Controlling bot behavior in RPG games using an artificial immune system

Mykola Korablyov[1,†], Oleksandr Fomichov[1,†], Oleksandr Tkachuk[1,*,†] and Igor Kobzev[2,†]

[1] *Kharkiv National University of Radio Electronics, Kharkiv 61166, Ukraine*
[2] *Simon Kuznets Kharkiv National University of Economics, Kharkiv 61166, Ukraine*

Abstract

Today, the development of models for controlling game characters using artificial intelligence methods is important. The work considers the creation of intelligent game bots that simulate the behavior of players in an RPG (Role-Playing Game). A game design for the project has been created, according to which a futuristic world is simulated, where the player controls a group of spaceships that compete with another group of similar spaceships. To describe the capabilities of ships, various attack or recovery options are used, where the main attention is focused on the features of their action from the point of view of game design. For effective control of game bots, a modified immune model of clonal selection, Clonalg-rpg, is proposed, in which bots are considered as antibodies of the system that interact with foreign antigens to the system, which for bots are ships from the project user team. The work of the modified Clonalg-rpg algorithm is described at the level of specific immune operators that perform corresponding actions with the population of antibodies and antigens. To analyze the effectiveness of the Clonalg-rpg immune algorithm, several game sessions were conducted with different compositions of user ship teams and bot teams. The results of experimental studies showed that the proposed Clonalg-rpg immune model is effective to implement, which makes it possible to use it in other game genres.

Keywords

control, game bot, combat, design, spaceship, immune model, clonal selection

1. Introduction

The rapid development and spread of gaming applications have significantly influenced the behavior, skills, and habits of society. This has led to the spread of game mechanics and features in the organization of work of many mobile and web applications, which has been called gamification [1-3]. Elements of gamification can now be observed in many areas of modern life – from food ordering and delivery services and distance learning systems to mobile banking applications [4-6]. Undoubtedly, today the gaming industry directly has a significant impact on the development of society and ceases to play the role of exclusively a means of recreation, but also creates jobs and entire industries.

Mastering the basic techniques is crucial to creating engaging and successful PC games. The foundation of every successful PC game is a well-thought-out game design that includes story, game mechanics, characters, and environments that work together to create an engaging gameplay experience. Game design is considered the most important aspect of computer game development.

Recent years have been characterized by the rapid development of artificial intelligence systems and the spread of free access to such systems by society to solve various every day and creative tasks, particularly in gaming applications [7, 8]. The field of artificial intelligence has ceased to be

Information Technology and Implementation (IT&I-2025), November 20-21, 2025, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
mykola.korablyov@nure.ua (M. Korablyov); oleksandr.fomichov@nure.ua (O. Fomichov); oleksandr.tkachuk@nure.ua (O. Tkachuk); ikobzev12@gmail.com (I. Kobzev)
0009-0005-2540-7741 (M. Korablyov); 0000-0001-9273-9862 (O. Fomichov); 0009-0006-2943-9887 (O. Tkachuk); 0000-0002-7182-5814 (I. Kobzev)

exclusively research and scientific, and has become popular and accessible to everyone, similar to what happened on the computer market in the mid-90s of the last century. Most existing systems of artificial intelligence organization are based on the biological principles of the human nervous or immune systems, as well as the principles of evolutionary research on the genome and behavior of animals and ants. Artificial intelligence has also significantly influenced the development of the modern gaming industry [9, 10].

The work considers the solution to the problem of creating intelligent game bots that simulate the behavior of players in an RPG (Role-Playing Game) game when countering the actions of a team of user characters, using artificial immune systems, in particular, the clonal selection model. Solving this problem will allow the creation of an intelligent system for controlling the behavior of bots in a computer game.

2. Game design of the project

The game application considered in the work simulates a futuristic world in the style of open space, where the player controls a group of spacecraft, the number of which ranges from 1 to 6 units, that compete with another group of similar spacecraft of the same number [11, 12]. The principle of game combat is borrowed from the popular RPG game Disciples II in the mid-2000s, but with a significantly simplified game design. A role-playing game (RPG) is a genre of video games in which the player takes on the role of a character in the game and plays it out, making appropriate choices and decisions that affect the development of the game. The main focus is on exploring the game world, completing tasks, and developing the character through gaining experience and new skills. The game provides for several game factions that can produce their specific types of spacecraft with special capabilities and properties. At the same time, each spacecraft in the user's team or the team of his opponent has a list of specific characteristics that determine its capabilities:

- cost – a characteristic that determines the amount of game money that the user must spend to purchase this ship;
- ship level – a characteristic that is one of the main indicators of a ship, which increases other ship parameters;
- combat experience – a characteristic that determines the experience units gained in battles with enemy ships, which determines the level of the ship;
- durability – a characteristic that determines the viability of a ship in battle with enemy ships;
- energy – a characteristic that determines the attack or defense increase coefficient, depending on the player's choice;
- attack – a characteristic that determines the level of damage that the user's ship inflicts in battle on the selected enemy ship, which directly affects its defense and strength;
- defense - a characteristic that determines armor, or an additional level of defense that reduces the number of attack units that an enemy ship inflicts on this spaceship;
- speed – a characteristic that determines the priority during battle when forming the queue of spaceships' moves;
- recovery – a characteristic that determines the number of ship strength units that a ship can recover instead of performing an attack on an enemy ship.

It should be noted that the game provides for several factions that create their types of spaceships and give them specific properties. For ease of implementation, the factions are conventionally designated by colors, namely: yellow, blue, and red. The main feature of the ships of the yellow faction is an increase in the strength indicator by 10%, compared to ships of other factions; the feature of the ships of the blue faction is an increase in the protection indicator by 10%, compared to ships of other factions; and the main feature of the ships of the red faction is an increase in the recovery

indicator by 10%, compared to ships of other factions. Thanks to this, each faction can increase the significant characteristics of its ships uniquely compared to ships of other factions. In addition, each ship, both from the user's team and from the opponent's team, can belong to one of the ranks. These ranks significantly affect the cost and parameters of this ship. The list of main ranks is defined as follows:

- support – a support ship, ranks first in the group of ships, has one melee attack method and three recovery methods;
- scout – a reconnaissance ship, ranks first in the group of ships, has one method of melee attack and one method of ranged attack, and also has one method of recovery, stands out among other ships for its speed;
- cruiser – a basic battleship, ranks first in the group of ships, has two methods of close and one method of long-range attack, and also has one method of recovery;
- frigate – reinforced basic combat ship, takes the first place in the group of ships, has two methods of close and two methods of long-range attack, has one method of recovery, differs from other ships in enhanced defense;
- destroyer – a heavy battleship, ranks second in the group of ships, has two methods of close and two methods of long-range attack, has two methods of recovery, and stands out among other ships with its enhanced durability;
- dreadnought – an extremely heavy battleship, ranks second in the group of ships, has three methods of close and three methods of long-range attack, has three methods of recovery, differs from other ships in enhanced strength and protection indicators, as well as a reduced speed indicator.

To describe the capabilities of ships, various attack or recovery options were used, where attention was focused on the features of their action from the point of view of game design. First, two types of attack were used in the description of the capabilities of ships: close and long-range attacks. In this case, a close attack implies the ability of the ship to attack only those enemy ships that are on the first line of the battlefield in front of it, i.e. on positions 2, 4 and 6 of the game battle map (Fig. 1). A long-range attack implies the ability to attack enemy ships that are on the second line in the enemy team, i.e. on positions 1, 3 and 5 of the game battle map.

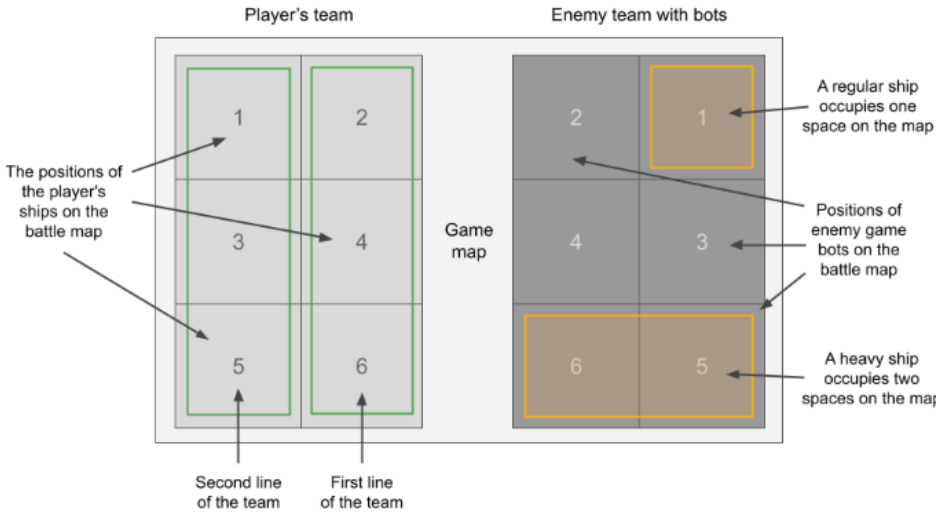

Figure 1: General view of the game battle map

It should be noted that in the description of the capabilities of the specified ships, three attack methods were used, with the first method involving a concentrated attack on one selected enemy ship, the second method involving a distributed attack on all enemy ships located on the game battle map, and the third method involving a reduction in the energy indicator of the selected enemy ship

by 30% of the current value. In addition, when describing the capabilities of game ships, two methods of recovery were defined, namely - the first method of recovery is that instead of attacking, the ship begins to repair itself, in the process of which it restores a certain specified number of strength units that it lost after the attack of the ships of the enemy team. The second method of recovery involves repairing one of the selected existing ships in its group, transferring a specified number of strength units to it. The third method of recovery involves the ship repairing all existing ships in its group, distributing a certain number of strength units equally among them. It should be noted that in the event of complete exhaustion of strength units during a game battle, the ship ceases to exist, and it is not possible to restore it in battle by performing the second or third methods of recovery. The recovery of such a ship occurs automatically after the end of the battle.

The way of attacking ships has a significant impact on the gameplay [13, 14]. Accordingly, and taking into account the defined types of close and long-range attacks, the appearance of the playing field for the battle between ship teams should be described. The entire playing field is conventionally divided into 3 parts, where the first part is occupied by the first player's ship team, the second part divides the teams among themselves, and the third part is occupied by the second player's team. It should be noted that in each team, ships are lined up in two rows - the one closest to the enemy and the one farthest from the enemy, and only ships that occupy two places on the game battle map occupy both rows, that is, they can be attacked by both close and long-range attacks. In addition, if all ships in the first line are destroyed, all ships in the second line can be attacked by any method of attack.

In the game, a great influence on the capabilities of a ship is its level, namely, it increases the strength, defense, and attack indicators by 1% of the initial values of a ship of a certain type of the first level, taking into account the characteristics of the faction. The level increase occurs due to the victory over the enemy team, where for every 100 units of strength of each destroyed ship of the enemy team, 10 units of experience are accrued, which are evenly distributed between all ships that participated in the battle. In addition, the method of assigning damage from attacks between ships, which is determined by the expression:

$$damage = \frac{e_1}{e_2} \times (A_1 - D_2), \quad (1)$$

where $damage$ – the value of the adjusted damage that a ship inflicts on another during an attack; A_1 – ship attack value 1; D_2 – ship defense value 2; e_1 – ship energy value 1; e_2 – ship energy value 2.

Accordingly, the strength indicator of the ship being attacked is reduced by the *damage* value if it has a positive value. If it has a negative value, the strength indicator of the ship being attacked does not change, i.e., the ship does not receive any damage at all.

An important influence on the gameplay is the way a game ship can be restored, which can be done during its turn instead of performing an attack. This indicator is determined as follows:

$$rp = R \times ke, \quad (2)$$

where rp - the value of the ship's restored strength; R – ship recovery characteristics; k - recovery factor, which is chosen randomly from the range [1.1; 1.25]; e – ship energy value.

An important feature of the gameplay is the ability to play in Player via Enemy (PvE) mode, i.e., a game where the user plays against a team controlled by the game itself.

3. Controlling game bots based on the clonal selection model

The task of controlling game bots in a battle with a team of user ships is to simulate the behavior of each ship of the enemy team, taking into account the capabilities and characteristics of each specific ship of the player's team. For effective control of game bots, it is currently advisable to use artificial intelligence systems, among which, along with neural networks, fuzzy logic, genetic algorithms, etc.,

the theory of artificial immune systems (AIS) occupies an important place. The most common AIS models used to solve various practical problems are the immune model of clonal selection, *ClonAlg,* and the artificial immune network model *AINet* [15, 16].

Controlling game bots can be considered an optimization problem, for which the immune model of clonal selection can be used [17, 18]. Bots are considered the antibodies of the system, interacting with foreign antigens, which, for bots, are ships from the project user team. Bots can interact with each other, executing the recovery command for themselves and the bots of their team.

The *ClonAlg* method is one of the most common methods that operates on the basis of the clonal selection model. This method can be used to solve the problem of controlling game bots that oppose the user's ship team, which controls them personally. The classic version of the *ClonAlg* immune algorithm provides for the possibility of changing antibody parameters during their interaction with a population of foreign antigens by cloning, mutation, and self-regulation of the antibody population through clonal selection and aging. In this case, changes in antigen parameters were either not foreseen at all or were insignificant. In addition, the possibility of antigen disappearance in the process of forming an immune response by the system was not foreseen as such. But in our case, during the confrontation of the user's ship team with the system's game bot team, such a possibility is foreseen.

With this method of operation of the immune algorithm, there is a destruction of all antigens that represent the user's ship team. Therefore, the original version of the immune method ClonAlg cannot be used to solve this problem without a number of changes. The proposed modified version of the Clonalg-rpg algorithm has several important features. First, the algorithm cannot directly change the values of the antibody features (set of game bots) in the process of its operation, but can only make decisions regarding the appropriateness of executing a particular command for each specific bot or its clone. Secondly, the choice of a solution for executing a particular bot command should be made from the point of view of the maximum efficiency of this action of destroying the entire population of antigens, or from the point of view of restoring the strength indicator of the entire population of antibodies, or a specific individual bot of it. Thirdly, the condition for terminating the operation of the modified immune algorithm *Clonalg-rpg* is the destruction of the entire population of antigens, represented by the set of ships of the game user, before the population of antibodies, represented by game bots, ceases to exist. Fourth, dynamic changes in the features of antibody bots and antigen ships occur constantly, after each action of the antigens and antibodies of the system.

Thus, the implementation of the modified version of the immune algorithm *Clonalg-rpg* should be based on four defined principles as the basis of the algorithm's work. In addition, an important feature of the clonal selection model is that in the process of work, antibodies focus primarily on interaction with antigens, and then use some opportunities for interaction with other antibodies.

To simplify the implementation possibilities of immune methods and algorithms, they are usually divided into specific immune operators that perform one or another action with a population of antibodies or antigens. The order of these operators in the algorithm cannot be changed during its operation. The immune operator concentrates on the implementation of one specific function, By the nature of their action and the possibilities of setting and configuration, immune operators are usually divided into universal and specific. At the same time, universal operators can be used in various immune methods because they perform actions that are relevant for different immune models. Among the universal immune operators, cloning, mutation, antigen population presentation operators, etc., are noted. Specific immune operators are operators that perform a function that is unique to a specific model, such as a clonal selection model or a negative selection model, etc. Specific immune operators include operators of clone selection, suppression, apoptosis, positive selection, etc.

The operation of the modified *Clonalg-rpg* method, created to solve the problem of controlling game bots, can be conditionally described at the level of immune operators as follows:

$$lonalg - rpg = \begin{bmatrix} Presentation(AB, AG) \rightarrow \\ Cloning(AB, CL) \rightarrow \\ Mutation(AB, CL) \rightarrow \\ ClonSelection(AG, CL) \rightarrow \\ SelfRegulation(AB, CL) \rightarrow \\ Termination(AB, AG) \end{bmatrix}^{AG>0}, \quad (3)$$

where *Presentation (AB, AG)* is the operator for presenting the set of antigens AG of the antibody population AB; *Cloning (AB, CL)* is the operator for cloning the antibody population; *Mutation (AB, CL)* is the operator for mutation of the formed clones; *ClonSelection (AG, CL)* is the operator for selecting clones; *SelfRegulation (AB, CL)* is the operator for self-regulation of the antibody population; *Termination (AB, AG)* is the operator for checking the possibility of stopping the algorithm.

The work of the *Presentation (AB, AG)* operator is to determine the affinities between antibodies and antigens, as well as the affinities between the population of antibodies of the IAS. In this case, affinity is defined as the degree of similarity of the features of the antibody and the antigen, and the work of the immune algorithm is reduced to ensuring that in the process of cloning and mutation, the clones-descendants of a particular antibody, reproduce the features of a particular antigen. Such a targeted action of the system is possible when solving the problem of classification, clustering, pattern recognition, prediction, etc., but is impractical when solving the problem of controlling game bots in an RPG game. Since increasing the affinity level between antibodies and antigens is one of the target tasks of the AIS, taking into account the specificity of the task, the affinity should reflect how quickly the AIS destroys the external antigen in the process of its work, and not recognizes and classifies it. Accordingly, when determining the *Clonalg-rpg* affinities, the current state of antigens with their apparent optimal state will be used, which is characterized by the following features: a zero value of the strength parameter and a minimum value of the energy parameter of the enemy ship (antigen). Thus, only two features of game objects out of the nine described earlier will be used in determining the affinity, which can significantly speed up the work of the immune algorithm by reducing the number of computational operations. Taking this into account, as well as the requirements for the range of affinity values, the determination of antigen affinity is described by the expression:

$$affAg_i = \frac{1}{1+(Ds_i - Dc_i)+(Es_i - Ec_i)}, \quad (4)$$

where $affAg_i$ is the affinity of the *i*-th antigen; Ds_i is the saved strength indicator of the *i*-th antigen at the start of the battle; Dc_i is the current strength indicator of the *i*-th antigen at the current battle iteration; Es_i is the saved energy indicator of the *i*-th antigen at the start of the battle; Ec_i is the current energy indicator of the i-th antigen at the current battle iteration.

The antibody cloning operator *Cloning (AB, CL)* is used to create a set of clones for each antibody in the current AIS population for further mutation. It should be noted that in different AIS models, the most common use is the static cloning operator or the proportional cloning operator. For the case of the *Clonalg-rpg* algorithm, the use of the proportional cloning operator is inappropriate due to the limited number of mutation options, as well as the use of static mutation for all antibodies, regardless of their features and characteristics. Therefore, the cloning operator in *Clonalg-rpg* will also be specific, namely, adaptive, that is, the number of clones for each antibody will be determined based on its characteristics and features. At the same time, the number of attack options of each ship (bot) will have a decisive influence on the number of clones created during the cloning process. Accordingly, the number of clones created through the use of the adaptive cloning operator is determined as follows:

$$n_i = C_i \times N_{AG}, \quad (5)$$

where n_i is the number of clones of the i-th antibody; C_i is the number of interaction variants between this antibody and any antigen; N_{AG} is the number of active antigens.

The clone mutation operator, *Mutation (CL),* significantly affects the speed of solving the target problem set for the AIS. This is because the mutation mechanism provides for changes in immune objects and brings the AIS closer to solving its problem. According to the principle of their operation, several of the most common mutation operators are distinguished: static, proportional, and inversely proportional. Given the specifics of the task of controlling game bots in an RPG game, the use of any of the specified types of mutation operators is impossible without significant modifications. Therefore, the *Clonalg-rpg* algorithm provides for the use of the adaptive mutation operator as a modification of the static mutation operator with the possibility of changing the object of interaction and the nature of the action. A feature of the adaptive mutation operator is that it does not directly change the parameters of antibody clones, but focuses on choosing a method of interaction with antigens. Accordingly, during mutation, a separate specific method of interaction with each active antigen is selected for each clone. In this case, an active antigen in the context of the task of controlling game bots in an RPG game is understood as a separate ship from the user's team of spaceships, and a specific method of interaction is understood as a particular method of attack for each ship of the user's team, or a method of attack for the entire team of ships by a particular bot.

The clone selection operator *ClonSelection (AG, CL)* plays a very important role in the work of the immune algorithm *Clonalg-rpg*, based on the principles of clonal selection. It is thanks to it that one immune object is selected from the list of mutated clones, or one way of interaction with a particular active antigen, or the entire set of antigens, which makes the greatest contribution to solving the target problem. Accordingly, in the process of clonal selection, the clone whose action led to the maximization of the affinity of a separate antigen, i.e., its transfer to an inactive state, or the maximization of the average affinity in the population of antigens, is selected. These conditions can be achieved by the clone by performing its specific action, which leads either to minimizing the strength value in a particular antigen or to a significant decrease in the strength in the entire set of active antigens. Thus, the work of the clonal selection operator in the Clonalg-rpg algorithm corresponds to the principles of the elite clonal selection operator, which uses the maximum affinity indicators with antigens and is used in such algorithms as ClonAlg and BCA [15].

The antibody population self-regulation operator *SelfRegulation (AB, CL)* is not specific to the ClonAlg algorithm, and was added to the *Clonalg-rpg* algorithm instead of the apoptosis operator, whose action was aimed at reducing the population of immune objects by replacing cloned antibodies with clones selected as a result of clonal selection. It should be noted that due to the peculiarities of the organization of the apoptosis operator implemented in ClonAlg, the number of operations for calculating affinities between antibodies and clones is minimized, but such operations are not completely excluded. In the *Clonalg-rpg* algorithm, instead of this operator, the antibody population self-regulation operator is used, which involves determining affinities or determining some other metrics between the antibody population and the clones selected in the clonal selection process. Accordingly, an affinity is determined between each clone, which determines the level of interaction with other active antibodies of the AIS. It should be noted that active antibodies are antibodies that are associated with a particular ship from the set of game bots controlled by the AIS.

At the same time, if a ship, which is a game bot, completely loses all units of the strength indicator during a game battle, it becomes inactive, does not participate in the game process, and cannot be restored by other game bots. Such a game bot ceases to be considered an antibody for the AIS, and accordingly, it is no longer subject to the actions of cloning, mutation, etc. operators, and is conditionally considered a memory cell that does not participate in further AIS processes. The main task of the immune operator of self-regulation is to determine clones that can replace active antibodies and transfer the AIS and the antibody population to a new state that brings the implementation of the immune algorithm task closer. It should be noted that after the selection of clones from the entire set of clones created from each active antibody, one is selected that is characterized by the maximum value of affinity for antigens. After that, in the process of self-

regulation of the AIS, a decision is made for this clone to determine the possible action with active antibodies. Such actions can be either calling the recovery command for the entire set of game bots, or calling the recovery command for one specific bot that is associated with this clone. The possibility of such actions is determined after calculating the internal affinities between the clone and active antibodies. Internal affinities are determined in several ways, depending on the method of interaction: interaction with a single antibody or interaction with the entire population. Accordingly, the first option is interaction with a single antibody from which this clone was created, in which case the expression for determining the affinity will have the following form:

$$aff^{'}Ab_i = \frac{1}{1+|Ds_i-(Dc_i+R_i)|}, \quad (6)$$

where $aff^{'}Ab_i$ is the affinity of the clone with the i-th antibody from which this clone was created; Ds_i is the saved strength indicator of the i-th antibody at the start of the battle; Dc_i is the current strength indicator of the i-th antibody at the current iteration of the battle; R_i is the recovery indicator of the i-th antibody.

The definition of the intrinsic affinity index of a clone to the entire antibody population is based on (6) and generalizes this expression as follows:

$$aff^{''}Ab_i = \frac{1}{S}\sum_{i=1}^{S}(1 + \left|Ds_i - \left(Dc_i + \frac{1}{S}R\right)\right|)^{-1}, \quad (7)$$

where $aff^{''}Ab_i$ is the affinity of the clone with the entire population of active antibodies, including the i-th antibody from which this clone was created; S is the size of the population of active antibodies; Dc_i is the current strength indicator of the i-th antibody at the current iteration of the battle; R is the clone recovery indicator.

The possible action method for restoring the strength characteristic for one antibody, or the entire population of antibodies, is selected according to the value of the higher affinity. That is, if the value $aff^{'}Ab_i$ is greater than the affinity value $aff^{''}Ab_i$, the first recovery method is selected. Otherwise, the second recovery method is selected, which the game bot can perform during its turn during the game. It should be noted that if the game bot has only one recovery method, the affinity between it and the active antibodies is determined only once.

At the final stage of the operation of the AIS self-regulation operator in the *Clonalg-rpg* algorithm, a final decision is made for each clone regarding the choice of its method of action – attack of a particular adversary, or group of adversaries, represented by a set of active antigens, or restoration of one or a group of bots, represented by active AIS antibodies. The final decision regarding the choice of action is made after comparing the affinities that characterize the interaction of this clone with a set of active antigens and interaction with a set of active antibodies. If the affinity, which characterizes the interaction with antibodies, is greater than the affinity, which describes the interaction with antigens, for the bot associated with this clone, a decision is made to restore antibodies. Otherwise, the game bot will attack the selected ship of the player's team, or all of its ships, depending on the type of attack. After this selection, the game bot implements the selected method of action, changing the characteristics of antigens or antibodies. It should be noted that since changes occur only in the characteristics of antigens and antibodies, and the clone symbolizes one or another possible method of action, further preservation of clones for the formation of immune memory is impractical. Therefore, after performing the selected action, the clone is removed from the system objects.

The operator for checking the possibility of stopping the battle *Termination(AB, AG)* is used to check the possibility of stopping the game battle process. The most common types of operators for terminating the AIS are static, criterion, full, and combined. The *Clonalg-rpg* algorithm uses the criterion operator for terminating the AIS, which involves stopping the work of immune operators in the absence of active antigens or active antibodies. Accordingly, if there is at least one active

antibody in the system, or the AIS interacts with at least one active antigen, the system will continue its work.

4. Experimental studies

To analyze the effectiveness of the *Clonalg-rpg* immune method, several game sessions were conducted with different compositions of the user's ship teams and bot teams. The choice of faction had a certain influence on the results of the battle, which led to the use of bonuses in different parameters of the ships. For the user to be able to select more than one type of spaceships, which are characterized by large values in the main characteristics, the user had a limitation in resources, namely in the funds that he could spend on forming his team with full access to all types of ships from each faction. Accordingly, the user is allowed to combine ships of different factions in his team. It should be noted that the teams of game bots can be generated randomly without any cost restrictions or loaded from a specific file with battle-specific settings. In the case of generating bot teams, both very strong, expensive teams and teams with low cost and capabilities are randomly formed. In addition, the number of ways to attack ships and the number of ways to recover had a significant impact on the course of the game battle, as well as on the speed of decision-making by the *Clonalg-rpg* algorithm.

When studying the effectiveness of the *Clonalg-rpg* algorithm, about a hundred tests were conducted with different compositions of user teams and teams of game bots of different types and different factions. During these tests, teams were formed not randomly, but through specific settings, taking into account that spaceships can be divided into three classes according to their parameters and capabilities, where the first class is formed by the cheapest *support* and *scout* ships, the second class is formed by *cruiser* and *frigate* ships, and the third most expensive class is formed by heavy ships, which occupy two places on the game battle map – *destroyer* and *dreadnought*. At the same time, the parameters of these ships, such as attack power, energy, durability, etc, in the game balance are adjusted in such a way that first-class ships are inferior to second-class ships by approximately 25%, and to third-class ships by approximately 50%. Accordingly, second-class ships are inferior to third-class ships by 25% in their basic indicators. However, for the sake of game balance, first-class ships outperform all other ships in terms of speed and make their move during a game battle first.

The *Clonalg-rpg* algorithm was tested on the same set of input data about the user's ship commands and game bots in three ways: without using the AIS, when the user controls both commands directly; without using the AIS by choosing the game bot's action method randomly; and with using the *Clonalg-rpg* algorithm, which controls the behavior of game bots in battle. The main indicator of the effectiveness of the immune algorithm for controlling game bots is the number of victories of the AIS over a person in game battles in which teams of ships of comparable or equal in their basic parameters take part. Took into account not only the percentage of victories of the AIS over the user, but also the number of ships remaining in the winning team. Accordingly, the following indicators were selected:

- the winner has 1 ship left;
- the winner has 2 ships left;
- the winner has 3 or more ships left.

Using such detail will allow us to investigate not only the fact of victory itself, but also how much the winner dominated his opponent during the game battle, which may indicate the level of skill of the game. It should be noted that most of the tests were conducted on two types of teams: a team consisting of 5 ships from the user and 5 ships from the opponent, as well as a team consisting of 4 ships from the user and 4 ships from the opponent. In this case, the ships were chosen to be the same in all teams in order to avoid possible advantages in the parameters of different ships from different factions. In addition, the composition of the team itself plays an important role, namely, in 5 v 5

teams the following list of ships is used: *support* and *scout* in positions away from the enemy team, *cruiser*, *frigate* and *destroyer* form the first line of resistance, taking into account that the *destroyer* occupies two places on the game battlefield. In the case of using a 4 v 4 team, two *destroyer* ships were used, one *support* in the second line from the enemy team, and one *frigate* in the first line from the enemy team. Table 1 shows the percentage of victories of the user's teams over the teams of his opponent, taking into account the different methods of controlling the enemy team.

Table 1
Frequency of user wins over opponent

Types of commands	Manual control	Random control	Clonalg-rpg control
5 v 5	50 %	85,15 %	50,35 %
4 v 4	50 %	96,75 %	49,15 %

It should be noted that for each type of control of the team of game bots, 20 game battles were conducted to obtain statistical data on the results of battles and the choice of the type of actions of bots during the game battle. According to the results presented above, the least effective is the control of game bots made randomly, which almost always led to a guaranteed defeat. In contrast, the organization of game bot control based on the *Clonalg-rpg* algorithm showed results comparable to those obtained by involving a person to control the enemy ships. Also, from the results obtained during the testing of the immune algorithm, it is clear that when the number of ships in the team is reduced by using heavy ships, which have a greater variability of possible actions, the chance of victory of game bots increases slightly, which may indicate that in the case of an increase in the number of attack and recovery options, the proposed immune method *Clonalg-rpg* will show greater efficiency than the user. To assess the skill of the winning team, the number of ships remaining in the winning team after the battle was recorded. These results are given in Table 2.

It can be concluded that using the method of controlling game bots randomly creates the easiest conditions for the user, because in most cases it was the user who won victories over a team of bots, which is controlled in a random decision-making method, while the user's team almost always has 2 ships left, even in the case of using teams of 4 ships. In contrast, according to the results obtained, it can be concluded that using the *Clonalg-rpg* algorithm creates difficult conditions for the user to win over a team of game bots, while the winner's team usually has only one ship left, while the others are lost during the game battle.

Table 2
Number of ships remaining in the winning team

Types of commands		Manual control	Random control	Clonalg-rpg control
5 v 5	1	100,00 %	12,40 %	93,75 %
	2	0,00 %	82,35 %	6,25 %
	3+	0,00 %	5,25 %	0,00 %
4 v 4	1	100,00 %	3,60 %	95,00 %
	2	0,00 %	88,25 %	5,00 %
	3+	0,00 %	8,15 %	0,00 %

A separate direction of observation was the type of action that the game bot chooses and performs during its turn. This choice plays an important role, since it is he who leads the team of game bots to victory over the user's team or defeat. When allowed to make your move in the order of the queue, which is formed based on the speed of all active ships from both teams participating in the game battle, you can choose either to attack the enemy or to recover. It is impossible for either the game bot or the ship that is under the direct control of the user to miss a move. Table 3 shows data on the

frequency of choosing a particular method of action by game bots that are under different types of control.

Table 3
Frequency of choice of action methods in 5v5 team battles

Variants of ship actions	Manual control		Random control		Clonalg-rpg control	
	5 v 5	4 v 4	5 v 5	4 v 4	5 v 5	4 v 4
I melee attack method	19,33 %	9,82 %	24,56 %	18,89 %	20,29 %	11,01 %
II melee attack method	9,83 %	10,42 %	11,22 %	10,56 %	10,86 %	8,57 %
III melee attack method	3,15 %	7,89 %	2,22 %	5,56 %	3,90 %	9,76 %
I method of long-range attack	22,00 %	6,05 %	17,89 %	10,56 %	20,00 %	6,25 %
II method of long-range attack	5,83 %	10,62 %	6,22 %	10,56 %	4,86 %	8,57 %
III method of long-range attack	3,15 %	7.89 %	2,22 %	5,56 %	3,90 %	9,76 %
I method of recovery	19,87 %	18,65 %	24,56 %	18,89 %	19,67 %	18,75 %
II method of recovery	14,02 %	22,42 %	8,89 %	13,89 %	14,52 %	22,32 %
III method of recovery	2,50 %	6,25 %	2,22 %	5,56 %	2,00 %	5,00 %

Based on the results of the study of decisions made by game bots according to different control methods, it can be concluded that the immune algorithm *Clonalg-rpg* gives a greater advantage to the III method of attack, both at close and long distances, minimizing the use of the II method of distributed attack of the enemy team's ships. Accordingly, in the case of a game bot's decision to attack the enemy under the control of *Clonalg-rpg*, the most attention is paid to the methods of concentrated attack of a specific ship, while, if possible, inflicting additional damage to the energy of the ship from the enemy team, which significantly affects its further capabilities in battle. In the case of recovery, the method of concentrated recovery of one selected ship is also preferred. This behavior can be described as a strategy of concentrated destruction of the weakest ships from the enemy team while maximally concentrated recovery of damaged ships of one's team.

The proposed *Clonalg-rpg* method is used to control game bots in RPG projects for the first time, so today it is quite difficult to make a comparison with other immune and non-immune methods that are used to organize a game bot control system. It is also important to note that the proposed immune method is easy to implement and modify, which makes it possible to further use it to control game bots in games of other genres.

5. Conclusions

The solution to the current problem of controlling game bots in RPG games using a modified immune model of clonal selection is considered. A game application has been developed that simulates a futuristic world in the style of open space, where the player controls a group of spacecraft competing with another such group of similar spacecraft. At the same time, each spacecraft in the user's team or in the team of his opponent has a list of specific characteristics that determine its capabilities. The game features several factions that create their types of spaceships and give them specific properties. Thanks to this, each faction can increase the significant characteristics of its ships uniquely compared to the ships of other factions. In addition, each ship, both from the user's team and from the team of its opponent, can belong to one of the ranks, which significantly affect the cost and parameters of this ship. To describe the capabilities of the ships, various attack and recovery options were employed, focusing on the features of their actions from the perspective of game design.

Game bot control is considered a special case of the optimization problem, for the solution of which the modified immune model of clonal selection *Clonalg-rpg* is used, according to which bots are considered as antibodies of the system that interact with foreign antigens to the system, which for bots are ships from the project user team. The work of the modified algorithm *Clonalg-rpg* is described at the level of immune operators that perform the corresponding actions with the population of antibodies and antigens representing the bots of the system and the user.

During the experimental study of the *Clonalg-rpg* algorithm's effectiveness, approximately 100 tests were conducted with varying user team compositions and game bot team types and factions. During these tests, the teams were formed not randomly, but through specific settings, taking into account that the spaceships were divided into three classes according to their parameters and capabilities. The results of the experimental studies showed that the proposed immune model, Clonalg-rpg, is effective and simple to implement, which makes it possible to use it in other game genres.

6. Declaration on Generative AI

During the preparation of this work, the authors used Grammarly to check grammar and spelling, paraphrase and reformulate. After using this tool/service, the authors checked and edited the content as needed and take full responsibility for the content of the publication.

References

[1] R. Panchanadikar, G. Freeman, L. Li, K. Schulenberg, & Y. Hu. A New Golden Era' or 'Slap Comps: How Non-Profit Driven Indie Game Developers Perceive the Emerging Role of Generative AI in Game Development." Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM (2024). 1–7. https://doi.org/10.1145/3613905.3650845.

[2] J.P. Lankoski, S. Björk. Game Research Methods. An Overview. Pittsburgh, PA: ETC Press (2015). https://doi.org/10.25969/mediarep/13581.

[3] J. Immonen. Primary Tools and Techniques in Game Development. Bachelor's thesis. Lappeenranta-Lahti University of Technology LUT (2024) 43 pages. https://lutpub.lut.fi/bitstream/handle/10024/168411/.

[4] K. Szolin, D.J. Kuss, F.M. Nuyens, M.D. Griffiths. "I am the character; the character is me": A thematic analysis of the user-avatar relationship in video games. Computers in Human Behavior, Vol. 143 (2023) 107694. https://doi.org/10.1016/j.chb.2023.107694.

[5] M. Barr. Video games can develop graduate skills in higher education students: a randomized trial. Computers in Education, 113 (2017) 86-97. https://doi.org/10.1016/j.compedu.2017.05.016.

[6] W.R. Boot. Video games as tools to achieve insight into cognitive processes. Frontier in Psychology, 6 (2015) 1-3. https://doi.org/10.3389/fpsyg.2015.00003.

[7] M.A. Quiroga, A. Diaz, F.J. Román, J. Privado, R. Colom. Intelligence and video games: Beyond "brain-games". Intelligence, Volume 75, (2019) 85-94. URL: https://www.elsevier.com/locate/intell.

[8] A. Simons, I. Wohlgenannt, S. Zelt, M. Weinmann, J. Schneider and J. vom Brocke. Intelligence at play: game-based assessment using a virtual-reality application. Springer (2023). https://doi.org/10.1007/s10055-023-00752-9.

[9] Y. Lum, W. Li. Techniques and Paradigms in Modern Game AI Systems. Algorithms, 15(8*)* (2022) 282. https://doi.org/10.3390/a15080282.

[10] X. Fan, J. Wu, L. Tian. A Review of Artificial Intelligence for Games. Part of the Lecture Notes in Electrical Engineering book series (LNEE, vol. 572) (2020) 1821. URL: https://doi.org/10.1007/978-981-15-0187-6_34.

[11] R. Snyder. 10 Best Spaceships in Video Games (2023). URL: https://www.dualshockers.com/best-spaceships-in-video-games/.
[12] USgamer Team. What's The Best Video Game Spaceship? (2019). URL: https://www.vg247.com/whats-the-best-video-game-spaceship.
[13] T.K. Mohd, F. Bravo-Garcia, L. Love, M. Gujadhur, J. Nyadu. Analyzing strengths and weaknesses of Modern Game Engines. International Journal of Computer Theory and Engineering, 15(1) (2023) 54-60. https://www.ijcte.org/vol15/IJCTE-V15N1-1330.pdf.
[14] M. Silić, B. Džaja, R. Rogulj, H. Turić. Optimizing Game Ready Asset Creation Pipeline: From Concept to Implementation in Unreal Engine 5. ORGANIZER, (2024) 732. https://hdl.handle.net/11159/654467.
[15] D. Dasgupta, S. Yu, L.F. Nino. Recent Advances in Artificial Immune Systems: Models and Applications. Applied Soft Computing. Elsevier (2011) 1574-1587. URL: https://https://doi.org/10.1016/j.asoc.2010.08.024.
[16] G. Samigulina, Z. Samigulina. Biologically Inspired Unified Artificial Immune System for Industrial Equipment Diagnostic. International Conference on Machine Learning, Optimization, and Data Science. Lecture Notes in Computer Science book series LNCS, vol. 13811 (2023) 77-92. https://www.researchgate.net/publication/369128674.
[17] M. Korablyov, O. Fomichov, O. Chubukin, D. Antonov, S. Dykyi. Immune Model for Controlling Characters in Computer Games. XI International Scientific and Practical Conference "Information Control Systems and Technologies" (2023) 214-226. https://ceur-ws.org/Vol-3513/.
[18] S. Shekhar, D.K. Sharma, D.K. Agarwal, Y. Pathak. Artificial Immune Systems-Based Classification Model for Code-Mixed Social Media Data. IRBM, Vol. 43, Iss. 2, (2022) 120-129. URL: https://doi:10.1016/j.irbm.2020.07.004.