

IRSTI 14.15.01

DOI 10.59941/2960-0642-2023-4-37-47

Implementation of artificial intelligence in “ToqyzQumalaq” mobile logic game

Zh. Nurbekova^{1*}, G. Aimicheva², T. Tolganbaiuly³, M. M. Galy⁴

¹Abai Kazakh National Pedagogical University, Almaty, Republic of Kazakhstan

^{2,3}Eurasian National University named after L. N. Gumilev Astana, Kazakhstan

⁴Opn, Bangkok, Thailand

*Correspondence: zhanat_n@mail.ru



Abstract. This research paper delves into the implementation of artificial intelligence (AI) in the mobile logic game “ToqyzQumalaq,” focusing on incorporating advanced algorithmic strategies to improve gameplay. The game’s complexity and strategic depth present unique challenges in AI development, addressed through the integration of algorithms like Minimax, Alpha-Beta Pruning, Greedy, and Particle Swarm Optimization (PSO). The study emphasizes the creation of evaluation functions for these algorithms, ensuring AI efficiency and human-like decision-making. This aspect is vital for maintaining the strategic unpredictability essential to “ToqyzQumalaq.” Extensive experimental testing against human players of various skill levels demonstrates the algorithms’ effectiveness. These tests reveal the strengths and limitations of each algorithm, providing insights into their application in the game. This paper contributes to AI in gaming, highlighting the challenges and opportunities in developing AI for complex games. Its findings are relevant not only to game developers but also serve as an educational tool, showcasing the practical application of AI and algorithmic strategies.



Keywords: mobile game development, machine learning, Minimax, Alpha-Beta Pruning, Greedy, PSO.



How to cite:

Nurbekova, Zh., Aimicheva, G., Tolganbaiuly, T., Galy, M. M. Implementation of artificial intelligence in “ToqyzQumalaq” mobile logic game [Text] // Scientific and pedagogical journal “Bilim”. – Astana: NAE named after Y. Altynsarin, 2023. – №4. – P. 37-47.

Introduction

“ToqyzQumalaq” is a national Kazakh board game for two players. The game is played by two players, who have 2×9 slots with 9 balls in each slot (Fig.1). The goal of the game is to get more balls in its own “Stone bank”. Moves are made by players alternately by successive spreading balls between slots from left to right. If the last ball falls into the opponent’s slot, bringing balls count to an even number, all of them displacement to the player, who made a move.

The game is up when one of the players will not have a ball to make a move – such a losing situation is called Hollow (Atsyrau) (Fig.2). If during the game the number of stones in one of the slots becomes equal to three, this slot goes into a special status called Sacred (Tuzdyk). The benefit of Sacred slot is that each ball falling in Sacred moves to the “Stonebank” of “Sacred” owner (Fig.3).

The aim of this study is to develop a mobile logic game “ToqyzQumalaq” with an

emphasis on artificial intelligence and machine learning implementation for optimal gaming decision making.

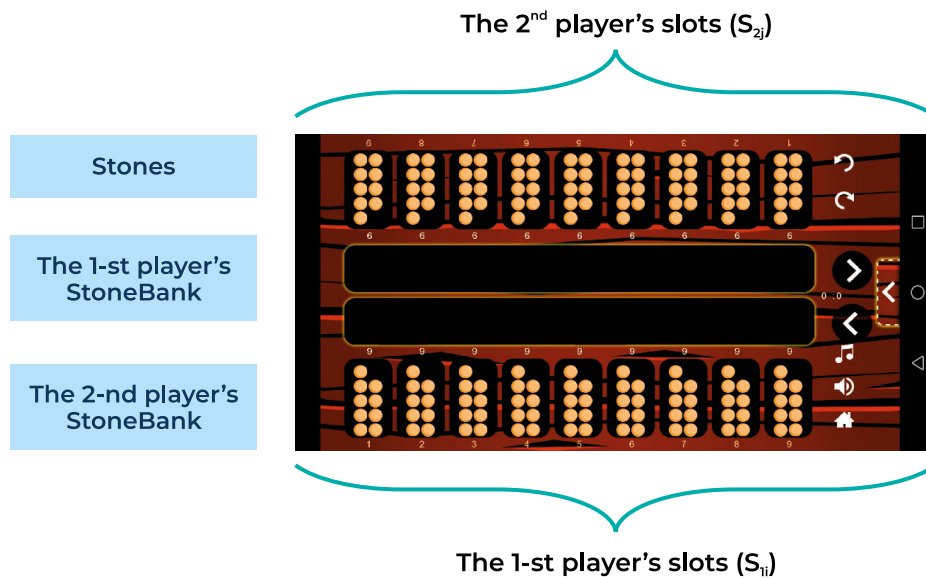


Figure 1. Initial state of the game board "ToqyzQumalaq"

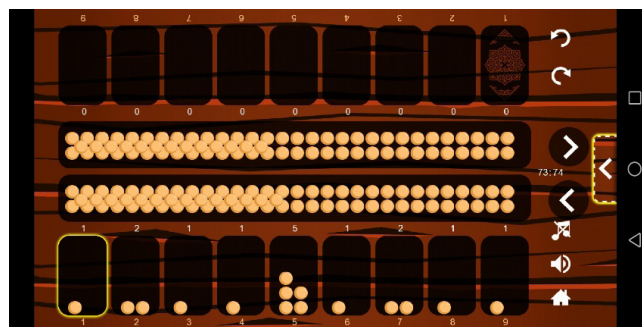


Figure 2. The "ToqyzQumalaq" game completion with the result 73:74 in 2nd player's favor

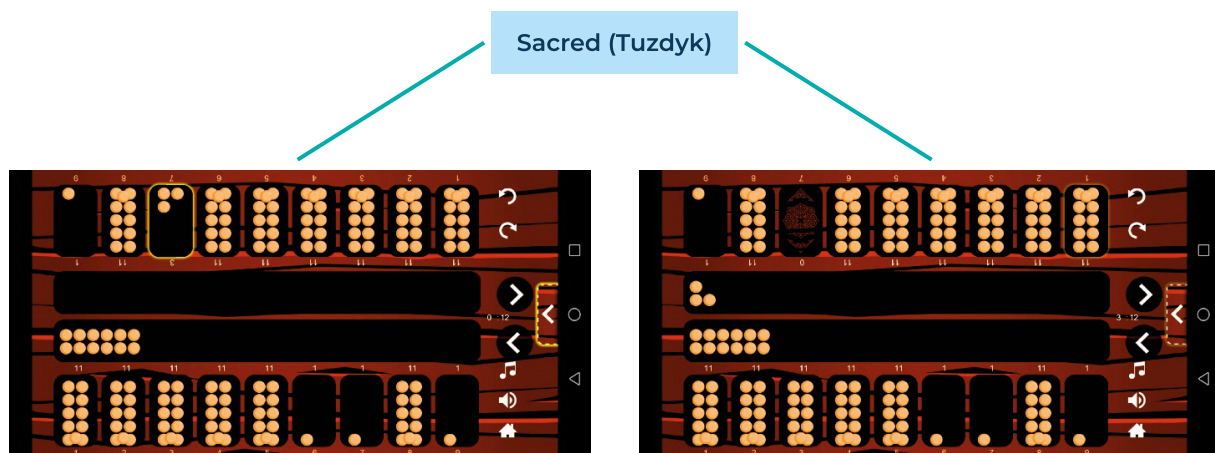


Figure 3. The slot №7 of the first player goes into a special status called Sacred (Tuzdyk)

Implementation of artificial intelligence and machine learning in mobile game applications requires utilizing the special search algorithms for optimize gaming decisions. Theoretical research in this field confirms relevance search optimization algorithms such as Minimax [1-3], Alpha-Beta Pruning [4], Greedy [5], PSO [6], suitable for such logic games.

In developed mobile game the artificial intelligence was implemented on three game levels as Easy, Medium and Hard using PSO, Greedy, Minimax and Alpha-Beta Pruning algorithms. The General mobile application architecture with the selection of artificial intelligence modules is shown in figure 4. Game rules and gameplay are modeled and implemented for Android and iOS platforms.

The efficiency of applied algorithms in respect to the making decisions by artificial intelligence have been estimated on the basis of experimental results. Experiments have shown the efficiency of the Greedy algorithm by comparison to other algorithms.

Development of mobile logic game “ToqyzQumalaq”

The formalization of requirements to the mobile logic game. “ToqyzQumalaq” is qual-

ified as a board combinatorial logic game. To formalize the rules and requirements to the gameplay the following mathematical model has been set up:

$$\sum_{i=1}^9 S_{1i} + \sum_{j=1}^9 S_{2j} + \sum_{i=1}^9 Sb_i = 182 \tag{1}$$

under constraints:

$$\begin{cases} S_{ij} \geq 0, \text{ for } i = \overline{1,2}; j = \overline{1,9} \\ 0 \leq Sb_i \leq 182, \text{ for } i = \overline{1,2} \end{cases}$$

where,

- S_{1i} – 1st player's slots,
- S_{2j} – 2nd player's slots,
- Sb_i – ith player's stonebank.

The initial state of the game model is set by the following conditions:

$$\begin{cases} S_{ij} = 9, \text{ for } i = \overline{1,2}; j = \overline{1,9} \\ Sb_i = 0, \text{ for } i = \overline{1,2} \end{cases} \tag{2}$$

The architecture of the mobile logic game “ToqyzQumalaq” has shown at figure 4.

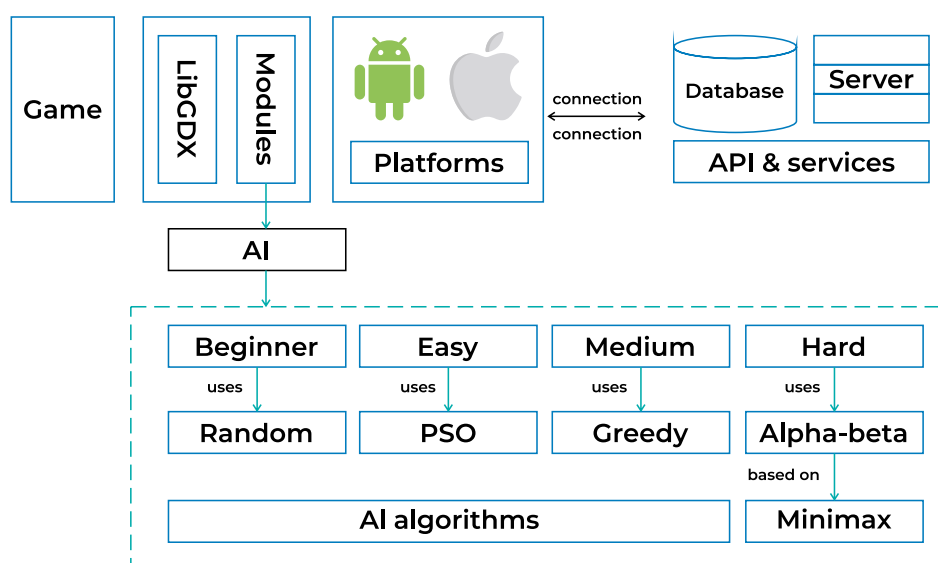


Figure 4. The architecture of the mobile logic game “ToqyzQumalaq”

Game mechanics of the mobile logic game “ToqyzQumaldaq”

One of the mobile game applications development stages is the game mechanics design, which is in the programming of player actions: considering the position of the special slot Sacred (Tuzdyk), player loss - Hollow (Atsyrau), control of in-game achievements are effective moves, wins, winning streak.

From the computational viewpoint of game mechanics design, the development and implementation of artificial intelligence using search algorithms for optimize gaming decisions is a laborintensive type of work.

The artificial intelligence is implemented on the three game levels:

- Easy level. For the completion of a move a machine learning module is used which is based on the PSO algorithm. For learning, the mobile application plays with itself, taking into account only such results of the game as victory, drawn, defeat. Based on this data, the evaluation function has been built;
- Medium level. For the completion of a move the Greedy algorithm is used, which ensure to get as many balls as possible and, at the same time, gives the minimum possible balls to the opponent;
- Hard level. For the completion of a move the Alpha-Beta Pruning algorithm is used which have been ensured to protect player's slots and to make the most effective game in the player loss situation Hollow (Atsyrau).

An important step in implementation search algorithms for optimize gaming decisions is in the building of the evaluation function of the game decision tree [4,5,6].

Considering the analytic formulation of the evaluation functions for each algorithm artificial intelligence in the mobile logic game “ToqyzQumaldaq” is used to implement.

The evaluation function for the Minimax and PSO algorithms is as follows:

$$E = \begin{cases} (Sb_i - Sb_{i+1}) + Tp_i - Tp_{i+1}, & \text{for } i = 1 \\ (Sb_i - Sb_{i-1}) + Tp_i - Tp_{i-1}, & \text{for } i = 2 \end{cases} \quad (3),$$

where,

E – evaluation function,

Sb_i – i th player's stonebank,

Tp_i – value of evaluation function for i -th player's Sacred slot.

The analytic formulation of the Greedy evaluation function is as follows:

For the first player:

$$E = (Sb_1 - Sb_2) * 5 + Qs_1 * 20 + Tp_1 * 15 - Tp_2 * 15 - Q_1(odd) * 5 + Q_1(even) * 5 \quad (4).$$

For the second player:

$$E = (Sb_2 - Sb_1) * 5 + Qs_2 * 20 + Tp_2 * 15 - Tp_1 * 15 - Q_2(odd) * 5 + Q_2(even) * 5 \quad (5),$$

where,

E – evaluation function,

Sb_i - i th player's stonebank,

Tp_i -value of evaluation function for i -th player's Sacred slot,

Qs_i - the number of balls that i -th player can win,

$Q_i(odd)$ -the number of the i -th player's odd slots.

The evaluation function for the player's Sacred (Tuzdyk) slot in relation to the queue is implemented in all algorithms and is as follows:

$$Tp_i = \begin{cases} 2, & \text{if slot} = 1 \\ 4, & \text{if slot} = 2 \\ 6, & \text{if slot} = 3 \\ 15, & \text{if slot} = 4 \\ 14, & \text{if slot} = 5 \\ 8, & \text{if slot} = 6 \\ 7, & \text{if slot} = 7 \\ 5, & \text{if slot} = 8 \end{cases}, \text{ for } i = \overline{1,2} \quad (6)$$

Search algorithms to optimize gaming decisions are implemented in Java. The chunk of the code about implementation of the Greedy evaluation function is shown at figure 5. The Greedy algorithm takes into consideration several factors: how many slots are at risk, how many slots are protected, how many slots can be attacked, the maximum number of balls that can be obtained,

the maximum number of balls that can be protected, the evaluation of the slot Sacred (Tuzdyk) depending on its position, and the difference between the number of balls at players. The parameters of Greedy evaluation function have their own coefficient that can make the game aggressive or defensive, depending on which coefficient is greater: at parameter “eat” or “protect”.

```
public int eval(){

    if(globalPlayer == 1) {
        int evaluation = 0;
        evaluation+=(P1StoneBank - this.P2StoneBank)*5;
        evaluation += this.QSP1*20;
        evaluation+=evalSacred(1)*15;
        evaluation-=evalSacred(2)*15;
        evaluation+=numberOfEven(this,1)*5;
        evaluation-=numberOfOdd(this,1)*5;
        return evaluation;
    }
    else{
        int evaluation = 0;
        evaluation+=(this.P2StoneBank-this.P1StoneBank)*5;
        evaluation += this.QSP2*20;
        evaluation+=evalSacred (2)*15;
        evaluation-=evalSacred (1)*15;
        evaluation+=numberOfEven(this,2)*5;
        evaluation-=numberOfOdd(this,2)*5;
        return evaluation;
    }
}
```

Figure 5. Implementation of the Greedy evaluation function in Java

In evaluating the Sacred slot (Tuzdyk) the highest coefficients have slots number 4 and 5. The lowest coefficient for Sacred (Tuzdyk) has slot № 1. Such evaluation of the Sacred slot position applies to both players. However, this evaluation can be changed

depending on the stage of the game, and mutual slots Sacred (Tuzdyk). The chunk of the code about implementation of the Sacred slot (Tuzdyk) evaluation function is shown in figure 6.

```
public int getEvalSacred(Greedy obj,int player){
    int eval = 0;
    if(player == 1)
    {
        switch(obj.sacred[0])
        {
            case 8: eval += 2;break;
            case 7: eval += 4;break;
            case 6: eval += 6;break;
            case 5: eval += 15;break;
            case 4: eval += 14;break;
            case 3: eval += 8;break;
            case 2: eval += 7;break;
            case 1: eval += 5;break;
        }
    }
}
```

Figure 6. Implementation of the Sacred slot evaluation function (Tuzdyk) in Java

The Minimax algorithm chooses the next move for which the value of the evaluation function will be the most effectively with minimal losses for the player who is making moves.

The Alpha-Beta Pruning algorithm evaluates a branch of the search tree and can be terminated prematurely without calculating all values of the evaluation function if it is found that the evaluation function value for that branch is in any case worse than the one calculated for the previous branch. The Alpha-Beta Pruning evaluation function assumes making an efficient move, to protect the slots, or to distribute rich slots, where there are a lot of balls, or to continue game by completing slots with the single ball and slots with 2 balls to avoid creating by opponent Sacred slot (Tuzdyk).

The Minimax and the Alpha-Beta pruning algorithms take into account the position of the Sacred slot (Tuzdyk) and the difference between the number of balls at players in a certain game situation.

The chunk of the code the Alpha-Beta Pruning evaluation function is shown in figure 7.

This program code is checked all non-empty slots on the ability to an effective move by using the method *getSelectedPit()*. In the beginning, the program evaluates the moves from the slots, based on the difference between the next states of the Stone-Banks. The move with the best difference is automatically chosen. The chosen move is verified on the effectiveness by *isProfitable()* function. If the chosen move have a good value of effectiveness, it will be selected, otherwise a game has to go to the protection mode. Method *toProtectSlots()* checks the presence of the slots that need to be protected. If any, *protectSlots(mainBoardState)* method searches a move, able to protect as many slots, without giving a chance to the opponent to make an effective move. If this condition is not met, the program searches for rich slots, slots with the maximum number of balls using the *hasRichSlots()* method. It is impractical to store a large number of balls in one slot. Therefore, sometimes we need to break such slots through *distributeRichSlots()*. If the last condition is negative, it means that the game goes into the mode of shifting the balls to build traps. This procedure is performed using the *doSomething()* method.

In the Minimax and Alpha-Beta Pruning algorithms, the parameters are summarized and give a certain assessment of the game decision. For example, when player making

move from slot №5, the evaluation is equal 20. As a result, the slot with the highest evaluation is selected.

```
int nextAlphaBeta = mainBoardState.getSelectedPit();
System.out.println("next move + " + nextAlphaBeta);
if (mainBoardState.isProfitable(nextAlphaBeta,
mainBoardState) > 0) {
    mainBoardState.printBoard(mainBoardState);
    System.out.println("Profitable move");
    choice = nextAlphaBeta;
} else {
    if (mainBoardState.toProtectStones()) {
        System.out.println("Protection of Stones");
        System.out.println(mainBoardState);
        choice =
mainBoardState.protectSlots(mainBoardState);
        System.out.println("Protected");
    } else {
        if (mainBoardState.hasRichSlots()) {
            System.out.println("Rich");
            choice = mainBoardState.distributedRichSlots()
        } else {
            choice =
mainBoardState.doSomething(mainBoardState);
        }
    }
}
```

Figure 7. Implementation of the Alpha-Beta Pruning evaluation function

Testing and debugging

To estimate the speed and efficiency of the used algorithms the experimental testing has been carried out. Algorithms have been tested on computer platforms with the following hardware characteristics: processor - i7 4710HQ, clock frequency

- 2.5 GHz, RAM - 8 GB, video card - 4 GB NVIDIA GTX850M.

To find out which of the algorithms better determines the optimal moves for winning during the experimental testing for each level of difficulty, 100 games have been played. The experimental results are presented in tables 1-3.

Table 1. The percentage of the algorithms' wins playing against each other

Random moves	52%	99%	96%	95%	97%
Greedy	1%	63%	37%	22%	33%
Minimax	4%	76%	66%	55%	63%
Alpha-Beta Pruning	5%	78%	45%	59%	61%
PSO	3%	67%	37%	39%	54%
	Random moves	Greedy	Minimax	Alpha-Beta Pruning	PSO

Table 2. The percentage of algorithms' wins against players of different levels while White's play

Player (professional)	8%	1%	1%	3%
Player (amateur)	12%	4%	4%	6%
Player (beginner)	92%	84%	85%	87%
	Greedy	Minimax	Alpha-Beta Pruning	PSO

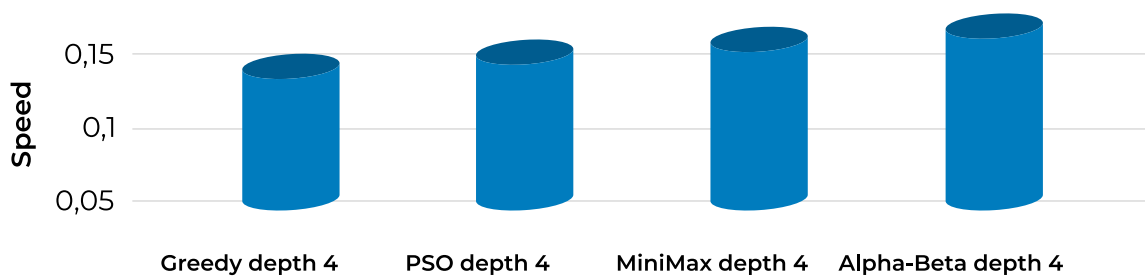
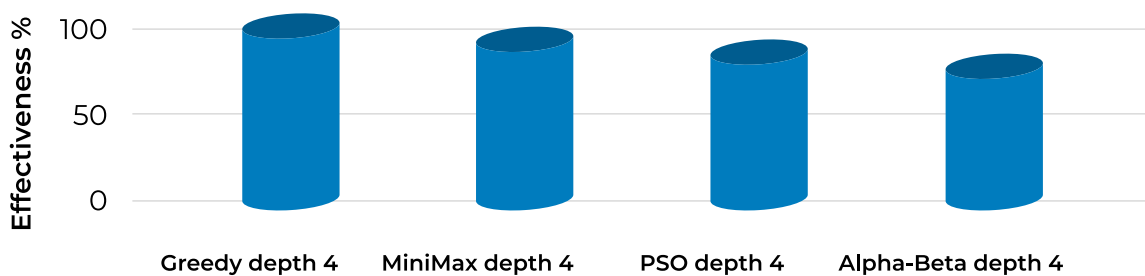
Table 3. The percentage of algorithms' wins against players of different levels while playing black

Player (professional)	5%	1%	1%	2%
Player (amateur)	9%	3%	3%	4%
Player (beginner)	90%	83%	82%	85%
	Greedy	Minimax	Alpha-Beta Pruning	PSO

During experimental testing the depth of the game tree was 4 half-moves. In case of increasing the depth of the game tree and providing more precision of the PSO evaluation function, artificial intelligence will play better. Considering that PSO was trained by playing with itself, it is currently playing at the amateur level. In the process

of self-training algorithm, PSO was considered only such results of the game as victory, drawn, defeat.

Estimation of the algorithms based on the represented test data showed that the first place is Greedy, then Minimax, PSO, Alpha-Beta Pruning (see Fig. 8-9).

**Figure 8. The rating of algorithms by «Computing speed»****Figure 9. Rating of algorithms by «Efficiency»**

In theory, Minimax and Alpha-Beta Pruning algorithms should work much faster than Greedy. But in reality, since the Greedy evaluation function is more precise, it works faster and more efficiently.

The main challenge in the “ToqyzQumalaq” game develop have been faced by programmers is a difficult and ambiguous system of assessing the situation. It happens due to several reasons:

- 1) The absence of scientific literature on the mathematical theory of programming game “ToqyzQumalaq”;
- 2) The parameters variability of the evaluation function at different levels of the game. Different players with different levels of “ToqyzQumalaq” game skills take into account different factors to achieve victory;
- 3) The necessity of taking into account a special slot Sacred (Tuzdyk). This element is the unique kind in the Mancala family of games, to which “ToqyzQumalaq” refers, and no one abstract logic games haven’t any analogue of Sacred (Tuzdyk). If in the chess the value of each piece is known, the process of evaluating the Sacred slot (Tuzdyk) is very difficult. Make it clear that even professional players are not able to answer the question of how to evaluate the value of Sacred (Tuzdyk). Value of Sacred (Tuzdyk) depends on various factors such as position, assignment stage, mutual position of Sacred slots (Tuzdyk), etc. In this respect, it is very difficult to give an precision mathematical evaluation of Sacred slot (Tuzdyk);
- 4) The difficulty of determining the end of the game Hollow (Atsyrau). In this game situation we need to act very accurately and perfectly correct;
- 5) The complexity of determining the decisive is mistaken. Often, it is very difficult to determine in game the moment where the mistake was made. In this indicator, “ToqyzQumalaq” is similar to Go (Chinese chess).

Conclusion

“ToqyzQumalaq” mobile logic game development allows investigation the issues of artificial intelligence implementation based on search algorithms for optimal gaming decision making. Minimax, Alpha-Beta Pruning, Greedy and PSO algorithms were implemented in the “ToqyzQumalaq” mobile logic game. During the experimental testing of the algorithms the best speed and efficiency was demonstrated by Greedy algorithm, then Minimax, PSO, Alpha-Beta Pruning. This is due to the relatively small depth of game moves, equal to 4 and the precision of the game decision evaluation function. Further research can focus on building a more precision model of the PSO and Alpha-Beta Pruning algorithms evaluation functions.

The project work has made it possible to realize educational goals by gathering knowledge from various fields (algorithmization and programming, artificial intelligence, mathematics, design, modeling, etc.) and facilitated to the formation of key competencies. An active participation of students in teamwork is allowed to form the skills of cooperation, an effective social interaction, a sense of collective responsibility and the ability to work in a team. In addition, the project method facilitates the formation of problem-solving skills. Working in such conditions, students have learned to make a valid conclusion based on the problem analysis, to self-study new knowledge, adapting them to their tasks.

This research can provide the framework for the further implementation of the mathematical apparatus in mobile logic game apps development. The results of the research can also be used in the educational process in the study of subjects related to mobile computing and the mobile game applications development.

References

1. **Stefani, D., Frederikus, J., Lazarusli, I. A., Lukas, S., & Widjaja, P.** (2019). Modelling and implementation of 9tka game with MaxN algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(1), 210-217.

2. **Abdelbar, A. M.** (2012). Alpha-Beta Pruning and Althöfer's Pathology-Free Negamax Algorithm. *Algorithms*, 5(4), 521-528.
3. **Elnaggar, A. A., Gadallah, M., Aziem, M. A., & El-Deeb, H.** (2014). A comparative study of game tree searching methods. *International Journal of Advanced Computer Science and Applications*, 5(5), 68-77.
4. **Putra, W. B., & Heryawan, L.** (2017). Applying alpha-beta algorithm in a chess engine. *Jurnal Teknosains*, 6(1), 37-43.
5. **Poetro, B. S. W., Hidayah, N., & Alifah, S.** (2018). Greedy reduction education game based on Android platform. *Journal of Telematics and Informatics (JTI)*, 6(1), 53-62.
6. **Singh, G., & Deep, K.** (2015). Role of Particle Swarm Optimization in Computer Games. In *Proceedings of Fourth International Conference on Soft Computing for Problem Solving: SocProS 2014, Volume 2* (pp. 259-277). Springer India.
7. **Nurbekova Zh.** *ToqyzQumalaq*, 2019. // <https://apps.apple.com/kz/app/toqyzqumalaq/id1338099217>

Реализация искусственного интеллекта в мобильной логической игре «ToqyzQumalaq»

Ж. Нурбекова^{1*}, Г. Аймичева², Т. Толганбайұлы³, М. М. Галы⁴

¹Казахский национальный педагогический университет имени Абая, Алматы, Республика Казахстан

^{2,3}Евразийский национальный университет им. Л. Гумилева, Астана, Республика Казахстан

⁴Orn, Бангкок, Таиланд



Аннотация. В данной исследовательской работе рассматривается реализация искусственного интеллекта (ИИ) в мобильной логической игре «ToqyzQumalaq» с акцентом на включение передовых алгоритмических стратегий для улучшения игрового процесса. Сложность и стратегическая глубина игры представляют собой уникальные проблемы для разработки ИИ, которые решаются путем интеграции таких алгоритмов, как Minimax, Alpha-Beta Pruning, Greedy и Particle Swarm Optimization (PSO). В исследовании особое внимание уделяется созданию оценочных функций для этих алгоритмов, обеспечивающих эффективность ИИ и принятие решений, подобных человеческим. Этот аспект жизненно важен для сохранения стратегической непредсказуемости, необходимой для «ToqyzQumalaq». Обширное экспериментальное тестирование против человеческих игроков разного уровня мастерства демонстрирует эффективность алгоритмов. Эти тесты выявляют сильные стороны и ограничения каждого алгоритма, давая представление об их применении в игре. Данная статья вносит вклад в развитие ИИ в играх, подчеркивая проблемы и возможности разработки ИИ для сложных игр. Ее результаты актуальны не только для разработчиков игр, но и служат образовательным инструментом, демонстрируя практическое применение ИИ и алгоритмических стратегий.



Ключевые слова: разработка мобильных игр, машинное обучение, Minimax, Alpha-Beta Pruning, Greedy, PSO.


«ToqyzQumalaq» мобилді логикалық ойынында жасанды интеллектіні іске асыру


Ж. Нурбекова^{1*}, Г. Аймичева², Т. Толғанбайұлы³, М. М. Галы⁴

¹Абай атындағы Қазақ ұлттық педагогикалық университеті, Алматы, Қазақстан Республикасы

^{2,3}Л. Гумилев атындағы Еуразия ұлттық университеті,
Астана, Қазақстан Республикасы

⁴Орп компаниясы, Бангкок, Тайланд

 **Аңдатпа.** Бұл мақалада ойын процесін жақсарту үшін озық алгоритмдік стратегияларды енгізуге баса назар аудара отырып, «ТоғызҚумалақ» мобильді логикалық ойынында жасанды интеллектіні (ЖИ) іске асыру мәселесі қарастырылады. Ойынның күрделілігі мен стратегиялық тереңдігі Minimax, Alpha-Beta Pruning, Greedy және Particle Swarm Optimization (PSO) сияқты алгоритмдерді біріктіру арқылы шешілетін ЖИ әзірлеу үшін бірегей мәселелерді ұсынады. Зерттеуде ЖИ тиімділігі және адам сияқты шешім қабылдауды қамтамасыз ететін алгоритмдер үшін бағалау функцияларын жасауға баса назар аударылады. Аталған аспект «ТоғызҚумалақ» үшін қажетті стратегиялық болжамсыздықты сақтау үшін өте маңызды. Шеберлік деңгейі әртүрлі адам ойыншыларына қарсы кең эксперименттік тестілеу алгоритмдердің тиімділігін көрсетеді. Бұл тесттер алгоритмдерді ойында қолдану арқылы түсінік бере отырып, әр алгоритмнің күшті жақтары мен шектеулерін анықтайды. Осы мақала күрделі ойындарға арналған ЖИ әзірлеу мәселелері мен мүмкіндіктерін көрсете отырып, ойындардағы ЖИ дамуына үлесін қосады. Зерттеу нәтижелері ойын жасаушылар үшін ғана өзекті емес, сонымен қатар білім беру құралы ретінде қызмет ететін ЖИ мен алгоритмдік стратегияларды практикада қолданумен де өзекті болып табылады.

 **Түйінді сөздер:** мобильді ойындарды дамыту, машиналық оқыту, Minimax, Alpha-Beta Pruning, Greedy, PSO.

Material received on 29.11.2023