2023

5(1), 01-19

İki Boyutlu Video Oyunlarında Yapay Zekâ (YZ) Tabanlı Kendi Kendine Karar Veren Karakter Geliştirme Uygulaması

Hakan Aydın*[1], Akif Yasin Ayas[2], Ali Çentinkaya[3], Zafer Güney[4]

Anahtar Sözcükler
Video Oyunu Geliştirme
Prosedürel Düzeyde Üretim
Yapay Zekâ
Unity2D

Makale Hakkında
Gönderim Tarihi
11 Mayıs 2023
Kabul Tarihi
26 Haziran 2023
Yayın Tarihi
28 Haziran 2023
Makale Türü
Araştırma Makalesi

Öz

Hayatın hemen her alanında kullanılmaya başlayan Yapay Zekâ (YZ) teknolojileri artık oyunlarda da kullanılıyor. Özellikle son zamanlarda birçok popüler video oyununda oyunu otomatik olarak oynayan bot sistemler için bu botların oyunun büyük ustalarını gelişmiş Derin Öğrenme (DL) yöntemleriyle yenmeleri mümkün hale geldi. Bu çalışmanın amacı, oyun karakterlerinin yetenekleri dahilinde ne yapacaklarına karar verebilen, Unity 2D Game Engine'de YZ tabanlı iki boyutlu bir video oyunu geliştirmektir. Oyun uygulamasında iki boyutlu oyun karakteri kullanıcı tarafından kontrol edilmektedir. Bu karakter, prosedürel düzlem oluşturma algoritmaları kullanılarak oluşturulan iki boyutlu bir oyun düzleminde hareket eder. Ayrıca oyunda mağlup edilen düşman sayısı ve ulaşılan seviye gibi istatistikler de kayıt altına alınmaktadır. Oyun Unity2D oyun motoruna dayalıdır ve C# ile yazılmıştır. Çalışmada sekiz farklı deney gerçekleştirilmiştir. Bu deneylerde oyuncunun karakter için tanımlanan farklı ekipmanlarla oyundaki başarı süresi ölçülmektedir. Bu süre 0,54 saniye ile 1,88 saniye arasında değişmektedir. Günümüzde bilgisayar oyunlarını daha iyi oynamak veya daha iyi oyunlar tasarlamak için YZ algoritmaları geliştiriliyor ve YZ algoritmalarını geliştirmek için yeni oyunlar tasarlanıyor. Bu bağlamda, bu çalışmanın YZ'nın bilgisayar oyunlarında kullanımına ilişkin literatüre katkı sağlayacağı değerlendirilmektedir.

Artificial Intelligence (AI)-Based Self-Deciding Character Development Application in Two-Dimensional Video Games

Keywords
Video Game Development
Procedural Level Generation
Artificial Intelligence
Unity2D

Article Info
Received
May 11, 2023
Accepted
June 26, 2023
Published
June 28, 2023
Article Type
Research Paper

Abstract

Artificial Intelligence (AI) technologies, which have started to be used in almost all areas of life, are now also used in games. Especially for bot systems that play the game automatically in many popular video games recently, it has become possible for these bots to defeat the grandmasters of the game with advanced Deep Learning (DL) methods. The purpose of this study is to develop an AI-based two-dimensional video game in Unity 2D Game Engine, which can decide what the game characters will do within their abilities. In the game application, the two-dimensional game character is controlled by the user. This character moves in a two-dimensional game plane that is created using procedural plane generation algorithms. In addition, statistics such as the number of enemies defeated and the level reached in the game are recorded. The game is based on the Unity2D game engine and is written in C#. In the study, 8 different experiments are performed. In these experiments, the player's success time in the game with different equipment defined for the character is measured. This time ranges from 0.54 seconds to 1.88 seconds. Today, AI algorithms are developed to play computer games better or to design better games, and new games are designed to improve AI algorithms. In this context, it is evaluated that this study will contribute to the literature on the use of AI in computer games.

Atıf: Aydın, H., Ayas, A. Y., Çetinkaya, A., & Güney, Z., (2023). İki boyutlu video oyunlarında yapay zekâ (AI) tabanlı kendi kendine karar veren karakter geliştirme uygulaması. *Bilgi ve İletişim Teknolojileri Dergisi*, 5(1), 1-246. https://doi.org/10.53694/bited.1247338

Cite: Aydin, H., Ayas, A. Y., Cetinkaya, A., & Guney, Z., (2023). Artificial intelligence (AI)-based self-deciding character development application in two-dimensional video games. *Journal of Information and Communication Technologies*, 5(1), 1-246. https://doi.org/10.53694/bited.1247338

***Sorumlu Yazar/Corresponding Author**: hakanaydin@topkapi.edu.tr
[1] Assoc. Prof. Dr., Istanbul Topkapi University, Faculty of Engineering, Istanbul, Turkey, hakanaydin@topkapi.edu.tr, https://orcid.org/0000-0002-0122-8512
[2] Istanbul Gelisim University, Department of Computer Engineering, Istanbul, Turkey, ayasysn@gmail.com, https://orcid.org/0000-0003-3529-3221
[3] Lect., Istanbul Gelisim University, Department of Electronics Technology, Istanbul, Turkey, alcetinkaya@gelisim.edu.tr, https://orcid.org/0000-0003-4535-3953
[4] Asst.Prof.Dr., Istanbul Topkapi University, Faculty of Engineering, Istanbul, Turkey, zaferguney@topkapi.edu.tr, https://orcid.org/0000-0003-1974-4264

Introduction

Artificial Intelligence (AI) aims to help machines behave clearly. One of the most important sectors of artificial intelligence development today is the computer gaming industry. The rapid development of technology has positively impacted the growth of the digital gaming industry. The digital gaming industry has become a creative industry arm that needs to be addressed in a large ecosystem (Stewart & Misuraca, 2013). Today, Artificial Intelligence (AI) technology is used in the field of computer and video games, as in every field, for purposes such as learning the tasks and stages of the game character. AI technologies, which have started to be used in almost all areas of life, are now also used in games. Especially for bot systems that play the game automatically in many popular video games recently, it has become possible for these bots to defeat the grandmasters of the game with advanced Deep Learning (DL) methods. Among the reasons for the use of AI in computer games is the acquisition of the ability of the computer or smart device to respond to the player skillfully and intelligently, plan throughout the game, develop moves, and beat the opponent. Today, with the use of AI in game technologies, it is possible to develop computer games that can react faster than humans and make high-level tactics. In the case of using AI in a computer game, all events that occur outside the player while the player is playing the computer game are processed with this technology. With AI, events such as passing a character while playing a game or enemy soldiers seeing and firing at the player in a war game are provided. Enemy soldiers who are constantly hit on the head in a war game can see this situation and can come by wearing stronger helmets with AI in the following sections. Another example of using AI in games is that the game offers a more challenging game to the players by automatically increasing the values of the opposing team players according to the player's good performance in the football match game. AI can also provide a computer with these and similar capabilities, as well as new capabilities in graphic design. Unlike the games played against computers and where the opponent is not very smart, it has become possible to create game characters who improve themselves and learn new tactics in every game with AI. Many new features can be added to games by using AI algorithms. In the case of using AI in a computer game, all events that occur outside the player while the player is playing the computer game are processed with this technology. With AI, events such as passing a character while playing a game or enemy soldiers seeing and firing at the player in a war game are provided. Enemy soldiers who are constantly hit on the head in a war game can see this situation and can come by wearing stronger helmets with AI in the following sections. Another example of using AI in games is that the game offers a more challenging game to the players by automatically increasing the values of the opposing team players according to the player's good performance in the football match game. AI can also provide a computer with these and similar capabilities, as well as new capabilities in graphic design. In the future, AI technologies will be used more in the development of computer games. There are many different AI techniques and methods used in the video game industry today. It can be a simple FSM algorithm or an advanced neural network that learns from feedback.

The digital games industry, which has a history of almost half a century, has become an industry of interest to many investors and developers, and a major source of revenue due to the rapid growth in the number of users and manufacturers and the increasing hardware developments in this field worldwide. As this industry falls within the scope of many sectors such as the software sector, the creative and design sector, and the hardware sector, it has now become a major industry thanks to the rapid development of both the hardware and software sectors and the desire of people to make a career in this field. This industry has also grown with the technologies developed in recent years (Tylor, 1879; Denning, 2021). Like any industry, the digital gaming industry is in flux and evolving.

In addition to the increasing desire to play games, the desire of independent producers to participate in the development of games also plays an important role in the rapid development of the digital game industry. The software department of the video game industry, which consists of elements such as game makers, game designers, sound designers, graphic designers, and story designers who develop a game application, as well as the platforms such as consoles, computers, and phones that run these games, and the hardware department that develops the hardware of these platforms, this sector covers a large economic area (Zackariasson, Wilson, 2012; Kim & Kang, 2021).

Video game is an entertainment and leisure activity software built on computer-based, text, or visuals that one or more people can use together on electronic platforms such as a computer or game console over a physical or online network (Frasca, 2001). Games that were drawn on the wall with chalk and games that were played with skipping stones have been replaced by various games such as chess, and these games have been replaced by games that are played in the digital environment with the development of computer technology today. Games developed with the evolution of technology under the name of research and iterative design provide players with visual and strategic content as well as in-game character development content; studies on health, tourism, entertainment sector, virtual world, and history lessons emerge. These studies help to give players an idea of related fields by assigning roles to players (Evans et al., 2021; Mochocki, 2021; Justesen, 2019; Chen et al., 2018; Hanes & Stone, 2019). One of the main reasons for these developments is the decrease in the cost of digital game development with the evolution of technology and the proliferation of helper applications and libraries in the field (Prato et al., 2014). The applications that contain the libraries and components used in game design in their entirety are called game engines. Besides game engines such as Cryengine, Frostbite, Source, 4A Engine, there are game engines produced by large companies for their own use. Unreal Engine, Unity, Godot, and GameMaker are the most commonly used game engines by small companies and independent producers. The historical development process of the digital gaming industry includes the early development phase before 1980, the growth phase between mid-1980 and mid-1990, the development phase until the end of 1990, the maturity phase of 2000-2005 and the progress from 2005 to present (O'hagan & Mangiron, 2013).

The purpose of this study is to develop an AI-based two-dimensional video game in Unity 2D Game Engine, which can decide what the game characters will do within their abilities. In the computer game we developed within the scope of the study, AI abilities were used in order for the game characters to survive as long as possible, to use the most advanced equipment, to successfully complete various stages of the game, and to develop more advanced game tactics. Among the main contributions of our study are the following points:

- An AI-based character-oriented 2D game is presented.
- To prove the concept, the game was designed and implemented with AI. The implementation has been done successfully with C# in Unity 2D Game Engine.

The hypothesis of our study was determined as AI-based decision-making methods can increase the survival times of game characters and enable them to develop more advanced game tactics. We evaluate that the following results we obtained from our study confirm our hypothesis. Namely, the 2D game developed in our study aims to increase the survival times of game characters and enable them to develop more advanced game tactics using AI-based decision-making methods. The successful implementation of the game in C# language on Unity 2D Game Engine

supports the accuracy of the hypothesis. Additionally, receiving positive feedback from users of the developed game is also a factor that confirms the hypothesis.

Our study confirmed the hypothesis that AI-based decision-making methods can enhance the gameplay experience by increasing the survival times of game characters and enabling them to develop more advanced game tactics. Our 2D game, developed using Unity 2D Game Engine and C# language, successfully demonstrated the feasibility of integrating AI into game development. Moreover, the positive feedback from the users of the game supports our hypothesis, indicating the potential of AI-based games to provide a more immersive and satisfying gaming experience.

The development of AI-based video games has gained significant attention in recent years due to the potential of AI to enhance gameplay and provide a more immersive experience for players. AI-based games can provide players with more challenging opponents and dynamic gameplay that adapts to their actions, which can lead to a more satisfying gaming experience. The development of an AI-based 2D game can contribute to the advancement of the field by demonstrating the potential of AI to enhance gameplay and create more sophisticated gaming experiences. The use of AI in game development can also lead to the development of new game design principles and techniques that can be applied to other fields beyond gaming. This study contributes to the field by presenting a new approach to developing 2D games that incorporates AI-based decision making for game characters. The successful implementation of the game using C# in Unity 2D Game Engine demonstrates the feasibility and potential of using AI in game development. The presented game can serve as a reference for other game developers interested in incorporating AI-based decision making into their games. The study can also contribute to the broader research on AI-based game development by providing insights into the challenges and opportunities of using AI in this context.

This research is organized as follows: In the Related Studies section of the article, findings obtained from previous research on the topic are presented. The Material and Method section includes details about the study. The findings are discussed in the Results and Discussion section of the paper. Finally, the Conclusion section presents the results obtained in the study.

Related Studies

An example of the use of AI is a game designed to play chess (Shannon, 1950). The "Nim" a two-player game, can be shown among the first examples of the use of AI in a game. This game is a mathematical strategy game that has been played since ancient times. In this game, one of the players removes at least one or that object and other objects around it from the game. The player who pulls the last object on the ground loses this game. In 1951, a checkers program was written by Christopher Strachey and a chess program by Dietrich Prinz using a machine called The Ferranti Mark 1 (Manchester Electronic Computer). Launched at Carnegie Mellon University in 1985, the Deep Thought project was a chess-playing computer that was meant to beat even professional players. In 1996, Deep Blue defeated then world champion Garry Kasparov in one of six matches. In this event that shook the world, the main actor was AI. Examples of the use of AI in games are the text-based Hunt the Wumpus and Star Trek games developed in 1972. AI was also used in games such as Speed Race and Qwak, which were released for arcade platforms in 1974. Developed in 1978, Space Invaders used AI technologies with advanced difficulty levels, different motion patterns, and hash function-based in-game activities using player inputs. Games such as Galaxian game developed in 1979, Pac-Man developed in 1980, Karate Champ developed in 1984, Dragon Quest IV

developed in 1990 can be given as examples of the use of AI. In computer games such as Creatures and Black & White, which were developed in 2005, AI methods have begun to be used for situations such as determining player movements. AlphaGo is a game by Google that uses AI. This game is a program that plays the “go game” developed by Google DeepMind. In October 2015, it became the first computer program to beat a professional “go player” on a 19x19 board without being given an advantage. Chen, (2016) states that AlphaGo beats the most diligent and deeply intelligent human brains. Released in 1992, the Wolfenstein 3D game was considered one of the greatest video games ever made. In the game, which was designed on a concept where you would be a shooter, the soldiers had a basic form of the FSM (Finite-state machine) algorithm. In this algorithm, the designers created a list of all possible events a bot could experience. At that stage, the team had to deal with an enemy military boat being shot from behind, aiming, etc. He acted by considering all the possibilities. Then, they programmed the behavior of the bot according to this list they compiled. The developers of the game industry are trying to make the AI behave like a human and create a game world from scratch without the need for a real human being. A scene from the computer game "Go" is shown in Figure 1. A critical point regarding the use of AI in the game in question is shown in this Figure. Accordingly, if black places a piece at X, its area is alive. And if white places a piece at X, black stones will eventually be removed

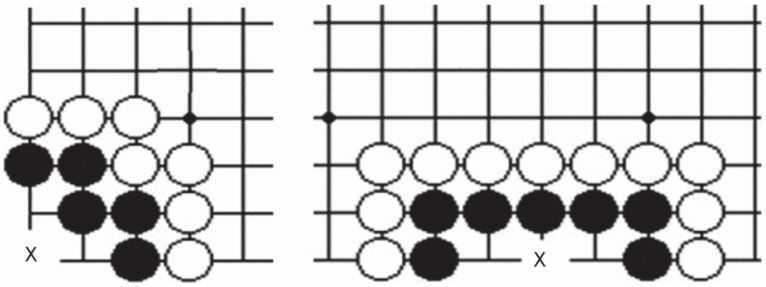

Figure 1. A scene from the computer game "Go" (Chen, 2016)

Serious game AI functionalities include player modeling (real-time facial emotion recognition, automated difficulty adaptation, stealth assessment), natural language processing (sentiment analysis and essay scoring on free texts), and believable non-playing characters (emotional and socio-cultural, non-verbal bodily motion, and lip-synchronized speech), respectively (Westera et al., 2020). The recently launched “gamecomponents.eu” portal funded by Horizon2020 is the technical platform for exchanging advanced game technologies and associated resources (Chen, 2016). Safadi et al. (2015) suggests an approach based on the use of a unified conceptual framework to enable the development of conceptual AI which relies on conceptual views and actions to define basic yet reasonable and robust behavior. Pirovano (2012) introduced the field of game AI and review the use of fuzzy logic in games, looking both at industry and research. There are many free tools on the market for independent producers who cannot find financial support for their productions, such as large corporations. Game engines can be cited as an example of one of these tools. Although they seem to be free at first glance, these game engines have a hidden and mandatory fee. For example, Unreal Engine, developed by Epic Games, requires 5% of independent production income, Source Engine, developed by Valve, requires $25.000, and Unity Engine, developed by the Unity team, requires a purchase above a certain income (Parlan, 2017; Westecott, 2013). In addition, market applications such as Steam, Epic Store, Google Play Store, where independent game publishers offer their games, also require their own shares. While it is very easy to buy and consume a game, it is difficult and costly to make it. Nowadays, video games are no longer technologies developed for entertainment purposes,

but economic goods (Kepenek, 2018). This is the price independent producers have to pay for their independence. In the computer environment, nothing is completely random. Behind every calculated randomness is a logically executed algorithm. In this study, a random motion algorithm, called the Drunken Man algorithm, is used for random partition generation algorithms (Deghhani, 2019). Fuzzy logic, artificial neural networks, genetic algorithms, and deep learning methods are used in the interaction of the player character and other game elements with both itself and the game environment (Tunç et al., 2020; Chivukula et al., 2018; Costa et al., 2019; Pfau et al., 2020; Lohokare et al., 2020). In the stud of Çetin & Sarıca (2020), a level identification procedure was performed with different methods and different parameters. This classification process was tested on the users of Renga, an infinite game.

Material and Method

Video games are electronic games that can be played on a device such as personal computer, game console, smartphone or tablet (Haaranen et al., 2014). These games are potentially not only a recreational tool as an increasingly popular activity, but also as a workspace (Palaus et al., 2017). These are applications designed by game developers for people to spend their time on consoles such as Atari and Commadore64 in the early years of technology and today on many platforms such as mobile phones, computers, and new new-generationgeneration game consoles (Lee et al., 2014; Clarke et al., 2014). The video game industry has grown rapidly depending on the development of the hardware and software industries. Video games are often divided into different genres. These are FPS, TPS, RPG, RTS, and MOBA. FPS; 3D games experienced from a first-person perspective. It is one of the most common types of digital games today. It is usually played online against other players. In addition to computers, this type of game can also be played on mobile phones thanks to gaming consoles and today's advanced technology. Games like Battlefield and Call of Duty are examples of this genre.

Game engines combine all the physical mechanisms, graphics components, libraries, and scripts (pieces of code) into one game; they are the most basic programs that control the input and output in the game. Large companies typically write a dedicated game engine for each game they develop, allowing them to fully master the game mechanics and operate at full efficiency. Small companies or indie developers usually use publicly available game engines that can be used universally. The convenient interface of game engines offers a lot of comfort especially to the producers who are trying to develop games independently from a big company. The game engine used in a game depends on the needs of the person or company developing the game (Bishop et al., 1998).

The most basic function of game engines is to provide input from the user. These input devices vary depending on the platform for which the game is written. General-purpose game engines adapt all of these input libraries so that the game maker does not have to write optimized code for each input device. One of the most important tasks of game engines is to create game graphics and draw them. They help translate two or three-dimensional models into the game. The job of the game engine is to optimize the quality of the graphics, the game performance, and the resources consumed. Another important function of game engines is physics engines that simulate physical effects such as acceleration, torque, and gravity in the game. Thanks to these pre-built physics formulas, the game developer does not have to create his own physics formulas for each object in the game. Game engines also design the interface between the game and the user. All text, buttons, menus, backgrounds, effects, etc. in the game are contained in this area.

In addition to game engines such as Cryengine, Frostbite, Source, 4A Engine, there are game engines made by large companies for their own use. Unreal Engine, Unity, Godot, and GameMaker are the most commonly used game engines by small companies and independent producers. Unreal Engine is a game engine written in the C++ programming language, developed by Epic Games in both paid and free versions. The biggest feature that distinguishes it from other game engines is that it can produce works with very realistic graphics quality. It is the game engine preferred by most advanced games in terms of 3D and cinematography. Unreal Engine is preferred by independent producers as well as large companies such as Ubisoft and Activision (Sanders, 2016). Godot is a game engine developed by two independent developers in 2014. It is used in 2D and 3D games. The ability to develop applications for many platforms may be one of the reasons for its preference. It uses a proprietary programming language called GDScript. Godot is mostly used by small companies or independent developers. Unlike other game engines, having its own programming language allows producers to better master the basic functions of the games created with this engine.

GameMaker Studio is a game engine developed by YoYo Games for 2D game development, supporting C++, C, Delphi, and C# languages. It supports many platforms, from new generation game consoles to smartphones. Because it is specifically designed for 2D games and supports cross-platform games, it has often been the preferred game engine for these types of game developers. However, due to its complex user interface and the lack of a free version, it is only used by advanced game developers (Moreno, 2020). Unity is a game engine that is mostly preferred by independent developers or small businesses. It uses the C# and JavaScript programming languages. The simple and useful user interface has made this game engine one of the most widely used game engines. It offers support for many different platforms. The biggest advantage of the Unity game engine is that it can create relatively high-quality applications with very few resources. 2 or 3-dimensional applications can be created (Haas, 2014). While the software is planned in the testing process, the corresponding testing activities should also be planned. The software development phases should be followed within the software, and a game application should be developed in accordance with the phases. To test the results of the study, scenarios should be created and suggestions and analysis should be incorporated into the software by soliciting feedback (Calp et al., 2018; Inal & Güner, 2015).

For this study, the game engine Unity2D, one of the most popular game engines, is used. Unity is one of the most popular game engines thanks to its ease of use and comprehensibility. Unity2D is a compiler that imports many different libraries and components to easily create a 2D world through the interface. Also, thanks to the Unity engine, it is possible to change the platform on which the game will be released without changing the fundamentals of the game. For example, a game developed for the Windows platform can be transferred to other platforms, especially mobile devices such as Android or iOS, without any changes to the infrastructure. In the study, C# is used as the programming language and VisualStudio as the compiler. Most of the graphic and sound elements used in the study are elements sold/given for free on the Internet and in the official Unity store. Some graphical elements are developed specifically for this study.

Types of Graphics

The types of games in which the game environment changes and takes place around the game character are called character-oriented games. In character-oriented games, the game is designed around the game character in the narrative and mechanical ways (Bostan et al., 2019). One of the most basic ways to achieve this is for the game

character to develop itself and the game as it progresses through the game by communicating with the game elements. Such game development methods have spawned many sub-genres of games, such as RPG and Roguelike, which are referred to as role-playing games. Today, character-driven game elements can be found in almost all RPG games, as well as many other types of games.

Character development varies from game to game and can take a number of forms. In this study, the player character can improve his skills and strengthen himself with the various pieces of equipment he uses thanks to his experience from the enemies defeated throughout the game. These equipages are shown in Figure 2.

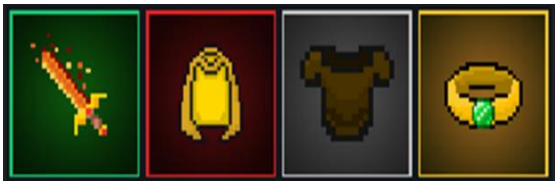

Figure 2. Types of equipment used by the player

Both equipment and power-ups increase the player's abilities and resources depending on how much experience he has gained. As the game progresses, the character must improve to deal with the enemies that become stronger. Abilities are stats that player characters use to improve their skills and traits. Each player character has a total of 6 abilities. These capabilities are shown in Figure 3.

Figure 3. Types of equipment used by the player

These abilities are as follows;

- Level: Level is the basic ability of the player character. This ability increases all other abilities.
- ATK (Attack): Offensive ability increases the damage the player character deals.
- AGI (Agility): The Agility ability increases the player character's running and attack speed.
- CON (Stamina): The Stamina ability increases the player character's Health.
- INT (Intelligence): The Intelligence ability increases the player character's energy pool.
- WIS (Wisdom): The Wisdom ability increases the player character's energy regeneration rate.
- VIT (Health): The Health ability increases the player character's life regeneration rate.

Resources are means that vary according to player abilities and allow player characters to use their abilities or survive. These resources are shown in Figure 4.

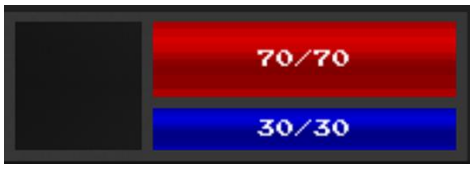

Figure 4. Player resources

Each player character has two resources, life, and energy. Life is the resource that keeps the character alive. It is associated with abilities health and stamina. The player loses the game when the health resource is completely depleted. Another resource is the experience resource, which fills up when the player defeats enemies, and when it is full, the character levels up.

The Energy Source is the source that allows the character to use his abilities. It is connected to the abilities of Intelligence and Wisdom. If the character does not have enough energy, he cannot use his abilities. Health and energy resources replenish over time based on health and wisdom abilities respectively. However, these resources can be increased by using items that replenish these resources during play. Items are objects that have a positive effect on a character's skills, strength, or resources. Items can be obtained by defeating enemies, opening treasure chests in chapters, or completing predetermined quests. Items can be easily divided into equipment and usable items.

Weapons and armor such as swords, bows, daggers that the character can carry and use can be given as examples of equipment. These sword-level types are shown in Figure 5.

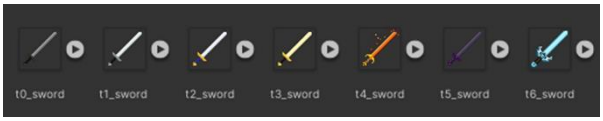

Figure 5. Sword levels

Usable items can be divided into two groups: consumable and non-consumable. An example of a consumable item is a potion that replenishes life and a talisman that increases the character's walking speed when used at will. Equipment samples are shown in Figure 6.

Figure 6. Equipment samples

Flow chart of the Designed System

In the computer game we designed, the enemy character makes a prediction by immediately comparing his skills, resources, and equipment to the player character's skills, resources, and equipment, and decides the action to take according to that prediction. Each enemy has a certain percentage of valor. If the player's percentage of defeat, which he subtracts from his prediction, is above the percentage of courage, he will take offensive actions against

the player, but if it is below that percentage, he will take defensive actions. These actions are actions such as life regeneration, fleeing and hiding from the player, calling for support. Two different algorithms have been developed for the partition creation algorithm. One of these algorithms is the natural partition generator and the other is the space style partition generator. The natural partitioning algorithm is shown in Figure 7.

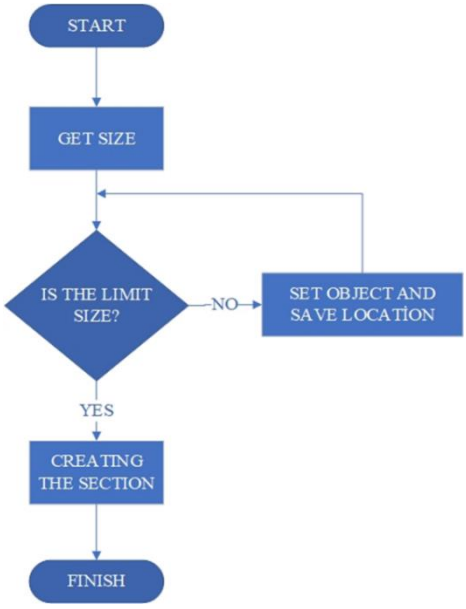

Figure 7. Natural partitioning creation algorithm

The natural partitioning algorithm is essentially as follows: A game object called "Roaming" is created at coordinates (0, 0) on the game scene. For each step, 1 to 4 dice (including 1 and 4) are rolled, where 1 represents up, 2 represents right, 3 represents down, and 4 represents left. The rover object is moved 1 unit in the appropriate direction according to the result of the dice, and its coordinates are recorded in the list. The rover continues this process until it reaches the section boundary. When the section boundary is reached, the "floor" object is added to all the coordinates contained in the list, and the playing field is created by surrounding these objects with "wall" objects. The "Door" object is added to the coordinate at the end of the list. The "Door" object is the game object that allows the transition to the next level. The space partitioning algorithm is shown in Figure 8.

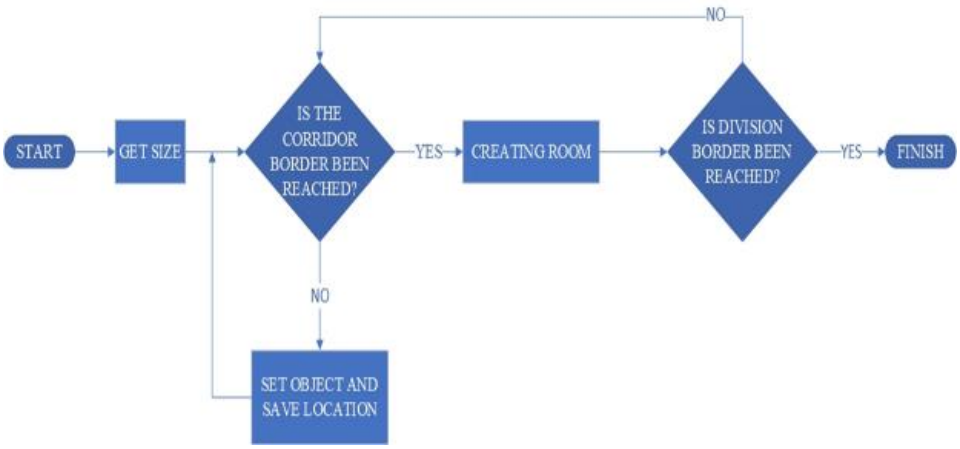

Figure 8. The space partitioning algorithm

The space partitioning algorithm shown works very similarly to the natural partitioning algorithm, but in addition to the partition boundary dimensions, this algorithm also uses corridor length and space dimension data. Just as in the natural partitioning algorithm, the rover object starts at (0, 0) and chooses a random direction according to the cube rolled. Instead of starting the algorithm from the beginning and moving only one step in a selected direction, in this type of algorithm the rover object moves along the randomly determined corridor length data in the selected direction. When the mobile object reaches the end of the corridor length data, it also creates a space according to the randomly determined space dimension data. Examples of random partitions in Figure 9.

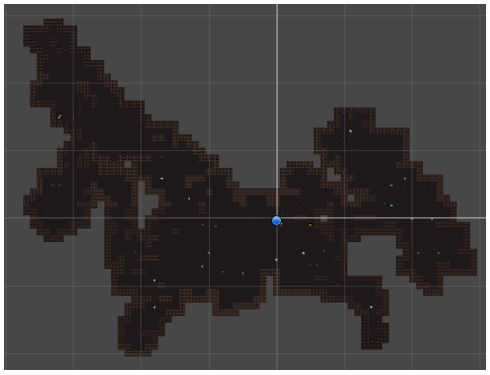

Figure 9. Examples of random partitions- 1.

Eight experiments were carried out in the application included in the study. In these experiments, the use of different equipment defined to the characters with AI was tested. The majority of the graphical and sound elements used in the said experiments are the items on the Unity official Internet WEB site. Some graphical elements were also specially designed by the authors within the scope of this study. First of all, the programs to be used in this study will be installed. The Unity Hub application, which will install the game engine for the Unity game engine, has been downloaded from its own site. The installation wizard for Unity Hub is running from the downloaded file. Different scene windows were used in the development of the game. These scenes are graphical scenes where the instant scene of our game is displayed. The game scene can be navigated using these window scenes. In addition, these scenes can be two- or three-dimensional. The game window is the window that shows how the changes made in the actual game will be presented to the end user. No changes can be made in this window. The function of this window shows the game maker how the game he made will appear to the end user. The file window is the window where the file directory where the sound, image, ready-made objects (prefab), code snippets (script), materials, physics and other file elements to be used in the game is displayed. Thanks to this window, the game maker can easily add new files to the project, use existing files, and make changes to them, instead of manually dealing with the operating system's file directory. Animation window is the window where animations of a game object are kept. Using this window, animations can be created by adding a series of photos to game objects. In our study, the coding of the animations that the player can make using AI is carried out with this window. For example, in the bending animation for the main character, when the character bends, the Collider element of the character object is also changed with the animation, thus reducing the collision area of the character. In the application, there are 4 different heroes that the user can choose. These heroes are "Warrior", "Bandit", "Hunter" and "Wizard". Each hero has unique abilities, equipment and animations. Player characters can move in the 2D game scene with WASD or arrow keys, attack with the left mouse button and use their special abilities with the right mouse button. These key sequences are designed in C# using the Unity game engine's Input system. The user can also change these

controls according to his own wishes. Both player characters have 10 unique animations. These animations are controlled by scripts written in C# and reflected on the game scene using the Animator feature of the Unity game engine. Each player character mirrors their own animation controls based on a basic player Animator.

The warrior character is a hero who wears an iron armor and attacks his enemies directly at close range using a sword and shield. He places emphasis on strength and defensive abilities. The sword he wields is a weapon that deals heavy damage at close range. His special ability is to use a shield, the player can use this shield to reduce the damage he takes. Thanks to the heavy armor he wears, he has protection against enemy attacks. Thanks to this equipment he carries, he has become strong and durable, but he cannot move very fast because of this weight. The bandit character wears leather armor and uses his daggers to sneak up on his enemies and defeat them. Emphasizes speed and sneaking abilities. The daggers he wields are a weapon that deals light damage at medium range. His special ability is sneak, the player can use his special ability to switch to sneak mode, so he can pass through the enemies undetected and make a surprise attack on his enemies. Using his rogue sneak, he moves quickly through his enemies and attacks them when they least expect it. The hunter character wears a leather armor and uses his bow to intercept his enemies from a distance. She values speed and survivability. The bow he uses is a weapon that deals moderate damage at long range. His special ability is to inactivate enemies, the player uses his special ability to shoot a pacifying arrow at his enemies, thereby defeating them one by one using the smash, divide, destroy technique. The hunter uses his superior survivability skills to seize enemies at their weakest moments. The mage character uses his wand and spellbook to interfere with his enemies from a distance. It gives importance to intelligence and skill abilities. The staff he wields is a weapon that deals heavy damage at long range. His special ability is to use magic, the player uses his special ability to defeat his enemies by using damaging spells such as fireball, electricity, or by using abilities such as teleportation, magic shield for his own benefit. They are physically weak as they don't wear armor made of iron or leather, but a Mage fights enemies with their wit and magical abilities.

The following abilities are used to improve the abilities and attributes of player characters. It has been shown in the study that these capabilities can be made by AI. Each player character has a total of 6 abilities. These capabilities are as follows:

- Level: Level is the basic ability of the player character. This ability increases all other abilities.
- ATK (Attack): The offensive ability increases the damage dealt by the player character.
- AGI (Agility): Agility ability increases the player character's walking and attacking speed.
- CON (Stamina): Stamina ability increases the player character's health tank.
- INT (Intelligence): Intelligence ability increases the player character's energy reservoir.
- WIS (Wisdom): The Wisdom ability increases the energy regeneration rate of the player character.
- VIT (Health): Health ability increases the player character's health regen rate.

Each player character also has a secondary ability. These abilities are not shown to the player as a score and can be changed by the equipment used by the player character. An example of one of these abilities is the Armor ability. The armor ability increases the player character's resistance to enemy damage. The numerical values of the abilities differ for each ability. For example, for each Intelligence point, the energy tank increases by 5 points, but

for each Agility point, walking speed increases by 0.04. Each player character has two resources, Health and Energy. Life is the resource that keeps the character alive. It is related to Health and Stamina abilities. The player loses the game when the health resource is completely exhausted. Another resource is the Experience resource, which fills up as the player defeats enemies, and when it's full, the character levels up. The energy source is the source that allows the character to use their abilities. It is related to Intelligence and Wisdom abilities. If the character does not have enough energy, he cannot use his abilities. Health and Energy resources refill over time based on Health and Wisdom abilities, respectively. However, these resources can be increased by using items that fill these resources during the game. Each player character can use 4 pieces of equipment. This equipment is respectively; weapon, ability item, armor and ring. The inventory panel is the player character's bag. All the equipment he carries on him is kept in this panel. By using this panel, the player can use various equipment to his character and can look at the features of these equipment. Each player character has 12 inventory slots. This chamber can be increased to 24, 36 and 48 pieces by using the bag during the game.

The game we developed in this study consists of many interconnected parts. The first part of the game is loaded when the user selects the game character and starts the game. The user skips the level by defeating the enemies in the first chapter and improving himself. As the chapters progress, the enemies in the chapter also get stronger. There are an unlimited number of levels in the game. The main purpose of the game is to advance as much as possible without being defeated by the enemies. The sections in the game and the contents of these sections are created by the computer using random section application algorithms. In this way, the game offers a different experience to the player every time it is played. After the game sections are created by the computer, treasure chests and enemies are placed in appropriate locations in the section. The contents of these treasure chests and the enemies are also selected semi-randomly by the computer algorithm.

Results and Discussion

In this part of the study, the experiments carried out and the results of these experiments are included. In the study, 8 different experiments were carried out in which the success time of the player in the game was measured with different equipment defined for the character, and which are listed in Table 1.

Table 1. Character Comparison: Levels and Equipment

Nu.	Operation Type	Results
1.	Attributes acquired through the use of starting items at the lowest level of the warrior character.	Average damage: 50,625, Walking speed: 3.325, Attack speed: 0.495, Health: 90, Energy: 25.
2.	Attributes acquired through the use of starting items at the lowest level of the bandit character.	Average damage: 31.5, Walking speed: 3.71, Attack speed: 0.47, Health: 70, Energy: 30.
3.	Comparison of attributes gained by player characters using their lowest level starting items.	Enemy defeat times in the same conditions; Warrior: 0.99 seconds, Bandit: 1.88 seconds.
4.	Attributes gained by leveling up the Warrior character and evolving items.	Average damage: 68.75, Walking speed: 3.57, Attack speed: 0.49, Health: 120, Energy: 35.
5.	Traits gained by the bandit character through level advancement and item progression.	Average damage: 44, Walking speed: 3.885, Attack speed: 0.445, Health: 90, Energy: 40.
6.	Traits gained through the use of mastery items at the highest level of the warrior character.	Average damage: 288.75, Walking speed: 4.2, Attack speed: 0.4, Health: 680, Energy: 120.
7.	Attributes gained by the rogue character through the use of mastery items at their highest level.	Average damage: 180, Walking speed: 6.055, Attack speed: 0.135, Health: 450, Energy: 220.
8.	Comparison of attributes gained by player characters through the use of mastery items at their highest level.	Enemy defeat times in the same conditions; Warrior: 1.2 seconds, Bandit: 0.54 seconds.

The table compares the attributes and abilities that different characters possess at different levels and with different equipment in a game. The data in the table demonstrates that players have different characteristics for different characters, and these characteristics affect their performance in the game. Additionally, factors such as character levels and equipment also influence the characters' attributes and abilities. The table provides insight into how the game's balancing and character development systems work and provides clues for players on how to improve their characters. The "Results" section of the table shows the results of a study conducted to measure the effects of different characters and equipment levels on various game attributes. These results can be helpful for players in determining their game strategies and character development. For example, players can see from the data in the table that the warrior character is better for quickly defeating enemies, but the thief character runs faster and has higher energy. Additionally, it is observed that characters become stronger as they level up and obtain better equipment. This data can help players develop their characters more effectively and be more successful in the game.

The points related to the experiments carried out in the study are presented below:

- In Experiment 1, the warrior character is equipped with a level 0 sword and shield, and the increase in his skill is observed and noted. If the warrior character is unequipped with an attack speed of 0.5 seconds, a walking speed of 3, 70 health, and 25 energies, these values increase, and his attack speed increases to 0.495 seconds, walking speed to 3.325, and health to 90. The energy value does not change.

- In Experiment 2, the rogue is equipped with a level 0 dagger and a cloak, and the increase in his abilities is observed and noted. When the rogue is equipped with no equipment, an attack speed of 0.5 seconds, a walking speed of 3, health of 70, and energy of 25 with starting items, these values increase, and his attack speed increases to 0.47 seconds, walking speed to 3.71, and energy to 30. The life source does not change.

- In Experiment 3, Warrior and Bandit characters equipped with starting items and character level 1 make an explosive attack on an enemy with 100 health under the same conditions. The experiment is performed against the same opponent, ignoring the opponent's damage reduction abilities. The Warrior character defeated this enemy in 0.99 seconds with an average damage of 50.625 and an attack speed of 0.495 seconds, while the Bandit character defeated this enemy in 1.88 seconds with average damage of 31.5 and an attack speed of 0.47 seconds.

- In Experiment 4, the Warrior character is equipped with a level 1 sword, a level 0 shield, armor, and a power ring, and an increase in his abilities is observed. With these pieces of equipment, the character's attack speed increases to every 0.49 seconds, his walking speed increases to 3.57, and his health and energy levels increase to 120 and 35, respectively.

- In Experiment 5, the rogue is equipped with a level 1 dagger, a level 0 cloak, armor, and a speed ring, and an increase in his abilities is observed. With these pieces of equipment, the character's attack speed increases to every 0.445 seconds, his walking speed increases to 3.885, and his health and energy values increase to 90 and 40, respectively.

- In Experiment 6, the warrior character is equipped with a level 6 sword and power ring and a level 0 shield and armor, and an increase in his abilities is observed. With this equipment, the character's attack speed increases to every 0.4 seconds, his walking speed increases to 4.2, and his health and energy values increase to 680 and 120, respectively.

- In Experiment 7, the rogue is equipped with a level 6 dagger and a level 0 speed ring, cloak, and armor, and an increase in his abilities is observed. When equipped with these pieces of equipment, the character's attack speed increases to every 0.135 seconds, his running speed increases to 6.055, and his health and energy levels increase to 450 and 220, respectively.

- In Experiment 8, both Warrior and Bandit characters equipped with ultimate items and character level 20 performed an attacking burst against an opponent with 600 HP under the same conditions. The experiment is performed against the same opponent, ignoring the opponent's damage reduction abilities. The Warrior character defeated this enemy in 1.2 seconds with average damage of 288.75 and an attack speed of 0.4 seconds, while the Rogue character defeated this enemy in 0.54 seconds with an average damage of 180 and an attack speed of 0.135 seconds.

Offensive action is decided that if our game character's villain is more than the game's enemy character, the enemy is fleeing. But in the opposite case, the enemy is attacking our game character. For example, a warrior character who attacks an enemy with a 100 life can strike 1 time in 0.495 seconds. Assuming that this character has averaged damage to each stroke, it can defeat the enemy in 2 strokes, 0.99 seconds. A Bandit character who attacks the same enemy can strike 1 time in 0.470 seconds. Assuming that this character has averaged damage to each stroke, it can defeat the enemy in 4 strokes, 1.88 seconds.

The result of the experiments (gameplay tests) is that the Warrior character, one of the player characters available in the game, is superior to the Bandit character at the beginning of the game, but since the Bandit character can evolve more, it is superior to the Warrior character at the upper levels of the game. Users who prefer a faster playstyle can choose the Bandit character, and users who prefer a heavier but powerful playstyle can choose the Warrior character. Thanks to the options available to the user and the variety of items in the game, character and user-centric production have been realized. In studying roguelike games similar to the game on the market, it is found that the most important features users who play these games look for in these games are character development and personalization, story depth, and playability. Considering these findings, the project focused on character development and customization, endless game playability, and variety. Thanks to the game map randomly generated by computer algorithms in the developed game application and the variety of items the user encounters in the game, the game provides the user with a unique experience in each session. The player character gains experience by defeating the enemies he encounters throughout the game and improves by using the items he finds. Thanks to the variety of these items and the fact that he is confronted with different items each time, the player's progression is not linear but depends on the player's choices and the luck factor that the game offers the player. The mentioned features extend the life of the game application and prevent memorization. During the game, each enemy that the player encounters has its own special AI. For example, one of the enemies in the game compares its strength with that of the player and predicts whether it can defeat the player. If the probability of defeating the player is above a certain percentage, it attacks the player, and if it is low, it runs away from the player. Apart from this, there are also actions that are left to the AI decision algorithm such as tracking the player, calling for support, and following the player. The phases of software development are followed and a game application is developed according to these phases. In the context of our research, the following points stand out based on the results obtained:

- The Warrior character is better than the Bandit character at the beginning of the game, but the Bandit character becomes superior at higher levels due to its ability to evolve more, according to the results of gameplay tests.
- The game offers users various character options and a variety of items to encourage character and user-centered production.
- The project focused on character development and customization, endless game playability, and variety.
- The game features a randomly generated map and a variety of items to provide a unique experience in each session.
- The player character gains experience by defeating enemies and using items found in the game.
- Each enemy in the game has its own unique AI and decision-making algorithm.
- The game application was developed by following software development phases.

It is evaluated that the articles we researched within the scope of our study generally focused on game development using a game engine. However, in the current information age, we believe that working on AI-based game characters and evaluating how realistic their artificial intelligence decisions are when added to games will have great contributions to the games according to the changing demands of computer game players. It is seen that the studies researched generally do not provide research on techniques used to increase the learning of the AI-based game world and the development of game characters. However, our study provides information on the game engine we used, the artificial intelligence techniques used, and the behavior of game characters. Additionally, our study used different scenarios to test the performance of AI-based game characters. Some of the various articles in the literature, like our study, focus on developing artificial intelligence-based games using Unity 2D Game Engine. However, the character-oriented structure of the game presented in our study is different from other articles and provides a different perspective.

Conclusion

In this study, we designed an AI-based two-dimensional video game in Unity 2D Game Engine, which can decide what the game characters will do within their abilities. We designed the game with C# on the Unity2D game engine. In the game application we developed, we used AI abilities so that the player characters can survive as long as possible, use the most advanced equipment, successfully complete various stages of the game and develop more advanced game tactics. In our game application, the two-dimensional player character is controlled by the user. In practice, the player moves on a two-dimensional game plane created using procedural plane generation algorithms. Various statistics such as the number of enemies defeated and the level reached in the game are also recorded. We conducted 8 different experiments in the study. In our experiments, we measured the success time in the game with different equipment defined for the player character. We measured this time at rates ranging from 0.54 seconds to 1.88 seconds. Today, AI algorithms are developed to play computer games better or to design better games, and new games are designed to improve artificial intelligence algorithms. In this context, it is considered that this study will contribute to the literature on the use of AI in computer games.

As future work, we will further improve the playability of the game we designed in this study with AI by adding new user performance, stats, and other features (more chapters and different themes, adding more usable items, decorations, and enemy types, etc.) based on AI.

References

Bishop, L., Eberly, D., Whitted, T., Finch, M., & Shantz, M. (1998). Designing a PC game engine. *IEEE Computer Graphics and Applications*, *18*(1), 46-53.

Bostan, B., Tinli, B., & Çatak, G. (2020). Worldbuilding components and transmedial extensions of computer role-playing games. *Kültür ve İletişim, 23*(45), 273-295.

Calp, M. H., & Utku, K. Ö. S. E. (2018). Planning activities in software testing process: A literature review and suggestions for future research. *Gazi University Journal of Science*, *31*(3), 801-819.

Chivukula, A. S., & Liu, W. (2018). Adversarial deep learning models with multiple adversaries. *IEEE Transactions on Knowledge and Data Engineering*, *31*(6), 1066-1079.

Clarke, R. I., Lee, J. H., & Clark, N. (2017). Why video game genres fail: A classificatory analysis. *Games and Culture*, *12*(5), 445-465.

Chen, J. X. (2016). The evolution of computing: AlphaGo. *Computing in Science & Engineering*, *18*(4), 4-7.

Chen, P. P., Guitart, A., del Río, A. F., & Periánez, A. (2018, December). Customer lifetime value in video games using deep learning and parametric models. In *2018 IEEE international conference on big data (big data)* (pp. 2134-2140). IEEE.

Costa, L. M., Souza, A. C. C., & Souza, F. C. M. (2019, October). An approach for team composition in league of legends using genetic algorithm. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* (pp. 52-61). IEEE.

Çetin, M., & Sarıca, Y. Artificial Intelligence Based Game Levelling. *Balkan Journal of Electrical and Computer Engineering*, *8*(2), 147-153.

Denning, A. (2021). Deep Play? Video Games and the Historical Imaginary. *The American Historical Review*, *126*(1), 180-198.

Dehghanı, M., Montazerı, Z., & Malik, O. P. (2019). DGO: Dice game optimizer. *Gazi University Journal of Science*, *32*(3), 871-882.

Frasca, G. (2001). Rethinking agency and immersion: video games as a means of consciousness-raising. *Digital Creativity*, *12*(3), 167-174.

Evans, M. C., Kapuscinska, A., Greenholt, M., Lin, J., Liu, X., Zhang, T., ... & Kaufman, G. (2021, May). Designing a Self-Efficacy Game for Health Literacy in Marginalized Communities. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-6).

Haaranen, A., Rissanen, T., Laatikainen, T., & Kauhanen, J. (2014). Digital and video games in health promotion: Systematic review of games and health behavior. *Finnish Journal of eHealth and eWelfare*, *6*(4), 153-163.

Haas, J. K. (2014). A history of the unity game engine. *Diss. Worcester Polytechnic Institute*, *483*(2014), 484.

Hanes, L., & Stone, R. (2019). A model of heritage content to support the design and analysis of video games for history education. *Journal of Computers in Education*, *6*(4), 587-612.

Inal, Y., & Güner, H. (2015). Ensuring success in a large scale software project: an examination of the learning styles and characteristics of the potential end users. *Gazi University Journal of Science, 28*(4), 535-540.

Justesen, N., Bontrager, P., Togelius, J., & Risi, S. (2019). Deep learning for video game playing. *IEEE Transactions on Games, 12*(1), 1-20.

Kepenek, E. B. (2018). Entrepneural Mindset in Video Gaming Sector: Evidence From Turkey. *Ankara Üniversitesi SBF Dergisi, 73*(2), 643-666.

Kim, J. Y., & Kang, S. H. (2021). Windows of Opportunity, Capability and Catch-Up: The Chinese Game Industry. *Journal of Contemporary Asia, 51*(1), 132-156.

Lohokare, A., Shah, A., & Zyda, M. (2020, October). Deep learning bot for league of legends. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Vol. 16, No. 1, pp. 322-324).

Lee, J. H., Karlova, N., Clarke, R. I., Thornton, K., & Perti, A. (2014). Facet analysis of video game genres. *IConference 2014 Proceedings.*

Mochocki, M. (2021). Heritage sites and video games: Questions of authenticity and immersion. *Games and Culture, 16*(8), 951-977.

Moreno, D. N. T. (2020). *Realidade Virtual para Treino de Situações de Catástrofe* (Doctoral dissertation).

O'hagan, M., & Mangiron, C. (2013). Game localization. *Amsterdam and Philadelphia: John Benjamins Publishing Company.*

Pirovano, M. (2012). The use of fuzzy logic for artificial intelligence in games. *University of Milano, Milano*.

Parlan, B. (2017). Independent Game Development: Prıce of Freedom. *International Journal of Social Sciences and Interdisciplinary Studies, 2*(2), 57-69.

Pfau, J., Liapis, A., Volkmar, G., Yannakakis, G. N., & Malaka, R. (2020, August). Dungeons & replicants: automated game balancing via deep player behavior modeling. In *2020 IEEE Conference on Games (CoG)* (pp. 431-438). IEEE.

Prato, G. D., Feijoo Gonzalez, C. A., & Simon, J. P. (2014). Innovations in the video game industry: Changing global markets. *Communications & strategies*, (94), 17-38.

Safadi, F., Fonteneau, R., & Ernst, D. (2015). Artificial intelligence in video games: Towards a unified framework. *International Journal of Computer Games Technology, 2015*.

Sanders, A. (2016). An introduction to Unreal engine 4. AK Peters/CRC Press.

Shannon, C. E. (1950). XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 41*(314), 256-275.

Stewart, J., & Misuraca, G. (2013). The industry and policy context for digital games for empowerment and inclusion. *Joint Research Centre of the European Commission (JRC).*

Tunç, U. Z. L. U., & ŞAYKOL, E. (2020). Evaluating a player's network class in a multiplayer game with fuzzy logic. *Gümüşhane üniversitesi fen bilimleri enstitüsü dergisi, 10*(1), 163-173.

Tylor, E. B. (1879). THE HISTORY OF GAMES. *Fortnightly*, *25*(149), 735-747.

Westera, W., Prada, R., Mascarenhas, S., Santos, P. A., Dias, J., Guimarães, M., ... & Ruseti, S. (2020). Artificial intelligence moving serious gaming: Presenting reusable game AI components. *Education and Information Technologies*, *25*(1), 351-380.

Westecott, E. (2013). Independent game development as craft. *Loading… The Journal of the Canadian Game Studies Association*, *7*(11), 78-91.

Zackariasson, P., & Wilson, T. L. (Eds.). (2012). The video game industry: *Formation, present state, and future.* Routledge.