Game Strategy Analysis Methodology

Miruna Chindea* miruna.chindea@ro.bosch.com
Andrei Vasile Iosif* andrei.iosif24@gmail.com
Lukacs Roland Elekes* lukyelekes@gmail.com
Dorian Gorgan* dorian.gorgan@cs.utcluj.ro

*Technical University of Cluj-Napoca, str. Memorandumului 28, Cluj-Napoca, Romania

ABSTRACT
Strategic games require users to improve their approaches to be faster and more efficient to win the game. This is not a trivial task, as there are many strategic options, and it is also unclear whether the strategy used is the best one. The research proposes a methodology to identify and analyze the strategic options to finde out the optimal strategy for achieving an objective or sub-objective within the game. The paper exemplifies and highlights the main issues of the methodology.

Author Keywords
Computer game; game strategy; game development methodology; game evaluation.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

DOI: 10.37789/rochi.2022.1.1.9

INTRODUCTION

Video games became part of our daily life as a popular form of entertainment. They are interactive applications, that usually require a continuous interaction of humans and computers or other platforms.

Video games are developed for every age category, there are games for children, who are learning how to speak, and there are applications for adults, and they fulfil different purposes. The most common goal of video games is entertainment, however there are other purposes, such as educational, cultural, scientific, etc.

In games, users interact with the scene and can configure the game using the graphical user interface. Users also use continuous interaction techniques to control the game. To evaluate the quality of the game and advance, various game strategies are implemented that use scoring and penalty systems. Today there is a wide variety of 3D games, adapted for all types of devices (mobile phones, tablets, laptops, etc.).

Most of the games available on the market provide an AI agent, that makes possible for the player to compete with the computer for the objectives set in the game. This is a challenging task for the developers because they need to make the game challenging for the user, but the game needs to be winnable to be more entertaining and motivating.

The video games are grouped in different categories such as Action, Adventure, Racing, Platform, Simulation, Role-Playing, etc. As an example, an Action game focuses on the interaction with users, they are challenging and require good skills from the users, such as reaction-time, hand-eye coordination. In Racing games, players participate in racing competitions, such as car races, etc. However, in most of the cases, video games combine aspects of different categories.

The objectives in the games are connected to the category and the type of the game. The player needs to follow these objectives for winning the game (e.g., collect a certain number of resources, find a special character, finish in time a racetrack, etc.), and the game strategy describes these objectives and how the player can reach them, how can he achieve good performance and results. A strategy is a sequence of actions planned to successfully achieve an objective and describes the way of doing it.

To achieve high performance the user needs to find out and to understand the best strategy not just by playing but as well by identifying, studying, simulating, and evaluating the various strategic options. Some specific approaches and techniques could be used to speed up the strategy identification process.

The main contribution of this research activity and paper is the development, description and experimentation of the methodology for the identification and analysis of game strategy.

The paper exemplifies and highlights the main challenges and issues of strategy identification and analysis methodology.

The paper is structured as follows. Next section describes related works on defining, developing and experimented different solutions for game strategy. The following sections present the definition, types, components, and approach phases of the game strategy. The Experimental use cases present three game strategy models: mathematic, algorithmic, and heuristic.

RELATED WORK

Many of the game strategies in which two competitors are in competition are solved by finding the Nash equilibrium, a notion defined and demonstrated in 1950 by the mathematician John Forbes Nash [1]. The Nash equilibrium describes a state of strategic equilibrium, in which a player

has no advantage, changing the strategy by himself. Each player's strategy is the best response to the strategies chosen by the other players.

To analyze all the possible combinations a user can choose to play during a strategy game, it must think of the variables of the game as an n-dimensional solution space [2]. The method is based on combining the computation power of a computer with the analytical thinking and decision-making capabilities of a human being. The solution provides the user with visual guidance throughout the n-dimensional space to explore better strategies for a specific game input.

There are several approaches for improving strategies in video games. Related works mainly focus on RTS games (e.g., Starcraft, Civilization IV), which are multiplayer games, meaning the multiple players are playing the game simultaneously. These games require the player to build buildings, cities, train workers or military units depending to attack or defend against other players. This building and developing of resources are part of the strategy and has many variables which can contribute to a successful plan of winning the game.

A use case based reasoning could be applied to train the system to learn and predict the individual strategies of the players [3]. The authors use the replays of a commercial game Real Time Strategy (RTS), to assess the behaviors of a human player and to build an intelligent system for learning human-like decisions and behaviors.

The RTS platform has been used as a challenging platform for implementing Reinforcement Learning (RL) techniques in real applications [4]. The authors have proposed a multi-layered framework that significantly reduces the computational complexity of RL by breaking down the state space in a hierarchical manner. RL is an area of machine learning in which the agent learns by trial and error by optimal actions for maximizing rewards in executing different tasks.

In the Civilization IV game RL has been used for modelling and improving the game strategy [5]. The authors used RL over states space and actions space of the game, by focusing on local strategies, low level decisions and the characteristics of characters in the game. The reward model is based as well on the score obtained by the player. The RL approach for developing strategies showed good results in quickly obtaining strategies for winning the game.

RL is mentioned in the literature as a good approach for strategical decision making in video games [6]. The work reviews the literature for AI techniques applied in RTS games and studies especially the game named Starcraft. The game uses Case-Based Planning as a planning technique that finds similar past situations to solve the current problem. Hierarchical planning breaks up the problem into sub-problems and the system tries to find solutions only for parts of the problem. By doing an abstraction, the complexity of the problem is reduced, and a solution can be built up incrementally.

Besides classical control methods and algorithms, in last decade important advances have been made in the field of navigation based on RL. The deep reinforcement learning approach is used to find out the control strategy for navigating throughout a dynamic environment with rapidly moving obstacles [7]. The goal is to reach the destination without colliding with walls or obstacles. The proposed RL architecture takes raw range sensor data and relative position data as input and generates discrete control commands.

The RL approach has been used to find out the game strategy in digital curling by reaching the Nash equilibrium [8]. The digital curling game is a two-player zero-sum extensive game in a continuous action space. The game has many action strategies, large search space and strong uncertainty, and it is a typical extensive form game. The extensive game is a tree-based form that expands in the form of multi-player interaction. Two reward mechanisms for the deep reinforcement learning have been experimented. For each round, the current and future rewards were evaluated, analyzing the pros and cons of the two reward mechanisms in the experiment.

Another game strategy modeling has been used to study the lane-changing for autonomous driving vehicles, by considering the dynamic factors in the traffic environment [9]. The lane-changing collision probability and the lane-changing dynamic risky coefficient are the dynamic influencing factors of lane-changing process for autonomous driving vehicles. The results show that under the decision-making behavior model of the lane-changing game, the average speed of vehicles increases and the average number of passed vehicles increases as well, which has higher stability, safety, speed gains, and lane utilization.

The distributed game strategy has been explored for unmanned aerial vehicle (UAV) formations with external disturbances and obstacles [10]. The strategy is based on a distributed model predictive control framework and Levy flight-based pigeon inspired optimization. It is designed an obstacle avoidance strategy based on topology reconstruction, by which the UAV can save energy and safely pass obstacles optimizing a cost function within a game problem.

The game strategy is an effective approach to study the migration of tasks from mobile devices to cloud to prolong the battery life [11]. The research adopts the game strategy to obtain a beneficial offloading group where all devices can simultaneously offload the tasks to achieve energy saving and the number of beneficial offloading devices is maximum for the group.

The financial market is a nonlinear stochastic system with continuous and discontinuous random fluctuations. A noncooperative game strategy in cyber-financial systems is

proposed in [12]. The set of local linear systems are interpolated to approximate the nonlinear stochastic financial system.

The differential game strategy is analyzed in [13], where a three-player conflict includes an attacker, a defender, and a target. Different optimal guidance laws are investigated and the feasible conditions for the attacker are analyzed through nonlinear simulations to accomplish an attack task.

An optimal strategy selection approach for moving target defense based on Markov game is first proposed in [14]. The moving target defense model is based on moving attack and exploration surfaces. Therefore, the random emerging of vulnerabilities is described as the cognitive and behavioral difference of offensive and defensive sides caused by defensive transformation.

Some studies analyze the gaming strategy and user activity patterns in-game data to detect the in-game particular activities [15]. The goal is to find out hidden values using various and enormous amount of user-based data to predict patterns of gameplay, in-game symptom, eventually enhancing gaming.

GAME STRATEGY IDENTIFICATION AND ANALYSIS

To control the strategy of a game, it is necessary to *identify* the elements that define the strategy - *type, objectives, components, actions, strategic options* or what values and parameters must be chosen to be successful. What are the actions and decisions by which a player builds up the sequence of strategic movements? The strategy of course depends on the category and type of the game, but the decisions the player makes at each action are *strategic options* that can speed up the achieving of certain local or global objectives. After identifying the strategy, the player can more easily control the movements to be more efficient in reaching a goal.

If the purpose of identifying the strategy, through simulation or prototyping, is aimed at the effective *implementation* or improvement of the strategy, then it is necessary to determine the most suitable implementation solutions, such as mathematics, logarithmic, through rules, heuristics, etc.

An already implemented game strategy must be *tested* and experimentally *validated* during execution. Experimentally measured data (e.g., time, errors, losses, etc.) must be analyzed. In fact, the entire game strategy must be *analyzed* in order to draw conclusions about the quality and performance of the strategy.

Let us further elaborate and exemplify the defining elements and techniques of game strategies.

STRATEGY DEFINITION

In game theory, the strategy of a player is any of the options he chooses in a setting where the outcome depends not only on his own actions, but also on the actions of others.

The strategy of the player will determine the action that the player will perform at any stage of the game. A very important notion in strategy is the notion of movement, which is an action taken by one player at a time during the game. Strategy based classification of the games is the following:

• *Cooperative* (players can do binding commitments) or *non-cooperative* (players cannot form alliances or all agreements must be self-executing).

• *Symmetrical* (the gains from applying a strategy depend only on the other strategies used, not by the players) or *asymmetrical* (the sets of strategies of the two players are not identical).

• *Zero amount* (the options of the players cannot increase or reduce the available resources) or with a *non-zero amount* (the win of a player does not necessarily affect the rest of the players).

• *Simultaneously* (players move simultaneously or the current players are not aware of the actions of previous players) or *sequentially* (current players are aware of the actions of previous players).

To conclude, the general definition of strategy is the sequence of operations and maneuvers performed by man or machine in the game, to achieve an intermediate goal or achieve the final goal, the victory.

STRATEGY TYPES

There are mainly two types of strategies. The first type of strategy is the *global or overall strategy*, which is chosen by the player throughout the game, for example winning the game, achieving a maximum score, or minimize the resources of the opponent. For the player to achieve the main objective of the game, it divides it into several sub-objectives. Therefore, the second type of strategy is the *local strategy*, applied at the level of an intermediate sub-objective. Some examples of local strategies are the strategy for getting to a certain position, reaching a score, getting a weapon, eliminating an opponent, trapping your opponent, looking for the best options, etc.

STRATEGY COMPONENTS

A game strategy consists of the following elements: (1) Mission of the game; (2) Resources; (3) Main objectives and sub-objectives; (4) Strategic options; (5) Evaluation metrics.

Mission of the game

The mission describes the active or passive objects in the game scene, the static or mobile objects, the interactions between the objects, the interaction of the user with the objects in the scene, the actors who populate the scene, in such a way as to reflect the values and priorities of those who decide on the strategy in the game.

Characteristic of the mission is that it is not an enumeration of the quantifiable elements to be achieved, but only suggestions, perspectives, and attitudes.

Resources

Resources are all the elements needed as inputs for strategies to be operational. *Material resources* - computer system, system configuration, components, interaction devices (e.g., monitor, keyboard, joystick, control console); *Human* - human potential of the user with a certain generic profile, number of players, user capacity; *Informational* - optimization criteria, monitoring metrics, algorithms, heuristics; *Rewards* - financial, points, time, bonus, ranking.

Main objectives and sub-objectives

The objectives of the game refer to the states through which the game evolves. The intermediate states between two main states are called sub-objectives. They are identified and distinguished by certain parameters, attributes, and values such as scoring value, maximum score, insufficient resources, a certain position, set of weapons, active opponent, inactive opponent, destroyed opponent, etc.

Strategic options

During the game execution the player must decide on the next *movements* and choose the most *appropriate values* of parameters, attributes to achieve the objectives.

The goal of the strategy is to optimize the parameters, attributes, and values to achieve the objectives. Solutions for *strategic approaches* or ways could be as follows: (1) parallelization, distribution, delegation, replanning of operations; (2) flexible resource management; (3) identifying solutions to achieve a local optimum; (4) extending local optimal conditions by applying new strategic options; and (5) combining local optimal solutions to achieve a global optimum.

Evaluation metrics

The evaluation metrics are the criteria according to which the strategy quality is assessed, e.g., capacity of resources, score, execution time, etc. Defining the metrics consists of detailing the elements such as main criterion, measurable parameters at monitoring, measurement method, values of the main parameters (e.g., normal, very good, good, sufficient, unacceptable).

STRATEGIC APPROACH PHASES

Strategy identification

The strategy could be identified as an already implemented *mathematical* or *algorithmic* model. The mathematical model is identified as a set of local or global optimal functions. Therefore, the algorithmic model is a set of intuitive rules (i.e., condition, action) or an algorithmic composition of models.

If the strategy is not described by either of the two previous models, strategy identification could be *heuristic* as follows:

- Tutorial on operations, game sessions, algorithmic rules, through explanations, demonstrations, programming techniques and exercises.
- Experiencing use cases, situations and game solutions.
- Simulation of operations, scenarios with comparative analysis of solutions.
- Achieving global solutions by combining local solutions.

Strategy Implementation

The implementation of the game strategy could be *mathematical* - by computing some optimal functions; *algorithmic* - by implementation of local and global algorithms; and *heuristic*, by contextual and adaptive application of different and diverse techniques such as machine learning, deep learning, combination of rules based on observations, etc.

Strategy execution

The strategy can be executed by both the player and the system in the following way: (1) *Automatic* - it is applied without user intervention along the execution, some initially specified values / menu options are used by the user or fixed values; (2) *Interactive* - the user interacts directly in the development and monitoring of the strategy, permanently following the way of achieving the objective; (3) *Combined* (human in the loop) - alternate between automatic and interactive execution mode. The alternating rule of the two modes can be automatic or interactive, as specified by the user.

Testing and validation

Assessing the quality of the strategy requires to set up:

- *quality criteria* - the quality of the strategy is assessed according with a set of criteria such as minimum resources, maximum score, minimum losses, minimum execution time, achievement of the objective / sub-objective, local / global optimization.
- *tasks and scenarios* - the tasks (strategic actions, options) that build up the specific strategy evaluation scenario.
- *measurement method* - what are the parameters to be measured / monitored, and what is the measurement method to infer the fulfillment of the criteria.
- *assessment metrics* - the reference values of the measured parameters (e.g., normal, good, unacceptable).

EXPERIMENTAL USE CASES

Heuristic identification of the strategy

In the following Endless Runner game, two game strategies will be analyzed and compared. In order to simplify the exemplification, the number of collected points (i.e., score) is given by the number of objects and points offered by

each of them. The player receives the final number of points and the time of each round of play, without having access to the mathematical computation mechanism. A more complex mathematical computation could compute the number of collected points taking in account other contextual conditions (e.g., cost of running, distance of collecting over a graph of movements, minimal distance, cost of picking the objects, etc.). Therefore, the identification of the best strategy is available just by experimentation.

• Strategy #1: Collecting objects that offer a small number of points (e.g., 5 points). It should be noted that these elements are often encountered along the route.

• Strategy #2: Collecting objects that offer a great number of points (e.g., 20 points). It should be noted that these elements are rare on the route.

The purpose of such a strategy is to get the highest possible score in the shortest possible time. The criteria according to which the quality of the strategy is assessed are the following:

• The score obtained by the player, measured in points. For the strategy to be evaluated in a round of play, the score obtained by the player must be at least 100.

• The time required to obtain the score, measured in seconds. For the strategy to be evaluated in a round of play, the time measured until the loss of a round (hitting an obstacle) must be at least 30s.

The metric used to evaluate the quality of the strategy is the StrategyScore, computed as the arithmetic mean of the ratios of the total number of points divided by the time of each round. This average score should be as high as possible.

$$\text{StrategyScore} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Total\_points[i]}{Time[i]}\right)$$

The two strategies were tested, 5 times (n=5 rounds), without counting the invalid experiments, as previously specified. The results obtained in the experiments, for each strategy, are illustrated in Table 1. The average score obtained for Strategy #1 is 5,660, and the average score obtained for Strategy #2 is 5,390.

Therefore, as can be seen from the results of the experiments performed, the score obtained by applying the first strategic option is higher than the score obtained when applying the second strategic option. In conclusion, on average, a player will get a high score in a shorter time if he chooses to collect multiple items that offer a small number of points (5 points), instead of collecting objects with large number of points (20 points).

Table 1. Comparative experiments on Strategy #1 and Strategy #2.

	Strategy #1			Strategy #2		
Exp. Nr.	Points	Time	Score	Points	Time	Score
1	175	32	5,469	160	34	4,706
2	720	122	5,902	480	89	5,393
3	235	43	5,465	560	103	5,437
4	325	66	4,924	260	47	5,532
5	510	78	6,538	700	119	5,882
Average			5,660			5,390

The strategy is modeled and implemented by functions, which are computed and minimized or maximized during the game running.

Algorithmic strategy model

The obstacle detection and avoidance strategy are used in video games where the player controls a vehicle in 3D space, with many degrees of freedom. The game strategy identification and validation are performed by comparing the execution of the two algorithms. Strategic options are about controlling the algorithms that define the strategy.

Strategy components

Objectives. The objective of the strategy is the safe navigation between two waypoints, denoted by A and B. Between the two points there may exist zero or more static obstacles. As sub-objectives, we may define the following:

1. Determining if an obstacle blocks the direct path between A and B
2. Avoiding one or more obstacles
3. Returning to the original path after the avoidance is finished

3D scene. The 3D scene and the game for which the obstacle avoidance strategy is applied was developed in Unity, and there is a 3D shooter from third-person perspective, in which the player controls a spaceship. Relevant for the strategy are the static obstacles from the 3D scene, represented by groups of asteroids.

The player spaceship can collide with one asteroid if the ship gets too close to it. For the purpose of the strategy solution exemplification, we model collisions using spherical colliders.

Interaction. The player can interact continuously with the spaceship and can modify the following parameters: increase or decrease forward acceleration and modify the values of the Euler rotation angles (roll = rotation along the "forward" axis; pitch = rotation along "right" axis; yaw = rotation along "up" axis). Thus, when faced with an obstacle that blocks the forward movement direction, the player has multiple options for avoiding the obstacle.

Evaluation metrics. The criteria for evaluating the quality of the proposed strategy are:

• *Navigation time*, which should be as small as possible. This reflects real-life scenarios/applications in which an autonomous vehicle should reach the destination in the shortest time possible, due to critical missions (for example, delivery of goods to remote or disaster-affected locations).

• *Collision rate*, which should be also minimized. The collision rate is defined as the number of times the ship collided with an obstacle, divided by the total number of trials.

Generic strategy for object avoidance

A generic obstacle avoidance strategy is presented in Algorithm 1. The procedure uses the current position of the vehicle and can modify the controls (forward acceleration, pitch, and yaw) that need to be applied to avoid one or more obstacles.

Algorithm 1: Generic obstacle detection and avoidance procedure

```
while GoalNotReached() do
    detections ← DetectObstacles(crrPos)
    ClasiffyObstacles(detections)  // Optional
    if length(detections) == 1 then
        avoidanceTarget ← detections[0]
    else
        obstacleDensity ← ComputeDensity(detections)
        if obstacleDensity > densityThreshold then
            avoidanceTarget ← FindCentroid(detections)
        else
            avoidanceTarget ← FindClosest(detections)
    if DistanceToObject(avoidanceTarget) < criticalDistance and
          !AvoidanceComplete(avoidanceTarget) then
        SingleObstacleAvoidance(avoidanceTarget)
    else
        // Move towards goal
end
```

The procedure starts with the detection and classification of obstacles that stand in the way to the goal. The focus of this paper is on static obstacles.

The avoidance part considers two general cases: avoiding a single obstacle or avoiding multiple obstacles. In the latter case, we may consider two strategic options, depending on the obstacle density as well as on the trade-off between fast collision avoidance and risk-free collision avoidance:

1. Avoid the entire obstacle group: this option is equivalent to avoiding a single larger spherical obstacle as a 3D bounding box of the group of objects. This strategic option offers a lower risk of collision, but may lead to increased navigation/avoidance time, depending on the configuration of the obstacle group.
2. Sequential obstacle avoidance: in this case, the obstacles are spread apart, and the ship can navigate safely between them. The agent will always select the closest obstacle that blocks the path to the goal as avoidance target. Using this strategic option we may have a higher risk of collision, but it may lead to decreased navigation/avoidance time, especially if the density of obstacles is small.

Single obstacle avoidance

In the single obstacle avoidance scenario, we assume that a single obstacle has been detected and the ship is currently located at a distance $d_{obs} < d_{max}$ from the obstacle. In this case, the ship continues to move along the initial path and the avoidance manoeuvre will start when the distance to the obstacle becomes smaller than the critical avoidance distance d_{min}.

The avoidance procedure for a single obstacle is described in Algorithm 2. When the critical avoidance distance is reached, a fixed number of rays are casted from the ship to the obstacle. The rays are equally spread on a cone and are tangent to the obstacle. Each ray represents a possible avoidance direction that the ship can take.

Algorithm 2: Avoidance for single obstacle procedure

```
Input: obstaclePos - position of avoidance target
Input: d_min - critical avoidance distance
Input: numRays - number of possible directions for avoidance
while ManeuverNotFinished() do
    if distance(crrPos, obstaclePos) < d_min then
        cost ← ∞
        bestPitch ← crrPitch
        bestYaw ← crrYaw
        for i ← 0 to numRays − 1 do
            rayAngle ← 2πi/numRays
            rayDirection ← ComputeTangentRay(rayAngle,
                  obstaclePos)
            newPitch ← −arcsin(rayDirection.y)
            newYaw ← arctan(rayDirection.x/rayDirection.z)
            crrCost ← max(abs(crrPitch − newPitch), abs(crrYaw
                  − newYaw))
            if crrCost < cost then
                cost ← crrCost
                bestPitch ← newPitch
                bestYaw ← newYaw
        end
        // Apply rotation with bestPitch and bestYaw
    else
        // Continue moving towards goal
end
```

The selection of a direction can be formulated as an optimization problem, in which we try to minimize some cost function. For example, we can consider a cost associated with each manoeuvre, in terms of fuel/energy consumption. Manoeuvres which require large deviations from the current direction can be considered as more costly than manoeuvres which only slightly modify the current course.

In Algorithm 2, the *ManeuverNotFinished()* function determines whether the ship has successfully avoided the obstacle and can return to the original trajectory. The simplest condition for this to be true is that the angle between the current direction vector of the ship and the direction from the ship to the centre of the obstacle is larger than a threshold (which can be set experimentally, for instance, 65º).

Strategy evaluation

For testing and validation of the proposed control strategy for avoiding single static obstacles we can use the following scenario:

1. Spaceship is moving forward with constant speed, and it must reach a certain goal position.
2. A static obstacle (asteroid) is instantiated on the path between the start position and the goal position. For each experiment run, a small random offset is added to the asteroid position. This way, the ship will be forced to take different directions each time, to minimize the proposed cost function.
3. For each experiment run, the total navigation time should be monitored. Also, we need to keep track of the collision rate over all experiment runs.

Heuristic identification of the strategy by exercises

In the case of racing, the strategy could be a succession of actions to drive correctly and efficiently in a dangerous turn. In such a case, the optimal strategy is determined by heuristics, which means that the player discovers for himself through exercises or attempts how to best perform the actions and specify the appropriate values for the parameters.

In racing competitions, there is a concept called the racing line (Figure 1). The racing line is the route a racing driver follows to take corners in the shortest time at the highest average speed [6]. By using all the available space on the track, cars can travel in a straighter line and travel faster before reaching the limits of grip. Determining the best line is an essential skill to master the race.

Furthermore, the driving technique is another key factor for obtaining good results, following the race line, adjusting speed, braking, and accelerating are parts of success, a player cannot race with only the acceleration. Another element is adapting the vehicle type to the racetrack, this means that on a track with many hard corners, a car with lower maximum speed, but higher acceleration due to lighter weight can be more suitable. This is an aspect that can be taken into consideration in further developments of the game.

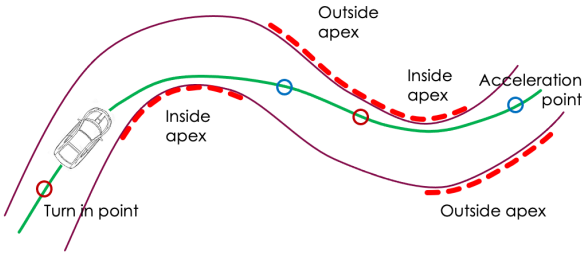

Figure 1. Learning the optimal trajectory through experiments [6]. The racing line is the green trajectory.

The player can configure the game for himself, which means that he can select a game mode, select the preferred car and the map (i.e., racetrack) on which he wants to play. The game provides three different game modes, the first one is a simple Free Drive mode, which allows the player to get used to the control of the car, practice his driving skills, hand-eye coordination skills, adapt them to the game. This game mode does not have objectives to be completed, however the user can practice some elements of the driving that are useful in winning the game in other modes. For instance, to get a better result (e.g., time, metrics) than other driving sessions.

The second game mode is called Best Time mode, here the objective is to finish the racetrack in the best time possible. The performance of the player is rated by stars depending on time intervals in which he completed the map, the best rating is three stars and the worst performance (i.e., completing the map but in a very long time) is rated with only one star.

The third game mode is the Race mode, in which the player competes with other cars, controlled by the computer (i.e., AI agents) with the objective of finishing the race on the best possible position.

In both last game modes, a race starts after the player finishes configuring the game and finishes when the player reaches the final of the racetrack. When the player reaches the final point, the race is finished and the performance of the player is evaluated in both cases, which can be based on the time needed for the race or the position on which he finishes the race.

Therefore, the global strategy is to win each game (i.e., race) by obtaining the best result in the selected game mode (e.g., finishing with best time, finishing on first position and additionally with best time). The local strategies are represented by actions done for obtaining local results, which could be smaller tasks such as reaching a certain position, eliminating an opponent, getting a better time and dynamics on a given road segment, turning in the wheels, determining the appropriate accelerating points, overtake an opponent car, when it is in front of the player.

The implementation of the strategy could be achieved through various solutions based on machine learning, the set of rules and algorithms.

Heuristic strategy identification by Genetic Algorithms

Genetic algorithms are adaptive heuristic search techniques, based on the principles of genetics and natural selection. The mechanism is like the biological process of evolution. This process uses a feature that only species that adapt better to the environment are able to survive and evolve over generations, while those that are less adapted fail to survive and disappear over time due to natural selection. We use these principles in searching for the optimum strategy of the game.

The game consists of the user moving a ball from the starting position to the destination position, marked by a flag (Figure 2). Each time the ball falls off the track, the race resumes from the starting position. The user has the possibility to modify the characteristics of the objects in the scene or the movement parameters (e.g., the ball speed, the friction coefficient of the track, the acceleration of the movement, etc.).

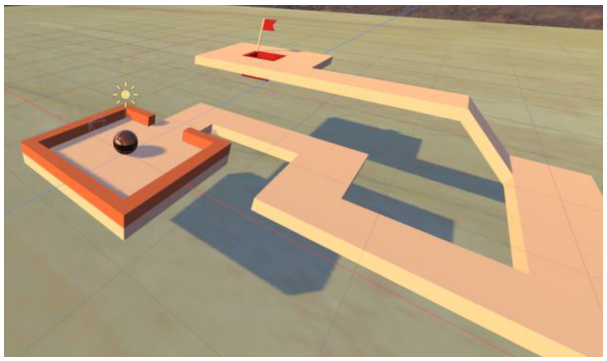

Figure 2. Movement of the ball over the track from starting position toward the flag position.

The genetic algorithm executed by the system is able to generate various solutions for moving the ball, in the context of options specified by the player. Therefore, through such a heuristic approach the player can experiment and learn new movement techniques from the system. He can then perform these moves himself to win the game. For example, with a given speed the ball is able to jump and overcome some parts of the track or may control the speed in order to minimize the time.

Heuristic strategy implementation by Machine Learning

The heuristic identification of the strategy could be achieved through machine learning approach. The user is able to experiment various strategy options but is not able to describe the strategy by algorithms, mathematically, or even to identify all the conditions that have strategical impact.

However, the user is able to specify the positive and negative cases of the actions and states in the game in order to train a neural network.

Let us exemplify the strategy implemented by machine learning approach, in the Windrunner game. The strategy helps the avatar to avoid and overcome a lot of obstacles (spikes), by minimizing the time and costs (Figure 3).

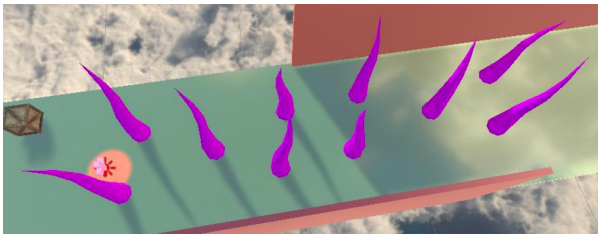

Figure 3. The avatar is passing through a set of static and dynamic spikes.

The reinforcement learning technique is used to train agents who interact over time with the environment in which they activate. Learning can be achieved both through traditional machine learning methods and through methods based on deep neural networks (deep reinforcement learning). Thus, the agent learns to select an appropriate action, depending on the context. During training, rewards are used to mark the correct actions of the agent, or penalties for mistakes [16]. A such a method is Proximal Policy Optimization [17].

The main abilities of the avatar are acceleration to a preset top speed, braking / deceleration, turning and changing the orientation of the avatar, jumps, shoot with a gun, and magnetic grip (creating a force of magnetic attraction on objects in the scene).

"Strengthening learning" methods were used to implement the strategy. The resulting model has been integrated into the game so that the strategy can be executed automatically, at the user demand.

CONCLUSION

Determining the optimal strategy of a game is not a trivial task, especially due to the enormous space of solutions. The paper describes the defining elements and components of a strategy and exemplifies the methodology by which the strategy of the game can be analyzed. The analysis consists in approaching the solutions for identifying, modeling, implementing, executing and evaluating the strategy of a game. Some examples highlight and explain the main problems of the proposed methodology.

Most of the examples were implemented and evaluated using the Unity technology.

REFERENCES

1. Nash, John, Equilibrium points in n-person games. Proceedings of the National Academy of the USA 36(1):48-49, 1950.
2. Catana M.C, Gorgan D., Analyzing Computer Game Strategies through Visual Techniques. Proceedings of the *12th Romanian Conference on Human-Computer Interaction*, ISSN 2344-1690, pp.41-48, 2015.
3. Hsieh J.-L. and Sun C.-T., Building a player strategy model by analyzing replays of real-time strategy games, in *2008 IEEE International Joint Conference on Neural Networks* (IEEE World Congress on Computational Intelligence), 2008, pp. 3106–3111.
4. Andersen K. T., Zeng Y., Christensen D. D., and Tran D., Experiments with online reinforcement learning in real-time strategy games, *Applied Artificial Intelligence*, vol. 23, no. 9, pp. 855–871, 2009. [Online]. Available: https://doi.org/10.1080/08839510903246526.
5. Amato C. and Shani G. High-level reinforcement learning in strategy games. In *AAMAS*, volume 10, pages 75–82, 2010.
6. Bugeja K., Spina S, and Buhagiar F., Telemetry-based optimisation for user training in racing simulators, 2017 9th *International Conference on Virtual Worlds and Games for Serious Applications* (VS-Games), pp.31-38, 2017.
7. Wang Y., He H., and Sun C., Learning to navigate through complex dynamic environment with modular deep reinforcement learning. *IEEE Transactions on Games*, 10(4):400–412, 2018.
8. Han, Y., Zhou, Q., and Duan, F., A game strategy model in the digital curling system based on NFSP. *Complex and Intelligent Systems, 8(3)*, 1857-1863, 2022.
9. Qu, D., Zhang, K., Song, H., Jia, Y., and Dai, S., Analysis and Modeling of Lane-Changing Game Strategy for Autonomous Driving Vehicles. *IEEE Access, 10*, 69531-69542, 2022.
10. Yuan, Y., Deng, Y., Luo, S., and Duan, H., Distributed game strategy for unmanned aerial vehicle formation with external disturbances and obstacles. *Frontiers of Information Technology & Electronic Engineering, 23(7*), 1020-1031, 2022.
11. Kuang, Z., Shi, Y., Guo, S., Dan, J., and Xiao, B., Multi-user offloading game strategy in OFDMA mobile cloud computing system. *IEEE Transactions on Vehicular Technology, 68(12)*, 12190-12201, 2019.
12. Chen, B. S., Chen, W. Y., Yang, C. T., and Yan, Z., Noncooperative game strategy in cyber-financial systems with Wiener and Poisson random fluctuations: LMIs-constrained MOEA approach. *IEEE Transactions on Cybernetics, 48(12*), 3323-3336, 2018.
13. Qilong, S. U. N., Naiming, Q. I., Longxu, X. I. A. O., and Haiqi, L. I. N., Differential game strategy in three-player evasion and pursuit scenarios. *Journal of Systems Engineering and Electronics, 29(2)*, 352-366, 2018.
14. Tan, J. L., Lei, C., Zhang, H. Q., and Cheng, Y. Q., Optimal strategy selection approach to moving target defense based on Markov robust game. *Computers & Security, 85*, 63-76, 2019.
15. Kang, H. N., Yong, H. R., and Hwang, H. S., An Analysis of Game Strategy and User Behavior Pattern Using Big Data: Focused on Battlegrounds Game. *Journal of Korea Game Society, 19(4)*, 27-36, 2019.
16. Li Y., Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274, 2017.
17. Schulman J., Wolski F., Dhariwal P., Radford A., and Klimov O., Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.