Designing an Educational Online Multiplayer Game for learning Programming

Konstantinos Maragos[1], Maria Grigoriadou[2]

[1]Informatics & Telecommunication Dpt., Panepistimiopolis, Athens, Greece, kmaragos@di.uoa.gr

[2]Informatics & Telecommunication Dpt., Panepistimiopolis, Athens, Greece, gregor@di.uoa.gr

This paper focuses on the design of TALENT, an adventure multiplayer game for learning introductory programming concepts. It describes the characteristics and the overall design of the educational game. In the multiplayer game world the student, represented by an avatar, is moving, collecting objects and engaging in programming activities in order to achieve the learning goals. In programming activities students make use of a special designed mini-language and a visual editor where the final program is constructed by drag and drop commands and instruction wizards. Information about the progress in programming activities and the achievement of learning goals as well as the learner's navigation and the use of tools, is used to construct the learner model. The student is able to see his learner model and to compare it with that of the peers. Also, the learner model enables the system to adapt to the individual user's current learning requirements. Finally a mentor type pedagogical agent inhabits in the environment and supports the learner by providing hints, motivation and guidance.

Keywords

Educational game, Pedagogical agent, Programming, Learner model, Tools for e-learning

1. Introduction

Few computing educators of any experience would argue that students find learning to program easy. The most common comment from students is that programming is "boring and difficult". This implies that one cognitive factor that might make learning to program difficult is motivation. If a student has the wrong motivation, he will find learning to program difficult. Different types of motivation have been described [1]. Extrinsic motivation express the desire to complete an activity in order to attain some expected reward, intrinsic motivation deriving from an interest in the subject and achievement - competitive motivation based on "doing well" and (sometimes) better than peers. Intrinsically motivated students may spend more time and effort learning, feel better about that learning, and use that learning more in the future [2]. Much of the existing research, in the computing education literature at least, focuses on new and interesting ways to teach programming [3]. One way is to focus on educational computer games that seem to improve intrinsic motivation through challenge, curiosity, control and fantasy [4].

In the next sections we will present the related work, games used for learning programming. We will present the characteristics that are missing from these environments which they form the reason for proposing the design of the educational game TALENT (Teaching ALgorithms EnvironmeNT), a multiplayer online educational computer game for supporting the learning of introductory programming subjects for university first year students.

© South-East European Research Center (SEERC)

2. Related work

Computer games like Robot Battle, RoboCode, CeeBot and ColoBot have been described to support the learning of programming. The first two games are open source multiplayer games, where the competitors write their own code that controls a robot that fights other identically-built (but differently programmed) in a playing field. Robots move, shoot at each other, scan for each other, and hit the walls (or other robots) if they aren't careful. The last two are commercial games where the user is engaged in programming activities trying to code a robot in order to accomplish the missions of the game. All the above games are strategy games where the user is programming a robot. Robot programming games seem to catch students' interests and encourage them to explore programming concepts [5]. Although these games provide motivation to students, they do not support two of the core characteristics of intelligent learning environments (ILE), which offer an individualized tutoring that is reported to provide the best learning for most people [6]. The first of these characteristics is the learner model, which is a representation of understanding, difficulties and misconceptions of the individual [7]. Learner model provides an additional resource through self-awareness and possible self-regulation of the learning process that is believed to enhance learning [8]. In addition, the learner model enables a system to adapt to the individual user's current learning requirements [9]. The second characteristic is a pedagogical agent a new opportunity to facilitate learning [10] that could lead to a more human and "social" learning environment [10]. Students interacting with pedagogical agents have been shown to demonstrate deeper learning and greater motivation [12].

In the next section we describe TALENT that utilizes game features and the two characteristics, described above, that are missing from the games presented so far, to support the learning of programming.

3. Characteristics and design of the educational game

TALENT is designed as a adventure multiplayer online educational game for supporting introductory programming subjects. Adventure games offer powerful opportunities for learning and development of problem-solving abilities [13] and are suitable for science concepts that may be hard to visualize or manipulate with concrete materials [14]. In addition, adventure games which are consisted of tasks provide a clear progression overall in the game [15]. In TALENT students explore the world, communicate with each other, exchange object with peers, and are involved in programming activities in order to succeed in the game. The communication between the students is done by a chat tool. The student can make use of the chat tool to exchange messages about the game or even to ask for an advice or help.

A mini-language supports the student in the game. The student makes use of a visual editor where drag and drop commands construct the final program and wizards help to eliminate syntax errors. The internal interpreter executes student's program step-by-step and the results of each command's execution are visible in the microworld. Programming activities, in current phase, are based on various programming subjects like variables, conditions (if-then-else) and loops (while-do, do-while). As the student advances in the game he comes upon a graded sequence of programming activities, a method that provides scaffolding [16]. These programming activities are based on database records and xml files and could be changed using the authoring tool of the environment. In the following subsections we will focus on TALENT characteristics as the game world, the mini-language that is used in programming activities, the learner model presentation that helps to adapt the system to student's learning requirements and the pedagogical agent that inhabits in the environment and supports the learner. Furthermore, we'll present the overall design, which combines the above characteristics and their interconnections.

© South-East European Research Center (SEERC)

3.1 The Game World

The overall idea is to keep the game world open. This means that a teacher will be able to alter the game world by using his own graphics, objects and places and activities presented to students. What remains constant is the general concept of the game. In this general concept the student, represented by an avatar, is exploring the game world and is engaged in various programming activities. Through the successful completion of these programming activities the student collects objects that help him to build his own space. In the first prototype the student takes the role of an archaeologist travelling all over the world in order to build his own room in a famous museum. The first time the student enters the game he must register. During this process the student gives his own personal characteristics (age, gender etc,), chooses a nickname and builds the presence of his own avatar so to be distinguished by the other players. These data records are saved in the game database.

In a recent work Carro et al [17] proposed a methodology to create adaptive educational game environments. They claim that in order to create an adaptive educational game environment one needs to create several different activities, indicate for each the learning goals involved and then group these activities into activities groups. Activity is the basic unit of the game structure and indicates a task to be performed. Following this methodology we divided the TALENT game world into three tiers Map, Place and Mission. An example with three places, seven missions and three different learning goals is shown in Figure 1.

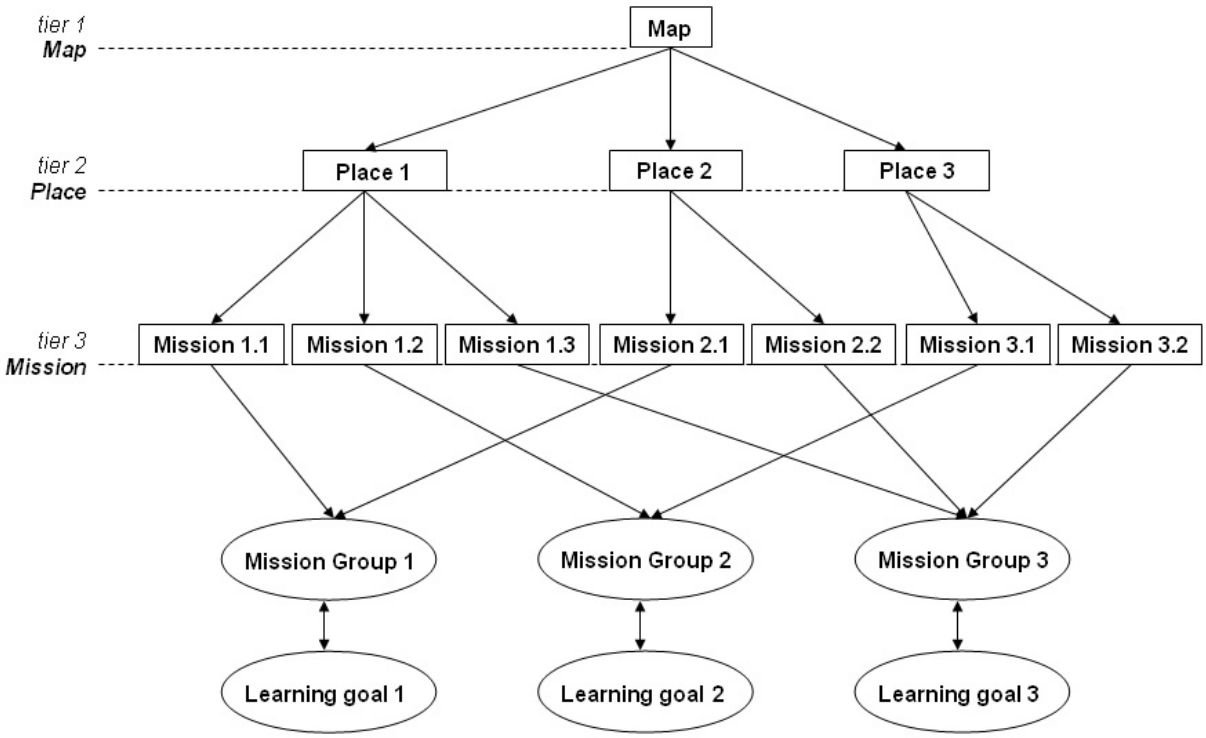

Figure 1. Example presents three different learning goals. Each learning goal corresponds to a Mission group. Each Mission group is formed by different missions from three different places.

The Map tier is the general map of the game world that is presented to the student as the game starts and outlines the places a student can visit. A Place is the part of the game that outlines the missions that are proposed to the student. Finally, Mission is a leaf in the game tree and contains the programming activity for the student to go through. Each programming activity has its own learning goals (for example, variables, conditions, loops) and programming activities from different places are grouped together according to their learning

© South-East European Research Center
(SEERC)

goals. Since each Mission tier holds one programming activity this leads to Mission Groups. The number of the Mission groups is equal to the number of the different learning goals.

To support an open game world where the teacher could be able to alter, useful information about the three tiers (Map, Place and Mission) are taken from database records and xml files. Mission tier contains not only the programming activity of the concrete mission but also the space (room) where the student is wandering, moving his avatar and collecting various objects that will be useful for completing the programming activity. The student can watch the movement of his avatar as well as the movement of all avatars in the same mission.The movement of student's avatar is done by clicking the mouse to the desired destination and the avatar finds and follows the best path to reach this point based on A* pathfinding algorithm [18].

3.2 The mini-language

The use of general purpose languages causes several obstacles for novice programmers as they are too big, too idiosyncratic, provide little leverage for understanding their basic actions and control structures and are oriented on number and symbol processing far from the students' everyday experience [19]. On the other hand, mini-languages are small (small syntax and a simple semantics), they are built on metaphors that intrinsically engaging and visually appealing to students, their operations are always visible revealing the semantics of language constructs and they do not rely on the syntax and semantics of a "big" language which may not be suitable for novices. This implies that the time required to master a mini-language is also small, so the students could spend their efforts on important issues like algorithm development, program design and debugging. So, applying the mini-language approach is ideal to support an introductory computer science course in a university setting.

In the educational game TALENT when a student is engaged in a programming activity he has to use the mini-language to instruct a robot vehicle to do the actions necessary to succeed its mission in the microworld. For example, the student must drive the vehicle to pick up a key and put it down on a concrete place at the end of a dangerous corridor in order to open a strange door. As designers we concentrated on a simple, operation visible, attractive and meaningful mini-language. The TALENT's mini-language provides simple syntax and semantics instructions like move, set variable, turn, fire, wait, pen, pick and display. The syntax and the definition for each command are presented in table 1.

Table 1. The mini-language supported commands with their syntax and definition.

Command	Syntax	Definition
Move	Move <distance>	Moves according to the value of the <distance>. The <distance> could be a number or a variable that has been set. The environment allows the student to apply a combination of number and variable with an operation (+,-,*, /). It allows also the generation of random number in a {min, max) range.
Set	Set <variable name> to <value>	Sets the variable with name <variable name> to the <value> given. <Value> could be a number or another variable that has been already set. The environment allows the student to apply a combination of number and variable with an operation (+,-,*,/). It allows also the generation of random number in a {min, max) range.
Turn	Turn [to | by] <angle>	Turns to or by the value given by the <angle> that can be a number or a variable already set.
Fire	Fire	Fires to the current direction

© South-East European Research Center (SEERC)

Wait	Wait <number of seconds>	Waits for the specified number of seconds before to execute the next command.
Pen	Pen [down | up] <value>	It draws (or not draws) with the pen colour represented by <value>.
Pick	Pick [up | down]	Pick up and object. If it has already picked up an object it puts it down.
Display	Display <message>	Displays a message. The message could be a text or a value of a variable or a combination.

In addition, the mini-language supports algorithmic structures for conditions (if-then-else) and loops (repeat, while). The command toolbox of the mini-language is shown in figure 2.

Figure 2. The toolbox of the mini-language commands. The name of each command is shown below its button.

In order to prevent student's syntax errors issued from keyboard usage, the environment provides support with instruction wizards. The student drags and drops a command from the mini-language toolbox and completes the instructions using a wizard window. In that way students are free to concentrate on solving the stated problem without too much worry about how to satisfy the syntax requirements [20]. The environment maintains both the microworld and the student program visible on the screen. The program is executed one instruction at a time, while the interpreter highlights programming constructs in the source code as they are being executed and the effect is simultaneously shown in the microworld. This makes the connection between the mini-language command or construct and its effect on the microworld obvious.

3.3 The open learner model

The most common uses for the learner's model are to decide whether to advance the learner to the next topic, to provide unsolicited advice and to find out learner's capabilities and the constructed knowledge so far [21].

The TALENT constructs the learner model based on information about learner's navigation, the use of tools, the progress in programming activities and the achievement of learning goals. Data that concerns the sequence of places and mission selection, the total time spent on a place or a mission, the total number of times that the student asked for help in a mission, the visits on a mission or a place, the number of times he communicated with the other students in the game and the progress on learning goals are saved in a separate database record according to student's nickname. According to the student's level of knowledge the system propose the next mission to be taken. When the user succeeds in all missions that correspond to the first learning goal, the system proposes the first mission for the second learning goal etc. Although the environment proposes the next mission to the student according to his abilities based on his model, student is free to select any mission. Furthermore, he is able to try to catch a later learning goal although he didn't complete the previous missions with success.

In addition, the model is open to student. This means that the model is a visible and interactive part of the learning environment which could give students power over their learning, a learning dialogue and motivation [22]. Furthermore, one can compare the information in his model with that produced with the mean data of all the other students. This

© South-East European Research Center (SEERC)

knowledge of the whole progress has been argued to motivate student to try to succeed more in the activities [23].

Information about the learner model (performance and use) provided by the environment at the coarse (overall level) and fine (learning goal level) grain is shown in Figure 3.

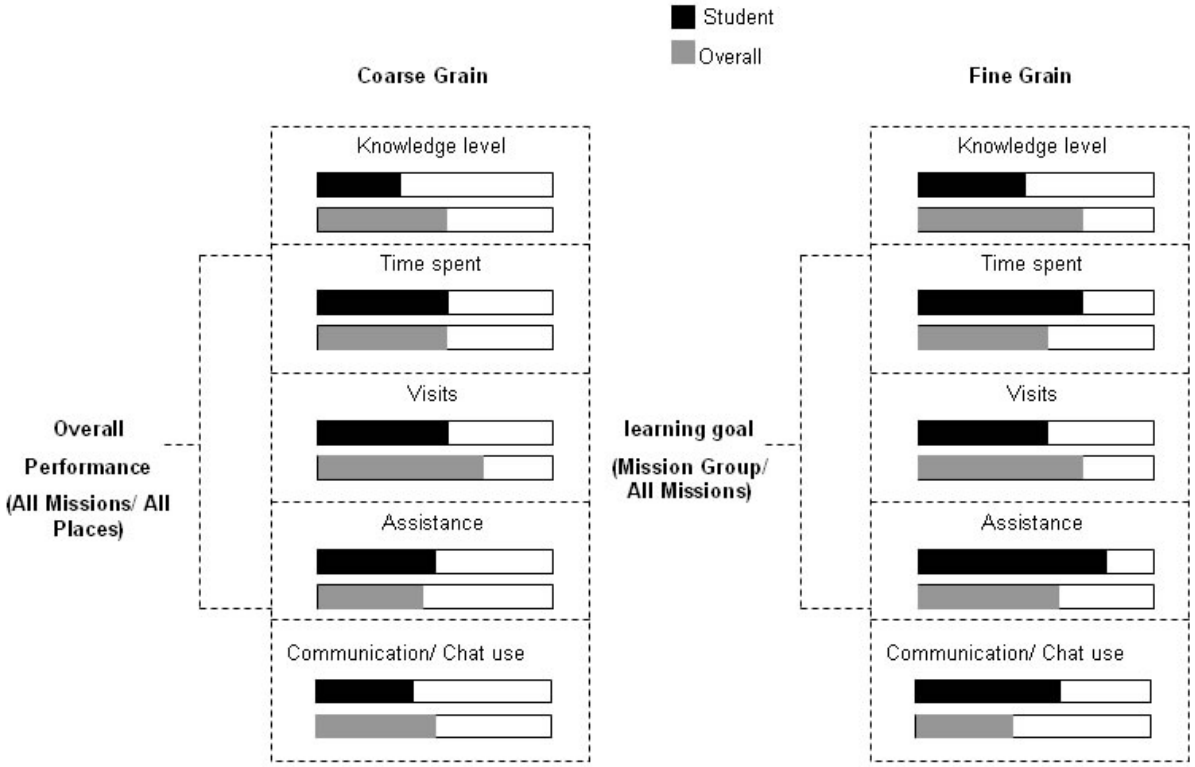

Figure 3. Information provided by the open learner model in coarse and fine grain.

At the coarse observation level (left column in figure 3) the information presented is Knowledge level, time spent, visits, assistance and communication between the students generated by all missions to all places. For example, the value of time spent at the coarse level represents the time that student spent on the missions (programming activities) compared with the time spent on all places (missions and moving/ collecting objects etc.) respectively. The same type of information is presented at the fine grain (right column in figure 3) where the results are based on the observation of a single learning goal (mission group) to all missions. For example the value of time spent at the fine grain level represents the time that student spent on a learning goal programming activities (mission group) compared with the time spent on all programming activities (all missions) respectively. At both levels the information for the student is compared with the same type of information provided by all players (overall).

3.4 The pedagogical agent

A Pedagogical agent is defined as an agent that engages interaction with learners. They can monitor student as they solve the problems, guide them or even collaborate with them as members of a team. The use of a pedagogical agent in an educational environment has the advantage to increase the communication between students and computers and also the ability to engage and motivate the students [24].

© South-East European Research Center
(SEERC)

Considering three types of pedagogical agents (Expert, Motivator and Mentor), Mentor type leads to overall improved learning and motivation [25]. A mentor pedagogical agent provides not only information but also motivational support and guidance.

In the TALENT educational game Alex (ALgorithm EXpert) supports the actions of the student. Alex is a mentor type pedagogical agent and is responsible to provide help to the student if the student asks for help or even he didn't ask for help but agent judge that he needs help. Alex intervenes in the environment providing hints. Alex is also responsible to motivate the student to continue his efforts and also to guide him in the game world proposing the next mission according to student capabilities. To accomplish this difficult mission the pedagogical agent needs to watch the actions of the student in the game world and also to consult his learner model (see Figure 4).

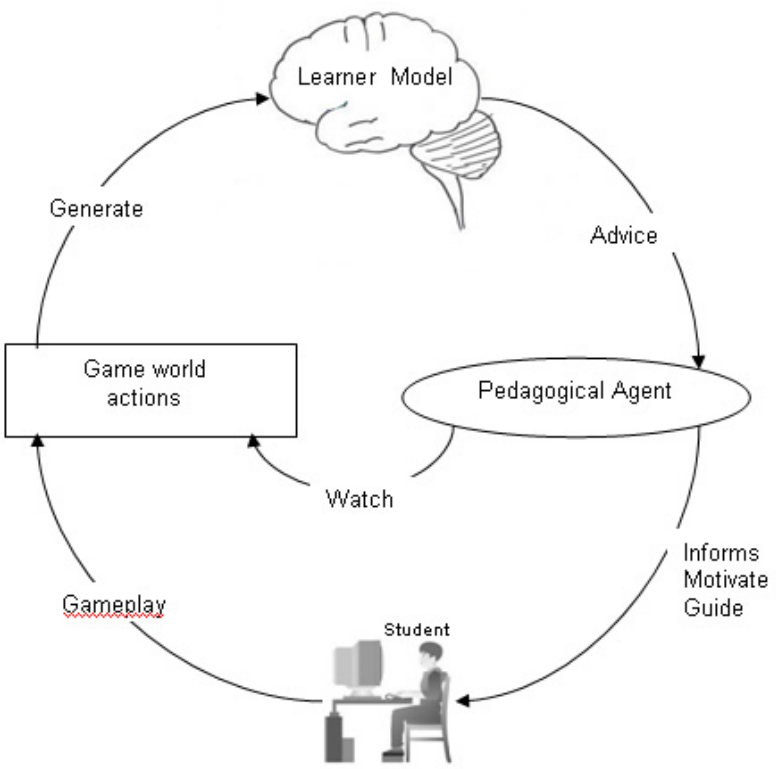

Figure 4. The pedagogical agent consults the learner model and watches the student's actions in order to provide support like information, motivation and guidance.

3.5 The overall design

In order to design TALENT we used the experiential game design model [26] altered accordingly to support a pedagogical agent and a learner model, characteristics that are found in Intelligent Learning Environments (ILE) [27].

The experiential game model describes learning as a cyclic process and learning is defined as a construction of cognitive structures through action in the game world. The game should provide clear goals and appropriate feedback in order to facilitate the flow experience [28]. The model supports double-loop learning [29] where the player can test different solutions in order to optimize his strategy and expand knowledge on the subject. As the student interacts with the game world in TALENT, the environment alters the learner's model. The pedagogical agent of the game relies on the information provided by the learner model to give assistance and appropriate feedback. The pedagogical agent takes the role of a mentor,

© South-East European Research Center (SEERC)

being able to provide hints, motivation and guidance. Scaffolding is provided as a graded sequence of missions and programming activities. The overall design is shown in Figure 5.

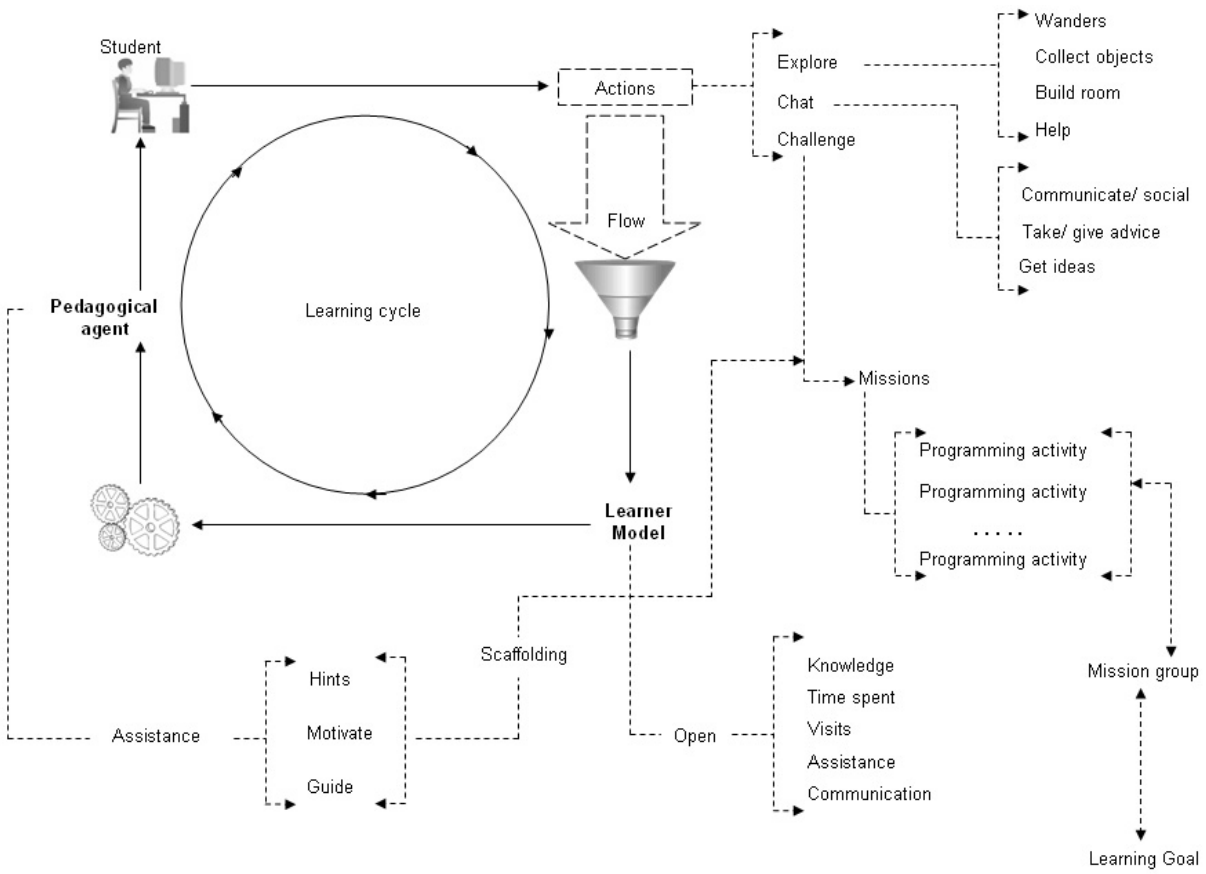

Figure 5. The overall design of the TALENT where learning is a cyclic process and scaffolding is provided by the pedagogical agent based on the information of the learner model. The learner model is constructed through the actions in the game world which has to facilitate the flow experience.

4. Conclusions

This paper has analysed the design of TALENT, a multiplayer online educational game for the learning of introductory programming concepts. The game world consists of three tiers where the third tier includes several different programming activities. In order to support adaptivity, these activities are grouped according their learning goal making different activities groups. In order to eliminate syntax errors the system uses a special editor where the student drags and drops the mini-language commands from a command toolbox. Data like the learner's navigation, the use of tools, and the achievement of learning goals are processed to produce the learner model. The learner model is used by the pedagogical agent to support the student with hints, motivation and guidance. The use of the above characteristics combined with game features could lead an educational environment to provide individualized tutoring while motivating and engaging the learner.

TALENT, in current phase, is under development. The system has to be tested with real data and for this reason a research has been planned. The results from this research will be analysed further in purpose to reveal the potential of the environment.

© South-East European Research Center (SEERC)

References

1 Entwisle, N., "Motivation and Approaches to Learning: Motivating and Conceptions of Teaching", *Motivating Students*, Kogan Page, 1998.

2 Malone, T. (1981). Toward a theory of intrinsically motivating instruction. Cognitive Science, 5(4), 333-369.

3 Jenkins, T. On the difficulty of learning to program. In Proceedings of 3rd Annual LTSN_ICS Conference (Loughborough University, United Kingdom, August 27-29, 2002). The Higher Education Academy, 53—58

4 Lepper, M. R. & Malone, T. W. (1988). Intrinsic motivation and instructional effectiveness in computer-based education. In R. E. Snow & M. J. Farr (Eds.). Aptitude, learning, and instruction: Vol. III. Cognitive and affective process analyses. (pp. 255 – 286). Hillsdale, NJ: Erlbaum.

5 Kevin J. Bierre , Andrew M. Phelps, The use of MUPPETS in an introductory java programming course, Proceedings of the 5th conference on Information technology education, October 28-30, 2004, Salt Lake City, UT, USA

6 Encyclopedia of Educational Technology, available online http://coe.sdsu.edu/eet/Admin/Intro.htm

7 Bull, S. (2004). Supporting Learning with Open Learner Models. 4th Hellenic Conference with International Participation: Information and Communication Technologies in Education, Athens, 2004. (Keynote)

8 Kay, J. (1997). Learner Know Thyself: Student Models to Give Learner Control and Responsibility, in Z. Halim, T. Ottomann & Z. Razak (eds), Proceedings of International Conference on Computers in Education, Association for the Advancement of Computing in Education (AACE), 17-24.

9 Cui, Y. & Bull, S. (2005). Context and Learner Modelling for the Mobile Foreign Language Learner, System 33(2), 353-367

10 Johnson, W. L., Rickel, J. W., & Lester, J. C. (2000). Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education,* 11, 47-78.

11 Hermans B. "Intelligent Software Agents on the Internet", FirstMonday, vol. 2, 3/1997.

12 Morozov M., Tanakov A., Bystrov D. (2001). Pedagogical Agents in the Multimedia Natural Sciences for Children, *The 3rd IEEE International Conference on Advanced Learning Technologies,* Athens, Greece, 9-11 July 2003 pp. 62-65

13 McFarlane, A. (ed.) 1997, Information Technology and Authentic Learning: Realising the potential of computers in the primary classroom. London: Routledge

14 Mitchell, A. & Savill-Smith, C. (2004) The use of computer and video games for learning: A review of the literature. Available: http://www.lsda.org.uk/files/PDF/1529.pdf (Accessed: 2005, July 25)

15 McFarlane, A., Sparrowhawk, A. & Heald Y. (2002) Report on the Educational Use of Games. Available: http://www.teem.org.uk/publications. (Accessed: 2005, July 18)

16 Graci, C., Odendahl, R., & Narayan, J. (1992). Children, chunking, and computing. *Journal of Computing in Childhood Education 3*, 3/4, 247-258.

17 R. M. Carro, A. M. Breda, G. Castillo, and A. L. Bajuelos "A Methodology for Developing Adaptive Educational-Game Environments", In P. De Bra, P. Brusilovsky, and R. Conejo(Eds.): AH2002, LNCS 2347, pp90-99. Springer- Verlag, Berlin Heidelberg 2002.

18 Lester, P. (2005). A* pathfinding for beginners. Available: http://www.policyalmanac.org/games/aStarTutorial.htm

19 Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997) Mini-languages: A Way to Learn Programming Principles. *Education and Information Technologies* **2** (1), pp. 65-83.

20 Baldwin, R., (2007). Learn to Program using Alice Syntax, Runtime, and Logic Errors. Available: http://www.dickbaldwin.com/alice/Alice0155.htm

21 K. VanLehn.(1988). "Student Modeling". In M.C. Polson and J.J Richardson, Foundations of Intelligent Tutoring Systems, (pp55-78).

22 Kerly, A. & Bull, S. (2007). Open Learner Models: Opinions of School Education Professionals, in K. Koedinger, R. Luckin & J. Greer (eds), Artificial Intelligence in Education, IOS Press, Amsterdam.

23 Bull, S. & McKay, M. (2004). An Open Learner Model for Children and Teachers: Inspecting Knowledge Level of Individuals and Peers. Intelligent Tutoring Systems: 7th International Conference, Springer-Verlag, Berlin Heidelberg, 646-655.

© South-East European Research Center (SEERC)

24 W. L. Johnson & J. W. Rickel “Research in Animated Pedagogical Agents: Progress and Prospects for Training”, in IUI 2001.
25 Baylor, A. L. & Kim, Y. (2005). Simulating instructional roles through pedagogical agents. *International Journal of Artificial Intelligence in Education*, 15(1)
26 Kiili, K. (2005a). Digital game-based learning: Towards an experiential gaming model. The Internet and Higher Education, 8, 13–24.
27 Maragos, K. & Grigoriadou, M., (2005), “Towards the design of Intelligent Educational Gaming systems” Proceedings of Workshop on Educational Games as Intelligent learning environments, Artificial Intelligence in Education, University of Amsterdam, Amsterdam
28 Csikszentmihalyi, M. (1990). Flow: The psychology of optimal experience. New York: Harper Collins.
29 Argyris, C. & Schön, D. (1974). Theory in practice: Increasing professional effectiveness. San Francisco: Jossey-Bass.

© South-East European Research Center (SEERC)