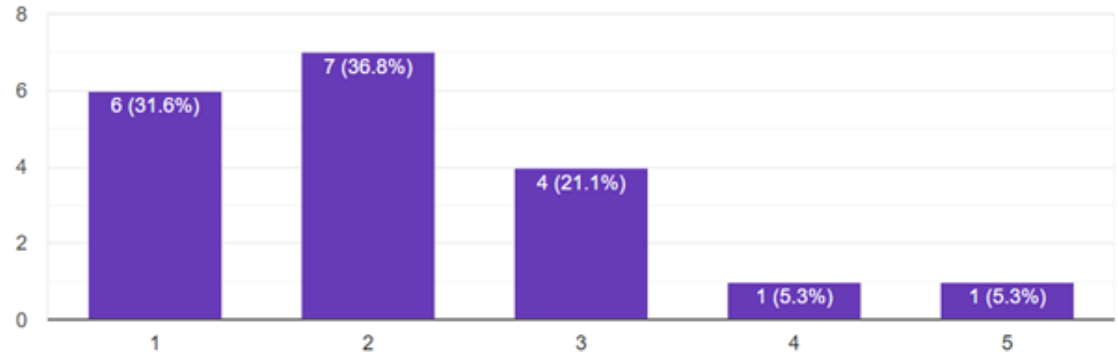
19 responses



The chart shows the result with 1 being very entertaining and 5 being very boring, most respondents (84.2%) found the game very entertaining or entertaining.

We also wanted to find out about the balance presented in the game with both the education and entertainment aspects.

19 responses

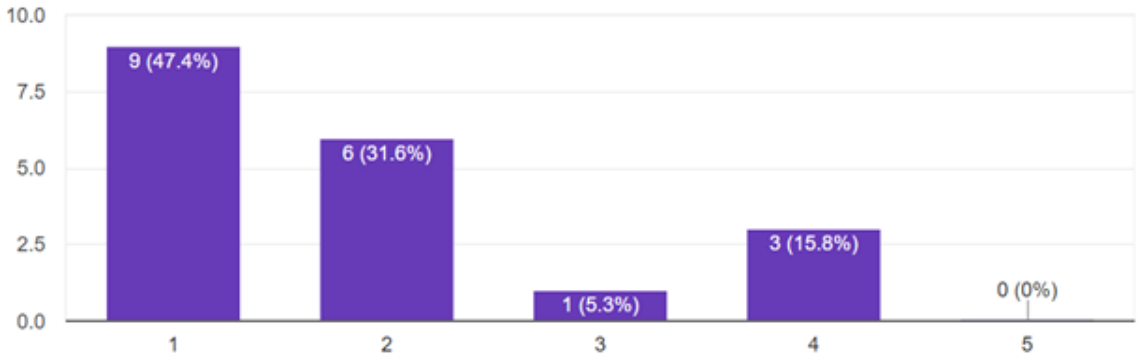


The chart shows the results with 1 being ‘strongly agree’ and 5 being ‘strongly disagree’. The answers

were more spread out with 31.6% strongly agreeing, 36.8% agreeing with a good balance presented in the game.

We also wanted to know about the player satisfaction level as the players need to feel immersed in the game and be satisfied with the experience to fully enjoy the game play and want to play the game again.

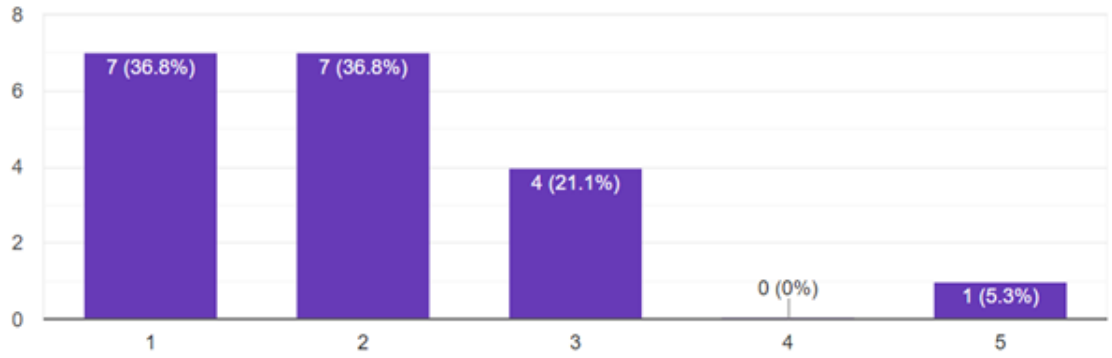
19 responses



The chart shows the result with 1 being very satisfied and 5 being very dissatisfied. 79% of the participants were satisfied with their experience.

Finally, we wanted to know about the likelihood of wanting to play the game again. It is crucial that the players want to play MortenKombat again, in order to increase the knowledge of the students.

19 responses



The chart shows the results with 1 being very likely and 5 being never, 73,6% of the respondents were very likely or likely to play the game again.

To sum it up, we think the game testing was very helpful for us as we got feedback that helped us with improving and making some changes to the game, it also gives a better understanding of our project and it will help us with adding input to the SWOT analysis. Additionally, the survey results will also help us on concluding the problem formulation.

4.7 SWOT Analysis

We did a SWOT analysis because we wanted to assess our projects internal strength and weaknesses as well as external opportunities and threats.

Strengths	**Weaknesses**
1. Game based learning 2. Customizable questions 3. Cross platform	1. The game play can get rather repetitive. 2. Weak animation/design 3. Violence in game (age restriction)
Opportunities	**Threats**
1. Free and open source 2. Big market for games	1. Competition 2. Requires java installation to run

Table on SWOT – Analysis

Identifying the factors:

Strengths:
Game based learning.
The fact that our game is an educational game is a strength, the game concept is to keep the player entertained while also teaching the player about certain topics. Studies have shown that game-based learning enhances learning effectiveness as well as enhancing learning motivation[39].

Customizable questions.
The educational content in the game can easily be modified by anybody. This makes it attractive to many educational institutes as well as anybody interested in playing the game.

Cross platform.
The game MortenKombat, is written in Java using the LibGDX as the framework, this makes the game easily runnable across multiple platforms with no modifications required.

Weaknesses:
Repetitive game play.
According to the survey results in the previous section, most respondents found the game entertaining and also were likely to play the game again however as of now the game play can get quite repetitive when played several times.

[39] https://core.ac.uk/download/pdf/81980848.pdf

Weak character animation/design.
When it comes to the artwork of the game, the animation are rather low quality and that's due to the fact that we had to find free animation online since no one in our group could make them.

Violence in game.
While the game in no way encourages violent behavior, it should be age restricted as we have fights in the game with skeletons and zombies.

Opportunities:
Big gaming market.
The gaming market is already huge generating billions in revenue, and it is expected to keep on growing[40]. There are many gaming genres with the most popular being the like of shooting and RPG games, however there seems to be a new and emerging market for game-based learning which is on the rise and is steadily growing[41]. This gives a great opportunity for a game like ours where it targets a big and well-established market with a new emerging trend.

Free and open source.
We don't plan to sell our game in any way, we are rather aiming to make it a free and open source. Our game will be accessible for everyone to download and play. The code will be up online for anybody what they please with it.

Threats:
Competition.
As we mention that the gaming market is huge which means that we have a lot of competition for the game. There are other well-established games that are educational like ours. Similar games are mentioned in the stake of art section.

Requires Java installation.
For the game to run on the computer, Java needs to be installed. Since our game is depended on Java, it could potentially pose a threat as it is dependent on the program's installation, if Java was to go out of style, this would affect our game.

[40] https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/
[41] http://seriousplayconf.com/wp-content/uploads/2018/07/Metaari_2018-2023_Global_Game-based_Learning_Market_Executive_Overview.pdf

Strategies:

	Opportunities	Threats
Strength	**S1** Game based learning & **O1** Free and open source It could be helpful in promoting the game **S2** Customizable questions & **O2** Big market can target a big audience	**S2** Customizable questions & **T1** Competition customizable questions make it desirable to many. **S3** Cross platform & **T2** requires java installation this problem can be solved by the use of in LibGDX
Weakness	**W1** game play repetitive & **O1** Free and open source possible to improve game through others **W2** weak animation & **O1** free and open source the game is free and possible improvement since it's open source.	**W2** weak animation /design & **T1** competition Hire a designer to improve the animation will reduce competition.

Strengths and opportunities:

S1 Educational game & **O1** Free and open source
The fact that our game is an educational game and it is free and open source should prove helpful to many interested parties. This could be very useful in promoting the game.

S2 Customizable questions & **O2** Big market
since there is exist a big market for games and our game is quite flexible with its content, it can attract a large and varying audience.

Strengths and threats:

S2 Customizable questions & **T1** Competition
While we have a lot of competition in the gaming market, the game has the Customizable questions feature which makes it appeal to many different possible candidates.

S3 Cross platform & **T2** requires java installation
As the game requires Java installation to run on desktop since it's written in Java, it also means it can run on multiple platforms as it is a cross platform. It's additionally *possible when using LibGDX to wrap JRE and the code* which eliminates the need to install java on your machine.[42]

[42] https://github.com/libgdx/libgdx/wiki/Bundling-a-JRE

Weakness and opportunities:

O1 Free and open source & **W1** game play repetitive
While the game play can get quite repetitive as of now if played often, the game is free and it is also open source, so there will be possibility for everybody to improve the game.

W2 weak animation & **O1** free and open source
The animations are pretty weak, but since we are not selling the game that shouldn't be a problem. Additionally, open source makes it possible for anybody to modify and improve the quality of game.

Weakness and threats:

W2 weak animation /design & **T1** competition
Since we have a lot of competition and weak animation, it would be wise to improve the animations. One possible solution would be is to hire a graphic designer to create the artwork for our game.

5. Technical documentation

In this section we talk about the technical aspects of our project. We will go over the choosing of framework to the design and which tools we use and how we use them. We will also talk about the implementation and an explanation of the code.

5.1 Waterfall Model

After putting into words our problem formulation, we made the decision to use the Waterfall model to structure our development process.

First phase is the Requirement Analysis. For that we made a market segmentation and stakeholder analysis, which helped us find out who was our target, and what members we should focus for our game. Then we made the game design requirements clear, so we know exactly what we want from the game to look like at the end. We used the game design section from the book *Java Game development with LibGDX*.

Second phase is System design. After knowing the Game design requirements from the previous phase, we took the output and researched frameworks/game engines in order to choose the best one for our needs. For educational reasons we were demanded to use Java for the project, but it gave us the perfect solution to be able to reach as many users as possible. Using Java, it gives us the opportunity to have a compiled version of the game that works on different operating systems, making the deployment easier for our game.
On top of Java we decided to use LibGDX as a library to implement our game. A framework is used also on top of LibGDX extending the functionality.

Third phase of the Waterfall model is Implementation. There we started developing the code of our game. We used class diagrams to model and understand the code that we were generating. We used the method of reverse engineering by taking inspiration from other open source games and implementing those concepts into our game. Because we working as a group and several members should make contributions to the code at the same time, we used version control. That allowed it to keep the project development on the right speed, as everything new/ changed was pushed so all the members were building on top of the newest version.

The implementation of the project was planned and divided in five different stages. That gave the project the option to have a planification within the member, and crucial information if we were meeting the expected deadlines. By dividing the project in different stages, we identified different functionalities that could be implemented in parallel allowing different members working simultaneously.

1 - 29/10	2 - 06/11	3 - 13/11	4 - 20/11	5 - 27/11	Final - 11/12
Stage of the maps	Creates maze with tiledmaps	Enemies in the dungeon	Interface	Implement questions into our game	Game testing (with interviews and/or surveys)
First Character	Chest that the hero can pick up	Make characters for our fights with different abilities	Create items and inventory	Implement critical hit from questions in the fighting	Improve after making a survey and game-test
		Intro - Choose characters	Main menu, with setup of the game (for example question difficulty)		Randomize things
			Make "system" to read our questions from a txt file		
		Make fight screen: turn-base fight with 1 hero vs 1 enemy	Upgrade the screen: turn-base fight with 3 hero vs 3 enemies		

Table on how the project-work were structured

The fourth and final phase for our project is the System testing. We had made a game testing in the university, where people were playing and afterwards filled in a survey that we had prepared beforehand. This gave us the opportunity to verify that our game was meeting our expectations and the requirement that we initially stabilize. It also gave feedback, and we fixed common issues that was spotted among the testers.

The fifth and sixth phase of the waterfall method has not been developed and are not analyzed. However, we will propose what ideas we have to deploy our code and how we are planning to maintain our code in the section of Future Perspective.

5.2 Game engine/framework comparison

Deciding on the right approach to create the game took a while since we needed to look at all of the available options. Firstly, it would be possible to create the game from scratch, with the addition of different libraries, i.e. by using javaFX. Another option would be LWJGL (Lightweight Java Game Library), it enables game development by enabling cross-platform access to popular native APIs useful in the development of graphics *(OpenGL,Vulkan)*, audio *(OpenAL)* and parallel computing *(OpenCL)* applications[43].

The official page discourages the use of LWJGL for novice programmers and recommends the use of frameworks or engines as they provide high access utilities, we decide to look at frameworks instead of starting with a low-level access library[44].

A framework is a platform for software developers that provides a foundation to be able to build programs for a specific purpose. The framework will provide predefined classes and functions that will process inputs and manage hardware device.
A programming engine is a program that acts as a core or implements essential functionality for another program.

In the table below, we will compare the main frameworks and engines we looked at in terms of their key features and how we can benefit from them in creating the game.
Research is mainly gathered from GitHub, Wikipedia, Reddit and Quora discussions to find relevant info regarding documentation and active community.

Framework/ engine	Cross-platform	3D or 2D oriented	Documentation	community	Remarks
Slick 2D	yes	2D	Official Wiki-page[45]. Forum[46].	GitHub	Good and easy to use however it's limited as its was discontinued some time ago.
FXGL	yes	2D	Wiki[47]. YouTube-tutorials. Example projects[48].	GitHub Gitter-chat	Based on JavaFx. Under development. Mainly for academic and hobby projects.
Mini2Dx	yes	2D	Wiki[49]. Javadocs[50].	Gitter-chat GitHub	Beginner friendly. Fairly new.

[43] https://www.lwjgl.org/
[44] https://www.lwjgl.org/
[45] http://slick.ninjacave.com/wiki/index.php?title=Main_Page
[46] http://slick.ninjacave.com/forum/
[47] https://github.com/AlmasB/FXGL/wiki
[48] http://almasb.github.io/FXGL/
[49] https://github.com/mini2Dx/mini2Dx/wiki
[50] https://mini2dx.org/documentation.html

					Not enough documentation.
JMonkeyEngine	yes	3D	Official website[51]. Wiki[52]. Books.	GitHub Forum	Complex. Suited for 3D game development. Extensive documentation. Engaged and active community.
LibGDX	yes	2D/3D	Wiki[53]. Javadocs[54]. Tutorials[55]. Books.	GitHub Forum	Well documented. Strong community. Feature packed.

Table on Framework Comparison

Having looked at some of the main features of these frameworks/engines, we have narrowed down our choices, we needed to consider our game requirements and make the appropriate choice. These frameworks all have their advantages and disadvantages depending on what you want to do with them. All mentioned frameworks/engines are open source and are free to use, they are additionally cross platform, and most are used for 2D game development except the last two where it's possible to make 3D games as well.
Slick2D is one of the frameworks we came across and researched, it's easy to use and has its own forum. Slick2D is however discounted by the author and has last been updated in 2015[56].

MonkeyEngine is also a popular choice for making games in java, however it is thought to be more complex and is more suited for 3D game development.
Mini2Dx is recommended for simple 2D games, it's easy to use and doesn't require much, the problem is that it's fairly new and not enough documentation exist.
After going through all the options, we concluded that the most appropriate choice for us to use is LibGDX. It's a very powerful java game framework and allows for great flexibility and functionality with game development. It's thoroughly documented and is well maintained.

We used the book Java Game Development with LibGDX From Beginner to Professional by Lee Stemkoski to understand LibGDX and get started with the game development. We chose this book since it's a recent release (2018) and the most up to date version available to us.

[51] http://jmonkeyengine.org/
[52] https://wiki.jmonkeyengine.org/
[53] https://github.com/libgdx/libgdx/wiki
[54] https://libgdx.badlogicgames.com/ci/nightlies/docs/api/
[55] https://libgdx.info/
[56] http://slick.ninjacave.com/

5.3 Design

In this section we will discuss the design behind the game. The tools used to make the maps and some of the graphics. How we got the music and where we found the animations and other assets. We also talk about the interface and how the screens are organized and what they display.

5.3.1 Graphic and audio design, interface

First of all, we talked about making the artwork ourselves. But since none of us has any experience in making game assets/sprites, and we wanted to focus on making the code for the game, we decided to find and use open source game art.

We used various different open source game art. The websites we used to find almost all of the sprites and assets to make the maps, are called opengameart.org and itch.io. On opengameart.org pixel artists upload their assets for free use. The artist can decide how his/her art should be credited or not, and what you can use their art for (Fx for commercial use, private game source etc.). itch.io is similar to opengameart.org, the only difference is that only some of the assets are free.

Figure 22: Screenshot of www.opengameart.org

To make the maps, we used a program called Tiled, which is a tile map editor. Most of the game assets were scaled in Adobe Photoshop Elements 18 so it would fit our 32x32 pixel game. When the map layout was done, object layers were used to create solid objects along with spawn points, enemies walking around in the map, animations etc. Finally, the maps got imported in the game.

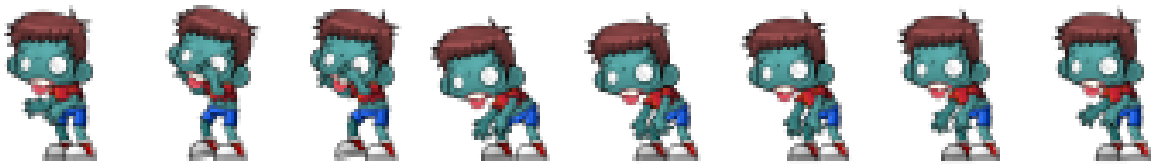


Figure 23: Sprite-sheet example from our game.

For the MenuScreen, the buttons were found on opengameart.org and the background pictures are from other games.

Finally, we have the FightingScreen. The characters and their animations were taken from different websites, which is linked in the references with artwork and music. Background picture is from Darkest Dungeons as this is our main inspiration.

All of the music used in the game, is taken from a popular game called ‘The Witcher 3: Wild Hunt’, ‘Monty Python's movie “Always look on the bright side of life” and the Star Wars VI movie “Victory

Celebration". We decided to borrow the music from other games and movies, because we don't have any experience in composing music and we saw it fitting for the game.

5.3.2 Tools used

Tiled

Tiled map editor is a free to use program to create 2D pixel maps for games. Although its primary feature is to edit tile maps, it also supports free image placement and other ways to annotate a level with extra information used by the game. Tiled is a very flexible program to use.

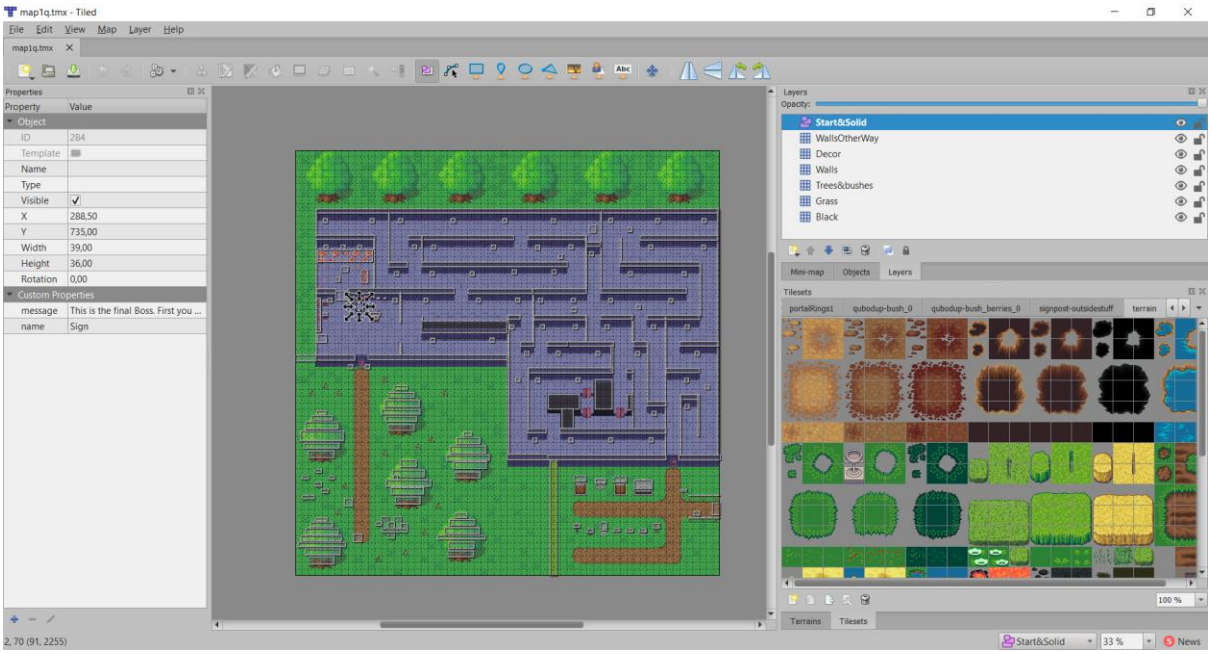


Figure 24: Tiled map screenshot: https://www.mapeditor.org/

Tiled supports straight rectangular tile layers, staggered hexagonal and projected isometric layers.
We use this program to create the layout of the maps in our game along with the initial position of our enemies, signs and any other actor from our maps.
When creating maps in Tiled, the assets/sprite sheets can be downloaded or created and added to the Tiled program. Then stamp brush is used to "draw" the map. The stamp brush allows efficient painting and copying of tile areas. There are several selection tools as well.
Beside the layer tools, there is also an object layer tool. In the object layer can be created solid objects, animations or in general other objects. Since Tiled works nicely with LibGdx, the object layers can easily be implemented to our game.
When the maps are finished, it can be saved as an .tmx file which LibGdx then will use to load the maps in the game.

Photoshop

Adobe Photoshop Elements 2018 is a licensed photo editor program. In our project, photoshop is mostly used for scaling the assets used in the maps so it can be used in Tiled. The buttons, text, tutorial, pause - winning - and losing screen are also made in photoshop by editing images found online.

5.3.3 Graphics and audio elements

As mentioned in section 5.5.1, we use music from The Witcher, Monty Python and Star Wars.

For the graphics, we use free assets or other open game source art. As already mentioned, most of the sprites and assets are found on ithc.io and opengameart.org. In the beginning of the project we wanted an overall theme for our game, and for our maps and sprites to have the same art style. But since we only use free assets, it is hard to find everything to have the same art style. That is why the graphics from each map can vary.

Making the fighting screen, we had to find some sprites for idle, attack and dead animation to use as our fighting actors. They have to be a specific size, since scaling the sprites sometimes made the animations worse. Again, for the fighting actors, we found them online for free (Links attached in the references for the artwork and music section).

We found a lot of different sprites to use, but most of them, when we animated them, didn't work well. After a deeper research, we found the assets needed and we implemented them in the game, although the design was not consistent. Some of the animation are not smooth in our game. The reason of this is when we use the animation are placed in tables, and if the animation does not fill the table it will be scaled. We decided to leave this at it is, as we should be able to solve this if we could get the animation sprite with a determined amount of pixels.

When talking about style, the overall theme of our game is a Dungeon theme. Each map has its own style, to make them easily recognizable and to help the player distinguish its location. The idea behind the map layouts is that the player has the feeling of going up a mountain where one of the bosses is located, or the player can go down to a dungeon where the remaining two bosses are located.

The enemies in the map have to be linked with the dungeon theme. A bat was the first thing to come to mind. After a short discussion we concluded it should be something undead and ended up being skeletons and zombies. But since we use free sprites, we couldn't find a skeleton where the animations would fit properly - Only for the fighting screen. So instead, for the mazes, we used a goblin sprite.

For the fighting screen we use a troll instead of a bat since we didn't find a sprite sheet for a bat with animations fitting for the fight.

At last, the enemies are placed in the maps where we saw them most fitting. The bats are in every map, the zombies are in the outdoor themed maps and the goblins are in the dungeon themed maps.

5.3.4 Interface, Screens/Menus

Main Menu:

When opening our game, the first screen the player sees is the Menu. The Menu has some options to choose from as described in the picture below:

Figure 25: Screenshot of MenuScreen.

Settings Menu:

Having clicked on the settings, the player will be directed to a new screen; The settings screen.

Here it can be chosen the difficulty of the game. The quick version has fewer enemies in each map, than the normal version. Here it can also change the volume for the in-game music and in the bottom left corner there is a back button, so you can go back to the Main Menu.

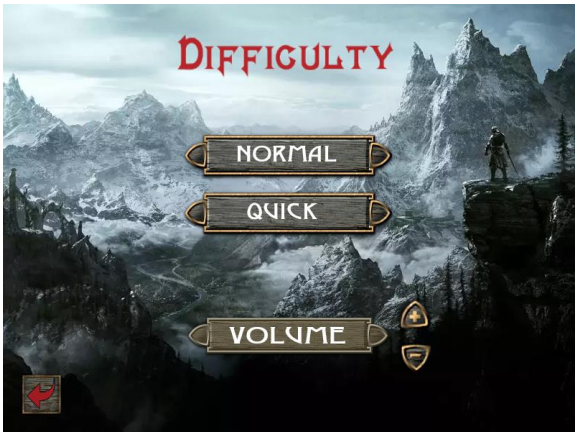


Figure 26: Screenshot of OptionsScreen.

Character Selection:

If start button in the Main Menu is pressed, the Character Selection screen appears.

This screen has the Drag & Drop ability where the player should choose 1 Fighter, 1 Mage and 1 Support. by simply clicking and holding on the cards and dragging them into the empty slot. This screen also has the back button and a start button which takes the player to a new screen.

Figure 27: Screenshot of SelectionScreen.

Tutorial & Story:

When the characters have been chosen and pressing the start button, it a small Tutorial. The tutorial is 3 different images describing how to play the game. In the bottom of the screen, a text flashes repeatedly saying that space should be pressed to continue. By hitting the space key, the player can go to the next image.

After the tutorial, a small story consisting of also 3 images will appear. When the player has been through the tutorial and story, the game begins.

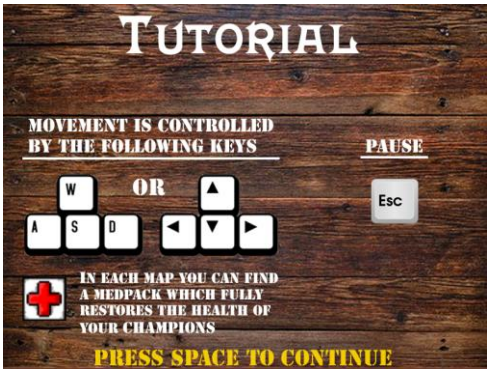


Figure 28: Screenshot of the Tutorial

Inside the Maps:

In most of the maps, there is an exit. Some of them even have two. Map 4, 6 and 7 are boss maps and there are no exits. After defeating the bosses, the player had. The exits also work as a ´Go Back´ to the previous maps.

Each map will have a sign, explaining to the player what map they are in and how many exits they should look for. Simply walk up to any of the signs, and the text will pop up on the screen.

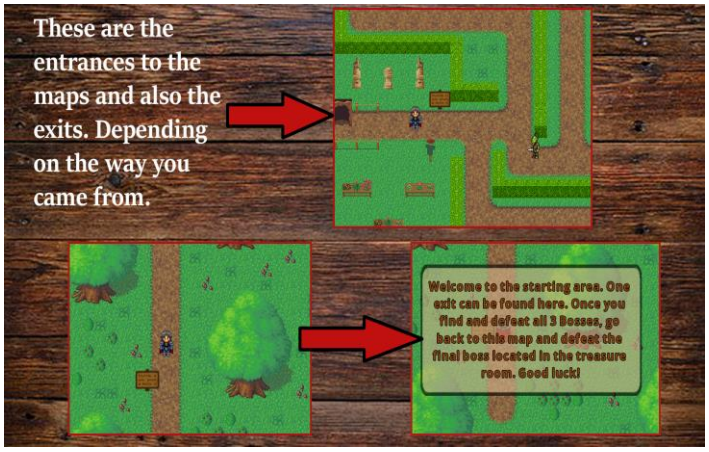


Figure 29: Screenshot of in-game signs and entrance.

Pause:

By pressing the esc key, a pause menu pops up. This will set the game on pause. The player can continue the game, change the volume of the in-game music and can exit the games to windows.

Figure 30: Screenshot of the PauseScreen

Fighting Screen:

Every map will have enemies walking around. By colliding with any of them the fighting screen will open. As described in the image below, there are different attacks during each turn. By hovering with the mouse, the different attacks, a description of them shows up. The Support and Mage both use Mana for their attacks. The blue text indicates the mana.

It can be seen who has the turn by their name. The one having the current turn, has a green name.

The health is indicated by the red numbers. If a character receives damage, the health flashes 3 times.

Finally, the label in the middle in the top part of the screen, shows how many turns have been taken. The turn counter goes up by one every time all of the characters had attacked. Including the enemies.

Figure 31: Visual explanation of the FightingScreen.

Trivia Questions:

During a fight, there will be a 30% chance to have a trivia attack. This means a trivia question will pop up which should be answered to continue. In the bottom, it is described whose turn it is and what happens if the answer is right or wrong.

If you answer right, your character attacks twice and if you answer wrong, you lose the turn.

A trivia question can also appear on the enemies turn. The same rules are applied for them too, if the answer is right or wrong.

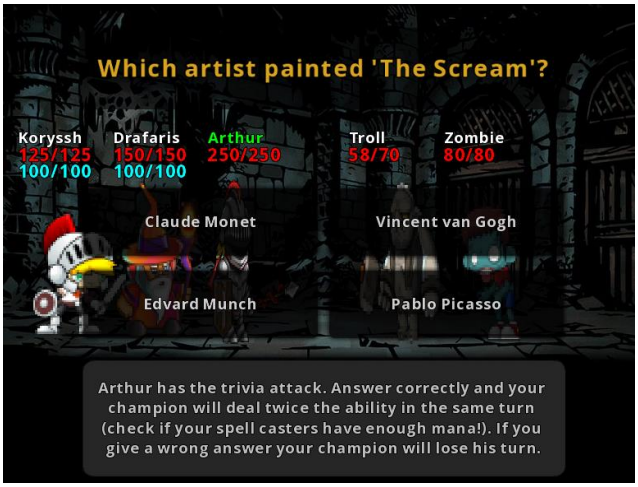


Figure 32: Screenshot of a trivia attack.

In the image below it can be seen, if the answer is right it turns green. But if not, it turns red.

Figur 33: Screenshot of a trivia attack with the correct answer.

Enemies & Bosses:

There are 3 bosses found in 3 different maps which you have to be defeated, to battle with the final boss. The bosses have more health and better attacks than the enemies found in the maps. When fighting a boss, there are no other enemies to fight as seen in figure 34.
But for the enemies in the map, different rules are applied.
Fighting a Bat summons 1 enemy.
Fighting a Goblin summons 2 enemies.
Fighting a Zombie summons 3 enemies.

Figure 34: Screenshot of one of the boss fights.

5.4 Implementation

In this section we will describe the code of our project and the structure that our game follows.

The project is developed on Java with the use if LibGDX libraries and the framework from the book *Java Game Development with LibGDX*[57].

The book gave the project a big opportunity to check developed code within different games exposed in the chapters. We used reverse engineering to check how things were implemented during the samples, for example the movement of a character in a map implemented in our game was highly influenced from the chapter 13 of the book.

Before checking for code samples at the book 'Java Game Development with LibGDX' we tried to learn LibGDX by reverse engineering the samples found at LibGDX official wiki page[58]. We tried to follow the execution of the code by following the instructions and learning the meaning and utility of some of the classes of LibGDX. It gave the members basic knowledge on how code was structure and was crucial to learn the first steps of LibGDX.

The code was implemented by several members and to be able to have a consistent code during the process we used GitHub. Using a version control software will allow us to code in our local machine and have a master copy of the code in a server where all changes are saved. We have followed 'best practice' advices from the University of Washington Computer Science & Engineering regarding to how we should use a version control software.[59]

57 https://books.google.dk/books/about/Java_Game_Development_with_LibGDX.html?id=ZcFHDwAAQBAJ&source=kp_book_description&redir_esc=y

58 https://github.com/libgdx/libgdx/wiki/Running-Demos

59 https://homes.cs.washington.edu/~mernst/advice/version-control.html

Version control best practice:

- Use a descriptive commit message when making any change in the project. It will give information on what has been changed on each version, and in case that we need to update until a previous version, it will give us a description on what version we are updating.
- Each commit should have a single purpose and should completely implement that purpose. This will make easier to spot any error or bug in the code. The utility of the version will be compromised if one commit contains code that solves different purposes.
- Make sure that you are working with the latest version of the project. Update your project every time before making any changes to modify the latest version.
- Coordinate with your team-mates. Working at the same time in the same file might generate some conflict that has to be solved afterwards. To avoid this, coordinate with your team so they are not working in the same file.
- Don't commit generated files. Version control is intended to be used on files that the user modify. To avoid this, use "gitignore" so files generated from Windows, MacOS, IntelliJ, Eclipse… are not included in the commits. [6]

In order to understand how the flow of our code works, we will use class diagrams. This reveals how the classes interact with each other, when a class is created, when they access to variables that belongs to another class etc. Those diagrams were generated from the code with IntelliJ IDEA class diagram functions. Some of the diagrams will be exposed in the next chapters. More detailed diagrams can be found at the appendix 1 to 4.

5.4.1 Structure

The project is organized in such a way that it is split into several packages. Each package contains related classes. Figure 35, is a screenshot we took from IntelliJ IDEA which contains all of the project files.
The packages will be briefly explained in this section along with their contents, in the upcoming sections we will expand upon each package and go in details about the main classes it contains and the methods.

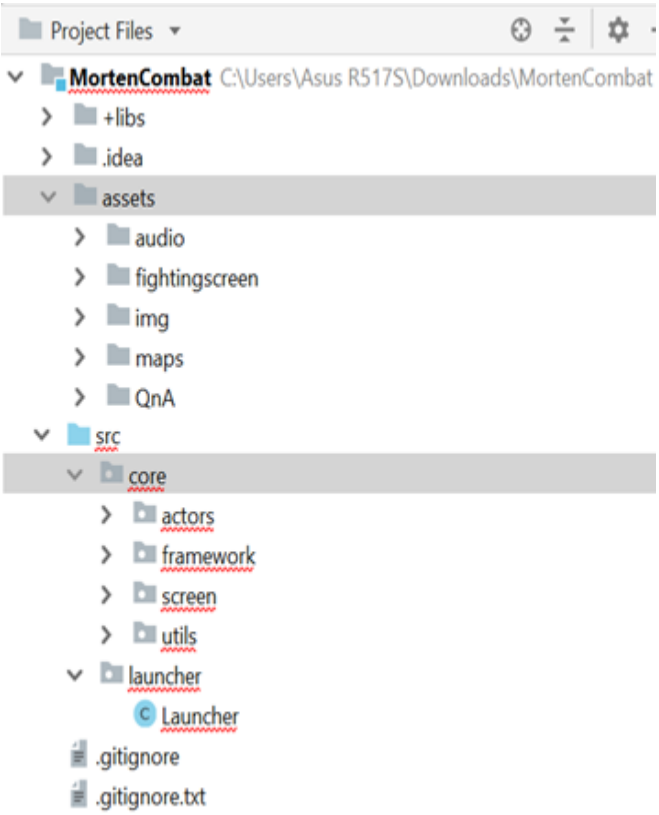


Figure 35: Overview of the packages.

Assets

Assets folder stores all the external files used in the project, for example in our case, the assets include 5 folders:

- Audio, which includes the game's background music and the different sound effects.
- Images, all the kinds of images we used to create the game.
- Fightingscreen, images specific (sprite sheets) for the FightingScreen class fighters.
- Maps, images specific to map design, include .tmx files for tiled maps.
- QnA, .txt file for questions and answers.

Framework

Framework package contains framework specific classes, the code in here is not written by any of the members, it is taken from the book, Java Game Development with LibGDX From Beginner to Professional by Lee Stemkoski. Some of the framework classes where modified by the project members to give a certain functionality that was missing for our project.
This is a short description of the functionality that the framework brings to the project:

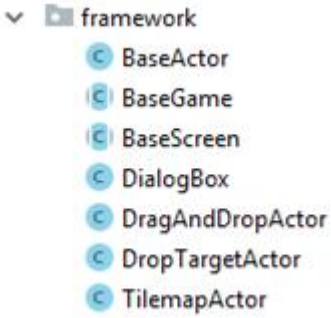


Figure 36: Overview of what the Framework package contains.

- BaseActor, extends functionality of the LibGDX actor class by adding support for texture/animation, movement, world boundaries and camera scrolling.
- BaseGame, it manages screens that appear during games.
- BaseScreen, it interacts with the actors that are created in the stage of the Screen. It will check each frame if there is any modification in the property of the actors and render any of those modification into the computers screen.
- DragAndDrop, it allows for BaseActors to be 'draggable'.

- DropTargetActor, class extends BaseActors that will allow the creation areas that can interact with DragAndDrop classes.
- TilemapActor, class that will read a tiledmap file (*.tmx) and will create the functionality to read objects and their corresponding properties that has been declared in the tmx file.

Utils package

The utils package consists of various important classes and a sub package, menu, with three other classes.

Utils package is where the game starts as it contains the MortenKombat class.

The MainMenuScreen class is the main screen utils class. This is the first screen in the game, it contains 3 buttons leading to other menus or exiting game, and it also includes the music and the volume for it.
OptionMenuScreen, is the option screen class, contains 5 buttons and the possibility to change music volume.
MapLayout, is an enum that establishes the map layout, it contains the data structure to connect the maps and how the player can move between them.

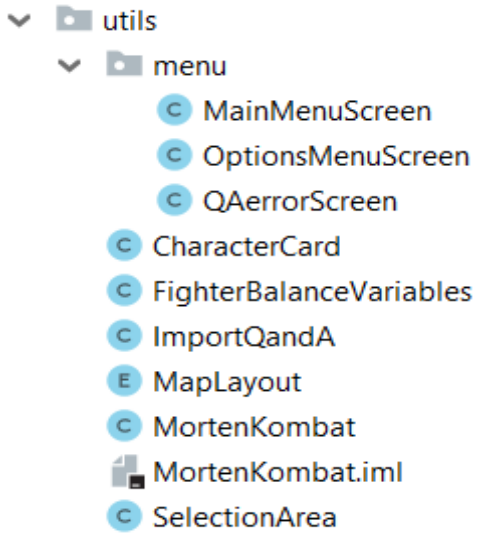


Figure 37: Screenshot of Screen package classes, IntelliJ IDEA.

Screen package

The screen package contains screens classes used throughout the game, the FightScreen is the longest and most complex class we have in the code. Only ExploringScreen, FightScreen and SelectionSCreen are going to be explained in this package. The rest of the classes are a simplified version of the previous ones.

ExploringScreen, represents the class where the player explores the dungeons created from the tmx files. It all sets how the actors should interact with each other.

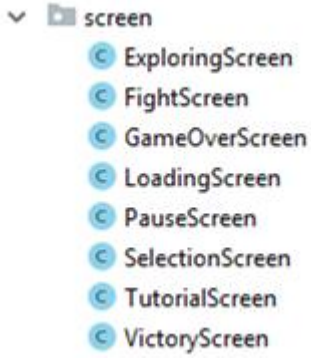


Figure 38: Overview of what the Screen package contains.

The class SelectionScreen allows the player to choose between 6 different characters for the game. This selection is saved at the MortenKombat class, and is been read afterward at the FightScreen to know which actors need to be created.

FightScreen, is the screen that is going to be active when the player collides with an enemy in the ExploringScreen. The class creates a turn-based fight between the champions selected in the SelectionScreen and the enemies. To implement this the class deals with the interaction of up to 6 different fighters. Also, the educational aspect of the game is introduced in the fight, making each action

of the fighters and the enemies has a probability to activate a multiple-choice question. The player is rewarded if he gives a correct answer, or is punished if he answers wrong.

Actor package

The actor package is split in two parts, exploringactors and fightingactors sub-packages as seen in figure 39. The files in the screenshot are not expanded because there are too many actors. The exploringactors contain the actors seen in the exploring map screen

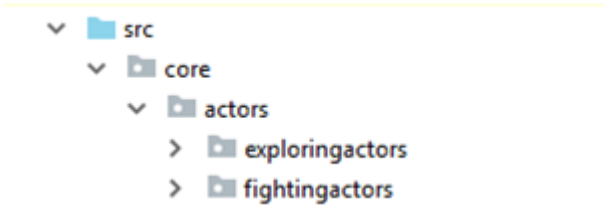


Figure 39: Screenshot of Actor package classes, from IntelliJ IDEA.

such as bats, zombies, skeletons, etc., while the fightingactors contains actors' classes that appear in the fighting screen such as warrior one, warrior two, mage one, support one, etc.

Project flow-chart

Before explaining in details, the implementation of the project MortenKombat, we will describe the flow of it for better understanding of main parts and providing a simplified overview.

The first class that is called when executing the program is the Launcher. It creates an instance of the object LwjglApplication which is the one responsible for rendering and creating the window at the operative system. Also, the game class MortenKombat is generated at this class.

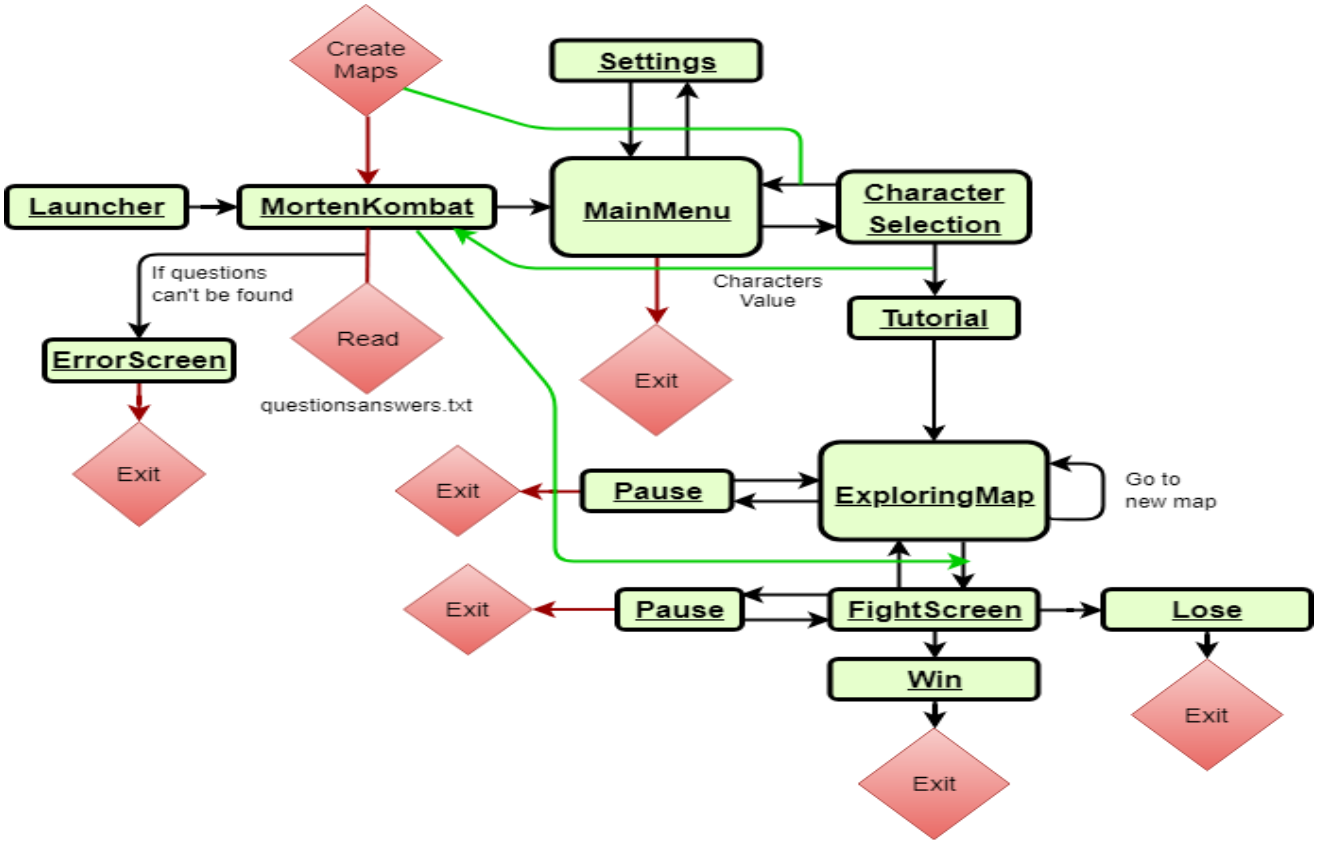


Figure 40: The project flow-chart.

After this is done MortenKombat is having the control of the program. It reads the questionandanswer.txt file and store that data in the game that will be used in the game. If it detects any problem with the file, such as it is not finding the file or it can't even create one question, it sets as an active screen; the ErrorScreen class.

If no problems are detected, it sets the active screen to MainMenu where the player can run into settings, or start the gameplay by pressing start button to activate the CharacterSelection screen.

When the CharacterSelection class is called we create all the ExploringMap screens. These maps are split in two collections: Quick mode and normal mode. In the OptionMenuScreen the player can choose which mode it will be loaded. If the player does not choose anything, the normal mode is loaded by default. In the CharacterSelection the player selects what champions he is going to use at the fight during the game. After this step we run into the first ExploringMap by activating the screen at the game class.

In the exploring map, the player navigates through a binary tree of inter connected ExploringMaps. In these maps, he is able to interact with doors that send him to another ExploringMaps, medic packs that heal their fighting Champions and with enemies that are moving through the maps. In case that he collides with an enemy a FightScreen is set to active.

In the FightScreen the player fights the enemies with the previously selected Champions. From this screen, it can move to three different screens depending on the state of the fighters that are involved in it. If all the champions die, the screen is changed to GameOver screen, but if the player beats all the enemies it returns to the ExploringMap and removes the actor that the player collided with previously at the ExploringMap. The last case is if the player beats the final boss during the FightScreen then we a VictoryScreen is shown.

For more complex diagrams, a class-diagram can be checked at the appendix.

5.4.2 Code explanation

Utils package

The utils package is the first package that is run in the project and interconnect all the screens, actors, and files that needs to be run to setup the complete game.

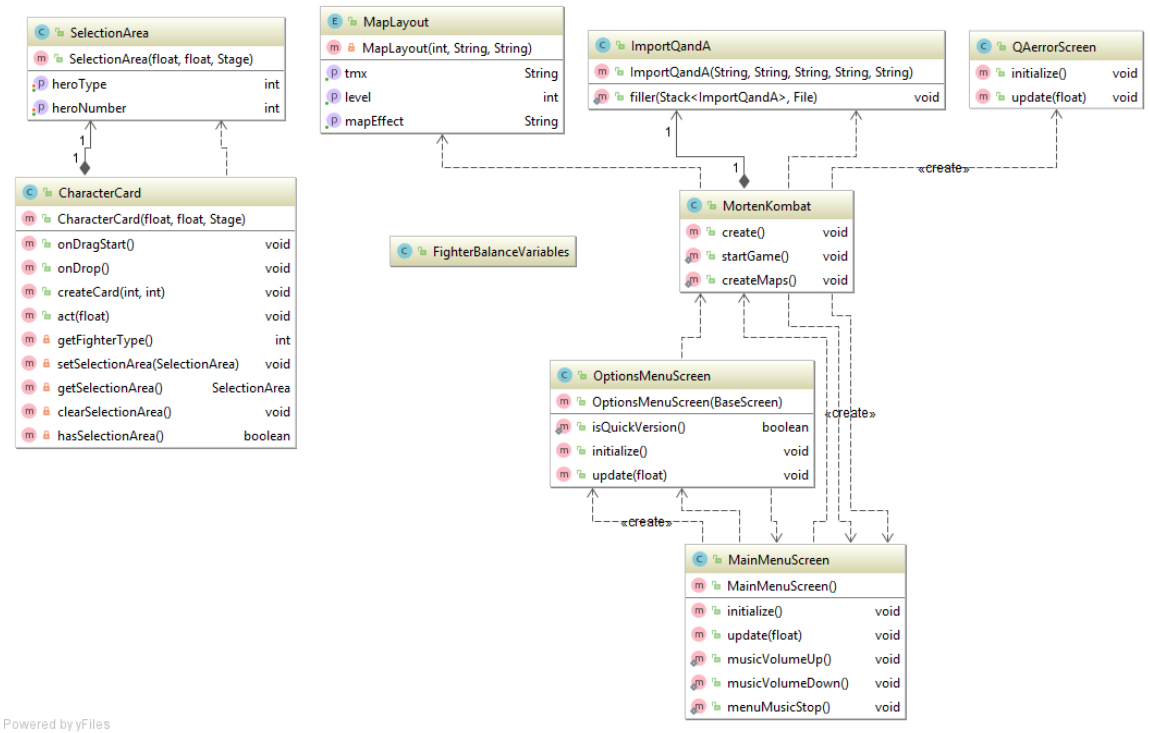


Figure 41: This is the utility packages with Menus

Exploring screen creation

We have two versions of the game, in the settings menu screen there are options for selecting what version the game should create. The shorter version contains less enemies and is, therefore, easier to complete. When the player goes from the screen MainMenu to CharacterSelection the maps are created and the layout is been read at MapLayout enum class.

When creating the maps, we need to know what is the previous map that the map is connected too, what is the location of the 'tmx' file that will load the map, and last if the map as any environmental effect such as dark, wind or ice effect. This information is stored at the enum MapLayout and by doing this it should be easy to modify in the future the enum to create different layout for our game by changing few lines of code.

```
56      public static void createMaps(){
57          int i = 0;
58
59          for (MapLayout map : MapLayout.values()){
60              ExploringScreen.mapName = map.getTmx();
61              ExploringScreen.mapEffect = map.getMapEffect();
62              if (map.getLevel() == 0){
63                  layout[i] = new ExploringScreen();
64              }else {
65                  layout[i] = new ExploringScreen(layout[map.getLevel() -1]);
66              }
67              i++;
68          }
69      }
```

Figure 42: This is how the ArrayList of maps is created and how the two constructors in ExploringScreen is used.

The code above creates all the maps that we have stored in the enum, and makes sure that the first map that is been created used another constructor as it does not have any previous map that it extends.

```
44      public ExploringScreen()  {
45          this.previousMap = null;
46
47          backgroundMusic = Gdx.audio.newMusic(Gdx.files.internal( s: "assets/audio/music/backgroundmusic.mp3"));
48          backgroundMusic.setLooping(true);
49      }
```

Figure 43: This is the first constructor, it creates the first map with all connections to null. Also creates the music for the exploring maps.

Reading questions

The game will read all the questions presented during the fights from a .txt file in the assets folder. The file will follow the following format, repeating every 5 lines, first the question, then next line will have the correct answer to the question, and the next three lines will have wrong answers.

```
40              dataReader = new Scanner(data);
41              while (dataReader.hasNextLine()) {
42                  if (i == 0) {
43                      question = dataReader.nextLine();
44                      i++;
45                  } else if (i == 1) {
46                      answer = dataReader.nextLine();
47                      i++;
48                  } else if (i == 2) {
49                      wrongOne = dataReader.nextLine();
50                      i++;
51                  } else if (i == 3) {
52                      wrongTwo = dataReader.nextLine();
53                      i++;
54                  } else if (i == 4) {
55                      wrongThree = dataReader.nextLine();
56                      i++;
57                  } else if (i == 5) {
58                      stack.push(new ImportQandA(question, answer, wrongOne, wrongTwo, wrongThree));
59                      fileError = false;
60                      i = 0;
61                  }
62
63              }
64              //pushing last question after reading.
65              if (i == 5) {
66                  stack.push(new ImportQandA(question, answer, wrongOne, wrongTwo, wrongThree));
67                  fileError = false;
68                  i = 0;
69              }
```

Figure 44: This is the method that creates the questions. It creates an object with the questions and answers and puts it into a stack.

The game reads the file and adds a question every 5 lines to a stack, when the file is read through it shuffles the stack with the questions and it makes a backup.
The backup is used if the game goes through all the questions. If that condition is meet, the question stack that is used at the FightingScreen uses the backup to get all the questions back into the game. Doing this we ensure that the player will never repeat a question unless it is necessary.

If the game can't find the file or the file doesn't contain a valid question, the game shows a 'file not found' screen on the launch of the game. It indicates to the player that an error has occurred and that it should check the README file of the game.

Fighter Balance class

This class contains the variables for the fighters and their attacks, this makes it easier to balance the game and get an overview of the different fighters.
It also makes the classes that uses these variables easier to maintain, giving any future coder a descriptive variable instead of a random integer.

```
public MortenFighter(Stage s){
    super(s);
    this.setHP( FighterBalanceVariables.MORTENHP);
    this.setMaxHP(FighterBalanceVariables.MORTENMAXHP);
    this.setFighterName(FighterBalanceVariables.MORTENNAME);

    attack = AnimationCreator.createAnimation( path: "assets/Fightingscreen/Boss/Morten-Attack.png", frameDuration: 0.14f, rows: 1, cols: 4);
    iddle = AnimationCreator.createAnimation( path: "assets/Fightingscreen/Boss/Morten-Idle.png", frameDuration: 0.14f, rows: 1, cols: 8);
    dead = AnimationCreator.createAnimation( path: "assets/Fightingscreen/Boss/Morten-Dead.png", frameDuration: 0.14f, rows: 1, cols: 9);

    setAnimation(iddle);

}
```

Figure 45: This is one of the bosses. It shows how it gets the HP and name from the fighter balance variables.

Menu

The first thing the player sees is the MainMenuScreen. In this screen the player has three buttons for play, settings and exit, the play and settings buttons create new screens and switch to those screens.
In the OptionsMenuScreen the player can change the volume of the music played throughout the game, which version of the game the player wants to play and a back to the main menu button.
Going back to the main menu deletes the current screen, both from the SelectionScreen and the SettingsScreen. This is to prevent the game from having multiple menus loaded at the same time and taking up memory.
The play button creates the CharacterSelectionScreen and all the ExploringMaps for the given version and switches to the SelectionScreen.

All the menus have the same structure with a background picture and a few buttons. For the buttons we create an event listener, so the player can interact with the buttons by mouse clicks. The class reads those clicks and executes certain code after that.

Actors Package

We have put the actors for our game in a main package named Actors. This main package is composed of two sub packages: exploringactors and fightingactors. In the first one we have put all the elements, that are taking part in the ExploringScreen, and in the second one we had put all the actors taking part in the FightingScreen.

Exploring Actors

When explaining the explorinactors we will focus on the class Hero, as it is the most complex one in the package and every other class that is part of the package is either using the same methods or is a simplified version of that.

Hero

Figure 46: Sprite-Sheet of the Hero Actor.

The Hero class is extending the class BaseActor, which is part of the framework we are using. (Described in 5.6.1 - Structure).

This class is copied from the book. (Treasure Quest - Chapter 12) Class Hero is creating an object in the ExploringScreen which is representing our hero. The hero object is used to navigate between the maps. All the movements of the object on the maps, are simulated using animation sprites, which are static images. An animation consists of multiple frames which are shown in a sequence at set intervals[60]. Therefore, our hero can simulate moving in four directions on the maps.

The animation sprites are rendered with a loop. The texture is divided evenly into rows and columns considering the width and the height of the frame. After that, within a loop, the frames are rendered sequentially at preset intervals. In the same class it sets the camera angle, so on the exploring screen we always see the picture from the top. Inclusively, it's specified which animation should be rendered in the starting position of every map and the facing angle.

[60] https://github.com/libgdx/libgdx/wiki/2D-Animation

```
30      public Hero(float x, float y, Stage s)
31      {
32          super(x,y,s);
33          this.s = s;
34          String fileName = "assets/img/hero.png";
35          int rows = 4;
36          int cols = 4;
37          Texture texture = new Texture(Gdx.files.internal(fileName),  useMipMaps: true);
38          int frameWidth = texture.getWidth() / cols;
39          int frameHeight = texture.getHeight() / rows;
40          float frameDuration = 0.2f;
41          TextureRegion[][] temp = TextureRegion.split(texture, frameWidth, frameHeight);
42          Array<TextureRegion> textureArray = new Array<~>();
43          for (int c = 0; c < cols; c++)
44              textureArray.add( temp[0][c] );
45          south = new Animation(frameDuration, textureArray, Animation.PlayMode.LOOP_PINGPONG);
46          textureArray.clear();
47          for (int c = 0; c < cols; c++)
48              textureArray.add( temp[1][c] );
49          west = new Animation(frameDuration, textureArray, Animation.PlayMode.LOOP_PINGPONG);
50          textureArray.clear();
51          for (int c = 0; c < cols; c++)
52              textureArray.add( temp[2][c] );
53          east = new Animation(frameDuration, textureArray, Animation.PlayMode.LOOP_PINGPONG);
54          textureArray.clear();
55          for (int c = 0; c < cols; c++)
56              textureArray.add( temp[3][c] );
57          north = new Animation(frameDuration, textureArray, Animation.PlayMode.LOOP_PINGPONG);
58          setAnimation(south);
59          facingAngle = 270;
```

Hero › Hero()

Figure 47: This is the code for the animations of our Hero's movement.

The method setBoundaryPolygon, is used to create a polygon around the actor. By using this polygon instead of squares will create better collision filling at the ExploringScreens.
At the beginning of the constructor of our Hero, are initialized the values of the acceleration, speed and deceleration of the movement of the object. Considering the complexity of the game, those variables are initialized as **private final integers,** so an easier future alteration is possible.
The class Hero consists of two methods as well: act() and createLight().
The method create() is setting the direction of the animation and pausing the animation when the object is not moving.
The method createLight() is creating the effect of darkness in the Dungeon maps (5,6 and 7). As only for those maps a radial gradient image is layered over the Hero to create the illusion of a lightened area around it.
As said in the beginning of the chapter, the explanation of the other classes part of the package ExploringActors will be omitted as they are a copy of the class Hero or a simplified version of it.

Fighting Actors

In the sub package FightingActors consists of all the objects used in the FightingScreen. When explaining this sub package, we will focus on the hierarchy of the classes enclosed and the explanation of the most complex class of all, as the others are containing the same or less functionalities.

Hierarchy

Inheritance allows programmers to create classes that are built upon existing classes, to specify a new implementation while maintaining the same behaviors, to reuse code and to independently extend original software via public classes and interfaces[61].

The hierarchy of classes in the sub package FightingActors is of a major importance for the proper execution of the game, as on this hierarchy is based the turn-based fight in our game.
Moreover, the hierarchy is important because it gives the possibility to make changes in the game faster and without going deeper in the code. For example, in the FightingScreen there are 9 buttons with the abilities of the Champions. Those buttons are initialized without any text values in the abstract class, Champion, and therefore inherited by the sub classes of Champion. This hierarchy gives the possibility to alternate the code in the future easily, adding or removing functionalities, and keeps it open to contributors.

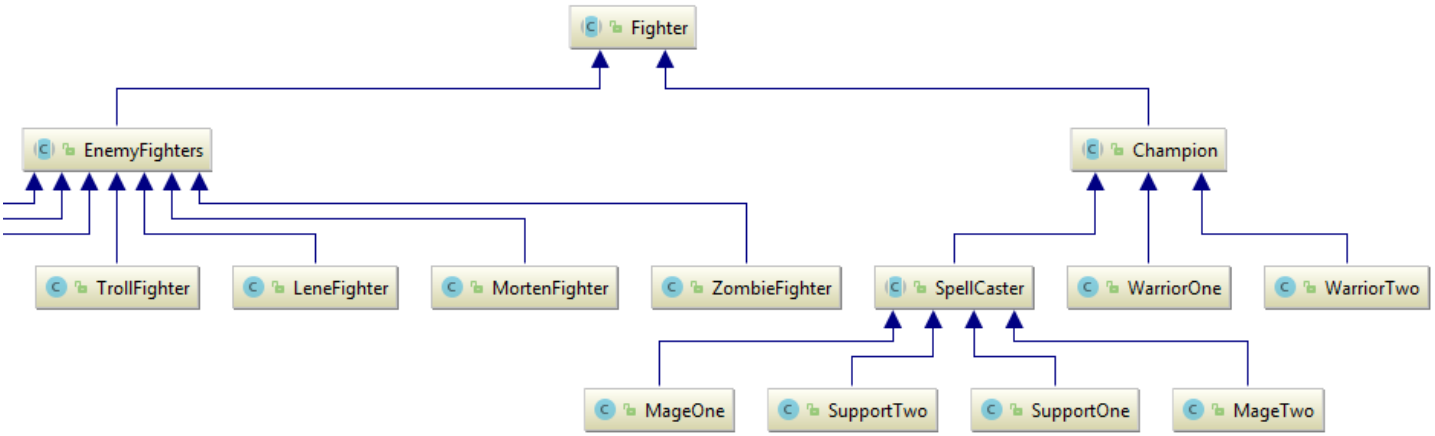


Figure 48: This is a picture of the fighter classes and their hierarchy.

Class Fighter

The class Fighter is an abstract class which unifies all the Actors in the FightingScreen. Every actor in the FightingScreen should have the following *attributes on the screen*: name, health points at the moment, max health points, animation depending of action (idle, attack, dead). Considering that, in the class Fighter we have declared variables satisfying those conditions such as HP, maxHP, fighterName, namePlate, etc.

[61] https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)

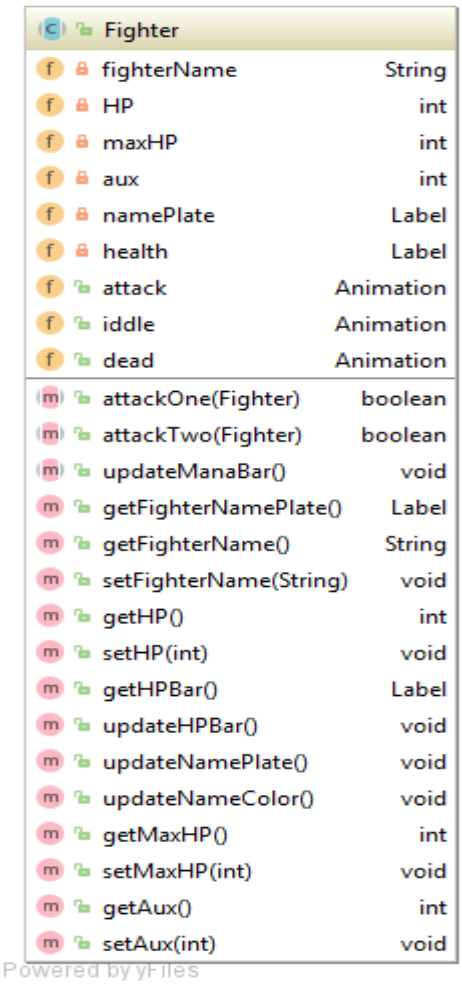


Figure 49: This is the fighter class with every field and method.

In the abstract class Fighter, we had implemented several methods that are common for both the EnemyFighters and the Champion. The methods attackOne and attackTwo are abstract Booleans. By doing so, in the fighting screen, where the methods are called, we are checking if the result is true or false, depending on that the ability they are implementing can or cannot be executed. When implementing those abstract methods in the subclasses we will performance a check if the attack has the right target (ally or enemies) for example.

On the FightingScreen we have three Champions(chosen by the player) and between one and three enemies (explained in the Screen Package afterwards). Both class Champion and **class EnemyFighter** are abstract classes and inherit all included in the abstract class Fighter.

EnemyFighter class does not create any new variables or implementation than the one declared at Fighter class. The main reason that we have extended this class from Fighter is to be able to check if a fighter at the FightingScreen is an instance of Champions or if an instance of EnemyFigher. Also, for a future perspective by doing this it will be easier to implement some functionality that only should be present in the enemies. All classes extending EnemyFighter have the same structure and methods but hold different variables.

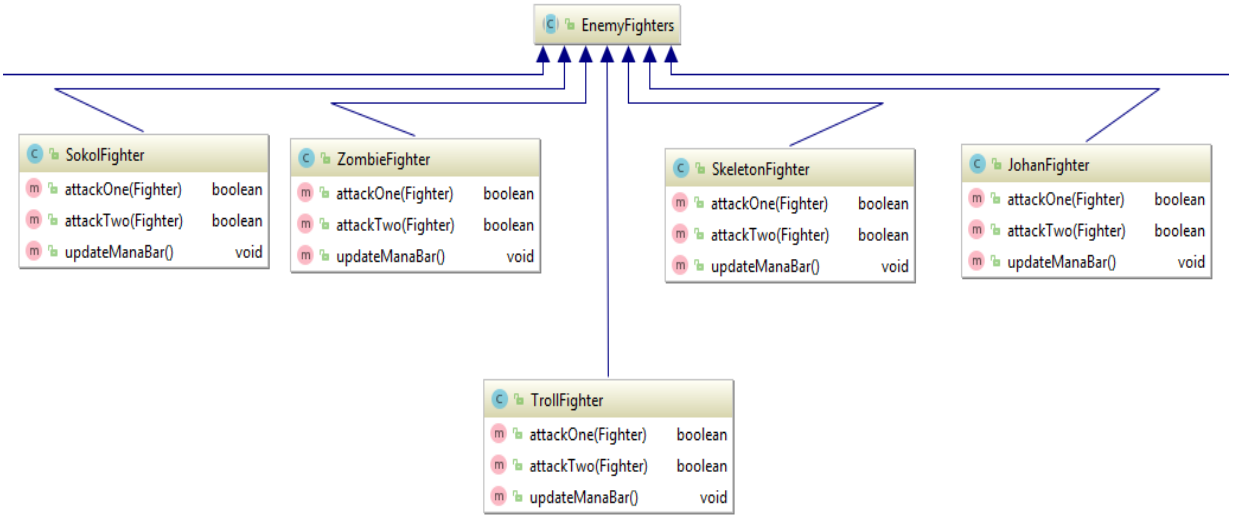


Figure 50: This picture shows some of the enemy fighters, their methods and their hierarchy.

The class Champion will bring some utilities that are common to all allies' fighters. It will create the buttons that the user has to interact with in the FightScreen. It creates aso AttackThree as the champions have one more ability that the EnemyFighters. AttackThree takes three arguments instead of one compared to the AttackOne, the reason behind this is to have the option of creating attacks that will affect multiple fighters at once.

There are three types of Champions characters a team is made of: Warrior, Support and Mage. Of those three only the Support and the Mage type have Mana as an attribute so this variable does not need to be declared in the abstract class Champion. Those variables would be declared in the abstract class SpellCaster which is extending the abstract class Champion as well.

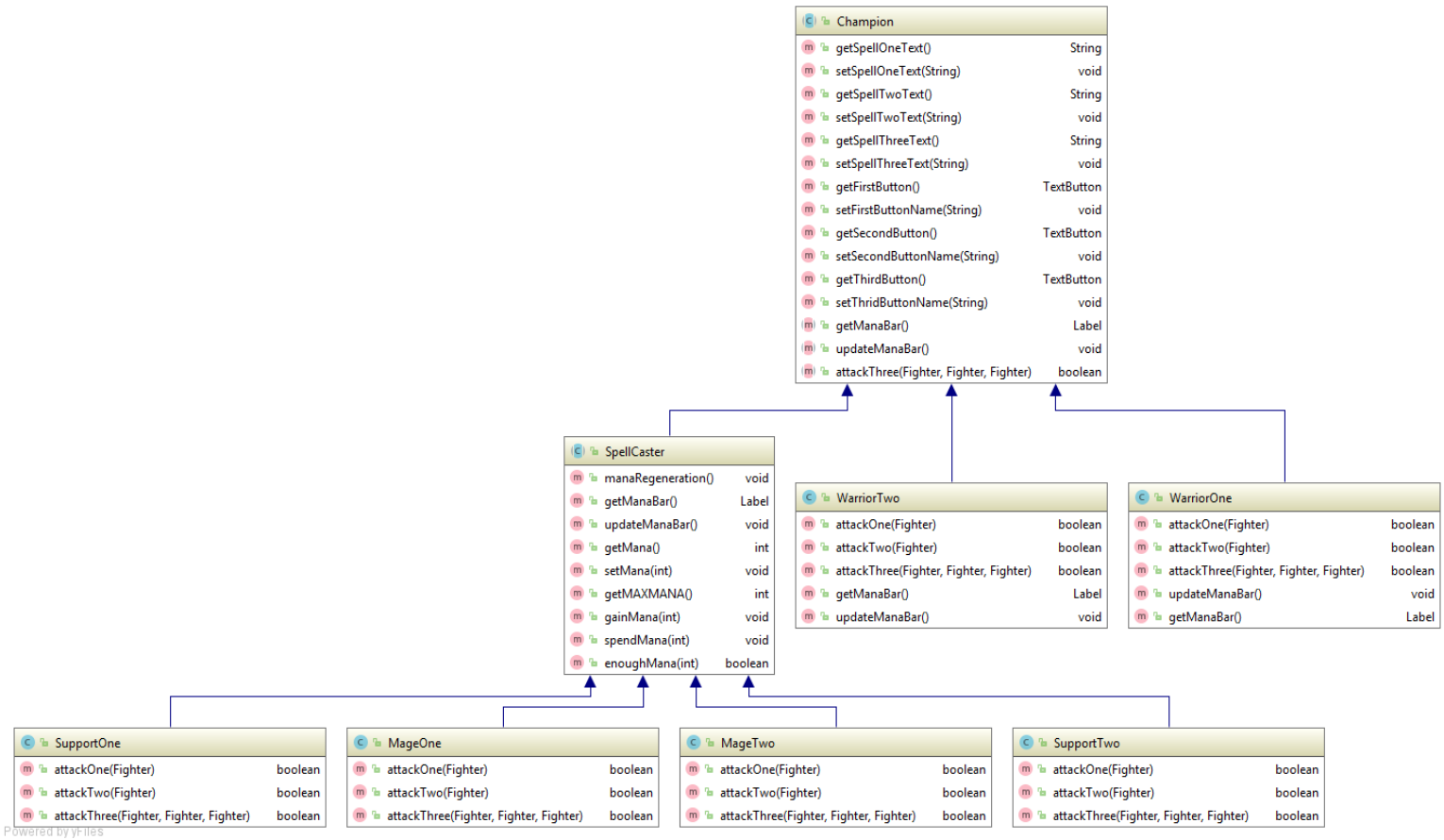


Figure 51: This picture shows the champions, their methods and their herarchy.

Because of the hierarchy and the similarity of methods used for the classes, only one entity of the Fighting actors will be described. The most complex classes are the ones extending SpellCasters, so for the description we will take the class; MageOne.

The MageOne class, extends the abstract classes; Spellcaster ← Champion ← Fighter. By using this hierarchy, we get all the functionality that has been implemented in those classes, and we do not have to rewrite it for every fighter that we want to implement in our game.

In the constructor of the class MageOne is set the name of the Actor, the HP and the maxHP. Additionally, is added the information for the buttons, which are shown on the FightingScreen. After that the animation sprites are rendered the same way as for the ExploringActors.

The mana related methods for every Actor extending SpellCaster, are declared in the abstract class SpellCaster. We have a method for checking if the SpellCaster has enough mana to execute the action. The method is returning a Boolean, so if the method returns true, the action is executed, but if not a "Not enough mana" sound is played.

The attack methods are implemented from the Fighter class. As described before, the attack methods return a Boolean. For the attack to be executed there should be a target of the attack. When implemented the attack method, conditions are added to check if the target is right. Warrior and Mage can target only enemies, Support can target both enemies and characters depending if the attack is healing or hit. In the class MageOne, (line 46 and line 61) we are checking if the target is an instance of EnemyFighters. If true (and if MageOne has enough mana) then the action is executed, if false a sound "You cannot click there" is played.

```
43      @Override
44      public boolean attackOne(Fighter fighter)
45      {
46          if (fighter instanceof EnemyFighters){
47              fighter.setHP(fighter.getHP()- MathUtils.random(10,18));
48              this.gainMana(FighterBalanceVariables.MAGEONEATTACKONEGAIN);
49              return true;}
50          cantclick.play();
51          return false;
52
53      }
54      /**
55       * Attack 50 and decrease mana 55
56       * @param fighter
57       */
58      @Override
59      public boolean attackTwo(Fighter fighter)
60      {
61          if (fighter instanceof EnemyFighters){
62              if (this.enoughMana(FighterBalanceVariables.MAGEONEATTACKTWOCOST)){
63                  fighter.setHP(fighter.getHP()-50);
64                  this.spendMana(FighterBalanceVariables.MAGEONEATTACKTWOCOST);
65                  return true;
66              }
67              else{
68                  return false;
69              }
70          }
71          cantclick.play();
72          return false;
```

Figure 52:This is the code for attack method.

All other classes part of the subpackage FightingActors are a copy or a simplified version of the described class MageOne, so further explanation of those will be omitted.

Screen package

The Package contains all the screens from the gameplay. Part of the package contains simple screens where the only function is to link different screen or to create a pause screen such as LoadingScreen or PauseScreen. The LoadingScreen is a fake screen, as nothing is loaded while it is shown. It is to give the player a chance to let go of the movement keys when moving between each map.

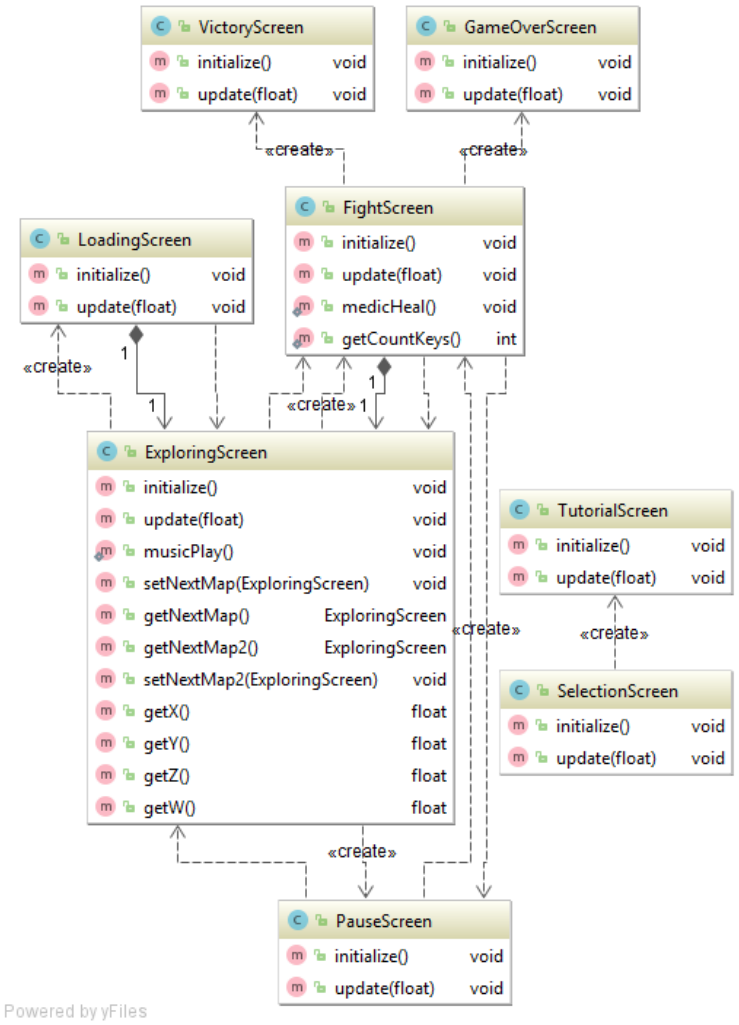


Figure 53: This is a picture of all the screens in the game.

On the other hand, we have the two classes, FightingScreen and ExploringScreen, where the user will spend the majority time in our game. The ExploringScreen creates the maze maps where the user has to navigate between them, and also checks the interactions between all of the actors that belong to the exploring map, and create a communication between different ExploringScreens.

The FightingScreen class is the most complex class that we have in our project. It handles the fight from the player champions and the enemies he collides with within the exploring map. Furthermore, it creates what we call a 'trivia question' during the fight. The player will have to answer trivia questions during the fight, and if the player answers correctly, the player is rewarded, but if the player answers wrong, the player is punished.
The trivia questions have a chance to be triggered each turn of each fighter in the FightingScreen class, and displays a question with a multiple-choice answer on the top of the screen.

Character selection

Before the player gets to start the game, three heroes need to be selected. The game has three different types of heroes: Fighter, Mage and Support. The player needs to select one of the two heroes of each type and move them to the selection area. Before the game can start, each hero slot must be filled.
This screen supports drag and drop functionality from the framework.
The different slots for the hero types are set to only accept their own type. The selection slots are objects of the SelectionArea class, each of the three areas has a hero type and a hero number. The heroes are objects of the CharacterCard class, have the same type as the SelectionArea. If the type of the card matches the type of the slot that the card is above, the card moves onto the slot and sets the slots hero number to match the cards hero number. If not, the card will be moved back to where it was picked up.

```
44      public void onDrop()
45      {
46          if ( hasDropTarget() )
47          {
48              SelectionArea sa = (SelectionArea) getDropTarget();
49
50              if (getFighterType() == sa.getHeroType()) {
51                  moveToActor(sa);
52                  setSelectionArea(sa);
53                  sa.setTargetable(false);
54                  sa.setHeroNumber(fighterNumber);
55              } else {
56                  moveToStart();
57              }
58          }
59      }
```

Figure 54: This is the code for letting go of a character card, checks if there is a slot below it and if the slot matches the card.

When the start game button is pressed, the game checks if all slots are filled by reading their hero number variable, and if it isn't, the game won't start.

FightingScreen:

The class FightingScreen has to deliver two major functionalities.

1. Implement a turn-based game with our champions and the enemies that we encounter during the exploring map.
2. Implement a question-answer system to check the knowledge of the player and reward or punish the player depending the answers given.

Like all the screens we have implemented before, they extend a class BaseScreen, that extends at the same time the class Screen from LIBGDX and implements the interface InputProcessor.

The class has two major methods that are implemented from the abstract class BaseScreen; Initialize() and Update(). Initialize() is creating all the objects and structure that is needed to be able to run the class and give us the functionality needed, while Update() checks each frame if there are any changes about any object from our class or any input from the user, update the game logic and finally render the changes into the screen.

Initialize() method:

First thing that is done in the Initialize() is loading a background picture to the screen, and creating a background music within a loop and with a volume that is stored at MortenKombat class as a static variable.

Next, we create all the labels needed for the interface such as the label that shows the turn in the screen, and the label that shows the ability tooltip in the screen to notify exactly what is each ability doing. We continue initializing all the buttons needed for our trivia question. Until here, everything within the code is self-explanatory and we are using the API from the framework and LIBGDX to generate all this.

Then we initialize the fighters in our screen. The class is able to handle up to 6 different fighters; 3 Champion's that will be the one that the player controls, and up to 3 enemyFighter's. The code reads a static variable from MortenKombat to know what type of Champions it has to create, while to create the enemyFighters we read the variable amountOfEnemies in the FightingScreen to know what kind of enemy we fight and how many they are.

Note that we are giving a value to the static variable amountOfEnemies before calling the constructor. This needs to be done as we encountered the issue of having the method Update() running in our class FightingScreen before the constructor is even called.

```
314                 FightScreen.amountOfEnemies = 2;
315                 MortenCombat.setActiveScreen(new FightScreen( prev: this));
```

Figure 55: This code fixes the previous code. Code from ExploringMap

Once we know what actors we have in our FightingScreen we initialize the data structure that help us create the turns in the game. We use a list to keep the data of what actors that are still alive in our fight and a stack that represents the turn. Using a stack simplifies us the idea of having a turn; all actors are filled into a stack, then shuffled and we give the privilege to make an ability to the actor who is in the peek. After using an ability we pop the actor from the stack and we continue until the turn is over,that means that the stack is empty. To be able to refill our stack and generate a new turn we can use our list of alive actors as it contains all the actors that are still alive in our fight.

The method Initialize() and Update() are running in a thread[62], and both of them use the data structure explained before: The list aliveFighers and our stack fightingTurn. If we use non-safe thread collections, we might encounter compilation during the game execution. To fix this we will use a CopyOnWriteArrayList to implement our list and ArrayDeque to implement our stack.
Now we create the event listeners for the Champion's ability buttons, Fighter's and the trivia buttons. We will explain how the abilities and the fighters interact.

[62] https://github.com/libgdx/libgdx/wiki/Threading

```
263         //creating listeners for buttons, activate the spell selector.
264         for (Champion c : champions){
265             c.getFirstButton().addListener(
266                     (Event e) ->
267                     {//making sure that only the fighter with the turn can listen to clicks.
268                         if (c != safeFighterStackPeek()) {
269                             return false;
270                         }
271                         tooltipText.setText(c.getSpellOneText());
272                         if ( !(e instanceof InputEvent) ){
273                             return false;
274                         }
275                         if ( !((InputEvent)e).getType().equals(InputEvent.Type.touchDown) ) {
276                             return false;
277                         }
278                         firstAttack = true;
279                         secondAttack = false;
280                         thirdAttack = false;
281                         abilityUser = c;
282                         activateSpellMouse ();
283                         return true;
284                     }
285             );
```

Figure 56: This is the code for the buttons for each champion in the FightingScreen

This scope of code creates an event listener for the object 'FirstButton' for all the champions that are in the list 'champions'. In the listener the first condition is to make sure that the listener pointing to the actor champion is on top of the peek, and it is his turn to activate an ability.

Also, when the player mouseover the button we would like to have a small label that shows a full description of the ability that is represented by the button. The next two conditions at the listener make sure that we are handling an InputEvent and that the event is been activated by a mouse click.

If all these conditions are met, we create and activate a "buffer" where we store what ability has been used, what champions is activating that ability and finally we change the mouse cursor to indicate that the player as an ability that needs a target.

The ability buffer can be cleaned in two ways. One is that we are listening to a right click event from the mouse (implemented in the update() method). If that is triggered, we remove the buffer of having an ability that needs a target and change the mouse cursor to the default one.

```
718         //going out of the spell buffer if we right click.
719         if (Gdx.input.isButtonPressed(Input.Buttons.RIGHT)){
720             activateDefaultMouse ();
721             firstAttack = secondAttack = thirdAttack = false;
722         }
```

Figure 57: This is code for when canceling an attack by right clicking the mouse.

The second option to clean the buffer is actually finishing the ability by finding a target. To do so, we create an event listener in the initialize() for each of our fighters.

```
323         //creating the listeners of all the actors. Will be the target of ours spells.
324         for (Fighter e : fightingTurn){
325             e.addListener(
326                     (Event o) ->
327                     {
328                         if ( !(o instanceof InputEvent) )
329                             return false;
330                         if ( !((InputEvent)o).getType().equals(InputEvent.Type.touchDown) ){
331                             return false;
332                         }
333                         //if you dont put this line, right click would also trigger the spells. and we only
334                         //want to trigger the spells with left mouse click.
335                         if (Gdx.input.isButtonPressed(Input.Buttons.RIGHT))
336                             return false;
337                         if (!aliveFighters.contains(o.getTarget())){
338                             //check if the target is dead before accepting the event
339                             cantclick.play();
340                             return false;
341                         }
342                         if (firstAttack){
343                             //if the first condition is false, the second condition is never check and never executed
344                             if (!isTriviaAttack && abilityUser.attackOne((Fighter)o.getTarget ()))
345                                 abilitySuccess(abilityUser);
346                             if (isTriviaAttack  && abilityUser.attackOne((Fighter)o.getTarget ())){
347                                 abilityUser.attackOne((Fighter)o.getTarget ());
348                                 abilitySuccess(abilityUser);
349                             }
```

Figure 58: This is the code for target selection.

When creating the event listeners for our fighters we check if the fighter is alive, and we also check that the right click event is not processed by the event when clicking on the fighters (The right click is been used to clean the ability buffer in the update() method).

Once those conditions have been fulfilled, we check what ability (each champion has three abilities that are linked to a different button) has been triggered by the event listeners related to our buttons. In our case, we explain what would happen if the player would have triggered the first ability from championOne.

We have a boolean variable, *isTriviaAttack*, that contains the value 'true' if the ability that is triggered has been preceded by a trivia question (how this value is changed will be explained in the next section when explaining the functions from update() method). Also, the abilities of all our champions return a boolean, true if they can use the ability (that means that we have the right target, and in case of spellcasters we have enough mana for it) or false if they cannot use the ability.

In our example, we would execute a method with championOne.attackOne((Fighter) o.getTarget()) where *o.getTarget()* refers to the target of our event listener.

Having in mind all those conditions, the line 344 returns true if the ability of the champion has the right target in our example with the championOne. If that is the case the method *abilitySuccess()* is called; by doing that it cleans the ability buffer and removes the fighter from the *fightingTurn* stack.

```
780     private void abilitySuccess(Champion user){
781         attacker = user;
782         abilityNotSuccess();
783     }
784
785     /**...*/
789     private void abilityNotSuccess(){
790         startTime = System.currentTimeMillis();
791         activateDefaultMouse();
792         tooltipText.setVisible(false);
793         tooltipText.setText("");
794         safeFighterStackPop();
795         if (safeFighterStackPeek() instanceof Champion)
796             tooltipText.setVisible(true);
797         isFightTurnOver();
798         firstAttack = secondAttack = thirdAttack = false;
799         triviaHasCheck = -1;
800     }
```

Figure 59: This code checks if the ability has been used and resets for the next turn.

The last step that is needed after creating all our event listener is to create a table that helps us organize all the position for all our objects that need to be displayed on the screen.

Update() method.

The update method checks for conditions each frame of the game. The main requirements that must implement are:

1. If the champion has the turn, check if we pop the trivia attack.
2. If the enemyFighter has the turn, check if it creates a trivia or normal attack to a random alive champion.
3. Update fighter labels, change animation when they attack or die, check if any fighter dies and remove them from the fightingTurn stack and aliveFighters list.
4. Check if any buttons has been pressed during the trivia attack, and return a boolean so Champions and enemyFighters know how to interact.

For the first requirement there is a 30% chance to make our champion turn be a trivia question attack. To check this, we need a variable with three states:

1. Has the trivia question chance been checked? (TRIVIATOBECHECKED)
2. The trivia triggered and is waiting for the user to answer (TRIVIAWAITINGFORANSWER)
3. The trivia has triggered and the user has answer (TRIVIAHASBEENANSWERED)

```
591         if (triviaHasCheck <= TRIVIAWAITINGFORANSWER && safeFighterStackPeek() instanceof Champion ){
592             if (MathUtils.random(0,9)< TRIVIACHANCE && triviaHasCheck == TRIVIATOBECHECK){//if true its a trivia question
593                 triviaHasCheck = TRIVIAWAITINGFORANSWER ;
594                 showTrivia();
595                 isTriviaAttack = true;
596             }else if (triviaHasCheck == TRIVIATOBECHECK){//it is not a trivia question
597                 triviaHasCheck = TRIVIAWAITINGFORANSWER;
598                 isTriviaAttack = false;
599             }
600             if (triviaHasCheck == TRIVIAWAITINGFORANSWER && (isAnswerButton1Pushed || isAnswerButton2Pushed ||
601                     isAnswerButton3Pushed || isAnswerButton4Pushed) ){
602                 stateOfAnswer(isAnswerButton1Pushed,isAnswerButton2Pushed,isAnswerButton3Pushed,isAnswerButton4Pushed);
603                 triviaHasCheck = TRIVIAHASBEENANSWER;
604                 if (!isCorrectAnswer)
605                     abilityNotSuccess();
606             }
607         }
```

Figure 60: The code for the trivia attack.

If the peak of our fightingTurn is an instance of a champion and the trivia has not been answered, we trigger this code above. What is happening is that we use the three states described before. So the first time we enter we are in the state triviatobechecked it will perform the 30% chance to create a trivia attack, calling the method showTrivia() and changing the boolean isTriviaAttack to true in the case that we have a trivia question. Then the code is waiting until the player pressed an answer button. When that happens, it checks if the player has answered correctly, triggering the *abilityNotSuccess* in case that he answers wrong, or letting the user make an ability if the answer is right, and since *isTriviaAttack* it uses the ability twice before poping the fighter from the *fightingTurn*.

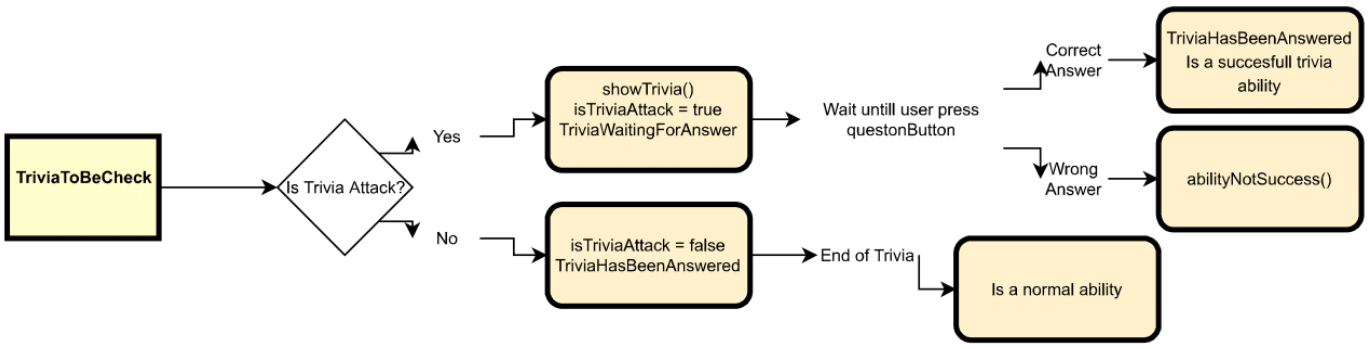


Figure 61: This table shows what happens during the trivia attack.

Implementing the attacks from the *enemyFighter* is really similar to the *champions* ability implementation. Inside the *Update()* method we check if the enemy has the turn by peeking the *fightingTurn* and checking if the object is an instance of an *EnemyFighter*.
Afterwards, we create a condition to make the enemies wait for between 3,5 and 4,5 seconds by comparing the System current time. Once that time has passed, we check if we need to trigger the trivia

question for our attack. This check is really similar to the one explained for the champion trivia question in figure 60.

```
648             if (safeFighterStackPeek() instanceof EnemyFighters){
649                 currentTime = System.currentTimeMillis();
650                 if ((currentTime - startTime) > enemyThinking){
651                     boolean isNonQuestionAttack = false;
652                     //30% chance to activate the question for enemies.
653                     if (triviaMustBeShown == TRIVIATOBECHECK && MathUtils.random(0,9)< TRIVIACHANCE ){
654                         triviaMustBeShown = TRIVIAHASBEENANSWER;
655                         showTrivia();
656                     }else if (triviaMustBeShown == TRIVIATOBECHECK) {
657                         triviaMustBeShown = TRIVIAWAITINGFORANSWER;
658                         isNonQuestionAttack = true;
659                     }
660                     stateOfAnswer(isAnswerButton1Pushed,isAnswerButton2Pushed,isAnswerButton3Pushed,isAnswerButton4Pushed);
661                     boolean hasBeenPressed = (isAnswerButton1Pushed || isAnswerButton2Pushed ||
662                             isAnswerButton3Pushed || isAnswerButton4Pushed);
663                     int championTarget = MathUtils.random(0,100);
664                     //60% chance to hit first target 25% second chance 15% last target
665                     //if target is dead, goes for the next one
666                     if (championTarget > 40 && aliveFighters.contains ( championOne ) &&
667                             (hasBeenPressed || isNonQuestionAttack) ){ //when a enemy choose to attack first target
668                         startTime = System.currentTimeMillis() ;
669                         if (isNonQuestionAttack){
670                             theEnemyAttacks ( (EnemyFighters)safeFighterStackPeek(), championOne );
671                             attacker = safeFighterStackPeek();
672                         }
673                         else if (!isCorrectAnswer){
674                             theEnemyAttacks ( (EnemyFighters)safeFighterStackPeek(), championOne );
675                             theEnemyAttacks ( (EnemyFighters)safeFighterStackPeek(), championOne );
676                             attacker = safeFighterStackPeek();
677                             startTime += 4000; //increasing animation time there is a 4s delay with the answer on screen
678                         }
679                         safeFighterStackPop();
680                         isFightTurnOver();
681                         triviaMustBeShown  = TRIVIATOBECHECK;
```

Figure 60: Scope of the code that deals the attacks from EnemyFighters

The code now is waiting until the player press a trivia answer button. When one of the trivia buttons has been pressed it changes the value of isAnswerButton'N'Pushed (where 'N' is the number of the button) to 'true' and the method stateOfAnswer checks if that answer was the correct one. If the answer given by the player was correct, the stateOfAnswer sets the variable isCorrectAnswer to 'true', and it is set to 'false' when the player gives a wrong answer to the trivia question.

Once we know if we had a non-trivia question, a correctly answered trivia question or wrong answered trivia question we randomize if we should attack championOne, championTwo or championThree. When that is decided, the enemyFighter deals an attack depending on the trivia question to a random champion and resets all the triviaMustBeShown value to the default one, pops the enemyFighter from the stack and checks if the turn is over with the method isFightingTurnOver().

```
1165    private void isFightTurnOver(){
1166        if (fightingTurn.isEmpty ()){
1167            fightingTurn.addAll(aliveFighters);
1168            for (Fighter sC : fightingTurn){ //regenerates mana at the end of each turn.
1169                if (sC instanceof SpellCaster)
1170                    ((SpellCaster) sC).manaRegeneration();
1171            }
1172            shuffleDeque(fightingTurn);
1173            turn ++;
1174            turnLabel.setText("Turn: "+turn);
1175        }
1176    }
```

Figure 61: Here the code recrates the turn for all alive actors. It updates the mana and increases the turn counter.

To maintain all our labels with up to date, we run a for each loop with all the elements of the list AliveFighters inside our Update() method. Inside this loop, we handle the following functionalities:

- Update values of our HP label and Mana label.
- Update the color of the name label to indicate whose fighter is the turn.
- Check if all champions or all enemies are dead so we can exit the FightingScreen.
- Check if any fighter gets their HP to zero. If that is the case, start dead animation and remove the fighter from the aliveFighters and FightingTurn.
- Check if there has been a difference of fighters HP value. If a difference was detected, the fighter has been "hit" by another fighter, and we make their HP label blink 3 times to indicate that.

The last major functionality that we are missing in our *Update()* method is to check if the player has pressed the right answer when the trivia is been activated for the champions. To be able to do that, we check if a button has been pressed and analyze if the given answer is the correct one with *caseOfAnswerButton'N'* method.

```
588        caseOfAnswerButton1(isAnswerButton1Pushed);
589        caseOfAnswerButton2(isAnswerButton2Pushed);
590        caseOfAnswerButton3(isAnswerButton3Pushed);
591        caseOfAnswerButton4(isAnswerButton4Pushed);
```

Figure 62: This is the answers that can be pressed during a trivia attack.

ExploringScreen

The game requires a world where different maps are connected. This function is generated by creating instances of the class ExploringScreen which they are interconnected with each other making a binary tree of maps.

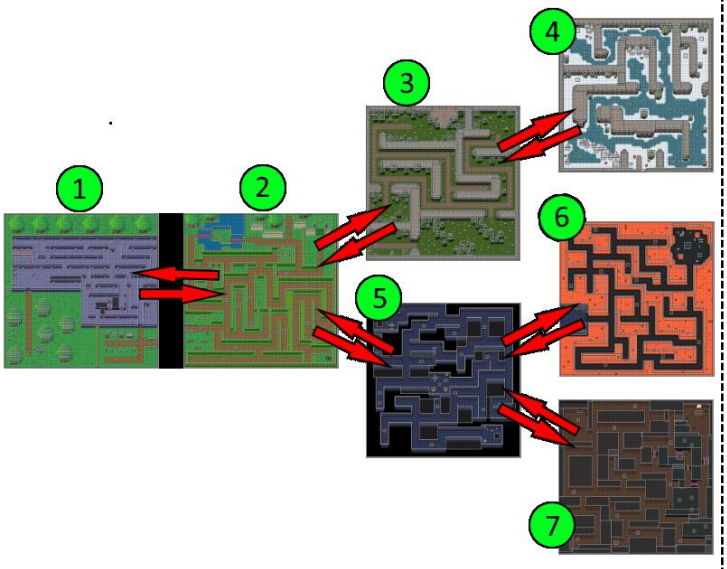


Figure 63: The picture shows the connections between the maps and the order they are created from the MapLayout enum.

To be able to create this data structure, we need to have a variable on each instances of the maps where they point to the previous map that they proceed, the next map that you could go from the map, and second next map that you could go in case that it exists.

```
24  public class ExploringScreen extends BaseScreen {
25      private ExploringScreen previousMap;
26      private ExploringScreen nextMap = null;
27      private ExploringScreen nextMap2 = null;
```

Figure 64:This is for going back to the previous maps.

All next map and previous map variables are set to null, until the second constructor changes them during map creation. Because the next map variable of a map can't be set before the next map is created, the constructor has to go back, when it is creating the next map and sets the previous map's next map variable to the current map.

```
59  @   public ExploringScreen(ExploringScreen previousMap) {
60          if (previousMap.getNextMap() != null)
61              previousMap.setNextMap2(this);
62          else previousMap.setNextMap(this);
63          this.previousMap = previousMap;
64      }
```

Figure 65: This is the second constructor. It is used for all maps except the first map.

The next two milestones from the class are: Initialize all the actors that will be present at the screen and check how all the actors interact with each other. The first one is implemented in the method Initialized() belonging to Exploring screen, while the second one is in the method Update().

In the initialize() method begins with reading a Tiled file (files with extension .tmx), and creating all the objects that we are reading from the tiledmap file. To be able to do this we have created a method *createMapObjects (tilemap, mapAttributes)*. This method goes through all the Rectangles that we have created in our tiledmap file. It searches for rectangles that have the content of our custom property with the title "name" and compares it with the argument *mapAttributes* from our method.

Figure 66: This shows the properties for a sign in the maps

In this case for example, the method creates a new Sign object with the position XY, and then assigns the text to that object from the property's "message" from the tiledmap file.

```
103         createMapObjects(tma, mapattributes: "Sign");
```

```
229 @   private void createMapObjects(TilemapActor tma, String mapattributes) {
230         for (MapObject obj : tma.getRectangleList(mapattributes)) {
231             MapProperties props = obj.getProperties();
232             switch (mapattributes) {
233                 case "Exit":
234                     new Exit((float) props.get("x"), (float) props.get("y"), mainStage);
235                     break;
236                 case "Sign":
237                     Sign s = new Sign( (float)props.get("x"), (float)props.get("y"), mainStage );
238                     s.setText( (String)props.get("message") );
239                     break;
```

Figure 67: This creates the map objects on the maps.

After creating all the objects needed in our code the only missing steps, is to create a dialogBox that we will use during the screen to display the text of our Sign object, and changes some properties of our hero object in case that we are in a wind, ice or dark map.

The update() method makes sure that all the changes that happens with our object are being updated and shown in the screen, and is checking what should each of the object do on each loop of the update.

The wind map effect persists during the exploration of the map, and is being activated periodically. We can get this period effect by counting how many update()'s have been run. The game is calling the update() method for each of frame of the game, and the frames are capped at 60 frames/s.

```
189         if (currentMapEffect.equals("wind")) {
190             windTimer++;
191             if (windTimer % FRAMESPRBLOW == 0) {
192                 windBlow();
193             }
194         }
```

Figure 68: This creates the Wind effect on map 3

The player also needs to be able to move the hero object during the map. To be able to do so we check in the update if arrows or WASD buttons at the keyboard are being pressed and moves the hero towards the direction of the keycap pressed.

Last, we program how each of the object interact in our screen. Some of our object interact with other object and also this kind of check need to be implemented. To do all this we have created a method called *actorObjectInteraction(String className).*

```
176
280     private void actorObjectInteraction(String className){
281         for (BaseActor a : BaseActor.getList(mainStage, className)) {
282
283             switch (className) {
284                 case "core.actors.exploringactors.Bat":
285                     for (BaseActor s : BaseActor.getList(mainStage, className: "core.actors.exploringactors.Solid")) {
286                         if (a.overlaps(s)) {
287                             a.preventOverlap(s);
288                             a.setMotionAngle(MathUtils.random(0, 360));
289                         }
290                     }
291                     if (a.overlaps(hero)) {
292
293                         musicStop();
294                         FightScreen.amountOfEnemies = 1;
295                         MortenCombat.setActiveScreen(new FightScreen( prev: this));
296                         a.remove();
297                     }
298                     break;
```

Figure 69: This is the code for how objects interacts with each other and our hero.

This method checks along all the actors that has been created in our screen.
In this case, for example, any actor that has been created from the Bat class is being checked if it collides with any actor that belongs to class Solid. If that happens, it prevents the overlap of those two objects and changes the direction of the Bat object to move in another direction generated randomly. Also, it checks if the Bat object collides with the hero object and in the case that it collided with the hero it makes the changes so we can change the screen of our MortenKombat game class to the FightScreen, and remove the Bat object from the ExporingScreen.

6. Future perspective

This chapter is for all the functionalities that we wanted to have in our game, but did not have the time to implement fully.

Design

We are using a lot of free art and music, we should buy or get some art made for our game; the background in the fights could change depending on where the fight takes place.

That way we could make the map creation of new maps easy, because we can give the tilemaps out, and everything made with them should match with the rest of the game.

Before the release we should get the animations and sprites made more uniform and make sure they match the theme. Because most of the animations differ in size, it looks bad in fights. We also want to have more animations for attacking and being attacked. Some of the attacks need some particle effects, this will make them more visible to the player and show more clearly what happens with each attack.

Implementation

In the original idea of our game, we wanted to have items to be found in the maze to help in the fights. The items could be boosts to a specific attack, or enable a new attack.
The items would be found in chests around in the maze which would also contain potions to restore health and mana in fights.

We also wanted to have a level system where after every fight you would get experience points, and the heroes could level up and gain more health and damage. This would help players who fight a lot and get people who skips many fights, they would have a harder time later on in the maze.

Some of the attacks are just different numbers for the damage dealt, we want to make some more interesting attacks like status effects and damage over time effects. This is especially needed for the bosses which are not very interesting compared to the normal enemies. It could also be used to make the normal enemies more interesting, and some attacks could even have a higher chance to give a question and effects like that.

Another functionality that we should have before we release it is saving and loading. If we want other people to make their own maps, we should have a way to randomize the connections between the maps. And with that we should have some interface that shows how the different maps are connected, otherwise it would very hard for the player to navigate, if it was random every time a new game was started.

The user can make their own questions locally, but we want to have some sets of questions on a server which the game could load remotely.
Our plan was to get our game used by schools, and to get the students more motivated we would make a point system for each set of questions. The leaderboards would be for each class, so that the students

could compete with each other for the best high score. We would make a website to see the leaderboards, more details on that in the deployment.

Deployment

The main goal of the whole project is to have an educational game, and we aim at making the educational purpose easier in an age were video games are so popular. To make sure that we can reach as many users, and have the biggest impact in the society we will make our game free.

We plan on making a website for our game, where the game can be introduced, giving some samples of the gameplay, and reviews of past institution that has use our game. Any user will be able to download a compiled version of the game, and they should be able to run it on any computer as long as Java is installed in their computers.

We plan to implement a forum at the website for teachers, were they can exchange question answer files, and let the community of teachers/educators develop complex and better targeted question answer that the ones we could provide.

Last in our website we will give the link to the GitHub of the project with the source code. This will improve the quality of the game by having new collaborators, allowing them to commit new changes to the existing code, updating and maintaining the code. The idea is to be able to register MortenKombat under an open source license, giving anyone the right to distribute, modify the code that has been developed.

Last, we will contact educational institutions and propose them the game so they can implement them in public schools. We will have this approach after launching the game from our website, and making sure that the game has been tested.

7. Conclusion

Developing Morten Kombat was totally new experience for all the members and we went through challenges from the very beginning.

After doing a game design and having the project requirement, we confronted our first challenge. After deciding that the project is going to be developed with LibGDX, we had 2 months, not only to learn Java, but also to be familiar with the functionalities and how we should implement the libraries from LibGDX.

To accomplish this, we learned how to read the Javadoc from Oracle and the home wiki page from LibGDX. Understanding and learning from technical documentation was vital to be able to develop the project in this time frame, as time was also a major issue for the project.

The time frame for the entire project was two months and a half, and after making our game design we realized that we might have a bigger project than expected. We found it crucial to have a good project management to ensure that the game could be split in smaller pieces and implement some of the functionalities in parallel. Doing so, we would optimize the working time of the members, as they did not have to be waiting for another member to finish a part of the code to start their task. This has been a key factor, and we would not be possible to realize MortenKombat if the implementation was not split in different stages.

The planification of the project was not enough to bring all the requirement that we proposed in the game design. We miss some of the requirements, as the last stages of the implementation of the code were extended. Those requirements were presented in the future perspective.

During the implementation we found it fundamental to have a written game design where all the team members could go and check the game requirement. It gave a clear view to each group members what they were implementing, making it easy to connect the work between members.

In conclusion, our problem formulation was:

Main Problem Formulation

-How can we develop a Java educational quiz game?

Sub - Questions

- How can we make our game flexible in terms of making it a learning tool capable of reaching a large audience with different educational levels?

-How can we make our quiz game more entertaining for the end user?

To answer the problem formulation, we came up with an educational quiz game, called MortenKombat, which is based in a maze/labyrinth. The educational aspect of the game is implemented in the form of randomized questions appearing during the battles throughout the maze.

The game was developed within LibGDX framework. Java Game Development with LibGDX book has been a milestone in the project and gave the required tools to understand the LibGDX library, design the game, and implement the project.

The game allows loading your own question into the game. This has been the essence of the developed product MortenKombat. This feature increases the replay-ability and makes the game more targetable to different kind of students. Without this option the game could only have a collection of preset questions making the educational part inefficient and less entertaining.

We wanted to step up from the market, generating an educational game that would bring joy and replay-ability to the players. We mixed a successful game genre, Role Game Playing, with educational games, trying to increase the entertainment of the game. This aspect was quite important for the project and we wanted to ensure that we got our preset goal.

MortenKombat had a game test with users, that was not involved in the game development. This was a process to demonstrate if we achieved our goals for educational and entertainment. Simultaneously, we collected information on common problems that were spotted from the testers. These problems were solved in an optimized version of MortenKombat.

In conclusion, MortenKombat brings a different perspective of an educational game while keeping the interest of the player. We created a game that provides adjustable questions to target the audience of the game.

8. References

INTRODUCTION REFERENCES

- https://educator.mangahigh.com/2017/03/08/gbl-growthmindset-edutainment/ (Article on the Psychology behind game-based learning. Visited 12-10-2018)
- https://www.frontiersin.org/articles/10.3389/fpsyg.2015.01056/full (Article on educational games and its effects on the brain. Visited 12-10-2018)
- https://www.darkestdungeon.com/ (A rogue-like game. Visited 12-10-2018)
- https://ucha.se/ (An online Bulgarian educational game. Visited 12-10-2018)
- https://politiken.dk/kultur/art6760839/Falder-du-ogs%C3%A5-i-s%C3%B8vn-til-old%C3%A6vl-Avanceret-spil-g%C3%B8r-oldtidens-Gr%C3%A6kenland-sexet (An article for games to help learn the very dry material. Visited 12-10-2018)

METHODOLOGY REFERENCES

- Waterfall model. (2018, October 15). Retrieved October 22, 2018, from https://en.wikipedia.org/wiki/Waterfall_model
- The Theory of Game-Based Learning. (2017, July 7). Retrieved October 22, 2018, from https://www.game-learn.com/the-theory-of-game-based-learning/
- Unified Modeling Language. (2018, October 4). Retrieved October 18, 2018, from https://en.wikipedia.org/wiki/Unified_Modeling_Language
- Class diagram. (2018, September 18). Retrieved October 18, 2018, from https://en.wikipedia.org/wiki/Class_diagram
- Reverse engineering. (2018, October 2). Retrieved October 18, 2018, from https://en.wikipedia.org/wiki/Reverse_engineering
- Retrieved October 22, 2018, from https://homes.cs.washington.edu/~mernst/advice/version-control.html
- Managing Stakeholders. (2017, May 16). Retrieved December 13, 2018, from http://scitechconnect.elsevier.com/3-simple-steps-managing-stakeholders/
- Research Onion. (Saunders et al., 2007) Retrieved December 12, 2018 from: https://www.researchgate.net/figure/The-research-onion-Source-Saunders-et-al-2007_fig1_303674706
- Research methods for business students – eClass (February 2, 2008) Retrieved December 14, 2018 from: .https://eclass.teicrete.gr/modules/document/file.php/DLH105/Research%20Methods%20for%20Business%20Students%2C%205th%20Edition.pdf.
- Research Onion. Retrieved December 14, 2018 from: https://onion.derby.ac.uk/.

STATE OF THE ART REFERENCES

- Article on Gaming. May 31, 2015. Retrieved October 19, 2018 from https://www.nielsen.com/us/en/insights/news/2016/gaming-gone-global-keeping-tabs-on-worldwide-trends.html
- LaShera McElhany, Ph.D. (May 17, 2016) The Hidden Value of Gaming in Education Retrieved October 17, 2018 from https://www.sagu.edu/thoughthub/the-hidden-value-of-gaming-in-education
- Paras, & Brad. (2005, June 2). Game, Motivation, and Effective Learning: An Integrated Model for Educational Game Design. Retrieved October 17, 2018, from http://summit.sfu.ca/item/281

ANALYSIS REFERENCES

Game Market Analysis

- Pdf on educational games, Per Backlund (Interaction Lab - University of Skövde, Sweden) & Maurice Hendrix (Serious Games Institute - Coventry University, UK) September 2013, Retrieved December 14, 2018 from https://www.researchgate.net/publication/261076302_Educational_Games_-_Are_They_Worth_the_Efforta_Literature_Survey_of_the_Effectiveness_of_Serious_Games
- Article based on Metaari's Global Game-Based Learning Market report. Rebekah Valentine (August 8, 2018) Retrieved December 14, 2018 from https://www.gamesindustry.biz/articles/2018-08-08-metaari-game-based-learning-market-will-reach-usd17-billion-by-2023
- Article on the popularity of educational games in the us. Symeon Retalis (May 23, 2016) Retrieved December 14, 2018 from https://blog.kinems.com/educational-games-are-growing-in-popularity-with-us-teachers-and-students/
- Article on the Game Awards 2018. Paul Tassi (December 7, 2018) Retrieved December 14, 2018 from https://www.forbes.com/sites/insertcoin/2018/12/07/heres-the-full-list-of-winners-from-the-game-awards-2018-including-goty/#f3c0dea1cc4a
- Research Paper on Educational Contribution of RPG Video Games. Martin Kratochvíl (March 29, 2014) Retrieved December 12, 2018 from https://dspace.cuni.cz/bitstream/handle/20.500.11956/61652/BPTX_2013_1_11410_0_352950_0_145703.pdf?sequence=1
- Public vote on best game genre. (2018) Retrieved December 13, 2018 from https://www.thetoptens.com/video-game-genres/
- Chart for the most sold games genres in 2017. (2018) Retrieved December 12, 2018 from https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/
- Article on game industry. Kellie Ell (July 18, 2018) Retrieved December 13, 2018 from https://www.cnbc.com/2018/07/18/video-game-industry-is-booming-with-continued-revenue.html
- Article on opening sales for RDR2. Shubhankar Parijat (October 30, 2018) Retrieved December 12, 2018 from https://gamingbolt.com/red-dead-redemption-2-grosses-over-700-million-in-biggest-entertainment-opening-weekend-of-all-time

- Article on Market Segmentation. Ana Gotter (August 13, 2018) Retrieved December 13, 2018 from https://www.disruptiveadvertising.com/marketing/market-segmentation/
- Article on the 4 types of Market segmentation. Hitesh Bhasin (May 25, 2018) Retrieved December 13, 2018 from https://www.marketing91.com/4-types-market-segmentation-segment/
- Article on the retro gaming industry. Damien McFerran (February 2, 2018) Retrieved December 13 from https://www.eurogamer.net/articles/2018-02-09-the-retro-gaming-industry-could-be-killing-video-game-preservation
- Educational game - Wikipedia. Retrieved November 29, 2018 from: https://en.wikipedia.org/wiki/Educational_game
- Learning Styles - Teach.com. Retrieved November 27, 2018 from: https://teach.com/what/teachers-know/learning-styles/
- Instruction on how to wrap LibGDX project with Java JRE https://github.com/libgdx/libgdx/wiki/Bundling-a-JRE

References for the artwork and music

- *Used for the map assets:*

  https://opengameart.org
  https://itch.io/game-assets/free/tag-2d
- *Background Pictures for Menu, Victory, Error and Game Over Screen*

  http://www.argentus.com/revamp-talent-management-avoid-hiring-headaches-leave-recruitment-pros/
  https://www.pinterest.dk/pin/19815845232430426
  http://www.ebaumsworld.com/pictures/knights/82507783/ 5
  https://www.pinterest.dk/pin/301530137531337369/
  This picture is originally from the game Skyrim:
  https://wallpaperscraft.com/download/skyrim_world_rocks_winter_cold_the_elder_scrolls_v_skyrim_2244/800x600

- *Music:*

  Menu Screen - https://www.youtube.com/watch?v=E9nZbZfWQoI
  Level Screen - https://www.youtube.com/watch?v=NknjE2SBPxw
  Fighting Screen - https://www.youtube.com/watch?v=7OApaSxaPYw
  Victory Screen - https://www.youtube.com/watch?v=II04E2GEJG8
  Lose Screen - https://www.youtube.com/watch?v=kE186w91YVU
- *Assets for Fighting Screen:*
  - Animated sprites:

    https://www.gameart2d.com
    http://unluckystudio.com/free-2d-fantasy-characters-assets-for-game/
  - Background Picture:

    This picture is originally from the game Darkest Dungeon
    https://twitter.com/sylvainsarrailh/status/766164871188770816
- *Assets for the Tutorial and Story:*

  https://www.pinterest.dk/pin/250020216790249800/
  https://silentfield.artstation.com/projects/oJOVw

https://www.redbubble.com/people/thebeststore/works/12959061-keyboard-escape-key?p=poster
http://sburngdl.weebly.com/rules-of-game.html

Appendix

Appendix(1)

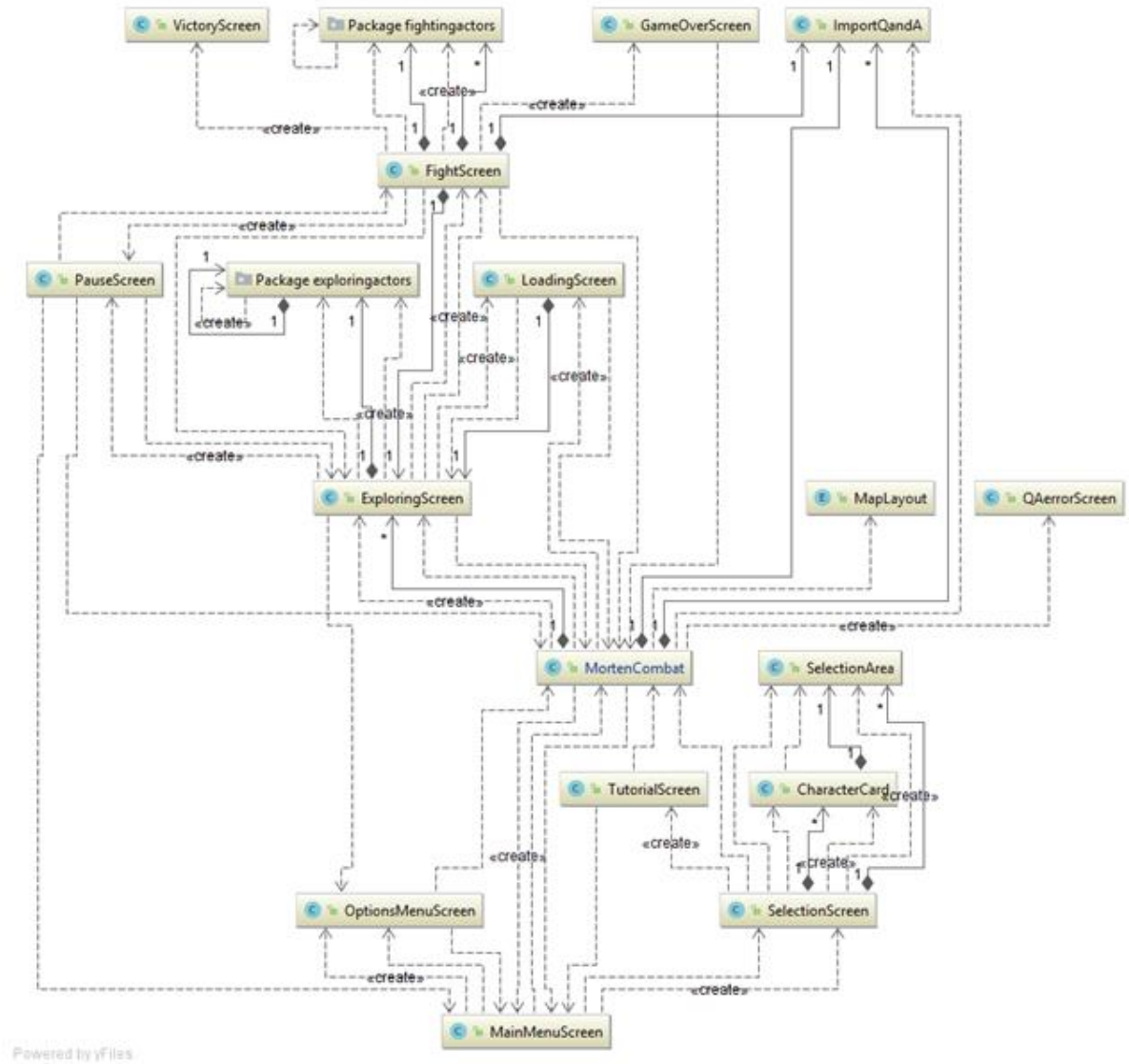


Figure - A generated minimized class diagram from IntelliJ IDEA

Appendix (2)

Figure B – This is all the public methods and fields in the Screen package.

Appendix(3)

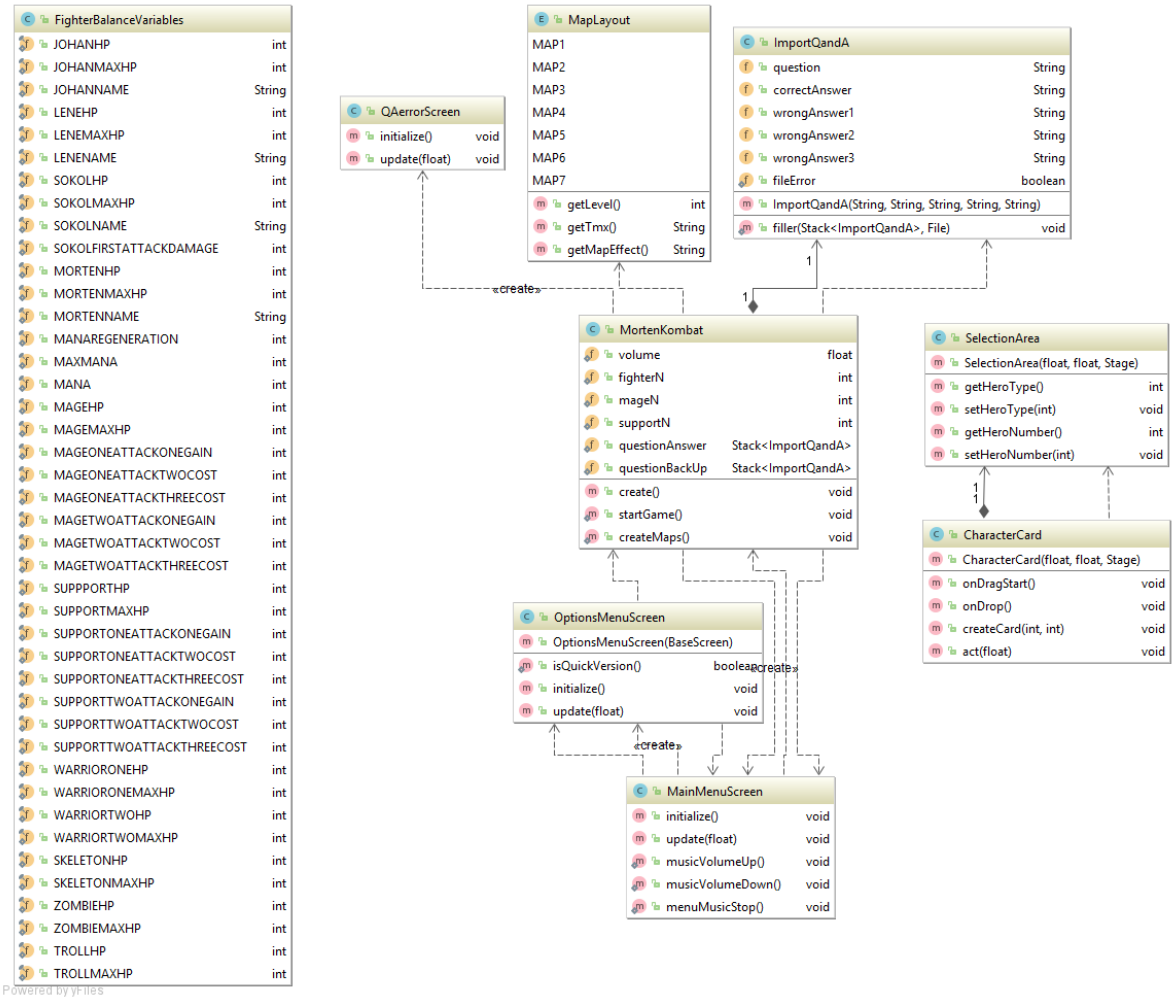


Figure C – This is the utilities

Appendix (4)

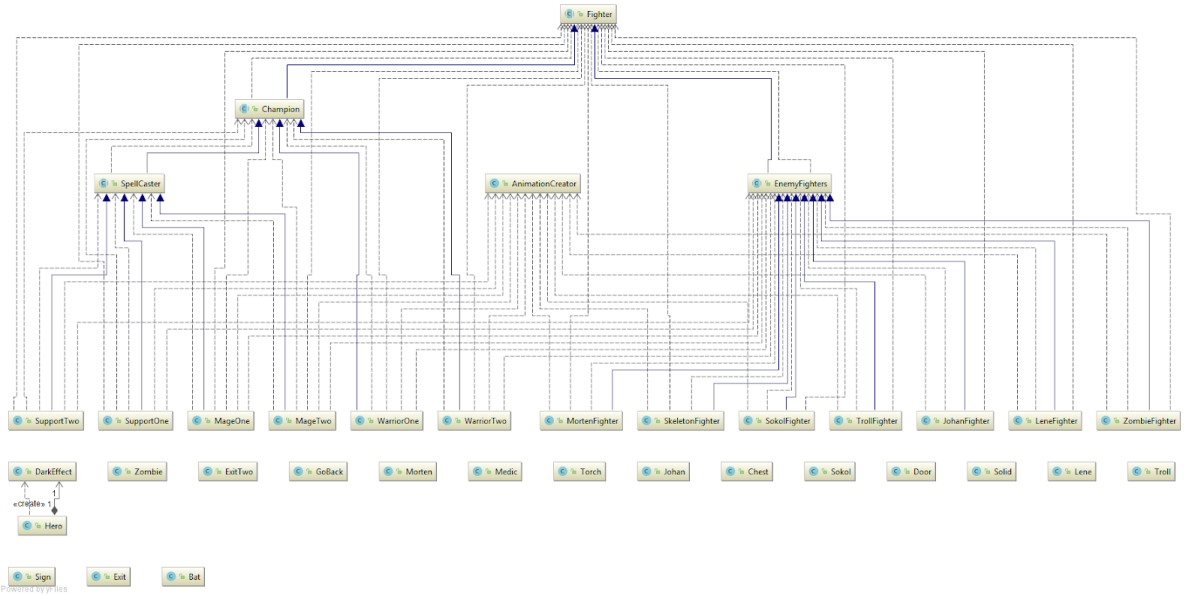


Figure D – This is the hierarchy of the Fighters

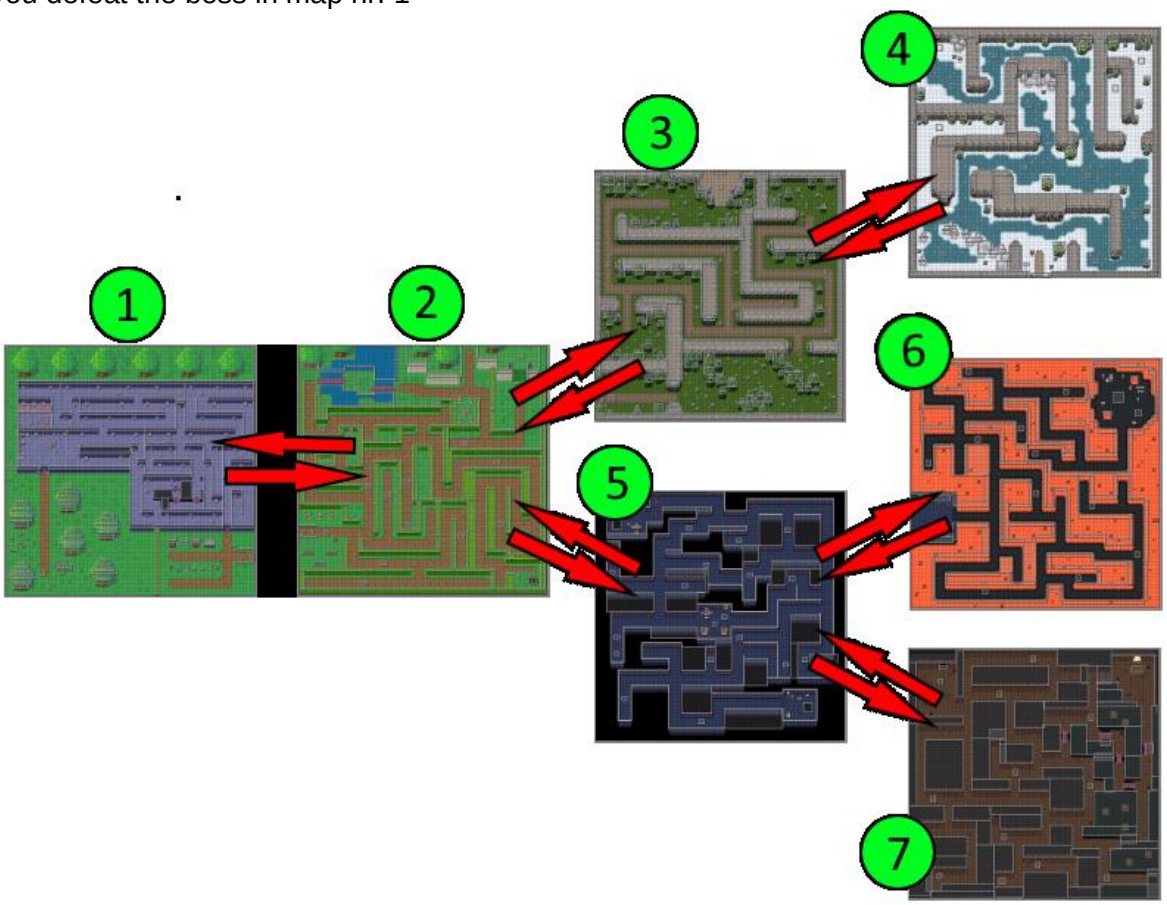
Appendix (5)

Introduction to game testing:

We are group 5 from ITCOM and we would like you to try our game-Morten Kombat which we created for the first semester project. The game is an RPG educational game, you will navigate mazes and answer trivia questions during fights. You will receive more instruction during game play. You will be asked to answer a survey after completing the game.

During the making of the game, the main focus for us was programming and thus that's where most effort went. You might question the graphics but that's due to the fact that we used what we could find online for free and as a consequence sometimes some character animations might not make sense and seem like rather a random choice. We appreciate your understanding.
There will be no personal information collected, and the survey results will purely be used to improve our game, some data collected might be presented in the report. Thank you for taking the time to play the game and answering the survey.

Tip: the map layout below is to help you find your way. Bosses are located in map no. 4,6,7 and 1. You have to go back to previous map(s) every time you defeat a boss and you win the game when you defeat the boss in map nr. 1



Appendix (6)

Game Survey 1

We thank you for playing Morten Kombat and participating in the survey. This survey is to help us get feedback on the game and make possible improvements.

* Required

1. **How easy was it to navigate through the different maps/mazes?** * *Mark only one oval.*

	1	2	3	4	5	
Very easy	◯	◯	◯	◯	◯	Very difficult

2. **Did you feel that there were clear different themes presented in the different mazes?** * *Mark only one oval.*

	1	2	3	4	5	
Strongly agree	◯	◯	◯	◯	◯	Strongly disagree

3. **Do you feel that the gliding, wind and dark room effects give you the impression and immersion of being in the different respective maps (ice, windy, dark maps)?** * *Mark only one oval.*

	1	2	3	4	5	
Strongly agree	◯	◯	◯	◯	◯	Strongly disagree

5. **In terms of length, how were the fights?** * *Mark only one oval.*

	1	2	3	4	5	
Very short	◯	◯	◯	◯	◯	Very long

6. **How did you find the questions difficulty?** * *Mark only one oval.*

	1	2	3	4	5	
Very easy	◯	◯	◯	◯	◯	Very difficult

How was the amount of questions presented during the fights? * *Mark only one oval.*

Very few	◯	◯	◯	◯	◯	too many

7. **How was the amount of questions presented during the fights?** * *Mark only one oval.*

	1	2	3	4	5	
Very few	◯	◯	◯	◯	◯	Too many

8. **Was the tutorial helpful?** *
   Mark only one oval.

	1	2	3	4	5	
Strongly agree	◯	◯	◯	◯	◯	Strongly disagree

9. **Did you feel challenged throughout the game?** * *Mark only one oval.*

	1	2	3	4	5	
Strongly agree	◯	◯	◯	◯	◯	Strongly disagree

10. **How entertaining did you find the game?** * *Mark only one oval.*

	1	2	3	4	5	
Very entertaining	◯	◯	◯	◯	◯	Very boring

11. **Did you think there was a good balance presented between the educational and entertaining aspect of the game?** *
    Mark only one oval.

	1	2	3	4	5	
Strongly agree	◯	◯	◯	◯	◯	Strongly disagree

12**. How satisfied were you with the gaming experience?** * *Mark only one oval.*

	1	2	3	4	5	
Very satisfied	⬭	⬭	⬭	⬭	⬭	Very dissatisfied

14. **Any comments you would like to let us know (your comments are highly appreciated)**

Appendix(7)

Game Survey (2)

We thank you for playing Morten Kombat and participating in the survey. This survey is to help us get feedback on the game and make possible improvements.

* Required

2. **How was the amount of enemies encountered?** * *Mark only one oval.*

	1	2	3	4	5	
Too few	◯	◯	◯	◯	◯	Too many

3. **In terms of enemy abilities, how did you feel?** * *Mark only one oval.*

	1	2	3	4	5	
Very underpowered	◯	◯	◯	◯	◯	Very overpowered

4. **In terms of length, how were the fights?** * *Mark only one oval.*

	1	2	3	4	5	
Very short	◯	◯	◯	◯	◯	Very long

4. **How did you find the questions difficulty?** * *Mark only one oval.*

	1	2	3	4	5	
Very easy	◯	◯	◯	◯	◯	Very difficult

5. **How was the amount of questions presented during the fights?** * *Mark only one oval.*

	1	2	3	4	5	
Too few	◯	◯	◯	◯	◯	Too many

6. **In terms of winning fights, what did you think was more important- fighting strategy or knowing the answer to the questions?** *

Mark only one oval.

	1	2	3	4	5	
Fighting strategy						Knowing the answer

7. **Was the tutorial helpful?** * *Mark only one oval.*

	1	2	3	4	5	
Strongly agree						Strongly disagree

9. **Did you feel challenged throughout the game?** * *Mark only one oval.*

	1	2	3	4	5	
Strongly agree						Strongly disagree

10. **How entertaining did you find the game?** * *Mark only one oval.*

	1	2	3	4	5	
Very intertaining						Very boring

11. **Did you think there was a good balance presented between the educational and entertaining aspect of the game?** *
Mark only one oval.

	1	2	3	4	5	
Strongly agree						Strongly disagree

12. **How satisfied were you with the gaming experience?** * *Mark only one oval.*

	1	2	3	4	5	
Very satisfied						Very dissatisfied

12. **How likely are you to play the game again?** * *Mark only one oval.*

	1	2	3	4	5	
Very likely	⬭	⬭	⬭	⬭	⬭	Very unlikely

13. **Any comments you would like to let us know (your comments are highly appreciated)**

Appendix (8)