Towards a 3D Virtual Game for Learning Object-Oriented Programming Fundamentals and C++ Language

Theoretical Considerations and Empirical Results

Fahima Djelil[1], Benjamin Albouy-Kissi[1], Adélaïde Albouy-Kissi[1], Eric Sanchez[2] and Jean-Marc Lavest[1]

[1] *Laboratoire Institut Pascal CNRS-UMR 6602, Université Blaise Pascal, Clermont-Ferrand, France*

[2] *EducTice, Institut Français de l'Education, Ecole Normale Supérieure, Lyon, France*

Keywords: Object-Oriented Programming, Game Based Learning.

Abstract: Object-Oriented Programming (OOP) paradigm is one of the most common paradigm in introductory programming courses. However, novices often have difficulties to understand the basic concepts which are of a high level of abstraction. Either tangible and virtual constructive games provide the students with a more familiar way for learning programming. This paper applies a construction game metaphor approach for learning OOP concepts and C++ syntax. After introducing some tangible and virtual constructive games for learning programming, we present an experimental prototype of a new 3D virtual game for learning OOP called PrOgO as well as the results of an experiment conducted with beginner student using PrOgO.

1 INTRODUCTION

Learning programming can be perceived difficult from novices (Yan, 2009; Overmars, 2004). A novice student has to face a primary challenge which is problem solving, he has to understand how a computer program runs and has to learn a specific syntax in which his program will be transcribed. Abstract concepts understanding is another challenge. The basic Object-Oriented Programming (OOP) concepts which are on a higher level of abstraction have become the most common programming concepts for introductory programming courses, and the transition to the OOP paradigm has proven to be more difficult than expected (Börstler et al., 2008). Therefore, a lot of systems have been designed to support the acquisition and development of programming concepts. They aim to take abstract programming concepts to the real world for learners. They are based either on tangible interfaces or on virtual simulations and games that connect with the learners interest. This has proven to be very engaging and empowering for novices (Cooper et al., 2000; Kölling, 2010).

In this work, we apply a construction game metaphor approach for learning OOP. Our ultimate goal is to provide novice students with realistic and relevant representations that will help them in learning OOP concepts and C++ programming syntax in an entertaining way. For this purpose, we present an experimental prototype of a new three-dimensional (3D) virtual game for basic OOP concepts such as class, object, object creation, method invocation, parameter passing and object destruction. First we present a literature review on tangible interfaces and virtual environments for learning programming (section 2), then we describe our first prototype of the game we called PrOgO (section 3) and finally, we describe an experiment we conducted with beginner students using this prototype, we will proceed to interpret the results (section 4) and to extract the corresponding conclusions (section 5).

2 THEORETICAL BACKGROUND

There have been several tangible interfaces for learning programming which aim to bring the abstract concepts of programming closer to the real world, to be more easily graspable (Jetsu, 2008). These interfaces consist of tangible components i.e. physical and mechanical that the user can assemble to construct program's sequences. One of the first tangible programming interfaces was Radia Perlman's Slot Machine (Perlman, 1976) which was based on the LOGO pro-

Djelil F., Albouy-Kissi B., Albouy-Kissi A., Sanchez E. and Lavest J..
Towards a 3D Virtual Game for Learning Object-Oriented Programming Fundamentals and C++ Language - Theoretical Considerations and Empirical Results.
DOI: 10.5220/0005475802890294
In *Proceedings of the 7th International Conference on Computer Supported Education* (CSEDU-2015), pages 289-294
ISBN: 978-989-758-108-3

gramming language. LOGO was one of the most widely used programming languages for beginners (Jetsu, 2008), allowing students to use the computer to control a robotic turtle to follow their instructions (Papert, 1980). Slot Machine allowed students to control either a real mechanical robot or a screen based simulation of such a robot (Perlman, 1976). AlgoBlock is another interface that is somewhat similar to Slot Machine. It consists of a set of physical blocks that connect to one another to form a program (Suzuki and Kato, 1993). Each block corresponds to a single command in the LOGO programming language. The term tangible programming language was invented by Suziki and Kato in order to describe the AlgoBlock collaborative programming environment (Suzuki and Kato, 1993). Topobo is another example of this kind of interfaces that was designed to learn programming thinking (Raffle, 2004), it consists of a set of active and passive blocks which can be combined together to assemble and animate robotic structures. The active blocks are embedded with a kinetic memory which allows recording and playback of physical motion of the constructed assembly. This constructed assembly is animated by moving passive blocks that are connected to the active ones. After the animation has been created, it can be played back by pressing a button (Raffle, 2004).

Several environments have been designed with a view to reduce the complexity of programming learning for novice students and increase their motivation. These environments use graphic representations to introduce abstract programming concepts and allow students to create games and animations by manipulating programs. Alice (Conway et al., 2000) and Greenfoot (Kölling, 2010) are examples of such environments, which strive to engage students by allowing them to write programs about games, stories and simulations. One of the important characteristics of Alice and Greenfoot is a design that explicitly visualizes fundamental concepts of OOP in a realistic and a meaningful context. Students do not typically start by manipulating source code, they rather start by creating objects by selecting graphics which represent classes. Once an object has been created, it can be placed into a 2D (Greenfoot) or a 3D (Alice) scene, when the object is selected the user can see all its methods which constitute all its available actions it can perform. Once a method has been selected, the user can see immediately the effect of its execution onscreen. After the students are introduced to the fundamentals of OOP, scenarios that have been already implemented are presented to them enabling easy manipulation of programs in their syntax. This characteristic is very interesting for novice students, since it has been proved (Cooper et al., 2003; Kölling, 2010) that this helps novices to avoid syntax errors while focusing on understanding abstract concepts before manipulating source code. According to (Utting et al., 2010), these environments' principal harks back to Logo's turtle. Since they reify objects so that the result of command execution is visible as the position, size, rotation, and other visible state and behavior of the object changes. These environments provide high-level commands like move and hide low-level details such as graphics primitives.

By analogy with any construction game such as Topobo, a screen based simulation of such a game that allows students to manipulate 3D graphics to create and animate robotic structures, whereas each graphical elementary block represents an object, would be relevant to introduce OOP concepts. Indeed, an object is an active entity within a program, including a set of attributes (knowledge) and methods (skills) that act on its attributes to achieve some functionalities which are the objectives of the program.

3 A 3D VIRTUAL GAME PROTOTYPE FOR LEARNING OOP FUNDAMENTALS AND C++ PROGRAMMING LANGUAGE

A constructive game dedicated to learning OOP fundamentals, should enable students creating objects, visualizing their state and behavior, interacting with them by executing their methods and make them communicate with one another. Direct manipulation of graphics representing object-oriented concepts can help students, as in real life, to think in terms of objects. The student should be able to create as many class instances as he likes. The concept of inheritance can be materialized by creating new objects based on other objects having the same behavior with new attributes or the same attributes with new behaviors. The concept of composition can be represented by objects that contain other objects. In order to enable the student to understand more easily the implementation of the abstract concepts in a specific syntax such as C++, all his manipulations executed directly on the graphics should be interpreted into source code. Since the representations are very precise, each student's action can be interpreted into a C++ instruction. This will constitute a complete program, that the student can visualize and modify in order to practice programming, after the abstract concepts are well understood.

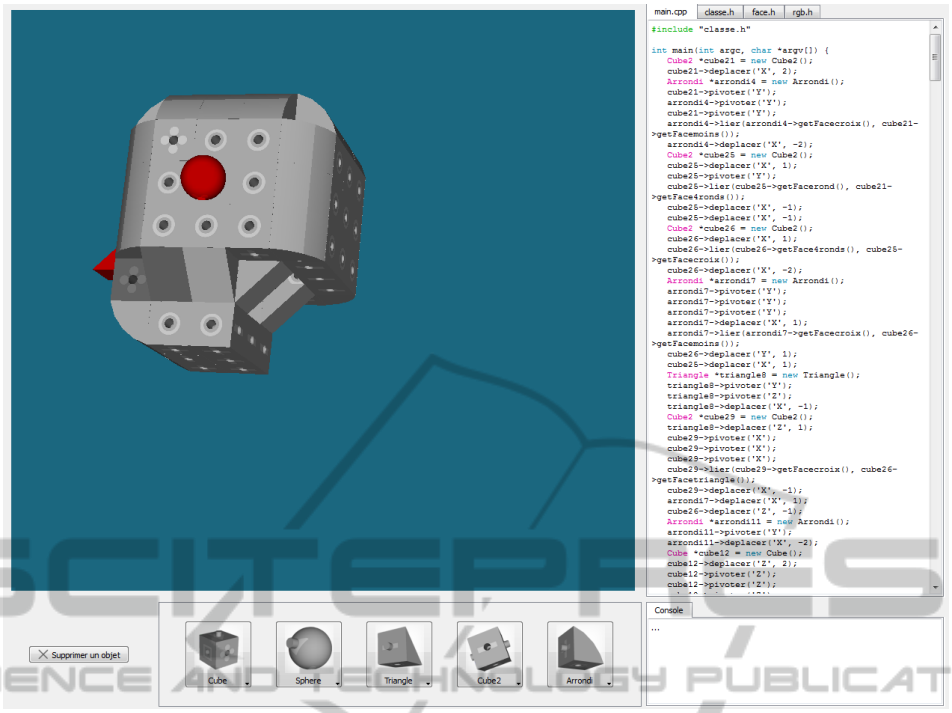


Figure 1: PrOgO main window showing a realization example.

This design principle provides students with a primary introductory step to OOP concepts, through fun and realistic 3D graphics, before manipulating source code. This also allows students to express their own creativity, and provides them with an engaging learning tool.

PrOgO (Object-Oriented PrOgramming) is an experimental prototype which has been developed as a proof of concept of such a game. This prototype generates C++ source code corresponding to direct manipulations of 3D graphics. Figure 1 shows the interface of PrOgO with an example of a student's realization. The interface comprises 3D elementary blocks representing classes that the student can instantiate to obtain objects, a 3D scene to place in the objects that have been created, and a separate tab anchored on the right side of the 3D scene to display the C++ code that is being generated during the direct manipulations of the 3D graphics. Once an object is created and placed into the 3D scene, the student can view all its methods, and invoke them. Once a method is executed, its result on the state of the object is immediately visualized, and the corresponding C++ code is automatically generated. Each object has three elements of state that are automatically visualized on screen: a position which is specified in x, y, z coordinates, a rotation and a color. Separate objects can connect to one another and assembled objects can be disassembled. Each of these can be manipulated trough method calls to create a robotic or a mechanic structure. Note that the code generated is not yet editable.

4 EXPERIMENTAL RESULTS

In the Institute of Technology of le Puy-en-Velay (University of Auvergne, France), the course Fundamentals of Computer Science is imparted in the first year of the Degree in Digital Imaging. This course consists of two main topics: Introduction to algorithms and programming, and OOP fundamentals. The first topic covers the basics about programming on C++: data types, variables and operators, control flow statements, arrays and pointers, procedures and functions, dynamic allocation. The students, at this stage, are initiated to C++ procedural programming and don't have knowledge about OOP. The second topic is an introduction to OOP in C++: classes and objects, encapsulation, composition, inheritance, polymorphism, etc.

We have experimented PrOgO with the first year students in Digital Imaging Degree. This experimentation aims to identify potential design, ergonomic and pedagogical issues of the game, and intends to provide a conceptual basis for a future evolutive prototype on the basis of the students' expectations in terms of learning and entertaining aspects.

This was conducted after the students have completed the first topic of the course, and once they started the second one. They have been introduced for the first time to some elementary concepts such as class, object, encapsulation and method invocation. The participants were a total of 48. Their average age was 18.7 years and all except one reported to play to video games very frequently (31 participants play every day and 16 of them play several times a week). They also mostly reported not to play to serious games (37 participants don't play against 11 who play regularly). We collected pictures of the graphical realizations (Figure 2) as well as the corresponding generated source code files, to determine whether the students were getting involved into the game. We also recorded all the students' actions within logs including objects creation and destruction and method calls within the 3D scene, as well as code selection within the tabs containing the code which is automatically generated and the code which is already implemented that includes the declaration of the available classes. This have been made in order to know whether the students make the effort to engage in understanding the relationship between the 3D graphics and the generated code, rather than only manipulating the graphics in the 3D scene. We also used an attitude survey which is focused on participant's experience and opinions about using PrOgO in the future for learning other OOP concepts. They were asked to rate the pedagogical and entertaining aspects of the game, to indicate whether they have improved their understanding of the concepts of class, object, method invocation and C++ syntax related to these concepts, and what aspects of the game they would like to be improved.

From the collected graphical realizations (Figure 2), we can observe that some students clearly succeeded to construct complex and meaningful structures. This indicates that the students were quiet engaged in the game and motivated to accomplish the game's objective. From the generated interaction logs, we observed that the students were mostly focused on manipulating graphics in the 3D scene. The logs have shown that the actions that have been mostly performed are objects instantiating and objects methods calls such as assemble(), translate() and rotate(). Some students were also interested to analyse the generated code, since the logs included actions such as "select code" and "select tab".

From the attitude survey, we collected the students' opinions expressed upon four levels (not at all, sparsely, quite, a lot) about the amusing and pedagogical aspects of PrOgO, as well as improvement of their understanding of the presented concepts. Table 1 summarizes the obtained results. We have linked

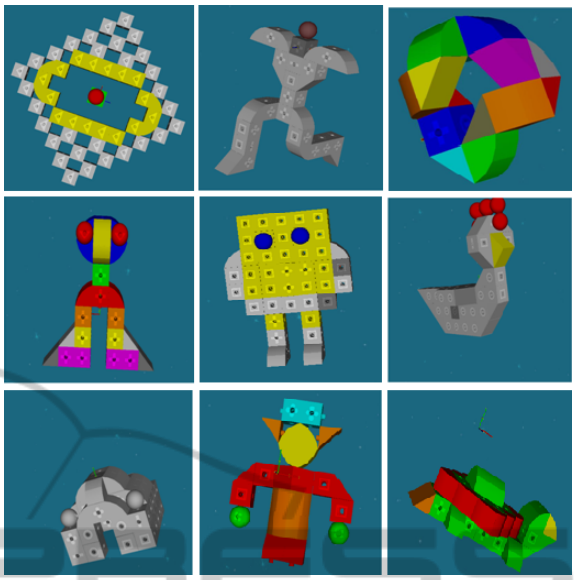


Figure 2: Some students' realizations.

the amusing judgement to the student's frequency of play to video games and the pedagogical judgement to the student's frequency of play to serious games and we compared the obtained results. We concluded that the students have mostly found PrOgO sparsely or quite amusing and quite pedagogical, regardless of the fact that they play every day or several times a week to video games, and the fact that they play or not to serious games, since the obtained results are not very different from the separate students categories, as it is reported in table 1: 51% of the students who play to video games everyday and 39% of those who play several times a week stated that they have found PrOgO "sparsely" amusing. 39% of the students who play everyday and 44% of those who play several times a week found it "quite" amusing. They are totally 47% to find it "sparsely" amusing and 41% to find it "quite" amusing. Moreover, no one of the students who play to serious games has found it "not at all" pedagogical, 58% rather found it "quite" pedagogical and 34% found it "sparsely" pedagogical, those who don't play to serious games have mostly found it "quite" pedagogical (67% against 22% who found it "sparsely" pedagogical).

We have also concluded that the students have mostly sparsely or quite improved their understanding of the presented concepts (class, object and related C++ syntax), regardless of the fact that they previously understood these concepts in classroom or not. As it is shown in table 1, the students who said that they have already understood these concepts in classroom have though improved their understanding (44% stated that they have "sparsely" improved

Table 1: Participants' opinions about PrOgO collected in the attitude survey.

comparison item	participants	not at all	sparsely	quite	a lot
how is PrOgO amusing?	students who play to video games everyday	10%	51%	39%	0%
	students who play to video games several times a week	17%	39%	44%	0%
	all	12%	47%	41%	0%
how is PrOgO pedagogical?	students who play to serious games	0%	34%	58%	8%
	students who don't play to serious games	3%	22%	67%	8%
	all	2%	25%	65%	8%
how did you improve your understanding of the concept of class?	students who think they have already understood in classroom	16%	44%	36%	4%
	students who think they have not understood in classroom	8%	59%	33%	0%
	all	12 %	51%	35%	2%
how did you improve your understanding of the concept of object?	students who think they have already understood in classroom	26%	37%	26%	11%
	students who think they have not understood in classroom	3%	54%	43%	0%
	all	12 %	47%	37%	4%
how did you improve your understanding of the syntax related to the concepts of class and object?	students who think they have already understood in classroom	39%	33%	28%	0%
	students who think they have not understood in classroom	3%	62%	32%	3%
	all	16 %	51%	31%	2%

their understanding of the concept of class, 37% have stated the same regarding to the concept of object, and 33% regarding to the syntax of these concepts). There are also a number of them who stated "quite" improving their understanding (36% regarding to the concept of class, 26% regarding to the concept of object, and 28% regarding to the syntax of these concepts). Most of the students who thought not to understand before in classroom have also presented quite similar percentages (59% stated improving "sparsely" their understanding of the concept of class, the percentages for the concepts of object and syntax are respectively 54% and 62%). A lot of them have reported having "quite" improved their understanding of these concepts, the percentages are respectively (33% , 43% and 32%).

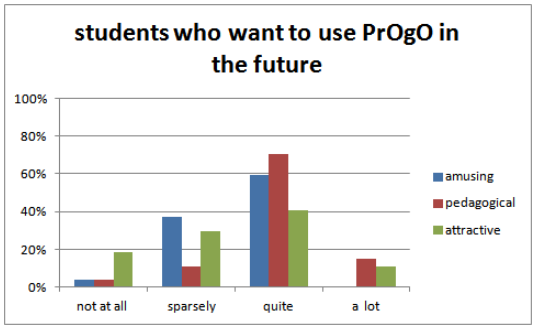
We gathered some students' statements regarding to what they learned, during the experiment such as: "I have learned how to declare an object and how to make method calls", "I understand more about the C++ syntax, and this tool can help me to review some concepts at home for example", "I understand better the syntax related to object instantiation and method calls, especially about pointers", "I understand better the relationship between the class and the object".

We have also collected the students opinions about using PrOgO in the future for learning additional concepts, and linked the results to their judgements regarding to weather they found it amusing, pedagogical and attractive. We concluded that the students who want to use PrOgO again are mostly those who found it quite amusing, quite pedagogical and quite attractive, and those who don't want to use it again are those who found it sparsely amusing, sparsely pedagogical and sparsely attractive (Figure 3).

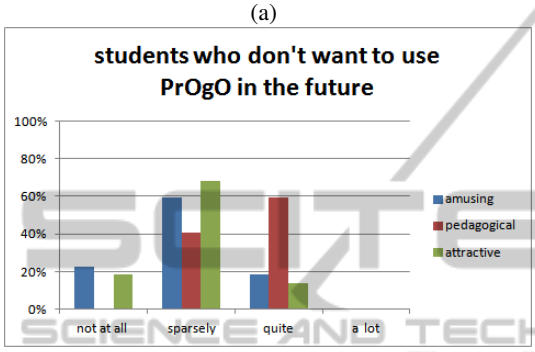
Furthermore, we asked the students what aspects of the game they would like to be improved. The answers were mostly about ergonomic and learning aspects. Most of the students indicated that they would like more graphical elements, and more options such as having the possibility to construct other objects from the available ones, in order to be able to manipulate groups of objects. Most of the students expressed an interest in modifying the generated code, to interact more easily with the virtual environment and to have more freedom in their actions.

5 CONCLUSIONS

In this work, we have applied a construction game metaphor for learning OOP and C++ programming



(a)



(b)

Figure 3: Using PrOgO in the future according to weather the students found it amusing, pedagogical and attractive: a) students who want to use PrOgO again. b) students who don't want to use PrOgO any more.

language. We have presented a first experimental prototype of a 3D virtual game called PrOgO by which abstract OOP concepts are represented in a meaningful and a visible way, and we experimented it with the first year students in Digital Imaging Degree.

From the analysis of results presented in the previous section, we concluded that although PrOgO is at the prototype stage, the students have mostly found it quite pedagogical and amusing. They also mostly sparsely improved their understanding of the concepts of class, object, object instantiation, method calls and C++ syntax related to these concepts. Moreover, more than half of them stated that they want to use PrOgO in the future for learning additional concepts, and most of them have expressed the wish to have more options regarding to the interaction interface and the content of the game, such as having the possibility to modify the generated code, which is something we have already planned to do before. Although this work is a preliminary study, such results are significant. In this sense, a stable release of the game could have a positive influence on the learning of OOP basics as well as C++ programming language.

Actually, we work on the design and the development of an evolutive prototype dedicated to tactile interfaces such as tangible tabletops and tablets. This work allowed us to validate our design principal, especially the construction metaphor on which the game is based on, and the students feedback allowed us to confirm the extensions we have already planned for a future evolutive prototype. A future work will also include the study of the impact of PrOgO on the students motivation and learning.

REFERENCES

Börstler, J., Nordström, M., Westin, L. K., Moström, J.-E., and Eliasson, J. (2008). Transitioning to OOP/java - a never ending story. In *Reflections on the Teaching of Programming*, pages 80–97. Springer.

Conway, M., Audia, S., Burnette, T., Cosgrove, D., and Christiansen, K. (2000). Alice: lessons learned from building a 3D system for novices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 486–493. ACM.

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, volume 15, pages 107–116. Consortium for Computing Sciences in Colleges.

Cooper, S., Dann, W., and Pausch, R. (2003). Teaching objects-first in introductory computer science. In *ACM SIGCSE Bulletin*, volume 35, pages 191–195. ACM.

Jetsu, I. (2008). Tangible user interfaces and programming. Master's thesis, University of Joensuu- department of computer science and statistics. page 70.

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14.

Overmars, M. (2004). Learning object-oriented design by creating games. *Potentials, IEEE*, 23(5):11–13.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children.

Raffle, H. S. (2004). *Topobo: a 3-D constructive assembly system with kinetic memory*. PhD thesis, Massachusetts Institute of Technology.

Suzuki, H. and Kato, H. (1993). Algoblock: a tangible programming language, a tool for collaborative learning. In *Proceedings of 4th European Logo Conference*, pages 297–303.

Utting, I., Cooper, S., Kölling, M., Maloney, J., and Resnick, M. (2010). Alice, greenfoot, and scratch - a discussion. *ACM Transactions on Computing Education (TOCE)*, 10(4):1–17.

Yan, L. (2009). Teaching object-oriented programming with games. In *Sixth International Conference on Information Technology: New Generations (ITNG'09)*, pages 969–974.