Immune Model for Controlling Characters in Computer Games

Mykola Korablyov[a], Oleksandr Fomichov[a], Oleksandr Chubukin[a], Danylo Antonov[a], Stanislav Dykyi[a]

[a] *Kharkiv National University of Radio Electronics, Kharkiv 61166, Ukraine*

Abstract

The rapid development of information technologies led to the emergence of new opportunities for game software developers. In the field of game application development, a large number of platforms and patterns of architecture development, as well as game environments, have appeared, which allow simplifying and automating the process of development and deployment of the game software. The game application used in the work simulates a futuristic space world where the player interacts with other characters in a free game space. An analysis of game resources, which can be controlled by the player and other game characters, has been carried out. The main features of the behavior of game characters controlling spaceships on the map of the game world are defined. The task of controlling the behavior of characters in game software can be considered as a classification problem, for which it is proposed to use artificial immune systems. Among the existing immune models, the most promising for practical application is the artificial immune network model, which was chosen as the basis for creating a model for controlling the behavior of characters in computer games. But it cannot be used to solve the problem of managing game characters without additional modifications. A modified model of the immune network – behavioral aiNET (baiNET) was proposed, which can be used to solve the problem under consideration. The software implementation of the game software tool, in which the control of the game characters is carried out using the baiNET model, has been completed. Experimental studies were carried out with different numbers of game characters on different-sized maps of the game world, which showed that the proposed baiNet immune model is effective and simple to implement and modify. This makes it possible to use it to control characters in games of other genres.

Keywords

Computer Game, Spaceship, Game Character, Immune Model, Game Application, Game Resources, Behavior Artificial Immune Network, Software

1. Introduction

Today, the development of game applications has a significant impact on the behavior and habits of society. A big role in this is played by multi-user games, around which social connections, fan groups, and business projects are formed [1, 2]. The COVID-19 pandemic acted as a catalyst for the growth of the popularity of game applications in the world [3]. Therefore, the gaming industry is developing very rapidly. Now popular games can be played on personal computers, on social networks, and on game consoles, as well as on mobile devices - phones and tablets.

The rapid development of information technologies led to the emergence of new opportunities for developers and publishers of game software, as well as simplification and improvement of already existing technologies [4, 5]. Thanks to this, in the field of game application development, a large number of platforms, game application architecture development patterns, as well as game engines -

ICST-2023: Information Control Systems & Technologies, September 21-23, 2023, Odesa, Ukraine
EMAIL: mykola.korablyov@nure.ua (M. Korablyov); oleksandr.fomichov@nurel.ua (O. Fomichov); oleksandr.chubukin@nure.ua (O. Chubukin); danylo.antonov@nure.ua (D. Antonov); stanislav.dykyi@nure.ua (S. Dykyi)
ORCID: 0000-0002-8931-4350 (M. Korablyov); 0000-0001-9273-9862 (O. Fomichov); 0000-0002-2410-4563 (O. Chubukin); 0009-0000-2079-3413 (D. Antonov); 0009-0007-5396-2413 (S. Dykyi)
© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3513/paper18.pdf

environments that allow simplifying and automating the process of development and deployment of game software appeared.

Usually, game software tools differ in their genre, as well as in the number of players. It should be noted that it is difficult to unambiguously determine the criteria for a game application to belong to one or another genre due to the fact that today there is no final and universal systematization of the classification of games, and data on the genre of a particular project may differ in different sources [6, 7]. However, there is a consensus gradually reached by corporations involved in the development of game applications, thanks to which the game's belonging to one or another genre can be determined unambiguously.

Artificial intelligence systems, which allow solving complex practical problems related to intelligent data processing and analysis under conditions of uncertainty, are now gaining great popularity. The design and development of artificial intelligence systems significantly affect the game software market [8, 9]. Artificial intelligence (AI) is widely used in the gaming industry (games). AI in games is usually used to create the player's opponents, which is the basis of all video games. AI-based games are based on a finite set of actions or reactions, the sequence of which can be easily predicted by experienced players. It is important to create an intelligent control model of game characters that are not under the direct control of users, to simulate the behavior of players of many user computer games and oppose or cooperate with the user.

The aim of the work is to develop a model of controlling characters in a computer game using the immune approach. In accordance with the specified goal, there is a need to solve the following tasks:

1. Analysis of the task and search for the necessary information.
2. Design and implementation of an artificial immune system (AIS) model to make a decision about the character's behavior in the game.
3. Setting up the immune operators of the model to optimize the process of managing game characters.
4. Design and implementation of a game application based on one of the modern architectural development templates.
5. Adaptation of the developed AIS model into a game for managing characters, taking into account the features of the game design.
6. Experimental studies of the AIS model in a game project.

Solving these problems will allow to design of an intelligent system for managing characters in a computer game.

2. Design features of the game application

The game application used in the work simulates a futuristic space world where the player interacts with other characters in a free game space [10, 11]. The player, as well as other game characters with automatic control, controls his own spaceship, which can move through the spaces of the game world, find and mine game resources, exchange them at certain points on the map of the game world or by trading with other characters for the development of his own ship. There are several locations on the map of the game world, where the player, together with other characters, can exchange collected resources, repair his ship and increase its level and characteristics. In addition, on the map of the game world, places of potential extraction of game resources of various types are periodically generated.

Among the types of game resources that can be controlled by the player and other game characters, the following types can be defined [10, 11]: energy, cryptocurrency, ore, metal, crystals.

Energy is the main resource consumed by the player's and game characters' autopilot spaceships when moving between locations on the game world map. Also, this resource is used to restore the strength of the ship after battles with game characters. In addition, energy is spent by spaceships for shots during the battle. It should be noted that spaceships in the game cannot independently generate energy, and can buy it from resource exchange centers.

Cryptocurrency is a universal resource for exchanging resources in the game, repairing the spaceship, and increasing its level. This type of resource can be obtained only at resource exchange points. exchanging ore, metal, or crystals for it. With crypto-currency at resource exchange points, the player and game characters who control other ships can purchase energy for their ships.

Ore is the least valuable resource in the game, which can be mined by the player and other game characters on their spaceships. Ore can be found in special mines on the map of the game world. Any spaceship controlled by the player or a character can collect ore from the mine until it is completely exhausted. Collected ore can be sold for crypto-currency or turned into metal, the price of which will be much higher.

Metal is a resource that can only be obtained by smelting collected ore with the player's spaceship or the ships of other game characters. Smelting metal from ore requires a lot of energy and the ore itself, but this resource earns players a much larger amount of cryptocurrency than ordinary ore.

Crystals are the rarest and most valuable resource that can be collected in a mine along with regular ore. The probability of finding crystals is very small, so this resource is valued the most. Crystals can be exchanged for crypto-currency at resource exchange points.

Thus, the player, together with other spaceships, can move around the map of the game world, find new deposits, collect ore from it, sell it at resource exchange points, or smelt metal from it. In addition, the player and game characters can attack each other for ore, metal, and crystals to then exchange these resources for crypto-currency at resource exchange points. Also, characters can protect each other from attacks and exchange resources with each other.

The behavior of game characters who control spaceships on the game world map is determined by the values of their main personality traits. which may change periodically during gameplay. Among the main character traits that affect the behavior of game characters are [10, 11]: aggressiveness, greed, law-abiding, victory, communication, mining, research, exchange, idealism.

Each of these features has the same range of values [0; 100], which makes it possible to simplify the process of measuring the distances between the features of a game character and the class of his behavior. It should be noted that when adding a new game character to the map, these traits are randomly assigned by distributing 300 points between these 9 basic traits of the game character's character traits.

The behavior of the game character is determined by the class closest in terms of characteristics from the following list [10, 11]: robber, defender, worker, scout, trader, engineer.

Accordingly, in the behavior of characters who are closer to the "robber" class, aggressive actions towards other characters will be the main ones, especially if they have a less developed spaceship. The behavior class "robber" is dominated by traits: aggressiveness, greed, and defeatism, all other traits may be of minor importance.

In the behavior of the characters of the "defender" class, the main actions will be the manifestation of aggression towards the "robbers" during their attempts to attack other spaceships. In the behavior class of the "defender" type, the following traits dominate aggressiveness, law-abidingness, and idealism, while all other traits may be of minor importance.

The behavior of characters from the "worker" class will be dominated by actions aimed at extracting ore and processing it into metal. According to this, the behavior class of the "worker" type has the following dominant features: law-abiding, mining, and communication, while other features are of minor importance.

The behavior class of game characters of the "scout" type has the main actions aimed at finding new minerals on the game world map and extracting crystals. Accordingly, in this class, the dominant traits are research, law-abiding, and greed, while all other character traits of game characters will have minor values.

In the behavior of the characters of the "merchant" class, the dominant actions are the exchange of resources with other game characters at a lower price than at resource exchange points. According to this, the main character traits of game characters of this class are exchange, communication, and greed. Other character traits of game characters of this class will have minor importance.

The behavior class of game characters of the "engineer" type has the main actions aimed at finding new ores, searching for crystals and smelting metals, as well as developing your ship. The main traits of this class of behavior are exploration, extraction, and idealism, while other character traits will be of minor importance.

Since in the game the user, as well as each game character, controls his own spaceship, the possibilities of the development of the ship play an important role in the gameplay. Accordingly, the spaceships controlled by the player and other game characters have a number of specific modules that perform different tasks [10, 11]: engine, battery, cannon, radar, drill, smelter, cargo module for ore, cargo module for metal, cargo module for crystals, repair module.

The engine allows the ship to move in the space of the game world of the application. While moving, the engine uses energy from the ship's battery. Leveling up the engine results in a reduction in the amount of energy the engine uses to move. The battery is used by the ship to store the energy that the ship consumes while performing its main functional capabilities. Increasing the level of the battery results in an increase in the amount of energy it can store.

The cannon is used by a ship to damage other ships in combat encounters. Leveling up a cannon increases the damage it deals to enemies in battles.

The radar allows you to find ores on the map of the game world while moving the ship. Increasing the radar level makes it more likely to find crystals in ores during mining.

The drill allows you to mine ore and crystals from ores. Increasing the level of drill allows you to extract more resources in one mining cycle.

A smelter is used to obtain metal from ore, during which a certain amount of energy is consumed. Increasing the level of the smelter allows you to reduce the amount of energy consumed by this module when making metal from ore.

Cargo modules for different types of resources allow you to store a certain amount of resources on the ship for their further transportation to points of sale. Increasing the levels of cargo modules increases the amount of resources that can be stored.

The repair module allows you to restore the ship from damage received during collisions with other ships. This recovery consumes a certain amount of energy, which decreases as the level of this module increases.

An important feature of the game process is the periodic change of the state of character traits of the character per unit of game time - once per minute. At the same time, one point is randomly added or subtracted from a random character trait of each character, after which each game character determines the nearest behavior classes of neighboring characters in terms of proximity.

The analysis of literature sources showed that there are a number of main artificial intelligence methods used in games, such as authoring, tree search, evolutionary computing, machine learning, etc. [12, 13]. It is noted that machine learning is the most common methodology, which will become the main method for games of high complexity. Unfortunately, publications on the use of AI in specific computer games, as a rule, do not indicate the approaches, methods and algorithms used for intelligent information processing, as well as the results of their effective use, which is a trade secret. This significantly complicates the comparative analysis of the proposed model both in terms of accuracy and time costs.

Despite the great interest in the theory of artificial immune systems, the use of immune models in the video game industry has not yet been widely used. Therefore, this work is aimed at studying the possibility of using immune methods for processing information and making decisions in game projects. It should be noted that the purpose of the work is not to form an immune control system for game characters that would be superior to other control systems based on the principles of neural networks, fuzzy logic, or the mathematical apparatus of SVM. The purpose of this work is to adapt one of the most common immune models for solving the problem of controlling game characters and to study the possibility of solving such a problem based on the immune approach.

3. Artificial immune system model for controlling the behavior of characters in games and the architecture of a game project

Today, the theory of artificial immune systems (AIS) is receiving a lot of research attention and is used to solve a large number of practical problems - from the problems of data classification and clustering to the problems of information protection, forecasting, and recognition of voice messages, etc. Therefore, among the existing systems of artificial intelligence, the theory of AIS occupies an important place along with genetic algorithms, neural networks, fuzzy logic, etc. [14, 15].

The problem of controlling the behavior of characters in game software can be considered a special case of the classification problem, for which it is proposed to use AIS. Then the possible options for actions of the game character form a set of classes, which are represented by the population of AIS antigens [14]. At the same time, the characteristics of the class are several parameters that have specific fixed values. These parameters are taken from a limited list of characteristics that determine the

behavior of the game character. Through cloning, mutation, interaction with other characters, and suppression of AIS [16, 17], a type of behavior is chosen for each individual character.

Among the most common immune models of AIS, which are used to solve scientific and practical problems, we can single out the clonal selection model, the artificial immune network (aiNET) model, and the positive/negative antibody selection model [14, 15]. Among the listed immune models, the most promising for practical application is the aiNET model, since it involves the organization of interaction not only between populations of antibodies and antigens but also the interaction between antibodies within the same population [14]. This makes it possible to use this model not only for solving practical problems with guided learning but also for solving problems with self-learning [18, 19]. In addition, due to the possibility of interactions between antibodies, the aiNET model is easier to modify than the other models listed, and it allows the formation of hybrid models that work on the basis of different biological and non-biological approaches to organizing calculations [20]. The aiNET model is commonly used to solve classification, clustering, and pattern recognition problems. After some modification, it can be used to solve other practical problems. Therefore, the aiNET model is chosen as the basis for creating a model for controlling the behavior of characters in computer games.

3.1. Game character data format

To ensure the high quality of aiNET work, at the beginning of its simulation, it is necessary to formalize the type of data and the ranges of their possible values. Formalization of the types of data used to describe antibodies and antigens is one of the important stages in the development and implementation of aiNET. It should be noted that the speed of determining the behavior classes of game characters and the accuracy of making this decision are significantly influenced by the list of features used when determining the affinities between immune objects of different types, which are used by the system during the formation of a specific immune response. We can talk about a linear relationship between the number of signs of an immune object and the speed of formation of an immune response by the system. This is due to the fact that an increase in the number of features of immune objects leads to an increase in the number of computational operations for the calculation of affinity during the operation of immune aiNET operators. It should be noted that now the most common types of description of the features of immune objects are [19, 20]:

- Feature vector.
- Matrix of features.
- Weighted array of feature vectors.

In the case of using a feature vector to describe immune objects, all its features are combined into one array. The main condition for using this form of presentation of the characteristics of the immune objects of the system is the use of the same range of possible values for all features combined into a common vector. Otherwise, the work of immune algorithms can lead to the formation of a large number of errors.

In the case of using a feature matrix to describe the characteristics of immune objects, all these parameters are distributed among a number of groups that have the same range of possible values. It should be noted that the matrix format has a number of features such as the need to normalize groups of features to form the matrix and the absence of weights in the rows of the matrix. These features can lead to the appearance of a large amount of redundant data, which negatively affects the speed of calculating the affinity between objects due to an increase in the number of computational operations.

The use of a weighted array of feature vectors involves the formation of a multidimensional array, the feature groups of which are balanced by predetermined weighting factors that significantly affect the determination of affinity when presenting immune objects by the system. This approach involves the heterogeneity of the features of immune objects and the use of different ranges of possible values for them, as well as a large volume of statistical data for the correct determination of the weight of each individual group.

In accordance with this, the work uses the first of the defined variants of the description of immune objects, namely, the feature vector. This is due to the fact that all character traits of a game character have the same range of possible values, so they do not need to be normalized and further adjusted to each other in order to be used to determine the class of behavior of a character in a game application.

3.2. Features of the immune network model for controlling the behavior of characters

Despite the fact that the aiNET model is one of the most widespread immune models, it cannot be used to solve the problem of controlling game characters without additional modifications. Accordingly, a modified model of the immune network – behavioral aiNET (baiNET) was proposed, which can be used to solve a specific practical problem. The operation of the baiNET model can be conditionally divided into two main stages:

1. The stage of formation of a specific immune response by reproducing a set of antigens that determine the classes of the behavior of game characters in the process of immune training.
2. The stage of determining the objects with which these antibodies will interact to implement the specific behavior of the selected class.

The implementation of these stages is carried out by successive execution of the corresponding immune operators. The proposed baiNET model at the level of sequential execution of immune statements is defined as follows:

$$baiNET = \begin{bmatrix} Presentation(AB, AG) \rightarrow \\ Cloning(AB, CL) \rightarrow \\ Mutation(CL) \rightarrow \\ Supression(AB, CL) \rightarrow \\ Termination(AB, AG) \end{bmatrix}^{aiNET} \rightarrow \begin{bmatrix} Selection(AB, AB', AB'') \rightarrow \\ Specification(AB', AB'') \end{bmatrix}^{Act}, \quad (1)$$

where $Presentation(AB, AG)$ is the operator of presentation of the population of the network of antibodies AB to antigens AG, from which the training sample was formed, as well as antibodies among themselves; $Cloning(AB, CL)$ – the operator for cloning the AB antibody population to obtain a set of CL clones; $Mutation(AB, CL)$ – mutation operator of formed clones; $Supression(AB, CL)$ – operator of suppression of antibodies and network clones; $Termination(AB, AG)$ – the operator for checking the termination of the process of forming the immune response of the network; $Selection(AB, AB', AB'')$ – the operator for determining AB', AB'' antibodies with which this or that antibody will interact, implementing the logic of its class of behavior; $Specification(AB', AB'')$ is an operator for defining AB', AB'' objects with which this or that antibody can interact within its class of behavior, and antibodies with which interaction should be avoided.

Since the greatest influence on the speed of work and the quality of decision-making based on the baiNET model are its immune operators, the specifics of their work and settings should be considered separately [21, 22].

The operator $Presentation(AB, AG)$, which is used in the baiNET model to represent populations of immune objects, is also responsible for defining the target objects for each antibody. It should be noted that this operator is usually called only once during the entire process of forming a specific immune response due to a large number of calculations. However, given the specifics of the game design of this project and the fact that the population of antigens is quite small and has only six objects, this operator is used together with the cloning and mutation operators of antibodies during the formation of each new population of objects. In addition, due to the small number of playable characters on the map, it is assumed that their number is measured in several hundred. All antibodies that describe the behavior of game characters can interact with each other without the need to use restrictions in the form of Natural Affinity Threshold (NAT) or any other criteria [22]. However, from the point of view of utility, such an interaction between all immune entities in the antibody population is not appropriate. The behavior of a game character can be influenced only by characters that are near him on the game world map. Therefore, during the work of the immune presentation operator, for each individual antibody, a circle of antibodies closest to it is distinguished according to coordinates on the map, and not according to character traits. Accordingly, to determine the nearest characters, the Manhattan distance by coordinates in two-dimensional space is used instead of the Euclidean distance, the calculation of which requires more computational operations. Next, among all the objects for each

antibody, 5 objects closest to it on the map will be chosen, with which it will interact during the formation of the immune response.

The cloning operator $Cloning(AB, CL)$ is used to generate a certain number of CL clones for each non-specific AB antibody. It should be noted that static cloning is used in the baiNET model, which involves the use of some external parameter, which sets the number of clones that are formed for each individual antibody that has not acquired the state of specificity for one of the external antigens. Accordingly, the number of clones to be used in the baiNET model during cloning will be 20 for each immune entity.

The mutation operator $Mutation(AB, CL)$ is usually used to make changes to the parameters of previously formed CL clones. In the baiNET model, this operator changes not only the clones but also the antibody itself. It should be noted that the method of mutation of clones and the antibodies on the basis of which these clones were formed during cloning are significantly different from each other. To clone an antibody, random static cloning is used with a mutation coefficient μ equal to one. To change the parameters of the clones, an inversely proportional mutation is used based on the affinity between the antibody and the antigen, with the limitation of the minimum value of the mutation coefficient according to the following expression:

$$\mu = rand\left[\, 0{,}25 \times \left(1 - aff_{ij}\right);\ 1 - aff_{ij}\right], \tag{2}$$

where μ is the mutation coefficient, and aff_{ij} is the affinity value between the i-th antibody, from which a clone was formed for further mutation, and the j-th antigen.

It should be noted that the use of this method of calculating the mutation coefficient can significantly speed up the process of achieving the state of specificity between an antibody and one of the antigens for a small number of populations of immune objects.

The suppression operator $Suppression(AB, CL)$ is used in order to bring the number of antibodies and clones of the immune network to the initial state in order to prevent uncontrolled growth of the network and the use of unnecessary computing resources. According to this, among all the clones whose characteristics were changed during mutation, the clone with the highest value of affinity to one of the antigens representing the classes of behavior of the game characters is selected. This assumes that at the first stage of suppression, the affinity for each of the antigens is determined for each clone. This is due to the fact that the number of behavior classes of game characters is a small value, as well as the number of clones. In addition, it should be noted the sequential process of antibody processing, i.e., the execution of all immune operators given in (1) takes place sequentially over each individual antibody, and not over the entire population together, except for the operator of stopping the process of formation of a specific immune response. Accordingly, antigen and other antibody presentation, cloning, mutation, and suppression occur for each individual antibody, after which this is repeated for the next antibody in the immune network until all antibodies have been processed. Only after that, the possibility of stopping the formation of an immune response by the network will be checked. Therefore, during suppression, the affinity to all antigens is determined for each clone, after which only the clone with the maximum affinity to one or another antigen remains. After that, the data of this clone and its affinity to the defined antigen will be stored in the antibody on the basis of which this clone was formed. According to this, at each individual time of operation of the immune network for each antibody, it is possible to determine the most probable class of behavior based on the maximum value of affinity. At the end of the suppression, all clones are removed from the network.

The baiNET model uses the $Termination(AB, AG)$ criterion operator for stopping the formation of a specific immune response. That is, the number of repetitions of the cloning, mutation and suppression operators to achieve the state of conditional specificity and stop the process of forming an immune response is determined using the specified number of antibody populations, which is used as an input parameter of the algorithm. Accordingly, the number of antibody populations that will be created during the formation of a specific immune response by the network does not exceed one population of immune objects.

The operator for selecting a group of antibodies for further interaction $Selection(AB, AB', AB'')$ is used to determine antibodies AB' and AB'', which will be interacted with in the process of implementation by the game character of a specific type of behavior selected in the process formation of an immune response by the network. It should be noted that this operator is a sequential process of

processing antibodies and using other data about antibodies than data in the process of forming an immune response. During the execution of this operator, the affinity is determined based on the coordinates of the antibodies on the map of the game world. It should be noted that this operator also uses an external parameter that limits the number of interactions with other antibodies to a value of 10 objects. That is, after the end of this operator's work, 10 other antibodies will be selected for each antibody, with which it will cooperate in the future to realize the peculiarities of its behavior.

The object specification operator $Specification(AB', AB'')$ is the last immune operator assumed in the baiNET model (1). The job of this operator is to determine for each of the antibodies that represent the behavior of the game characters, which antibodies it will try to interact with during the implementation of the behavior, taking into account the characteristics of the selected class, and which antibodies should be avoided. It should be noted that the work of this operator is performed sequentially and separately for each of the antibodies in the network.

Accordingly, for each antibody, among the selected 10 closest to it on the map of the antibody game world, the objects with which it will try to cooperate and the objects with which it should avoid are determined. The definition of these objects occurs thanks to the use of specific interaction coefficients, which are added to the main character's traits of the character and affect his behavior. For most traits, these coefficients are 1, but for traits such as aggressiveness and winning, the coefficients will be 4. For traits such as law-abiding, mining, communication, and exchange, these coefficients will be 1.5. According to this definition of affinities between antibodies, at this stage of the work, aiNET will be carried out using the proposed formula for calculating affinity:

$$aff_{ij} = \left(1 + \sqrt{k_1(p_{1i} - p_{1j})^2 + k_2(p_{2i} - p_{2j})^2 + \ldots + k_n(p_{ni} - p_{nj})^2}\right)^{-1}, \quad (3)$$

where aff_{ij} is the affinity value between the i-th and j-th antibodies; k – coefficient of interaction based on character traits; p - signs of immune objects.

Thus, there will be an interaction between objects if the affinity between them, calculated according to (3), exceeds 0.5. Otherwise, the antibody representing the game character will try to avoid interaction or move around the game world map away from this object. It should be noted that if this operator is subject to the action of an antibody with the "robber" behavior class, it will pursue and attack ships that have the least developed "cannon" module. Due to the fact that all ships have the same speed on the game world map, characters with the "robber" behavior type cannot catch up with other objects during their movement, and only attack stationary ships. A ship becomes stationary in one of three cases: extracting ore, exchanging with another ship, and increasing the level of one of the modules. It should be noted that while the ship is at the resource exchange point, it cannot be attacked.

3.3. Implementation of MVI architecture for a game project

The software implementation of the game software, in which the control of the game characters is carried out on the new baiNET model, was carried out on Unity 2017 using the .NET platform and the C# programming language. In addition, the program is implemented on the basis of the common MVI programming architectural pattern [24]. This architectural pattern involves the use of the level of user intent to call a certain functionality, which is implemented as a functional Intent entity. The main features of MVI are the division of the software tool into three (in some variations - four) levels: Model, View, Intent, and State. The developed game application is a fairly complex software tool and is conventionally divided into several main modules with different functional purposes:

- User interface module, implemented at the View level.
- Data presentation module, implemented at the Model level.
- Immune algorithm and operator's module implemented at the Model level.
- Behavior module of game characters, implemented at the Model level.
- The game world map module is implemented at the Model level.
- File operations module, implemented at the Model level.
- Infrastructure interaction module, implemented at the Intent level.

The user interface module presents data types that allow you to organize the interaction with the player and visualize the gameplay of the computer game. It is this module that provides interaction between the user and the functionality of the game application, as well as the process of rendering its dialog windows.

In the data module, there are classes used to work with user data, game character data, immune objects represented by a set of antibodies, training antigens, clones, and game character behavior classes. It should be noted that, in accordance with the basic provisions of the SIS theory, the classes of antigens, antibodies, and clones are arranged in the same way and are inherited from the basic cell class. In addition, the organization of these types of data was formed taking into account the principles of object-oriented programming and SOLID concepts [25].

The module of the immune algorithm and its operators contains the implementation of the baiNet model, classes with methods for calculating Manhattan distance, calculating affinity, and mutation coefficients, as well as the implementation of all immune operators used in this model. At the same time, each immune operator is implemented in a separate class, which is organized on the basis of the Singleton programming pattern [26], which simplifies the possibilities of its use in the immune algorithm, as well as the possibility of modification in the future in the case of scaling the proposed immune approach.

The behavior module of game characters implements the logic of the work of game characters who control spaceships during a game session. It should be noted that these game characters are under the control of the baiNET immune network. For the most part, the behavior of the characters is expressed by a set of Intents, which implement the intentions of this or that game character to implement the behavior determined by the features of this or that class of behavior of bots of the game application.

The game world map module is responsible for the organization and implementation of global and general events on the map, such as the appearance of a new ore or depletion of an old ore, adding new bots instead of game characters whose spaceships were destroyed by robbers or defenders as a result of collisions with other game characters.

The file operations module contains classes that allow you to work with standard file types presented in CSV and XML formats. These formats store all information about registered users, as well as game sessions of each of the registered Accounts. It should be noted that information about registered users of the game application is stored in an XML format file by data serialization. Accordingly, at the start of the game application, this file is deserialized, which downloads data about users and their ratings. Also, an important feature of the game application is the preservation of information about the actions of the user and bots on the map of the game world. Accordingly, the game application implements full logging of actions throughout the user's game session.

The infrastructure interaction module is represented by a list of Intent types that are invoked to provide interaction between the user interface layer and the model. According to the rules of the MVI architectural pattern, objects of the Intent type allow the user to call certain operations and perform certain actions on the game world map during the game. Each functional possibility that is provided in the game has its own Intent, which allows you to change the state of one or another part of the data presented at the Model level. It should be noted that the Intent itself may not have a full implementation of all the necessary functionality that the user is trying to launch. This logic is invoked by the Intent from the types implemented at the Model level in the correct sequence, causing the model states to change.

In Figure 1 shows the file structure of the program, which is divided into three main levels in accordance with the principles of the MVI architectural template. Thus, the Intent folder contains a list of data types that implement one or another functional intent of the user.

The View folder contains a list of scripts that are added to the UI to implement the logic of the user interface. It should be noted that all these scripts are inherited from one ViewScript file, which implements the logic of the window class, and provides the ability to switch the windows of the game application. The Model folder has a decentralized and distributed structure in accordance with the basic principles of SOLID, which provides opportunities for its rapid scaling, editing, and problem-solving in case of errors in the application. It should be noted that the States folder is implemented as a separate part of the Model folder, which stores a list of universal states of the application model along with localized information about data that describes the user and individual bots.

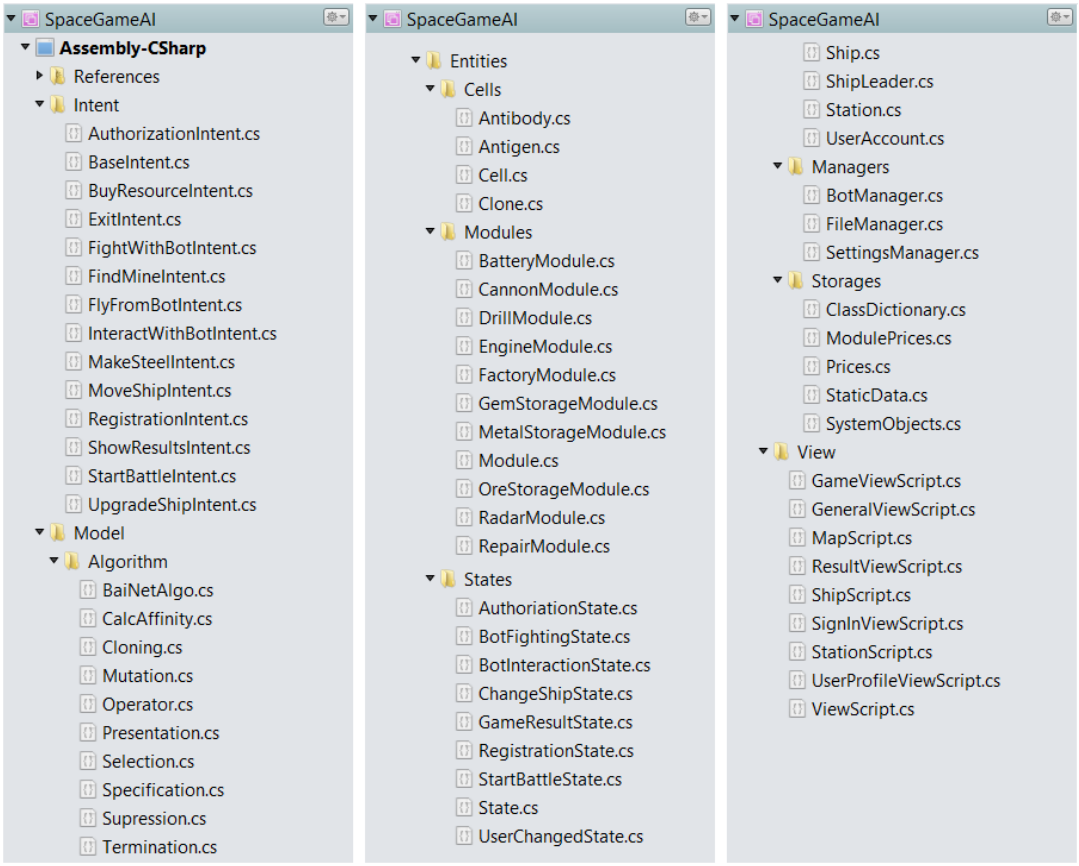

Figure 1: File structure of the application

It should be noted that the design of an artificial intelligence system in game projects, in most cases, is a trade secret of the organization and is classified information, since it can have a significant impact on the behavior of players and the monetization of a game project. In this regard, the task of comparing the proposed method of controlling game characters with the methods and models used in commercial game projects is difficult to implement.

4. Results of using the baiNET model in the game project

To analyze the proposed model of controlling game characters in a computer game, several game sessions were conducted with different numbers of game characters on different size maps of the game world. An important factor in this was the setting of the maps, which regulated the number of mines that simultaneously exist on the map of the game world, their mining capabilities, and the probability of mining these mines for crystals instead of ordinary ore, from which metal can then be obtained for further sale. The characteristics of the data sets are shown in Table 1.

Table 1
Datasets for customizing game world maps

Identifier	Number			Volume of mines	Probability of mining crystals
	Characters	Exchange points	Mines		
Map 1	50	2	10	100	10%
Map 2	100	3	5	50	1%
Map 3	100	3	25	200	20%
Map 4	200	4	10	50	1%
Map 5	200	4	50	300	10%

It should be noted that the generation of spaceships controlled by game characters takes place at various resource exchange points. At these points, ships cannot attack each other, even if they are classified as "robbers" and "defenders". In addition, at these points, ships cannot exchange with each other, exchange can only take place with a system represented by several exchange points.

When comparing the performance of the baiNet model in different conditions of use, the behavior of immune objects that formed the population of antibodies and their relationship to a certain number of classes represented by a set of antigens was studied. In the process of research on bots, attention was focused on patterns and peculiarities of the behavior of game characters in different roles, as well as on the dynamics of changing these roles during the operation of the game application. An important aspect of the work of the game application is the implementation of its role by each of the game characters. By implementing the role of a game character, we will understand the performance of actions that were predicted by the corresponding class of behavior. The test results of the baiNET model for controlling game characters are shown in Table 2.

According to the given results, it can be concluded that some classes are very rarely implemented in full before the ship is destroyed by a robber or the behavior class is not changed during the game application. At the same time, it is possible to distinguish classes that are mainly implemented in cases of a large number of game resources on the game world map, as well as classes that are mainly implemented in the case of a small number of resources on the game world map.

As can be seen from the results of the tests given in table 2, the “engineer” class is implemented the least, especially in the conditions of a lack of resources on the game world map due to the fact that in most cases the choice of this class as a game character is accompanied by an insufficient amount of resources for constantly increasing the level of spaceship modules. It can also be noted that regardless of the features of the game world map, classes focused on mineral exploration and resource extraction are implemented with a high level of probability. Also, among the behavior classes of game characters, the classes "merchant" and "defender" can be distinguished, the probability of their implementation increases in the case of using cards with a large number of resources and a large number of game characters. The probability of implementation of the behavior class "robber" directly depends on the lack of resources: the smaller the number of resources on the map and the more game characters, the higher the probability of implementation of this behavior class.

Table 2

Realization of roles by game characters based on the baiNET model

Identifier	Map 1	Map 2	Map 3	Map 4	Map 5
Robber	78%	93%	73%	95%	65%
Defender	50%	56%	65%	60%	77%
Worker	90%	80%	95%	80%	95%
Scout	90%	90%	95%	85%	95%
Trader	80%	75%	90%	70%	70%
Engineer	35%	20%	50%	18%	48%

It should be noted that the proposed baiNet immune model for controlling game characters is characterized for the first time, as is the artificial immune network model. Therefore, today it is quite difficult to make comparisons with other immune and non-immune models that are used to organize the management system of game characters. It is also important to note that the proposed baiNet immune model is easy to implement and modify, which makes it possible to further use and modify it for controlling characters in games of other genres.

5. Conclusions

Today, the development of game applications has a significant impact on human behavior and habits. At the same time, a large role is played by multi-user games. Thanks to the development of information technologies, new opportunities have appeared for game software developers, as well as for the simplification and improvement of already existing technologies. The use of artificial intelligence

systems allows for solving complex practical problems related to intelligent data processing and analysis under conditions of uncertainty. The development and development of artificial intelligence systems significantly affects the game software market. It is important to create an intelligent model of controlling game characters, to simulate the behavior of players of many custom computer games, and to oppose or cooperate with the user.

As a game application, a space world is used, in which the player controls his own spaceship, which can move through the spaces of the game world, find and mine game resources, exchange them at certain points on the map of the game world or by trading with other characters to develop their own ship. Considered game resources that can be controlled by the player and other game characters. The main features of the behavior of game characters who control spaceships on the map of the game world are considered. Since each game character controls his own spaceship, spaceships have a number of specific modules that perform different tasks in order to develop the ship in the gameplay.

The task of managing the behavior of characters in-game software is considered a classification task, for the solution of which AIS is used. The possible options for actions of the game character form a set of classes, which are represented by the population of AIS antigens. Among the existing AIS immune models, the aiNET model is the most promising for practical application, as it involves the organization of interaction not only between populations of antibodies and antigens, but also the interaction between antibodies within the same population, and is used to solve the problems of classification, clustering,

Therefore, the aiNET model is chosen as the basis for creating a model for controlling the behavior of characters in computer games. But it cannot be used to solve the problem of managing game characters without additional modifications. Accordingly, a modified model of the immune network - behavioral aiNET (baiNET) was proposed, which was used to solve a specific practical problem. Since its immune operators have the greatest impact on the speed of work and the quality of decision-making based on the baiNET model, the peculiarities of their work and settings are considered.

The software implementation of the game software was carried out on Unity 2017 using the .NET platform, the C# programming language, and the MVI architectural programming pattern. Experimental studies were conducted on the use of the baiNet model with different numbers of game characters on different-sized maps of the game world, which showed that the proposed baiNet immune model is effective and simple to implement and modify. This makes it possible to use it to control characters in games of other genres.

6. References

[1] P. Lankoski, S. Björk, Game Research Methods. An Overview, Pittsburgh, PA: ETC Press, 2015. Doi:10.25969/mediarep/13581.

[2] F. Schröter, J.-N. Thon, Video Game Characters Theory and Analysis, DIEGESIS, 3.1 (2014) 40-77.

[3] J. Oceja, V.J. Villanueva-Blasco, A. Vázquez-Martínez, V. Villanueva-Silvestre, S. Al-Halabí, Keep Playing or Restart? Questions about the Evaluation of Video Game Addiction from a Systematic Review in the Context of COVID-19, Sustainability, 15 2 (2023) 1456. Doi:10.3390/su15021456.

[4] W.R. Boot, Video games as tools to achieve insight into cognitive processes, Front Psychol 6, 1 3 (2015). Doi:10.3389/fpsyg.2015.00003.

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

[6] A. Gast, Identification with Game Characters. Effects of visual attributes on the identification process between players and characters, Stockholm, 2017.

[7] 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, 143, (2023). Doi:10.1016/j.chb.2023.107694.

[8] M.A. Quiroga, A. Diaz, F.J. Román, J. Privado, R. Colom, Intelligence and video games: Beyond “brain-games”, Intelligence, 75 (2019) 85-94.

[9] A. Simons, I. Wohlgenannt, S. Zelt, M. Weinmann, J. Schneider, J. vom Brocke, Intelligence at play: game-based assessment using a virtual-reality application, Springer (2023). Doi: 10.1007/s10055-023-00752-9.
[10] USgamer Team, What's The Best Video Game Spaceship? (2019). URL: https://www.vg247.com/whats-the-best-video-game-spaceship.
[11] R. Snyder, 10 Best Spaceships in Video Games (2023). URL: https://www.dualshockers.com/best-spaceships-in-video-games/.
[12] X. Fan, J. Wu, L. Tian. A Review of Artificial Intelligence for Games, Part of the Lecture Notes in Electrical Engineering book series, vol. 572, 2020, pp. 1821. Doi:10.1007/978-981-15-0187-6_34.
[13] Y. Lum, W. Li. Techniques and Paradigms in Modern Game AI Systems, Algorithms, 15 8 (2022) 282. Doi:10.3390/a15080282.
[14] D. Dasgupta, S. Yu, L.F. Nino, Recent Advanced in Artificial Immune Systems: Models and Applications, Applied Soft Computing, Elsevier (2011) 1574-1587. Doi: 10.1016/j.asoc.2010.08.024.
[15] M. Read, P.S. Andrews, J. Timmis, An Introduction to Artificial Immune Systems, In Handbook of Natural Computing, Springer, Berlin, Germany (2012). Doi: https://doi.org/10.1007/978-3-540-92910-9_47.
[16] K.B. Bahekar, Classification techniques based on Artificial immune system algorithms for Heart disease using Principal Component Analysis, International Journal of Scientific Research in Science, Engineering, and Technology, IJSRSET, 7 5 (2020) 150-160. Doi: 10.32628/IJSRSET207542.
[17] S. Shekhar, D.K. Sharma, D.K. Agarwal, Y. Pathak, Artificial Immune Systems-Based Classification Model for Code-Mixed Social Media Data, IRBM, 43 2 (2022) 120-129. Doi: 10.1016/j.irbm.2020.07.004.
[18] H. Park, J.E. Choi, D. Kim, S.J. Hong, Artificial immune system for fault detection and classification of semiconductor equipment, Electronics, 10 944 (2021) 1-14. Doi: 10.3390/electronics10080944.
[19] S.S.F. Souza, F.P.A. Lima, F.R. Chavarette, A New Artificial Immune System Based on Continuous Learning for Pattern Recognition, Revista de Informatica Teorica e Aplicada, RITA, 27 (2020) 34-44. Doi: 10.22456/2175-2745.102061.
[20] 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), 13811 (2023) 77-92.
[21] M. Korablyov, O. Fomichov, N. Axak, Classification of objects based on a tree-shaped artificial immune network model, Advances in Intelligent Systems and Computing V, Springer, (2021), 160-172.
[22] M. Korablyov, O. Fomichov, M. Ushakov, M. Khudolei, Dendritic Artificial Immune Network Model for Computing. Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems (CoLInS 2023). Volume III: Intelligent Systems Workshop Kharkiv, Ukraine, 2023, 206-217 pp.
[23] C. Lan, H. Zhang, X. Sun, Z. Ren, An intelligent diagnostic method based on optimizing B-cell pool clonal selection classification algorithm, Turkish Journal of Electrical Engineering and Computer Sciences, 28 (2020) 3270–3284. Doi:10.3906/elk-2002-75
[24] A.K. Bheemaiah, MVI Design Pattern (2022). URL: https://www.codementor.io/@anilkumarbheemaiah/mvi-design-pattern-1q20f2zcx0.
[25] E. Ghergu**,** Fundamental Software Design Principles for Quality Coding, Software Development (2022). URL: https://www.pentalog.com/blog/it-development-technology/solid-principles-object-oriented-programming/.
[26] K. Paralkar, Singleton Design Pattern – How it Works in JavaScript with Example Code (2022). URL: https://www.freecodecamp.org/news/singleton-design-pattern-with-javascript/.