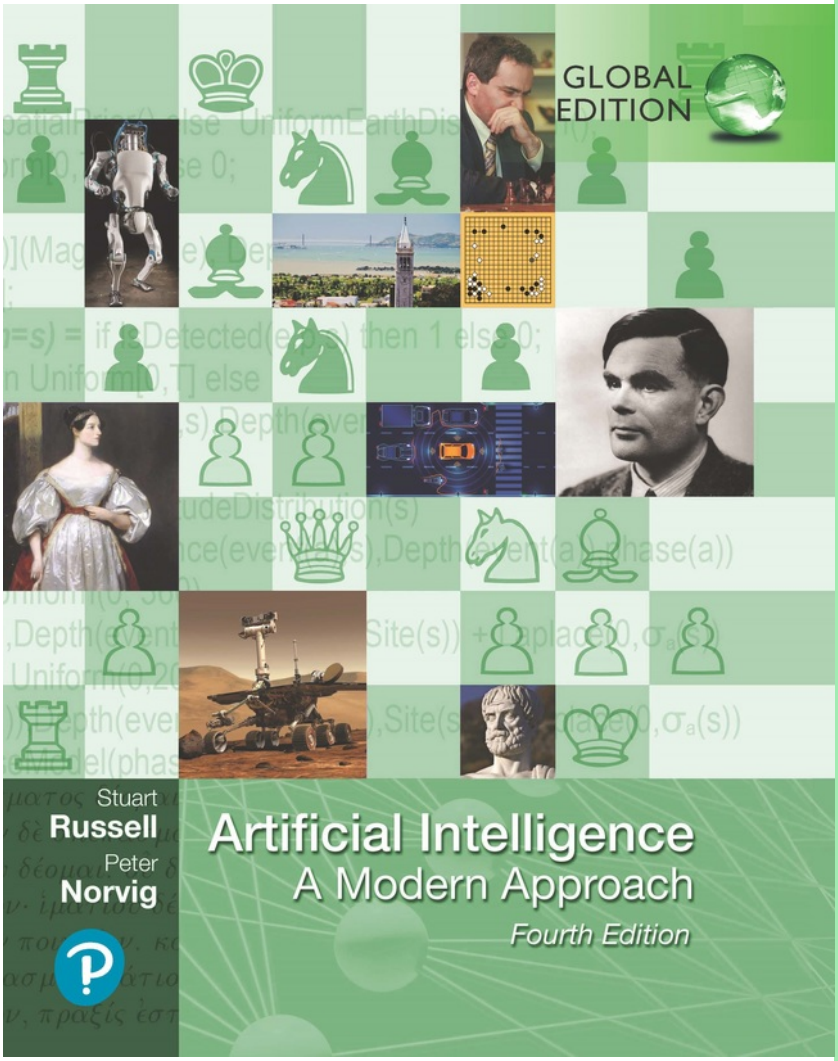
Assignment Two

CEN455
ARTIFICIAL INTELLIGENCE



Prepared By:

Rawan Ibrahim

21091000188

Prepared For:

Dr Musab ALRAWI

Submission Date: 23-12-2022

GROUP REPORT RUBRIC (10%)

Attribute		Poor	Below Average	Satisfactory	Good	Excellent	Score
Originality of Answers (30 marks)		Complete plagiarism	Minimal effort to indicate originality	Modifications into 'own' effort	Shows effort of producing originality work	High evidence of original work. Due credit is given to the original source	
		5	**10**	**15**	**20**	**30**	
Accuracy of Answers (30 marks)		Complete deviations from questions	Not enough points of answers	Acceptable effort	Moderate; answers are relevant to the questions	Very accurate; shows the link between questions and answers	
		5	**10**	**15**	**20**	**30**	
Flow of ideas (20 marks)		Shoddy work	Not well planned	Evidence of linkage between points of argument	The good linkage between points of arguments	Very well thought out	
		1	**5**	**10**	**15**	**20**	
Writing style (10 marks)		Poor expressions and full of jargons	Poor command of language	Uneven	Simple and easy	Good, understandable writings	
		2	**4**	**6**	**8**	**10**	
Grammar (10 marks)		Full of grammatical error	Too many of grammatical error	Acceptable level of grammatical error	Minimal grammatical error	No grammatical error	
		2	**4**	**6**	**8**	**10**	
Total **100** marks							

Index

Introduction

Before all else, we need to observe the evolution of the games over the past couple of decades. The basic, ordinary games we first knew of were like the deck of cards, dominoes, or monopoly. All of those games were played physically with cards, tiles, or a board, yet that became more reformed with the presence of artificial intelligence.

Nowadays, not only do we have those games on our devices, but also we have video games that are more responsive, real, and challenging. However, we will focus more on the mind-challenging games that we know for ages and how they evolved to be played *with* the computer rather than *on* it.

For all we know, artificial intelligence has its factors that create a system for it to be more metamorphosed for the good of the gaming experience as well as any other adaptive environment it creates. One of those factors is the data structure used in the game itself, like the diverse states that the player can go through as the game continues.

Another element is the multi-agent environment which is from the main characteristics of a simple game like dominoes. Through the following pages, some definitions and properties will be clarified as simply as possible to understand the role of the multi-agent environment.

Afterward, the searching problems will be presented under the shadow of what is called Adversarial Search, one of the most impressive types of searching in AI. Furthermore, the algorithms that functioned in dominoes will be discussed to get a small picture of the ability of the computer to defeat a human in a game. In the last section, we will discover more definitions regarding heuristics so that we figure out yet another way of how the computer response is as quick as we observe when we play.

Section 1: Domino Game

Domino is a game of 28 tiles that starts from a double-blank to a double-six-tile. The game can be played with a minimum of 2 and a maximum of 4 persons, however, in our case we will focus more on the 2-person rules; assuming one is the computer and the other is human. [1]

The game starts by shuffling the tiles, as their faces are flat down, and picking random 7 tiles for both of the players. The game starts with the player with the highest double tile, placing it on the ground. This tile is called the spinner and it is the only tile that can be surrounded by its four sides.

The game continues as the players try to place tiles with matching values and connect them with the tiles on the ground. In case one of the players does not have a matching value, they can draw a tile from the “boneyard”; which is the rest of the unused tiles that stay hidden from the players. If the player still does not have a playable domino, the other player shall take his turn. [2]

The players' goal is to leave all their tiles on the floor. Who has reached this target, is considered to be the winner of the game. Afterward, the phase of counting the points earned comes in turn. The winner sees if the ends of the chain’s tiles are a multiple of 5 (or sometimes a multiple of 3), if it is then the points are added to his score. Not to mention, the unplayed tiles of the other player will be added and rounded up to the nearest multiple of 5, then added to the winner’s score.

Link of the code of the game: https://onlinegdb.com/vgu9FZor0
Note that the code does not work on this compiler and needs to be imported.

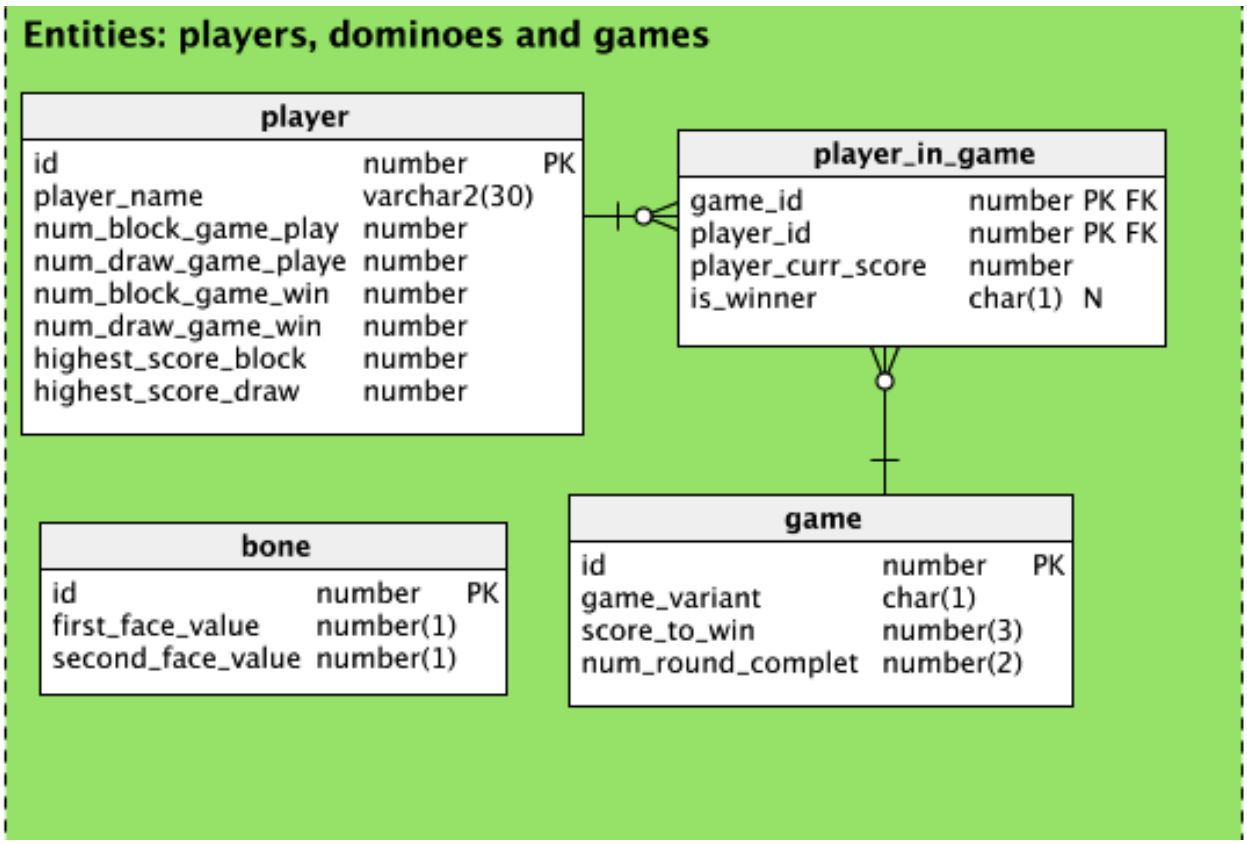
Section 2: Data Structure

In view of the fact that we want to dig deeper than just knowing the basic rules of Dominoes, data structure analysis is essential. For whatever system to be produced, it needs to be planned or rather “designed”. The data types of Dominoes are divided into two sections: the main entities and the side crucial details.

The Main Entities [3]

The main entities are the factors like the players, the domino tiles, and the game. The main entities are the inseparable components of any game to be complete for that matter. For instance, there is no game without players or tiles.

In order to make it more convenient, we subdivided this section into 4 tables: player, player_in_game, bone, and game. The player table is the one that carries the information of the players in the game and it is joined by the player-in-game table. Moving onto the bone table; which contains the details of tiles of the number of dots on both sides. Lastly, the game table is the one with the game’s details like the minimum score to win the game and the number of rounds progressed so far.

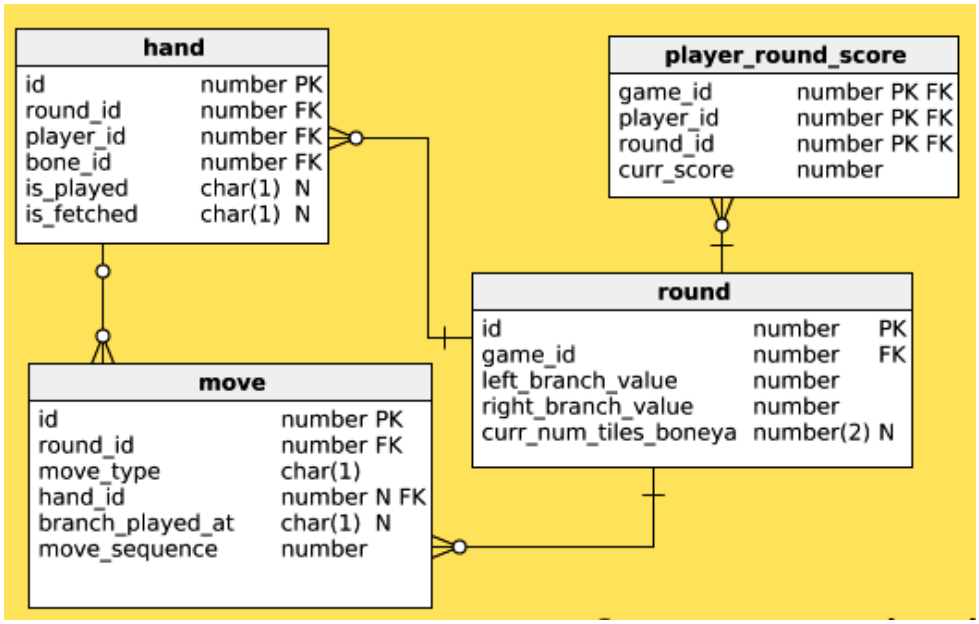


The Side-Crucial Details [3]

On the other hand, the second section of the data plan is absolutely effective due to the fact that it influences the chances of winning or losing the game. In this segment, the player_round_score, round, hand, and move tables will be explanatory enough to satisfy most of the elements needed to be considered for the winner to be declared.

First, the player_round_score table has information regarding the scores of each player per round. Also, it takes into account the game, player, and round ids as well as the current score of the players. Second, the round table is a phrasal statement of the current state of the round in terms of the positions of the tiles and the number of tiles in the boneyard. Third, the hand table is the table that discusses the tiles given to the players at the beginning of the round. This table is connected with the round, player, and bone ids along with the tiles played. Lastly, the move table, shows how the moves with the tiles progressed during the round. It is connected with the hand and round tables so it can use their ids to link it with the rest of the data stored as the move_type and the branch_played_at.

All in all, both sections are intertwined; since the data stored in each table, whether it is in the same or different section, need to be shared with other tables to form the simplest data structure of the game. For example, we have player and bone tables connected with the hand table, also game and round are linked, and of course, the player_in_game table is joined with the player_round_score table.



Section 3: Multi-Agent Environment

In artificial intelligence, there are some terminologies that need to be clarified in order to use them freely as we analyze the way of operation in any case provided. Starting with an “agent” which is the robot, human, system, or whatever senses and reacts at a given instance. [4]

The sensors are the gadgets used by the agents to sense the environment and decide the action it needs to perform afterward. The environment, however, is the agent’s surroundings that the actuators (also called effectors) respond to and affect. There are many types of environments, but the one we are most interested in is the multi-agent environment. [5]

Defining Multi-Agent Environment

The multi-agent environment is one where there are multiple agents operating in a synchronized technique to express their responses through the effectors. This kind of environment is commonly used with games for the reason that to win any game, various actions shall be decided. Well, we can ask how are these agents connected then. [6]

There are two cases of the multi-agents which are one decision maker and multiple decision makers. In the latter case, we have two assumptions of how the system works. The first one is that each agent has a distinct mission that needs to achieve, therefore, has different preferences for performing their actions. This leads us to the gaming theory, one of the most vital theories used in artificial intelligence; the strategic aspect of reasoning, with respect to the other players’ actions. [7]

Characteristics of Multi-Agent Environment

For the multi-agent environment to work and achieve its goal, there are some properties the agents must obtain. As a start, social skills are a must to connect with the other agents. Moreover, the sense of synchronization and reactivity is quite effective so all the agents can have neither delayed nor an early performance of any function. That’s why the planning stage is favorable in such a case.

Section 4: Search problems and Computer games

It is time to digest some facts that are yet to be understood concerning the computer games that we play almost daily. A simple computer game hides inside it numerous mathematical problems as well as obstacles that need to be passed using statistics. Needless to say that decision-making is a requisite for the game to keep processing the changing elements with each move and decide its next one thoroughly. Accordingly, the decision-making process requires an aiding feature, which leads us to discuss the search problems. [8]

Defining Search Problems

Search problems are problems that have an initial and final state as well as a search space. The solution to a search problem lies in applying a search algorithm that is responsible for finding a way to start from the initial state and end at the goal state. Those searching algorithms are used in the games as a primary component of the decision-making feature. Because artificial intelligence is always trying to mimic the human brain and thinking, these algorithms are counted as the technical solution for strategizing the next couple of steps in the game. This case can be referred as an Adversarial Search. [9]

Types of Adversarial Search [10]

Games can be divided into 4 categories, 2 are divided based on the information provided to the agents in the game which are: perfect information, and imperfect information games. Whilst the other 2 are depending on the stability of the rules of the game like deterministic, and non-deterministic games. Note that a game can be included in two categories like tic-tac-toe, which falls under the categories of imperfect information and deterministic games.

- Perfect Information Game: It is the one with the full information given to the agents, having absolutely no move covered or hidden during the game.
- Imperfect Information Game: It is one with incomplete details about what the opposite player can play
- Deterministic Game: It is the one with no randomness involved in the game and everything is applied according to the set of rules identified.
- Non-Deterministic Game: It is the last type that has randomness included, those games are also called games of luck. It depends on whatever the dice or the card gives to the player.

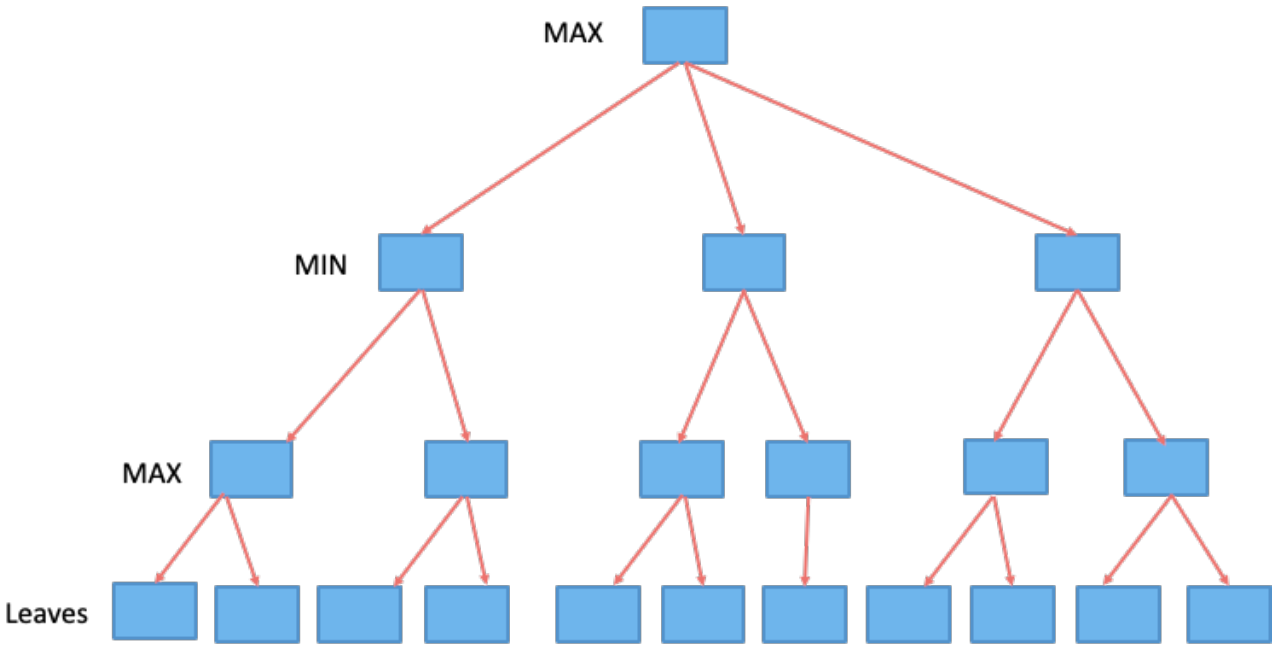
Section 5: Algorithms used in Dominoes

Zero-Sum Game

All computer games have algorithms that work to make the game as interactive as it can be in case the game is played by two humans. Dominoes is an example of what is called a zero-sum game. A zero-sum game is one where when a player wins, the other loses, and the net total becomes zero. [11]

Minimax Algorithm

One of the algorithms used in Dominoes is the Minimax searching algorithm where we will keep assuming that the game is played by only 2 players (MAX and MIN), one of them being the computer and the other one being human. The players take turns alternately starting with the MAX player playing the first move. MAX always maximizes the result and on the contrary MIN minimizes the result. [12]



So as MAX starts from the top, assume that it has 3 (might have more in real life example) moves that it can play. By computing each node the minimax value will be approached as the players reach the leaves, where the best achievable result will be reached. The minimax algorithm uses the depth-first search way using a game tree. Also, the leaves represent the terminal states of the game which is the state where the game ends.

Section 6: Heuristic Functions in Gaming Agents

Games like chess and dominoes and all the others use algorithms like the search algorithms mentioned in section 4 and more specified algorithms like in section 5. However, those are not enough for the game to be complete, not enough to reach its goal easily and quickly. As we wait for the computer to make its move in a game, it continuously searches for the optimal and safest move, yet it needs to be quick or we can close the game. That is what the heuristics are for. [13]

What is a heuristic function?

The heuristic function is a method that approaches an available decision within a fair duration. This means that heuristics don't always give the best option because they prefer focusing on giving a quick output. [14]

What is heuristic evaluation? [https://www.geeksforgeeks.org/heuristic-evaluation/]

The heuristics are used as a sort of evaluation to test and evaluate the user experience by taking in-depth usability of the system as a measure. The main target of the heuristic evaluation is to catch the hidden bugs in the system and provide approaches to solve them. Also, this stage can take place on any part of the design timeline. [15]

Conducting heuristic evaluation

The keys to conducting heuristic evaluation start by determining the scope of the evaluation, and defining the parameters that need to be checked as well as the deadline of the evaluation. The next key is acknowledging the end user since the system will be targeting its interests. Setting the heuristics is also quite vital to give the evaluation a direction of guides. Not to mention, summarizing the results of the evaluation is essential and helpful when it comes to solving the issues found. [15]

T*he difference between heuristic evaluation and usability testing*

Even though it may seem that heuristic evaluation is akin to usability testing, there are some differences that remain clear. For instance, heuristic evaluation usually supports detecting hidden issues, however, usability testing detects the real vital issues in the system. All in all, the usability testing functionality is more generalized than heuristic evaluation, therefore it is more pricey. [16]

Now we need to move a little deeper than understanding the concept of the heuristic functions theoretically and make a quick, more sophisticated, review. But before heading straight into the core, again, some terminologies need to be interpreted. One of the most important definitions will be related to the states of a computer game. [17]

Game States

A game is standardly established by the following states:

- S_0: The starting state, which specifies how the game is handled at the beginning.
- TO-MOVE(*s*): The player whose turn it is to move in *s*.
- ACTIONS(*s*): The set of legal moves in *s*.
- RESULT(*s*, *a*): The transition model, which defines the state resulting from taking *a* in *s*.
- IS-TERMINAL(*s*): A terminal test, which is true when the game is over and false otherwise. States, where the game has ended, are called terminal states.
- UTILITY(*s*, *p*): A utility function(also called an objective function or payoff function), which defines the final numeric value to *p* when the game ends in terminal state *s*. Some games have a wider range of possible outcomes.

Where **s** is a state, **a** is an action, and **p** is a player.

Heuristic Evaluation Function

As the computer tries to come up with a quick evaluation of the chances of winning, it uses the heuristic evaluation function in something called heuristic alpha-beta tree search. Its main concept is to replace the UTILITY function with a new function named EVAL. EVAL is the estimation of a utility's state. This function is used in what is called a Cutoff Test which is another replacement for a terminal test that is carried out with the help of the H-MINIMAX formula. [18]

H-MINIMAX(s, d) for the heuristic minimax value of state s at search depth d:

H-MINIMAX(s,d) =

{ EVAL(s,MAX) if IS-CUTOFF(s,d)

$\max_{a \in Actions(s)}$H-MINIMAX(RESULT(s,a),d+1) ifTO-MOVE(s)=MAX

$\min_{a \in Actions(s)}$H-MINIMAX(RESULT(s,a),d+1) ifTO-MOVE(s)=MIN }

Conclusion

To wrap it all up, the games that we play hold a deeper and more trailblazing than we might have thought. It carries out thousands and maybe millions of calculations and algorithms just to simulate a single move. This is all because the basic identity that artificial intelligence is trying to obtain is the rationality and the common sense of humans, which in our case the players use.

For all that is aforementioned, it stays unsatisfactory for a game to be created and the computer to be considered a good player. Accordingly, there are lots and lots of other algorithms that are used that we didn't mention. Some of them are complementary to what we discussed whereas others might be distinctive and have way more improved ways of reaching their target.

To prove how far artificial intelligence has reached in the gaming field, there is a business career opened for those who take the games on a serious level called "Professional Gaming" or "Esports". That shows us how games are more than just some entertainment activity for some people, it became some people's job to be professional gamers. Winning tournaments nationally and even internationally is the leading way through this career and it gets more and more challenging by the day as AI keeps developing itself making the game harder for the players. [19]

It is not so hard to predict that artificial intelligence will not stop just here, whether in the gaming field or any other one. The rate we have achieved so much with the aid of AI is quite impressive and even scary for some. Our minds cannot even process what can be become next with AI working this well, but only one thing is certain. AI can be the perfect opponent in a computer game!

References

Section 1 References:

1. https://www.dominorules.com/the-basics
2. https://www.mastersofgames.com/rules/dominoes-rules.htm

Section 2 References: 3. https://vertabelo.com/blog/the-secrets-of-dominoes-or-a-domino-game-data-model/

Section 3 References:

4. https://www.javatpoint.com/agent-environment-in-ai
5. https://www.geeksforgeeks.org/types-of-environments-in-ai/
6. [Artificial Intelligence: A Modern Approach Global 4th Edition, Chapter 17.1, Pages 589-594]
7. https://www.intechopen.com/chapters/56573

Section 4 References:

8. https://www.section.io/engineering-education/understanding-search-algorithms-in-ai/
9. https://techxplore.com/news/2021-05-games-mind-algorithms-game.html
10. https://www.javatpoint.com/ai-adversarial-search

Section 5 References:

11. https://marketbusinessnews.com/financial-glossary/zero-sum-game-definition-meaning/
12. https://www.javatpoint.com/ai-adversarial-search

Section 6 References:

13. https://www.techopedia.com/definition/5436/heuristic
14. https://www.baeldung.com/cs/heuristics
15. https://www.geeksforgeeks.org/heuristic-evaluation/
16. https://www.geeksforgeeks.org/difference-between-usability-testing-and-heuristic-evaluation/
17. [Artificial Intelligence: A Modern Approach Global 4th Edition, Chapter 6.3, Page 202]
18. [Artificial Intelligence: A Modern Approach Global 4th Edition, Chapter 6.1.1, Page 193]

Conclusion References: 19. https://www.uopeople.edu/blog/how-to-become-a-professional-gamer/