ABSTRACT

In recent years, all over the world, students are less interested in science, especially in computer science where the number of students is shrinking. Also, 50% or more students who initially choose computer science soon abandon the study. Hence, it is essential to attract students to learn computer science, in particular computer programming, via non conventional teaching approach. A promising medium is serious game for learning computer programming, given its fun gameplay characteristics. However, several games intended for computer programming are either hard to be extended, lack of dynamicity, or are not challenging which lead to boredom and lack of motivation in learning. On the other hand, computer science unplugged learning for primary-aged children via mathematical based activities using, for instance, cards and boards, is considered to be very compelling and improving conceptualization. Moreover, problem solving skill acquisition in form of puzzle based learning for university students are immediately pertinent to their problem solving skills development due to its attracting and intellectually challenging nature. Given these facts, this paper presents game contents targeted for different prior knowledge in programming which will be used in serious game for learning programming.

Keyword: Game Contents, Serious Games, Computer Aided Learning, Learning Computer Programming

1. INTRODUCTION

The advancement of information technology has brought many changes on how we work and live. Learning as central part of human activity is also affected by this innovation. Digital learning has been widely adopted for various objectives e.g. learning basic knowledge in early childhood development [1], learning mathematics in elementary school [2], and learning various subjects in universities [3, 4]. Prensky [5] stated that mostly people see learning as a painful activity that someone has to go through to acquire necessary knowledge and skills. Prensky [5] compared it with playing games. The clear distinction is that playing is a fun and engaging activity which gives motives to people to play it voluntarily. Therefore, Prensky strongly argued that conventional learning is outdated and the modern way of learning is through a real gameplay so that student will learn while having fun. More moderate views from Blunt [3] and Bellotti et al [6, 7], suggested the use of serious games as tools to support learning and to motivate students. According to Zyda [8], a serious game is “a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives”. Therefore, any game built to differ from pure entertainment can be considered as a serious game. Serious games have been extensively studied in recent years for their ability to enhance general development such as logic, memory, problem solving [9, 10], induce motivation in learning and improve academic performance [3, 11]. Serious games are adopted in academic curriculum mostly in the area of management and economics [3]. Hence, there are still many benefits of their application which could be further investigated. For instance, the benefit in collaboration development [1, 11, 12], motivational factors [2, 13, 14, 15], contents selection and authoring [1, 6, 7], genre [14, 15], strategy and competition, behavior [10], personalized learning, etc.

2. BACKGROUND

In recent years, all over the world, students are less interested in science, especially in computer science where the number of students is shrinking, 50% or more students who initially choose computer science soon abandon the study [16]. Computer science curriculum 2008 of Association for Computing Machinery (ACM) mentioned that fluency in a programming language is prerequisite to the study of most of computer science [17]. Being the core of study in computer science, it is essential to introduce computer programming in an attractive style while keeping students interests on the subject after entering university. According to Bayliss [18], three approaches exist to motivate students to learn programming: (1) building novice programming environment, (2) introducing programming contest, (3) game.

Alice2 [19] is one of example of novice programming environment to teach programming using drag and drop environment. This allows users to learn logic and
playing game (RPG), real-time strategy game, and 2D or actually playing real games such as Role
to program in specific object-oriented (OO) language.

An example is Colobot, a video game about colonizing a planet using robots that players have to program in specific object-oriented (OO) language. However, the real engagement with rich experiences and fun in learning can be provided through playing games due to its gameplay nature [5]. As a matter of fact, exploitation of games as part of educational curriculum has been investigated in literatures.

Some works have already been done in introducing games to computer programming either as assignments [20, 23, 24, 25] or actually playing real games such as Role Playing Game (RPG), Real Time Strategy game, and 2D adventure game [13, 16, 26]. Muratet et al [16] developed a real-time strategy (RTS) game about how to acquire resources and locate opponents using some given functions. Mohammed et al [26] developed an adventure 2D game that utilized local culture and provided interaction with the users via a cellular phone like graphical user interface (GUI). The GUI enables users to write code snippet per line until the required functionality achieved. Chang et al [13] built a mini and incomplete MORPG game with tasks in form of simple quests such as answering true or false, multiple choices, filling the blanks, and simple coding.

Game as assignments was done by Chen et al [24] by developing a restricted graphical like game of airplanes. Overmars et al [20] discussed the use of GameMaker as a development tool for teaching object-oriented (OO) concepts, whereas [23] use 3D computer game like tool to teach computer animation using C-Sheep and Open GL as renderer. Sung et al [25] developed GTA (Game-Theme Programming Assignment) to teach introductory programming using Microsoft XNA. All of these works wanted to attract students to learn programming.

The work of Muratet et al [16] is good for training logics and competition. However, it only provides limited number of functions which can be used. Moreover, it has merely five missions which are inextensible. There is no mechanism to introduce new levels or other type of missions or tasks. Other work by Mohammed et al [26] provides a good interaction by providing cellular phone like programming GUI. However, there is only coding quest that is supported. New type of tasks, additional tasks and stories require some amount of efforts to be implemented. Moreover, students have to learn the game mechanics before playing the game which diverge students from learning the actual skill i.e. programming. MORPG game developed by Chang et al [13] suffers from boredom since it lacks of dynamicity within the game i.e. attributes of players such as health and strength, and the virtual world are static. There is no reward in gaining more health or strength by solving given tasks.

Sung et al [25] provides good examples of game as assignments. This work introduced programming logic into some visualized problems which is good for beginner and intermediate level students to boost their interests to programming. At first, the GTA GUI induced experimentation, yet as students become more proficient, the advantage of visualized problem diminished. The fact that more experienced students prefer assignment without elaborate setups was confirmed by Guzdial [27]. Hence, it lacks of tasks variety with respect to different targeted skills.

In short, up to now there is no serious game designed and developed for learning computer programming. In particular contents with rich variety of tasks intended for different targeted skills while maintaining the motivational level high.

3. RESEARCH QUESTIONS

Creating well suited contents for different targeted skills is essential in learning. This can be observed from an extension program named computer science unplugged learning, intended for primary-aged children designed by Bell et al [28]. This program introduces knowledge on computer science such as data, information, and some basic algorithms and it has been perceived well by students. The activities in the program highly utilize mathematics and analogy represented by for instance cards and boards are considered to be very compelling and improving conceptualization for their age. Other observation is puzzle based learning used to enhance problem solving skill to university students [9]. The result was immediately observable in their problem solving skills development due to its attracting and intellectually challenging nature.

This paper aims at investigating ideas on contents for different type of users based on their knowledge and skills in programming for serious game for learning programming, as well as nurturing problem solving skills, to be effective, and to be motivational. Contents will be developed based on topics covered in computer science curriculum 2008 of ACM for computer programming [17]. This paper is a preliminary stage of developing a novel serious game which will lead to future research in
answering the possibility of computer game to support teaching, and learning programming in particular. The early steps will try to determine the following questions.

- Who are the target user group of the game?
- What prior knowledge and skills of each target has?
- What are the examples of contents for each targeted skill?

**4. THEORETICAL BASIS**

As aforementioned, learning is less motivating compared to playing game since playing game is a fun and engaging activity. Generally fun comes from activities that we enjoy to do and the more we do, the better we get. Prensky [5] listed a collection of fun activities defined by Garneu: beauty, immersion, intellectual problem solving, competition, social interaction, love, creation, power, discovery, advancement and completion, and application of ability. The key of engaging characteristic of games is the gameplay factor such as game’s rules, players’ choices, the difficulty of the road to success, the game balance which keeps the player in the “Flow Zone”, and a long term goal with clear short term goals. Depending on the game genres, the engaging activities can be as it follows.

- Puzzle game: the physical and mental challenge within the puzzles
- First Person Shooter (FPS) game: the opponents’ speed and abilities.
- Strategy game: the available options and tactics to be employed.

Another example of engaging activity is mind game by asserting uncertainty of information in a class since students need to sort out the correct information from the false.

The difficulty of road to success and ability to keep the game balance according to the flow zone is the important aspect in creating contents for game. On pedagogical point of view, incongruity theory by Lankveld et al [29] stated learning may takes place if there is low positive difference between environment complexity against the internal mental model of the context, see Figure 1. Moreover, Vygotsky, a constructivism theorist, stated that in order learning to be optimal, the knowledge and skills should be kept in the Zone of Proximal Development (ZPD) [30]. This means learners should be kept in a narrow zone where the knowledge and skills is neither too difficult nor too easy to master under a proper guidance. These theories emphasized on the conformity of learning material with the prior knowledge of the learners. Another theory is Sweller’s delivery strategy called Cognitive Load Theory (CLT) [31]. Sweller divided the memory of the brain into two i.e. short term and long term. Short term memory is essential to process perceived information prior to be stored in long term memory as knowledge. Yet, the short term memory is limited and overloading it inhibits learning. Therefore, information should be delivered in gradual manner which will give learners time to structure their knowledge and develop understanding. Providing partially worked example called Faded Working Example (FWE) for the learner to be completed is one way in CLT.

In general, the idea is that the game/contents should leverage the player’s personal connects (previous knowledge and experience), in order to better support knowledge acquisition.

**5. REQUIREMENTS ON DESIGNING CONTENTS**

Prior to providing contents with various difficulty levels to better suit with the skill of the users, the target user groups should be defined. Afterward, in each target group, various tasks with multi-level challenges can be introduced. This will allow the game to adjust with the progress of the users. There are four dimensions of multi-level challenges can be employed [2] as follows.

- Task difficulty: prior knowledge needed to solve problems
- Task complexity: the number of sub-tasks which compose the main task
- Resources: what users have to complete the tasks e.g. time constraints
- Opponents: the number of opponents and their abilities

Subsequently, a delivery scenario can be developed in order to correctly cast the suitable tasks for learners. This is called adaptive learning i.e. a feature will be made by taking account the user profiles in term of their corresponding user groups and their performance which is highly suggested by Frazer et al [15]. However, this feature will not be discussed further. Instead, some examples of tasks with different targeted skills with their multi-level challenges will be presented.
Target User Groups

Prior to developing the contents, targets should be defined. Computer science unplugged [28] intended as an extension program for primary aged children to introduce basic computer science are considered to be very compelling. Therefore, in order to attract prospective students to computer science, this type of approach is interesting to be explored as part of the contents where pre-university students are mostly the target. In addition, students who already enrolled in computer science major should be retained and motivated. GTA [25] are good examples for beginner students to keep them interested with the computer science. As students become more proficient, more challenging should be exploited [27]. Programming contests [21] or programming from scratch using Faded Working Examples (FWE) described in Section 4 may provide these challenges with topic of higher level problem solving. Therefore, there are three targets that will be on focus as shown by Table 1.

<table>
<thead>
<tr>
<th>Targeted Users</th>
<th>Knowledge and Skill Level</th>
<th>Tasks Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-university students</td>
<td>High school mathematics and algebra No previous knowledge</td>
<td>• mathematical based activities using cards and boards</td>
</tr>
<tr>
<td>First year university students</td>
<td>No previous knowledge</td>
<td>• support curricular activities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• low to medium level problem solving and logic</td>
</tr>
<tr>
<td>Second year and upper university students</td>
<td>Basic programming</td>
<td>• support curricular activities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• higher level of problem solving and logic</td>
</tr>
</tbody>
</table>

Computer Programming Curriculum

Given defined targets in Table 1, the tasks for the second and the third targets should be based on computer science curriculum. According to joint ACM and IEEE curriculum, fundamental programming stresses on fundamental programming concepts, basic data structures, and algorithmic processes to reach fluency in programming language (PL) [17]. Table 2 shows the topics covered by programming fundamentals.

<table>
<thead>
<tr>
<th>Courses</th>
<th>Materials</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM/IEEE: Programming fundamentals</td>
<td>Fundamental constructs</td>
<td>• Basic syntax and semantic, variables, types, simple I/O, control structures</td>
</tr>
<tr>
<td></td>
<td>Algorithmic problem solving</td>
<td>• Problem solving strategies, role of algorithm</td>
</tr>
<tr>
<td></td>
<td>Data structures</td>
<td></td>
</tr>
</tbody>
</table>

| UNIGE: fundamentals of computer  | Introductory concepts       | • Computer architecture, hardware, software, operating system, compiler, applications, binary encoding of information |
|                                 | Software design             | • problem analysis; concept of algorithm, software design methodologies |
|                                 | PL C++                      | • structure of a program in C++, basic data types, variables, control structures, pointers, dynamic memory allocation, functions and recursion, I/O, debugging |
|                                 | Data structures             | • queues, stacks, lists, trees and their operations: access, insertion, deletion, sorting |
|                                 | Algorithms                  | • search and sorting algorithms, fusion algorithms, recursive algorithms |
|                                 | OO programming              | • classes, objects, messages and methods, design of simple classes, constructors and destructors, overloading, of functions and operators, inheritance, polymorphism |

Joint ACM and IEEE curriculum is a suggestion of the materials needed in order students to have equal competences. Therefore, it does not strictly have to employ all the materials in the lesson. Some materials e.g.
secure programming can be excluded if the competence output does not require that specific areas [17]. A comparison for fundamental of computer course in University of Genoa (delivered in two semesters) can be observed in Table 2. Both curriculums have similar materials to develop fluency in programming which mainly consist of fundamental concepts, basic algorithms, data structure and OO programming.

**Topics for Each Target User Group**

Given the target users, and teaching materials in Table 1 and Table 2, we have preliminary constructed the topics for each target user in Table 3.

<table>
<thead>
<tr>
<th>Target Users</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-university students</td>
<td>Binary numbers, data representation, text and image representation and compression, simple algorithms</td>
</tr>
<tr>
<td>First year university students</td>
<td>All materials for pre-university students, PL, data structures, algorithmic problem solving, recursion</td>
</tr>
<tr>
<td>Second year and upper university students</td>
<td>All materials for 1st year students, OO programming, algorithms complexity</td>
</tr>
</tbody>
</table>

Bloom taxonomy classified the learning objectives into three domains: cognitive, affective, and psychomotor [33]. Within each domain, learning at higher level is dependent on having attained prerequisite knowledge at lower level. This taxonomy justified the topics selection presented in Table 3. The three domains of learning objectives will be further investigated to determine the level of taxonomy of each task given the domains. Nonetheless, an illustration of tasks to support cognitive domain is as follows.

- **Pre-university students:** to give ideas on computer science. Hence, it will evolve around knowledge and comprehension (level two in the cognitive domain [33]).
- **First year university students:** previous level added with application, problem solving, and logical thinking (level four in the cognitive domain [33]).
- **Second year and upper university students:** previous level added with synthesis and evaluation (level six in the cognitive domain [33]).

The tasks can be in form of true/false, multiple choice, pair matching, put in the right place, and find the error. However, more variety of tasks will be considered in the future. In each targeted skill, multi-level challenges can be implemented by defining tasks difficulty level, tasks complexity, and resources restriction to complete the tasks. An important thing which should be noted is the Bloom taxonomy merely drives the topics delivered, not how the game will progress.

## 6. CONTENTS ILLUSTRATION

Based on the topics with respect to different targets above, examples are proposed as preliminary idea on the contents of the game for computer programming. Basically the game will be designed to have a low learning curve for mastering it and motivate the users to play using Csikszentmihalyi’s flow theory [34] instead of using conventional Bloom taxonomy driven learning [33].

### Pre-University Students

The objective of this segment is to introduce computer science as an attractive field to learn. One of possible ways is through mathematical based activities [28] which are syntax free as follows.

1. **Activity 1: Binary number**
   - Given a group of jars with the following volume in liters (L): 1, 2, 4, 8, 16 (Figure 2). How to obtain 5L from those jars if ‘?’ can only be represented by 0 or 1 which means you don’t use the jar or you use the jar, respectively.
   - Similar as before, how you can obtain 9L and 30L.

   ![Figure 2. Group of jars](image)

   - **This activity gives an idea to students about binary representation and its use in computers in representing data and information. For instance, the answers of the question are 00101, 01001, 11110 for 5L, 9L, and 30L, respectively. It allows students to understand the amount of bits that required in representing certain number in the computers. Increasing complexity can be done by, for instance, adding jars that represent additional bits used. Also, this type of activities can be expanded to introduce the concepts of ASCII codes.**

2. **Activity 2: Understanding logic and algorithm**
Five cards are drawn from the stack which results in the numbers in Figure 3a. Then, the cards are shuffled and put in face down position (Figure 3b). The task is to sort the cards in Figure 3b in increasing order. However, only two cards are comparable at a time. What is the minimum number of comparisons until the cards successfully sorted in order?

This gives an idea to students about sorting algorithm. The activity requires interactive graphical user interface (GUI) to flip card and compare two cards at a given time instance. This activity can be sliced into minor steps or expanded to various searching algorithms e.g. binary searching, selection sort, and hash table for different level of challenges.

**First Year University Students**

The objective of this segment is to retain the interest of the new students in computer science, in particular programming course. Puzzles and visualized tasks may help them understand the materials as follows.

1. **Task 1: Data structure**

   Figure 4 visualizes an empty array, a filled array, and a piece of codes that represent the empty array used within one pass loop structure. The task is to fill the elements of the empty array as the elements of the filled array by using the loop.

   The purpose is to give an idea to students about value assignment in array and problem solving using simple mathematical operations. The extension of this task can be different type of mathematical operation, sorting, or two arrays addition or subtraction. Adding the size of the array is one of examples in increasing complexity. Also, worked example can be removed which make the problem more complex.

2. **Task 2: Algorithm analysis**

   You are given the following code in Java.

   ```java
   public static void main(){
       k=0;
       for(int i=0;i<10;i++){
           k++;
       }
       System.Out.println(k);
       System.Out.println(i++);
   }
   ```

   Which one is the correct output?
   A. 10 and 11
   B. 9 and 11
   C. 10 and 12

   The purpose is to allow students to perform syntax and semantic analysis. The level of difficulty can be increased by adding nested loop. Adding possible answers or variables to be observed are possible to increase complexity. Resource restriction can be in the amount of time to answer the question.

**Second and Upper Year University Students**

This segment differs from previous segments in term of the complexity, since target users are expected to have acquired basic programming. This target skill needs more challenging problems to retain the amusement such as complex puzzles used in programming contests and OO programming as follows.

1. **Problem 1: Mathematics**

   Find the solution for equation: \(4x^2 + 12x - 20 = 0\) by using all coefficients as input.

   This problem assesses the mathematical capability of the students as well as their analytical skill to observe all possible cases in the problem. This problem can be sliced into several steps using Faded Working Example (FWE) to assist students. However, the more advanced the students, the less assistance will be provided. Competition can be induced by recording the best time used by students beside the correctness of the program.

2. **Problem 2: OO Programming**
Given two instantiated stack objects A and B with their corresponding operation i.e. *push* and *pop* (Figure 5), sort the numbers in stack A using stack B in decreasing order.

The purpose is to allow students to understand a type of data structure i.e. stack and utilize it for processing information. This problem can be sliced into several steps using Faded Working Examples (FWE) depending on the students’ level.

For both problems above, FWE can be in form of how the problems are presented. For instance, students with lower skill level have to arrange a given set of unordered pieces of codes, whereas students with higher skill level have to complete a given program by filling several lines of missing codes. Moreover, visualization such as elements’ movement in the stacks (Problem 2) may be provided as hints for different skill level.

7. CONCLUSION AND FUTURE WORKS

This paper served as the first stone for developing serious games to support learning in computer programming. It defined different target user groups of computer programming games with respect to the users’ prior knowledge and learning objective. Also, preliminary examples of tasks as part of the contents for educational game were presented. Utilizing familiar subjects such as mathematical based activities are proposed for pre-university students to introduce computer science and attract them to study in computer science. Puzzles and visualized tasks are applied for first year to retain their interests in studying computer programming, whereas complex puzzles akin to problems in programming contests are employed for second and upper year university students.

Future work is planned to develop a richer variety of tasks for the contents which supports Faded Working Example (FWE), and to design the tasks and user model, the gameplay with rich interaction, the assessment, and personalized delivery for the game. Available technologies will be explored to realize the game and tested to students in computer programming course. Indicators represent students’ skills and motivation in learning will be constructed and later on measured to confirm the expected impact on students’ perception in learning computer programming with the objective of the game. Afterward, automated learning can be developed which will propose personalized learning for each different user. Some sensors and actuators may be incorporated in order to acquire users’ needs and to provide feedbacks within the game. Also, the results of playing game will be evaluated to draw research finding.

8. ACKNOWLEDGEMENT

This work is supported by the European Commission and Education, Audiovisual and Culture Executive Agency (EACEA) in form of doctoral research grant namely Erasmus Mundus Interactive and Cognitive Environment (EM-ICE) PhD. Therefore, we would like to express our gratitude for providing us such grants. Moreover, we would like to thank the non-blind reviewers of this work: Dr. Wei Chen and Prof. Matthias Rauterberg (TU Eindhoven, the Netherlands), Prof. Carlo Regazzoni (University of Genoa, Italy), and Dr. Mahendrawati Erawan (ITS Surabaya, Indonesia).

9. REFERENCES


Proceedings of the 5th IEEE international conference on Computational Intelligence and Games.


