UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Daniel Nael

Icarus – a Real-Time Strategy Game in Space

Bachelor's Thesis (9 ECTS)

Supervisor: Raimond-Hendrik Tunnel, MSc

Tartu 2019

Icarus – a Real-Time Strategy Game in Space

Abstract:

The thesis describes the development of a real-time strategy game in space called *Icarus*. It was made to offer variety in its corresponding computer game genre. An overview of its game design and implementation is given. The game design of *Icarus* was created in accordance to the basic principles of the real-time strategy genre and the field of game design. The game was playtested on potential players to find issues and assess its quality. Based on the test results several improvements were made after the playtesting and some were planned for the future development.

Keywords:

Computer game, real-time strategy game, game design, computer graphics, software development, Unity, usability testing, playtesting

CERCS: P170 Computer science, numerical analysis, systems, control

Icarus – reaalaja strateegiamäng kosmoses

Lühikokkuvõte:

Antud lõputöö kirjeldab kosmose reaalaja strateegiamängu Icarus arendust. Arvutimängu loomise eesmärgiks oli pakkuda mitmekesisust vastavas arvutimängužanris. Töös antakse ülevaade Icaruse mängudisainist ja implementatsioonist. Icaruse mängudisain loodi reaalaja strateegiamängu žanri ja mängudisaini põhitõdedele toetudes. Mängu testiti võimalike mängijatega, et leida probleeme ja hinnata kvaliteeti. Testimisel saadud tulemuste alusel on sisse viidud mõned parandused. Osa parandusi on kavandatud teha tulevikus mängu edasiarendusena.

Võtmesõnad:

Arvutimäng, reaalaja strateegiamäng, mängudisain, arvutigraafika, tarkvaraarendus, Unity, kasutatavuse testimine, mängu testimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaat-juhtimisteooria)

Table of Contents

1 Introduction

Computer game development is a highly growing industry that can bring profit to every smaller country[1]. As with literature and cinematography, the field of computer games includes a lot of different genres. One of the older genres in computer games is *real-time strategy* which features games that are complex in terms of implementation and playability[2]. Therefore, the games in this genre require a lot of concentration and time from both game developers and players.

The goal of this bachelor's thesis was to create a real-time strategy game *Icarus* to offer strategy gamers a familiar, enjoyable and uniquely stylized experience. The design of *Icarus* follows the general principles of the corresponding genre and *game design*. Three levels were designed for the created game. Those levels were used to playtest the game and see if it is entertaining for the users.

Icarus is a real-time strategy game in space where a player has a spaceship that is also a big city. At first the Cityship has no ability to move or do anything besides floating still in space. However, just floating is unacceptable because the solar system (the game world of *Icarus*) is dying which causes expanding of dangerous zones that destroy everything in their way. In addition, some computer-controlled space pirates want to destroy the player's Cityship. To overcome all of these *challenges* a player of *Icarus* must build structures on their Cityship and use smaller spaceships to gather *resources* and fight with *enemies*.

Chapter 2 focuses on the background that was necessary to understand for designing *Icarus*. It first provides a definition of the real-time strategy genre and thereby mentioning the games that started the genre. Mainly, it contains descriptions of the games that were similar to and an inspiration for creating *Icarus*. The chapter ends with a comparison of prominent real-time space strategy games and *Icarus*.

Chapters 3 and 4 are focused on the creation of *Icarus*. They include significant parts of its game design and compelling implementation choices. Furthermore, they give an overview about some of the challenges that occurred and were overcome while developing *Icarus*.

[1] Opinion of Estonian minister of culture Indrek Saar
http://www.ituudised.ee/uudised/2018/04/17/minister-eestil-on-eeldused-mangutoostuse-ekspordiks
[2] Real-Time Strategy High-level Planning
https://www.inter-actief.utwente.nl/studiereis/pixel/files/indepth/StefanWeijers.pdf

Chapter 5 describes the testing of *Icarus*. Firstly, it gives a detailed overview about the methodology of the testing. Secondly, the chapter introduces the results from *playtesting* with the suggestions from the testers on how to improve the created computer game. Finally, it covers the enhancements made during the work of this thesis and the aspects of *Icarus* that could be made better in the future.

Some of the terms used in this thesis are defined in the Glossary (Appendix I). The launch guide is in Appendix II and a short game guide in Appendix III. Source code and asset files are accessible in the repository (Appendix IV) and in Accompanying Files (Appendix VI). The game itself is also in Accompanying Files (Appendix VI). A video *(demo.mp4)*, that demonstrates the gameplay of *Icarus*, is in Accompanying Files (Appendix VI). Illustrations 1 and 2 show how *Icarus* looks after the development for this thesis.

Illustration 1. The main menu of *Icarus*

Illustration 2. In game screenshot of *Icarus*

2 Background

To understand *Icarus* and its design, it is relevant to understand what real-time strategy as a game genre means. Therefore, subchapter 2.1 defines real-time strategy game based on the first prominent real-time strategy games from the early years of the genre. Subchapter 2.2 features real-time strategy games that inspired *Icarus* and in subchapter 2.3 similar games to *Icarus* are compared with it.

2.1 Defining Real-Time Strategy

There are many real-time strategy games. Each real-time strategy game has its own design and gameplay *mechanics*. Naturally most of them feature at least some of the basic *elements* common to the real-time strategy genre.

The most important element for a real-time strategy game is the fact that the game advances in real time [1]. This means that all actions, that a player or *artificial intelligence* (computer) makes, happen continuously during a game session. That idea was introduced by the first game classified as a real-time strategy game in 1992 – *Dune II* (Illustration 3) [2]. Most of the other strategy games before *Dune II* advanced incrementally in turns – *turn based strategy* games[3].

Illustration 3. *Dune II*

Although, the computer game known as the first real-time strategy game is *Dune II*, there were some games that resembled real-time strategy games. One of them is a computer game *Herzog Zwei* from 1989. It gives its player direct control over a *unit* that has three

[3] *League of Legends* Couldn't Exist Without *Dune II*
https://motherboard.vice.com/en_us/article/vv7gad/league-of-legends-couldnt-exist-without-dune-ii

functionalities: attacking enemy units, transporting the player's stationary units and moving them in the game world[4]. Although it is not a real-time strategy game, it features another key element of this genre – a unit.

Usually a player of a real-time strategy game can control more than one unit. Units are game objects that can be directly interacted with. They are used to realize three main gameplay objectives of a real-time strategy game:

- building and/or managing a base,
- gathering resources,
- fighting with enemies in real time to dominate over an area in the game world [2].

These real-time strategy game elements mentioned in this subchapter define a real-time strategy game and are featured in *Icarus*. In the history of real-time strategy there are many games that brought new features to the genre. The design and implementation of *Icarus* was inspired by some popular real-time strategy games on the market.

2.2 The Inspiration

Icarus is a real-time strategy game in space. Its game design is inspired by other real-time strategy games, but more influential ones are *Age of Mythology* and *Homeworld*. In addition, *Icarus* was inspired by other science fiction medium.

Illustration 4. *Age of Mythology Extended Edition*

[4] *Herzog Zwei* Is the Best Real-Time Strategy Game You Never Played
https://kotaku.com/herzog-zwei-is-the-best-real-time-strategy-game-you-nev-1832626883

Age of Mythology[5] is a real-time strategy game from 2003 but was released again in 2014 with improved graphics and new content under the title of *Age of Mythology: Extended Edition*[6] (Illustration 5). It gives the player an opportunity to experience the main objectives of real-time strategy (that were mentioned in subchapter 2.1) in a setting of ancient civilizations like Norse, Egyptian or Greek where mythical creatures and gods are real.

Homeworld[7] (1999) and *Homeworld* 2[8] (2003) are real-time strategy games in space that were also remastered with modern graphics and new content in 2015 in the *Homeworld Remastered Collection*[9] (Illustration 5). *Homeworld* franchise does not contain implicit *base building*, instead it gives the player a mothership unit that contains sections where buildings can be built on. *Homeworld* games encourage players to build big fleet of spaceships that are mainly used for resource gathering and fighting. One of the most notable features of *Homeworld* games is the possibility to control a *floating camera*. This differs from having a top-down view (using *movable camera*) of the game world like in most of the real-time strategy games on land (including *Age of Mythology*). One major highlight in *Homeworld* games is the fact that units do not move only in one 2D plane but can move to other spatially parallel planes at different heights.

Illustration 5. *Homeworld Remastered Collection*

[5] *Review of Age of Mythology: Extended Edition*
https://www.pcgamer.com/age-of-mythology-extended-edition-review/
[6] *Age of Mythology: Extended Edition*
https://store.steampowered.com/app/266840/Age_of_Mythology_Extended_Edition/
[7] Review of *Homeworld*
https://www.gamespot.com/reviews/homeworld-review/1900-2537718/
[8] Review of *Homeworld 2*
https://www.gamespot.com/reviews/homeworld-2-review/1900-6075223/
[9] *Homeworld Remastered Collection*
https://store.steampowered.com/app/244160/Homeworld_Remastered_Collection/

One should get inspiration from everywhere when designing a game [3]. *Icarus* got its inspiration for having a Cityship from a science-fiction television show *Stargate: Atlantis*[10]. The show included a big city-like ship named Atlantis that was discovered by an international space team from Earth. Using the idea of a real-time strategy game in space from the *Homeworld* franchise and combining it with the ancient setting of *Age of Mythology* with some influences from *Stargate: Atlantis* television show, the idea of *Icarus* was created. That is why *Icarus* has a space and ancient Greek themed art style (Illustration 6).

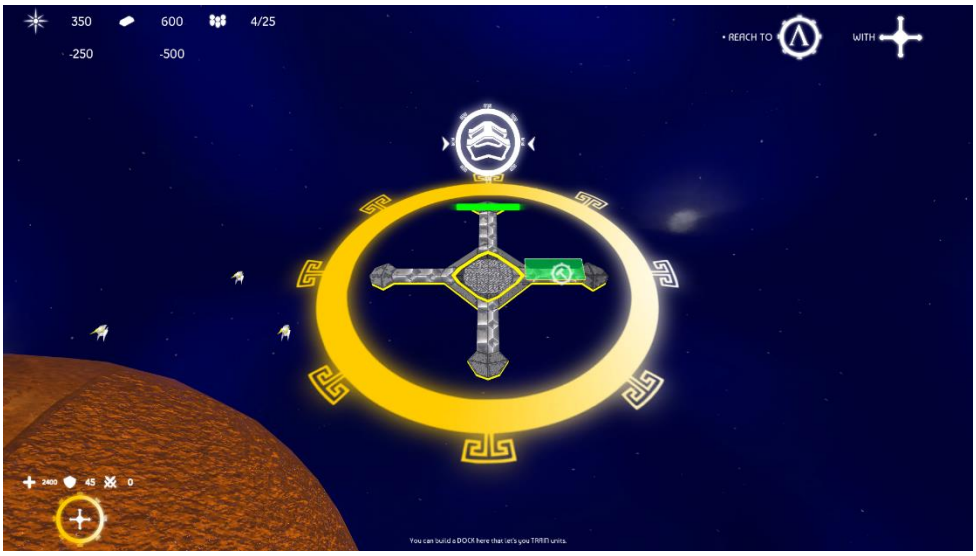

Illustration 6. *Icarus*

There are also some more inspirations from other games that are mentioned in later chapters where the game design choices of *Icarus* are described more in depth. *Icarus* is a real-time space strategy game and thus it is more logical to compare its elements with other real-time strategy games in space rather than those that take place on land.

2.3 The Comparison

The most important computer games to compare to *Icarus* are games from the *Homeworld Remastered Collection* because they belong to the same genre as *Icarus* and are also an inspiration for *Icarus*. Remastered versions of *Homeworld* were chosen because they are topical due to the fact that they were released in 2015 (the original games were released in 1999 and 2003). Although being an inspiration for *Icarus*, these games differ from *Icarus* in a number of important ways.

[10] *Stargate: Atlantis*
https://www.imdb.com/title/tt0374455/

Firstly, the camera system in *Icarus* is significantly different. *Homeworld*, as explained in subchapter 2.2, features a 3D camera movement. That means the camera has a freeform view around the game objects whereas in *Icarus* the movable camera has always a top-down view. *Icarus* and *Homeworld* games allow the player to move and zoom in/out the camera but *Icarus* does that more in a constrained way that is common to most other real-time strategy games. The reason behind this is to complement *Icarus'* simplistic visual style. These constraints could be more familiar for real-time strategy players because this is featured commonly in the games of this genre.

In the *Homeworld* franchise units can move in different lower or higher 2D planes. In *Icarus*, however, units can move only on a single 2D plane. This design choice was also made to pursue a more simplistic style of controls in a real-time space strategy.

Both *Icarus* and *Homeworld* feature a map that the player can use to see the size of the game world and to locate game objects like units and resources. *Homeworld* games do not show the location of enemy units to the player if they are not in the visibility range of the player's units. However, in *Icarus* all the game objects that exist are shown to the player because it is likely overwhelming to avoid the expanding dangerous zones and fight with enemies at the same time without always knowing where they are located.

Homeworld games and *Icarus* are similar because of their space theme. However, *Icarus* has also an ancient Greek theme influence while *Homeworld* games have a space sci-fi style. This results in different visual aesthetics in both games.

The main unit in *Icarus* is the Cityship which must be defended from the enemies. Homeworld franchise also features a central unit that needs protection. As with *Icarus*, it allows the player to build on it. However, the building process and structures differ.

There are some more minor distinctions and affinities between the *Homeworld* games and *Icarus*, but the major ones were introduced in this subchapter. *Homeworld* games and *Icarus* feature some notable similarities and differences. As explained earlier, this is due to the fact that *Homeworld* franchise has an influence on *Icarus'* game design.

3 The Game Design

Icarus was designed via an *iterative design* process as Zimmerman and Salen [4] defined it. The basic idea of that process is to make design decisions when the game is still in development based on results from playtesting and prototyping by the developer itself. Through this, the game gets its basic mechanics. There are principles of game design that commonly almost every game follows [3, 4]. To offer valuable play time for the player *Icarus* is designed to pursue some of the fundamentals of game design. Subchapter 3.1 explains the story behind *Icarus* and how it got its title. Subchapter 3.2 contains the overview of the basic game mechanics and how the player can use them to overcome the challenges in the game. Lastly, subchapter 3.3 gives a synopsis of the *level design* of *Icarus*.

3.1 The Story Behind Icarus

In Greek mythology[11] Icarus was a young man who was given wings by his father Daedalus. The wings had feathers which were fixed with wax. Daedalus taught Icarus to fly and asked him to be cautious, neither to fly too low nor too high to avoid making the feathers wet of sea water or causing the wax to melt by the sun. Icarus forgot all his father's words and flew too high which resulted in the melting of the wax and the loss of his wings. After that Icarus fell into a sea and drowned.

In the game *Icarus* the player resembles both Daedalus and Icarus. The player has a main spaceship that is also a big city floating in space. It lacks for engines for moving but because of the surrounding dying solar system it is essential to have this ability. So, the player must develop required technologies to make the Cityship move. They must control it later not to fly too near to dangerous areas that damage and eventually destroy the ship. Additionally, there are space pirates that want to destroy the Cityship. Therefore, the player must also build weapons on the Cityship and create combat-oriented Battleships to protect it while escaping from the dying solar system. For building new spaceships and improving the Cityship, the player needs to collect resources from space.

The challenges and objectives mentioned in the previous paragraph create a state of emergency for the player. This kind of crisis management is one of the appeals in real-time

[11] Story of Icarus
https://www.greekmythology.com/Myths/Mortals/Icarus/icarus.html

strategy games [1]. The player can overcome these challenges by developing a strategy with the knowledge of how the game mechanics of *Icarus* work.

3.2 Game Mechanics

According to Schell [3], mechanics of a game consist of its rules and procedures that characterize game's goal. Due to these aspects a player can understand the main goal and how is it achievable. Subchapter 3.1 briefly mentioned the challenges of objectives of *Icarus,* but in this subchapter they are explained more rigorously.

3.2.1 Challenges

Without challenges and goals there would not be almost any gameplay [3]. *Icarus* has two main challenges that the player must overcome. The first challenge – a dangerous zone (Illustration 7) is a red circular area that expands over time. All units – including enemies – are damaged over time (lose 10% of their maximum health value in one second) while being in this area. There are at least two dangerous zones per level in *Icarus*.

Illustration 7. A dangerous zone

The idea of shrinking safe zone is also featured in other computer games. For example, a *tower defense* real time-strategy game *Creeper World*[12] is all about fighting with dangerous area that grows across the game map. Although, in *Icarus* it is not possible to fight with dangerous zones. The player can only avoid them.

[12] *Creeper World: Anniversary Edition*
https://store.steampowered.com/app/422910/Creeper_World_Anniversary_Edition/

The dangerous zone mechanic in *Icarus* is more similar to the safe zone mechanic featured in computer games of the *battle royale*[13] genre. Most of the battle royale games feature a shrinking safe zone and the players must be inside it. *Homeworld Remastered Collection* also features a similar idea to *Icarus'* dangerous zones. There is a level where the game world contains areas that have radiation clouds. As with *Icarus*, all units are damaged over time when they are located in these radiated areas. Although, these radiated areas do not expand over time like dangerous zones in *Icarus*.

The "Supernova"[14] level from a popular real-time strategy game *Starcraft II*[15] also features similar element to dangerous zones. There is a wall of fire that expands over time from one side of the game world and destroys everything it touches. In general, there are real-time strategy games that include levels with similar game mechanics to *Icarus'* dangerous zones, but this is not used as a *core mechanic* like in *Icarus*.

The second challenge is the fact that space pirates attack the player's Cityship in iterations over time. Space pirates have the same warship unit types as the player (subchapter 3.2.2). With each new iteration an additional warship is spawned. This requires the player to make decisions and train new units continuously during the gameplay.

3.2.2 Units

As mentioned in subchapter 2.1 the units and their management is important in a real-time strategy game. *Icarus* features four types of units that have their own gameplay purpose. All the units can be given commands and have a health value, a defense value and the ability to move (excluding the Cityship in the beginning of any level of *Icarus*) in the game world. The most important unit is the Cityship (Illustration 8) of which there can be only one of (per player) in a level.

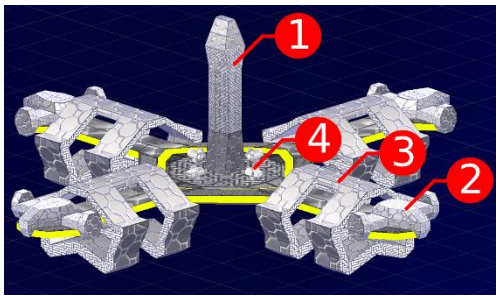

Illustration 8. A model of the Cityship with buildings

[13] The History of Battle Royale: From Mod to Worldwide Phenomenon
https://www.digitaltrends.com/gaming/history-of-battle-royale-games/
[14] "Supernova" level from *Starcraft II*
https://liquipedia.net/starcraft2/Campaign/Supernova
[15] *Starcraft II*
https://starcraft2.com/en-us/

The Cityship is the main character of *Icarus*. The player must defend it from dangerous zones and space pirates. Losing the Cityship will trigger the game's lose condition. To win the game, the Cityship must be moved to a location marked as finish.

In *Icarus* the real-time strategy game objective of base building is realized using the Cityship. The player can build four types of structures on it. The first structure is a Tower (Illustration 8, no 1) that gives the Cityship the ability to move but very slowly. To make the Cityship faster an Engine (Illustration 8, no 2) must be built. Due to the space pirates it is also possible to build weapons on the Cityship (Illustration 8, no 4) that attack the enemies. Finally, to build new units and increase the population capacity, building a Dock (Illustration 8, no 3) is needed. Building structures or other units requires resources that can be collected with units called Workers.

In *Icarus* there are two types of resources: solar power and metal. They are located throughout the game world. Creating a new unit or a building costs a different amount of solar power and metal for every unit. Therefore, the player must use Workers (Illustration 9) to collect more resources, but these units can not defend themselves from attacking enemies. Workers realize the gathering resources objective of real-time strategy games.

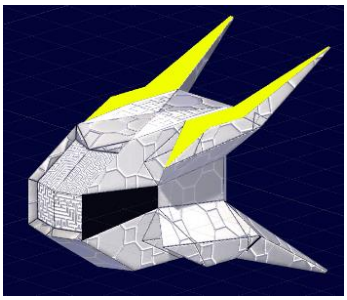

Illustration 9. A model of a Worker

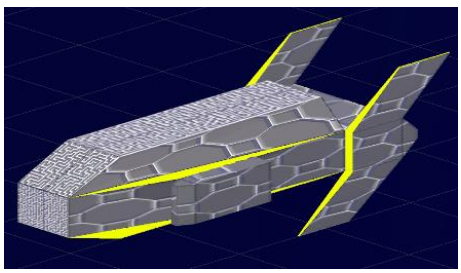

Illustration 10. A model of a Battleship

The unit that can defend itself from enemies is named Battleship (Illustration 10). It is an efficient unit for fighting with pirates. However, it is not a fast ship and therefore it could happen that the unit can not move to a location quickly enough when the player needs it.

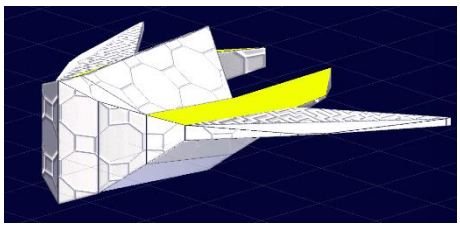

Illustration 11. A model of a Fighter

For traveling long distances with high speed a Fighter unit can be used (Illustration 11). It is a fast unit that also is meant for battling with space pirates. It is not that strong as a Battleship and can be destroyed easily by enemy Battleships. So, the player must decide what kind of unit is needed during the gameplay.

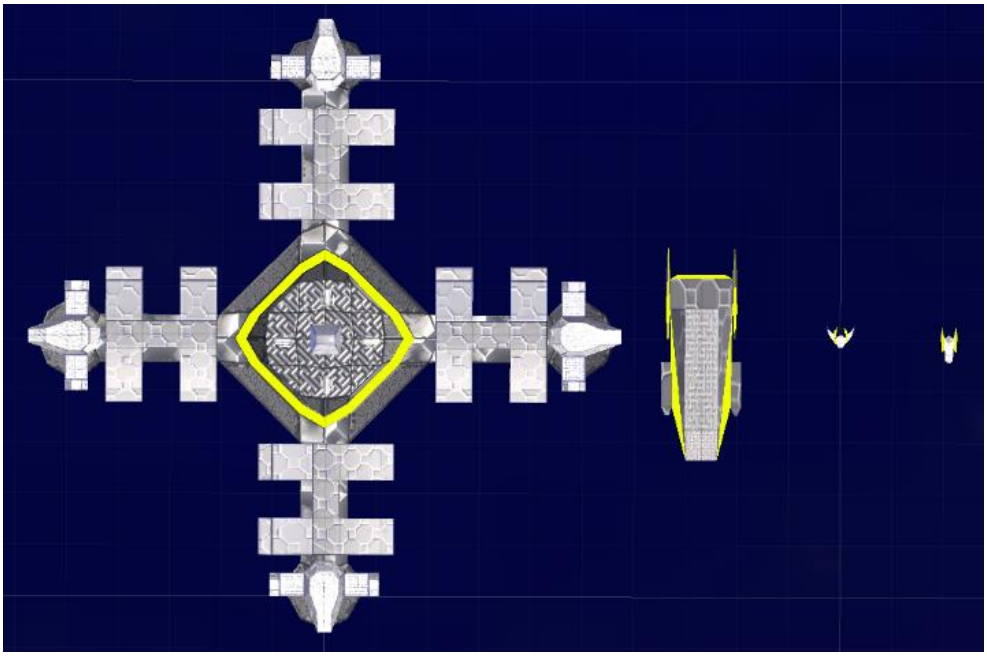

Illustration 12. Size comparison of all units of *Icarus*: (from left) Cityship, Battleship, Fighter and Worker

Units (Illustration 12) are fundamental elements of *Icarus*. Controlling them is the main action that a player does during the gameplay. Hence, it is crucial to have levels in the game that allow the player to effectively use all the units.

3.3 Level Design

Designing game levels is not a trivial exercise because it requires a lot of careful thinking [3]. For this thesis only three levels were made. They were designed to have an ascending difficulty curve. Each level introduces new gameplay elements to the player. The **first level** (Illustration 13) of *Icarus* is a tutorial level. It is made to familiarize the player with the game controls and mechanics gradually. Therefore, it is relevant not to give the player a lot of challenges in this level. This means that the player can have time to understand how the game works.

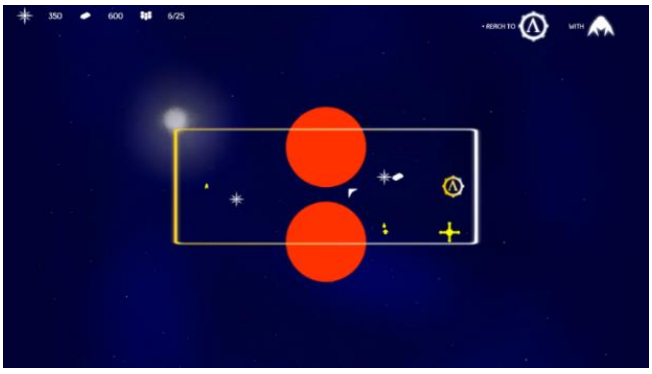

Illustration 13. Map of the first level

After the player somewhat understands the basic elements of *Icarus*, the **second level** (Illustration 14) features only the first challenge of *Icarus* – dangerous zones. There are four dangerous zones across the game world that expand in iterations. One iteration of a dangerous zone consists of being idle for 20 seconds and then expanding also for 20 seconds. The values for expanding and idling times were determined empirically.

In the beginning of the second level it is important for the player to gather resources to build structures on the Cityship. These structures give the Cityship abilities that were mentioned in subchapter 3.2.2. For that the player has initially three Workers and enough resources to build a Dock on the Cityship. The player is not limited to building only a Dock. On the contrary, it is possible to create all types of structures if the player has the required amount of resources. Hence, the player must know and plan what and when to build because they have to reach to the finish before the dangerous zones destroy the game world.

Illustration 14. Map of the second level

In the **third level** (Illustration 15) there are also four dangerous zones and their locations are identical to the locations of the dangerous zones in the second level. The time of expanding and the idling time of dangerous zones is now 125 seconds. From one perspective it gives the player more time to collect resources and build, but on the other hand the player must be more careful because in one iteration the dangerous zones expand more than in the previous level. In addition, the third level introduces the second challenge – space pirates. They spawn in iterations near to the finish location and with each iteration an additional enemy unit is spawned. However, the maximum number of concurrent enemy units per level is 80.

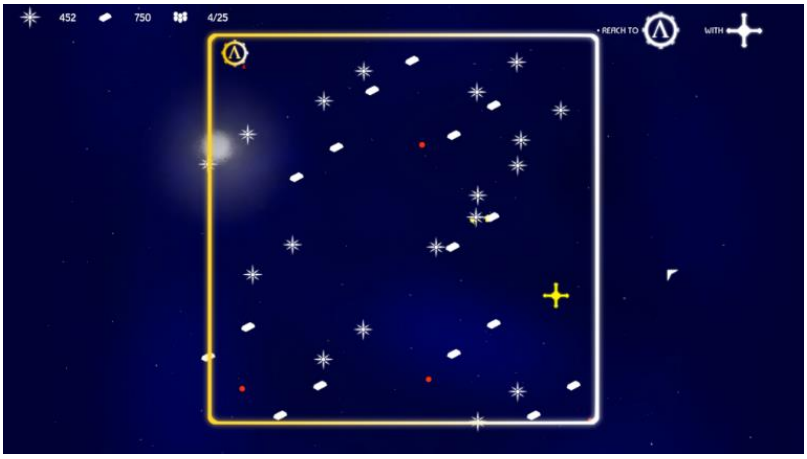

Illustration 15. Map of the third level

In general, the level design of *Icarus* is made to provide a challenge while keeping in mind not to overwhelm the player. A good game achieves a flow where the player's skills and challenges are in balance [3]. Because the level design was created through iterative design, it is indispensable to test if it works on actual players. Before that, however, it is important to understand the key parts of the implementation of *Icarus*.

4 The Implementation

To create *Icarus*, game design was very important and to realize it, technologies and implementation parts are necessary. Subchapter 4.1 briefly mentions significant technologies and applications used to create *Icarus*. Subchapters 4.2 and 4.3 describe the most important systems that were implemented in *Icarus*.

4.1 Technologies

According to Schell [3] the term of technology can have different meanings in computer game development. It is essential to know the difference between *foundational* and *decorational* technology. Foundational technology contains everything that is indispensable for a game. Therefore, decorational technology indicates everything that improves the development and/or the experience of a game.

One of the foundational technologies of *Icarus* is the Unity[16] game engine because *Icarus* was created with it and uses its fundamental systems like handling user input and 3D graphics *rendering*. The reason behind choosing Unity was to concentrate more on game design rather than recreating existing technologies. In addition, Unity features multiplatform development that gives the opportunity to build *Icarus* also for example as a mobile application in the future.

Alternatives for using Unity game engine were Godot[17] and Unreal Engine 4[18]. However, using these engines would have required more time on learning their systems. The author of this thesis was more familiar with Unity game engine.

Most of the art assets in *Icarus* were created during the development. These assets include the textures and models for the units and the graphical user interface images. For that 3D modelling was done with a 3D modelling software Blender[19], 2D images and textures were drawn with a *bitmap graphics* editor Gimp[20] and a *vector graphics* editor Inkscape[21].

The scripting language to implement the logic behind *Icarus* was C#. The object-oriented programming paradigm in C# allowed to write code that follows useful *software design patters* in software development. These patterns made it easier to follow the iterative design

[16] Unity https://unity.com/
[17] Godot https://godotengine.org/
[18] Unreal Engine 4 https://www.unrealengine.com/en-US/
[19] Blender https://www.blender.org/
[20] Gimp https://www.gimp.org/
[21] Inkscape https://inkscape.org/

process because it was easy to implement new functionality and remove unnecessary features. For example, the entire unit system of *Icarus* relies on the *component pattern.*

4.2 Components of a Unit

One of the hardest tasks of implementation of *Icarus* was to create a consistent system for units. This is because there are different types of units with different tasks, but they also need to have identical functionalities as well. Because of that the component pattern was used. Its main idea is that an object can have numerous individual components that need not to be connected but can still send information to each other [5]. Although Unity's own fundamental `GameObject` class uses the component pattern, it is possible to create entire unit's system without using it. However, component pattern was used in *Icarus'* unit's system for the aim of clarity and universality of the code.

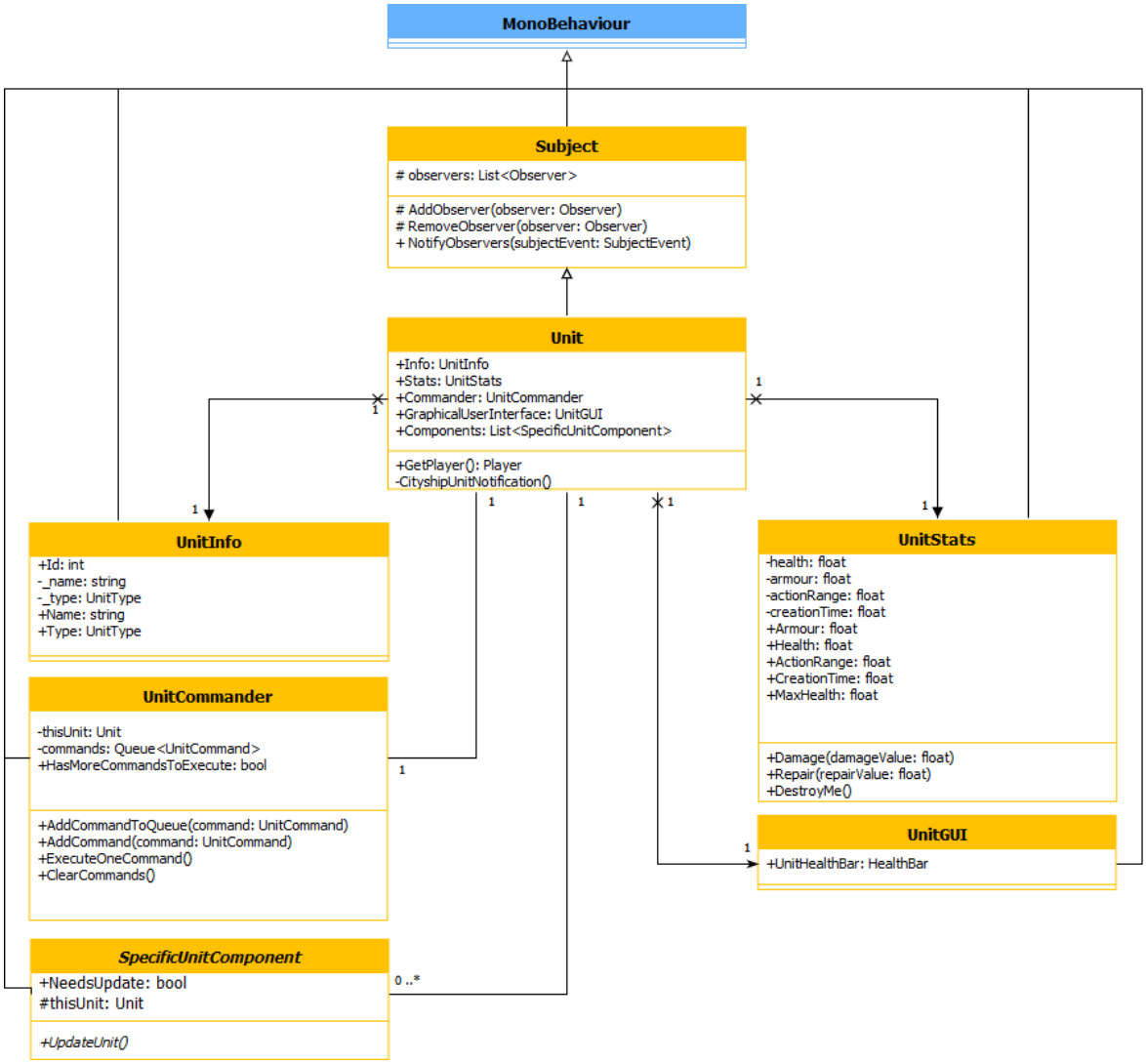

Illustration 16. UML diagram of the component system of a unit

In *Icarus* all units have components (Illustration 16). In general, each unit must have initially `Unit`, `UnitInfo` and `UnitStats` components. The `Unit` class initializes all the other components that a specific unit needs by adding them to the unit's game object. For example, each unit needs its own graphical user interface that is controlled by the `UnitGUI` class.

To access these newly added components the `Unit` component holds their references. Due to that it is not required to search a component if it is needed because it can be accessed using the unit's `Unit` component. This also means it is a mediator from *mediator pattern* that is used for allowing other components to communicate with each other [6].

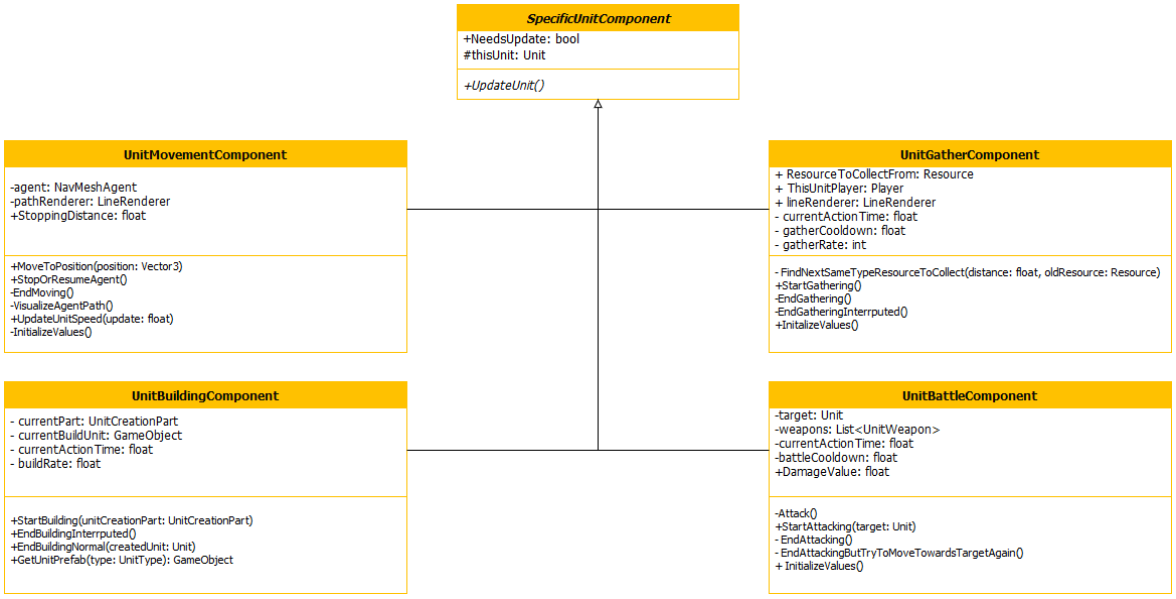

Illustration 17. UML diagram of specific unit components

The `Unit` class includes a list of `SpecificUnitComponent` objects (Illustration 17). These are the components that mostly realize the main actions of the unit in the game world. For instance, all spaceships have a `UnitMovementComponent` that allow their movement in the game world, Battleships have additionally a `UnitBattleComponent` for fighting with enemies etc. These components only feature the logic to perform specific tasks that are started when the player gives according commands.

4.3 Commands of a Unit

The player can give commands to all units in *Icarus*. For this gameplay requirement the *command pattern* was used. Straightforward explanation for commands in the command pattern would be that they are instances that call methods to realize specific actions [5]. For

example, in *Icarus* giving a movement command for a unit creates a new instance of the `MoveCommand` with required values such as position to move to and a unit to execute the command on. Then it is executed on the unit that got this command and unit can start moving to the position it was ordered to move using its movement component. The basic flow of executing a single command can be seen on Illustration 18.

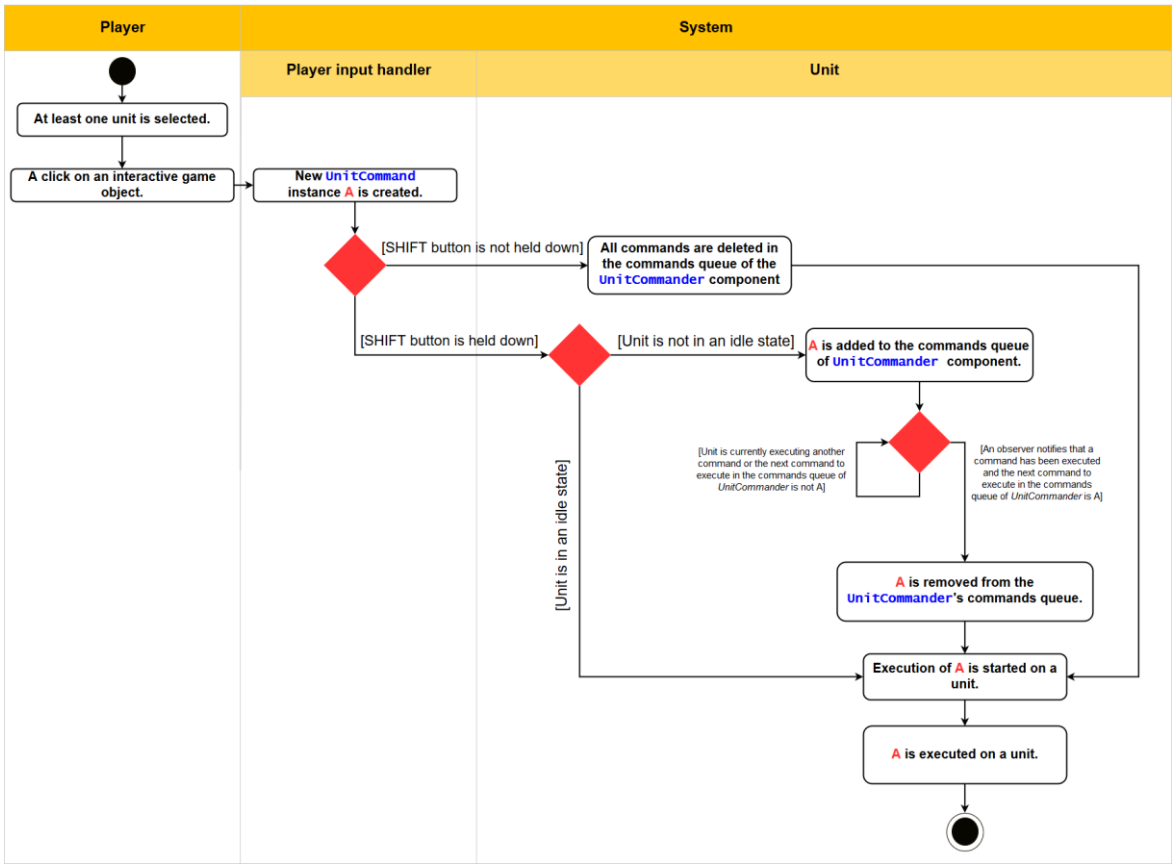

Illustration 18. The flow of executing a command

Total Annihilation from 1997 is a real-time strategy game that is a notable game because it gave an opportunity for the player to give units multiple commands, which were executed in the given order [1]. *Icarus* also features giving multiple commands to units. It is realized with a queue where, as a result of using the command pattern, all the commands can be inputted. New command in the queue will be executed when unit's current task is completed.

To know exactly when the next command can be executed, the *observer pattern* was used. The observer pattern consists of two main elements: an observer and a subject. The subject has a list of observers that are notified when an event happens and act accordingly to this event [5]. Every unit in *Icarus* is a subject that has observers (Illustration 19). In the computer game observers are used also to update the graphical user interface, notify when different actions happen with a unit etc. So, when a unit completed a task then an observer

for commands is notified and if there is a next command in the command queue of the unit, it is executed as well.

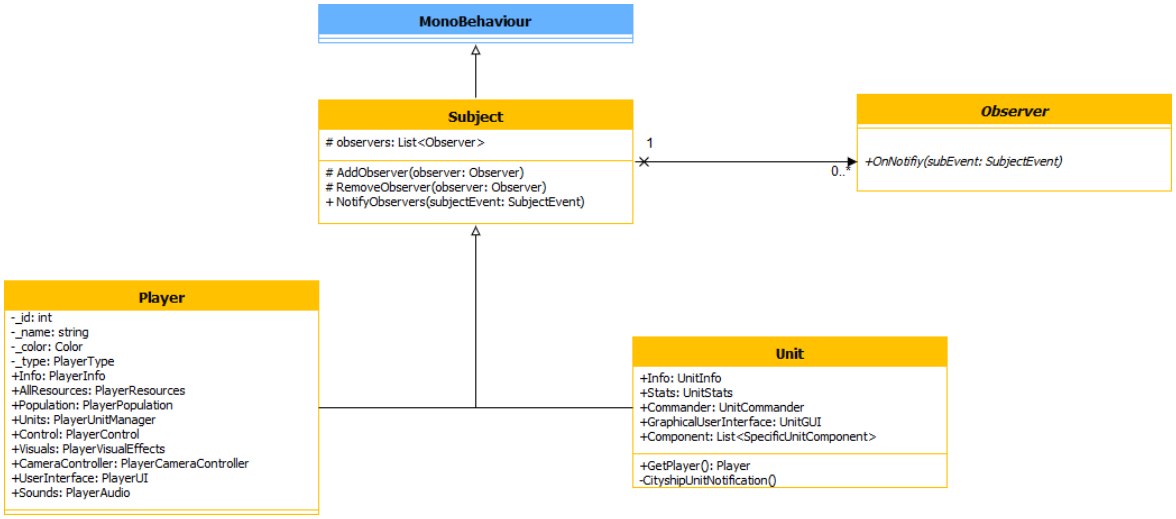

Illustration 19. UML diagram of observer pattern in *Icarus*

Using the component, mediator, command, and observer patterns made it possible to create a consistent fundamental system for units that could be expanded relatively easily. All the other smaller systems like unit's building, fighting, movement rely on this fundamental system. Therefore, although these are important for *Icarus*, their implementation is fairly straightforward. Hence, it is not conveyed in this thesis that how they were implemented. However, it is essential to understand if the implementation of *Icarus* works on an actual player. To assess this, *Icarus* was tested on potential players.

5 The Testing

Testing is important for all software. To get feedback if the design choices and implementation of *Icarus* work, the game was tested on potential users. Subchapter 5.1 describes the methodology of the testing of *Icarus*. Analysis of the feedback from the testers is featured in subchapter 5.2 and how *Icarus* was accordingly improved, and will be in the future, in subchapter 5.3.

5.1 Methodology

Playtesting is crucial because it notifies the developer about all the problems that a computer game has [3]. *Icarus* was playtested on five people because of that. This is enough to find the majority of the usability issues [7]. Part of the testing focused on two usability principles [8]. Namely, it was assessed how the player of *Icarus* "understood" its user interface and how "satisfied" they were using it because these are important aspects of a computer game.

All the testers had an individual session, where they played *Icarus* without a significant knowledge of the computer game. They played it while they were observed how they play and interact with the game. The gameplay video with testers voice from the sessions (Appendix V) were also recorded and analysed more thoroughly after sessions. During the gameplay, the testers were not assisted externally and they had to figure out themselves, how to complete the three levels of *Icarus*. In addition, the testers also had to complete a questionnaire after the game. It featured questions about the background of the tester, overall experience of the gameplay, game mechanics, user interface and visual style. The results helped to get an understanding what design choices worked and what should be improved.

5.2 Results

As previously stated, the questions for the testers covered several topics about *Icarus*. Subchapter 5.2.1 describes how familiar were the testers with computer games and real-time strategy games. Subchapters 5.2.2 to 5.2.7 examine the feedback given by the testers.

5.2.1 The Tester

According to the feedback, the five testers of *Icarus* play computer games at least once in week. The most popular genres played among the testers are action/adventure, strategy and role-playing games. They also play real-time strategy games like *Starcraft II*, *Cossacks*, *Age*

of Empires, *Trash*, *Stronghold Crusader*, *Men of War 2*, *Age of Mythology* and *Warcraft 3*. Therefore, the experience the testers got while playing *Icarus* could point out major shortcomings of the basic real-time strategy elements in the game.

5.2.2 The Overall Experience

Regarding the gameplay experience in terms of enjoyment, the testers were asked to give a rating on a linear scale from 1 to 6 where 1 meant "awful" and 6 "perfect". In addition, they were asked to explain their rating and to bring out their favourite and least favourite moments from the gameplay experience.

The feedback was mostly positive but not perfect. The average score for the overall gameplay experience was 4.2 out of 6 (Illustration 20). Some of the testers felt that *Icarus'* dangerous zone mechanic made the game "fun". On the other hand, two testers mentioned that the game became "uninteresting" when the dangerous zones had covered the solar power and metal resources. The testers could not build new buildings without the required resources. For example, they could not give the Cityship the ability to move or make it faster. This problem occurred because players did not understand what kind of decisions to make in the beginning. The game became "fun" again when the players understood how it works and what it wants from the player.

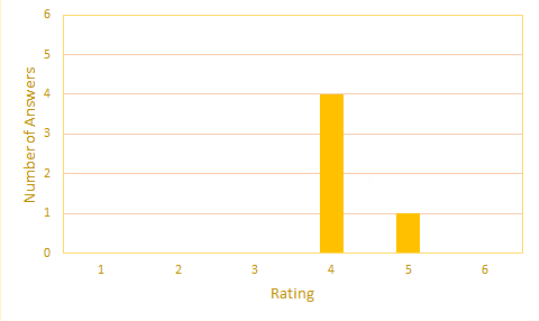

Illustration 20. The ratings of the overall experience of the gameplay

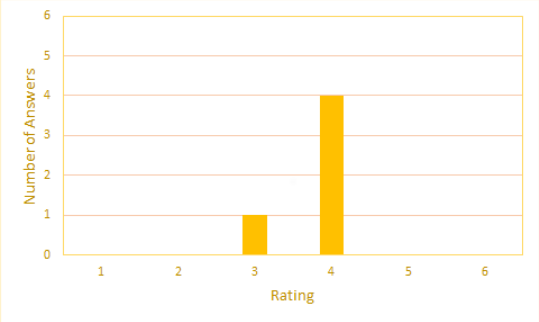

Illustration 21. The ratings of the difficulty of the gameplay

The confusion also made the game difficult at first for the testers. Though, after getting used to the game it became somewhat easy for them. Still, for some of the testers there were some technical aspects that annoyed them. For example, it would have been better if the game featured *hotkeys*, that could have been used to do some actions (like making new units) quicker. To assess the difficulty of the gameplay, the players had to give a rating on a scale

of 1 (“impossible”) to 6 (“very easy”) (Illustration 21). Additionally, they were asked to explain their rating. The average score of the difficulty of the gameplay was 3.8.

The feedback for the overall gameplay experience was mostly positive. Although, there were some shortcomings. To understand more about the weak points of *Icarus*, the testers were asked to individually assess each of the core mechanics.

5.2.3 Game Mechanics

The core mechanics assessed during testing were moving units, creating new units, building on the Cityship, gathering resources, avoiding dangerous zones, fighting with enemies and the map feature. Testers had to rate each mechanic on a linear scale of 1 (“awful”) to 6 (“perfect”). Lastly, they chose their favourite and least favourite game mechanic and suggested some improvements.

For the game mechanic of moving units in *Icarus* the ratings were mixed (Illustration 22). The average score of moving units was 3.8 out of 6 which notifies that there are some problems. One of the issues was with the pathfinding of units. It was not logical for some of the testers how a unit chose its path to move to their desired destinations. Also, when giving multiple commands to move to multiple points, the units made a short stop at each of the locations.

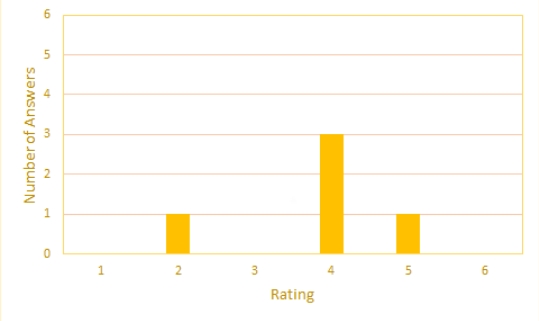

Illustration 22. The ratings of moving units mechanic

In terms of creating new units the average rating was 4.4 out of 6 (Illustration 23). That means this mechanic could be improved. One problem stood out that it was uncomfortable to create new units by clicking with a mouse. As mentioned in subchapter 5.2.2 there should be some hotkeys that let create new units. This would make creating new units easier and faster for the player.

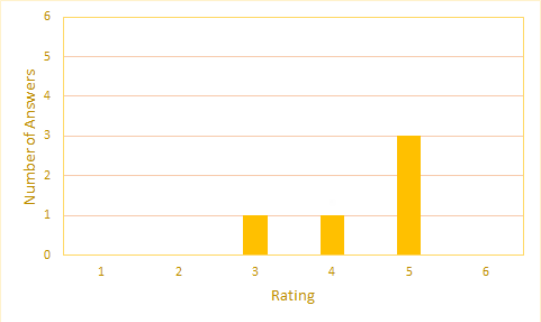

Illustration 23. The ratings of creating units mechanic

Building on the Cityship had a variety of ratings (Illustration 24) and an average score of 3.6 out of 6. Some users also experienced difficulties with it. It did not feel natural for the testers to build on the Cityship using the right mouse button. It took some time for some of

the testers to figure out how building on the Cityship works. After realizing how to build on Cityship, it still felt "clunky" to select a specific part where to build a structure. One suggestion was again to assign hotkeys to make it easier.

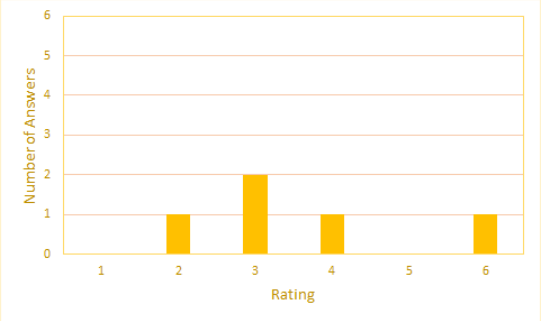

Illustration 24. The ratings of building on the Cityship mechanic

The gathering resources mechanic got an average rating of 4.2 out of 6 (Illustration 25). This means that it had issues too. The main reason behind this score could be that the gathering system in *Icarus* had a bug that was found during the playtesting. It resulted in the fact that units did not start collecting resources when a player ordered them to do it. In addition, some of the players suggested that the resource objects in the game world should show a value of how many resources are left in them. Although, most of them understood eventually that the main indicator for the resource value was the object's size (in the game world). One suggestion was that the player should be notified if a resource is depleted and Workers do not collect it anymore.

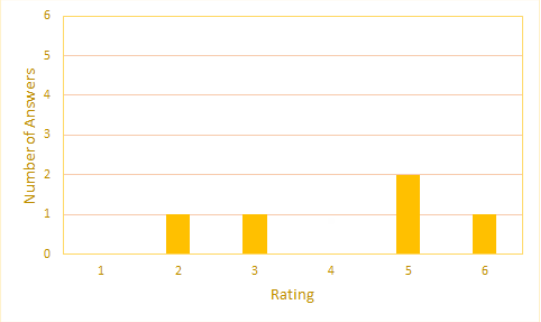

Illustration 25. The gathering resources mechanic ratings

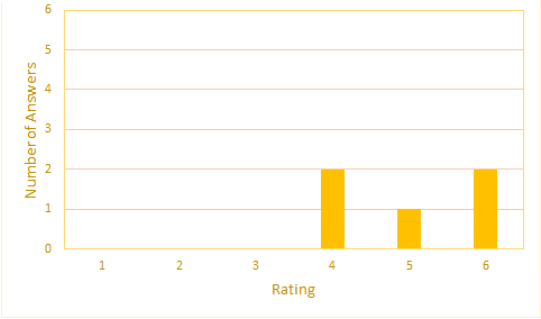

Illustration 26. The ratings of the dangerous zone mechanic

Dangerous zone mechanic was the most favorite mechanic among most of the testers (Illustration 26). With a rating of 5 out of 6 it still needs some more "polishing". However, the testers did not point out specific problems with dangerous zones. One problem could be that some of the testers did not like when their ships moved in to the dangerous zones via automatic pathfinding and they had to avoid these areas manually.

The game mechanic of fighting with enemies in *Icarus* is the most problematic (Illustration 27). An average score 2.6 out of 6 confirms it. Firstly, the enemies flew in to the dangerous zones, therefore committing suicide. Only one of the testers liked it because then he could

concentrate more on escaping with the Cityship. Secondly, during the gameplay some smaller enemies moved inside the Cityship and the testers could not see them. Overall enemies and their AI require a lot of further attention.

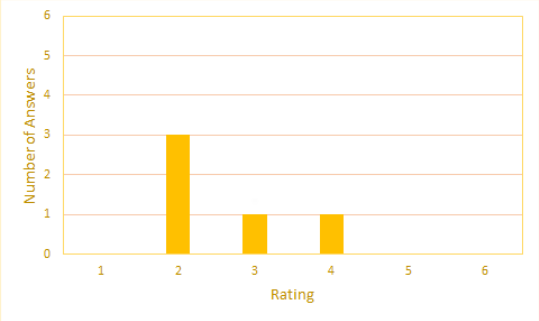

Illustration 27. The ratings of fighting with enemies mechanic

The map feature's average rating was 4.2 of 6 (Illustration 28). Based on the feedback, the map feature generally made it easier for the testers to see where the units, resources and dangerous zones are. During playtesting one tester forgot that this feature existed and did not use it all. One other tester wished to have the ability to click a specific location on map and then move there with camera in game world. This feature was implemented but it was not clear for the tester how to use it. So, regarding these examples it could be that one of the biggest problems of map feature is that the players should be more informed about this mechanic.

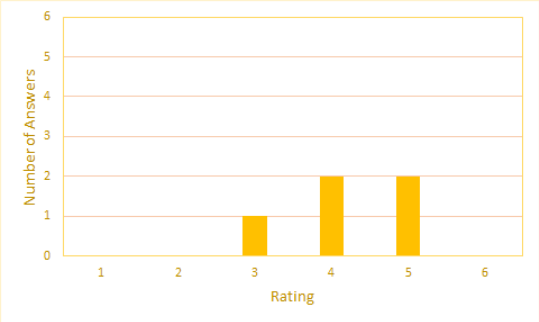

Illustration 28. The results of the map feature mechanic ratings

Based on the collected results, all the game mechanics require some improvement. The best game mechanic is the activity of avoiding dangerous zones and the worst fighting with enemies. Subchapter 5.3 describes what improvements were made based on the test results during the work of this thesis and what are planned for the future. Regarding the game mechanics it is essential to understand how level design of *Icarus* worked for the player because levels introduce new game mechanics. Therefore, if a level does not logically offer for a player to use game mechanics, they become more irrelevant for the player.

5.2.4 The Level Design

To get an overview which levels were enjoyable for the testers, they had to choose which levels they liked the most (Illustration 29). Level 3 was a favourite level among 80% of the testers. The reason behind this is that after playing first two levels of *Icarus* they figured out what and when to do during the gameplay. Also, level 3 was more compelling because it featured both dangerous zones and enemies.

The level that most users did not like was level 2 (Illustration 30). This is because they did not understand yet how all the game mechanics of *Icarus* work. For example, they did not understand what to build on the Cityship in order to give it the ability to move. This ability is essential for the player in order to win. The testers realized what they had to do in the end when the dangerous zones already covered all the metal resources that were needed to build the structures. This brought frustration among the testers because they had wasted their time and resources for unnecessary actions.

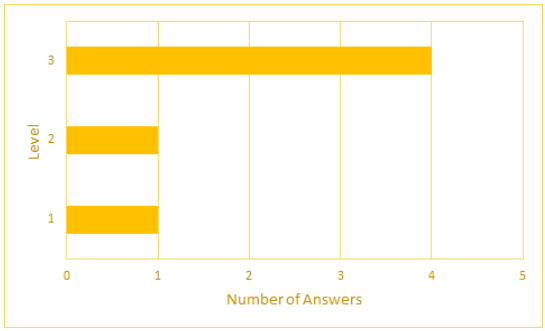

Illustration 29. The favorite levels among the testers

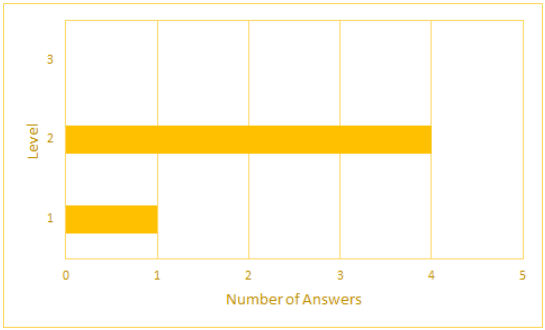

Illustration 30. The least favorite levels among the testers

Level 2 was also the most difficult level for all testers (Illustration 31). This is likely also because of the lack of knowledge of *Icarus'* game mechanics. So, the reasons are the same as why level 2 is also the most unpleasant level in *Icarus*.

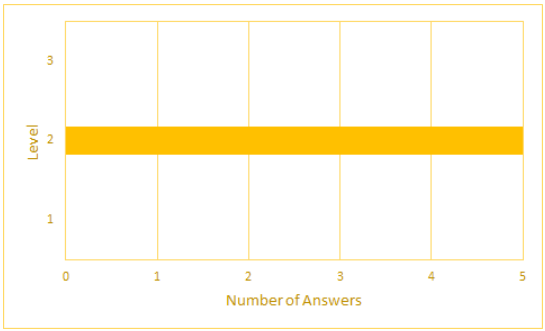

Illustration 31. The most difficult levels among the testers

The level design in *Icarus* needs enhancements. Because the testers did not understand all of the basics of the game mechanics in the first tutorial level made the second level the most difficult. In general, the testers learned the fundamentals in the second level by a trial and error approach. By the time the testers started the third level, they understood about most of the crucial elements of *Icarus*. This made this level more enjoyable, which was also supported by the additional challenge of enemies. The reason behind the fact that the testers did not learn the basics in the first level could be also because of the found design flaws of *Icarus'* user interface.

5.2.5 The User Interface

Interface is a layer between the computer game and must give the feeling of being in charge of what is happening [3]. To assess if the testers of *Icarus* felt being in control, they had to rate the controls and graphical user interface (GUI) of the game. They also brought out what should be improved and what they liked.

Regarding the evaluation of controls of *Icarus* in terms of comprehensions the testers gave a rating on a scale of 1 ("very confusing") to 6 ("very intuitive") (Illustration 32). The average rating was 4.6 out of 6 which means the controls are more intuitive than confusing. The testers liked how simply they could give orders to units and how easy it was to control the game. Still, there are some aspects that should be improved. For example as mentioned in the subchapter 5.2.3 it was not immediately understandable to build buildings on the Cityship with the right mouse button. One of the testers thought that the camera should move if the cursor is located in the edges of the screen like in some other real-time strategy games. Recognizable UI design is important because then a whole system is more usable for a user[22]. It was brought out that it would be good if there was a hotkey to select between units.

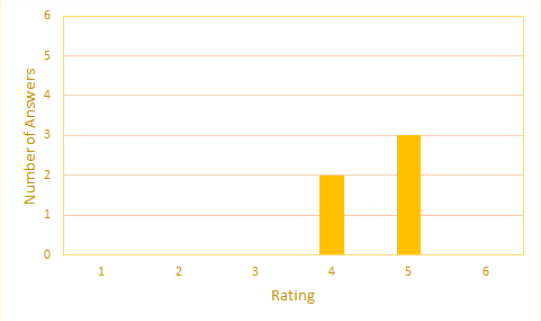

Illustration 32. The ratings of the comprehension of the controls

Computer games are mostly visual and therefore a graphical user interface has an important part in a computer game. According to Krug's Law of usability [9], it must be simple and consistent. Thus, the testers rated how usable was the graphical user interface (Illustration 33) of *Icarus* on a scale of 1 ("awful") to 6 ("perfect"). With an average score of 4.8 out of 6 the usability of GUI satisfied the testers. Nevertheless, regarding this topic there are also some suggested improvements that could be pursued:

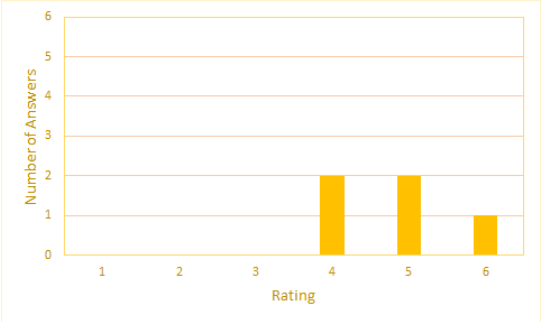

Illustration 33. The ratings of the usability of the GUI

[22] Philips M. Boost Your UX with These Successful Interaction Design Principles. https://www.toptal.com/designers/interactive/interaction-design-principles

- there should be a place to show the number of units that are selected;
- the cost of units need to be shown closer to where one creates them;
- the GUI should have different colours.

The testers also rated the graphical user interface in terms of comprehension on a scale of 1 (“very confusing”) to 6 (“very intuitive”) (Illustration 34). The average rating for this was 4.8 which means it has some issues. For example, the graphical user interface element that showed the cost of units should be closer to the icon that indicates what type of unit the user creates.

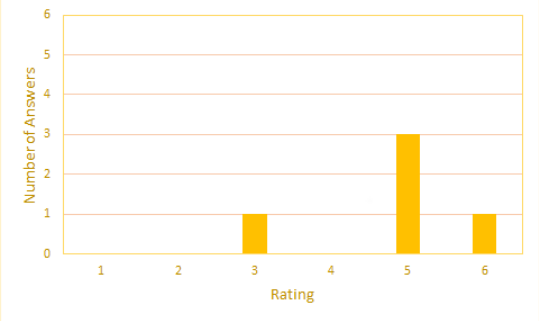

Illustration 34. The ratings of the comprehension of the GUI

In general, the graphical user interface and the controls of *Icarus* were mostly intuitive. However, there is some room for improvement. Due to the fact that a computer game is mostly a visual application it is essential to understand if it is “visually satisfying” for players.

5.2.6 The Visual Style

The visuals of *Icarus* feature a mixture between the style of Ancient Greek and science-fiction. To assess if this design decision is beneficial to the game, the testers were asked to give a rating on a scale of 1 (“awful”) to 6 (“perfect”) (Illustration 35). With a rating of 5 out of 6 it seems that the testers mostly liked it.

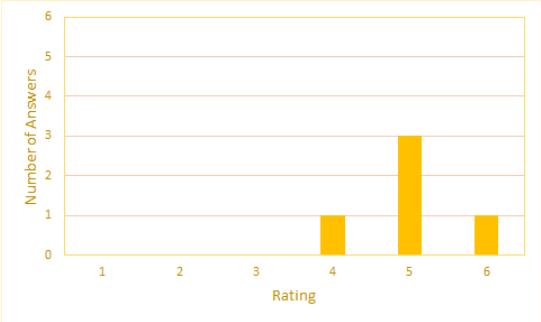

Illustration 35. The ratings of the appearance

It is indispensable that in a computer game a player distinguishes between the relevant and irrelevant game objects [9]. Units are the most important game objects for the player of *Icarus*. Therefore, making them easily noticeable for the player is necessary. To see if the units in *Icarus* were easily visually

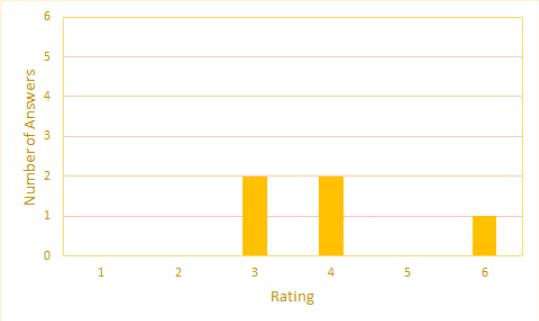

Illustration 36. The ratings of how hard was to make difference between the units and the background

separable from background the testers had to assess this on a scale of 1 (“very hard”) to 6 (“very easy”) (Illustration 36). With an average score of 4 out of 6 it was somewhat easy. Although, there were some notable suggestions on how to improve it:

- create an icon above the units when camera is zoomed out,
- give the units a visual “aura”.

The appearance of *Icarus* was overall attractive for the testers. Still, it is not perfect and also requires some attention. Thus far, the test results have described the good and the bad aspects of *Icarus*. Next subchapter concludes the test results of *Icarus*.

5.2.7 Conclusion

A computer game is meant for an audience that plays it. Players are also important because with a positive experience from a computer game they could promote it indirectly to new audiences [10]. Therefore, the final part of questionnaire of the testing of *Icarus* was to assess if the testers would introduce it for new players. Hence, they had to rate on a scale of 1 (“very unlikely”) to 6 (“very likely”) on how likely they would be to recommend *Icarus* to their friends (Illustration 37). An average score of 4.4 out of 6 indicates that it is possible that the testers could recommend *Icarus* to their friends. This could mean that *Icarus* would be played if improvements are made.

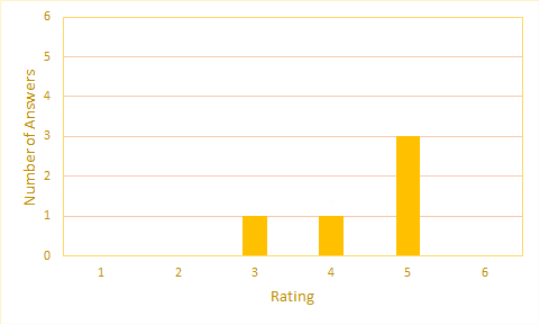

Illustration 37. The results of how likely *Icarus* would be recommended among the testers friends

5.3 Improvements

Subchapter 5.2 introduced issues with *Icarus*. Some of these bugs and design flaws were fixed after the testing for this thesis. The most important changes were:

- unit cost (Illustration 38, no 1) and information text (Illustration 38, no 2) GUI elements rearrangement;
- the player can now move the camera with the mouse cursor when it is in the edge of the screen;
- Workers gathering system improvement (the bug that did not let workers collect resources was fixed);

- the player can now see the value of how much resources are left in a resource object;
- hotkeys feature addition (the player can now select between all units, the Cityship and idle Workers with keyboard buttons).

Illustration 38. The GUI of Icarus before (left) and after (right) improvements

The majority of fixes and improvements are planned to be implemented in the future. This is due of the fact that it is not possible to do it in the scope of a bachelor's thesis. Most important aspects that require attention in the future are:

1. level design improvements (most importantly the redesign of the tutorial level);
2. enhancement of the visuals (most importantly the aspect that units should be more distinguishable from the background of the game world);
3. more advanced artificial intelligence of enemies;
4. improved notification system (for example, the player should be notified when a Worker stops gathering resources);
5. designing more levels;
6. playtests with new users;
7. more balanced levels and mechanics;
8. improvement of the pathfinding of units (they should not fly in to the dangerous zones).

6 Conclusion

As a result of the work of this bachelor's thesis a real-time space strategy game *Icarus* was made. It was designed to pursue the core principles of the field of game design and real-time strategy game elements. Numerous game mechanics and game objects were designed and implemented. Most important of them are units that realize main gameplay objectives of a real-time strategy game, including base building, resources gathering and fighting with enemies in real-time. The mechanic of avoiding expanding dangerous zones in *Icarus* was implemented to create a state of emergency that is one of the appeals of real-time strategy games. Although, there are games that include levels with similar game mechanics to the dangerous zones, computer games, that use it as a core mechanic like *Icarus,* were not found.

Software design patterns were used to realize cohesive systems. The main unit systems were implemented using the component, mediator, command and observer pattern. These patterns made it easier to follow the iterative design process. Although, more systems were made for *Icarus,* their implementation was relatively straightforward. Therefore, they were not introduced in this thesis.

Three levels were created for *Icarus* in order to playtest it on possible players. Testing featured also topics from usability testing. The testers were familiar with real-time strategy games, but they had never played *Icarus*. The gameplay and the voice of testers was recorded during test sessions. In the end they had to fill a questionnaire that included questions about the background of the tester, overall experience of the gameplay, game mechanics, user interface and visual style. Generally, the testers enjoyed the gameplay but there were various shortcomings that made their experience uncomfortable. Some of these issues were fixed during this thesis but the majority of them will be improved in the future.

Special thanks go to the supervisor Raimond-Hendrik Tunnel whose suggestions and help made the quality of the computer game *Icarus* and this thesis better. Gratitude goes to all of the testers of *Icarus* without whom a lot of issues in the game would not have been found. Thanks go to all people behind the Computer Graphics, Computer Game Development and Design, Computer Graphics Project and Programming Patterns in Computer Games courses in the University of Tartu because they gave me the basic knowledge to design and implement *Icarus*.

References

[1] Geryk B. A History of Real-Time Strategy Games. 2006. https://web.archive.org/web/20110427052656/http:/gamespot.com/gamespot/features/all/real_time/ (01.05.19)

[2] Moss R. Build, gather, brawl, repeat: The history of real-time strategy games. 2017. https://arstechnica.com/gaming/2017/09/build-gather-brawl-repeat-the-history-of-real-time-strategy-games/ (01.05.19)

[3] Schell J. The Art of Game Design – A Book of Lenses. United States of America, Burlington: Morgan Kaufmann Publishers. 2008.

[4] Salen K, Zimmerman E. Rules of Play – Game Design Fundamentals. England, Massachusetts London: The MIT Press Cambridge. 2004.

[5] Nystrom R. Game Programming Patterns. [S.I.]: Genever Benning. 2014.

[6] Cooper J. W. C# Design Patterns: A Tutorial. United States of America, Boston: Addison-Wesley Professional. 2002.

[7] Nielsen J. Why You Only Need to Test With 5 Users. 2000. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (01.05.2019)

[8] Nielsen J. Usability 101: Introduction to Usability. 2012. https://www.nngroup.com/articles/usability-101-introduction-to-usability/ (01.05.2019)

[9] Despain W. 100 Principles of Game Design. [S.I.]: New Riders. 2012.

[10] Steinberg S. Video Game Marketing: The New Bible Part 1. 2011. https://www.gamesindustry.biz/articles/2011-05-09-video-game-marketing-the-new-bible-article (01.05.2019)

Appendix

I. Glossary

1. **Real-time strategy** – a game genre that includes strategy games that advance in real-time and typically consist of objectives like building/managing base, gathering resources and fighting with enemies [2].
2. **Game design** – process of creating fundamental principles for a game [3].
3. **Challenge** – an aspect of a game that gives the player pleasure (usually a problem that must be solved) [3].
4. **Resource** – a system of a game that is used to realize the game mechanic of resources gathering/management. For example, in some real-time strategy games the player can collect wood and gold with their units.
5. **Enemy** – an opponent of a player in a game which provides challenge. For example, in *Icarus* the opponents are computer-controlled space pirates.
6. **Playtesting** – a testing of a game where players play it to assess if the gameplay experience meets the requirements of its game design [3].
7. **Game mechanic** – a game element of rules and procedures of a game that characterize its goal [3].
8. **Game element** – fundamentals of a game. The four basic elements are mechanics, story, aesthetics and technology. [3]
9. **Artificial intelligence** – the field of computer science that aims to make computers simulate human intelligence[23].
10. **Turn-based strategy** – a game genre that includes strategy games that advance in turns[24]. For example, chess is a turn-based strategy game.
11. **Unit** – a game object in real-time strategy games that a player can directly interact with.
12. **Base building** – objective of a real-time strategy game that includes building structures.
13. **Floating camera** – a movable camera that does not have constrained rotation and thereby giving the user the ability to rotate it in 3D[25].

[23] http://vallaste.ee/
[24] https://www.techopedia.com/definition/1923/real-time-strategy-rts
[25] https://www.whatgamesare.com/2011/10/camera-comes-first-game-design.html

14. **Movable camera** – a camera in a computer game that can be moved by the player, but has constrained rotation[25].
15. **Iterative design** – a process where game design decisions are made based on results from playtesting and prototyping by the developer itself when the game is still in development [4].
16. **Level design** – a process that includes creation of levels of a computer game.
17. **Tower defense** – a computer game genre that includes strategy games that has the objective to protect a point from enemies by building structures that fight with enemies[26].
18. **Battle royale** – a computer game genre that includes the idea of players fighting until the last one of them is "alive" ("last man standing")[13].
19. **Core mechanic** – action that a player does often in a game [4].
20. **Foundational technology** – technology which contains everything that is indispensable for a computer game [3].
21. **Decorational technology** – technology that indicates everything that improves the development and/or the gameplay experience of a computer game [3].
22. **Rendering** – a process that converts data in to a form to display or print it[23].
23. **Bitmap graphics** – graphics which represent images as bitmaps (representation of a graphical image that consists of rows and columns of points that have specific values)[23].
24. **Vector graphics** – graphics which represent images as geometrical formulas[23].
25. **Software design pattern** – a solution to an issue in software design that arises frequently[27].
26. **Component pattern** – a software design pattern that allows an object to have numerous individual components that need not to be connected but can still send information to each other [5].
27. **Mediator pattern** – a software design pattern that includes a mediator that is used for allowing other components to communicate with each other [6].
28. **Command pattern** – a software design pattern that includes commands which are instances that call methods to realize specific actions [5].

[26] https://www.giantbomb.com/tower-defense/3015-413/
[27] https://sourcemaking.com/design_patterns

29. **Observer pattern** – a software design pattern that consists of two main elements: an observer and a subject. The subject has a list of observers that are notified when an event happens and act accordingly to this event [5].
30. **Hotkey** – a key or a key combination of a computer keyboard that executes specific functionality if pressed.

II. Launch Guide

In order to run *Icarus*, one must follow these steps:

1) Extract the archive “AccompanyingFiles.zip”.

2) Open directory “/Build”.

3) Run the file “Icarus.exe”.

4) Choose desired screen resolution (recommended “1920 x 1080” if available) (Figure 1).

5) Choose desired graphics quality (recommended “Ultra” if available) (Figure 1).

6) Press “Play!” button (Figure 1).

Minimum system requirements:

- CPU: Intel Core i5-7300HQ CPU @ 2.50GHz
- GPU: Intel HD Graphics 630
- RAM: 8 GB
- Operating system: Windows 7 (or newer)

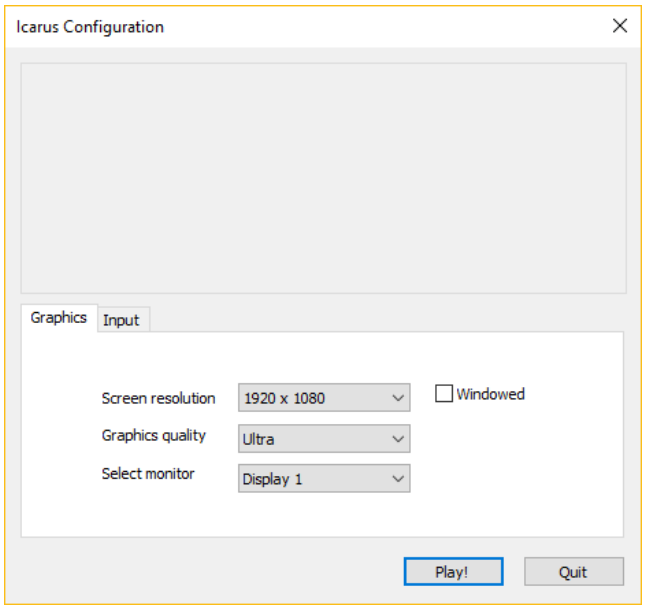

Figure 1. Unity launcher

III. Game Guide

All objectives are shown in game.

Controls:

W (or up arrow)	Move forwards with camera.
A (or down arrow)	Move backwards with camera.
S (or right arrow)	Move right with camera.
D (or left arrow)	Move left with camera.
Space	Open map.
Shift	Hold it down to: 1. add multiple commands to a unit; 2. select multiple units
Escape	Pause the game.
Right mouse button	Give a command to selected unit(s).
Left mouse button	1. Select a single unit; 2. select multiple same type units that are visible (double click on a unit); 3. hold it down to select multiple units with a selection box; 4. interact with graphical user interface.
Middle mouse button	Move camera to clicked location in the game world.
Tab	Select between all units.
I	Select a first Worker that is in an idle state.
C	Select the Cityship.

IV. Source Code

The source code and assets of *Icarus* can be found at https://bitbucket.org/danielthen/icarus/src/master/

V. Some Videos of Test Sessions

Session 1	https://www.youtube.com/watch?v=vHuBqsQE_eY&feature=youtu.be
Session 2	https://www.youtube.com/watch?v=StDJW3MAOKE&feature=youtu.be
Session 3	https://www.youtube.com/watch?v=2E5nmvcJk8M&feature=youtu.be
Session 4	https://www.youtube.com/watch?v=x5HtIHNpE7k&feature=youtu.be

VI. Accompanying Files

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

- /Source – the folder that contains the Unity project files (source code, assets, etc) of *Icarus*.
- /Build – the folder that contains the game *Icarus* and necessary files to run it.
- /Testing – the folder that contains questionnaire of test sessions and answers of the testers to it.
- demo.mp4 – a simple video demonstrating several gameplay elements of *Icarus*.

VII. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Daniel Nael**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Icarus – a Real-Time Strategy Game in Space,

   (title of thesis)

   supervised by Raimond-Hendrik Tunnel.

   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Tartu, **10.05.2019**