2017 21st International Conference on Process Control (PC)
June 6–9, 2017, Štrbské Pleso, Slovakia

Computer game as a tool for machine learning education

Petr Dolezel
Faculty of Electrical
Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
Email: petr.dolezel@upce.cz

Miroslav Dvorak
Faculty of Electrical
Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
Email: miroslav.dvorak@student.upce.cz

***Abstract*—The effective education is one of very important and everlasting challenges of human society. With each generation of students, new approaches have to be implemented to keep the process of education prosperous. This paper introduces a small piece to a set of modern tools for education of Informatics and Electrical Engineering. To be more specific, an interactive software for machine learning testing and demonstration is presented in this paper. The software is designed especially to be used as a motivation and a first encounter to these areas of technical studies, while it supports individual efforts of the students. In the paper, the software architecture is described and, in the second half of the paper, some possibilities of software usage in education process are suggested.**

I. Introduction

Education is a process of sharing of knowledge and experience. It is also the process of creating and transforming of moral features and attitudes. Therefore, it is a part of socialization.

Experiments have proven to be effective means of teaching, especially for their higher level of attractiveness and interactivity in comparison to other means of education. While teaching using experiments is well implemented in some areas of education, other branches would rather use different approaches.

One of the latest issues of education is the shift in leisure time activities of students. Especially in the area of Informatics and Electrical Engineering, the students are often and frequently (like every five minutes or more) distracted by the reminders and requests on their mobile phones and tablets; either it is the message from some farming mobile phone game or a new post on a selected social network.

Along with these behavior changes, new approaches of education in the area of Informatics and Electrical Engineering are being implemented. Traditional face-to-face learning is replaced by e-learning [1], [2], laboratories are being rearranged to be accessed online [3], [4], these days even exams are prepared to be passed online [5]. All these new approaches are designed for students to organize their studies independently with a very low linkage to other students or the institution.

In conformity with the mentioned trends, this contribution provides a comprehensive tool for teaching of artificial neural networks and soft computing related courses. To be more specific, a software for machine learning testing and demonstration is presented in following sections. While trying to catch a student's eye for more than five minutes, the tool is based on a classical and wide-spread computer game called Achtung - die Kurve [6]. This approach has already proved to be effective [7]. Despite the simplicity of the game, it has drawn a lot of attention due to its undeniable addictiveness. The game mechanism is as follows: the players leave a trail while moving, and try to make the opponent hit a wall or a trail. The last surviving player wins. Thus, some level of tactics as well as dexterity has to be displayed by the player to be successful.

The human player can be clearly replaced by an artificial entity, which is the main point of this contribution. We introduce a comprehensive tool to design an artificial player into the game called Achtung - die Kurve. The tool is meant to be used by students to get some basics of machine learning, artificial neural networks, deep learning and similar topics in an enjoyable and interactive way. Matlab is chosen as the programming language, since it is probably the most widespread tool for technical computing, used not only by coders, but also by many engineers, academicians and students. The work presented here is linked to the previous authors' works [8] and [9].

The text is organized as follows. The tool architecture is described comprehensively in Section II. Then, several illustrative examples of tool usage in education are suggested, and the paper is finished with conclusions. The tool is available on a website [10].

II. Software architecture

At the beginning of the game called Achtung - die Kurve, each player starts at a random spot on the playing field. After the game is started, all players move with a constant speed, having the ability to turn left or right, although the turning speed is limited. As the player moves across the playing board, it draws a permanent, solid line. When the dot collides with any section of the line or the boundary of the playing field, the player instantly loses, although the line remains in the playing board until the end of the game. The player may try to draw barriers to block the path of other players, forcing them into

978-1-5386-4011-1/17/$31.00

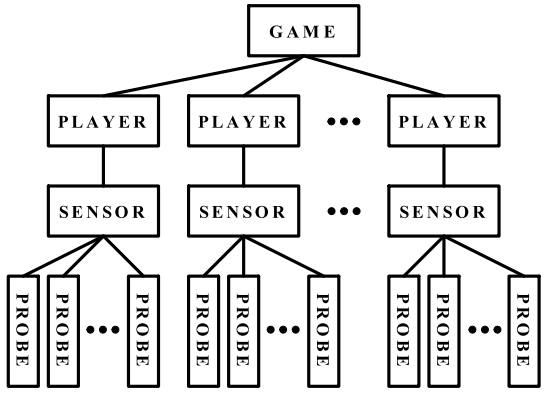

Fig. 1. Diagram using unified modeling language.

a collision. The game is won when all but one of the players have collided [6].

The following comprehensive tool is proposed to design an artificial player to the game. It consists of several classes, each corresponding to some functional block of the game.

- *Game* - an instance of this class controls the whole game including a graphical output.
- *Player* - instances of this class represent the players of the game.
- *Sensor* - an instance of this class is used by the player to inspect the surroundings and to find possible obstacles.
- *Probe* - each sensor uses several probes to determine the obstacles in various directions.

The diagram, which describes the software architecture, is shown in Fig. 1.

The game is performed by creating the instance of *Game* and iterative calling of its method *performStep()*. In the following list, the important properties and methods of all the used instances are described. See the documentation of the tool for the complete list.

A. Game

Using this tool, only one instance of the class *Game* is initialized. It uses the following *properties*.

- *players* - a cell array of all the players.
- *board* - a two-dimensional array which represents the playing field. Zeros are free cells, while non-zeros indicate obstacles.
- *draw* - a boolean property, which allows the graphical visualization of the game.
- *alive* - a boolean property, which indicates, that at least one of the players is alive (the game is not over).

Apart from the *properties*, it utilizes several *methods*.

- *Game()* - constructor.
- *performStep()* - this method performs one step of the game - each player moves one step ahead.
- *testEnd()* - a method, which tests the possibility of ending the game - all players are dead.
- *countSteps()* - this method returns the number of the steps performed in the game.
- *getRank()* - this method returns the rank of the players after the game is over.

B. Player

The instances of *Player* are created by the instance of *Game*. Some of the *properties* of those instances are listed below.

- *direction* - direction of the player, real value between 0 and 2π.
- *ordAct* - two values array with the current coordinates of the player within the board.
- *game* - the instance of the currently used game.
- *sensor* - the instance of *sensor*, which detects the obstacles around the player.
- *decider* - the instance of a class *Decider*. This class should be written by the user. It should implement the method *decide()*, which applies the decision rule according to the information from *sensor*.

The relevant *methods* are listed below.

- *Player()* - constructor.
- *performStep()* - this method performs one step of the player.
- *testLife()* - a method, which tests whether the player is dead or alive.
- *getStepsNumber()* - this method returns the number of the steps performed by the player.

C. Sensor

Each instance of *Player* includes one instance of *Sensor*. The *properties* of the instance are listed below.

- *player* - the player, which uses the sensor.
- *number* - the number of probes used by sensor.
- *game* - the instance of the currently used game.
- *range* - the range of the probes.
- *angle* - operating angle of the sensor.

Apart from the constructor, *sensor* implements only one method.

- *Sensor()* - constructor.
- *measure()* - this method returns the distances of obstacles detected by every probe.

D. Probe

An instance of the class *Probe* is initialized in the method *measure()* of the instance of *Sensor*. After the initialization, it moves in the constant direction till it reaches the obstacle or the maximum number of steps (*range*). In the list below, some important *properties* are shown.

- *direction* - direction of the probe, real value between 0 and 2π.
- *game* - the instance of the currently used game.
- *stepsNumber* - number of steps performed to reach the obstacle.

Probe implements the following methods.

- *Probe()* - constructor.
- *performStep()* - this method performs one step ahead of the probe.

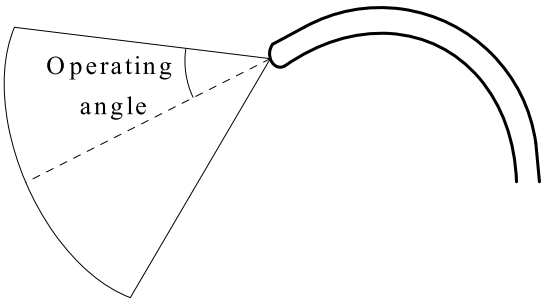

Fig. 2. Operating angle of player.

TABLE I
EXPERIMENT 1

Feature	Value
Board width	320
Board height	200
Number of players	1
Sensor.range	10
Sensor.angle	$0.125\pi; 0.175\pi; 0.25\pi; 0.5\pi; 0.75\pi; 0.9\pi$
Cost function	Number of steps till death

- *getStepsNumber()* - the method returns the number of steps performed by the probe before it reaches the obstacle.

III. EXAMPLES AND DEMONSTRATIONS

Several illustrative examples are performed and described here. This section should suggest the various usages in teaching. Note that a multilayer feedforward neural network is used as *decider* here. However, many paradigms can be applied, including expert system, support vector machine, decision tree and others. In addition, most of the mentioned paradigms have to be optimized using a defined learning algorithm or search technique. In our examples, differential evolution is applied. However, other stochastic techniques (genetic algorithm, simulated annealing, evolutionary strategy, ...) could be implemented, too.

The following examples are described only in a generalized way, since the purpose is to demonstrate the possibilities of the tool, rather than to prove the optimal solutions.

A. Issue 1 - Operating angle of sensor

One issue is to determine the suitable operating angle of the sensors. As described above, the operating angle can be set by the user and the behavior of the player is significantly related to it. Graphically, the issue is depicted in Fig. 2. To determine the suitable operating angle value, the experiment can be prepared as seen in Table I.

After optimization of *decider*, the behavior of players as shown in Fig. 3 is achieved. The differences in behavior of various players are clear. Considering only the number of steps reached, the suitable value of *Sensor.angle* is 0.9π.

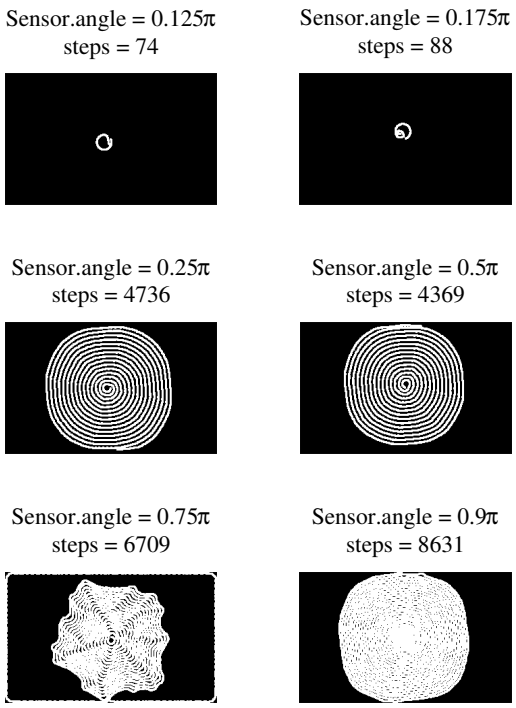

Fig. 3. Results of Issue 1.

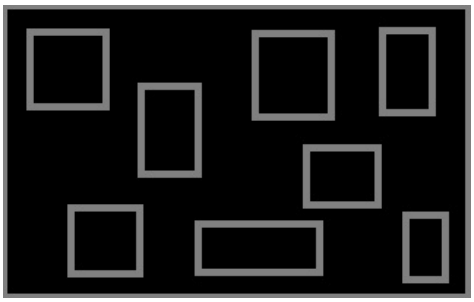

Fig. 4. Maze for player.

B. Issue 2 - Obstacle avoidance

Another issue can be defined to prepare a proper experiment for obstacle avoidance or maze solving. It is necessary to provide a board with obstacles drawn on it. The example can be seen in Fig. 4. The parameters for the experiments are defined by Table I, while *Sensor.angle* is set to 0.9π.

After several learning procedures, the resulting players are designed - see Fig. 5.

C. Issue 3 - Confrontation

The main purpose of the tool is to design a player which will beat other players in the game. The intuitive experiment can be arranged as seen in Table II.

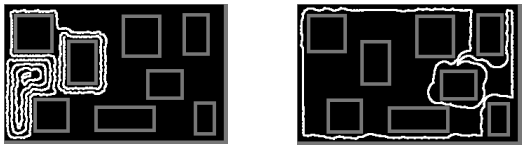

Fig. 5. Results of Issue 2 - two attempts with random initial conditions.

TABLE II
EXPERIMENT 3

Feature	Value
Board width	320
Board height	200
Number of players	4
Sensor.range	10
Sensor.angle	0.9π
Cost function	Number of steps of player 1 till death

Apparently, player 1's *decider* is optimized while the other players are controlled by *deciders* gained in previous experiments. The problem of this setting is in its cost function. Although there are four players in the game, the cost function does not force them to play belligerently. They just try to avoid each other - see Fig. 6.

Therefore, the cost function can be modified to appreciate the aggressiveness of the player. Thus, instead of the number of steps till the death, the rank of the player is used as the cost function (rank = 1 if the player dies as the first one, rank = 4 if the player dies as the last one). The resulting behavior of the player is shown in Fig. 7, where the suitable situation is postured. It is obvious that after detecting the nearby opponent, the player tries to cross its path. However, this feature is apparent only in these specific situations. In whole figure, the behavior shift is not so obvious - see Fig. 8.

D. Issue 4 - Competition

As students are naturally competitive beings, it is convenient to organize a competition of students' works. For an example,

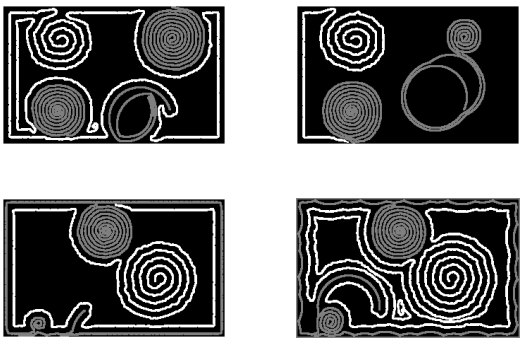

Fig. 6. Results of Issue 3 (Player 1 is white) - four games with random initial conditions.

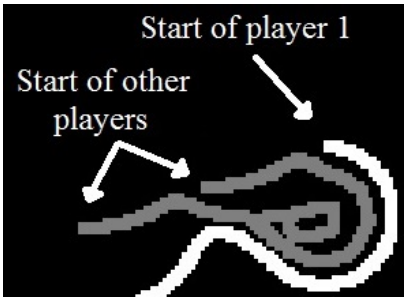

Fig. 7. Results of Issue 3 (Player 1 is white) - detail of aggressive behavior.

Fig. 8. Results of Issue 3 (Player 1 is white) - aggressive behavior.

a competition between the players designed in previous issues is prepared - player 1 is designed in issue 1, player 2 in issue 2, player 3 is the non-aggressive player from issue 3 and player 4 is the aggressive player from issue 3. A hundred of matches are arranged with random initial conditions and the players are evaluated in the following way: the winner of each match gains four points, the second one gains three points, the third one gains two points and, finally, the last one gains one point. The final numbers are summarized in Fig. 9. The quaint observation is the highest rank of the player one, which was designed according to the simplest approach. On the other hand, the aggressive player got the lowest number of points. However, the absolute numbers are quite similar, which indicates, that the rank of the players is very relevant to initial conditions.

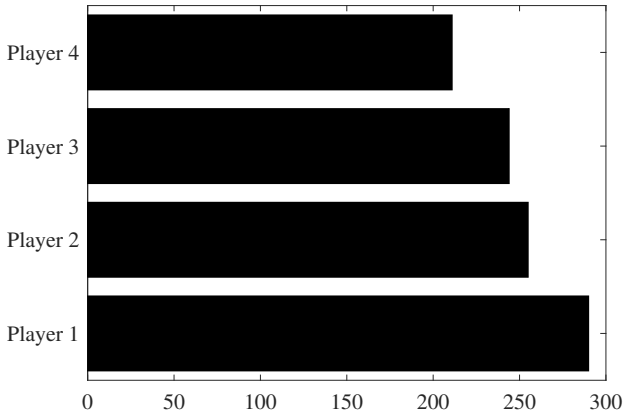

Fig. 9. Results of competition.

IV. Conclusion

A software tool, which can be used for teaching and training of machine learning, artificial neural networks, deep learning and similar topics, has been developed and presented in this paper.

The presented tool is clear, highly enjoyable and based on the wide-spread Matlab software. The architecture of the tool is straightforward and naturally structured. As demonstrated in section III, the usage of the tool is simple, descriptive and entertaining. In addition, the tool is licensed as an open source, which means, that any user can adjust its functionalities freely.

Acknowledgment

The work has been supported by the Funds of University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

References

[1] J. Ligusova, J. Ligus, and I. Zolotova, "Cybernetic education centre," in *2013 24th EAEEIE Annual Conference (EAEEIE 2013)*, May 2013, pp. 133–138.

[2] M. Watfa, "Cloud computing and e-learning: Potential pitfalls and benefits," in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, Aug 2016, pp. 140–144.

[3] M. Kaluz, J. Garcia-Zubia, M. Fikar, and L. Cirka, "A flexible and configurable architecture for automatic control remote laboratories," vol. 8, no. 3, pp. 299–310, July 2015.

[4] M. Kaluz, L. Cirka, and M. Fikar, "Virtual and remote laboratories in education process at FCFT STU," in *2011 14th International Conference on Interactive Collaborative Learning*, Sept 2011, pp. 134–139.

[5] X. F. Li, J. H. Wang, and W. W. Gao, "Examination system in the cloud computing platform based on data mining," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Dec 2013, pp. 1605–1608.

[6] (2016) Achtung, die kurve - wikipedia page. [Online]. Available: https://en.wikipedia.org/wiki/Achtung,_die_Kurve!

[7] B. R. C. Marques, S. P. Levitt, and K. J. Nixon, "Video games as a medium for software education," in *2012 IEEE International Games Innovation Conference*, Sept 2012, pp. 1–4.

[8] P. Dolezel, M. Mariska, I. Taufer, and L. Havlicek, "Artificial neural network promotion," in *2013 International Conference on Process Control (PC)*, June 2013, pp. 214–218.

[9] M. Mariska and P. Dolezel, "Agent-based software framework supporting fast modeling of service systems for repast simphony 2," in *Mendel*, June 2013, pp. 415–420.

[10] (2017) Tool for experiments with machine learning. [Online]. Available: https://www.researchgate.net/publication/313817405_Tool_for_experiments_with_Machine_Learning