



## Assignment of bachelor's thesis

<b>Title:</b>	Implementation of AI Turn-Based Strategy Game in Virtual Reality
<b>Student:</b>	Karen Akopian
<b>Supervisor:</b>	doc. Ing. Mgr. Petr Klán, CSc.
<b>Study program:</b>	Informatics
<b>Branch / specialization:</b>	Web and Software Engineering
<b>Department:</b>	Department of Software Engineering
<b>Validity:</b>	until the end of summer semester 2021/2022

### Instructions

Introduce AI logic into a turn-based strategy game and implement the latter in the virtual environment of Unity. Use the following procedure:

1. Get to know in detail with the literature and content of turn-based strategy games and provide research on this topic.
2. Get acquainted and learn to work in the system of virtual reality Unity.
3. Design organization and structure of 3D scenes for a turn-based strategy game.
4. Propose AI logic for this game.
5. Design and create related 3D objects.
6. Visually program scenes in Unity.
7. Build and test a minimum viable virtual version of the proposed game.
8. Extend and optimize the minimum viable implementation according to the test results.
9. Compare the virtual implementation with the desktop variants.

---

*Electronically approved by Ing. Michal Valenta, Ph.D. on 21 November 2020 in Prague.*





**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

Bachelor's thesis

# **Implementation of AI Turn-Based Strategy Game in Virtual Reality**

*Karen Akopian*

Department of Software Engineering  
Supervisor: doc. Ing. Mgr. Petr Klán, CSc.

May 14, 2021



---

## **Acknowledgements**

I would like to express my gratitude to my supervisor, doc. Ing. Mgr. Petr Klán, CSc., for his support for my thesis. Finally, I would like to thank my parents for all the support and encouragement they have given me during my studies.



---

## Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No.121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 14, 2021

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2021 Karen Akopian. All rights reserved.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

### **Citation of this thesis**

Akopian, Karen. *Implementation of AI Turn-Based Strategy Game in Virtual Reality*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.



---

## Abstrakt

Tato práce popisuje proces navrhování a implementace tahové strategické hry s využitím herního robota. Cílem této práce je představit chytrého robota, který by hrál proti člověku. Výsledkem této práce je implementovaná hra v enginu Unity. Hra je vyvinuta pro brýle pro virtuální realitu a pro stolní počítače.

**Klíčová slova** design, implementace, virtuální svět, virtuální realita, vizualizace, Unity, AI

---

## Abstract

This thesis describes the process of designing and implementing a turn-based strategy game with the use of a game bot. The goal of this thesis is to introduce a clever bot that would play against a human being. The result of this work is implemented game in Unity engine. The game is developed for virtual reality glasses and desktop variants.

**Keywords** design, implementation, virtual world, virtual reality, visualization, Unity, AI



---

# Contents

<b>1 Introduction</b>	<b>1</b>
<b>2 Objectives</b>	<b>3</b>
<b>3 Game Theory</b>	<b>5</b>
3.1 Strategy . . . . .	5
3.2 Payoff Matrix . . . . .	5
3.3 Saddle point payoff matrix . . . . .	6
3.4 MiniMax algorithm . . . . .	6
3.4.1 Search tree using Minimax . . . . .	6
3.4.2 Alpha-beta pruning . . . . .	7
<b>4 Virtual reality</b>	<b>9</b>
4.1 Uses of virtual reality . . . . .	12
4.2 Developing for virtual reality . . . . .	12
4.2.1 Virtual reality in Unity . . . . .	12
<b>5 Creation of a New Game</b>	<b>15</b>
5.1 Implementation . . . . .	15
5.2 Design in Desktop and VR variants . . . . .	17
5.3 Algorithm . . . . .	17
5.3.1 Evaluation . . . . .	17
5.3.2 Positioning . . . . .	18
5.3.3 The use of MiniMax algorithm. . . . .	19
5.3.4 Alpha-beta pruning . . . . .	20
5.4 Testing . . . . .	22
<b>6 Conclusion</b>	<b>25</b>
<b>Bibliography</b>	<b>27</b>

<b>A Acronyms</b>	<b>29</b>
<b>B Contents of enclosed CD</b>	<b>31</b>

---

# List of Figures

1.1 Garry Kasparov playing against Deep Blue . . . . .	1
3.1 Search tree . . . . .	7
3.2 Search tree after using alpha-beta pruning algorithm . . . . .	8
4.1 Mobile VR . . . . .	10
4.2 Oculus Go Headset . . . . .	11
4.3 Oculus Go Controller . . . . .	11
4.4 Screenshot from game Half-Life: Alyx . . . . .	12
4.5 Unity interface during play mode with headsets . . . . .	13
4.6 Unity interface with added controller for VR game . . . . .	13
5.1 Planning stage. . . . .	15
5.2 Interface for time left as well as stats of the game unit. . . . .	16
5.3 Game during the play (the player decides where to move the unit.) . . . . .	16
5.4 Game in VR glasses. . . . .	17
5.5 Visual priorities for the game unit. . . . .	18
5.6 Leaves evaluation with depth 2. . . . .	19
5.7 Tree evaluation for depth 1. . . . .	19
5.8 Tree evaluation for depth 0. . . . .	20
5.9 Tree evaluation for the left side. . . . .	20
5.10 Tree evaluation for depth 0 after right side evaluations. . . . .	21
5.11 Tree evaluation for the left side of depth 1. . . . .	21
5.12 Difficulty selection inn VR. . . . .	22
5.13 Board with units. . . . .	22



# Introduction

Games have always been a beginning for AI developers to test, learn and develop their newest techniques. Chess is the most famous game in this context and the one that has been studied used the most by computer scientists. Many of the great minds of early computer science such as Alan Turing and John von Neumann devised their own approaches towards a program that would be able to play chess. Then, IBM's "Deep Blue" computer won versus the reigning world champion Garry Kasparov in 1997. Which was a huge event for the recognition of artificial intelligence and its capabilities and further development of AI.



Figure 1.1: Garry Kasparov playing against Deep Blue

There were a lot of games that took an inspiration from chess, because chess was one of the best known games in the history, so to take a basic chess game-play and add classes spells and number of units were a pretty logical decisions that made the game more complex, hence lead to more impressive

## 1. INTRODUCTION

---

results and being the hit on the world market. This is the game called "Heroes of Might and Magic III." (HoMM) Gameplay consists of game techniques are used in chess. To illustrate, two players make moves followed one after another on a game board which is divided to tiles. The gameplay consists of attacking enemy units leading to the victory if one of the sides loses all of its troops. The game is the inspiration of this thesis. Besides the fact that players can play one versus another, it is possible to play against AI. HoMM has AI which during the battle looks where the enemy can go, to create some simple tactics against the locations of the units. Also, units attack with certain priorities.

The purpose of this thesis to use chess AI techniques to implement smart bots which will calculate the moves in advance to win the opponent. The thing is to use not the priorities of units like in HoMM but overall the board value for both sides.



---

# Objectives

This paper is intended to analyze upon the implementation of artificial intelligence (AI) turn-based strategy game in virtual reality. Artificial intelligence which help the opponent to make strategically well-thought moves on the battlefield during the game. Virtual reality, in turn, ensures the user experiences a wider scope of the game environment. The following steps of the implementation will be covered:

### **Game Theory.**

This chapter introduces the concept of game theory explaining strategic interaction among rational decision-making algorithms which are used in artificial intelligence (AI). The chapter explains what is Payoff Matrix and its saddle point. Also, it introduces MiniMax algorithm.

### **Introduction to virtual reality.**

This chapter introduces the fundamental terms of virtual reality, its use in today's world and game sphere. Also, it presents game engine Unity concepts as well as development process used for this game.

### **Creation of the virtual world.**

This chapter presents the virtual world's environment with all intractable objects within the scene so that the game is playable for the user. Also, it elaborates on how the provided strategies are used within given virtual world environment going through all the game development stages and describes them to their deepest.



---

# Game Theory

Any time when there is a situation where at least two players compete with each other and involve known payouts or quantifiable consequences, we can use game theory to help determine the most likely outcomes. The game provides with the possibility to perform certain actions by some actors which leads to a particular result. Therefore, it is possible to make the best decision depending on other outcomes.

## 3.1 Strategy

To find the way which leads to victory, it is necessary to have some plan which will outplay the opponent. During the strategy time, sides decide the certain actions to perform and take care of other possible outcomes from other sides. To make the right plan, it is necessary to understand the values of each piece to decide whether to sacrifice it or not, where to move, and what the opponent would do after certain actions are performed.

## 3.2 Payoff Matrix

Payoff is the payout a player receives from arriving at a particular outcome (the payout can be in any quantifiable form like money, years, and etc.). As mentioned above, every player performs an action that leads to a certain result. In other words, for each such game, we can represent all of the information about the game in a matrix. This matrix is called the Pay-off matrix. It is a matrix with a list of first player's strategies as rows and a list of second player's strategies as columns. The entries in the pay-off matrix are what one player gains for each combination of strategies. If this is a negative number then it represents a loss for one player, a positive number is a profit, balance/tie must be equal to zero. In the table below there are four possible outcomes where each player can take two actions: A and B

		Player Y	
		A	B
Player X	A	2	-2
	B	4	-6

### 3.3 Saddle point payoff matrix

Maximin value is equal to the maximal value of rows and Minimax is the minimal value of columns. The best strategies for the player is the row and column that the saddle point belongs to.

		Player Y		
		A	B	
Player X	A	2	-2	-2
	B	4	-6	-6
		4	-2	

At this point, it is -2 in both cases. So, player X should always play A, whereas player Y should always play B to get the most suitable outcome. This is what we want to get in MiniMax algorithm.

### 3.4 MiniMax algorithm

MiniMax is a recursive algorithm used in decision-making to provide an optimal move for the player assuming that the opponent is also playing optimally. In fact, it performs a depth-first search algorithm for the exploration of the complete game tree and chooses the best way for the best outcome.

While performing the search, we divide the algorithm into two main parts: Player's move and Bot's move. Talking about numbers, we want to maximize profit for a bot, at the same time player wants to do the same for themselves.

#### 3.4.1 Search tree using Minimax

In this algorithm, we create the recursive tree of all possible moves to explore all of the outcomes on a given depth, and the position is evaluated at the ending "leaves" of the tree.

After that, we return either the smallest or the largest value of the child to the parent node. It will depend on whether it's a bot or a player to move. This means that we try to either minimize or maximize the outcome at each level. In the following figure there is an example of the tree where we maximize the bot and minimize the player:

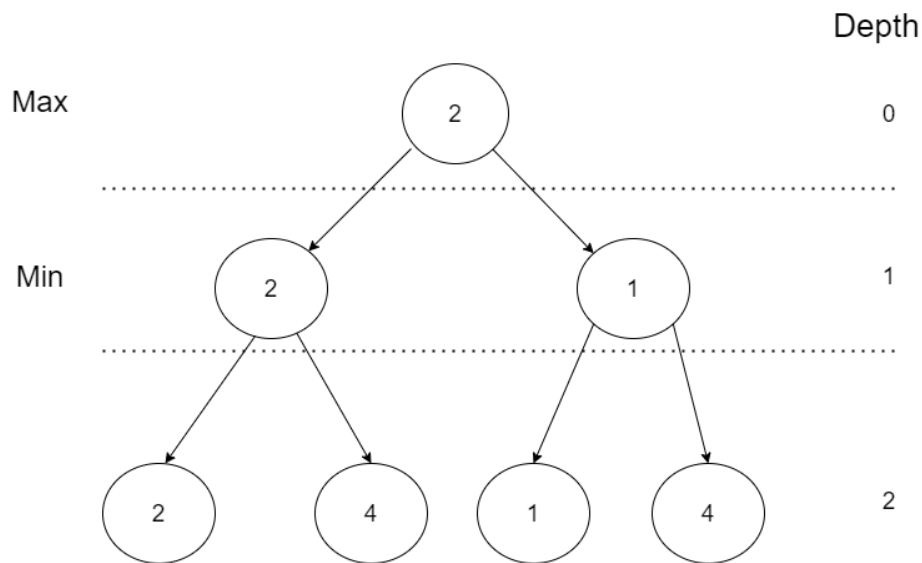


Figure 3.1: Search tree

In the example figure above, the evaluated outcomes of actions lead to 2, 4, 1, 4 (Which are the leaves of the tree.) First, the algorithm goes to number 2, then it goes to its sibling, which is 4, and in the parent node it chooses the minimum value, in this case, it is 2. Then, we go to the other sibling, which is a node with number 1 with depth 2. We find the other sibling which is 4 and put the minimum value to the parent, which is 1. Now, we go to depth 0 and compare children from depth 1 and get the maximum value, which is 2. This means that we want to choose the left path for the best outcome.

The effectiveness of this algorithm heavily depends on the search depth we can achieve. The more we look into possible outcomes the more chance to make the right choice. In chess-like games calculating the best possible move is with depth equal to 18. For new players, it is fine to look into a depth of 3. For the middle-level player, it is fine to play with 8 depth levels. Whereas experts can play with 10 and higher levels. As it performs DFS for the game-tree, so the time complexity of minimax algorithm is  $O(b^m)$ , where  $b$  is branching factor of the game-tree, and  $m$  is the maximum depth of the tree. Space complexity of minimax algorithm is also similar to DFS which is  $O(b^m)$ .

### 3.4.2 Alpha-beta pruning

Alpha-beta pruning is an optimization method to the minimax algorithm that allows us to omit some branches in the search tree. This helps us evaluate the minimax search tree much deeper, while using the same resources. The

### 3. GAME THEORY

---

alpha-beta pruning is based on the situation where we can stop evaluating a part of the search tree if we find a move that leads to a worse situation than a previously discovered move. The alpha-beta pruning does not influence the outcome of the minimax algorithm — it only makes it faster. The alpha-beta algorithm also is more efficient if we happen to visit first those paths that lead to good moves.

The algorithm maintains two variables: alpha and beta. They respectively represent the minimum score that the maximizing player is assured of and the maximum score that the minimizing player is assured of. Initially, alpha is negative infinity and beta is positive infinity, i.e. both players start with their worst possible score. Whenever the maximum score that the minimizing player (i.e. the "beta" player) is assured of becomes less than the minimum score that the maximizing player (i.e., the "alpha" player) is assured of (i.e. beta less than alpha), the maximizing player need not consider further descendants of this node, as they will never reach better outcomes.

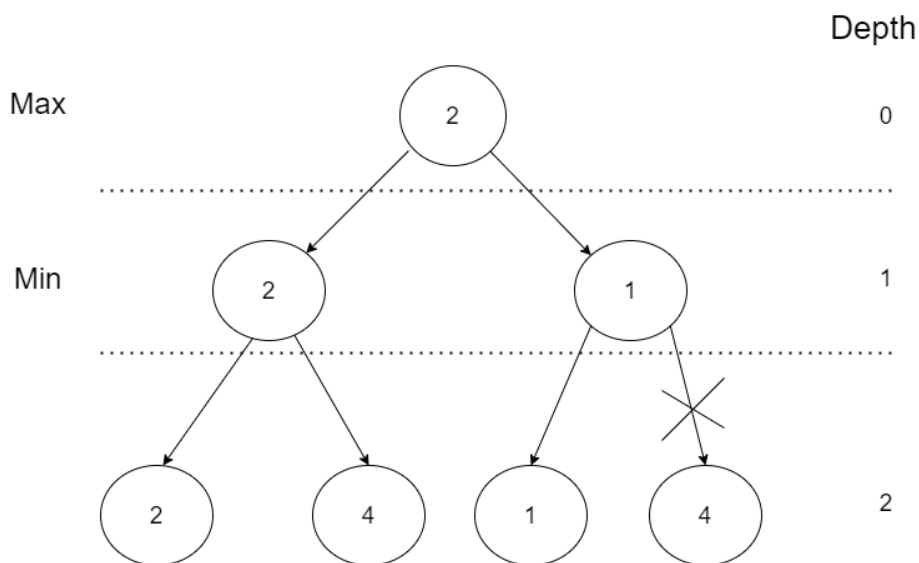


Figure 3.2: Search tree after using alpha-beta pruning algorithm

In the figure above, beta is equal to 2, since it was the best outcome for the minimization, alpha is negative infinity. When we go to the right node in depth 1, we know that the beta was 2, and we want to find maximum value for the parent. This means that the node will never become greater than 1, so we can omit last calculation which would lead us to number 4. So, with this algorithm, omitted a part of the tree which lead us to the same result with less performance used.

---

## Virtual reality

The most important subject of virtual reality is trying to approach it in a way that's close to the reality of the vision. Every headset aims to perfect its approach to creating very interesting 3D surrounding conditions. Each VR headset puts up a screen (or two - one for each eye) in front of eyes this way, processing any interaction with the real world. Two lenses, that track the object which moves around the screen and is usually placed between the screen and the eyes that change to fit new conditions based on individual eye movement and positioning, so it adjusts focus and clarity, like objects that far away can be in lower quality. The visuals on the screen are given either by using a mobile phone or HDMI cable connected to a PC. There are two different two main types of VR headsets: Mobile VR and Room Scale VR.



Figure 4.1: Mobile VR

The VR headset above is a Mobile VR, the concept is based on using your phone as a screen for VR and goggles were an additional adjustment. The main limitation is the power of the system. Even though phones now are powerful as they have never been before, for instance phone would've not been able to run game as small as this one. So it was limited by simple tasks such as watching Videos made specifically for VR or using to look at landscapes in google maps, which was impressive for the phone based VR set but nevertheless not that big in the grand scheme of things.

The VR headset used for this project is Oculus Go glasses is an example of Room Scale VR. you can see on Figure 4.2. Room Scale VR or PC VR headsets rely on a constant link to a powerful computer to monitor the user's location and frequently use external sensors or cameras to do so. This allows users to immerse themselves in a virtual world of high quality, but it requires a certain amount of setup space as well as a powerful PC configuration.





Figure 4.2: Oculus Go Headset

To use all virtual reality (VR) features, it is necessary to have VR Headset as well as controllers (can be one or two controllers). Controllers are used to handling hands movement to be able to interact with the environment. Such controllers have buttons as well as triggers and analog sticks. An example of such VR controller can be seen in the figure:



Figure 4.3: Oculus Go Controller

### 4.1 Uses of virtual reality

Virtual reality (VR) has been used in a variety of areas such as education, architecture, and healthcare. However, the most common usage of it is witnessed in entertainment applications such as 3D cinema and video games.

The latter incorporates real-world simulations like skiing, swimming, or flying on the plane. In addition, regular games. In Figure 4.4 there is a picture of what the player sees when he/she plays Half-Life: Alyx, they see their hands and it is possible to move every finger as well as hands as if it is in the real world.

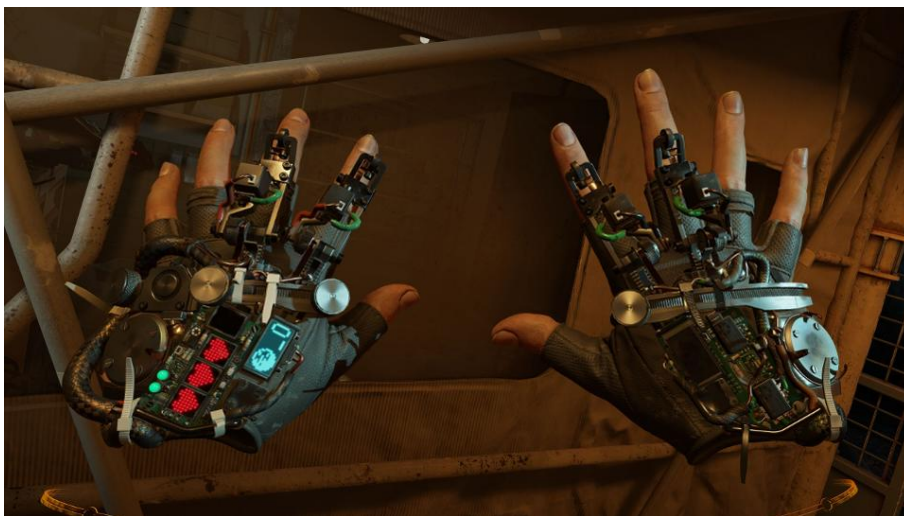


Figure 4.4: Screenshot from game Half-Life: Alyx

### 4.2 Developing for virtual reality

The game needs to have some scene where the player could play and see what is happening. Then it is necessary to have some controls where the player would take some actions so that the game would be interesting to play. And all of this needs to happen multiple times per second to give the user a feeling of reality. There are two main engines that are used to develop real and non-real worlds: Unity and Unreal Engine. Unity was chosen to implement the game in the latter.

#### 4.2.1 Virtual reality in Unity

Unity game engine offers a kit for VR development, it is possible to use VR glasses during the play to get what is happening during the play. This is very

## 4.2. Developing for virtual reality

useful for the debug. It is possible to look at the right and left eye separately as well as together. Also, we can look at current position and rotation of the controller and camera. When we are talking about creating a game we have to think about which

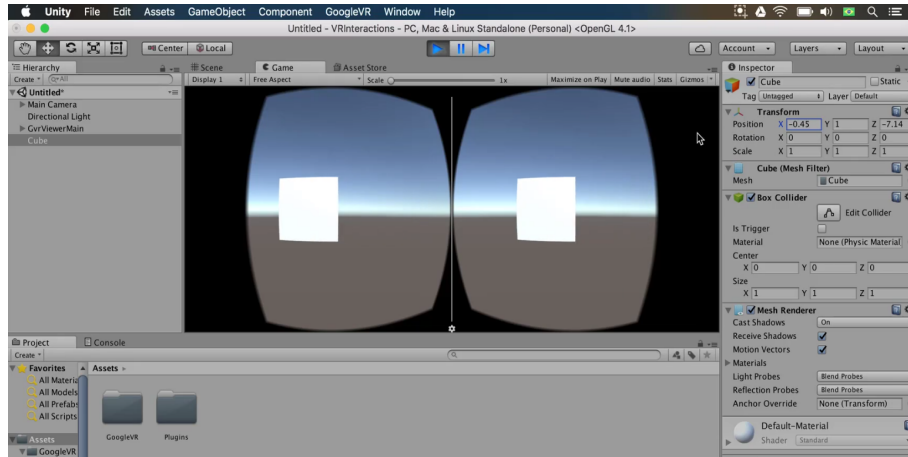


Figure 4.5: Unity interface during play mode with headsets

Unity VR kit for Oculus devices offers some assets to use for the camera and controller. In fact, it is enough to put scripts for a camera to the camera object, and controller scripts to the game objects with visuals.

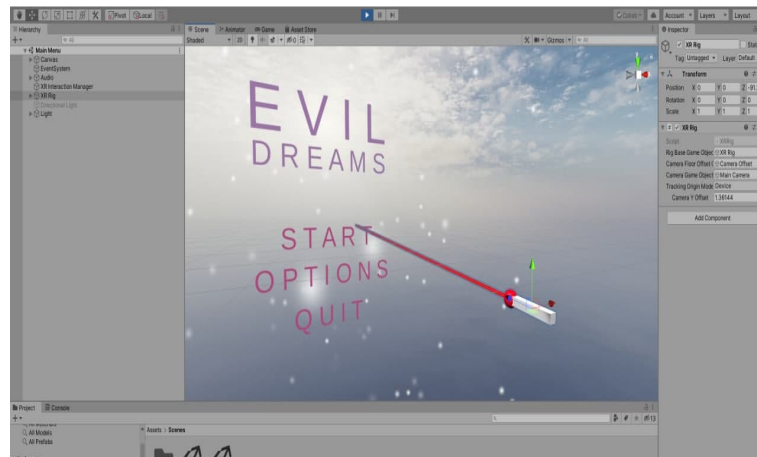


Figure 4.6: Unity interface with added controller for VR game

In the implemented game it is a simple "microphone" style controller, it can be seen in Figure 4.6. Also, it is necessary to use interactive casters and events so that the game environment will be interactive for the user. The

#### 4. VIRTUAL REALITY

---

casted ray can be used for buttons on the user interface as well as aiming at correct places of the scene, which are the game objects like tiles and units.

As explained above, the controller moves along with the 3d model of the microphone which points to a certain point of the screen with red ray for visibility.

## Creation of a New Game

The world itself should have objects to play with. These are the environment itself so that the user is able to feel the real world, the board to play the game, all the units, and the user interface to get some additional information about the gameplay. The game will have 3 different difficulty levels: easy, medium, and hard which can be chosen before the start of the game, their differences will be discussed later.

### 5.1 Implementation

The game has a set of rules: there is a board with 80 tiles divided into a 10x8 grid. Players take two sides 2x8 to place their troops. When the game starts there is a planning stage. There is a user interface element below the board with all available units. The player should put every unity on a board and hit on pink "Play" button on the right bottom corner to start the battle.

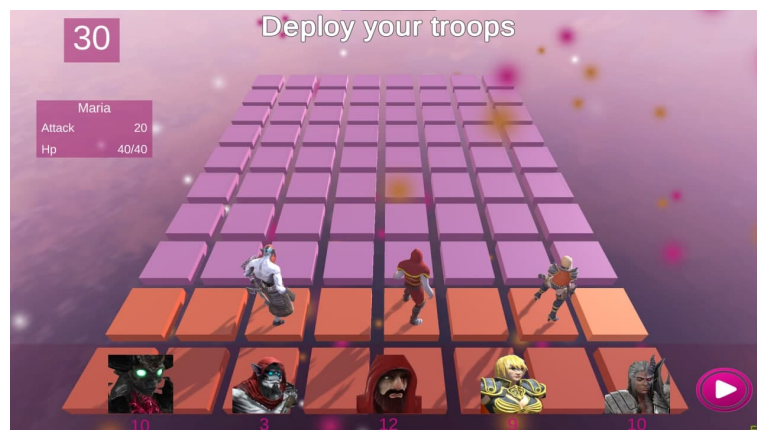


Figure 5.1: Planning stage.

## 5. CREATION OF A NEW GAME

---

During the game, there is an army turn user interface element that shows from left to right the order of played units as well as units stats which shows hovered units HP, damage, and amount.

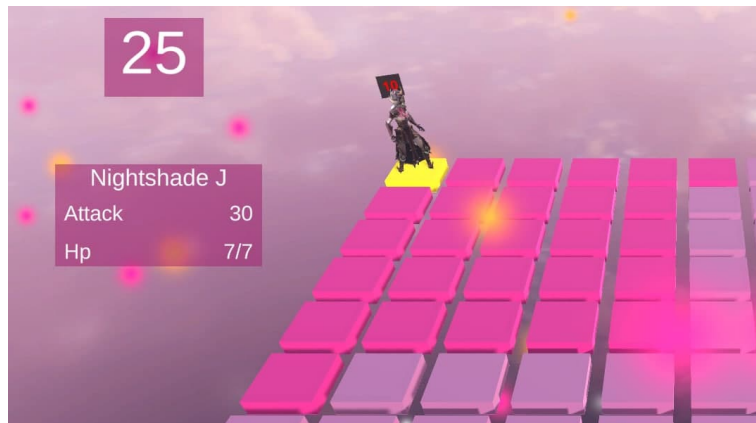


Figure 5.2: Interface for time left as well as stats of the game unit.

During the player's turn, it is possible to click on a tile to move the unit as well as to hit the enemy next to the new position. When the unit hits, it deals damage equals to its current amount times the damage of the unit. Then, the player should hit on "Next Turn" button or wait until the timer ends up changing turn. The enemy will attack by itself players' units. The one who survives with at least one unit wins.

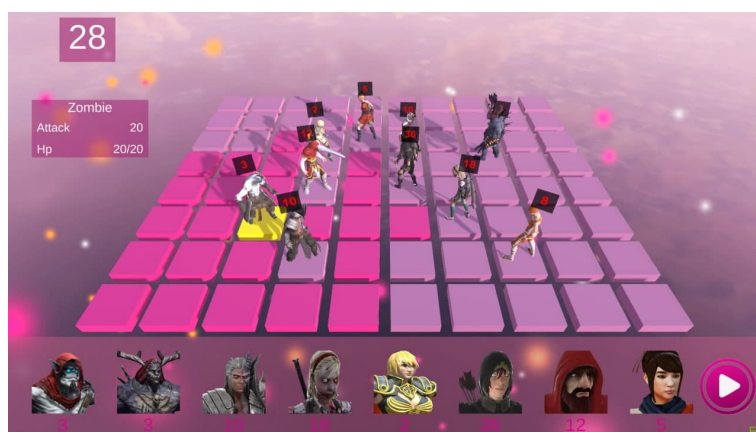


Figure 5.3: Game during the play (the player decides where to move the unit.)

## 5.2 Design in Desktop and VR variants

The game was developed for PC as well as VR for comparison. Basically, the main difference between them is UI. Figure 5.4 shows how the game looks inside of the headset. For VR games, UI is not set to the corners of the screen. The user interface is located in the world space as a 3d object, for the user, it seems that he/she is flying. Since rotating the head and looking at the corner of the world space to push button play is hard, it is better to put the button in the middle of the screen. Also, other user interface objects such as turns and time/stats should be rotated and look at the player to make everything visible and readable.



Figure 5.4: Game in VR glasses.

## 5.3 Algorithm

As explained before, there are turns of units and a board. This section will describe how to use discussed algorithm for the game like this.

### 5.3.1 Evaluation

Unit's value  $v$  is calculated as the number of units  $n$  times unit's health  $h$  plus current health  $c$  which is

$$v = n * h + c$$

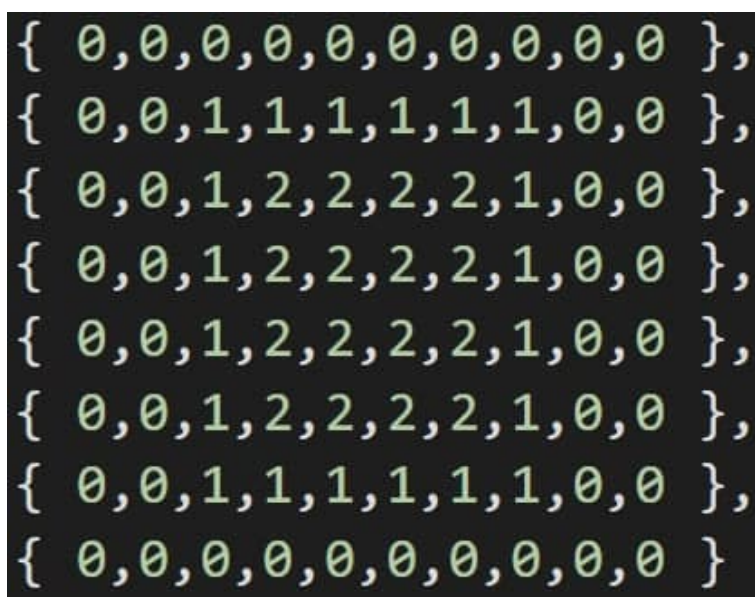


To calculate the board, we calculate it for the player and bot separately and get a sum of all units' evaluated values. For the perfect balance, their total evaluated values must be the same. So, if we subtract them we get 0. We will use this as the start of the comparison. Every time the unit kills another unit, the overall score is being recalculated, so that when the bot's unit kills the player's unit, the overall number will be more than 0, and less than 0 when the player kills the bot's units.

If we get all possible moves and find out which one gives us the greatest number, then that move will be the most profitable. With this approach, the bot will make simple moves that wouldn't lead to anything special.

### 5.3.2 Positioning

The best place where to put a unit is in the middle of the board so that if we get two same results, the bot will put a unit in the middle. An example of the table with prioritized middle can be seen in Figure 5.5



{	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	}
{	0	,	0	,	1	,	1	,	1	,	1	,	1	,	1	,	0	,	0	}
{	0	,	0	,	1	,	2	,	2	,	2	,	2	,	1	,	0	,	0	}
{	0	,	0	,	1	,	2	,	2	,	2	,	2	,	1	,	0	,	0	}
{	0	,	0	,	1	,	2	,	2	,	2	,	2	,	1	,	0	,	0	}
{	0	,	0	,	1	,	2	,	2	,	2	,	2	,	1	,	0	,	0	}
{	0	,	0	,	1	,	1	,	1	,	1	,	1	,	1	,	0	,	0	}
{	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	}

Figure 5.5: Visual priorities for the game unit.

The number in the table adds a certain number to the result. Now, all units will prioritize center over corners, Also, this approach would give us a possibility to place units before fighting right in the middle.



### 5.3.3 The use of MiniMax algorithm.

As discussed in previous chapters, for a game like this, it is necessary to find all possible outcomes and choose the best one (for the player and the bot). With minimax algorithm, we make a depth tree to analyze outcomes. Leaves give us the last evaluated number with formula bot's total health - player's total health. See the following example tree with depth 2 in Figure 5.6

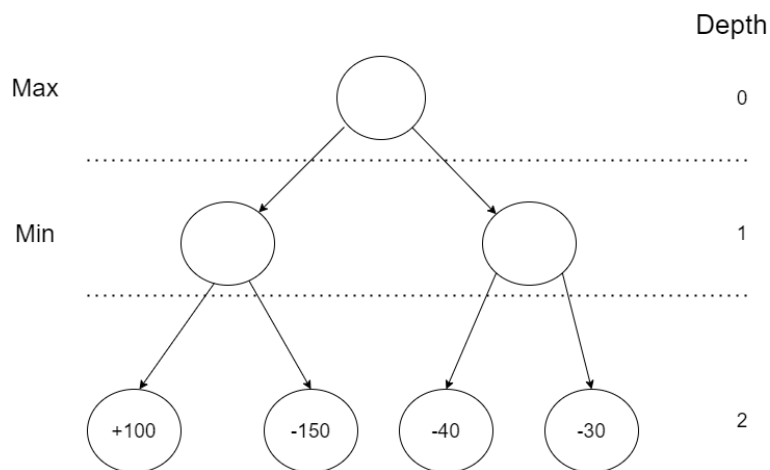


Figure 5.6: Leaves evaluation with depth 2.

In depth 1, we evaluate the best outcome for the player. He/she wants to minimize the total number. In the next step, we choose the best numbers for the player, -15 is less than 100 and -40 is less than -30, so we get a certain number and now we have in Figure 5.7:

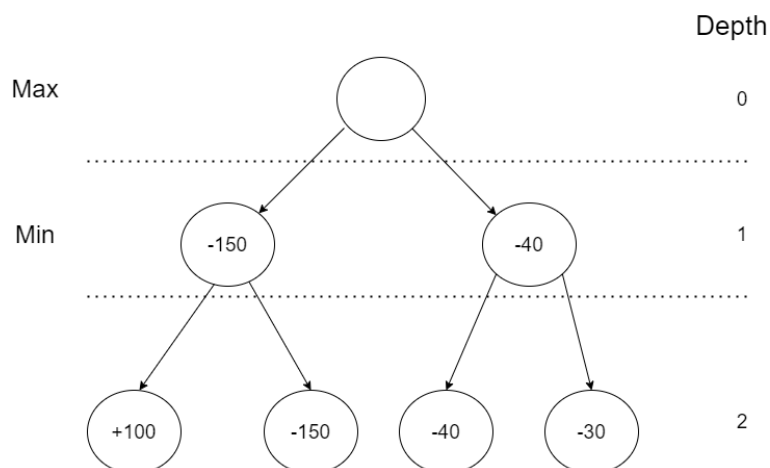


Figure 5.7: Tree evaluation for depth 1.

## 5. CREATION OF A NEW GAME

---

The last step is to choose the best outcome for the bot and maximize the value. In this case, the best outcome is -40, so the right action is the correct option to choose and go through for the best profit.

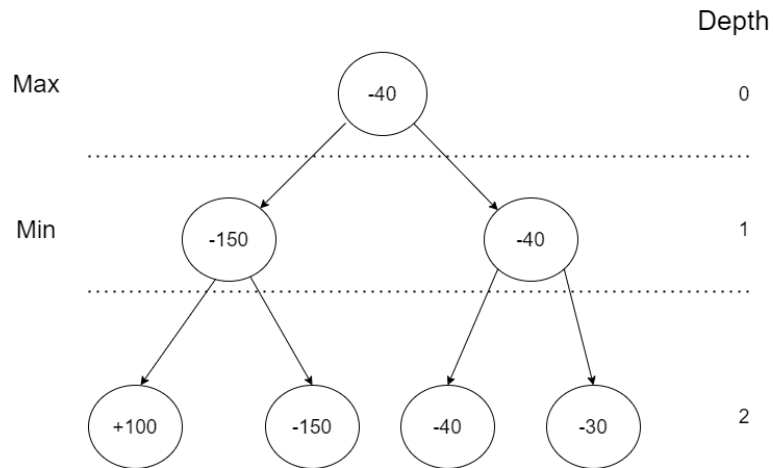


Figure 5.8: Tree evaluation for depth 0.

### 5.3.4 Alpha-beta pruning

For alpha-beta pruning, we specify two variables: alpha and beta. Initially, alpha is negative infinity and beta is positive infinity. Let's search through the tree from right to left. First, we would go to number -30 and -40 and put the best option to the parent, set up alpha as negative infinity and beta as -40.

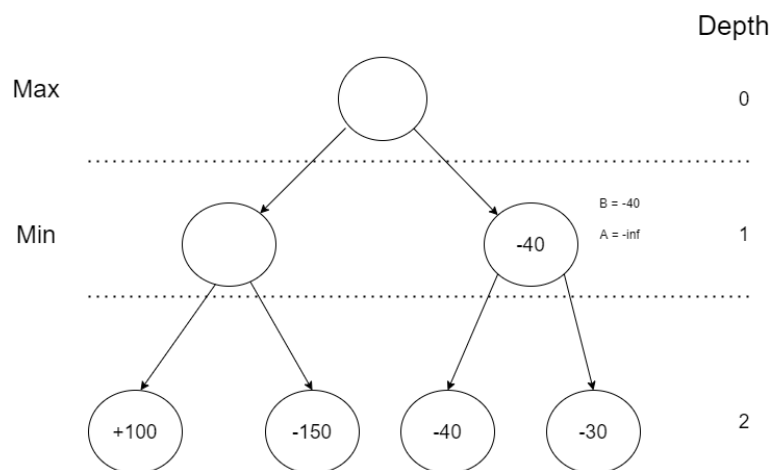


Figure 5.9: Tree evaluation for the left side.

Transfer this value to a parent and set its alpha to -40 and beta for infinity. Now, we transfer the alpha and beta of the parent in depth 0 to the other child (left one).

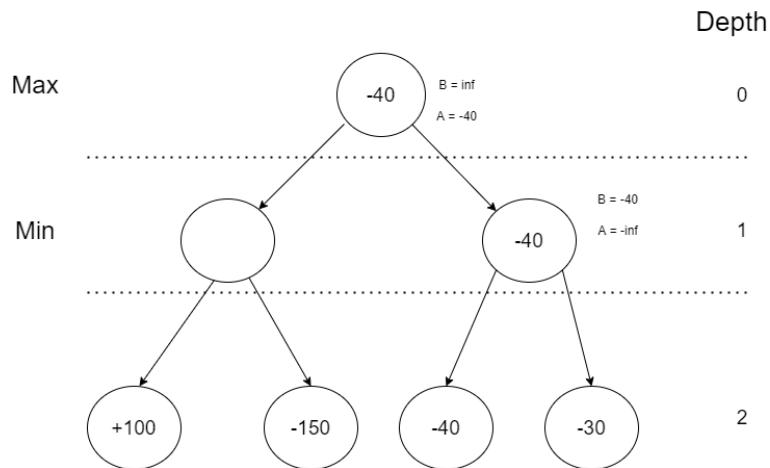


Figure 5.10: Tree evaluation for depth 0 after right side evaluations.

Then we go to another side of the tree and we see number -150. We know that the left side would lead us to a number that is less or equal to -150 which is less than our beta, so we can omit the further evaluation of the branch.

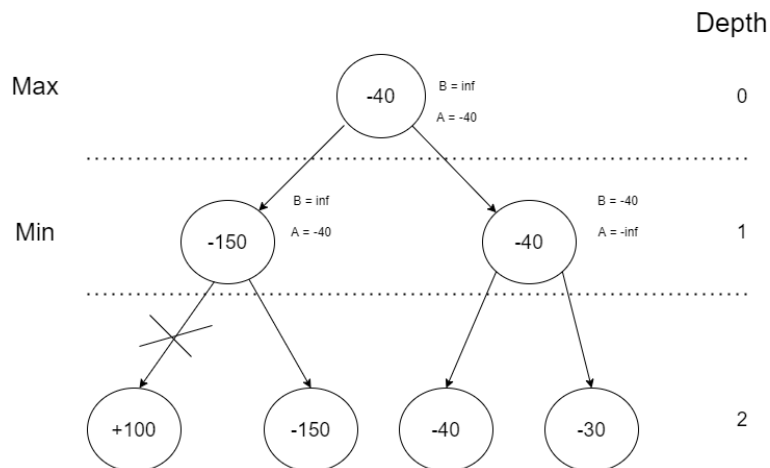


Figure 5.11: Tree evaluation for the left side of depth 1.

## 5.4 Testing

As mentioned before, the game was developed for three different difficulties: easy, medium and hard. Their difference is maximum depth. Easy mode has depth 2, medium mode has depth 4, hard mode has depth 8.



Figure 5.12: Difficulty selection in VR.

Consider the following situation: there are three units on the board. The left-most unit amount of 12 is Alien looking at right is the last unit for player A, the other two units with amounts of 22 (Erika) and 8 looking at left (Arissa), are the player B units.

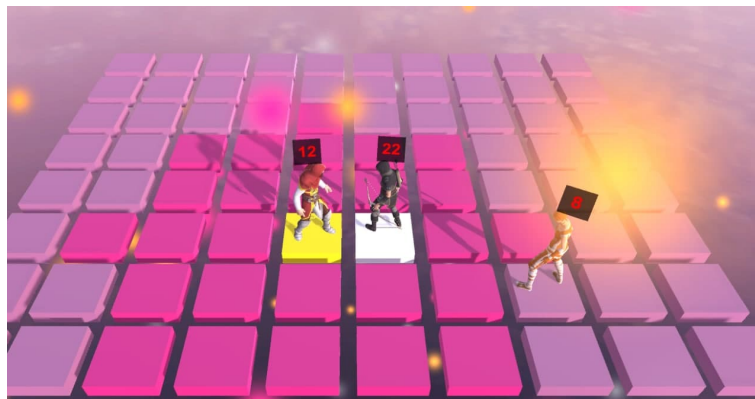


Figure 5.13: Board with units.

Overall evaluation of boards are

$$12 * 20 + 20 = 260$$

$$22 * 12 + 12 + 8 * 18 + 17 = 432$$

To remind, the damage is calculated with the formula amount of units times its attack. Alien has 20 damage, Erika has 5 damage, Arissa has 17.

Now, there two choices for player A, to Erika or Arissa. When a unit hits an enemy, the enemy responds with a hit.

The first option is to hit Erika, then she will receive damage in the amount of 240 leaving the amount from 22 to 3, then she responds with  $3 * 9$  damage leaving Alien with an amount of 10. Once this happens, then Arissa will go and hit Alien with damage equals 136 leaving the alien with an amount of 3, in this situation the player B can't lose.

The second option is first to hit Arissa. Then the amount of Arissa will become 0, Alien leaves with 12. Next time when Erika hits, she would hit in the amount of 110 receiving back 100 damage which leaves Erika with an amount of 10. At this point, player A always wins.



---

## Conclusion

The objective of this thesis was to create a turn-based strategy game with the smart bot and implement it for virtual reality.

Bots smartness was achieved with the use of minimax algorithm and simple improvements in logic. Virtual reality world was implemented in Unity engine with the board and playable units with certain mechanics.

As for future research, it would be a good idea to add some predefined moves as it is in typical chess games (openings) to make first moves generation faster. Also, for the virtual world, adding and changing the user interface with animations would make the game more enjoyable. Besides that, adding more mechanics to the game like hitting units from long distances or adding some spells would make the game more flexible.





---

## Bibliography

- [1] Deep Blue computer chess-playing system *written by The Editors of Encyclopedia Britannica*, url: <https://www.britannica.com/topic/Deep-Blue>
- [2] Mini-Max Algorithm in Artificial Intelligence, url: <https://www.javatpoint.com/mini-max-algorithm-in-ai>
- [3] Game Theory url: <http://people.maths.ox.ac.uk/griffit4/MathAlive/3/game-theory-notes.pdf>
- [4] Saddle points url: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/game2.html>
- [5] Deep Blue computer chess-playing system url: <https://www.britannica.com/topic/Deep-Blue>
- [6] Unity 3d documentation url: <https://docs.unity3d.com/Manual/index.html>
- [7] Unity Virtual Reality Development documentation url: <https://unity.com/unity/features/vr>
- [8] Oculus Go url: <https://www.oculus.com/go/>
- [9] Building chess AI url: <https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/>
- [10] Wiki page of Heroes of Might and Magic 3 url: <https://heroes.thelazy.net/index.php/Main-Page>
- [11] AI behavior of Heroes of Might and Magic 3 url: <https://heroes.thelazy.net/index.php/Difficulty-level>



## Acronyms

**VR** Virtual reality

**3D** Three-dimensional

**UI** User interface

**PC** Personal computer

**DFS** Depth-first search



---

## Contents of enclosed CD

<code>exe</code> .....	the directory with executables
├─ <code>EvilDreamsPC</code> .....	the directory with PC executable
├─ <code>EvilDreamsOculusGo</code> .....	the directory with Oculus Go executable
<code>src</code> .....	the directory of source codes
├─ <code>PC version</code> .....	the directory with PC project sources
├─ <code>VR version</code> .....	the directory with VR project sources
├─ <code>thesis</code> .....	the directory of $\text{\LaTeX}$ source codes of the thesis
<code>text</code> .....	the thesis text directory
├─ <code>thesis.pdf</code> .....	the thesis text in PDF format