UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Silver Kirotar

Real-Time Strategy on Platforms Game Design

Bachelor's Thesis (9 ECTS)

Supervisor: Raimond-Hendrik Tunnel, MSc

Tartu 2018

Real-Time Strategy on Platforms Game Design

Abstract:

This thesis describes the design of and the game mechanics in real-time strategy and platform computer game genres. The paper represents, how the mechanics from these genres were combined as one genre, real-time strategy on platforms, and were used to design a new computer game. The designed game *Queen Aerisilium* was created as a part of this thesis. The game is a real-time strategy game, which is using the mechanics of a platform game genre. The general user-friendliness and the peculiar game mechanics of the developed game are evaluated by user testing.

Keywords:

Computer game design, real-time strategy, platformer, computer game, computer graphics, Unity

CERCS:

P170: Computer science, numerical analysis, systems, control

P175: Informatics, systems theory

Reaalajas strateegia platvormidel arvutimängu disain

Lühikokkuvõte:

Käesolev bakalaureusetöö kirjeldab reaalajas strateegia- ning platvorm arvutimängude žanrite disaini olemust ning mängumehaanikaid. Töö kirjeldab, kuidas nende žanrite mehaanikad kombineeriti üheks mängužanriks – reaalajas strateegia platvormidel – ning mille alusel disainiti uus arvutimäng. Töö raames loodi mäng nimega „Queen Aerisilium“. Tegemist on reaalajas strateegia mänguga, milles kasutatakse platvormimängu mehaanikaid. Kasutajatega läbiviidud testimise põhjal on hinnatud valminud mängu üldist kasutajamugavust ning selle omalaadseid mängumehaanikaid.

Võtmesõnad:

Arvutimängu disain, reaalajas strateegia, platvormimäng, arvutimäng, arvutigraafika, Unity

CERCS:

P170: Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

P175: Informaatika, süsteemiteooria

Table of Contents

1 Introduction

Computers have been used to play games since the end of the 1950s, but playing computer games was not popular until the 1970s and 1980s, when the personal computers (PC) were introduced to the public[1]. PC games are usually called video games[2] (games) and there are different genres to categorise them. Real-time strategy (RTS) and platform (platformer) genres were two among the earliest game genres.

The popularity of real-time strategy games started growing in the 1990s [1]. There is an incomplete list of RTS games that are known to be released before the year 2017, currently there have been listed 365 games[3]. Some of the most popular RTS game series include: *StarCraft*, *Company of Heroes*, *Age of Empires*, *Total War*, and *Command & Conquer*[4].

The development of platform video games started at the beginning of the 1980s. The popularity of platformers started growing ever since. In addition, platformer games are still popular thanks to the mobile game market[5]. By the time being, there are currently over 270 known platform game series[6]. Some of the most popular platform game series include: *Super Mario*, *The Legend of Zelda*, *Rayman*, *Prince of Persia*, *Sonic the Hedgehog*, and *Earthworm Jim*[7].

Both platform and RTS games have a large player base, most of whom are long-time players. However, for the last 10 years, there has been a downfall in the creation and popularity of real-time strategy games [2, 3]. The purpose of introducing the platformer game mechanics in an RTS game was to give the players something new that could potentially freshen their interest in RTS games.

This thesis describes the main game mechanics from both real-time strategy and platformer game genres. Those game mechanics are then combined into a new game genre and then used for the design and implementation of a single prototype game – *Queen Aerisilium*. The development of the prototype was complemented in the MTAT.03.328 Computer Graphics Project course.

---

[1] https://en.wikipedia.org/wiki/Early_history_of_video_games
[2] https://en.wikipedia.org/wiki/Video_game
[3] https://en.wikipedia.org/wiki/List_of_real-time_strategy_video_games
[4] https://www.lifewire.com/top-real-time-strategy-game-series-813087
[5] https://en.wikipedia.org/wiki/Platform_game
[6] https://en.wikipedia.org/wiki/List_of_platform_game_series
[7] https://www.gamesradar.com/best-platform-games-ever-arent-mario

The thesis starts with the analysis and amalgamation of the real-time strategy and the platform game design elements in chapter 2. Chapter 3 explains the choices between different technologies, which could have been used and which was chosen for developing the game. Chapter 4 gives an overview of the game design special to *Queen Aerisilium*. Chapter 5 provides an overview which parts of the game design are implemented in the game, and what kind of difficulties appeared during the development. Chapter 6 describes how the game was being tested, the game design issues that were discovered, and the overall feedback from the testers. Chapter 7 provides a short overview of the accomplished work, the conclusion, and author's final remarks.

Some of the terms used in this thesis are defined in the Glossary (Appendix I). The build and the game design document of *Queen Aerisilium*, the project and the source code files, a demonstration video of the gameplay, and questionnaire data are available in the Accompanying Files (Appendix II).

2 The RTS-Platform Amalgamation

One of the most important aspect of any game is its mechanics. Different game genres use specific features and mechanics that should exist in that type of a game. The common way to describe a game is by its genre. Thus, it is important to formulate and analyse the mechanics from RTS and platform game genres. The objective is to describe the results of combining these two game genres. As combining these two genres has been tried out before, directly or indirectly, this thesis provides a short description about these similar games and compares them to the *Queen Aerisilium*.

2.1 Game Mechanics

Game mechanics describe the different goals in a game, ways for the player to achieve these goals by playing the game, and the results when achieving the goals was successful or unsuccessful [4:41]. Game mechanics form the core of a game, they are the relations between the objects and the possible interactions. Simply put, the mechanics are what gives the players a sense of the game being truly a game. J. Schell has come up with 6 different types of mechanics, that should qualify for most of the games [4:130–154]:

1) **Space** – the word "space" is often used for describing dimensions, a continuous area i.e. expanse, or a gap between specific time[8]. Schell has defined the space in games as a mathematical construct, which includes the first and second meanings from the definition of the word "space". In the terms of the space being a game mechanic, one would think of this space as an abstract construction. Abstract construction means that this space does not have any specific rules. The space of a certain game is described by Schell with the answers to these three simple questions:
    a. Is the space discrete or continuous?
    b. How many of dimensions are in this space?
    c. Are there bounded or unbounded areas in this space, and whether these areas are connected to each other or not?

[8] https://en.oxforddictionaries.com/definition/space

2) **Objects**, **Attributes**, and **States** – mechanics that describe the different things which exist in a game. Things that can be seen or manipulated in games are referred to as objects. The attributes are the parameters (e.g. the position) of an object. All attributes have a state that they are currently in, the values that define the state can usually be changed. States usually influence the behaviour of an object.
3) **Actions** – the type of mechanics, which describe what the players can do in the game. Actions are divided into two categories:
    a. Operative actions – actions that can be initiated by simply interacting with the game, like moving around or making in-game decisions.
    b. Resultant actions – actions that are more meaningful in the larger scale. They describe the influence of operative actions in achieving the goals of the game, like strategically sacrificing a character to trick the opponents.
4) **Rules** – the fundamental mechanics, which define the principles that the previously mentioned mechanics and goals are based on. For example, the characters controlled by the player can only move from the left to the right, where moving is an operative action.
5) **Skill** – mechanics with the goal to challenge the players. Games are generally designed to have different levels of difficulty, e.g. easy, medium, and hard. Players can usually adjust the game difficulty to the level that provides them the challenge most suitable for their personal skill.
6) **Chance** – mechanics used to provide the players uncertainty in the outcomes of every other mechanic mentioned before. Chance is used to make the games less tedious and to surprise the players.

Gameplay defines the way how a player can specifically interact with the mechanics in that certain game. It also includes the plot and the means on how to play the game. Gameplay is often described as features of what can be done in the game, like shooting and crafting objects. The most important factor in gameplay is what kind of experience it gives to the players [5].

The idea of the *real-time strategy on platforms* (RTSoP) amalgamation is to combine the basic game mechanics from both the real-time strategy (RTS) and platform games (platformer) as one game genre. The intention of combining these mechanics is to give the players a new kind of gameplay experience as a combination from both video game genres, and to renew their interest in RTS genre.

2.2 Platformer

The gameplay of a platformer takes place on platforms (ledges). Characters in the game are objects with different states like moving and jumping. The players generally have the possibility to make their characters perform different actions like walking around or jumping from one platform to another [6:118]. The platforms can have different states like being fixed in one place, moving or floating around, and be moveable by the player. Platforms could be arranged on different heights, thus creating a two-dimensional space [7].

One dimensional gameplay means that the progress of the game advances only in one direction [8:163]. Arranging platforms on different heights will add a second dimension to the gameplay, like in *Super Mario* (1985 – 2017)[9]. Some older games with one-dimensional gameplay added a hint of the second dimension, like having to jump over obstacles in *Athletic Land* (1984)[10]. There are also three-dimensional platform games, where the gameplay is not one dimensional, like *Alice: Madness Returns* (2011)[11]. See Figure 1 below for example images from these games.

Figure 1. From left: *Super Mario*, *Athletic Land*[12], and *Alice: Madness Returns*[13].

Scrolling is a game mechanic, which means that the main gameplay moves in one continuous direction. This results in that the players are bound to that area in game, to where it has progressed. Scrolling is often used in platform games and it is characteristic to the side scroller game sub-genre [7, 9].

[9] https://en.wikipedia.org/wiki/Super_Mario
[10] http://www.mobygames.com/game/msx/athletic-land
[11] https://en.wikipedia.org/wiki/Alice:_Madness_Returns
[12] http://www.mobygames.com/game/msx/athletic-land/screenshots/gameShotId,133373/
[13] Screenshot from the video: https://www.youtube.com/watch?v=UHH8HRtjl-o

2.3 Real-Time Strategy Games

Strategy video game (strategy game) is a computer game genre, where achieving victory relies greatly on player's skilful thinking and planning. Games in this genre are generally divided into 4 categories (sub-genres). These categories come from the combination of the following: either the gameplay is turn-based or real-time and focused on strategy or tactics[14].

Turn-based strategy is a sub-genre of strategy games. The gameplay in these games is turn-based and has multiple players (some of these players can be can be controlled by the game's artificial intelligence algorithms[15]). The actions of the players take place in turns. This gives them time to come up with the next actions while waiting for the start of their own turn or even during that turn[16].

Real-time strategy games can be war or simulation games (not limited to). The gameplay of an RTS game is related to the turn-based strategy games, and similarly can have multiple players. In RTS the players should come up with their actions and use them quickly, concurrently with their opponents [10].

The purpose in an RTS war game is building a better (military) base or creating a stronger offensive force, and preferably faster than the opponents. The key to winning depends on, as the name emphasizes, using better strategies than the opponents. Quick acting also tends to contribute towards winning.

There are usually different strategies that the players can pay more attention to, like emphasising on the economic dominance or the battle-tactics [7]. One of the characteristic strategies in RTS games is *turtling*, this is a type of approach, where at least one of the players builds up their defences and baits their opponents into attacking them. The goal of this strategy is to let the player's opponents deplete their resources until the player has assembled a stronger offensive force and senses it is time for a counter-attack. This way the opponents usually do not have time or resources to assemble a defensive force, and thus will be quickly overrun [11].

[14] https://en.wikipedia.org/wiki/Strategy_video_game
[15] https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games
[16] https://en.wikipedia.org/wiki/Turn-based_strategy

2.4 The Result of Amalgamation

Real-time strategy on platforms (RTSoP) is a combination of RTS and platformer genre. The mechanics and properties of the platformer games that were used for the RTSoP game in this thesis are the following:

1) Platforms are on different heights.
2) Platforms are fixed in one place.
3) Players can move their characters from one platform to another.

All the mechanics about the RTS games named in Chapter 2.3 were used in designing *Queen Aerisilium*. This means that the RTSoP is different from a regular RTS game solely by the fact that there are platforms and characters can only be moved to the other platforms, when one of the players has built bridges between these platforms.

Queen Aerisilium can be mistakenly labelled as a side-view or rather as a side-scrolling platform game. The gameplay is viewed from the side and the game is 2D, which are both characteristic to a side-scroller. However, this game is not using the scrolling mechanic [9].

2.5 Similar Games

There are a few similar games that are using mechanics from both the RTS and platformer game genres. A 3D RTS game named *Driftland: The Magic Revival* (in early access since 2017)[17] (*Driftland*) is using platforms that are depicted in the game as flying islands, which can be connected by attaching bridges between them, see Figure 2 below.

Figure 2. From left: Misplacing a Wooden Bridge in *Driftland*, placing it in a correct spot, and building in progress.

[17] http://www.stardrifters.com/driftland/

In *Driftland*, players can move their characters from one platform to another on foot by walking over bridges, or by crossing the void between platforms while being mounted on a flying beast[18] (see Figure 3). Although in the game it seems like that the characters that are mounted on a flying beast cannot be dismounted. Currently, the game's unofficial wiki[19] lacks information about mounting and dismounting.

Figure 3. A Knight on top of an Eagle flying over the gap between islands in *Driftland*.

There is also a series of games titled *Clonk*, from which the games are mainly a combination of the mechanics from three different game genres: action, RTS, and platformer. Games from the *Clonk* series were developed by RedWolf Design[20]. The company has published a total of 8 games (1994 – 2008) with the *Clonk* title. One of these games is *Clonk Rage* (2008), which is the latest published title by this company[21].

The player in these Clonk games can control the characters, known as Clonks, with a mouse or a keyboard. The mechanic for controlling units with a mouse works similarly as in other popular RTS games, like the *Age of Empires*[22] series and *Driftland*. The player can use the mouse to command the units to do different actions like move somewhere or gather a

[18] https://driftland.gamepedia.com/Flying_beast
[19] https://driftland.gamepedia.com/Driftland_The_Magic_Revival_Wiki:General_disclaimer
[20] https://en.wikipedia.org/wiki/Clonk
[21] http://www.clonk.de/
[22] https://en.wikipedia.org/wiki/Age_of_Empires

resource. The gameplay can take place in the sky (see Figure 4) and it is possible to build bridges between platforms (see Figure 5).

Figure 4. Floating islands in the sky in *Clonk Rage*.

Figure 5. Building bridges in *Clonk Rage*.

Overall, there are some popular games using mechanics from both RTS and platformer genres, but describing existing games is not the goal of this thesis. The objective is to design and implement a game, named *Queen Aerisilium*. For the latter, a development environment is needed. Thus, before beginning to implement that designed game, a game engine is chosen.

3 Technologies Used

The author decided to use a game engine because game engines usually have a dedicated development environment, which can speed up the development process. Building the developed game with a game engine is also easier than without[23]. Some of the game engines also have cross-platform support, which means that the games can be easily distributed for different operating systems. For developing the game *Queen Aerisilium*, three different game engines were considered: Unity[24], GameMaker Studio 2[25], and LÖVE. The Unity game engine was chosen out of these three for reasons described in the following chapters.

3.1 GameMaker Studio 2

GameMaker Studio 2 is a game engine for creating 2D video games. GameMaker offers drag and drop action sequences for those who do not have experience in programming. It has its own scripting language, the Game Maker Language, to write scripts for the game elements. The environment was developed by YoYo Games Ltd[26]. YoYo Games is currently offering three different licenses for making computer games with GameMaker Studio 2[27].

1) The Creator license that costs 39$ and lasts for 12 months.
2) The Developer license that costs 99$ and is permanent.
3) The Trial license that is free, can be used to watch tutorials and build one game.

The first option would have been best for creating the game for this thesis only, as it lasts for a year. The second option of GameMaker Studio's license could have been a better choice for the author as it allows him to start developing alone in his own pace, while being able to publish the game or updates years later. In the long term, there would be a possibility to increase the team and easily buy more licenses. In the short term, for the creation of this thesis, it did not seem reasonable to spend money on buying the license, if there are other suitable game engines that can be used for free. The third option was not suitable for the author, because it does not allow the creation of executable packages thus publishing and

[23] https://gamedev.stackexchange.com/questions/859/what-are-the-advantages-and-disadvantages-to-using-a-game-engine
[24] https://unity3d.com/
[25] https://www.yoyogames.com/gamemaker
[26] https://en.wikipedia.org/wiki/GameMaker_Studio
[27] https://www.yoyogames.com/get

sharing the game is not possible. It also has certain limitations on how many different types of resources can be in the game[28].

3.2 LÖVE

LÖVE[29] (also known as Love2D[30]) is a free to use, open-source framework for creating 2D games. The programming language Lua is used to write scripts for the game elements. LÖVE does not have its own development environment or tools, but as it is open-source, one could create their own environment or use the templates[31] made by other users. LÖVE's libraries[32] contain many modules that were made by the enthusiasts of this engine. Unfortunately, their names are quite unmeaningful and this makes using these modules difficult and tedious. For instance, one camera module has a name STALKER-X[33], which is irrelevant to its functionality.

3.3 Unity

Unity is a cross-platform game engine for creating both 2D and 3D games. The platform was developed by Unity Technologies SF[34]. Unity Technologies SF is currently offering three different licenses.

1) Unity Personal that is free to use with these limitations: The company's annual gross revenues and raised funds must be less 100 000 dollars[35]. Unity Personal is free to use for students to improve their skills outside the classroom[36].
2) Unity Plus that costs 35$ per month and can be used with certain limitations: The company's annual gross revenues and raised funds must be less than 200 000 dollars[37].
3) Unity Pro that costs 125$ per month and can be used without revenue and fundraising capacity limitation[38].

---

[28] https://help.yoyogames.com/hc/en-us/articles/230407528-GameMaker-Studio-2-Trial-Limitations
[29] https://love2d.org/
[30] https://gamedevelopment.tutsplus.com/articles/how-to-learn-love-love2d--gamedev-4331
[31] https://github.com/CodeZombie/Love2D-Template-Environment
[32] https://github.com/love2d-community/awesome-love2d
[33] https://github.com/SSYGEN/STALKER-X
[34] https://en.wikipedia.org/wiki/Unity_(game_engine)
[35] https://store.unity.com/products/unity-personal
[36] https://store.unity.com/education
[37] https://store.unity.com/products/unity-plus
[38] https://store.unity.com/products/unity-pro

Unity was used as the game engine for this project because the author has a fair amount of experience with this game engine and programming in C#. Unity and C# were also taught in two different university courses that the author took. Compared to GameMaker Studio 2 and LÖVE, Unity also has 3D support, which means that the 2D graphics could be replaced by 3D graphics without further complications and thus change the 2D game into 2.5D.

The game was created using the Unity Personal version 2017.3[39], which was the latest available version during the beginning of the development of *Queen Aerisilium*. Unity Personal was chosen because compared to Unity Plus and Unity Pro it is free to use, and the project is currently being developed for educational purposes, without any fundraising or revenues.

Choosing a framework with essential libraries and a comfortable development environment is crucial for speeding up the game making process. Thus, the Unity game engine was selected for developing *Queen Aerisilium*, due to the reasons described in previous paragraphs. The next chapter explains the most important game design choices, which are partially implemented in the prototype game.

[39] https://unity3d.com/unity/whats-new/unity-2017.3.0

4 Game Design

The following chapters describe the most important RTSoP genre-related game design choices for the elements in the game *Queen Aerisilium*. These game design choices are also described in more detail in the game design document (see the Appendix II).

4.1 Bridges

There are not many RTS games that use bridge building as a game mechanic. Buildable bridges are known to be used in a few RTS games. However, here are some games that do use this mechanic: *Banished*[40], *Driftland*, *Empire Earth*[41], and the *Clonk* series.

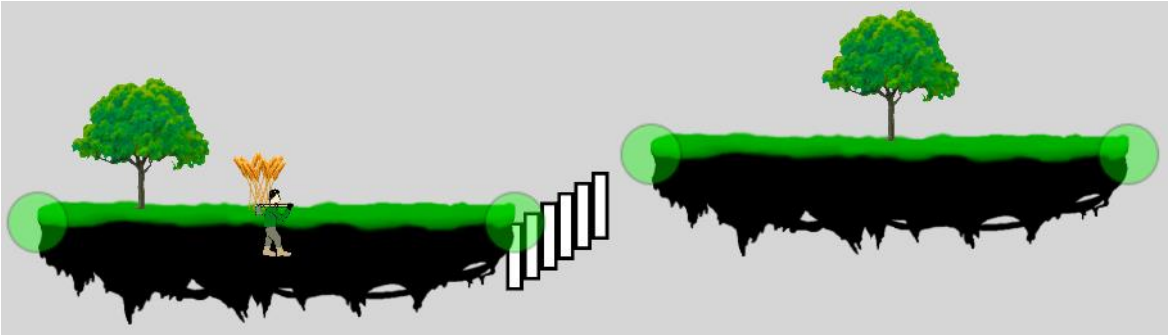

Figure 6. Bridge (white boxes) connected to one of the connection points (green circle) in the middle of two platforms (floating islands).

Queen Aerisilium uses bridge building mechanic to connect platforms, or rather floating islands, and to gain access to different platforms. It is yet to be decided, if the players should be able to destroy bridges or not. Every platform has a total of two **connection points**, first on the left and second on the right. Bridges can be built between two connection points, each from a different platform. See Figure 6 below for an example.

4.2 Units

Currently, there have been designed 5 types of units in the game design document, these are: swordsmen, workers, scouts, archers, catapults. See Figure 7 for the example images of the first 4 units. The following chapters describe in what ways they are unique to RTSoP. These units are categorised in three groups: infantry, civilian, and siege units. Infantry units are swordsmen, scouts, and archers. Workers are civilian and catapults are siege units.

[40] http://www.shiningrocksoftware.com/game/
[41] https://en.wikipedia.org/wiki/Empire_Earth

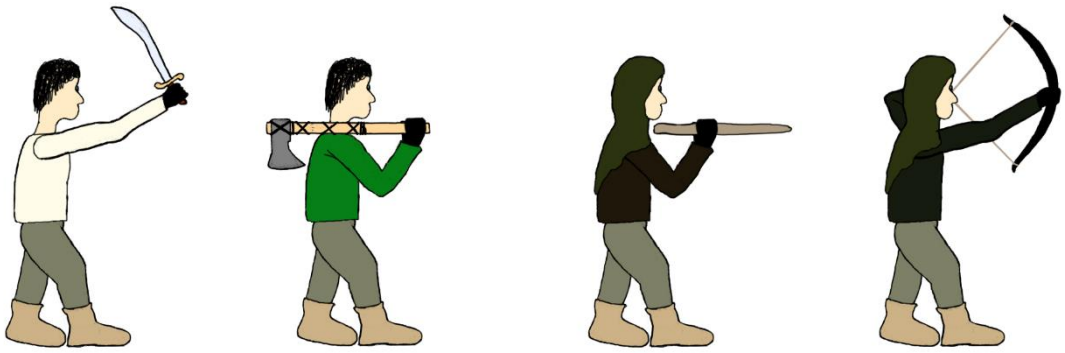

Figure 7. Placeholder images for current units in *Queen Aerisilium*. From left: a swordsman, a worker, a scout, and an archer.

All the previously mentioned units have offensive and defensive actions which grant them the ability to attack enemy units and buildings, or defend themselves from them. These units have attributes like health, attack range, cooldown between strikes, and a chance to miss their strikes. They can travel between islands by walking over a bridge. Like in most RTS games, units are controlled with the mouse.[42]

4.2.1 Swordsmen and Workers

Swordsmen and workers are regular units who do not directly differ from units in regular RTS. These units can attack enemy units that are at their elbow and on the same platform, bridge or in the connection point (see chapter 4.1).

A swordsman[43], like the name suggests, is a skilled infantry warrior whose main offensive weapon is a sword. Swordsmen are the strongest in both offensive and defensive abilities of all infantry units. Their main weakness is that they can be easily killed by archers and catapults from a distance.

A worker is a peasant[44] who can do all sorts of manual labour, e.g. being a woodcutter, a farmer, or a miner when needed. Workers also build houses and bridges. They are fairly experienced in using axes and thus having moderate offensive abilities, but their defensive abilities are poor, due to lack of training and armour. A worker can only gather resources in the prototype game made for this thesis.

[42] https://en.wikipedia.org/wiki/Real-time_strategy
[43] http://www.dictionary.com/browse/swordsman
[44] http://www.dictionary.com/browse/peasant

4.2.2 Scouts

Scouts are a type of infantry units that can move from one platform to another without needing bridges between platforms, they are specifically designed to discover new islands. Other units cannot get to another platform without bridges, this action is specific to the scouts only. Scouts walk the fastest, but have the smallest amount of health among all the units. They also have the weakest offensive and defensive abilities, thus being overall the weakest unit in the game.

The idea of a scout travelling to another platform without a bridge is that they throw a rope to another platform. After throwing, the character would then attach the other end of the rope to the platform they are currently standing on. When the rope is in place, the scout simply needs to climb it, as shown in Figure 8 below.

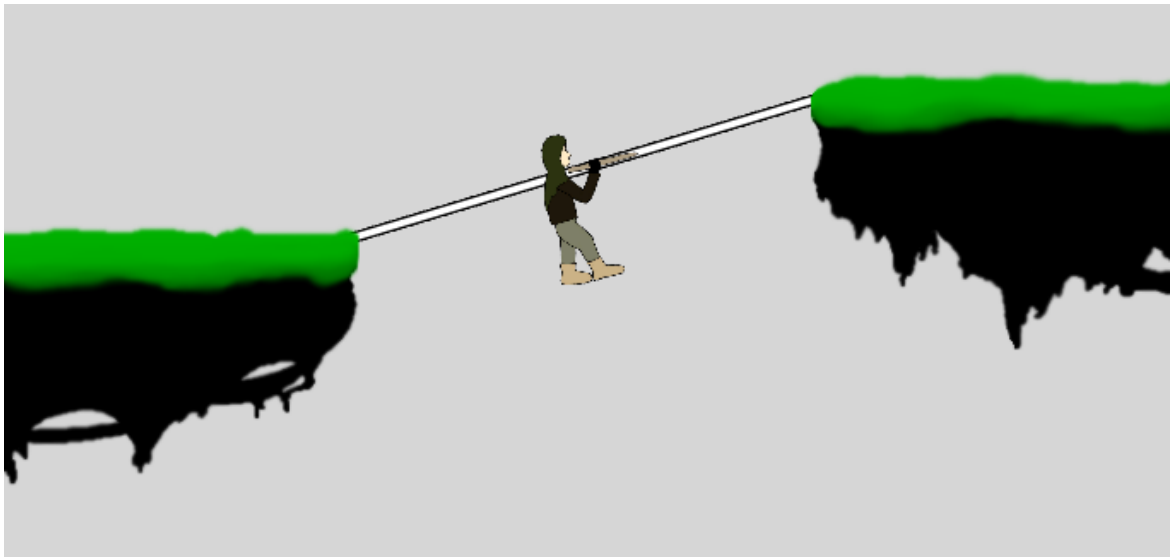

Figure 8. An illustration (this feature is not implemented in the *Queen Aerisilium*) of a scout crossing two platforms (islands) by climbing on a rope.

A scout is essential for the game design simply because they are designed to discover new lands that remain hidden in the fog of war[45] (see more about it in the game design document, in Appendix II).

[45] https://en.wikipedia.org/wiki/Fog_of_war

4.2.3 Archers

The archer is a strong infantry warrior whose main offensive weapon is a bow. An archer can attack enemies from afar by shooting arrows with a longbow, or by stabbing those that are within the reach of their long knife. Unlike the regular infantry units, archers can also attack distant enemies, who may be on a different platform. Archers have the greatest attack range, moderate offensive and lesser defensive abilities, and short cooldown between consecutive shots. To reduce the chances of exploiting the great attack range of archers, they have the biggest chance (see chapter 2.1 about chance) to miss their targets.

In *Queen Aerisilium*, the design of the archers is like in some (e.g. *Clonk* and *TowerFall*[46]) of the platformer games instead of regular RTS games. Although, aiming in these platformer games is usually done manually by the player, but in this game, aiming and shooting would be automatic. Arrow shooting mechanic and other attacking methods were not implemented in the prototype, due to the players not having any enemies yet.

4.2.4 Catapults

The catapult, specifically a mangonel[47], is a siege weapon that launches stone projectiles. This siege weapon is the most effective unit at demolishing enemy buildings. Mangonels have an area of effect (AoE) attacking style, which means they can damage multiple enemy units simultaneously. AoE is important for the game design because a player can stack together many archers and other units. This way the player could potentially make a great amount of damage to their enemies. With AoE, this advantage can be quickly eliminated.

Catapults are not currently in the developed game. Just as the other attacking methods mentioned in the previous subchapter. The following chapter describes some of the mechanics that are implemented in the prototype game.

[46] http://www.towerfall-game.com/
[47] https://en.wikipedia.org/wiki/Mangonel

5 Implementation

This paragraph describes some of the most difficult methods that were vital for the mechanics from RTS, Platformer, or RTSoP genre, and thus were implemented in *Queen Aerisilium*. To get more insight about the features implemented in the game, see the player manual in Appendix III.

5.1 Architecture

A problem came out when the author added the first unit to the game. The unit was using a Rigidbody2D[48] and a CircleCollider2D[49] components to find collisions with different objects. When the number of units was increased by a few dozen units, the framerate (frames per second or FPS) dropped drastically, and it was nearly impossible to play the game.

Three tests were conducted manually to prove the issue. The results were measured with Unity Profiler Window[50]. The FPS was measured while the units were standing still and while they were moving. Each test included 256 worker units, which were stacked together. The number of units is roughly taken by relying on non-functional requirements that are listed in the game design document (see Appendix II). The goal is that the game should eventually support at least 4 players while each player can play with 50 units at once. The tests conducted were the following, distinguished by the components used:

1) RigidBody2D and CircleCollider2D (trigger) – the average result was about 100 FPS while the units were standing and 1 while they were moving.
2) CircleCollider2D (trigger) – the average result was about 500 FPS while standing and about 20 while moving. This implementation is currently used in the latest version of the game.
3) None – the average results were about 500 FPS while standing and about 300 while moving. First frame at the beginning of walking spiked down to 60 FPS at most, which is the frame when all the units get their target location.

These test results indicate that the final game should have its own targeted physics system, which could potentially be almost as effective as the results from the third test. To solve this problem, a child-parent relationship system was implemented. There is one parent for all

[48] https://docs.unity3d.com/ScriptReference/Rigidbody2D.html
[49] https://docs.unity3d.com/ScriptReference/CircleCollider2D.html
[50] https://docs.unity3d.com/Manual/Profiler.html

the platforms and one for the bridges. Each unit is a child of the platform or bridge they are currently standing on. Buildings and resources can only be the children of a platform, the parent is defined when the object is created. See Figure 9 for the illustration.

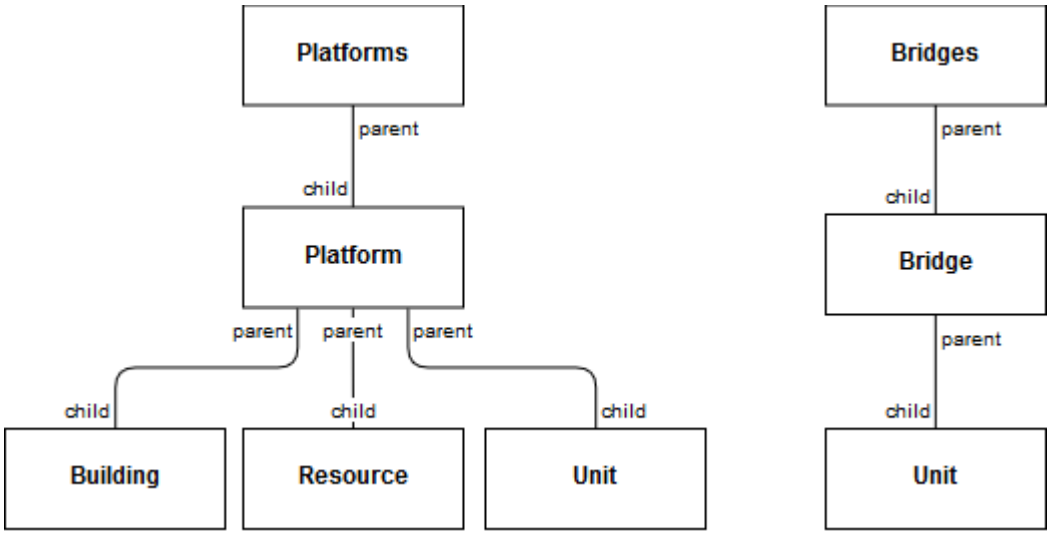

Figure 9. Child-Parent relationship between the game's objects.

This new child-parent relationship system is more efficient than using regular colliders due to there is no need to go through all objects to find the correct collision. Instead, the algorithm first checks on which platform or bridge an action was made, and then finds the corresponding object on that platform or bridge.

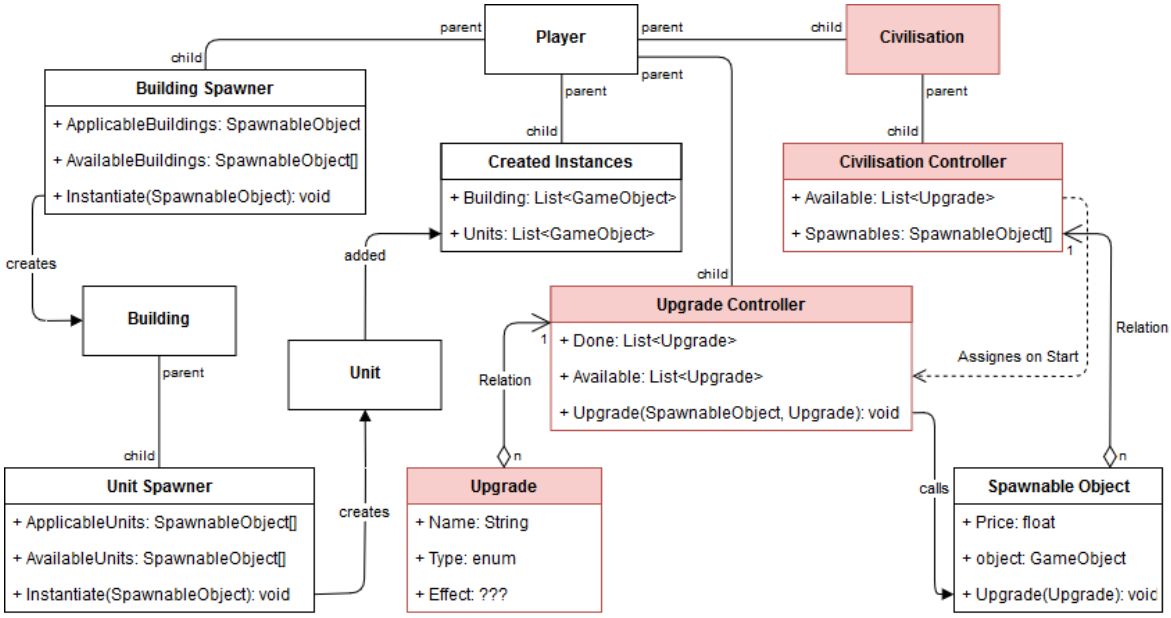

Figure 10. Illustrative image of how the code could be structured to add different upgrades and civilisations (red boxes) without any further complications.

Another problem occurred while designing the code structure. The goal was to create a modular system for spawning both units and buildings. This system had to take into consideration the player's civilisation and the upgrades for these units and buildings.

Current solution (see Figure 10) for this issue would be that there is a class called SpawnableObject. This class holds the information about the game's object (a unit or a building), the reference to the already instantiated objects of the same type, and other attributes like the price and health.

The idea is that whenever an upgrade is initiated, the SpawnableObject would get a notification about this upgrade, and then forward it to all the previously instantiated objects. The notifications would be received using the observer pattern[51]. The objects will then change their attribute values accordingly.

5.2 Pathfinding

Every platform has two connection points, which are the nodes in a big graph created from all the platforms in the game. These two nodes are already connected with each other, due to having the island between them. Bridges use these same nodes, and thus do not have the nodes of their own. At the start of the game, the platforms are not connected with each other. The pathfinding algorithm attempts to find the path from one platform to another by looking up the connections between the two nodes. When two paths are found, the shortest will be selected. See Figure 11 for the structure of the graph.

The current implementation of the pathfinding uses a modified version of Dijkstra's algorithm[52], as opposed to the A* algorithm[53] which is typically used in the popular RTS games (e.g. *Age of Empires*). The current implementation works well, but the algorithm goes through the graph every time a unit is assigned to move somewhere (see Figure 11 and 12). This procedure is quite inefficient and slow for assigning hundreds of units at the same time (see conducted tests in chapter 5.1). The idea is to modify the implementation so all possible paths are updated every time when a player builds a new connection (bridge) between 2 nodes. This issue is intended to be corrected in the future development, which is beyond the scope of this thesis.

[51] http://www.gameprogrammingpatterns.com/observer.html
[52] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
[53] https://en.wikipedia.org/wiki/A*_search_algorithm

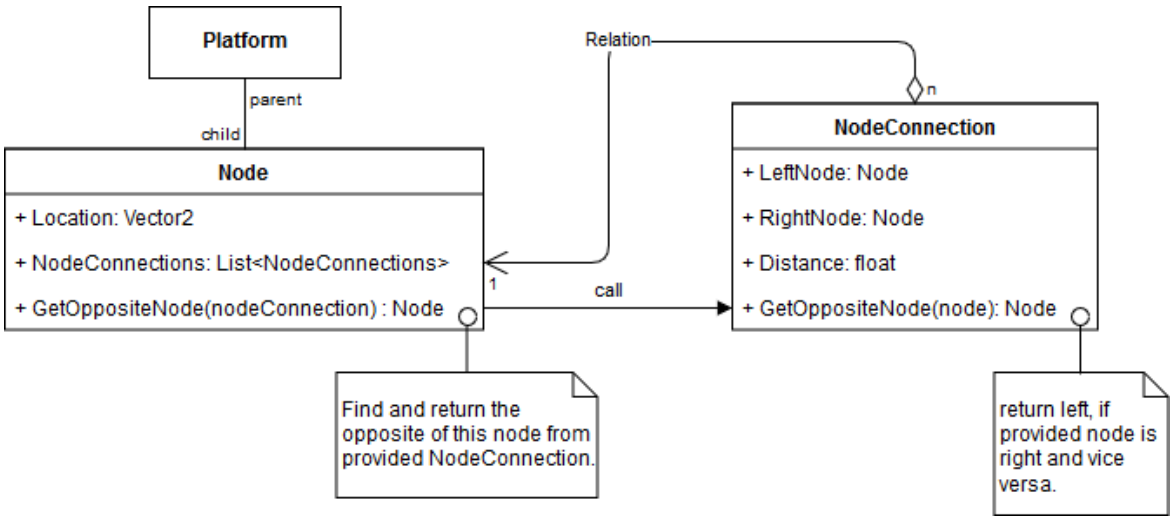

Figure 11. Every platform is a parent of 2 Nodes, each Node remembers the NodeConnections it is attached to.

The pathfinding algorithm requires 4 parameters: two NodeConnection objects and two Vector2 locations. One object is for the starting and one for the target platform or bridge. Similarly, the locations for the starting and the target points. The algorithm calculates two shortest paths from the starting object because every NodeConnection has 2 nodes. If the paths from both nodes exist, the shortest path will be chosen.

The modification to Dijkstra's algorithm is that it simply considers the starting location distance to the first node and the target location between two nodes. Specifically, considers the starting and target locations on the platforms, for deciding the actual length of the path.

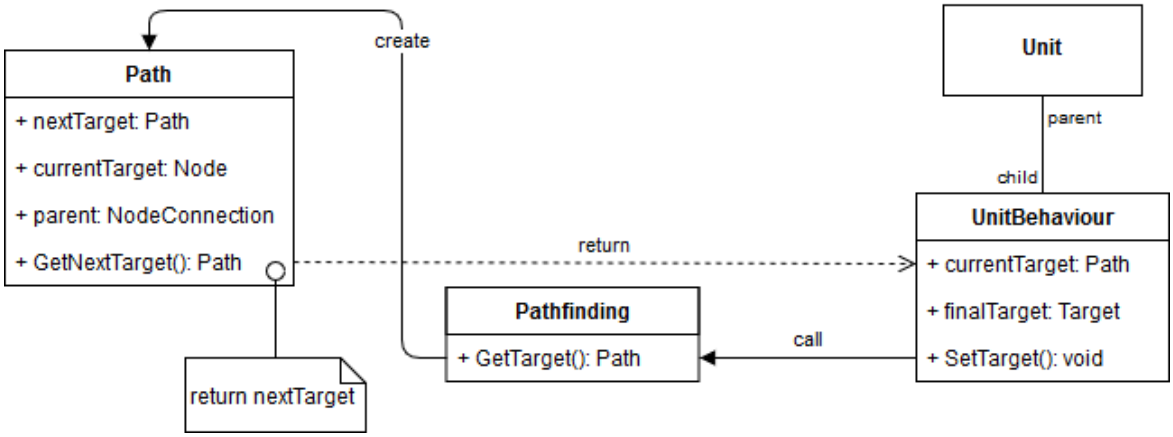

Figure 12. Unit sets a new target (*SetTarget* method) in the UnitBehaviour script, by calling the method *GetTarget* in the Pathfinding script.

The calculated paths will be set as the targets for the selected units (see Figure 12).

6 Testing and Results

Software testing is done to ensure the quality of the products or services that are being tested. Every software has a target audience[54]. The RTSoP game developed within this thesis is targeted for RTS players. Testing, in this case, is necessary to evaluate whether the prototype game of *Queen Aerisilium* gives these players an understanding, how playing a game from the RTSoP genre would feel like.

6.1 Method

The technique that was selected for the testing was *usability testing*[55]. Testers were selected on the basis that they must be familiar with the RTS concept and experienced at playing both RTS and platform games. The testing was kept short and simple, due to the prototype game is lacking actual gameplay (like attacking enemy opponents) and was created for demonstrative purposes.

Use cases that a tester had to achieve:

1) Make a character move.
2) Gather resources.
3) Create houses.
4) Create units from houses.
5) Select multiple characters or buildings.
6) Create a bridge between two platforms.
7) Make a character move from one platform to another.

A total of three testers were selected by the author for doing the usability testing. Each of them played the game somewhere between 5 and 15 minutes. The players were told that it is an RTS game, but they had no other instructions before starting the game and were expected to try out the game intuitively. The author ensured that they had fulfilled the necessary use case goals. Use cases were not introduced to the players and were only hinted when the player did not know what to do next. The author had to instruct one player how to assign a worker to gather resources with the cursor.

[54] https://en.wikipedia.org/wiki/Software_testing
[55] https://en.wikipedia.org/wiki/Usability_testing

After testing, the testers were asked 8 questions. One of the questions was optional and could be answered as a free-form text and the others had selective answers. See Appendix IV for the questionnaire and Appendix II for the result data.

6.2 Problems and Analysis

The testers did not find the game mechanics and interfaces difficult to learn or use. The hardest part seemed to be the understanding of what they are expected to do in the game. The first thing that a player should do in this game is to select a character, and at the first glance, it did not seem to be intuitive for them. As mentioned in the previous subchapter, one player also needed help with assigning a worker to gather resources. From this point, testers started to understand how to play the game and testing went on without problems and tried to find issues in the game.

Overall, most of the test results are rather positive, especially for the interfaces and the mechanics of the game. According to the testers, it turns out that the game design idea itself is not very clear in the prototype, even though they understand how the mechanics from RTS and Platformer work here. As one of the testers pointed out, the game is missing enemies and a losing condition. This means that there might simply not be enough features or mechanics (e.g. discovering land with a scout or using workers to build a house) that would make understanding the goal of the game design easier.

6.2.1 Found Issues

During the testing, the players found a total of 2 errors in the game. The first error was that sometimes, when a new house was created, newly created units might walk through the air towards to one of the connection points or to a point that is located on another platform (see Figure 13). Although, they started moving only when they were assigned to move somewhere. This problem appeared because the implementation of pathfinding relies on child-parent relationship. The parent platform of a newly created house was sometimes set incorrectly, and the units got the same parent as the house. This error is now fixed.

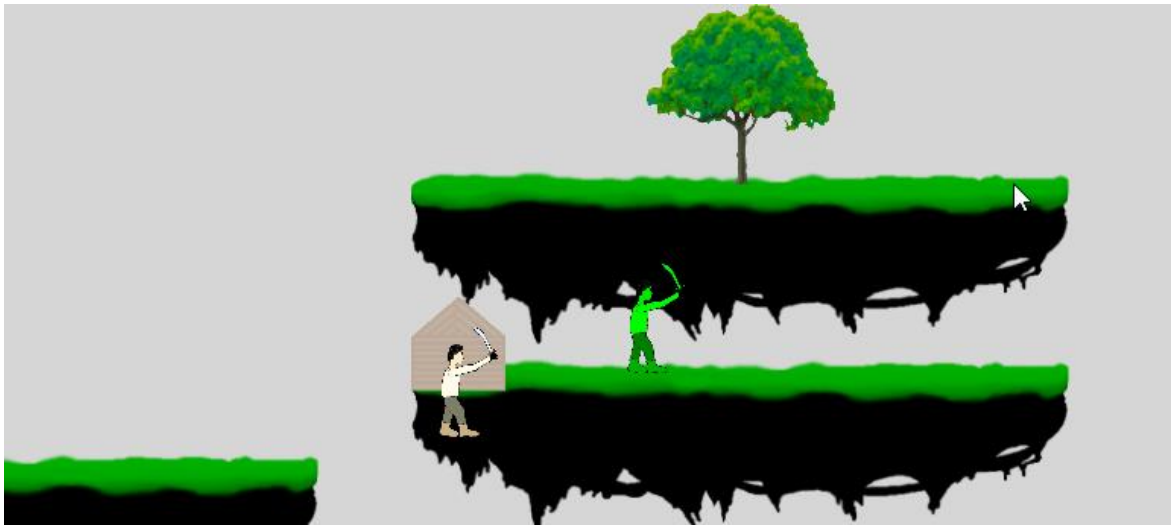

Figure 13. Selected swordsman (right) is moving diagonally from the house to the cursor.

The second error was that the workers could gather resources that were on a different platform, without going there. This only happened if there was no route between the source and destination platform. The reason behind this problem was that the game thought the worker had arrived at the destination where the resource was located, but the target in the path was in the same location where the worker already stood. This error is now fixed too.

Now that the there is some insight about the game from the testers and the issues that were found are now fixed, it is worth mentioning that the work on the game *Queen Aerisilium* will continue after the submission of this thesis. There are still many mechanics and other features to be added in the game by the design document (see Appendix II). Finally, the attributes that will be added to the units and buildings will need balancing, and that includes a lot of additional testing, besides just the usability testing.

7 Conclusion

During the writing of this thesis, a game genre was defined, real-time strategy on platforms (RTSoP), which is a combination of mechanics from the RTS and platformer genres. At the same time, the game Queen Aerisilium was designed and developed, which is based on the previously mentioned RTSoP genre.

For designing *Queen Aerisilium*, different types of game mechanics, defined by Jesse Schell, were explained, that can be potentially used to define any game. Then there have provided the definitions of different game genres, like RTS, platformer, side-scroller, turn-based strategy, and the strategy game genre. Along with the definitions, a short overview of games, with the examples of related mechanics, is provided that are related to both RTS and platformer genres. The development began with selecting the game engine, which resulted in choosing Unity.

This thesis processed the main game design choices, and how they apply specifically to the RTS and platformer genres. Then described the design and implementation of some of the most important mechanics and what kinds of problems appeared during the development. Finally, a usability testing was carried out to collect feedback for the developed prototype game *Queen Aerisilium*. Unfortunately, for one of the testers, the idea of the game design was not quite clear. As the prototype game does not have some of the core mechanics (e.g. see chapter 4.2.2 about the scouts) that are described in the game design document, implementing these should make the design clearer.

The author is planning to continue with the development of *Queen Aerisilium* during his free time and potentially in the Computer Graphics Project courses during his Master's programme. The goal is to finish the game according to the game design document, add multiplayer support, and publish the game.

The author's gratitude goes to his supervisor Raimond-Hendrik Tunnel, for assisting to improve the quality and the completion of this thesis. Thanks to the friends who provided moral and technical support during the development of *Queen Aerisilium* and writing this thesis. And finally, the testers who helped to evaluate the current prototype, and find errors and other deficiencies in the game.

8 References

[1] Moss R. Build, gather, brawl, repeat: The history of real-time strategy games. Ars Technica, 2017. https://arstechnica.com/gaming/2017/09/build-gather-brawl-repeat-the-history-of-real-time-strategy-games/ (19.03.2018)

[2] Jensen K. T. What happened to the real-time strategy genre? Geek.com, 2016. https://www.geek.com/news/what-happened-to-the-real-time-strategy-genre-1649869/ (30.01.2018)

[3] Evans-Thirlwell E. The decline, evolution and future of the RTS. PC Gamer, 2016 http://www.pcgamer.com/the-decline-evolution-and-future-of-the-rts/ (30.01.2018)

[4] Schell J. The Art of Game Design. Elsevier Inc. 2008. http://www.sg4adults.eu/files/art-game-design.pdf

[5] Technopedia, Gameplay. https://www.techopedia.com/definition/1911/gameplay (19.04.2018)

[6] Altice N. I AM ERROR. Cambridge: The MIT Press. 2015.

[7] Greenslade A. Gamespeak: A glossary of Gaming Terms. The Specusp- here, 2006. https://web.archive.org/web/20070219082328/www.specusphere.com/joomla/index.php?option=com_content&task=view&id=232&Itemid=32

[8] de Castell S, Jenson J. Worlds in play. Peter Lang Publishing, Inc. 2007. https://books.google.ee/books?id=WykINIyYSb0C

[9] Technopedia, Side Scroller. https://www.techopedia.com/definition/27153/side-scroller (16.03.2018)

[10] Technopedia, Real-Time Strategy (RTS). https://www.techopedia.com/definition/1923/real-time-strategy-rts (17.03.2018)

[11] Technopedia, Turtling. https://www.techopedia.com/definition/27523/turtling (17.03.2018)

Appendix

I. Glossary

Attack range The maximum or minimum range that the target can be away from a character to be attacked with a specific weapon.	Rünnaku ulatus Maksimaalne või minimaalne kaugus, mis saab olla sihtmärgi ja karakteri vahel, et teda saaks tabada kindla relvaga.
Area of effect[56] A term, that is used to describe actions that influence multiple objects within a specified area.	Mõjuala Mõiste, mida kasutatakse tegevuste kirjeldamiseks, mis mõjutavad mitmeid objekte, mis asuvad määratletud alas.
Cooldown[57] The amount of time that a player needs to wait after an action before being able to do it again.	Jahtumisaeg Aeg, mille möödumist peab mängija ootama peale mõnda tegevust, et seda korrata.
Difficulty[58] A setting in a game that defines how hard it is for a player to play that game.	Raskusaste Säte, mis määrab mängus ära selle, kui raske on mängijal mängida seda mängu.
Feature A distinctive characteristic of a game design element.	Iseärasus (või eriomadus) Mängudisaini element, millel on omapärased tunnusjooned.

[56] https://en.wikipedia.org/wiki/Glossary_of_video_game_terms#area_of_effect
[57] https://en.wikipedia.org/wiki/Glossary_of_video_game_terms#cooldown
[58] https://en.wikipedia.org/wiki/Degree_of_difficulty#In_video_gaming

Game engine[59] A combination of a runtime library and development environment that is designed for creating computer games.	Mängumootor Kombinatsioon käitusteegist ning arenduskeskkonnast, mis on disainitud arvutimängude loomiseks.
Game design[60] (Computer game design) The process of designing a game, along with its contents and mechanics.	Mängudisain (Arvutimängu disain) Mängu disainimise protsess koos selle sisu ning mehaanikatega.
Health[61] (or hit points) An attribute of a unit, that shows the amount of damage that a unit can tolerate.	Elud Üksuse atribuut mis tähistab seda, kui suuri kahjustusi on ta võimeline kannatama.

[59] Kurvet A. Modelling and Game Engines. https://courses.cs.ut.ee/2017/cg/fall/Main/Lectures
[60] https://en.wikipedia.org/wiki/Game_design
[61] https://en.wikipedia.org/wiki/Glossary_of_video_game_terms#health

II. Accompanying Files

The ZIP archive file containing the accompanying files has the following structure:

- */Build* – the folder containing the executable game of *Queen Aerisilium*.
- */GGD/queen_aerisilium_gdd.pdf* – the game design document of *Queen Aerisilium*.
- */Questionnaire/numeric_questionnaire_data.xlsx* – the file containing numerical (questions 1 to 7) data from the answers to the questionnaire.
- */Questionnaire/textual_questionnaire_data_(modified).txt* – the file containing textual (question 8) data from the answers to the questionnaire.
- */Source* – the folder containing the Unity project and source code files of the game.
- */Videos/demo.mp4* – a video demonstration of the gameplay in *Queen Aerisilium*.
- */Videos/starting_the_game.mp4* – a tutorial for starting *Queen Aerisilium* (in Windows).
- *readme.txt* – the file containing short instructions how to run the game.

III. Player Manual

Using mouse:

- Cursor – move the cursor to the edges of the screen to navigate around the scene.
- Right-Click – cancel selection, disable (any) building controller, or set target for selected units.
- Left-Click – select one object by clicking on it or drag the mouse to select multiple objects (same type only), build the selected house or bridge in the cursor's location, or click on a button.
- Mouse Wheel – zoom in in one direction and zoom out in the other direction.

Using keyboard:

- Arrow keys – navigate around the scene.
- B – enable or disable house building controller.
- C – enable or disable bridge building controller.
- Delete – destroys the first object in selected objects' set.
- Escape – goes back into the menu.
- Right Shift + Delete – destroys all selected objects.
- X – selects all units in the game (for testing).

Using keyboard and mouse:

- Shift + Left-Click - Select additional objects without forgetting previously selected objects (same type only).

Playing the game:

- There are buttons in the bottom-left corner of the screen, which indicate the bridge and houses in the game. Clicking on them will activate a controller to build them.
- When a house is selected, a button or multiple buttons appear in the bottom-middle of the screen, which indicates the units that can be bought from that house.
- When a worker (the one with an axe) is selected, one can assign them to gather resources, by clicking on a tree or wheat field.
- After pressing the *Bridge* button, two green circles appear on every platform, these circles indicate where to click to build a bridge.
- When resources have depleted, one cannot build houses, bridges or units.
- There are cheat codes to help with the testing and they can be written while the game is running: “cheese”, “liivi2”, “lumber”.

IV. Questionnaire for Usability Testing

1) How would you rate this game overall?
    1. Bad.
    2. Rather bad.
    3. Rather good.
    4. Good.
2) You felt the need to read the manual.
    1. Yes.
    2. Rather yes.
    3. Rather no.
    4. No.
3) Interfaces and mechanics were easy to learn.
    1. No.
    2. Rather no.
    3. Rather yes.
    4. Yes.
4) Interfaces and mechanics were easy to use.
    1. No.
    2. Rather no.
    3. Rather yes.
    4. Yes.
5) Game design idea of the game was clear.
    1. No.
    2. Rather no.
    3. Rather yes.
    4. Yes.
6) Have you played RTS games before?
    1. No.
    2. Have tried a few times.
    3. Yes.
    4. A lot
7) Have you played platformer games before?
    1. No.
    2. Have tried a few times.
    3. Yes.
    4. A lot.
8) If you encountered any bugs or issues while playing, please describe them here.
    1. Free-form text.

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Silver Kirotar**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Real-Time Strategy on Platforms Game Design,

(title of thesis)

supervised by Raimond-Hendrik Tunnel, MSc,

(supervisor's name)

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **14.05.2018**