Turkish Journal of Engineering
https://dergipark.org.tr/en/pub/tuje
e-ISSN 2587-1366

Word-based game development on Android with an efficient graphical data structure

Mustafa Batar[*1]

[1]*Burdur Mehmet Akif Ersoy University, Faculty of Engineering and Architecture, Department of Computer Engineering, Burdur, Turkey*

Keywords
Word-Based games
Mobile games
Android
Graphs
DAWG

ABSTRACT

Today, new games are released every day, and the virtual reality market is developing in a similar way to the rapidly growing smartphone ecosystem about 10 years ago. In addition to this, mobile games that take place with smartphones in people's daily lives can be downloaded to their phones for free, without paying any money, only with an internet connection. In this sense, a mobile game on android has been designed and developed about word games for this study. These word games need to have fast feedback and fast research time to the users and the players. In this context, Directed Acyclic Word Graph (DAWG) has been used and applied for giving fast feedback in the developed game "Kelimetris" in the study. The game "Kelimetris" has been explained in detail step by step with showing its captures, screenshots, UML diagrams and code blocks. In addition, this study has showed – the graphical data structure – DAWG's efficiency and usability in word-based games on mobile phones on Android. As a result, this study will have had a positive effect on the relationship between data structures and mobile games with the contribution of the developed game "Kelimetris" and the finite state machine DAWG.

1. INTRODUCTION

Cultural historian and philosopher Johan Huizinga has one of the most enduring evaluations of the game. Huizinga game; "A voluntary action or activity that is freely consented, but carried out within the limits of certain time and place in accordance with fully mandated rules, has a purpose in itself, accompanied by a sense of tension and joy and a consciousness of 'being different' from 'ordinary life'." (Huizinga, 1995). It is possible to adapt the definition of game that Johan Huizinga has stated in this paragraph to games (adventure, simulation, action, platform games, and etc.) (Yılmaz, & Çağıltay, 2015).

The game is directed by the players, within the framework of their own wishes, the rules of the game and depending on the rules. In a game, no matter what kind of game the player plays, s/he has to comply with the rules drawn by the game scenario or the group principles developed by the virtual groups established in the game for her/his "success in the game" (Bates, 2004). It depends on the game scenario; the character cannot go out of it in order to reach his goal. Within the framework of these rules, the place where the player is located is the virtual digital game world. In addition, it is the place where the player can interact with other players in the game where the character's struggle continues. Dialogue during game streaming, especially with other players in the virtual world can establish. At the same time, the player can develop their characters in the games and provide a financial income on top of that (Aarseth, 2001).

According to Huizinga; at the same time, s/he stated that after playing once, the game can be conveyed as having a spiritual value in the memories and can be repeated at any time (Aarseth, 2003). Although some rules are games, they are not independent of everyday life. As a result, the addiction of the game increased. The game and game stages have gained new identities and images with technologies that have reached virtual reality, and have continued to take place in different positions in human life with new functions and different gameplay diversity (Su & Zhao, 2011).

Nowadays, there are thousands of different types of games in the market (Rouse, 2005). Generally, these games are related to war, strategy, race, gambling, etc. When it is looked at the games, it is realized that some users play them to take pleasure, some of them play to spend their times. However, in this study, a word-based

* **Corresponding Author**
*(mbatar@mehmetakif.edu.tr) ORCID ID 0000-0002-8231-6628

Cite this article
Batar M (2022). Word-based game development on Android with an efficient graphical data structure. Turkish Journal of Engineering, 6(3), 256-261

Research Article / DOI: 10.31127/tuje.987141
Received: 25/08/2021; Accepted: 07/09/2021

game has been chosen in order to design, develop and implement.

Word games and puzzles are spoken or board games often designed to test ability with language or to explore its properties. It has to be thought that it is to benefit individuals in several ways (Mitchell, 2001). These games enrich vocabulary of users while enjoying, they improve user's memory and exercise their mind muscles for long-term learning and retention. Thus, they have slight contribution to the users not to catch Alzheimer's disease. Furthermore, children can learn and understand new and complex words in an easy and enjoyable way. Moreover, these games require the person to read and write. After a time, they improve a person's reading and writing speed. Also, text-based games can help people (especially people in hospital, whether admitted there or waiting to hear news about a loved one) to relieve a stressful day in a way that other games cannot (Ahl, 1983).

In this study, a word game which is named "Kelimetris" has been designed and developed on Android. In "Kelimetris", stones are falling rapidly from top to bottom. One has to click on the stones to select the letters. What is needed to do is to derive meaningful words by selecting the right stones. There is no need to create words related to the each other. For each correct word, user gets points based on the points of letters. When the screen is filled with stones, the game is over.

2. The Game "Kelimetris"

General Game Functionality has described main game menu. This menu has included four main sections: "How to Play", "About the Game", "Best Players" and "Start Game". Figure 1 has showed the use cases in general game functionality in the developed game "Kelimetris".

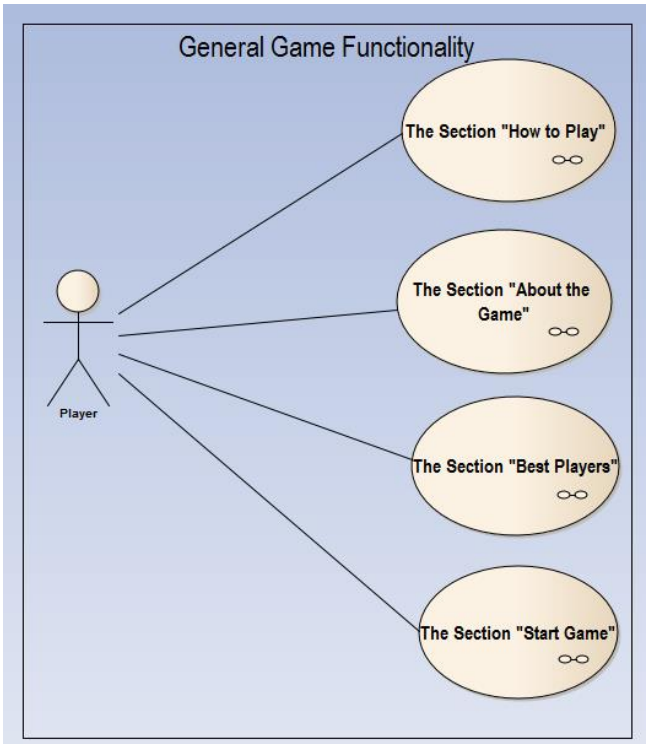

Figure 1. Use cases of the general game functionality

Figure 2. Screenshot of main game menu

In order to start the game (the main menu of the game has been given in Figure 2), if the player doesn't have a profile, the player has to create a new profile or if it exists, existing profile has to be selected. After the game is started, the player has to try to construct words by selecting as many letters as possible. The player increases his/her own score for each valid word after submission. Otherwise, s/he needs to construct new ones. In addition to this, the activity diagram of the player in order to play the game "Kelimetris" has demonstrated in Figure 3.

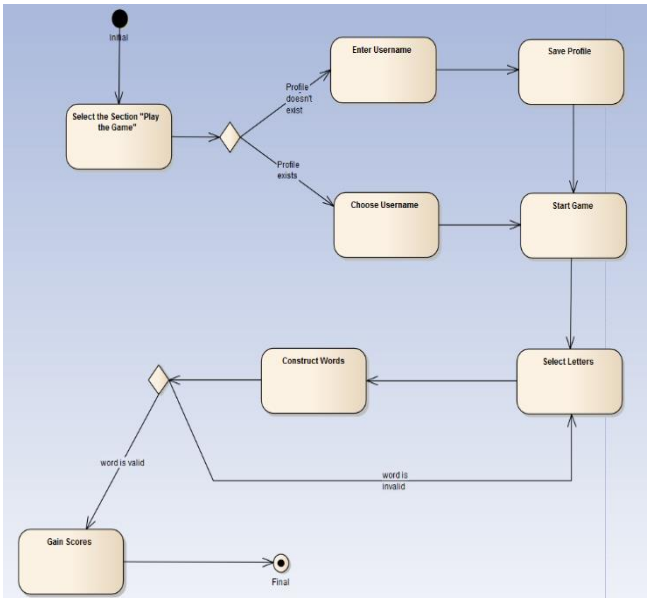

Figure 3. Activity diagram of the section "play game"

In "Kelimetris" game scenario, once the player starts the game in the main menu, then the stones that embedded a character drop consecutively in an order. The stones land on the top of each of one after another through a pattern and the stone locations are different to prevent that one stone overlaps the other. A location of a stone is not given as a location of another one before the stone leaves the window. Everything about the arrangement of the locations to the stones is done in Port super class. Also, it is needed to define some classes extending Port: these are Port1, Port2, Port3, Port4 and Port5 which are specialized to settle the stones in different columns. These ports have been captured in Figure 4 in the following.

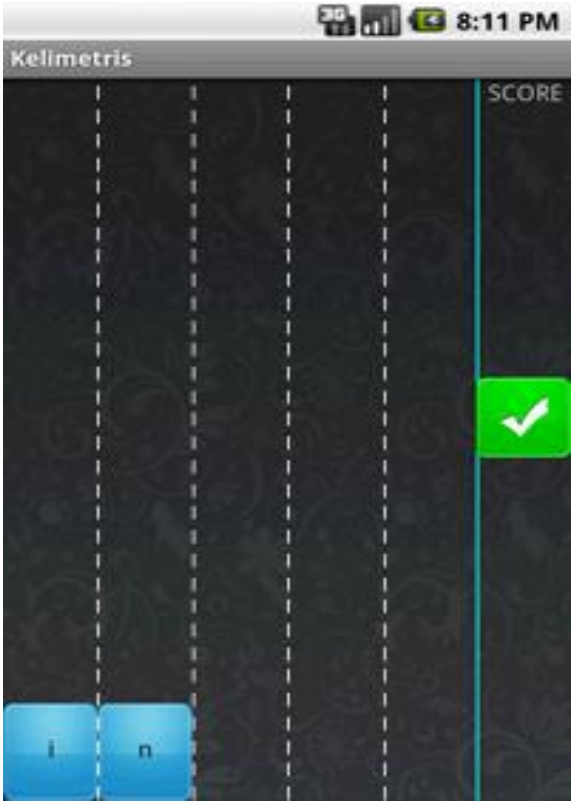

Figure 4. Capture of ports in the game

Each stone location is an integer variable and they are kept in an array named as Locations[] in port class. When any one of the classes called Port1, Port2, Port3, Port4, Port5 which extend Port is initiated, base elements of Locations[] array are initialized to -1 value by the initializeLocations() method. That means all the locations in that port are available at the beginning to settle a stone.

```
int [] Locations=new int[7];
int btnId=0;

Port(){
        initializeLocations();
}
public void initializeLocations(){
        int i;
        for(i=0;i<=6;i++){
                Locations[i]=-1;
        }
}
```

When a stone is created, its location needs to be assigned to it, therefore getButtonLocation() method gives an available location by traversing in Locations[] array and comparing if an element of it is -1. If it is, then it means this location is available and the method returns the index with the element of -1.

```
public int getButtonLocation(){
        int x=0,i;
        for(i=0;i<=6;i++){
                if(Locations[i]==-1){
                        x=i;
                }
        }
        return x*100;
}
```

After returning the index that represents an available location, the element at that index is assigned to the button ID value by insertButton() method. That means the location is not available for the coming other stones.

```
public void insertButton(){
        int i,current=0;
        for(i=0;i<=6;i++){
                if(Locations[i]==-1){
                        current=i;
                }
        }
        Locations[current]=btnId;
}
```

There is an updateLocations() method which is invoked if the created word is confirmed. The method updateLocations() actually gets IDs of the buttons that were used to make meaningful word, as the word has a meaning, the letters of that word must be removed from the game window. Their locations also have to be available in the Locations[] array, updateLocations() replace the ID values regarding the past word by -1. As the array is used to keep locations, there is no dynamically changing size of the collection after the stone is hidden (as shown in Figure 5), just the value of its location index in Location[] array in Port class is assigned to -1. Assigning -1 means that location is made available to land another stone. New dropping stones – as given in Figure 6 – come to these empty locations.

```
void updateLocations(Vector<Integer> Clickeds){
        int i;
        for(i=0;i<=6;i++){
                if(Clickeds.contains(Locations[i])){
                        Locations[i]=-1;
                }
        }
}
```

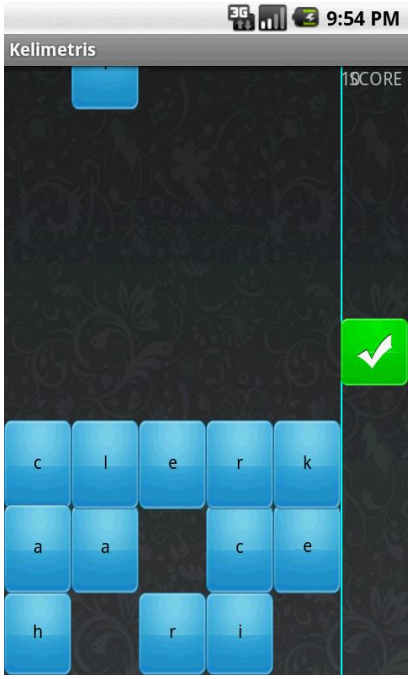

Figure 5. Capture of hidden stones in the game

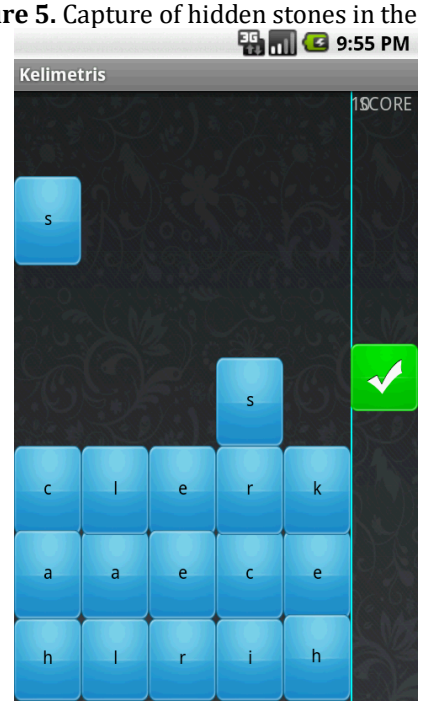

Figure 6. Replacement of new stones in the empty areas

Also, there is isLocationAvailable() method which returns true if a Location[] array includes at least one -1 value meaning available location exists. It is used to make the game over, if the game window is filled, then game has to be stopped. In this context, the location adjustment has been shown in Figure 7 in the following.

```
boolean isLocationAvailable(){
        int i;
        for(i=0;i<=6;i++){
                if(Locations[i]==-1){
                        return true;
                }
        }
        return false;
}
```

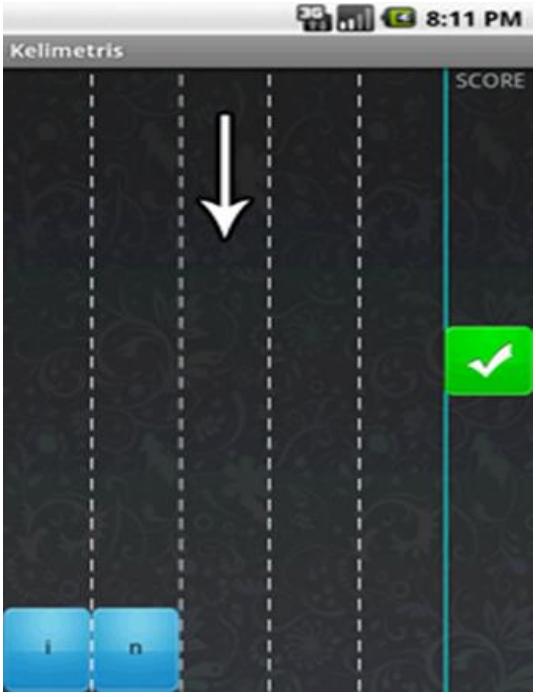

Figure 7. Drop of the stones in the game

A mechanism is needed to provide an immediate movement to each stone after it is created. It is seen that the movement of the stone (as given Figure 7 above) can be succeeded by assigning a layout parameter to this, by changing some values of this parameter periodically, and Mytask class, which extends TimerTask class, provides the mechanism. In addition, extending TimerTask class gives us an advantage to provide a continual movement by invoking a method in a specific time period.

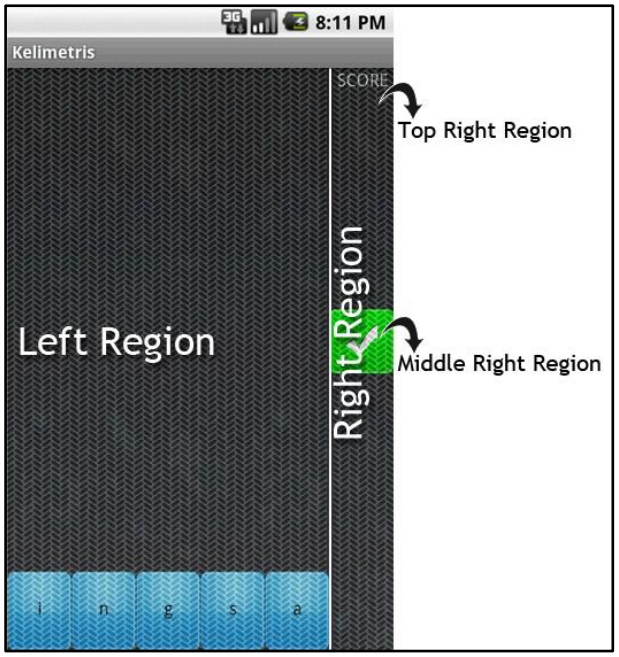

Figure 8. The regions of the window in the game

The game window has been divided into some parts as shown in Figure 8 above: the score is displayed on the right side of the layout and the buttons drop on the left side of the layout. There is a stable button stands for submission, when it is clicked; the word generated by the player is sent for looking up to a function of LetterFactory instance. If the word is confirmed, the Flag variable is

assigned to be true and the update locations methods regarding each port are invoked, update locations methods result in removal of the buttons which were used for obtaining last confirmed word. A variable called portcounter type of integer is assigned, it keeps track of at which port a button will drop.

When a stone is generated, a letter is assigned to it. We have lines of words of a text file which the words are randomly taken from, and shuffled. Each character of these shuffled words is set one by one to each released stone. Words are used when their all letters are assigned to the stones, and then the new words are brought from the file. This process goes on as long as the game is being played. All things that regarding the above are done by LetterFactory class. The method addFromFile() opens a text file and fetches corresponding to valid words at the random lines in the file. Then fetched words are pushed into a vector named words. wordsIndexes are random integers corresponding to random lines in the text file so as to get the words randomly from the file.

In addition to this, in computer science, a directed acyclic word graph (DAWG) (Appel & Jacobson, 1988) is a data structure that represents a set of strings, and allows for a query operation that tests whether a given string belongs to the set-in time proportional to its length. In these respects, a DAWG (Crochemore & Vérin, 1997) is very similar to a tree, but it is much more space efficient. The entry point into the graph represents the starting letter in the search. Each node represents a letter, and you can travel from the node to two other nodes, depending on whether you the letter matches the one you are searching for. It is a directed graph (Aoe, Morimoto, Shishibori & Park, 1996) because one can only move in a specific direction between two nodes. In other words, one can move from A to B, but one can't move from B to A. It is Acyclic (Perrin, 1990) because there are no cycles. One cannot have a path from A to B to C and then back to A. The link back to A would create a cycle, and probably an endless loop in your search program. The structure of DAWG (Jansen & Boekee, 1990) has been given in Figure 9.

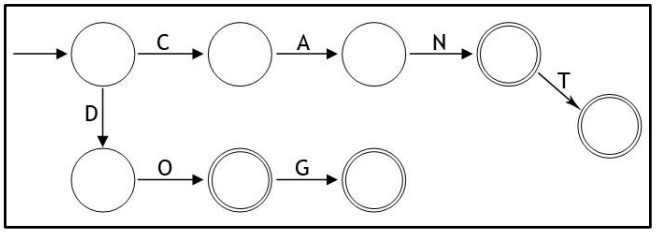

Figure 9. DAWG working structure

Based on DAWG in Figure 9, the words "can", "can't", "do" and "dog" could be gained and created easily.

Moreover, each letter has its own point. The points have been determined based on their usage rate. That means if a letter places in many words, its point will be less than the others that place rarely. Table 1 in the following has showed the standard points of each letter in the game "Kelimetris" for scoring.

Table 1. Letter points in the game

Letter	Value	Letter	Value	Letter	Value
A	1	I	2	R	1
B	3	İ	1	S	2
C	4	J	10	Ş	4
Ç	4	K	1	T	1
D	3	L	1	U	2
E	1	M	2	Ü	3
F	7	N	1	V	7
G	5	O	2	Y	3
Ğ	8	Ö	7	Z	4
H	5	P	5		

3. Results of the Game "Kelimetris"

As the consequences of the performance test in which trying to search in a text file for a particular word, it results that it takes 0,8 seconds for a file that consists of 3.000 lines of words. However, when it comes to handle a file that consists of 60.000 lines of words, approximately it takes over 10 seconds. There is need a dictionary which includes at least 80.000 words to be played in the game. Therefore, DAWG Algorithm implementation has been applied and developed for this game "Kelimetris". The results of searching a word in "Kelimetris" have been given in Table 2 in the following.

Table 2. Word search time in the game "Kelimetris"

The Technique	The Number of The Words	The Search Time
Read from the File	1000	~0,6 seconds
Read from the File	48000	~10 seconds
DAWG	1000	~0,001 seconds
DAWG	48000	~0,5 seconds
Read from Database	1000	~1 seconds
Read from Database	48000	~5 seconds

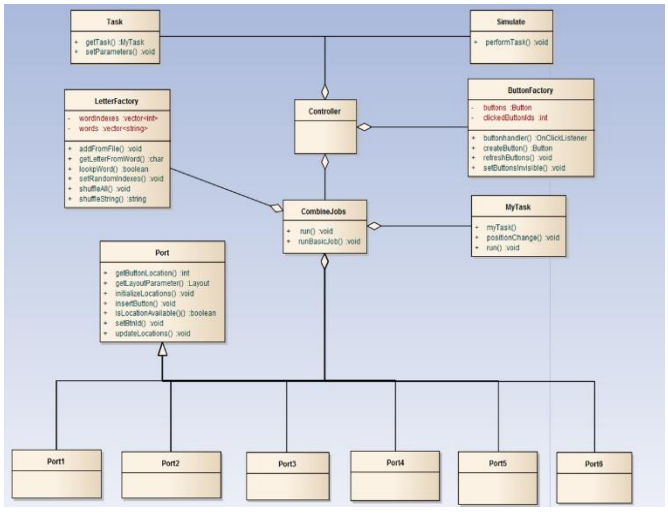

Figure 10. The class diagram of the game

According to data in Table 2, it has been easily seen that DAWG is very efficient and useful data structure in searching a word in a mobile game with its fast feedback.

Finally, the game "Kelimetris" has been designed, developed and accomplished to complete based on the class diagram shown in Figure 10.

4. CONCLUSION

As a result of the development of technologies, the game industry has become a large economic market in a short time. From the first electronic game in 1947, to Tennis for Two in 1958, from Pong to Pac-Man, from Commodore 64 to Gameboy and Playstation, from Snake on a Nokia mobile phone to social media games, from Angry Birds on smartphones Until the first Pokemon Go game, which was developed using virtual reality, the game industry has always made progress, except for short pauses. By making large investments in the mobile game market, companies have entered into the competition of game making. The increase in internet connection speed, tablets, smartphones and social media have made the gaming industry a constantly evolving industry.

Based on this evolving gaming industry, there is need to give fast responses to the players and the users in the games. In this context, the developed game "Kelimetris" on Android in the study has given a graphical data structure DAWG (directed acyclic word graph) to improve fast feedbacks and to shorten waiting period in the game. Thus, the study has brought an innovation into the mobile game development, and the literature as well.

ACKNOWLEDGEMENT

Ethics committee approval document is not required for this study, and the author declares that there is no conflict of interest.

Conflicts of interest

The authors declare no conflicts of interest.

REFERENCES

Aarseth E (2001). Computer Game Studies, Year One, Game Studies: International Journal of Computer Game Research, 1(1).

Aarseth E (2003). Playing Research: Methodological Approaches to Game Analysis. Computer Game Theory Compendium.

Ahl D H (1983). Creative Computing Video & Arcade Games.

Aoe J, Morimoto K, Shishibori M & Park K-H (1996). A Trie Compaction Algorithm for a Large Set of Keys, IEEE Transactions on Knowledge and Data Engineering, 8(3), 476-491.

Appel A W & Jacobson G J (1988). The World's Fastest Scrabble Program. Communications of the ACM, 31(5), 572-578.

Bates B (2004). Game Design: The Art and Business of Creating Games. Boston, MA: Thomson Course Technology.

Crochemore M & Vérin R (1997). Direct Construction of Compact Directed Acyclic Word Graphs, 8th Annual Symposium, CPM 97, Aarhus, Denmark, 116-129.

Huizinga J (1995). Homo Ludens, (Çev. MA. Kılıçbay), İstanbul: Ayrıntı Yayınları.

Jansen J A & Boekee D E (1990). On the significance of the directed acyclic word graph in cryptology, Advances in Cryptology — AUSCRYPT '90, Lecture Notes in Computer Science, 453, Springer-Verlag, pp. 318–326, doi:10.1007/BFb0030372, ISBN 3-540-53000-2.

Mitchell B L (2001). Game Design Essentials, Indianapolis: John Wiley & Sons, Inc.

Perrin D (1990). Finite Automata, in: J. van Leeuwen, ed., Handbook of Theoretical Computer Science, Elsevier, Amsterdam, Vol. A, 3-57.

Rouse R (2005). Game Design, Theory and Practice (Wordware Game Developer's Library). USA: Worldware Publishing.

Su H, Zhao V (2011). Alive Character Design: For Games, Animation and Film, Beijing: CYPI Press.

Yılmaz E & Çağıltay K (2015). History of Digital Games in Turkey, Digra Uluslar Arası Konferansı.