GAME-BASED APROACH IN IT EDUCATION

Olga Shabalina, Pavel Vorobkalov, Alexander Kataev, Alexey Tarasenko

Abstract: A concept of using game based approach for IT education is presented. The learning course notation rules for using in games are suggested. Techniques for integration of learning components to the game are described. The realization of the learning game for programming language C++ is shown.

Keywords: educational games, digital game-based learning, course notation, game engine, learning engine.

ACM Classification Keywords: K.3.1 Computer Uses in Education; K.3.2 Computer and Information Science Education

Conference: The paper is selected from Fourth International Conference "Modern (e-) Learning" MeL 2009, Varna, Bulgaria, June-July 2009

Introduction

Software development is a continuously expanding area. At present time are more than one million computer software engineers in Russia. They are employed in most industries and their largest concentration is in computer systems design and related services. Computer software engineers is one of the occupations projected to grow the fastest, their employment is projected to increase by 38% over the 2006 to 2016 period [1]. Demand for computer software engineers will increase for a number of reasons. Organizations adopt and integrate new technologies and seek to maximize the efficiency of their computer systems. Computer networking growth needs specialists for developing Internet, intranet, and World Wide Web applications. Electronic data-processing systems continue to become more sophisticated and complex. Information security concerns have given rise to new software needs. Businesses invest heavily in software that protects their networks and vital electronic infrastructure from attack. Implementing, safeguarding, and updating computer systems and resolving problems will fuel the demand for growing numbers of high-quality software engineers.

From the other hand the IT labor market growth contradicts a predicted shortage of quality computer engineers IT skills [2]. Admissions and retention rates to some university computer software specialties are falling and there are serious concerns about the quality of the software which they create. Economic crisis has changed the situation in Russia and all over the world, staff reductions take place in different areas and IT area as well, but those of lower professional skill are the first ones being reduced. Thus the problem of training of high-quality software engineers is still being topical.

Teaching computer software engineers faces some specific problems. A set of basic skills needs to be understood and applied at once from very early in the learning process. The IT field is being dynamically changed, and sometimes trainers are behind the time and behind trainees [3]. To a great extent it is because of trainers and trainees are from totally separate worlds. Trainers and teachers raised in a pre-digital generation and educated in the styles of the past, while trainees, so called Games Generations ((N-gen, D-gen)[3], raised in the digital world and are native speakers of the digital language of computers, video games and the Internet. The trainees find today's training (and education) so incredibly boring that they don't want – and often refuse – to do it.

There are a number of approaches to making the education easier and more appealing, and a remarkably promising one from these is using computer games. Marc Prensky [3] suggested a new approach – combining together serious learning and computer games into a newly medium — Digital Learning Games. Use of computer games for non-entertainment purposes (so called Serious Games), and its sub-field Digital Game-Based Learning are an emerging area of research and interest to this area is growing rapidly.

Game-based approach achieves high learning results in areas that are difficult to study and where gaining skills is of importance. To a great extent it can be referred to the computer science disciplines. One can mention such subjects, demanding skill gaining, as algorithmization and programming, artificial intelligence, computer graphics and so on.

By now the potential of digital game-based learning remains still largely unrealized, and design and implementation of new games in education process is in great demand.

IT games concept

Saying ‘educational game’ we mean a learning system, which realizes some or all components of learning process (learning theory, gaining skills and experience, estimation of knowledge level) in a game context. Our approach is based on three positions: learner must get the course information through its interpretation in a game world; learner must see the result of his learning in a game context; results of learning must influence game results.

We have analyzed what kinds of games would be more effective and suitable for educational games on ICT subjects. We decided to use core games for learning courses in a whole and for gaining skills.

Core games need sequential logical playing and are time-taking. The idea of educational core games is based on the correspondence between learning course and gameplay. Normally the game scenario consists of game levels. Each level includes several game quests. In educational game quests include some course information and assignments for getting practice. Each assignment solution needs knowledge of related course chapters, on the other hand information and assignments are associated with a game story. Thus levels completing forces player to learn new course chapters and solve level assignments, so the game scenario relates to the learning course.

The correlation between game levels and course chapters provides using such games both for individual learning and for using at universities or at schools. Teachers might use the game level by level according to their studies, so learners would complete playing the game when they complete studying the course.

A game-related course description has been elaborated for learning trough games. A course content should be divided into theory information blocks and. Each block relates to one or several chapters of the course. Information blocks can be of two types: displayable and reference blocks. Displayable information blocks are parts of the game script and are displayed for a player. Reference information blocks contain additional information on the corresponding chapter and are available for the player on demand. A set of displayable and reference information blocks form full learning course theory. Assignments should be described as assignment blocks and include assignment description, solution description, and solution interpretation. Assignment description is a displayable information block, representing different kinds of quests. Assignment solution requires writing some text, answering questions, testing or fulfilling some other actions. Solution description includes solution testing rules or/and solution examples. Solution interpretation includes right and wrong solutions visualization techniques or/and their influence on a gameplay.

Course description is made by an instructional designer and a game coder in three stages (Table 1). First stage should be made by an instructional designer. This level describes blocks' content, their sequence and links. Second stage is made both by instructional designer and a game coder. A game coder describes solution testing types and solution visualization techniques. The last stage is made by a game coder. For coding information blocks and links mark-up language is used. Blocks sequences, assignment interpretation and solution blocks are coded in an engine input language.

Table 1: Course description levels

Level	*Type of description*	*Who acts?*	*Description language*
Logical level	Blocks' content Blocks' sequence Blocks' links	Course designer	Natural language
Extended logical level	Solution testing types Solution visualization techniques	Instructional designer Coder	Course description language
Implementation level	Codes of information blocks Links Blocks sequences assignment interpretation blocks Solution blocks Marked up text presentation Assignment solutions	Coder	Engine input languages: (mark-up language, scripting language)

We elaborated techniques for integration of learning components to the game (Table 2). We represent learning course using Formatted text (HTML) with hyper-links support. While playing a learner must solve assignments. We use engine input languages, depending on a learning course, for the solutions interpretation. We check the solutions using matching techniques.

One of the modern tendencies in the field of e-learning is using so called adaptive learning systems. Such kind of systems can adapt to the needs of users and provide individualized learning. As we plan to develop adaptive learning games, we use Bayesian approach for estimation of a learner knowledge level [4] and nonlinear story for adaptation of learning process to individual users.

Implementation

We started with learning games on programming languages, because our students start the learning process from learning programming languages, a fundamental subject for software developers. Mastering programming is a difficult-to-learn subject. It needs a high-level of abstraction, without a direct feedback to a real world, and includes skill gaining. To a great extend learning programming languages means gaining skills in programming. We've found out that there are only few examples of games on learning programming, though this subject is very good for teaching through games. These games use specific built-in language (KuMir, ColoBot, CeeBot), or they need an advanced level of programming for playing (AI Competition Games). They are not suitable for learning university programming courses in a whole.

Our concept [5] is based on a Role Playing Game (RPG) genre. The main game character is a transformer, who lives in a virtual world. A player can program different transformer shapes and his behavior. Different situations in the game require different characteristics of the main character. Transforming shape allows players to get optimal character parameters for playing. At the beginning a player programs the basic character abilities (moving, jumping, and attacking). The results of programming are used by the player to control character behavior in the game. Then the player can program controlling algorithms for intelligent behavior of the character (overcoming different obstacles, going out labyrinths and so on). Effectiveness of the code determines effectiveness of character behavior. Unreasonable or foolish behavior prompts player to improve program code. Thus the player obtains knowledge and gains skills while playing.

Table 2: Techniques for integration of learning components

Learning components	*Techniques*	*Realization*
Course presentation	Formatted text with hyper-links support	HTML
Gaining skills	Solving assignments (using Input language)	Depends on an engine input language
Estimation of knowledge level	Solution verification (comparison with a set of right solutions)	Matching techniques
Dependence of a game process on a learning process	Only right solutions enable the game process Nonlinear story	Adaptive techniques Bayesian approach

As it was mentioned above, each learning course has some specific characters. For learning programming languages they are: input language for assignment solutions is programming language and correspondingly assignment solution itself is a programming code. We use two techniques for checking programming code: verification (checking source code using corresponding rules) and running (executing source code and checking the result). We use verification for checking assignments that have simple solutions (simple code) for a player. Real implementation methods using for realization of these solutions in the game are much more complicated. Interpretation of right solutions of verifiable type uses the real implementation methods. Methods, showing that the assignment hasn't been executed, are called for interpretation of wrong solution of verifiable type.

Verification method is based on regular expressions. Verification of programming code requires specialized matching because of more than one (practically infinite) number of possible right solutions. The specialized library has been developed for taking into considerations of these peculiarities (different possible identifiers names, blocks order, notations of mathematical expressions).

For visualization of runnable solution the code of solution is being executed a player can see the result of his coding directly through game characters behavior. Execution results can be either numbers (as calculation results) or reaching some goals (e.g., a game character has arrived to a given point using the elaborated code). Execution results are compared with right solution results. Code running allows checking solutions not provided by course authors, but having the same results. Examples of checking ways are shown in the Table 3.

Table 3: Examples of checking ways

Solution check	*Description*	*Example (C++ code)*

Verification	Comparison of code with correct solutions provided by course developers	`MoveForward()`
		`{`
		`    x++;`
		`}`

Running	Executing source code and checking the result	`while(!WallBehind())`
		`{`
		`    MoveForward();`
		`}`

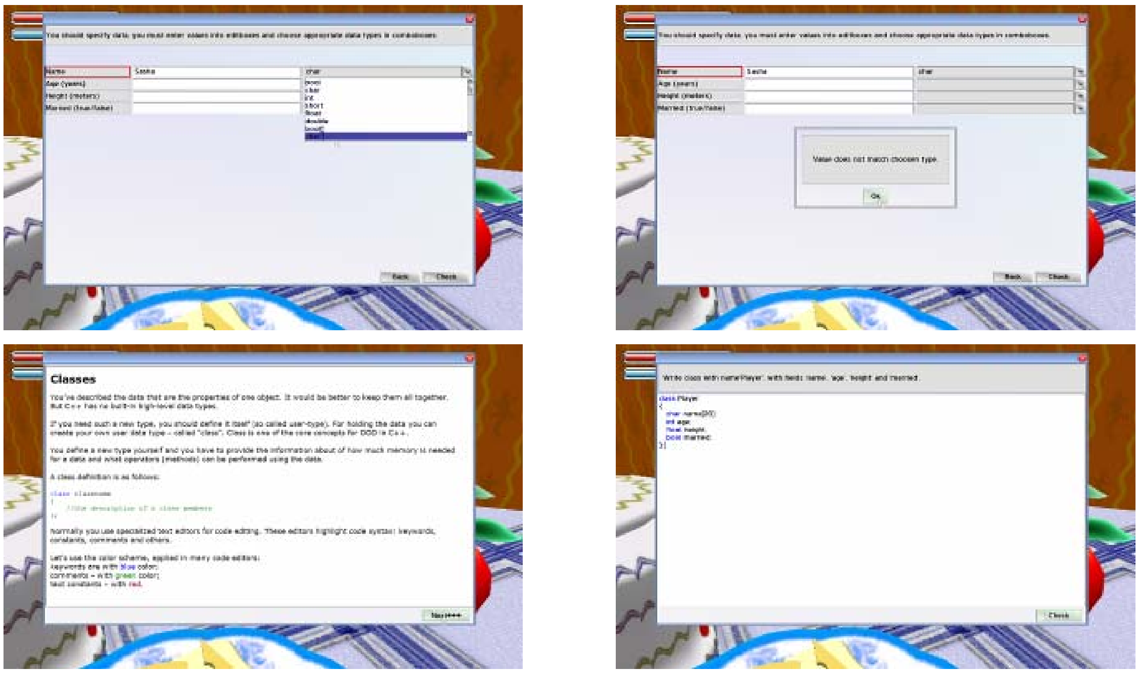
Game architecture is based on common game engine architecture, but it is extended for using in educational games and consists of two high-level subsystems: a game engine and a learning engine. The game engine is based on the graphical engine Ogre3D and enlarged with game logic and advanced user interface (for advanced text displaying and editing). Script core provides game engine modules interaction. Using scripts simplifies game logic programming and avoids recompiling game engine after learning course modification. The learning engine completes game engine functions with learning process support, which includes information course blocks and assignments control. Learning engine modules and routines are exposed to scripting core for organizing course sequence control and assignments solutions check from game scripts.

Game implementation is based on free for commercial use libraries and utilities (Table 4).

Table 4: Game implementation

Component	*Tool*	*Why is it chosen?*
Graphical engine	Ogre3D	Supports modern technologies, has extensible architecture, and is allowed for commercial usage.
Scripting language	Lua	Handy syntax, easy integration process, fast interpreter
Learning subject	C++	The .NET Framework built-in C++
Store course blocks	Encrypted zip archives	Open format
Engine	C++ language	High performance, flexibility, open source libraries

Games usually start from input by a player some information about himself. And as each programming language course begins from learning data types, we use the input process for learning basic data types. A player fills in not only the information about himself, but he must also choose appropriate data type for each field from the given list (Figure 1).The list includes data types, their purpose and range description. After it he needs to create a class with these fields as class properties. For explaining a concept of object-oriented design the real game class hierarchy is used.



You should specify data: you must enter values into editboxes and choose appropriate data types in comboboxes.

Name	Sasha	char
Age (years)		
Height (meters)		
Married (true/false)		

Value does not match chosen type

Classes

You've described the data that are the properties of one object. It would be better to keep them all together. But C++ has no built-in high-level data types.

If you need such a new type, you should define it itself (so called user-type). For holding the data you can create your own user data type – called "class". Class is one of the core concepts for OOD in C++.

You define a new type yourself and you have to provide the information about of how much memory is needed for a data and what operators (methods) can be performed using the data.

A class definition is as follows:

```
class classname
{
    //the description of a class members
};
```

Normally you use specialized text editors for code editing. These editors highlight code syntax: keywords, constants, comments and others.

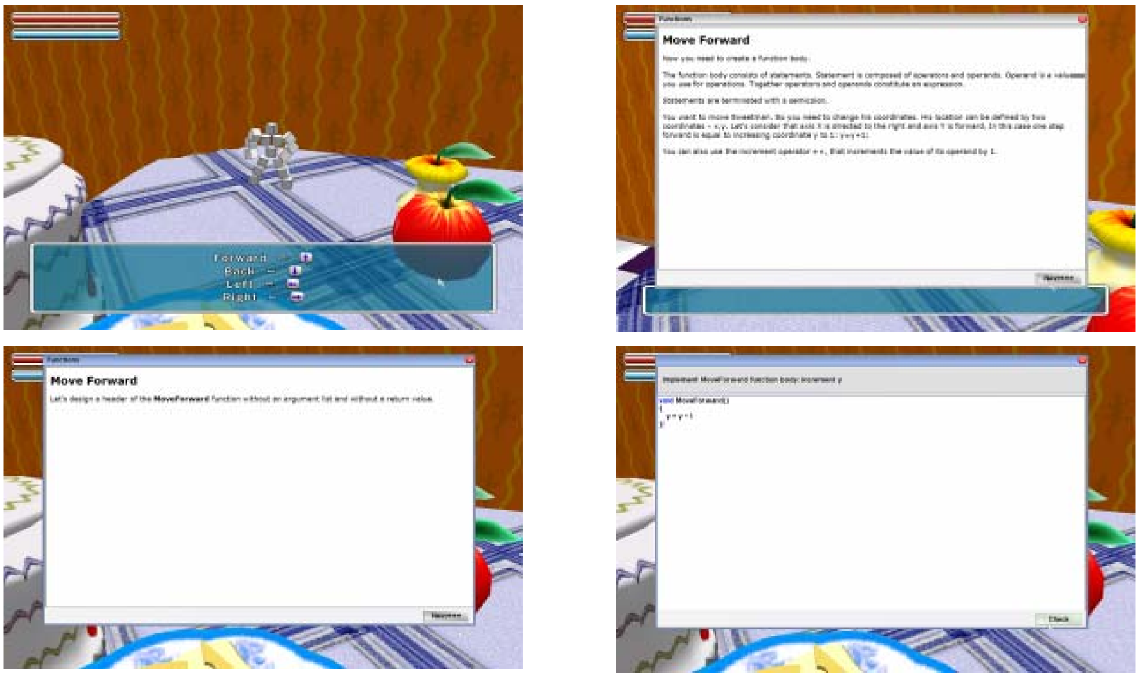
Let's use the color scheme, applied in many code editors:
keywords are with blue color;
comments – with green color;
text constants – with red.

Write class with name'Player', with fields: 'name', 'age', 'height' and 'married'.

```
class Player
{
  char name[20];
  int age;
  float height;
  bool married;
};
```

Figure 1: Studying data types

Next a player sees that his character can't move. So he needs to teach him how to do it, i.e. to develop programs for his moving. He designs functions for moving forward, back, left and right step-by-step. In such a way a player learns functions. After he has coded a function he can see the result. If his code is correct, transformer can move to the corresponding direction. If not, transformer can't move (Figures 2, 3).



Move Forward

Now you need to create a function body.

The function body consists of statements. Statement is composed of operators and operands. Operand is a value you use for operations. Together operators and operands constitute an expression.

Statements are terminated with a semicolon.

You want to move Transformer. So you need to change his coordinates. His location can be defined by two coordinates – x,y. Let's consider that axis X is directed to the right and axis Y is forward. In this case one step forward is equal to increasing coordinate y to 1: y=y+1.

You can also use the increment operator ++, that increments the value of its operand by 1.

Move Forward

Let's design a header of the **MoveForward** function without an argument list and without a return value.

Implement MoveForward function body: increment y

```
void MoveForward()
{
  y=y+1
}
```

Figure 2: Studying functions

After writing correct functions for moving a player can move along the table. Then he is asked to add the functions as class methods to a class Player. Thus while playing he elaborates new properties and methods (changing speed, jumping, attacking and so on), and in such a way he improves the character's behavior.

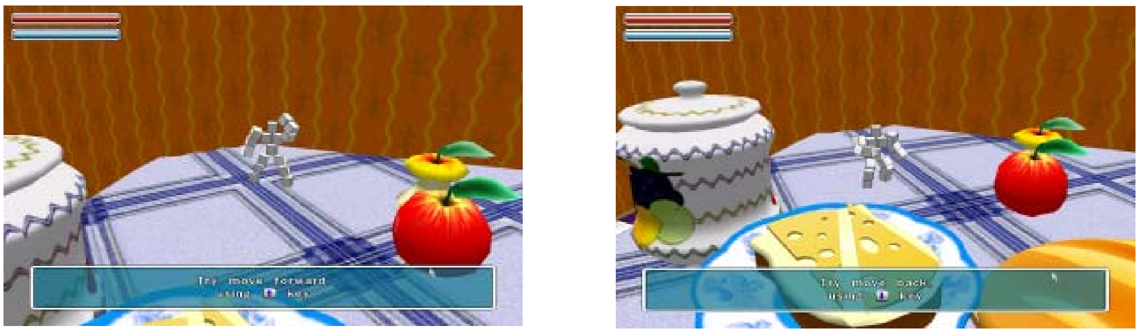


Figure 3: Checking the result

After some time of playing he needs to move down from the table, but he can't do it using current shape, so needs to get a new shape – the Worm (Figure 4).

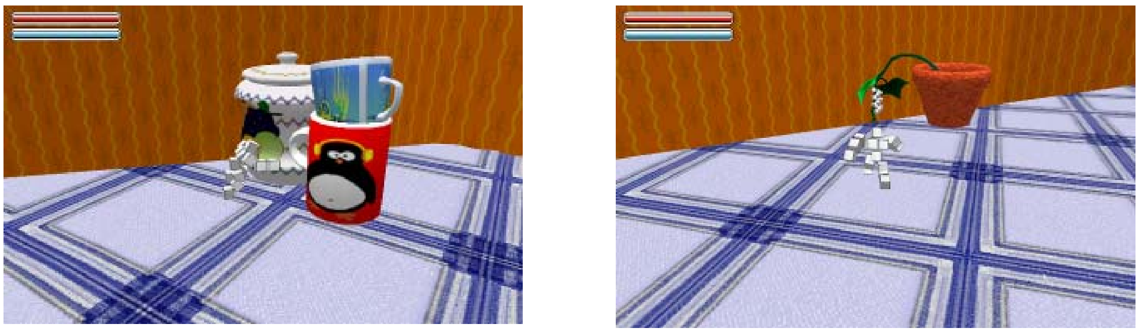
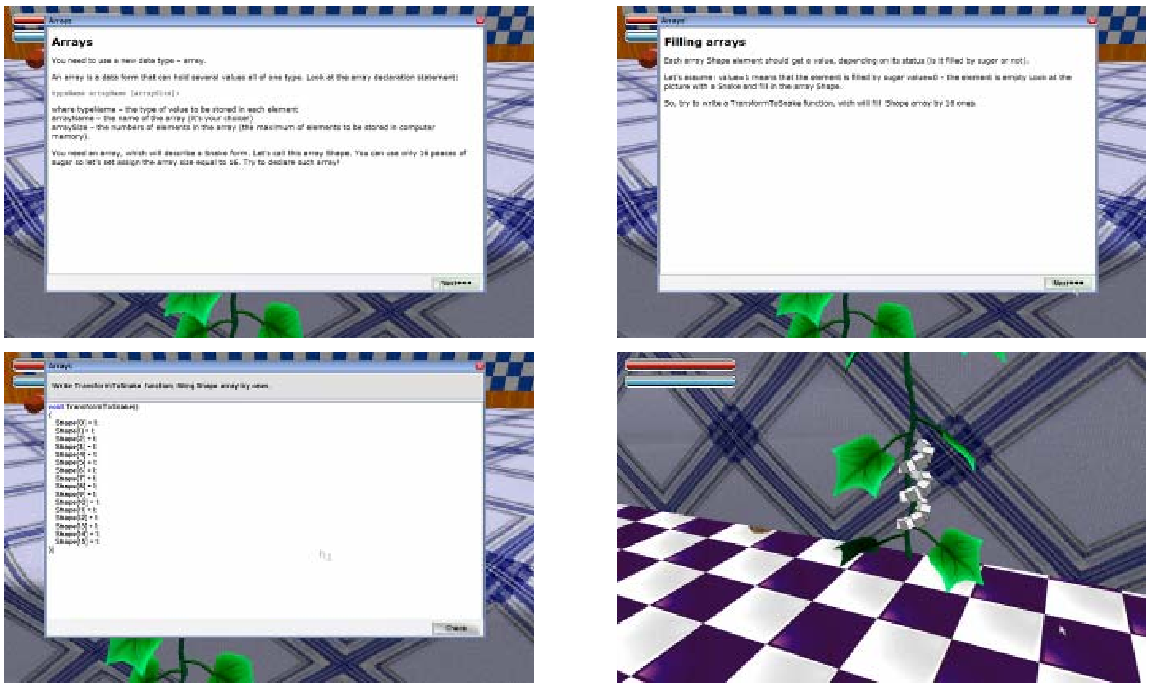


Figure 4: Changing a shape



Arrays

Arrays

You need to use a new data type – array.

An array is a data form that can hold several values all of one type. Look at the array declaration statement:

```
typeName arrayName [arraySize];
```

where typeName – the type of value to be stored in each element
arrayName – the name of the array (it's your choice!)
arraySize – the numbers of elements in the array (the maximum of elements to be stored in computer memory).

You need an array, which will describe a Snake form. Let's call this array Shape. You can use only 16 pieces of sugar so let's set assign the array size equal to 16. Try to declare such array!

Arrays

Filling arrays

Each array Shape element should get a value, depending on its status (is it filled by sugar or not).

Let's assume: value=1 means that the element is filled by sugar value=0 – the element is empty Look at the picture with a Snake and fill in the array Shape.

So, try to write a TransformToSnake function, wich will fill Shape array by 16 ones.

Arrays

Write TransformToSnake function, filling Shape array by ones.

```
void TransformToSnake()
{
  Shape[0] = 1;
  Shape[1] = 1;
  Shape[2] = 1;
  Shape[3] = 1;
  Shape[4] = 1;
  Shape[5] = 1;
  Shape[6] = 1;
  Shape[7] = 1;
  Shape[8] = 1;
  Shape[9] = 1;
  Shape[10] = 1;
  Shape[11] = 1;
  Shape[12] = 1;
  Shape[13] = 1;
  Shape[14] = 1;
  Shape[15] = 1;
}
```

Figure 5: Studying arrays

For doing it he needs to learn a new data type – array (Figure 5).Further game situations need new shapes (Helicopter, Ball, Weight) and corresponding behavior for each shape. Thus the player step by step learns object-oriented design and C++. At the end he creates a class diagram close to the reductive class diagram applied in the game.

To develop learning games for other programming languages it is necessary to describe a learning course according to the course description rules. The concept and game engine can also be used for other IT games development. In this case the ways of solution check should be defined according to appropriate learning course and proper tools should be implemented.

Conclusion

We consider that using this approach allows gaining and improving knowledge and skills in computer science, and raising the motivation to study.

Bibliography

1. Computer Software Engineers [online] http://www.bls.gov (appeal date 02.03.2009)
2. Madeline Alsmeyer, Judith Good, Katherine Howland, Graham McAllister, Pablo Romero and Phil Watten. Supporting the learning of programming in a social context with multi-player micro-games. Department of Informatics, University of Sussex. http://shareitproject.org/workshop/shareit.pdf
3. Marc Prensky. Digital Game-Based Learning. Paragon House Publishers, 2004.
4. Shabalina O.A. Models and Techniques for Knowledge Control in Adaptive Tutoring Systems, Unpublished Ph. D. Dissertation, Astrakhan University, Russia, 2005.
5. Shabalina_et_al. Educational Games for Learning Programming Languages // Methodologies and Tools of the Modern (e-) Learning: suppl. to Int. Journal "Information Technologies and Knowledge". - 2008. - Vol. 2, [Int. Book Series "Inform. Science & Comput."; № 6]. - P. 79-83.

Authors' Information

Olga Shabalina – PhD, senior lecturer; CAD department, Volgograd State Technical University, Lenin av., 28, Volgograd, Russia; e-mail: O.A.Shabalina@gmail.com

Pavel Vorobkalov – PhD, senior lecturer; CAD department, Volgograd State Technical University, Lenin av., 28, Volgograd, Russia; e-mail: pavor84@gmail.com

Alexander Kataev – PhD student; CAD department, Volgograd State Technical University, Lenin av., 28, Volgograd, Russia; e-mail: kataevav@mail.ru

Alexey Tarasenko – Second Higher Education student; CAD department, Volgograd State Technical University, Lenin av., 28, Volgograd, Russia; e-mail: volgatav@mail.ru