# TEACHING INTELLIGENT VIRTUAL AGENTS PROGRAMMING THROUGH SIMULATED CHILDREN'S GAMES

Jakub Gemrot, Martin Černý, Cyril Brom
Faculty of Mathematics and Physics,
Charles University in Prague,
Malostranské náměstí 25, 118 00, Prague 1,
Czech Republic
E-mail: gemrot@ksvi.mff.cuni.cz, cernym@gmail.com, brom@ksvi.mff.cuni.cz

**KEYWORDS**
Educational Games, AI Education, Game AI.

**ABSTRACT**

Developing intelligent virtual agents (IVAs) is a great challenge for computer programmers. Lifelike virtual environments present various obstacles, especially on the lower AI level. Navigating through 3D worlds is notoriously difficult to handle properly as well as quick and appropriate reactions to rapid changes of the environment. When teaching basics of IVA development at our university, we noticed that students find the complexity of virtual environments intimidating. Although lectures on AI theory help, a substantial amount of hands-on experience is indispensable to gain proficiency. We thus searched for ways to start with very simple tasks and at the same time keep the students engaged and motivate them to experiment with AI development at home. In this paper we report on two 3D virtual environments we developed on top of Unreal Tournament 2004 for the introductory classes of our course.
The environments are inspired by children's games and are focused primarily on combining simple high-level decisions with navigation. Tournaments of bots were held for both environments to conclude parts of the course. Evaluation over two years of the course shows that the environments helped students to focus on subparts of the IVA development and that the tournaments motivated the students to experiment with IVA behaviors outside the borders of the course.

**INTRODUCTION**

As the virtual entertainment industry grows, there is an increasing demand for education in game development and related fields. One of the very interesting areas in game development is AI and development of intelligent virtual agents (IVAs) in particular. In this context we run an IVA development course at our university since 2005 for a mix of undergraduate and postgraduate students. The course focuses on the modeling of IVA behaviors in complex 3D environments (partially observable, dynamic, continuous, non-deterministic, multi-agent) from both theoretical and practical perspective. The focus of the course is not on classical symbolic AI (A*, planning, etc.) as this is covered by prerequisite AI courses, but rather on reactive reasoning and "intelligence without representation". The theoretical part covers a broad range of topics from neuroscience and psychological background to reactive planning methodologies. In the practical part of the course, students develop bots for deathmatch mode of Unreal Tournament 2004 (Epic Games Inc. 2004).
While the course was relatively successful, we noticed that the practical part of the course was overly challenging for the students who often felt lost and frustrated. One of the most problematic areas was navigation through the 3D environment and other low-level tasks related to geometry of the environment. This was intentional in a way, because our experience indicates that it is those difficulties involved in the 3D worlds that make game AI both hard and unsuitable for classical AI approaches. But we recognized the need for environments that would retain the delicate intricacies of 3D but would accept simple scenarios and require less sophisticated higher level reasoning. Such environments will let the students progress with smaller but quicker steps and experience a sense of success and reward more often.
Furthermore, we knew that as in other areas of software development, practical experience — a lot of practical experience — is indispensable. Even more so for virtual environments, where even things that seem very easy at first sight (e.g. navigating correctly in the environment) often introduce unexpected difficulties and the actual implementation is at least as important as the general idea for the overall success of the behavior. We thus wanted to motivate students to experiment with the AI at home, as the time allocated for classes was far from sufficient to gain proficiency.
To summarize, we wanted to follow the long-known pedagogical principles: start with the simple; learning should be fun; practical experience promotes learning (Comenius 1648).
We sought inspiration in the way humans learn to navigate seamlessly in the real world. We realized that children master movement in the real world by playing simple games where fast movement in the environment is vital for victory. We found those games to be a great source of inspiration. They have simple rules and most of the students already know the rules, simple behavior is sufficient to achieve reasonable results and still there are plenty of possibilities to improve over the simple approach.
In this paper we report on our use of virtual counterparts of children games to teach navigation and basics of reactive decision making. We chose two children games: Tag! and Hide & Seek. We implemented the games in UT 2004 and used them as test environments during the introductory part of our course. To further motivate students to play with the

AI at home, we introduced non-obligatory tournaments of bots in both games. We report on feedback from the students and the implications of using similar games in course curriculum.
The rest of the paper is organized as follows: First, we discuss related work on teaching AI and programming in general, than we briefly detail the curriculum of our course and motivate our use of UT2004 in the classroom followed by the discussion of requirements we imposed on the environments and details of the individual environments. Finally, we report on the evaluation of the games during two years of the course.

## RELATED WORK

Multiple approaches were tried to increase engagement in general computer science/programming courses. Insights from general pedagogic research are transferred to computer science education e.g., (Lockwood and Esselstein 2013, Porter and Simon 2013, Kafai et al. 2013). These efforts are orthogonal to our research as our educational games can be incorporated in virtually all teaching methodologies. Gamification of the course was proposed (Decker and Lawley 2013) — similarly to this approach, we have implemented a flexible grading system that allows students to score points for various activities. Developing games as part of a class has also been suggested (Bayzick et al. 2013).
Bayliss (2009) shows that games have been be successful in both attracting, motivating and retaining students of computer science. It is noted that correctly chosen open-ended assignments stimulate creativity and let students "play" with the task. She also reports on caveats of the approach, including the high requirements on the teacher side and possible technical problems with game technology.
In the context of AI, educational scenarios based on Pac-Man and other simple game environments (DeNero and Klein 2010, McGovern et al. 2011, Bezakova et al. 2013) have been proposed. Those are, however not applicable to our case as they focus on classical AI techniques and do not involve an environment comparable in complexity to 3D computer games.

## COURSE CLASS DETAILS

The course lectures cover various topics related to IVA development: reactive planning, subsumption architecture (Brooks 1991) , behavior oriented design (Bryson 2001), steering, evolutionary algorithms, neural networks, background in ethology, neuroscience, psychology and psychophysics; belief-desire-intention architecture (Georgeff et al. 1999) and multiagent systems. We have reported on the curriculum of the course lectures in more detail in (Brom 2009). In the following text, we focus on the practical classes we have developed and evaluated since.
To provide students with hands on experience in IVA development we have created Pogamut (Gemrot et al. 2009) — a platform for prototyping of bots' behaviors for Unreal Tournament 2004 (UT2004) (Epic Games Inc. 2004) in the Java programming language. UT2004 is first-person shooter (FPS) that was very popular in the 2000s. Even though an older game today, the graphics of UT2004 still appeal to students and the game complexity does not differ from its sequel Unreal Tournament 3 (Epic Games Inc. 2007) or other recent FPS games.
During the practical classes, students are taught how to hierarchically decompose behaviors using behavior oriented design (BOD) in a top-down manner and then implement the behavior on top of Pogamut platform using a bottom-up approach.
Respective practical classes are focusing on different technical aspects of the Pogamut platform, teaching students only a limited set of behavior primitives (sensors and effectors) available to bots every class, allowing students to gradually explore different aspects of UT2004 bot behaviors.
The task of the students is to implement simple behaviors using the newly learnt primitives and to incorporate them in a bot they incrementally create. As the set of behavior primitives grows, students are able to construct more complex behaviors, starting with a simple follow-me-bot to a bot covering all the aspects of a deathmatch game.
Practical classes cover the following behavior aspects: low-level movement, environmental reasoning, navigation, item collection, combat and team work.
The ultimate objective of the practical classes is to teach students how to structure IVA behaviors for game-like tasks within UT2004 environment. The final proof of their ability to do so is a successful implementation of a death-match (DM) bot that is able to beat less-skilled human players.
Students are graded based on points they can get for multiple types of activities. Those include: attendance, homeworks, quick tests in class and several optional tournaments of bots throughout the semester.

## REQUIREMENTS AND ANALYSIS

The initial course runs comprised of classes that focused on the DM mode too much. We were teaching students how to incrementally build their bots by introducing new behavior aspects that students were adding into an existing one. Even though we tried to mask as much complexity as we could in the Pogamut platform, AI for UT2004 simply needs to handle too many issues. There are numerous relevant sensory data messages UT2004 exports (21 messages, 189 attributes) that the bot has to handle as well as command messages the bot needs to use correctly (13 commands, 34 attributes). The bot has to reason about 10 weapon types and 17 item types that are available within the game.
Due to the high complexity, students could not create bots that would cope with at least the most important game aspects until late in the semester which was not very rewarding and it did not motivate students well for two reasons. Firstly, knowing that creation of DM behavior does not fit into single class and single homework, students were not experimenting with the behavior at home; they rather waited before we explained them all behavior aspects required to create DM bot and then experimented with the DM bot only once. Secondly, students had become easily bored as the ultimate task was the same for every class even though details differed. Therefore, we decided to restructure the classes and devise new sets of tasks that are assigned to students.

From the teaching point of view, a task is characterized with knowledge (K) that is needed to solve the task and needs to be taught beforehand and with the logic (L) that the student should discover himself while solving the task. The knowledge can be further divided into three parts:

K1) the environmental mechanics involved in the task,
K2) set of behavior primitives (e.g. move, shoot, see-player) and/or higher-level actions provided by the platform (e.g. navigation) required by the solution and how to use them,
K3) reasoning techniques (e.g., non-trivial use of A*) required to solve the task.

To create the behavioral logic, a student should undergo following three steps:

L1) analyze the task,
L2) design the behavior structure,
L3) implement the behavior using the platform.

The (complicated) structure of knowledge and logic required for a deathmatch bot is shown in Fig. 1.

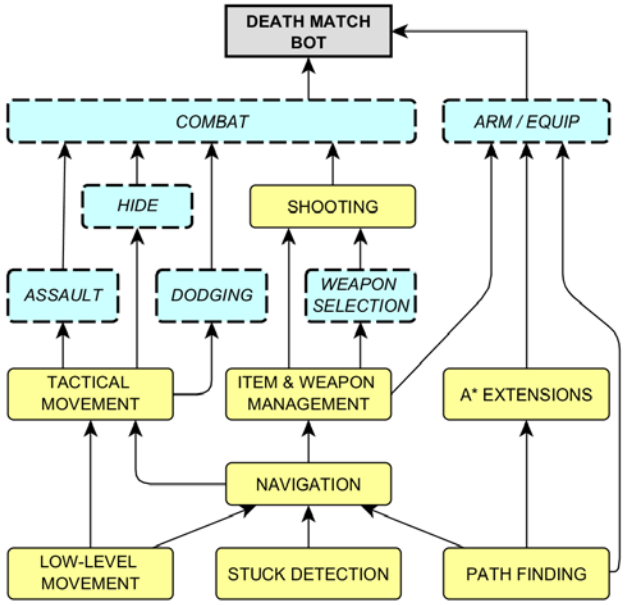


Figure 1: Structure of knowledge (full lines) and behavioral logic (dashed lines, italics) required for DM bot and their classroom dependencies.

The challenge is to choose the proper set of tasks. The tasks should be of increasing complexity and connected with each other so that students can consolidate their knowledge and skills by reusing them in portions of the more complex tasks. The task also needs to allow for incremental buildup of the necessary knowledge and that lets the students to complete all logic development steps in reasonable time so that they stay focused and motivated. The knowledge and logic design for a task should be dealt with in the same class or in close succession: Separating the knowledge from its use in agent logic leads to poor learning performance as the theory is no longer supported by practice and separating the individual logic development steps prevents the student from getting immediate feedback on the quality of his design (e.g. wrong analysis may not be spotted until the student fails at implementing it).
We noted that the students struggled the most with the very introduction to the platform and with navigation and movement. Moreover, once classes on those topics were over, only few steps remained to their first attempts at DM bot — although there was still a lot of knowledge to master, students were already familiar with the overall design and philosophy of the Pogamut platform and thus progressed faster.
To conclude, we needed to devise tasks that would cover the basics of Pogamut and navigation and would need no other knowledge.

## THE ENVIRONMENTS

Based on the requirements identified in the previous section, we have designed two environments inspired by children's games. Both environments were run in UT 2004 and the game logic was implemented as an extension to the Pogamut platform.

### Tag! Game

Tag! is inspired by classical children game, where one player is the "chaser" and tries to catch other players (labeled here as "evaders") by touching them. Once a player is caught, the chaser role is passed to the caught player. The former chaser becomes an evader, but a "no-tags-back" rule is enforced: the former chaser is immune (cannot become chaser again) until the role is passed to yet another player.

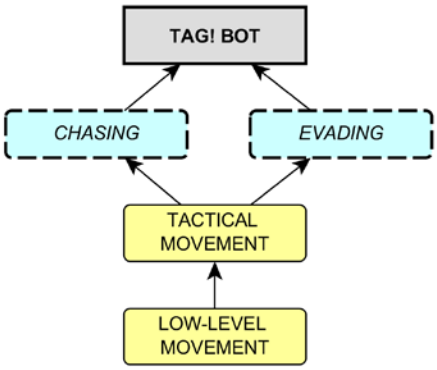


Figure 2: Structure of knowledge (full lines) and behavioral logic (dashed lines, italics) required for Tag! bot and their classroom dependencies.

The game has many interesting properties. A) The simplest strategy for chaser resp. evader is to run directly to chosen evader resp. run directly away from the chaser, therefore students can create simple Tag! bots very quickly. B) The game works well with a very simple environment (e.g. flat square or rectangle), therefore students do not need to be taught about environment representation and navigation. C) Having move, dodge and jump commands is enough to create different chasing and evading strategies (even in the simple environments), thus it provides room for students'

creativity. D) The Tag! Game can be scored (how many times a bot has become the chaser, how fast a bot can pass the chaser role on) and therefore it is possible to conduct tournaments between student bots.
For reasons above, the Tag! Game makes a good candidate for the first real task for students. Technicalities required to solve the task (K1-3) can be explained quickly (45 minutes) along with coding the simplest Tag! bot implementation (L1-L3) together with students (45 minutes).
We played Tag! with four bots --- this is the minimum to allow for complex strategies to be taken: the chaser always has two possible targets (the last evader is immune to him). Since all bots move at the same speed, "tactical movement" (computing move vectors based on positions of other bots and other context) is vital factor of success. The classroom dependencies of Tag! bot are shown in Fig. 2.
The students are shown basic vector math hints and asked to create "tactical movement" separately for the chaser and evader roles. Even though a simple game, students are very creative at this part. Interestingly, the move, dodge and jump commands create a very large space of strategies and counter-strategies. One of the keys to success is the ability to predict future positions of the other bots, which is especially beneficial to the chaser who may "cut corners" to catch the evader more quickly. On the other hand a bot may decide to exploit the opponent's prediction mechanism and gain advantage by behaving unpredictably.
For instance, the bot can speed its running using dodges and jumps. However, it cannot jump too frequently as the bot cannot control its movement while in the air, therefore the opponent can reliably predict the bots position as soon as it notices that the bot is in the air. While evading, the evader can try to run smoothly in circles, which creates an endless evading behavior a simple "direct running chaser" cannot beat. It is also beneficial to implement timeouts to chasing behavior or chase only those bots one has successfully tagged before.
The bots are not aware of other bots that they do not see directly — they only know the location where they were seen for the last time — so improvements can be made by controlling the direction of bot's gaze to maximize the amount of information available for decision making. Tag! also serves as a good exercise of basic vector math, which is necessary for many more complex decisions in 3D environments.

**Hide & Seek Game**

Another classical children game is Hide & Seek, where one player is the seeker and tries to seek out others that are hiding within the environment (labeled here as hiders). We have implemented the variant played most commonly in Czech Republic: All players start at a designated base. The hiders are given a short time to hide, while the seeker is blindfolded (does not receive vision data and cannot issue commands). To score a point, seeker needs to find another player (see him) and then return to base to "ground" him. However, if a hider manages to reach the base before he is grounded, he scores a point and cannot be grounded anymore. The game ends once all hiders are grounded or have reached the base.

Whereas Tag! focuses on low-level movement, Hide & Seek focuses on the environment representation and reasoning (path-finding and line-of-sight) and navigation (path-following). Even though the game is more complex, it still retains interesting properties. A) The simplest strategy for a seeker is to run randomly around the environment until it spots a hider, then return to the base. Analogically, for a hider, the simplest strategy is to hide at a random place and try to reach the base as soon as hider finish counting. B) The environment reasoning complexity can be lowered by, again, designing a simple environment (e.g. grid-based 2D maze without rooms). C) The Hide & Seek Game can be scored (according to the number of found hiders or the number of escapes).
One of the maps we used is shown in Fig. 3.

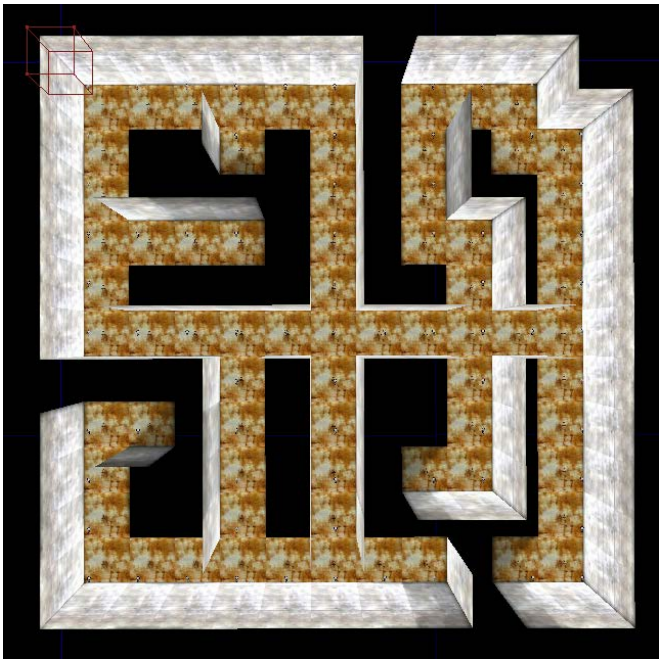

Figure 3: A sample map for Hide & Seek. The base is at the crossroads in the center of the map.

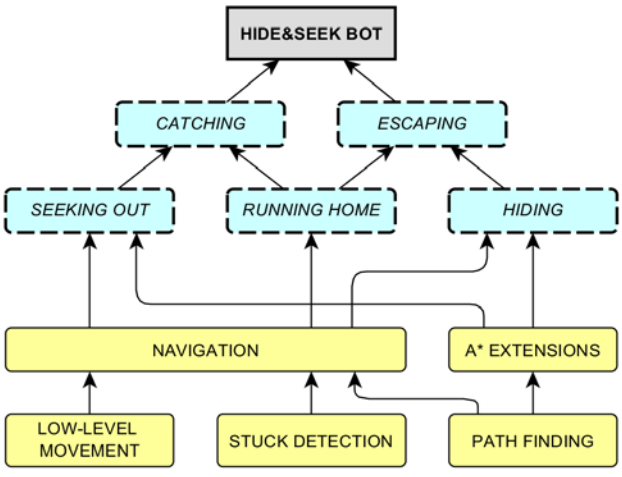


Figure 4: Structure of knowledge (full lines) and behavioral logic (dashed lines, italics) required for Hide & Seek bot and their classroom dependencies.

Table 1: Categories of qualitative feedback on tournaments held during the course and count of answers that belong to the category.

| Category & Sample Answers | Count |
|---|---|
| Good motivation for the homework<br>*The tournaments motivated me to try inventing unusual solutions to the problems.* | 6 |
| Creates competitive environment (in a good sense of the word)<br>*It was nice to see some results of my work on bots and have some comparison with other students.*<br>*It was an excellent idea to compare the bots. Everyone could see, what all can be achieved.*<br>*Some of the resulting videos were funny.* | 6 |
| Spice the course up<br>*Spiced the course up, nice to do a homework that is somewhat more practical.* | 5 |
| Interesting, but no time to compete<br>*Tournaments were interesting, but due to my other activities, I did not have a time to compete truly.* | 4 |
| Interesting, but not motivating<br>*Tournaments were interesting, but they did not motivate me really.* | 1 |
| Other<br>*Comments about possible improvements of tournaments.* | 5 |

It is more time demanding to teach technicalities that are required to solve the task. Firstly, the navigation along with environment representation and simple path-finding needs to be taught and practiced (90 minutes). Then, the class detailing A* implementation in Pogamut and options for its adaption is required in order to provide students with tools for obtaining multiple paths towards target or searching for paths that lead through places not visible to the seeker (90 minutes).

The classroom dependencies of Hide & Seek are shown in Fig. 4. Note that low-level movement was already taught in the Tag! scenario and that after students have understood the necessary prerequisites of Tag! and Hide & Seek bots, only item and weapon management and shooting need to be explained before they can start working on a death match bot (see Fig. 1).

Hide & Seek also provides a rich strategy space. In order to gain information about positions of other players, the seeker needs to roam away from the base, making it possible that a hider reaches the base safely. Hiders on the other hand may try to spot the seeker without being spotted and have to decide when there is a reasonable chance they will make it to the base. Thus in both roles, the bot needs to balance the risks and gains of its behavior.

## EVALUATION

The Tag! and Hide & Seek Games were introduced to the course curriculum in 2013 and received positive feedback. Therefore, the course ran during 2014 without changes. Here we present data from these two years from the total of 27 students (26 males, 1 female, Czechs) out of which 7 were undergraduate and 20 were postgraduate students. Importantly, the overall performance of students during the classes and the final exam improved significantly. The final exam was almost the same for the last three runs of the course and involved coding of two complex behaviors in a lab. In 2012, the average time to complete the first behavior was 2 hours, 50 minutes (sd: 30 minutes) and only two students completed the second behavior. In 2013 and 2014, the average for first task dropped to 1 hour, 29 minutes (sd: 31 minutes). All students also finished the second behavior, on average in 3 hours, 15 minutes (sd: 47 minutes). Although multiple factors may be involved (most notably innovations to Pogamut platform and prior knowledge of the tasks gained from students from previous year), the results are encouraging.

Data about tournaments was gathered through questionnaires that were part of the final exam of the course.

Students were asked three quantitative questions related to the tournaments held throughout the course[1]:

1. Did you find tournaments (organized during the practice lessons) interesting? (11-Likert like scale, 0 - not at all, 5 - somewhat interesting, 10 - very interesting). The average score was 8.5 (sd: 1.69).
2. Did you put extra effort into homeworks that were used for tournaments? (11-Likert like scale, 0 - not at all, 5 - some effort, 10 - I have tried my best) The average score was 6.5 (sd: 1.6).
3. How many extra hours have you invested into doing your homework for a single tournament (at average)? The average was 4.25 hours (sd: 1.51).

Qualitative feedback ("Give any comments on the tournaments") answers could have been clustered to categories displayed in Table 1. The qualitative answers were very positive, with only one student that explicitly stated that he was not motivated by tournaments and over 60% of students explicitly expressing positive impacts on motivation.

## CONCLUSIONS

We have presented two educational games suitable for teaching basics of IVA development. We have shown that tournaments in both games helped to motivate students to spend extra hours (over 4 on average) working on the assignment. The games themselves were instrumental in keeping students' attention and helped them learn how to

[1] Tournament results including all compiled bots, UT2004 replays as well as some videos can be downloaded from: pogamut.cuni.cz/pogamut-devel/doku.php?id=human-like_artifical_agents_2013-14_summer_semester

navigate agents in virtual environments, reason about the environment and use vector math to calculate trajectories.
As a future work we intend to further ease development of bots for both games and start tournaments for high school student interested in game AI.

## ACKNOWLEDGEMENTS

Human data were collected with APA principles in mind. This research is supported by the Czech Science Foundation under the contract P103/10/1287 (GACR), by student grants GA UK No. 655012/2012/A-INF/MFF and 559813/2013/A-INF/MFF. This research is partially supported by SVV project number 267 314.

## REFERENCES

Bayliss J.D., 2009. "Using Games in Introductory Courses: Tips from the Trenches". In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '09, 337-341.

Bayzick J.; Askins B.; Kalafut S.; and Spear M., 2013. "Reading Mobile Games Throughout the Curriculum". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 209-214.

Bezakova I.; Heliotis J.E.; and Strout S.P., 2013. "Board Game Strategies in Introductory Computer Science". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 17-22.

Brom C., 2009. "Curricula of the course on modelling behaviour of human and animal-like agents". In *Proceedings of the Frontiers in Science Education Research Conference*. 71-79.

Brooks R.A., 1991. "Intelligence Without Representation". *Artificial Intelligence*, 47, 139-159.

Bryson J.J., 2001. *Intelligence by design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agent*. Ph.D. thesis, MIT, Department of EECS, Cambridge, MA.

Comenius J.A., 1648. Novissima Linguarum Methodus. Publisher unknown, Leszno, Poland, chap. Methodi linguarum novissimae fundamentum, Ars Didactica.

Decker A. and Lawley E.L., 2013. "Life's a Game and the Game of Life: How Making a Game out of It Can Change Student Behavior". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 233-238.

DeNero J. and Klein D., 2010. "Teaching introductory artificial intelligence with pac-man". In *Proceedings of the Symposium on Educational Advances in Artificial Intelligence*.

Gemrot J.; Kadlec R.; Bída M.; Burkert O.; Píbil R.; Havlíček J.; Zemčák L.; Šimlovič J.; Vansa R.; Štolba M.; Plch T.; and Brom C., 2009. "Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents". In F. Dignum; J. Bradshaw; B. Silverman; and W. Doesburg (Eds.), *Agents for Games and Simulations*, Springer-Verlag, LNCS 5920. 1-15

Georgeff M.; Pell B.; Pollack M.; Tambe M.; and Wooldridge M., 1999. "The belief-desire-intention model of agency". In *Intelligent Agents V: Agents Theories, Architectures, and Languages*, Springer. 1-10.

Kafai Y.; Griffin J.; Burke Q.; Slattery M.; Fields D.; Powell R.; Grab M.; Davidson S.; and Sun J., 2013. "A Cascading Mentoring Pedagogy in a CS Service Learning Course to Broaden Participation and Perceptions". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 101-106.

Lockwood K. and Esselstein R., 2013. "The Inverted Classroom and the CS Curriculum". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 113-118.

McGovern A.; Tidwell Z.; and Rushing D., 2011. "Teaching introductory artificial intelligence through java-based games". In *AAAI Symposium on Educational Advances in Artificial Intelligence*, North America.

Porter L. and Simon B., 2013. "Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1". In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '13, 165-170.

## WEB REFERENCES

Epic Games Inc., 2004. "Unreal Tournament 2004". www.unrealtournament.com. Last checked: 2014-05-25.

Epic Games Inc., 2007. "Unreal Tournament 3". www.unrealtournament.com. Last checked: 2014-05-25.

**JAKUB GEMROT** was born in Havířov, Czech Republic and went to the Charles University in Prague where we studied artificial intelligence and graduated in 2009. After the study he participated in several game development projects as an AI consultant, the most notably in the upcoming RPG Kingdom Come: Deliverence. He is one of the main authors of the Pogamut framework, which he actively maintains, and is currently finishing his doctoral thesis on controlling of intelligent virtual agents.