Mugla Journal of Science and Technology

FIRST-PERSON USER GAME SOFTWARE DESIGNING BASED ON RULE-BASED ARTIFICIAL INTELLIGENCE TECHNIQUE

İlhan TARIMER*
Faculty of Technology, Muğla Sıtkı Koçman University, Muğla, itarimer@mu.edu.tr
https://orcid.org/0000-0002-7274-5680
Ömercan ŞEN
Faculty of Technology, Muğla Sıtkı Koçman University, Muğla, omersansen16@gmail.com
https://orcid.org/0000-0003-2382-0653

Received: 06.06.2018, Accepted: 16.10.2018 Research Article
*Corresponding author DOI: 10.22531/muglajsci.449029

Abstract

In this study, a game designing based on rule-based artificial intelligence on a 3-D game engine has been developed. In the existing computer games in which artificial intelligence is not used, it is seen that a fixed story is used. Since as a novelty rule-based artificial intelligence is used in this game, the story of the game has been dynamically defined based on artificial intelligence, different story combinations have been created within the developed game. The design of the game has been edited according to the possible actions of the user. A main rule has been set in the content, this rule has been applied by artificial intelligence, the animations and the existing rule-based artificial intelligence technique have been run in the process. The important difference of this game in which the rule-based artificial intelligence technique is used is revealing of story dynamically, and the progressing of game that depends on the first-person user (FPU).

Keywords: Rule-based artificial intelligence, first-person user (FPU), game.

KURAL TABANLI YAPAY ZEKÂ TEKNİĞİNE GÖRE BİRİNCİ ŞAHIS KULLANICILI BİR OYUN YAZILIMI TASARIMI

Öz

Bu makalede, 3-B oyun motoru üzerinde, birinci şahıs kullanıcılı olarak kural tabanlı yapay zekâ tekniğine göre bir oyun tasarımı geliştirilmiştir. Yapay zekânın kullanılmadığı mevcut bilgisayar oyunlarında sabit bir öykünün kullanıldığı görülmektedir. Bu oyunda bir yenilik kural tabanlı yapay zekâ kullanıldığı için, geliştirilen bu oyunda yapay zekâya bağlı olarak oyunun hikâyesi dinamik bir şekilde tanımlanmış ve oyun içerisinde farklı hikâye kombinasyonları oluşturulmuştur. Bu oyunun tasarımı, kullanıcının yapması olası hamleler üzerine kurgulanmıştır. İçerikte ana bir kural belirlenmiş, bu kural yapay zekâ ile uygulanmış, animasyonlar ve mevcut kurala dayalı yapay zekâ tekniği süreç içerisinde yürütülmüştür. Kural tabanlı yapay zekâ tekniğinin kullanıldığı bu oyunun önemli farkı, hikâyelerin dinamik olarak ortaya çıkması ve oyunun gidişatının birinci şahıs kullanıcıya bağlı olarak belli olmasıdır.

Anahtar Kelimeler: Kural tabanlı yapay zekâ, birinci şahıs kullanıcı, oyun

Cite
Tarımer, İ., Şen, Ö., (2018). "First-person user game software designing based on rule-based artificial intelligence technique", Mugla Journal of Science and Technology, 4(2), 198-204.

1. Introduction

From the past to the now, many maps and a lot of stories have been designed. The internal structure of the games along with the development of the game industry has changed. From the past in which the rule-based artificial intelligence technique begun to be used in the game, to the nowadays, stories are mostly penetrated to the game dynamically. Therefore, it has not need to create much stories. The game has a script which was written in its own and dynamic narratives that can be created by the present artificial intelligence within the game.

In the literature researches, we have seen several publications regarding to 3D game designing. The reference [1] has presented a 3D game designing and explored the challenges. The reference [2] has discussed the scene optimizations for mobile phones by using Unity 3D. In [3], the authors focused on the use of 3D game engines with the emerging experience based design approach. In [4], the authors made a contribution to smart game market.

The rule-based artificial intelligence algorithm technique is an algorithm that allows player characters to be moved according to a possible game style managed by the computer. The movements of characters managed by computer cannot get out of the existing rules. The solely option that can be made against player's attacks is to apply existing rules and to make artificial intelligence to generate story dynamically within the play [5, 21]. Artificial intelligence programming is a process that it calculates possible movements, which can be made by user, then a new programming realizes after occurring new circumstances [1]. Rule-based artificial intelligence algorithm technique is used in survival games, in first-person games, and in strategy and simulation games. This technic has been used

since it operates over probabilities and it is open to be developed in the study.
The animations, character controls, and menu designing are seen in [6]. In [7] the importance of artificial intelligence is declared. The map designing, adding new characters to game, and how to create animations specified in [8]. In [9], the last version of map, final animations and the last clone soldiers are told.
In this study, it has been intended that the artificial intelligence creates different game stories by using rule-based artificial intelligence at the scenes and by taking the user as a basis. The soldiers created by artificial intelligence monitors user continually. The content of the developed game has had partially mobile game features, and had partially computer features; hence, it can be said that it is designed together with different specifications from both two platforms.
In the developed game, there are 2 map designing's. Soldier characters are added to these maps according to rule-based artificial intelligence; the animations of walking, running, changing bullets and shooting are added to these soldier characters. The system of gathering scores obtained from executing the game is designed; a menu is created from the scores gathered by the user and all the scores are accumulated within the menu. The map designing is done within the game engine by purchasing the necessary objects from Asset Store [10]. The soldier objects, score enhancing and score reducing objects are placed to the map.
The developed game has three hierarchical structures as between character to character, between gun to camera, and between character to camera [11]. The soldier designings and animations in the game are made on the blender program; the coding operations are made on C# [12]. The platforms of Unity 2D and 3D are sophisticated and multiple purpose game engines that computer game can be designed; they support to write codes on both C# and Java script with their drag and drop functionality [5]. Animation designing of objects within the game has been made by its game engine, and the physical properties have been also added. At the game in which solid object properties are used, the player is prevented from passing through these objects [13]. It was emphasized over map designing's in the game. In [14], existing simulations and physical features of animations were showed. Smed and Hakonen recognized components, relationships, and aspects common to all games [15]. Bettner and Terrano explained the design architecture, implementation, and some of the lessons learned creating the multi-player code for the Age of Empires games in their paper [16].
The content flow for this paper is as following: The design tools and methods have been explained, and the game engine in which we created our designing has been introduced in the second section. The main algorithms of this study have been given in the third section. The fourth section explains the obtained findings and the evaluations. In the last section, the results, and suggestions for future have been put forth.
In the design, game of the Age of Empires [17] has been taken as a sample. In this game, a field pool was created and players were offered the possibility to move in different combinations in accordance with the rules set. The rest of other soldiers can be actuated by giving a command to a single soldier. If the rule-based artificial intelligence technique was not used, all the soldiers would have to command separately.
Since this work is designed on a dynamic story, when the game starts, the user draws the course of the stage and navigates in three different routes on the map. When the user starts to the game, the threshold value (distance) is activated to look at the distance between artificial intelligence and itself. If the threshold value captures the specified distance for user and artificial intelligence, the rules that are programmed for artificial intelligence are activated; then, they run towards the user, fire to the user, operate the animations.

2. The Tools and the Methods

Artificial intelligence software(s) are the intelligent systems, which were created by the software designers (programmers) that are used in non-organic systems. The fundamental topics of artificial intelligence are knowledge representation, inference and learning. Artificial intelligence is regarded to intelligent computer systems, and it can be defined as an intelligent system that is also based to the calculations. Artificial intelligence completely and accurately manages the designed system without any human intervention.
Artificial intelligence, which is almost available in all games programming, generally uses rule-based algorithm technics. Lopes and Bidarra have put soldiers and soldier animations to the maps. They made the artificial intelligence to produce movements against the possible user movements [18].
Systems using artificial intelligence have a certain learning cycle. At the end of the learning process, the elements of artificial intelligence evolve to respond to changing situations. The success of the developed artificial intelligence system is measured according to the successful responses to variable and dynamic problems.
The algorithmic map of the game is created firstly, while a game is designing by rule-based artificial intelligence. The first rule in this map is to make a distance-based work between user and artificial intelligence. The animations based to the distance such as running, walking, and shooting are executed. In order to execute these animations, the distance is taken as a basis in the part of artificial intelligence coding. The soldiers (and their animations) that are working (mobilizing) depending on the distance between the user and the object(s) can follow the user until the game over in case of the user keeps the certain distance interval. It is possible to make changes to able to tell story in the game. Depending the user's possible movements, both game flow and story of the game are changed dynamically, since the game is designed as considering the user.

2.1. Unity 3D

Game engines creates backstage of video games. From paint and draw studies up to the mathematics which control every corner of the screens, this engine decides to everything [19]. In this context, Unity is defined as a game development platform for computers and mobile devices. In the Unity 3D game engine which converts the written codes to visuals, C# and Java script, are used as programming languages. Through Asset Store application market, open source applications can be downloaded and they can be run inside the game engine. The Asset Store gives a lot of material such as objects, object animations, maps, characters that are already ready to use for designing [10, 20].
Unity 3D is a game engine that runs with drag and drop logics. In this game engine, what the object to desire executing the written codes, should go over the object, and click on it by drag and drop function. Unity 3D that is known (being) at the forefront with the survival and simulation games, has recently come to the forefront with mobile games as well. With offering both open sources, and fast and easy map designing's, the designers can create games quick and to use quite easy within Unity 3D. Both hierarchical structure and one-to-one relationship of objects are quite important in this game engine. It is very important to collect existing a lot of objects under one object, to make them as whole. Objects are not independent of

each other [21]. In [22], Chen discussed improving interactive experiences in game design.

3. Rule-Based Artificial Intelligence Technique Map, Flow Diagram and The Program's Structure

The functions, parameters, vector definitions and object definitions were given in the programming section of rule-based artificial intelligence technique. The less number of functions have used in the programming, since avoiding long term compiling problem. In order to know how much bullets and lives are there for the first-person user during playing, the texts have been created to shoot at a target in a certain interval. These have been programmed to be visible by using texts.

Artificial Intelligence Algorithm

Input: Parameters, gameobject, getcomponent
Output: Sahne.exe(Windows, Mac, Linux)

```
1:   Void Start()
     kalanmermi = 120;
     zaman = maxzaman;
     hasar = Random.Range (5,15);
2:   Void Update()
     mermiyazi.text = ""+mermi+"/"+toplammermi;
     if (Input.GetMouseButton (0) && mermi > 0 && shoottime
     <= Time.time) {
                    shoottime = Time.time + firerate;
                    mermi--;
                    muzzleflash.emit = true;
3:   Void OnTriggerEnter(Collider nesne)
     If(nesne.gameobject.tag==”Azaltici”)
     Destroy(nesne.gameobject)
4:    Public ParticleEmitter muzzleFlash
     muzzleFlash.emit= true;
     muzzleFlash.emit=false;
5:    Public Gameobject klonasker
     GetComponent<Animation>().Play("Run");
6:    Vector3 pozisyon
7:    GetComponent<Animation>().Play
8:    Vector3.Distance
9:   Application.Loadlevel()
10:  Void OyunDevami
```

Void Start(): For beginning, character's lifes were given “100”; remaining–bullets, time and initial damage definitions were made.
Void Update(): Character's exchange of bullets, closed and open state of muzzle-flash, control of the bullet over the text was done in the update.
Void OnTriggerEnter(Collider nesne): Both the score and life regeneration were done within the game, when to touch to the reducing or increasing objects; it is also effective on soldiers.
Public ParticleEmitter muzzleflash: The explosion effect was given, when the soldier fired the gun.
Public Gameobject klonasker: In order to make soldier animations, a clone soldier was created; its codes were removed and the animations were added solely to it.
Vector3 pozisyon : To measure the distance between the soldier and the character, the *vector3* depending upon the character was defined.
GetComponent<Animation>().Play : The code line which needs to run animations was written.
Vector3.Distance : This was used to fulfill position of the existing soldier to the place of clone soldier in the game and to align position of the camera to the position of the character.
Application.Loadlevel() : It was used at transition amongst the scenes.
Void OyunDevamı() : A difficult map was created by using this function for the character, in the second stage, the character was able to fall down, to restart where the soldier left off, a starting point was created by using an object and it was made a function.
Below is the algorithm of the relationship between the soldier and the character in the rule-based artificial intelligence technique.

Algorithm Between Soldier and Character

Input: Parametreler, gameobject, getcomponent
Output: Sahne.exe(Windows, Mac, Linux)

```
1:   Void Start()
     Can=100;
     Can=maxCan;
2:   Void Update()
     Poz=new Vector3(karakter.position.x,transform.position.y,
     karakter.position.z);
3:   Void CandanDusme(){
     If(Physics.Raycast(Camera.main.transform.position,
     Camera.main.transform.forward,out,hit,30)){
     If(hit.transform.tag==”asker”){
     yz =hit.transform.gameObject.GetComponent
     <Yapayzeka>();
     yz.Can - = hasar;
     }}}
4:   Void CanBari(){
     CanBar.transform.localScale = new Vector3
     (Can/100,1,1);}
```

Void Start(): For beginning, character's lifes were given “100”. It was defined a variable named “MaxCan” and it was provided to have not more than 100 lives, by equalizing maxcan value to Can.
Void Update(): The current position of the character is constantly changing, so it was written under the update function. By defining a new *Vector3*, the x, y, z positions of the character are dropped into the new *Vector3*.
Void CandanDusme() : This function adjusts the camera's position regarding to the current position of character. If any contact is made with the soldier, the life value of the life variable on y-z decreases up to the certain amount.
Void CanBari() : The *canbar* on the Canvas was opened. The opened bars run related to the function of CanDusme. If the character is damaged, when the user is touched to a soldier, the life value falls under both the function and the canvas. The distance algorithm between the soldier and the character is given below.

Algorithm Due to Distance Between Soldier and Character

Input: Parametreler, gameobject, getcomponent
Output: Sahne.exe(Windows, Mac, Linux)

```
1:   Void Start()
     Can=100;
     Can=maxCan;
2:   Void Update()
     mesafe                                         =
     Vector3.Distance(transform.position,karakter.position);
3:   Void Mesafeler()
     If(mesafe<20 && mesafe>10)
     {
     yuruyus= true;
     ates=false;}
4:   Void Yuruyus()
     {If(yuruyus)
     {hiz=4;
     transform.position                                    =
     Vector3.MoveTowards(transform.position,karakter.positio
     n,
     hiz* Time.deltatime);}}
```

Void Start(): For beginning, character's lifes were given "100". It was defined a variable named "MaxCan" and it was provided to have not more than 100 lives, by equalizing maxcan value to Can.

Void Update(): It was defined a variable named "Mesafe". This variable (Distance: Mesafe) defines for bringing some animations such as walking, shooting to the states of active and passive by measuring the distance between soldier and character.

Void Mesafeler(): The walking and the shooting animations were run by using the distance variable which was defined in the Update() function with the If loops.

Void Yuruyus(): A certain speed was given to the soldier in the function. A straight line was drawn on the backplane between the character and the soldier by using MoveTowards function. The soldier runs towards the character on this line.

Below, the addition algorithm of the clone soldier was given.

Addition Algorithm of the Clone Soldier to the Game

Input: Parametreler, gameobject, getcomponent
Output: Sahne.exe(Windows, Mac, Linux)

```
1:   Void Start()
     Can=100;
     Can=maxCan;
     Public gameobject oluasker;
2:   Void Update()
     Poz= new
     Vector3(karakter.position.x,transform.position.y,
     karakter.position.z);
3    Void oluasker()
     {If(can<=0)
     {Instantiate(olenasker, transform.position,
     transform.rotation);
     Destroy(gameObject)}}
```

Void Start(): For beginning, character's lifes were given "100". It was defined a variable named "MaxCan" and it was provided to have not more than 100 lives, by equalizing maxcan value to Can. A public variable named "gameObject" was defined as OluAsker.

Void Update(): Since the current position of the character changes continually, it is written under the update function. By defining a new Vector3, the x, y, z positions of the character are dropped into the new Vector3.

Void oluasker(): The current soldiers within the game were copied, the Oluasker was tagged, and the necessary animations were added. The oluaskerler where placed at the closed position were put on the existing soldiers in the map; thus, it was provided that the oluasker appears, when the character disarms the army; it was dropped by the dying animations. It is provided that the dead soldier is replaced with the soldier by If condition.

4. Tests and Findings

Table 1 gives test results of the designed game's performance at the computers with different CPUs and operating systems.

Table 1. Performance tests of the designed game

Operating System	Used CPU	Optimization	Result
Windows XP	1.09 %	Yes	Successful
Windows Vista	1.14 %	Yes	Successful
Windows 7	1.43 %	Yes	Successful
Windows 8	2.12 %	No	Successful
Windows 8.1	2.49 %	No	Successful
Windows 10	3.06 %	Yes	Successful
Mac OS X 10.7 Lion	4.86 %	No	Successful
Mac OS X 10.8 Mountain Lion	4.35 %	No	Successful
Mac OS X 10.9 Mavericks	3.97 %	No	Successful
Mac OS X 10.10 Yosemite	3.64 %	No	Successful
Mac OS X 10.11 El Capitan	3.15 %	Yes	Successful
Ubuntu	5.86 %	No	Successful
Arch Linux	6.17 %	No	Successful

From Table 1, it is inferred that the game designed in the study can operate in all OSs. It can be said that this game uses less CPUs especially in Windows and Mac OS systems. All Windows versions apart 8 and 8.1 allows to make optimization for the game. Another Mac OSs, additionally Ubuntu and Arch Linux OS don't allow to optimization. The first map view of the game is given in Figure 1 as a bird's-eye view. This map has been obtained by writing the codes.

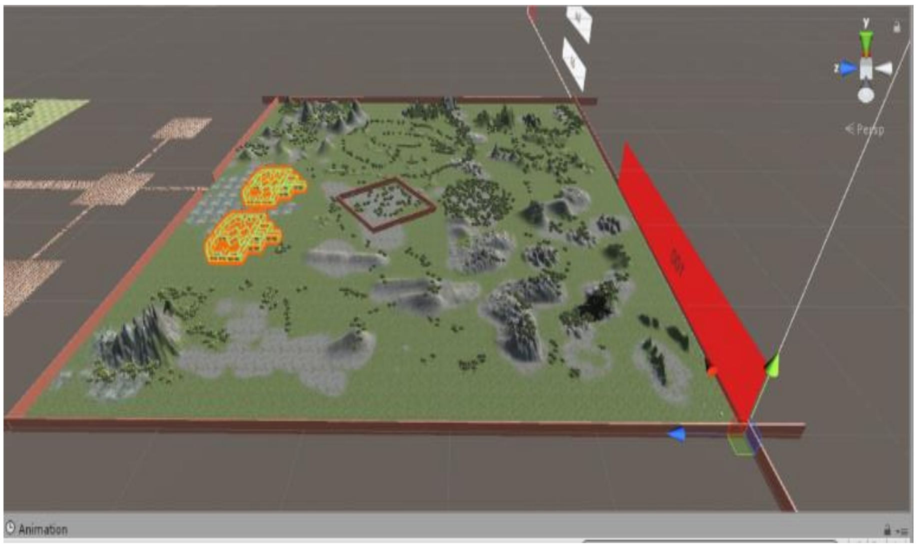

Figure 1. Overview of the map

The map seen in Figure 1 presents models of land, tree and mountain, grass texture, and labyrinth. The labyrinth was designed entirely from the cubes. A menu at one canvas within the stage has been created. In the menu, a life bar is added, and '100' value is donated to this life bar. As user takes damages, amount of this value decrease, in case of it becomes to 'zero', the game will start again. The land has not been used in its standard dimensions, but has been expanded. The design of this map was made within the game engine by downloading the necessary materials such as trees and mountain models from Asset Store. The obtained structure is used in the background design of the game. In Figure 2, the start menu is given.

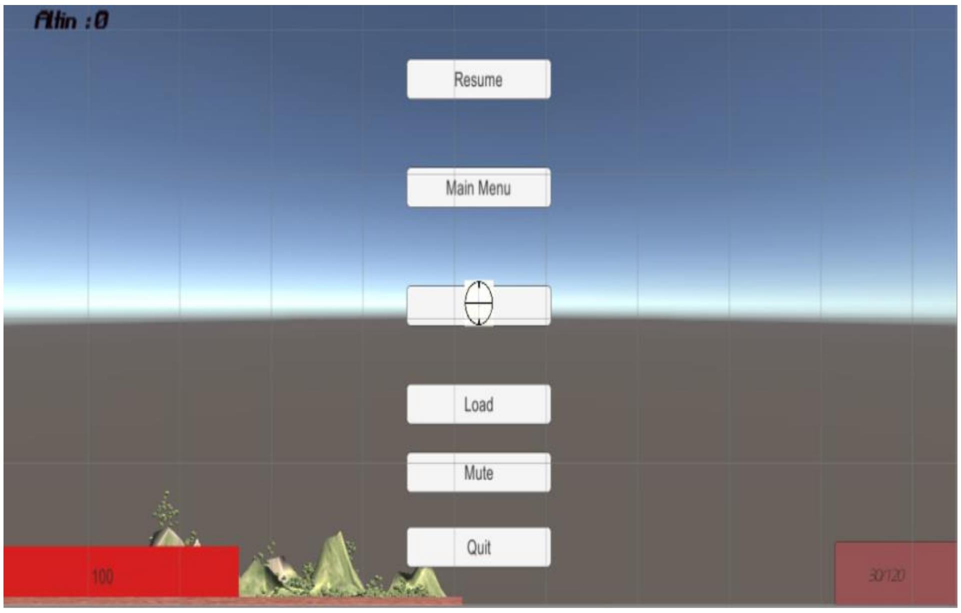

Figure 2. The overview of the designed start menu

The menu given in Figure 2 contains canvas, buttons and texts. The pause menu is recorded as a separate scene in the game, the functions are assigned to the buttons and the coding process is performed.
The value of life bar is '100'. There are 30 bullets, and totally 120 bullets in a cartridge clip. The user can monitor increasing and decreasing in life and bullet values on the game screen. The game's menu is designed under the canvas and panel where the game engine gives it. During playing the game, the pause menu is displayed on the screen when to click on letter "P".
Figure 3 shows the starting image of the scene within the game is given.

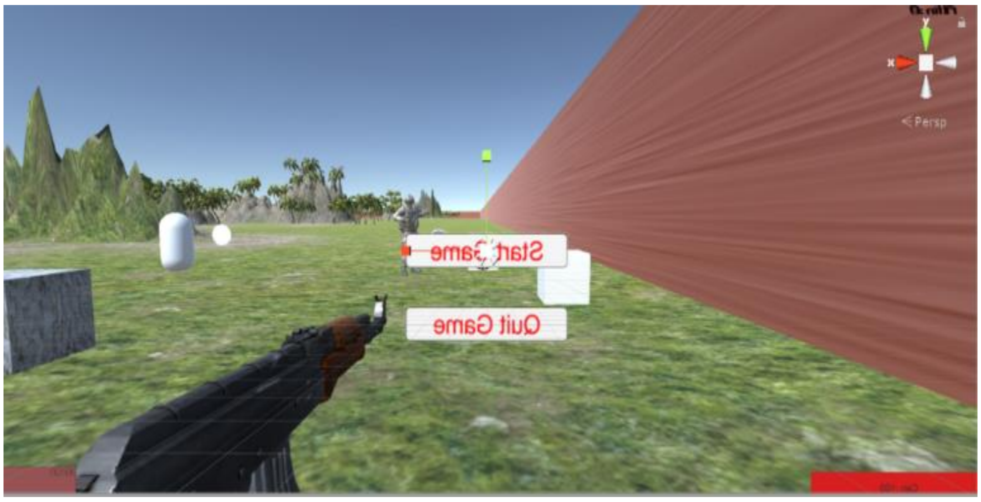

Figure 3. The screen view of the game from angle of user camera's and the first-person eye

Fig. 3 is the perspective view of the scene containing the first-person user for this game. There is one FPU Controller script, one weapon, one camera, audio files of several animations such as walking, jumping, running in the character. It has been provided that the user looked at to the game map with a comfortable and from a wide view in this design. The several scenario algorithms that may occur due to artificial intelligence in the game are given below:

The First Scenario: If the character tries to continue the game without disarming the soldiers, the enemy will die from the fire and the game will start again.

The Second Scenario: On the 2nd map, the character falls down. In this case it can throw the enemy into the sea near the sides of the road and get rid of it. Alternatively, it can fire and escape; Because of a few bullets and the many soldiers for the second map, it has to do one of two things.

Not only general features are seen; but also, mobile gaming features such as gold-picking gold, gold reducer, and game restarter are found in the game. Therefore, on the one hand, features of mobile games have been taken; on the other hand, the internal stories were revealed by taking the features of computer games such as wide map, interaction with the enemy. A design has emerged in which the game's story a first-person, rule-based artificial intelligence technique.

The player movements are calculated in the first-person user. In this game, artificial intelligence learns completely in accordance with the user. For this reason, the rule-based artificial intelligence algorithm technique is consistent, operates completely as based the user, evaluates all probabilities. In case of the story of game and the scenes grows, this technic needs to take supports of other artificial algorithm technics.

3. Results and Recommendations

In this study, by using rule-based artificial intelligence algorithm technique, a structure which is fixed in the map of game, and a structure which is dynamic in the interior design of the map has been obtained. In order to calculate the user's number of movements, the distance has been determined as the main factor and based to the distance, many different scenarios and stories depending to artificial intelligence have able to created inside the game.

The most prominent difference of this study from the previous works is that it is a study developed that is depending rule-based artificial intelligence technique. Otherwise, a story written by the developer would have to be loaded to artificial intelligence. Then, artificial intelligence would need to reveal the story from the movements made according to the developing situations.

A constant game structure and a non-fluent game are obtained in intermediate level games where the rule-based artificial intelligence technique is not used. In this intermediate level game design, to use rule-based artificial intelligence has saved time to the developer and made easiness. This situation has enabled the user to become fluency with different story contents in the game.

The in-game story(s) develops instantaneously and depending the user's movements in main story of the game. It is not known how to develop the game differently with different moves each time. Therefore, every possibility should be considered in new game designs. The stories that artificial intelligence can generate should be considered and tested to make sure that there is no errors in the game.

4. Acknowledgements and Future Works

This study was carried out within the scope of a thesis at Muğla Sıtkı Koçman University, Faculty of Technology. We would like to thank all the faculty staff who contributed to the work with their suggestions and ideas. Should the people who are concerned to access the project source file, materials, maps, and objects, send an e-mail to omersansen16@gmail.com

5. References

[1] Labschutz M., Krosl K., Aquino M., Grashaft F., Kohl S., and Preiner R. (supervisor), Content Creation for a 3D Game with Maya and Unity 3D, Proceedings of the 15th Central European Seminar on Computer Graphics, p.p. 1 – 8, 2011.

[2] Jie J., Yang K., Haihui S., Research on the 3D Game Scene Optimization of Mobile Phone Based on the Unity 3D Engine, International Conference on

Computational and Information Sciences, ISBN: 978-0-7695-4501-1, Chengdu, Sichuan Chengdu, pp: 875-877, Oct. 21 to Oct. 23, 2011.

[3] Kumar S., Hedrick M., Wiacek C., Messner J.I., An Experienced-Based Design Review Application For Healthcare Facilities Using A 3D Game Engine, Journal of Information Technology in Construction, ISSN: 1874-4753, ITcon Vol. 16, , p.p. 85–104, 2011.

[4] Bae J.H., Kim A.H., Design and Development of Unity3D Game Engine-Based Smart Social Network Game, International Journal of Multimedia and Ubiquitous Engineering, Vol.9 No.8, pp.261-266, 2014.

[5] Doğan A., Artificial Intelligence, Kariyer Publishing, 2002.

[6] Satman G., Unity 3D, KODLAB Publishing, 2015.

[7] Yılmaz A., Artificial Intelligence, KODLAB Publishing, 2017.

[8] Menard M., Game Development with Unity, 2011.

[9] Lukosek G., Dickinson C., Doran J., Unity 5: Learning C# by Developing Games, 2016.

[10] https://assetstore.unity.com/Access: 02.09.2018.

[11] Randima F. (editor), GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Time Graphics, Reading, MA: Addison-Wesley, 2004.

[12] Lengyel E., Mathematics for 3D Game Programming and Computer Graphics, Second Edition. Hingham, MA: Charles River Media, 2003.

[13] Kerlow I.V., The Art of 3-D Computer Animation and Imaging (Second Edition). New York, NY: John Wiley and Sons, 2000.

[14] Salen K. and Zimmerman E. Rules of Play: Game Design Fundamentals, the MIT Press, 2003.

[15] Smed J, Hakonen H., Towards a Definition of a Computer Game, TUCS Technical Report No 553, Turku Centre for Computer Science, ISBN 952-12-1217, ISSN 1239-1891, September 2003.

[16] Bettner P., Terrano M., 1500 Archers on A 28.8: Network Programming in age Of Empires And Beyond, GDC2001, Vol. 2, Pages 30, 2001.

[17] https://www.ageofempires.com/ Access: 03.09.2018.

[18] Lopes R. and Bidarra R, Adaptivity Challenges in Games and Simulations: A Survey, CIG, Pages 83–108, 2011.

[19] Hatipoğlu T., Game Programming with Unity 3D, KODLAB Publishing, 2016.

[20] Ünsal M., Game Developing with Unity 3D, Abaküs Publishing, 2015.

[21] Karaboğa D., Artificial Intelligence Optimization Algorithms, Atlas Publishing House, Pages 75-112, İstanbul, 2004.

[22] Chen J., Flow in Games (and Everything Else). Commun, ACM, Pages 31–34, 2007.