Visualizing Computer Programming in a Computer-based Simulated Environment

Dr. Belsam Attallah
Assistant Professor, Division Chair, Department of Computer Information Science, Higher Colleges of Technology (HCT), UAE
Senior Fellow, Higher Education Academy (HEA), UK
Member, British Computer Society (BCS), UK

Abstract—**This paper investigated the challenges presented by computer programming (sequential/traditional, concurrent and parallel) for novice programmers and developers. The researcher involved Higher Education in Computer Science students learning programming at multiple levels, as they could well represent beginning programmers, who would struggle in successfully achieving a running program due to the complexity of this theoretical process, which has no similar real-life representation. The paper explored the difficulties faced by students in understanding this challenging, yet fundamental, subject of all Computer Science/Computing degree programmes, and focused on the advantages of visualization techniques to facilitate the learning of computer programming, with recommendations on effective computer-based simulated platforms to achieve this visualization. The paper recommended the application of virtual world technologies, such as 'Second Life', to achieve the visualization required to facilitate the understanding and learning of computer programming. The paper demonstrated extensive evidence on the advantages of these technologies to achieve program visualization, and how they facilitated enhanced learning of the programming process. The paper also addressed the benefits of collaboration and experimentation, which are ideal for learning computer programming, and how these aspects are strongly supported in virtual worlds.**

Keywords—Computer programming; programming; object-oriented programming; programming language; parallelism; multi-threading; multithreading; concurrency;; visual; visualization; visual environment; virtual worlds; second life; virtualization.

GOALS AND METHODS

The goals of this research are to assemble literature related to the difficulties faced by novice programmers and students learning computer programming at the Higher Education (HE) level, investigating the advantages of program visualization techniques to this process and recommending an effective computer-based simulated environment to achieve this visualization.

Both quantitative and qualitative research methods have been applied to achieve the outcomes of this research (questionnaires, observations and students' feedback). An intensive literature review has been carried out to document the problem formulation, and to support the research outcomes and recommendations.

PAPER OUTLINE

The research paper covers the following:

1) An introduction to the research, its aim and objectives.

2) Literature review on the problem formulation: This covers the complexity acknowledged by researchers and educators of the programming process at different levels.

a) Traditional programming.

b) Multithreading programming (concurrency and parallelism).

3) How visualization techniques are employed in a collaborative and simulated virtual environments to facilitate the learning of programming.

a) How collaboration environments are exploited in the learning of programming.

b) The employment of various visualization tools in the learning of programming.

c) The application of virtual world technologies in the learning of programming.

d) Former applications of virtual world technologies in the learning of programming.

4) The conclusion and future scope of the research.

I. INTRODUCTION

There is significant research acknowledging the level of complexity in the computer programming subject generally and at the Higher Education (HE) level. Programming skills require in-depth understanding of the complex theoretical concepts within this subject, which are recognized to be difficult to grasp by learners due to lack of real-life representation. Students who struggle in understanding and learning the abstract concepts of computer programming are likely to either withdraw from their course or choose another career path that does not involve programming [1].

This paper focuses on identifying the challenges faced by novice programmers and HE students in learning the different levels of computer programming, and provides recommendations on the techniques and platforms needed to overcome these challenges. In addition to visualization, the paper also explores the advantages of simulation, collaboration, interactivity and experimentation to support the process of learning computer programming.

II. Researchers and Educationists Acknowledging the Difficulties Faced by the Students Trying to Learn Computer Programming

A. Traditional Programming

Programming is *"a central element of the discipline of computing, an important practical skill for computing, and an essential component of the undergraduate curriculum"* [2]. A large number of researchers and educators investigated and confirmed the complexity of the programming theory process. Programming curriculum is an essential and fundamental subject in Computer Science degree programmes that all students in this field are required to learn [3]. Programming languages have extensive and complex syntaxes, which results in great learning difficulties for novice learners and a high dropout rate from qualifications including this subject [3]. In spite of the advances within other Computer Science fields; learners still believe that their computing courses are dominated by programming subjects [2].

There are various factors which may contribute to the loss of students' interest in Computer Science degree programmes, the most significant of which is the difficulties faced by students in the programming module of these courses; these difficulties result in high failure and dropout rates in preliminary programming modules at the HE level, which could reach as high as 30%-50% [4]. The difficulty in learning and teaching programming concepts is, therefore, confirmed by the high rate of failure and withdrawal in the introductory programming courses at universities [5].

Computer programming forms a common issue of concern amongst many universities due to the problems faced by their HE students in this subject in their first year of studies. In [6], authors confirmed that programming is a compulsory subject and an essential component in Computer Science curriculum, and that many novice learners often drop out from their degree courses due to either performing poorly or failing in programming subjects, which are considered the most hated and feared areas in a Computing qualification. In emphasizing the difficulty of the programming process, the reference clarified that programming techniques and skills are also hard to teach, not only because the traditional teaching methods are not very effective in the areas of scripting and problem solving, but also because such skills are best learned through experience. The difficulty in teaching this subject becomes even more challenging when trying to teach object-oriented programming to beginning learners [6].

In addition to Computer Science studies, programming is a very common subject in many fields of technology that are taught by a large number of universities in the world, although some courses only deliver the basics of it [7]. Unfortunately, learners usually face difficulties in understanding this subject even in the introductory courses, as these difficulties are not only because of the complex theory concepts in the subject, but also in various issues related to program construction, which often resulted in decreasing students' retention rate [7]. Novice learners in introductory programming courses are required to comprehend the concepts, syntax, and semantics of a programming language and then be able to apply their understanding in coding a program and solving programming problems; therefore, students of such courses consider learning to program as a difficult subject [8].

In [9], authors explained that *"Programming is one of the essential and most difficult skills to learn in the computer field and other disciplines. Programming can seem more troublesome for novices who have not learned programming concepts, usage and other basic programming skills".* Beginning learners of programming find it non-inspiring to learn this subject, and this is one of the reasons why the majority of students in this field cannot do coding by themselves [9]. A non-user-friendly graphical environment makes the learning of programming difficult and programming problems more complex; while an interactive learning environment, where support and guidance are provided for students would help in overcoming a large amount of these difficulties [9].

In [10], authors confirmed that due to the various difficulties faced by beginning learners when trying to understand and learn computer programming, a large number of them fail this subject, and consequently withdraw from their Computer Science courses. Despite the fact that researchers and educators identified the challenges faced by novice learners in this subject, they are still struggling to recommend effective measures to support practitioners in this challenging area [10]. The reference explained the outcomes of the research carried out on beginning HE students, who considered computer programming as a traditional theoretical subject (similar to history), and that it is based on reading rather than practicing. These outcomes also showed that the students felt demotivated to get involved in the learning process as they failed to understand the programming instructions or achieve encouraging results. The reference indicated that the highest complexity faced by their students in learning programming is not only the understanding of the basic concepts, but also in how to successfully apply these concepts in a more advanced construct. Although certain students understand the syntax and semantics of a programming language, they fail to employ them correctly to achieve a functional program [10].

The major cause of non-completion in Computer Science degree programmes, is the difficulties faced by students in the transition period from Further Education (FE) to HE, where many of them having either little or no confidence in their programming skills; therefore, one of the significant challenges in HE Computer Science education is to have an effective learning platform in order to achieve major enhancements in students' understanding, learning and achievement in the programming subjects [11].

Despite the fact that programming nowadays is considered a highly valuable skill, novice learners often express strong reactions to learning this subject due to the difficulties they face in understanding it [12]. Not only students face difficulties in this field, but also its lecturers, who sometimes find programming issues more challenging than students do, e.g. 'understanding programming structures' and 'designing a program to solve a certain task' [12]. Research confirms that teachers of programming continuously investigated new methods to support their learners to overcome the difficulties they face at the start of Computer Science studies [13]. Physical lectures and traditional teaching methods failed most of the time in encouraging programming learners to get involved in relevant programming activities [13]. The skills required by learners to become good programmers are far beyond the syntax and semantics of a programming language, and the complexity of this subject results in high levels of failure at the start of Computer Science studies, as learners consider that they do not even understand the most basic concepts of programming due to their abstract nature, which has no similar real life representation [13].

The 'Grand Challenges in Computing Education' Conference, hosted by the British Computer Society (BCS), 2004, indicated the teaching and learning of computer programming as a major concern within the academic community worldwide. It clarified that learners view this subject as 'dry' and 'boring' rather than 'enjoyable' and 'creative', and this has demotivated people to apply for Computer Science qualifications. The Conference added that this was also accompanied by poor achievement and retention rates in Computer Science courses, which resulted in the opinion that, even after graduation, students of Computer Science studies clearly expressing their dislike of programming and their unwillingness to study it [2].

B. Multithreading Programming (Concurrency and Parallelism)

This area is considered one of the most complex subjects in Computer Science studies. This is due to the high degree of complexity in its theory concepts related to the threading mechanism that is applied by the computer operating system in the processor and memory units, which accordingly, makes the programming of it even more difficult.

The different executions of a multithreaded program may present different sets of results based on the structure of the threads and the way they communicate with each other within the program. This non-deterministic situation makes a multithreaded program difficult to write, test and debug [14].

A number of researchers and educators confirmed the complexity of coding a multithreaded (concurrent and/or parallel) program. Multithreaded programs are not only extremely difficult to write, but they are also very difficult to analyze, debug, and verify, as these processes are much harder than those in a sequential program [15]. Research in this area emphasized the negative impacts of the non-deterministic situation in the multithreading process. Conventional wisdom has assigned the difficulties of understanding this process to non-determinism, as repeated executions of the same program given the same input value(s) could well show different behaviors [15]. The complexity of multithreaded programs lies in the large number of states that the program could possibly be in at any given time [16]. The process of debugging a multithreaded program is a challenging task that requires certain specialized knowledge and tools; this is due to the difficulty in determining the state in which the program was at the time of failure, which is a frustrating situation for developers [16].

These complex multithreading concepts are difficult to grasp by novice learners; this is because of the large number of false assumptions made by students on the scheduling process of multiple threads in a program, and that they are unable to imagine what actually happens during the program execution due to the non-deterministic nature of threads scheduling, which makes it extremely possible that successive executions of the same multithreaded program produce different outcomes [17]. It is also difficult to teach multithreading programming, as lecturers need to find a way to visualize these complex concepts to students to facilitate their understanding and increase their confidence regarding program testing and debugging [17].

In [18], authors explained the complexity of the multithreading concepts by clarifying the process of having multiple threads within one program. It indicated that each thread is performing a task that works separately from the rest of the program, which makes the concept difficult to understand by many programmers. In sequential programs, the lines of code written by programmers are executed sequentially, which is the reason behind not understanding the situation of having a number of little programs (i.e. multiple threads), each of which has its own execution sequence, running inside one large program [18].

Due to the increased requirements on maximizing computer performance and productivity, multithreading nowadays is unavoidable for programmers; however, multithreaded programs are particularly difficult to write and debug correctly, and they are much more demanding and challenging than writing and verifying a sequential program [19]. The complexity of multithreading programming is widely acknowledged; however, the necessity of it has become more urgent [20]. People are quickly overwhelmed by the concept of concurrency, as they find it much more difficult to understand and learn compared to sequential code, as partially ordered operations could well make even careful people miss possible thread overlaps [20]; while parallelism caused the computer applications to become more complex resulting in increased difficulties in their design, implementation, verification, and maintenance, which has become widely acknowledged by developers [21].

III. Employment of Visualization Techniques using Collaborative and Simulated Virtual Environments to Facilitate the Learning of Programming

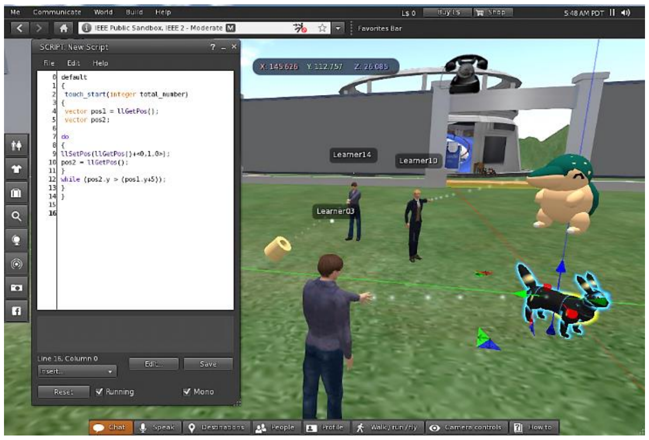
A. Utilizing Collaboration Environments

Many researchers highlighted the advantages of 'Collaborative Learning' to facilitate the understanding and learning of computer programming, and that collaborative learning is strongly achievable in virtual worlds.

In [22], authors defined 'Collaborative Learning' as an effective teaching and learning approach that is focused on adding value to students' understanding via interacting with others, where they are encouraged to share ideas and talks. Virtual worlds, e.g. Second Life, provide the students with a new opportunity to have the experience of interactive education in a computer-based simulated environment that facilitates achieving the objectives of collaboration, engagement and experimentation [23]. Collaborative simulation activities form around half of the reviewed education literature in virtual worlds (over 100 academic papers) [24], while educators have long employed role-playing and simulation as a pedagogic tool in the education sector [25].

The proceedings of the 2013 International Higher Education Teaching and Learning Association Conference: Exploring Spaces for Learning, handled the issue of engaging and retaining HE students using 'cutting-edge' technologies and innovative pedagogies, one of the major areas of which was: 'Collaboration and immersion discover best practices in a virtual world of Second Life' [26].

In the field of computer programming, collaborative environments offer significant support to students in programming activities, which is an effective approach for learning this subject [1]. In [27], author discussed the use of scientific visualization in the field of 'Big Data', indicated that the visualization process of scientific data, which is key to its analysis and understanding, is not a simple task to achieve. As human beings are 'optimized' to interact within a 3D world, a virtual world environment such as Second Life or OpenSim enables people to walk into a representation of their data, while collaborating and interacting with each other within the same virtual space [27]. This is largely applicable to visualizing the data of program variables and the program execution during runtime. Constructivist activities or problem-based learning, e.g. in Computer Science simulations, form the strongest examples of the use of virtual tools, as virtual worlds provide a strong support for collaborative work and learner interaction in a simulated environment [24] (see Fig. 1 below).



```
default
{
 touch_start(integer total_number)
{
 vector pos1 = llGetPos();
 vector pos2;

do
{
llSetPos(llGetPos()+<0,1,0>);
pos2 = llGetPos();
}
while (pos2.y > (pos1.y+5));
}
}
```

Fig. 1. Learning computer programming collaboratively in virtual worlds.

Research demonstrated that collaborative learning is considered an effective pedagogical feature for preliminary programming courses, as programming with peers is particularly appropriate for learning how to code a program [10]. The environment that promotes collaboration is able to offer important support for the activities to learn computer programming, as students need to communicate within their group, argue and give opinions, which encourages the type of reflection needed for effective learning of programming [10]. The virtual simulated environment *"enables synchronous collaboration among students because the system permits two or more avatars to edit the same object and share the same code while programming it"* [10]. Constructivists and educators involved in constructionist learning might be able to recognize the potential in this environment, as it provides them with an accessible means for the creation of rich, immersive and appealing 3D framework for situated and experiential learning, and also communication tools to support dialogue and collaborative learning [10].

B. *Application of Different Visualization Tools in the Learning of Computer Programming*

In [1], authors explained the features and applications of different visualization tools/environments and a number of other similar program visualization software, which were created by developers to facilitate the learning and understanding of computer programming. These tools were ALICE, JELIOT, BlueJ and RAPTOR, which have been used to teach introductory computer programming courses (sources from 2000-2005). Students used these environments to drag and drop chunks of code into a canvas in order to achieve a visual representation of the computer program. This resulted in isolating these blocks of code from the rest of the program, which consequently, meant that these environments lacked both a comprehensive view of program visualization and also students' engagement in a platform that does not support collaboration [1].

In [28], authors explained some other visualization tools such as jGRASP, which presented a static visualization of program execution, and ViRPlay3D/ViRPlay3D2, which presented some aspects of virtual world environments (avatars to represent learners exercising programming in a sandbox); however, this platform only facilitated the scripting process, but lacked support for collaborative learning, which is a strong feature offered by virtual world technologies.

C. *Application of Virtual World Technologies in the Learning of Computer Programming*

This paper focuses on a different visualization technique, which involves the application of virtual world technologies to visualize complex theory concepts of computer programming in order to enhance students' understanding and learning of this subject at the HE level. The research involved HE in Computer Science students in a university center in England, UK. Visualization scenarios were designed in the virtual world of 'Second Life' to support the learning of challenging programming concepts as part of the HE Computer Science Year-1 and Year-2 programming courses. These visualization scenarios were scripted by the researcher using the programming language embedded within Second Life, called 'Linden Labs Scripting Language (LSL)'. Many researchers confirmed the similarity of syntax and semantics between LSL and C++ language, which the selected HE students were studying as part of their Computer Science qualification. In [29], authors highlighted that the LSL's main syntax and operators are expressive of those in Java and C++ programming languages. It explained that Second Life implements a compiler for the LSL language that contains C++ source code. In [10], authors confirmed the above by saying that the programming of objects in Second Life is performed by the use of LSL scripting language, the keywords and structure of which are similar to those in C Language. The way the variables are declared in LSL language is the same as that in C++, and the multiple methods of creating a loop in LSL are almost identical to those in C++ [30].

In the visualization scenarios designed for this research, a number of eye-catching 3D objects were chosen to be programmed by learners within Second Life, e.g. Pokémon. This was meant to enable them to visualize the execution of challenging program instructions in order to improve their understanding of the relationship between the scripts and the actual implementation process and results. The type of the 3D objects was selected to add interest for learners and make their learning process enjoyable. These visualization scenarios enabled learners to view the immediate effects of script changes on each 3D object, i.e. visualizing the program execution. This assisted learners to understand how each program instruction works. Particular emphasis was placed on instructions related to loops and functions – for the Introduction to Programming course, and on classes and objects – for the Object-Oriented Programming course.

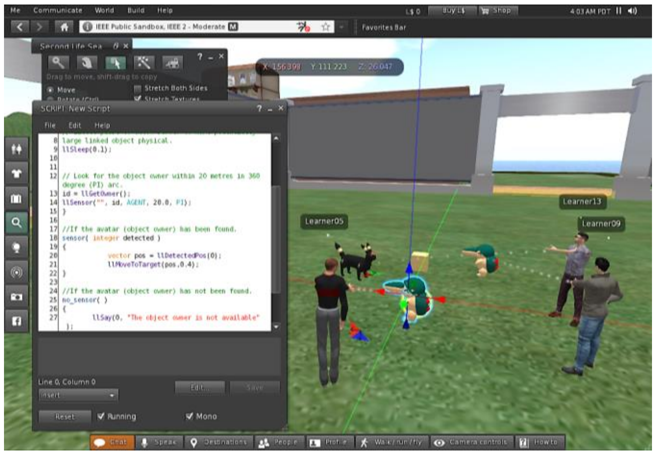
To demonstrate the benefit obtained from these visualization scenarios, below is an example of a program instruction handled by this research, which was visualized within Second Life. Learners found this instruction extremely difficult to understand and to imagine how it works and what the potential execution outcomes are. They considered visualizing this instructions' execution within Second Life very beneficial to their understanding of its function, structure and results. The advantages of visualizing programming instructions within the virtual platform were confirmed by students' answers to the following question asked by the researcher to the learners at the end of a whole session explaining the 'For Loop' in the physical classroom: "Which of these two For-Loop scripts result in moving the object six steps towards the X-axis?"

```
(1)
For (i=0; i<6; i++)
    llSetPos(llGetPos()+<i,0,0>);
(2)
For (i=0; i<6; i++)
llSetPos(llGetPos()+<1,0,0>);
```

Some students were confident of their answer, and some were not. Those who were not 100% confident were permitted by the researcher to provide a prediction based on their current/background understanding of programming. It was a surprise to both the researcher and learners that all the answers of confident learners were wrong, while around half of not fully confident learners gave the correct answer; however, they were unable to correctly justify it. This was then followed by using the virtual environment to visualize the execution of the above code. When the students worked on moving their 'Pokémons' in Second Life, they were able to view the difference in the number of steps moved by the object as a result of the execution of each script sample. Following this visualization, they were able to provide confident explanations on how each 'For Loop' of the above works. All learners confirmed that the process of explaining this instruction using the resources of a physical classroom, i.e. whiteboard, flipchart, projector, etc. did not facilitate the understanding of how this instruction works to the same degree that the visualization of it in virtual worlds did. This basic script was just an example of how confusing and complex programming instructions could be for novice learners [31].

The object-oriented programming visualization scenario, on the other hand, focused on using certain metaphors and sculptures within virtual worlds to visualize the challenging abstract concepts of 'classes' and 'objects'. This was meant to improve the understanding of what they mean, how they work, and why we need them in an object-oriented program. In this visualization scenario, learners were allowed the opportunity to compare, for example, between a portrait of a flower on the wall (a class) and an actual sculpture of the same flower planted in the ground (an object of the class) and their properties and functions, which mimicked classes and objects in an object-oriented program [31].

When learning programming collaboratively in virtual worlds, each student can have their own 3D object(s) to code; however, working collaboratively with other learners within the same virtual space enabled them the opportunity to communicate their ideas and script changes to each other. This allowed them to view the influences of these changes on the object behavior of their peers compared to that of their own objects, and consequently, modify their scripts successfully to achieve the required outcomes. Therefore, collaborative learning can strongly facilitate the learning of programming and develop the other set of skills necessary for this subject (see Fig. 2 below). The virtual world of Second Life "*enables synchronous collaboration among students because the system permits two or more avatars to edit the same object and share the same code while programming it*" [10].



```
large linked object physical.
llSleep(0.1);


// Look for the object owner within 20 metres in 360
degree (PI) arc.
id = llGetOwner();
llSensor("", id, AGENT, 20.0, PI);
}

//If the avatar (object owner) has been found.
sensor( integer detected )
{
            vector pos = llDetectedPos(0);
            llMoveToTarget(pos,0.4);
}

//If the avatar (object owner) has not been found.
no_sensor( )
{
        llSay(0, "The object owner is not available"
 );
```

Fig. 2. Sharing the programming of 3D objects in virtual worlds between multiple learners.

A detailed questionnaire was distributed to students to capture their feedback on the application of virtual worlds to visualize the programming process [31]. The outcomes were as follows (see Fig. 3 below):

- The thoughts of slightly over 50% of learners, who initially considered this subject as difficult to understand and learn, were reversed following exercising programming in virtual worlds.
- Twenty-one percent more learners confirmed that effective understanding and learning of the complex theory concepts of this programming subject were achieved following the visualization activities in virtual worlds.
- Ninety-four percent of learners confirmed that affective quality was improved in the virtual platform. This figure was almost double the percentage obtained for affective quality in the physical world.
- Thirty-seven percent more learners confirmed that visualizing and learning this level of programming within virtual worlds is more engaging.

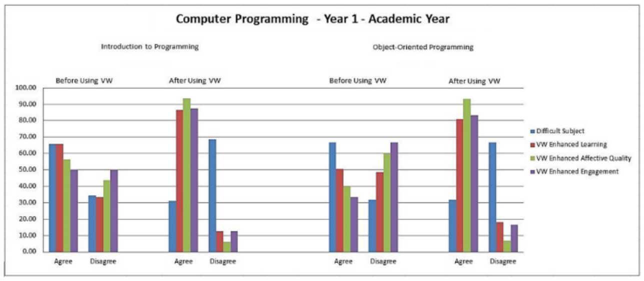


Fig. 3. Computer programming questionnaire, Year-1.

With regards to the introduction to object-oriented programming in Year-1, and as can be seen in Fig. 3 above, the questionnaire on the visualization scenario showed that:

- In agreement with the outcomes of the Introduction to Programming visualization scenario, the thoughts of slightly over 50% of learners, who initially considered this subject as difficult to understand and learn, were reversed following exercising programming in virtual worlds.
- Thirty-one percent more learners confirmed that effective understanding and learning of the complex theory concepts of this programming subject were achieved following the visualization activities in virtual worlds.
- Ninety-three percent of learners confirmed that affective quality was improved in the virtual platform. This figure was a lot higher than double the percentage obtained for affective quality in the physical world.
- Eighty-three percent of learners confirmed that visualizing and learning this level of programming within virtual worlds is more engaging. This figure was, again, a lot higher than double the percentage obtained for the physical world.

Moving to the Object-Oriented Programming, which is a more complex subject compared to normal programming (as highlighted earlier), this subject is delivered in Year-2 and the students were introduced to the subject towards the end of their Year-1 introductory programming studies. It was reasonable to expect that students at this higher level (Year-2) would be more confident in learning programming compared to Year-1 students, as these Year-2 students have already studied the introduction to programming in their Year-1. However, the results of the below questionnaire revealed otherwise.

The questionnaire on the Object-Oriented Programming visualization scenario (which was designed by a different lecturer), revealed that as high as 77% of Year-2 students still consider the object oriented programming as a difficult subject. This confirmed that computer programming is an area of concern to HE students in Computer Science courses at all levels. The complexity faced by students in learning programming is not only the understanding of the basic concepts, but also in the process of applying these concepts correctly to achieve more advanced constructs. Although some students understand the syntax and semantics of a programming language, they fail to use it correctly to create a program [10].

The questionnaire showed that more than three quarters of students confirmed that Object-Oriented Programming is a difficult subject, and a slightly higher percentage of students agreed that the virtual world exercise improved their understanding and learning of the complex theory concepts of the subject, while 90% of them agreed to enhanced affective quality and 64% found this learning process more engaging [31] (see Fig. 4 below).

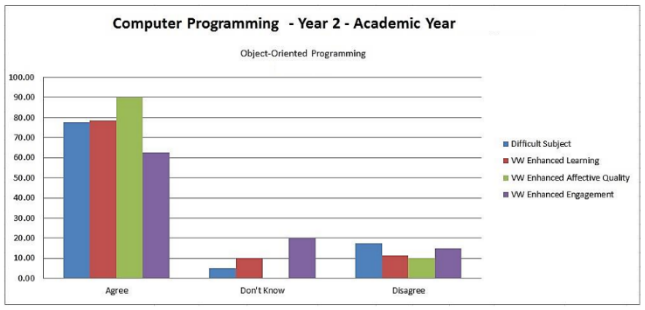


Fig. 4. Computer programming questionnaire.

On the other hand, the visualization scenario designed in virtual worlds to visualize the complex theory concepts of multithreading techniques (Concurrency and Parallelism) was applied on the BSc in Computer Science students, who found this module to be the most challenging amongst all the other modules. Before using virtual worlds to visualize the multithreading techniques, the researcher was used to drawing a number of sketches on the whiteboard for students to represent the computer Random Access Memory (RAM) and processor, with a number of arrows representing the data flow between these two components (for individual examples). More drawings and arrows were then added to show how the operating system controls the swapping and priority of tasks (threads) inside a computer and the time slots allocated to them within the processor. However, these sketches on the whiteboard could get very crowded and confusing for students, especially when more components are added (drawn on the board), e.g. the input/output devices and virtual memory. In addition, there was no clear representation on the board of the sequence of actions and their individual consequences.

In order to visualize the multithreading techniques, the researcher carried out a thorough investigation for a comparable real-life example that requires a similar queuing process to receive the service (i.e. the queuing of threads in RAM by the operating system), and how the structure of the queue is affected by a higher priority arrival. The intension was to build the scenario in virtual worlds on the selected real-life example, in order to achieve a clearer and intuitive illustration for the students to enable them to compare the situation to that inside a computer system.

The outcome of the investigation was to choose a buffet restaurant example with a single restaurant keeper/waiter, where customers need to queue to get food, ice cream and drinks. The comparison between this real-life example and the multithreading techniques was as follows: The customers' queue represents the queue of tasks/threads within the computer RAM waiting to be served by the processor, while the food buffet, ice cream counter and the drink machine represent different resources/cores within the computer processor. The single restaurant keeper, who coordinates the providing of services, represents the operating system, while the restaurant tables and chairs represents the computer virtual memory having stand-by tasks (seated customers in the restaurant example) waiting for a space to join the queue in order to get served. Finally, the counter on the side having plates and cups, where the customer needs to go out of the queue to get a plate, represents the input/output devices in a computer system, where a task in RAM needing an input value cannot be served by the processor until it gets it. Fig. 5 and 6 show the virtual restaurant designed in Second Life to visualize the multithreading techniques.

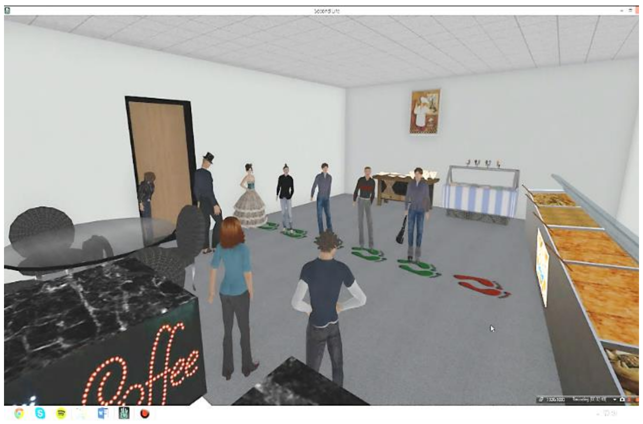


Fig. 5. Multithreading Techniques visualization scenario (queauing technique in RAM with tasks in the virtual memory).

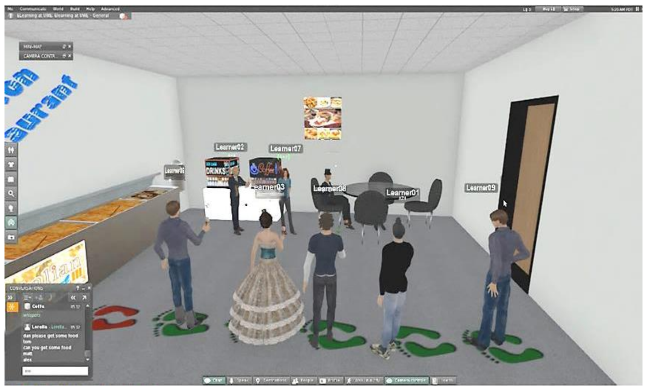


Fig. 6. Multithreading Techniques visualization scenario (swapping of tasks between the RAM and virtual memory).

In the virtual restaurant scenario, some students were required to act the role of customers queuing to get food, desserts or a dink (representing computer tasks queuing in RAM), while other students were required to sit down around the tables when the queue was full (representing tasks stored in the computer virtual memory) waiting for any of the queuing customers to finish, then the restaurant keeper (another student representing the role of the computer operating system) would ask them, one by one, to join the queue. Throughout this process another student is asked to act the role of a VIP customer who arrived to a busy restaurant having a full queue and a number of other seated customers waiting to be served (representing a high priority task joining a full RAM) [31].

Within this visualization scenario, a number of different multithreading situations were explained to the students, using the above restaurant metaphors, with their impacts and outcomes, e.g. when a higher priority task is placed by the operating system at the start of the queue in RAM changing the order of execution for all the remaining tasks, having a full queue with or without tasks in the virtual memory, and having a single core (Concurrency) or multiple cores/processors (Parallelism). Being part of this number of different situations and their visual impacts facilitated students' understanding of the complex abstract concepts of the multithreading process and the various factors affecting the execution of tasks in a computer system. In addition, the situation of role-playing the different computer components contributed greatly to this enhanced learning [31].

The observations by the researcher confirmed that the students found this visualization scenario extremely useful in understanding the different aspects of the multithreading process and the need for it. The researcher directed well-selected questions to the students (during and after the virtual exercise) to test their level of understanding and learning, and also to record their evaluation of their experience in virtual worlds.

The outcomes of the questionnaire distributed to students to capture and record their feedback on this visualization scenario showed the following outcomes (see Fig. 7 below):

- The thoughts of slightly over 50% of learners, who initially considered this subject as difficult to understand and learn, were reversed following exercising programming in virtual worlds.
- As high as 96% of learners confirmed that effective understanding and learning of the complex theory concepts of this programming subject were achieved following the visualization activities in virtual worlds. This figure was almost double the percentage obtained for the physical world.
- Hundred percent of learners confirmed that affective quality was improved in the virtual platform compared to 62% for the physical world.
- Hundred percent of learners confirmed that visualizing and learning this level of programming within virtual worlds is more engaging compared to 57% for the physical world.

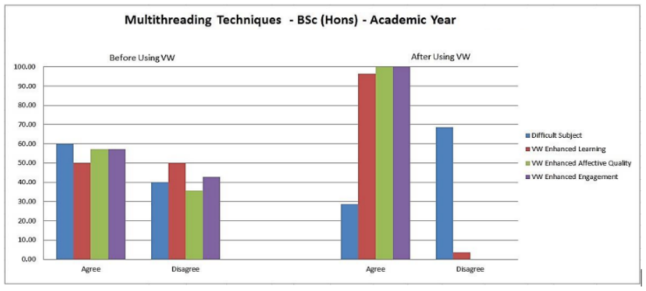


Fig. 7. Statistics of Multithreading Techniques, B.Sc. Students.

The process of visualizing the complex theory concepts of programming was more effective in virtual worlds as it engaged students in this immersive environment much more than the situation in the physical world where the viewer watches the program code passively. Interactivity and experiential learning were strongly achieved here. The virtual world environment inspired expressive and dynamic discussions on programming concepts, as students built their own visualization of the program, and followed the presentation of it more engagingly.

The researcher's observations throughout the different stages of this research demonstrated that students felt more relaxed in repeating their programming activities in virtual worlds when making mistakes or when not fully achieving their targets. This was due to the flexibility offered by the virtual world environment, and the fact that there were no physical consequences involved, e.g. being embarrassed in front of other students and/or the lecturer. This resulted in more engagement and involvement in the learning process, and consequently enhanced students' acquired skills and achievement in this field. As students were represented in avatars within the virtual world platform, they had less hesitation in asking basic questions or requesting more information. The facility of having a private channel in Second Life was very beneficial to students in carrying out private chatting (via text) with the lecturer. This inspired more interaction especially for the shy students, and increased their self-confidence in discussing their concerns without feeling embarrassed for lagging behind others.

D. *Previous Applications of Virtual World Technologies in the Learning of Computer Programming*

As highlighted in the previous sections of this paper, research activities in the learning of programming area were explained by [10]. However, their paper indicated that although the main target of the research work was to investigate the possibility of using the Second Life virtual world as a platform for the teaching and learning of an imperative computer programming language, the research focused primarily on investigating the potential problems that could be faced by both teachers and students in this environment, and whether such problems could be solved and how.

In [1], authors carried out a study in Deakin University and Monash University, Australia, regarding the learning of computer programming in virtual worlds. It investigated the affordances of Second Life for 'experiential problem-based learning pedagogies', and the potentials and limitations of this platform for learning the programming subject. The study generated very positive answers in terms of the advantages of Second Life virtual world for learning computer programming.

In [32], authors explained an application of Second Life in the computing courses of the School of Computing, University of Portsmouth, UK. It described that Second Life was used in two areas: 1) Human Computer Interaction (HCI) Unit, and 2) Computer Engineering Projects Unit, both of which involve a great deal of programming requirements.

In [33], authors also studied the application of Second Life to engage and motivate the HE Computing students of the Computer Information Systems Department at Borough of Manhattan Community College, New York, USA. It explained that a teaching and learning platform was designed in Second Life to assist the students in overcoming the difficulties in their study. It clarified that the designed platform included a lecture area, group study rooms and interactive teaching and learning activities, which aimed at better engagement of students and the improvement of the retention data within the Computer Science programme.

In [13], authors introduced Second Life in the learning of computer programming in two higher education academic institutions in Portugal, where they used the 3D virtual world environment to visualize and contextualize some programming aspects. The use of visualization helped the programming students to better understand these aspects, because visual representations are easier to retain and handle, and that having an instant visualization of instruction results enabled students to directly judge whether their idea was right or wrong [13]. Second Life users were able to create avatars and 3D objects, and to program their behavior using the Linden Scripting Language (LSL); the benefit of this is the students' ability to execute the programming code concurrently and that several students are able to simultaneously work over the same code and/or object, which provided the advantage of immediate presentation of program execution [13].

IV. Conclusion and Future Scope

Learning computer programming forms a cause of concern to a large number of novice programmers and students studying this field at the HE level. Research revealed that these concerns are the main reason behind HE students' withdrawal from their computing courses, achieving poorly or failing the modules that include programming concepts.

Research also showed that there are a number of software tools to visualize program structure for learners; however, the majority of them promoted static visualization, which did not generate the degree of support needed for the programming complex theoretical process.

This research demonstrated that there are strong indications of benefits of visualizing the program structure in virtual worlds, as this platform offers great advantages such as collaboration, simulation, interactivity and experiential learning, which are ideal for learning computer programming. This did not only cover enhancements to students' understanding of the programming complicated process, but also increased their engagement in the sessions, enhanced affective quality and improved their achievement.

The future scope could be utilizing virtual reality technology to facilitate the learning of programming with a comprehensive comparison between the advantages and limitations of both computer-based simulated environments. Aspects such as lecturer/students' satisfaction, ease of use and the technical issues involved could form the main points of the proposed comparison.

References

[1] Sajjanhar and J. Faulkner, "Exploring Second Life as a Learning Environment for Computer Programming," Deakin University and Monash University, Australia, Scientific Research, Creative Education, vol. 5, no. 1, pp. 53-62, January 2014.

[2] McGettrick, R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove and K. Mander, "Grand Challenges in Computing Education," British Computer Society (BCS). The Computer Journal, vol. 48, no. 1, pp. 42-48, January 2005.

[3] L. Morgado, B. Fonseca, M. Esteves, P. Martins, "Improving teaching and learning of computer programming through the use of the Second Life virtual world," The Polytechnic Institute of Leiria and the University of Trás-os-Montes e Alto Douro, Portugal, British Journal of Educational Technology, vol. 42, no. 4, pp. 624-637, July 2011.

[4] S. Dasuki and A. Quaye, "Undergraduate Students' Failure in Programming Courses In Institutions Of Higher Education In Developing Countries: A Nigerian Perspective," American University of Nigeria. EJISDC, vol. 76, no. 8, pp. 1-18, 2016.

[5] J. Kaasbøll, O. Berge, R.E. Borge, A. Fjuk, C. Holmboe and T. Samuelsen, "Learning Object-Oriented Programming," Norway: University of Oslo and InterMedia. 16th Workshop of the Psychology of Programming Interest Group, Carlow, Ireland, pp. 86-96, April 2004.

[6] Miliszewska and G. Tan, "Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming," Victoria University, Melbourne, Australia. Issues in Informing Science and Information Technology, vol. 4, pp. 277-289, 2007.

[7] Lahtinen, K. AlaMutka and H.M. Järvinen, "A Study of the Difficulties of Novice Programmers," Tampere, Finland: Institute of Software Systems, Tampere University of Technology. ACM Digital Library, vol. 37, no. 3, pp. 14-18, June 2005.

[8] R. Matthewsa, H.S. Hinb and K.A. Chooc, "Practical use of review question and content object as advanced organizer for computer programming lessons," The University of Nottingham, Malaysia Campus & Multimedia University, Malaysia. Elsevier, Science Direct, vol. 172, pp. 215-222, January 2015.

[9] M. Sasaki, S.M. Taheri and H.T Ngetha, "Evaluating the Effectiveness of Problem Solving Techniques and Tools in Programming," Gifu University, Japan. IEEE Xplore, Science and Information Conference, London, UK, July 2015.

[10] Fonseca, M. Esteves, L. Morgado and P. Martins, "Using Second Life for Problem Based Learning in Computer Science Programming," Pedagogy, Education and Innovation in 3-D Virtual Worlds. The Polytechnic Institute of Leiria and the University of Trás-os-Montes e Alto Douro, Portugal. Journal of Virtual World Research, vol. 2, no. 1, ivwresearch.org, April 2009.

[11] M. Huggard, "Programming Trauma: Can It Be Avoided?" Proceedings of the British Computer Society (BCS), Grand Challenges in Computing: Education. Newcastle, England, pp. 50-51, March 2004.

[12] Costa and M. Piteira, "Learning Computer Programming: Study of difficulties in learning programming," IPS–ESTSetúbal & IPS–ESTSetúbal, Lisboa, Portugal. ACM Digital Library. ISDOC '13 Proceedings of the 2013 International Conference on Information Systems and Design of Communication, pp. 75-80, July 2013.

[13] M. Esteves, B. Fonseca, L. Morgado and P. Martins, "Contextualization of Programming Learning: A Virtual Environment Study," Portugal: Polytechnic Institute of Leiria & University of Trás-os-Montes e Alto Douro. 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY, October 2008.

[14] H. Cui, J. Wu, J. Gallagher, H. Guo and J. Yang, "Efficient Deterministic Multithreading Through Schedule Relaxation," Cascais, Portugal: Department of Computer Science, Columbia University. SOSP'11 - Proceedings of the 23rd ACM Symposium on Operating Systems Principles, pp. 337-351, October 2011.

[15] J. Yang, H. Cui, J. Wu, Y. Tang and G. Hu, "Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading," Columbia University, Communications of the ACM, vol. 57, no. 3, pp. 58-69, February 2014.

[16] J. Roberts and S. Akhter, "An Introduction to Multi-threaded Debugging Techniques," Go Parallel, Intel Corporation, USA, 2011.

[17] Malnati, C.M. Cuva and C. Barberis, "JThreadSpy: Teaching Multithreading Programming by Analyzing Execution Traces," ACM Digital Library, PADTAD '07 Proceedings of the 2007 ACM workshop on Parallel and distributed systems: testing and debugging, pp. 3-13, July 2007.

[18] Rick, T. Mohiuddin and M. Nawrocki, "LabVIEW Advanced Programming Techniques," Chapter 9: Multithreading in LabVIEW. Boca Raton: CRC Press. Taylor & Francis Group, LLC, 2007.

[19] M. Huisman and C. Hurlin, "Permission Specifications for Common Multithreaded Programming Patterns," France: INRIA Sophia Antipolis, December 2007.

[20] E.A. Lee, "The Problem with Threads," Electrical Engineering and Computer Sciences, University of California, Berkeley, USA. Technical Report No. UCB/EECS-2006-1, IEEE Computer, vol. 39, no. 5, pp. 33-42, May 2006.

[21] M. Duranton, D. Black-Schaffer, S. Yehia and K. De Bosschere, "Computer Systems: Research Challenges Ahead. The HiPEAC Vision 2011/2012," High Performance and Embedded Architecture and Compilation, Seventh Framework Programme. HiPEAC Compilation Architecture, October 2011.

[22] A.E. Woolfolk, M. Hughes and V. Walkup, "Psychology in Education," Harlow: Pearson Longman, 2008.

[23] Y. Huang, S. Backman and K. Backman, "Student attitude toward virtual learning in Second Life: A flow theory approach," Taylor & Francis [Online], Journal of Teaching in Travel & Tourism. Clemson University, Clemson, South Carolina, USA, vol. 10, no. 4, pp. 312-334, November 2010.

[24] Duncan, A. Miller and S. Jiang, "A taxonomy of virtual worlds usage in education," British Journal of Educational Technology (BJET), vol. 43, no. 6, pp. 949-964, January 2012.

[25] S. Fitzsimons, “An Exploration of Teaching and Learning in A Virtual World in The Context of Higher Education,” PhD thesis, School of Education Studies, Dublin City University, Dublin, July 2012.

[26] P. Blessinger and C. Wankel, “Proceedings of the 2013 International Higher Education Teaching and Learning Association Conference: Exploring Spaces for Learning,” St. John's University, New York, USA. Conference organized by: UCF, HETL (Higher education Teaching & Learning), Faculty Center for Teaching & Learning, 2013.

[27] Cioc, “Immersing Yourself in Your Data: Using Virtual World Engines to Solve “Big” Data,” Astrobetter [Online], March 2013. [Accessed 12 July 2016].

[28] J. Sorva, V. Karavirta and L. Malmi, “A Review of Generic Program Visualization Systems for Introductory Programming Education,” Aalto University, Finland. ACM Digital Library. Vol. 13, no. 4, Article No. 15, November 2013.

[29] W. Moldenhauer, J.C. Browne and C. Lin, “Bringing Verification to a Virtual World,” CiteSeerX. Honors Thesis, Department of Computer Sciences, University of Texas, Austin, USA, May 2007.

[30] J. Gomez, “Chapter 4: Logic,” LSL Wiki [Online], 2012. [Accessed 02 May 2016].

[31] Attallah, “The affordances of virtual world technologies to empower the visualisation of complex theory concepts in computer science: Enhancing success and experience in higher education,” PhD, University of the West of England, June 2015.

[32] J. Crellin, E. Duke-Williams, J. Chandler and T. Collinson, “Virtual Worlds in Computing Education,” School of Computing, University of Portsmouth, UK. Taylor & Francis [Online], Computer Science Education Journal, Web-based technologies for social learning in computer science education, vol. 19, no. 4, pp. 315-334, December 2009. [Accessed 07 June 2016].

[33] Y. Chen, J. Doong, and W. Ching-Song, “A 3D Virtual World Teaching and Learning Platform for Computer Science Courses in Second Life,” Computer Information Systems Department, Borough of Manhattan Community College, CUNY, New York City, New York, U.S.A & Department of Information Management China University of Technology, Taipei, Taiwan. IEEE Xplore [Online], December 2009. [Accessed 09 May 2016].