Developing a 3D Interactive Tool for Learning OOP Concepts

Dr. Arwa Abdulaziz Allinjawi, King Abdulaziz University, Saudi Arabia
Wejdan Eissa Moussa, King Abdulaziz University, Saudi Arabia
Raniyah Mutlaq Almalki, King Abdulaziz University, Saudi Arabia
Maryam Abdulrahman Alamoudi, King Abdulaziz University, Saudi Arabia

The Asian Conference on Education & International Development 2016
Official Conference Proceedings

Abstract
Object Oriented Programming (OOP) recently became the most influential programming paradigm. Several studies indicated deficiencies in learning introductory OOP courses. In King Abdulaziz University (KAU), Jeddah-Saudi Arabia, during the second semester of the year 2014-2015¸ a survey that had been designed and distributed to female students of introductory OOP course. The result of the survey showed that students faced difficulties in understanding OOP, specifically 47% of students faced difficulties in understating Polymorphism concept while 59% faced difficulties in implementing Polymorphism. Some of the visualization tools visualize the execution of programs, using visual hints, and interactivity to increase motivational aspects and to improve the students' understanding of programming concepts. This paper showed the development of "OOPVisual". It is a 3D interactive visualization tool that simulates OOP concepts to help students with their understanding.

Keywords: Object Oriented Programming; Polymorphism; Visualization; 3D Interactive Tool; Animation; Drag-and-Drop Method.

Introduction

Programming is the heart of computer science. Therefore, often CS programs start with introductory programming courses. Almost every university teaches OOP somewhere in its CS curriculum (Lahtinen, et al., 2005). Studies (Biju, 2013; Goosen & Pieterse, 2005; Sheetz, Irwin, Tegarden, Nelson & Monarchi, 1997) have emerged in recent years to prove that some novice programmers have difficulties in learning OOP concepts. In King Abdulaziz University (KAU), Jeddah-Saudi Arabia, during the spring of 2015¸ it has been analyzed through a statistical study conducted on female students of introductory OOP course that students faced difficulties in understanding and implementing OOP concepts, specifically with Polymorphism. On the other hand, some of the visualization tools visualize the execution of programs which they often used to increase the motivational aspects of programming courses. Such tools introduce animation, visual hints, sounds, and interactivity to employ several different learning styles which support the student activity.

Learning programming concepts with the engagement of visualization tools would help the students focus on the actual programming task instead of wasting time in syntax errors. In addition, these tools will enrich the programming courses to be actively motivated courses to students (Nevalainen, Seppo, & Sajaniemi, 2006).

In this paper, section 2 addresses some universal OOP difficulties faced by students. It also introduces some of the available visualization tools and provides a brief comparison between the proposed OOPVisual tool and the other tools. Section 3 discusses the difficulties of OOP concepts for female students of King Abdulaziz University. Section 4 focuses on the proposed OOPVisual tool, a 3D interactive visualization tool, that simulates OOP concepts using drag and drop technique and selecting from menus. This tool will allow students to be more comfortable with programming without dealing with syntax errors and complex design techniques. Furthermore, allows students to focus on programming concepts rather than the tedium of debugging code. Finally, the paper summarizes the survey results and illustrates the future work of the OOPVisual team.

Literature Review

I. OOP Difficulties

In the programming field, understanding OOP concepts is a difficult task for students. Some researches (Biju, 2013; Goosen & Pieterse, 2005; Sheetz, Irwin, Tegarden, Nelson & Monarchi, 1997) claim that many novice programmers lack understanding some of OOP concepts like Classes, Constructor invocation, Encapsulation, Overloading, Object creation, Inheritance relationships between classes, Polymorphism, and other OOP concepts.

Some of the reasons that led to the lack of understanding OOP concepts as Biju (2013) suggested, is having a previous experience in procedural programming which makes it more difficult to learn and understand OOP.

Another study (Oliveira, 1998) shows some of the theories of OOP are based on the representation of the real world, abstraction, re-usability, and inheritance are as difficult for some students to comprehend. Moreover, Bashiru and Joseph (2015) suggested other reasons such as when executing a program, the student may not understand what exactly happens inside the computer. Some students face difficulties in understanding how the OOP program can solve a given problem. Furthermore, some of the available tools for learning and teaching OOP are difficult to use.

II. Visualization Learning Tools

Nowadays, the traditional techniques of teaching and learning programming must be redesigned to be more suitable for the new generation of students. Animations and computer games innovative are increasingly popular in becoming teaching tools that make education more enjoyable.

Visualization tools such as Alice (Cooper, et al., 2003; Dann, et al., 2003), Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010), and Greenfoot (K¨olling, 2010). are often used to increase motivational aspects of the courses and to support the student activity by employing several different learning styles. It is also used to supplement programming courses and enable the easier transition to actual programming tasks. Often, these tools contain animation, visual hints, sounds, and interactivity. The main objective for these tools is to improve the student understanding in programming concepts by reducing the amount of physical programming required to complete the tasks. Program visualization may offer important insights into the learning and teaching of programming. The following sub-sections present three of the well-known dynamic visualization tools for learning programming (Kasurinen, Purmonen, &Nikula, 2008).

i. Alice

Alice is an approach to teaching introductory courses in computer science. It is a 3D interactive animation program with visualization environment. Novice programmers build animated 3D movies and game's characters as they learn introductory OOP (Cooper, et al., 2003; Dann, et al., 2003). It supports creating animations and building virtual worlds through a graphical user interface (Figure 1), where the student can drag and drop basic programming blocks to create programs. Alice tool supports the main OOP concepts such as arrays, inheritance, and recursion but it doesn't support the Polymorphism concept (Kelleher & Pausch, 2007).

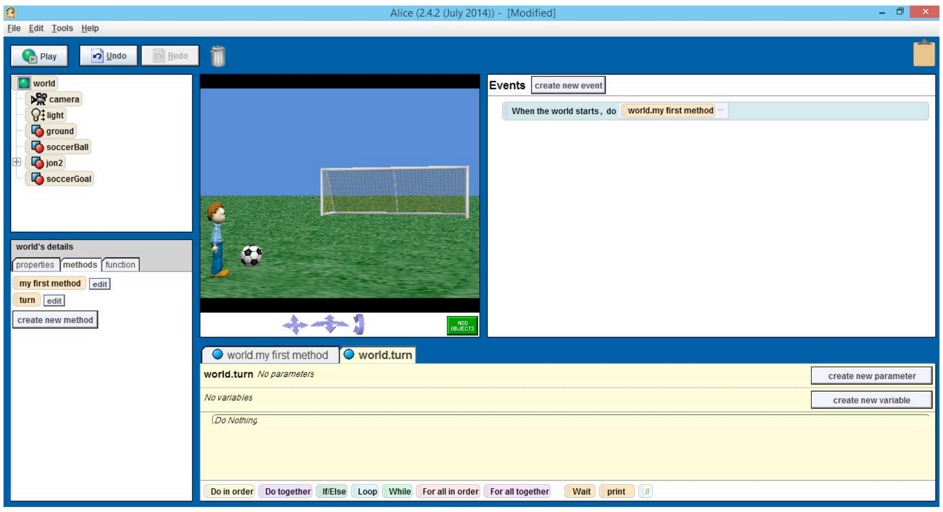


Figure 1: Alice's graphical user interface.

ii. Scratch

Scratch is a 2D visual programming environment that lets students create interactive and media-rich projects (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Students can create a wide range of projects with Scratch, including animated stories, games, book reports, music videos, science projects, tutorials, simulations, and music projects. The Scratch application is used to create projects containing media and scripts. Programming is done by snapping together colorful command blocks to control 2D graphical objects called sprites which move on a background called the stage (Figure 2) to help students make their projects personally engaging, motivating, and meaningful. Scratch makes it easy to import many kinds of media (images, sounds, music) while supporting around forty languages (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010).

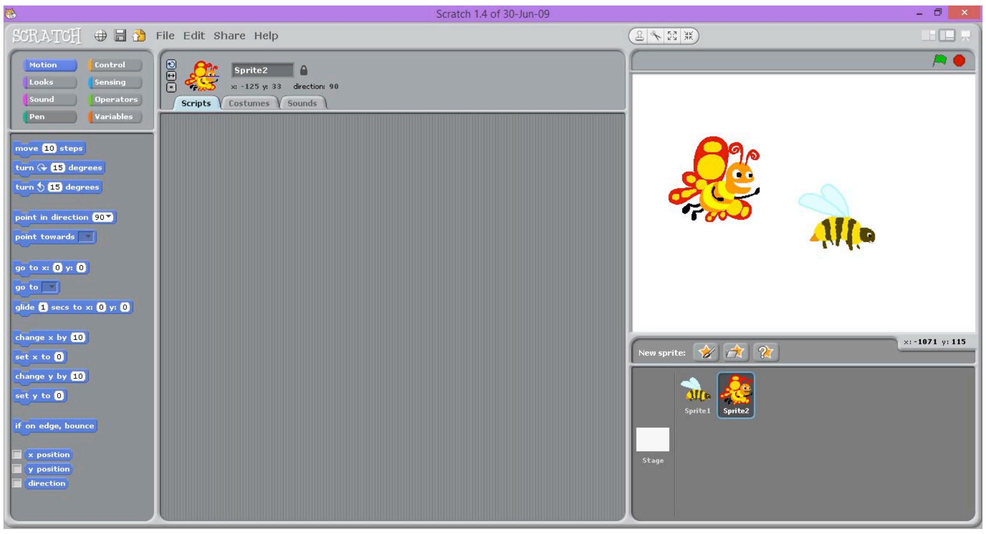


Figure 2: Scratch's graphical user interface.

iii. Greenfoot

Greenfoot is an educational development environment highly specialized for the development of interactive, graphical applications, based on the Java programming language Greenfoot (K¨olling, 2010).

Using Greenfoot, students can develop engaging and interesting programs, such as games and simulations, quickly and easily while learning fundamental programming concepts. Greenfoot (Figure 3) is designed as a 2D system. Creating 3D scenario is hard work, and Greenfoot offers little support for making this easier than it is in standard Java. However, many argue that this tool may not be well suited for novice programmers because of its advanced syntax complexity. Nevertheless, Greenfoot can still be a valuable tool for facilitating the transmission from a non-textual micro-world to a real programming language (Papadopoulos & Tegos, 2012). Both Alice and Scratch can be used with novice students while Greenfoot scales better for proficient users Greenfoot (K¨olling, 2010).

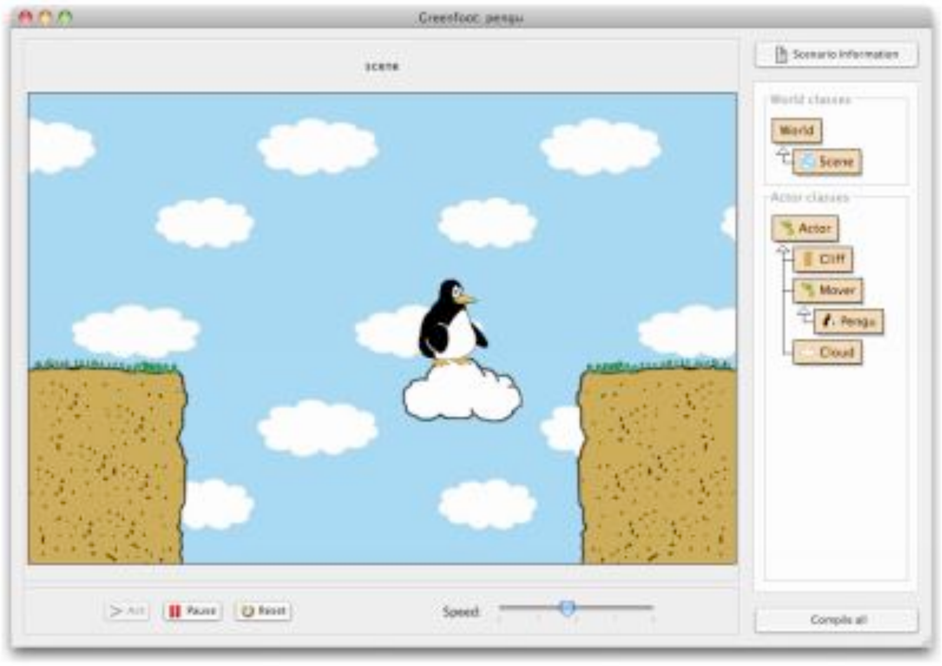


Figure 3: Greenfoot's graphical user interface (K¨olling, 2010).

III. OOP Difficulties In KAU

According to the statistical study conducted during the spring of 2015 on students of introductory OOP course (CPCS-203), with a total of 186 respondents. The results showed that students faced difficulties in understanding OOP concepts, particularly the Polymorphism. 47% of the respondents faced difficulties in understanding the concept of Polymorphism (Figure 4) while 59% of them faced difficulties in implementing it (Figure 5).

Another survey had been designed and distributed during the fall of 2015 to different academic levels of female students of Faculty of Computing and Information Technology in KAU. The data was gathered from 150 respondents. The main objective of this survey is to help us as developers to design OOPVisual tool interfaces. Although the respondents have never used any visualization tool before, they were extremely motivated and interested in using the proposed OOPVisual tool.

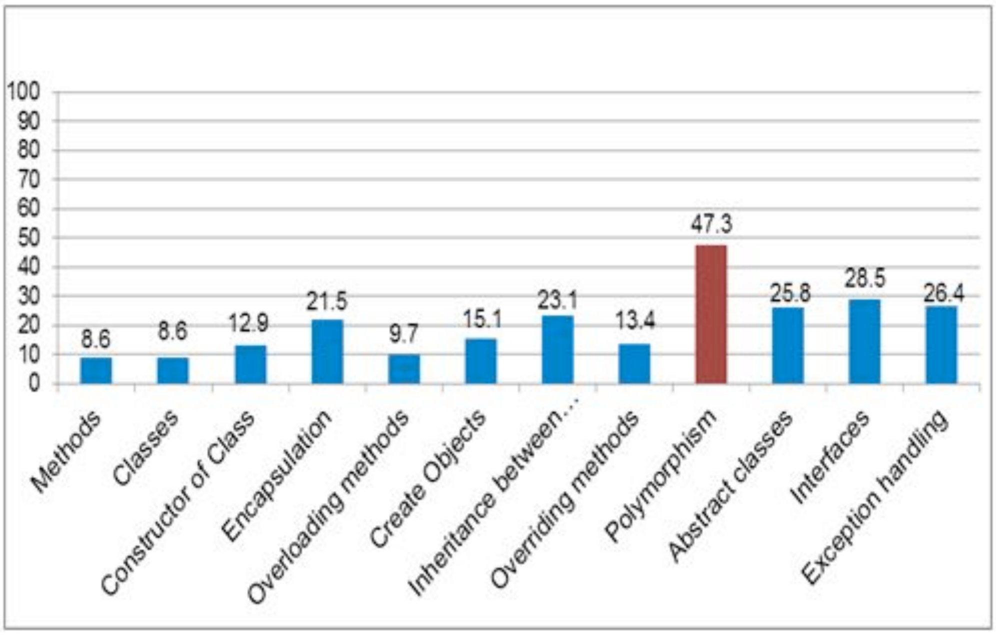


Figure 4: The number of students who have difficulties in understanding each of OOP concepts.

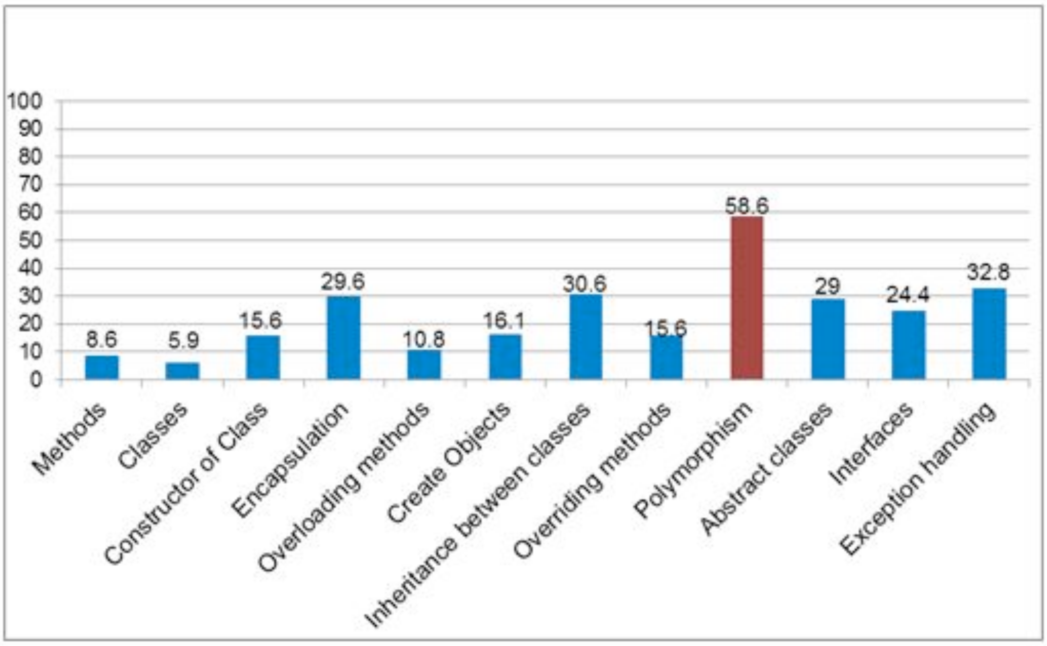


Figure 5: The number of students who have difficulties in implementing each of OOP concepts.

What is Polymorphism?

Polymorphism is one of OOP concepts that allows programmers to create versatile software design. The benefit of this concept is to enable the programmer to write a program in the general rather than in the specific (Lewis & Loftus, 2009).

A Polymorphism reference is a variable that can refer to different types of objects which can be established using inheritance or using interfaces (Deitel & Deitel, 2011). Lewis and Loftus defined the interface as “A set of abstract methods that will be implemented by particular classes” (Lewis & Loftus, 2009).

OOPVisual

The main objective of this project is to develop a 3D learning interactive visualization tool to help students in learning OOP concepts, especially in Polymorphism.

OOPVisual Design

OOPVisual design based on selecting from menus and drag and drop techniques with elimination burdens of writing texts. It consists of eight concepts' tutorials that explain the polymorphism via four levels by following the given tasks and instructions. In addition, five exercises and eight quizzes to let the student practice on polymorphism.

In addition, 'create your own scene' to create any scene without any instructions and tasks to follow. Finally, a help video about OOPVisual interface to guide the students how to use the tool. The following subsections explain the main interfaces of OOPVisual tool.

Home interface (Figure 6) contains five buttons as the following:

a) Concepts Tutorial button.
b) Exercises button.
c) Quizzes button.
d) Create your own scene button.
e) Help video.

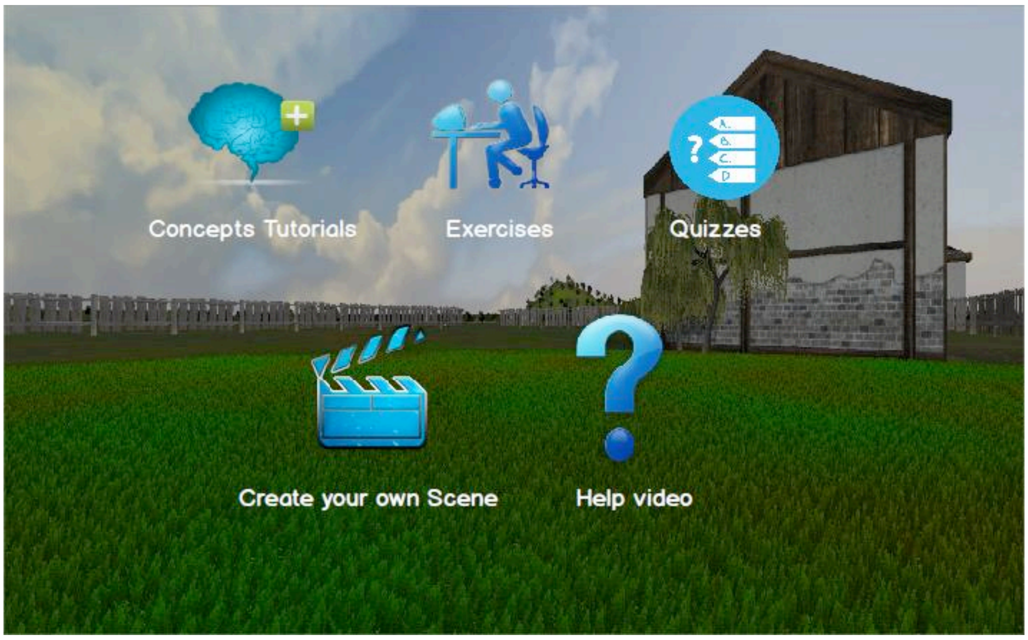
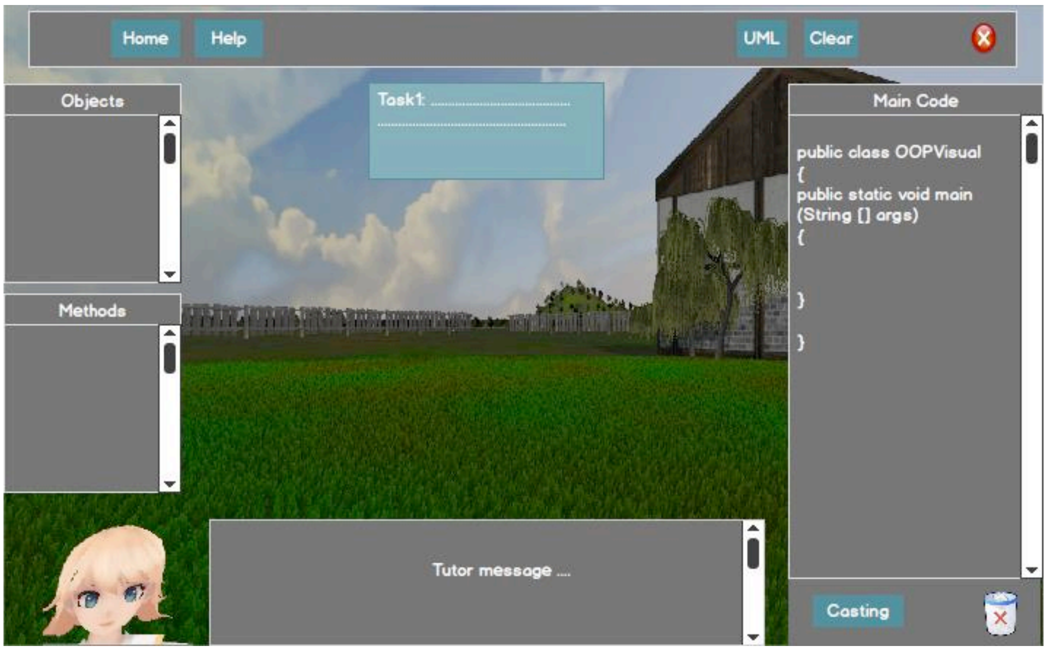


Figure 6: OOPVisual home interface

Figure 7 shows the user interface for a tutorial of concepts' tutorials. Concepts tutorials divided into four levels, each level concern with a specific part of Polymorphism concept. Each tutorial has tasks with some instructions to be followed by the student in order to complete it and move to next tutorial. There is some restriction on tutorials, the student cannot do anything else the instruction, means cannot create an array if she/he has to create an object and so on.

At the end of each tutorial, there is a confirmation message whether to repeat the tutorial, move to next one or back to home.

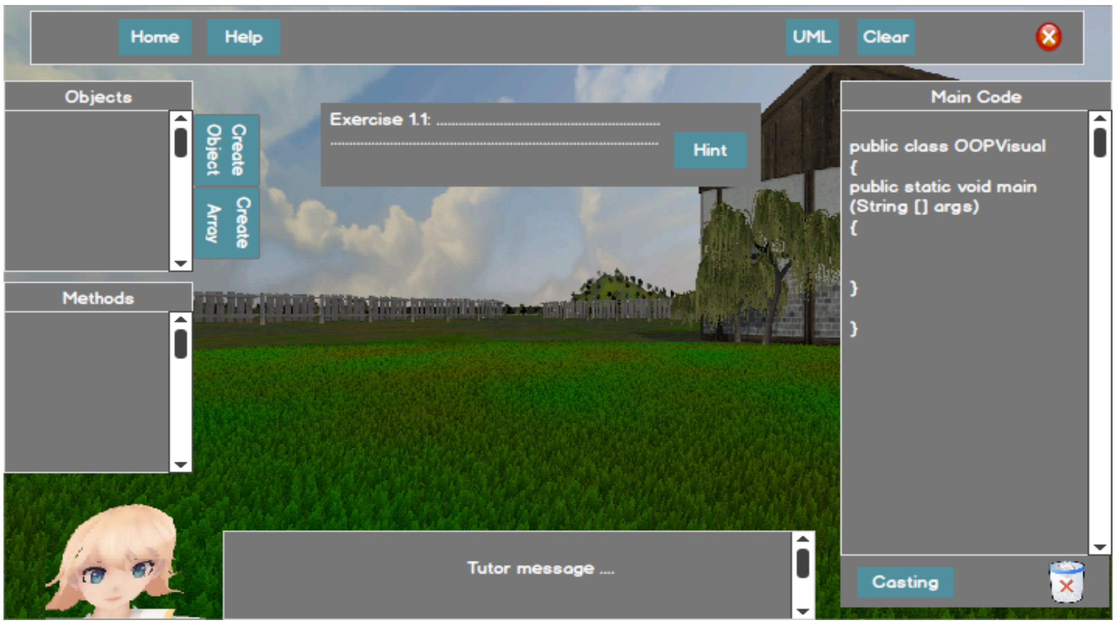


```
public class OOPVisual
{
public static void main
(String [] args)
{


}

}
```

Figure 7: OOPVisual Tutorial interface

Figure 8 shows OOPVisual exercise interface, which its main objective is to encourage students to learn by mistakes, where everything like buttons, menus or reference types will be available in the scene without any restriction. A scenario for each exercise will be given. After each action made by the student trying to accomplish the given scenario, a message with (ü) or (x) sign in addition to sounds will be displayed as a feedback.

To prevent the frustration of trying to achieve the required scenario, there is a hint button beside the scenario. Once Hint button clicked it will guide him/her for the correct action.



```
public class OOPVisual
{
public static void main
(String [] args)
{


}

}
```

Figure 8: OOPVisual exercise interface

Figure 9 shows the OOPVisual quiz interface. In order to let the student test his/her understanding of the polymorphism concept, we will provide him/her by some quizzes. In each quiz, a scenario will be displayed as a video with multiple choice answer for some questions regards the displayed video. After each selection for an answer, a message will be displayed as a feedback of the selected answer whether it is correct or not.

At the end of each quiz, a dialog box displayed to ask the student whether to back to the related tutorial, moving to next one or back to home

Quiz interface consists of nine parts explained as the following:

a. Home button: to back to home interface at any time.
b. Help button: that explains the quiz interface as a video.
c. Play button: to play and pause the video.
d. Repeat button: to repeat the video many times.
e. Exit button: to exit from the program.
f. The displayed video window.
g. Questions panel.

h. Next button: to move to next question. The student must answer the question in order to move to next one.
i. At the end of the quiz, 'Done' button to complete the quiz.

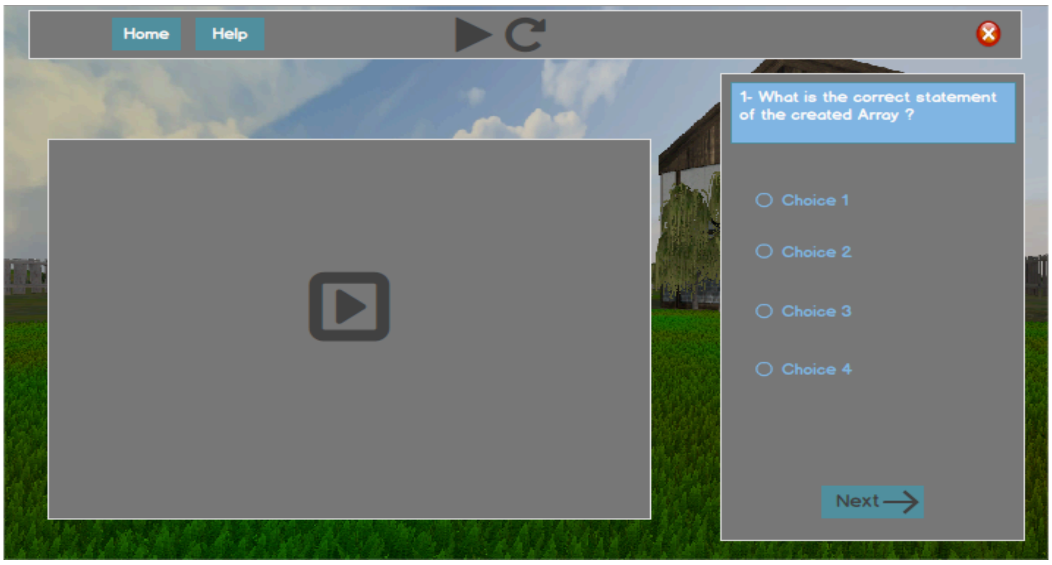
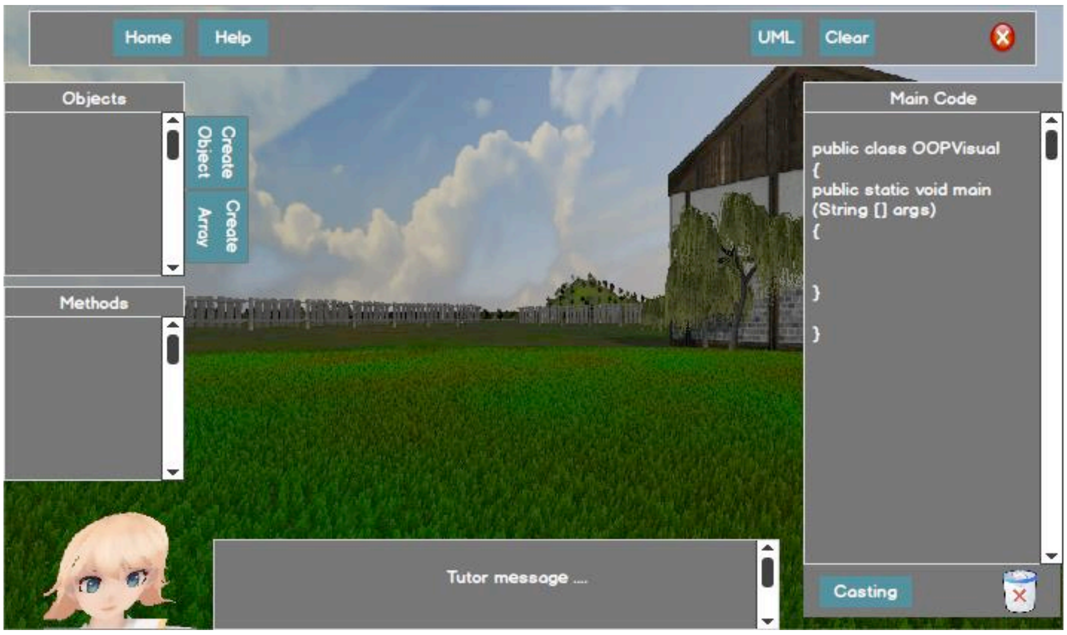


Figure 9: OOPVisual quiz interface

Figure 10 shows 'Create your own Scene' interface, to let the student create her own scene while learning polymorphism without any tasks or instructions to follow.



```
public class OOPVisual
{
public static void main
(String [] args)
{

}

}
```

Figure 10: OOPVisual Create your own Scene interface

In the following subsections, we will explain the shared parts between all interfaces.

1. **Top bar**

a. Home button: to back to home interface.
b. Help icon: to open the tutorial video that guides the student how to use OOPVisual.
c. Clear button: to clear the environment of all objects and methods.
d. UML button: in order to understand the relationships between the classes in OOPVisual, UML button used to display the UML class diagram.
e. Exit button: to exit from the program.

2. **List of Objects and Methods**

Each interface has panel displays all the objects added to the scene. In addition, another panel displays a list of all methods of the selected Objects. This list contains two buttons (Figure 11):

2.a. **Create Object button**.
2.b. **Create Array button**.

Once "Create Object" or "Create Array" is clicked, it will open the Gallery (Figure 12) that contains all the animals in the Farm grouped together based on their characteristics.

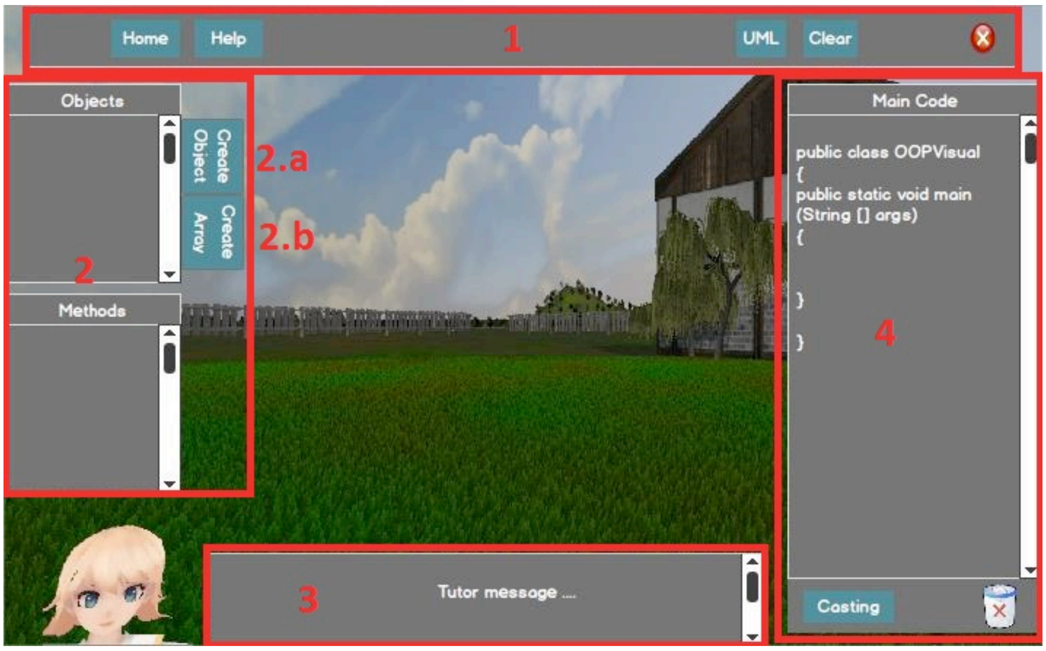
3. **Tutor**

In this part of the interface, the tutor guides the student and warning him/her for any mistakes. For example, if the student dragged and dropped Abstract class such as "Mammal", the tutor will warn her with a message via the tutor panel. As known in Object Oriented Programming, we cannot instantiate an object of Abstract Class.

4. **Main code**

Home interface has another panel for Main Code. Once the student dragged and dropped any object in the gallery and gives it a name, for example, Horse with name "myHorse", the statement will be added into the main code as Horse myHorse =new Horse ();

This panel has two buttons, one for changing the reference type of object after creation "casting", and the other one "Recycle bin" is for deleting an object.



```
public class OOPVisual
{
public static void main
(String [] args)
{

}

}
```

Figure 11: Shared part of interface

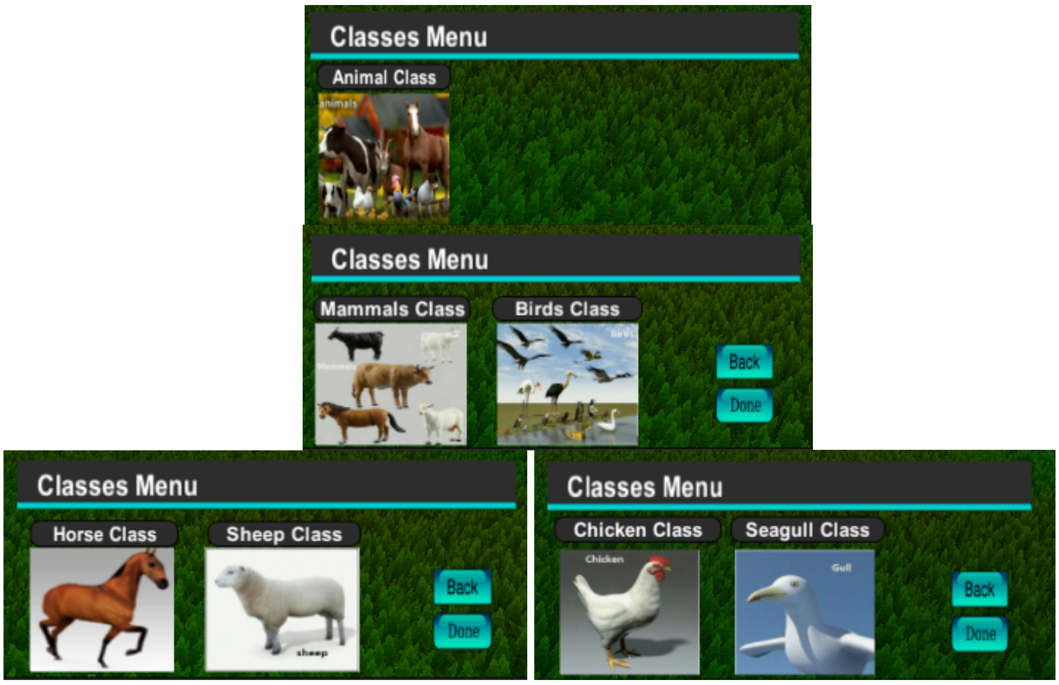


Figure 12: Gallery of Animals.

OOPVisual VS. Alice, Greenfoot, Scratch

Table 1 shows a comparison between OOPVisual, Alice, Scratch and Greenfoot to declare the main features supported by OOPVisual.

Table 1: Comparison between Different Visualization Tools with OOPVisual Tool.

Criteria			**Alice**	**Scratch**	**Greenfoot**	**OOPVisual**
Graphical User Interface	Visual Representation	Primitive	ü	ü	ü	ü
		UML Modeling	x	x	x	ü
	User Interface		ü	ü	ü	ü
Pedagogy	OOP Scope		ü	x	ü	ü
	Paradigm	Procedural	x	ü	x	x
		OOP	ü	x	ü	ü
Visualization		Dynamic	ü	ü	ü	ü
		Multimedia	ü	ü	x	ü
		3D	ü	x	x	ü
Supported Concepts		Classes	ü	x	ü	ü
		Inheritance	ü	x	ü	ü
		Polymorphism	x	x	ü	ü
Error Message			ü	x	ü	ü

Some of the terminologies used in the comparison:

- Visual Representation:
  - Primitive: tools that could be grouped into simple or composite objects, and worlds.
  - UML modeling: universally acceptable visual representations, such as UML diagrams or flow charts.
- Paradigm: programming pattern, that has two types procedural and OOP.

As shown in the table and based on the literature review, Alice and scratch do not support the polymorphism concept at all, while Scratch only works in a 2D.

Although Greenfoot supports the Polymorphism concept, it's not suitable for novice programmers according to its advanced syntax complexity. OOPVisual is a 3D dynamic, multimedia tool that explains the Polymorphism in a friendly user interface.

Conclusion

Many novice programmers lack in understanding some of OOP concepts. According to the discussed statistical study in KAU, female students faced difficulties in understanding Polymorphism. Visualization offers a technique for seeing the unseen. Recently, visualization entered in many areas, including the use of visualization tools to help illustrates programming to novice students. However, all these tools do not have a specific objective such as illustrating the Polymorphism concept in a 3D.

OOPVisual is a 3D interactive visualization tool for learning OOP concepts, particularly the Polymorphism concept. The tool acts as an interactive and animated environment. It consists of concepts tutorials, exercise, quizzes, create your own scene and help video.

Future Work

OOPVisual team future plan is to add more classes in the tool like fruits & vegetables, and people. Also, to add button "play" to play the whole scene in create your own scene interface, "Do together" button to play some methods together, add attributes for each object. In addition, implementing new exercises and quizzes to enhance the learning of polymorphism concept.

Furthermore, pre and post survey will be distributed to determine whether the proposed tool has helped the students with their OOP understanding.

References

Bashiru L., and Joseph A. A. (2015). “Learning Difficulties Of Object Oriented Programming (Oop) In University Of Ilorin - Nigeria: Students’ Perspectives”. *Dubai: Proceedings of Eighth The IIER-Science Plus International Conference*.

Biju, S. M. (2013). Difficulties in understanding object oriented programming concepts. In *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering* (pp. 319-326). Springer New York.

Cooper, S., Dann, W., & Pausch, R. (2003, February). Teaching objects-first in introductory computer science. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 191-195). ACM.

Deitel, P., & Deitel, H. (2011). *Java How to program*. Prentice Hall Press.

Goosen, L., & Pieterse, V. (2005). Roller coaster riding: highs and lows of understanding OO.109.

Kasurinen, J., Purmonen, M., & Nikula, U. (2008). A study of visualization in introductory programming. In *Proc. 20th annual Meeting of Psychology of Programming Interest Group, Lancaster, UK*.

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, *50*(7), 58-64.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005, June). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 14-18).

Lewis, J., & Loftus, W. (2009). *Java software solutions: foundations of program design*. Pearson/Addison-Wesley.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.

Nevalainen, S., & Sajaniemi, J. (2006, September). An experiment on short-term effects of animated versus static visualization of operations on program perception. In *Proceedings of the second international workshop on Computing education research* (pp. 7-16). ACM.

de Oliveira, C. A., Conte, M. F., & Riso, B. G. (1998). Aspects on Teaching/Learning with Object Oriented Programming for Entry Level Courses of Engineering.

Papadopoulos, Y., & Tegos, S. (2012, October). Using microworlds to introduce programming to novices. In *Informatics (PCI), 2012 16th Panhellenic Conference on* (pp. 180-185). IEEE.

Sheetz, S. D., Irwin, G., Tegarden, D. P., Nelson, H. J., & Monarchi, D. E. (1997). Exploring the difficulties of learning object-oriented techniques.*Journal of Management Information Systems*, *14*(2), 103-131.

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

Contacts email: aallinjawi@kau.edu.sa, wmoussa0001@stu.kau.edu.sa, ralmalki0029@stu.kau.edu.sa, malamoudi0001@stu.kau.edu.sa