Creative Education
2014. Vol.5, No.1, 53-62
Published Online January 2014 in SciRes (http://www.scirp.org/journal/ce)
http://dx.doi.org/10.4236/ce.2014.51010

Exploring Second Life as a Learning Environment for Computer Programming

Atul Sajjanhar[1], Julie Faulkner[2]
[1]Deakin University, Burwood, Australia
[2]Monash University, Clayton, Australia
Email: atuls@deakin.edu.au, julie.faulkner@monash.edu

Received August 27th, 2013; revised September 27th, 2013; accepted October 4th, 2013

Computer programming can be challenging for beginners because of the need to understand abstract programming concepts. In this paper, we study the use of the Second Life (SL) virtual world for learning computer programming concepts. We conduct an empirical study for learning computer programming in SL by addressing affordances of SL for experiential problem-based learning pedagogies. We present preliminary findings, the promises and the limitations of Second Life as an environment for learning computer programming.

Keywords: Problem Based Learning; Computer Programming; Experiential Learning; Second Life

Introduction

Failure to grasp computer programming skills can lead to demotivation of computer science students (Jenkins, 2001). Some students who struggle with programming may drop out of the course while others might choose a career path which does not involve programming (Miliszewska & Tan, 2007; Stamouli et al., 2004).

Learning computer programming requires a hierarchy of skills (Sloane & Linn, 1988). Skills required for computer programming are broadly divided into high level skills and low level skills. High level computer programming skills are: analytical skills for problem analysis and developing a conceptual solution (Esteves et al., 2009). Low level computer programming skills are used to generate syntactically correct computer programs, i.e. articulate conceptual solutions in a programming language. The focus of this paper is on high level computer programming skills. High level skills require a comprehension of abstract programming concepts which tend to be difficult to grasp, because they lack real life analogies (Dunican, 2002).

Environments, such as ALICE (Dann, Cooper, & Pausch, 2000), JELIOT (Ben-Bassat Levy et al., 2003), BlueJ (Kölling, Quig, Patterson, & Rosenberg, 2003) and RAPTOR (Carlisle, Wilson, Humphries, & Hadfield, 2005) have been used to teach introductory computer programming. In these environments, blocks of code are dragged and dropped into a canvas to create visual representations of a computer program. Isolating the programmer from intricacies of programming syntax means that these environments do not have a steep learning curve which is conducive for student engagement. These environments are suitable for teaching high level computer programming skills. However, the drawback of these environments is that they do not inherently support collaboration.

Guzdial et al. (1996) and Menchaca et al. (2005) suggest that collaboration is an effective approach for computer programming. According to Casamayor et al. (2009), collaboration is significant, because it stimulates learning, promotes feelings of belonging to a team, encourages creativity, eases communication and increases achieved personal satisfaction. Collaborative environments can offer important support to students in their activities for computer programming. According to Newman (1989), collaboration in problem solving not only provides an appropriate activity but also promotes reflection, a mechanism that enhances the learning process. Reflection encourages the act of articulating and ordering thoughts, organizing them in a coherent form to provide fresh insights into practices and motivations. Britzman (2003) argues the importance of "second thoughts", running through and over ideas in multiple and repetitive ways in order to draw more from the ideas and consider how something could be done differently.

There are several vendors which provide Multi-User Virtual Environment (MUVE) or virtual world environment for teaching/learning. Virtual worlds provide a collaborative learning environment which affords a contextual embodied experience and has the potential to offer student engagement aligned with real-world experiences (Sajjanhar, 2012). Second Life (http://secondlife.com/) is a 3D virtual world developed by Linden Lab which was founded in 1999. Another vendor of 3D virtual world is Active Worlds (www.activeworlds.com). 3D virtual worlds have already been used as pedagogical media (Dickey, 2003). In this paper, we focus on Second Life (SL), because it is the most mature of virtual world platforms, i.e. it is robust and feature-rich which is reflected by its high usage figures compared with other competing platforms (Warburton, 2009; Dalgarno, 2010). SL is used in pedagogy (Journal of

Virtual Worlds Research, 2009): it is used in medical and health education (Boulos, Hetherington, & Wheeler, 2007); it is used for teaching languages, including Arabic (Kern, 2009) and Chinese (Henderson et al., 2010); it is also used for researchers to collaborate (Novak, 2010).

The aim of this paper is to investigate the potential of SL as an enabling environment to acquire the disparate skills required for computer programming. The research question addressed is: To what extent does SL facilitate understanding of computer programming concepts? We conduct an empirical study in which academics and students provide relevant feedback about SL; data from the study are used to draw inferences about SL in the context of a programming learning environment. The rest of the paper is organized as follows: Section 2 describes the proposed approach including pedagogies of interest; evaluation is documented in Section 3. Conclusions are given in Section 4.

Proposed Approach

SL has a proprietary scripting language, namely, Linden Scripting Language (LSL); it is used to control the behavior and interactivity of virtual entities. LSL can be used to manifest concepts in the virtual world. Hence, LSL can be used to acquire higher level computer programming skills such as problem analysis and conceptualization. SL provides a means for educators to use various pedagogies suitable to teach computer programming, namely, problem-based learning and experiential learning. In this section, we address these pedagogies in the context of SL as a learning environment for computer programming.

Experiential Learning

Experiential learning is proposed by Kolb (1984). According to Wrzesien et al. (2010), the aim of experiential learning theory is to engage learners in direct experience. Experiential learning as defined by Kolb (1984) consists of a cycle of the following stages: concrete experience, reflect on experience, abstract conceptualization and an active experience.

Girvan et al. (2013) have mapped stages of experiential learning to the stages in creating programmable artefacts in SL. LSL is the native programming tool in SL but has a steep learning curve for a novice. Barriers to engagement are lowered by providing a low-floor tool i.e. a tool which does not require a steep learning curve. Scratch for Second Life (S4SL) was designed by Rosenbaum (2008) as a low-floor and high-ceiling (powerfully expressive) programming tool for SL. It is a visual environment which employs the drag-and-drop approach to create blocks of script outside SL; the code from S4SL is pasted into an object in SL to add behavior and interactivity to otherwise static objects. Although Girvan et al (2013) use SLurtle for creating artefacts, the mapping is identical when S4SL is used as the programming tool as shown in **Figure 1**.

According to Girvan et al. (2013), the experiential learning cycle of **Figure 1** provides learners with concrete experience which they can observe and reflect upon. The learners can reassess the code by comparing the visualization at the concrete experience stage with the original plan.

Problem Based Learning

This paper is inspired by experiential learning theory and

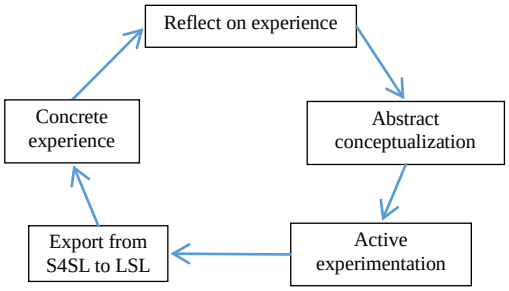


Figure 1.
Stages of experiential learning and stages to create programmable artefacts in SL. Adapted from Girvan et al. (2013).

PBL. Torp and Sage (2002) highlight the similarity between these pedagogies by defining PBL as focused, experiential learning organized around the investigation and resolution of messy, real-world problems.

PBL was first introduced in medical education over 30 years ago; however, it has since been used in a wide range of disciplines (Savery, 2006). Problem based learning (PBL) is considered effective for programming courses (Hwang et. al., 2008). Savery (2006) summarized the characteristics of PBL and notes that the key to success is the selection of ill-structured problems and a teacher who guides the learning process. PBL is described as facilitated problem-solving by Hmelo-Silver (2004). In PBL, problems once articulated by the teacher act as catalysts that initiate the learning process (Duch, 2001). Effective problems engage students and motivate them to gain a deeper understanding (Duch, 2001) by allocating more responsibility to the students (Esteves, Fonseca, Morgado, & Martins, 2009).

PBL in computer programming is afforded in SL by virtual objects. A programming tool in SL is used to define the behavior of these objects; learners analyze the behavior of these objects. Problems are formulated around computer programming; students are tasked to replicate or enhance the behavior of virtual objects by using a programming tool in SL. Visualization plays an important role in confirming the successful solution to a problem; the highly visual feedback allows the students to relate the program to the behavior of the object (Esteves, Fonseca, Morgado, & Martins, 2010). If the behavior of an object is not as expected then the student has an opportunity to reflect and revisit the problem.

Evaluation

In order to address the research question, SL is evaluated as a learning environment for computer programming concepts. The instrument is described in Section 3.1. Experimental Setup is described in Section 3.2. Data collection and data analysis is addressed in Section 3.3.

Instrument

It is proposed that content experts evaluate the system before it is made available to learners. Technology Acceptance Model (TAM) proposed by Davis (1989) is widely used for evaluating acceptance of end-user computing technologies. Theoretical grounding of TAM is the theory of reasoned action (TRA) proposed by Fishbein and Ajzen (1975). According to TRA, beliefs influence attitudes, which in turn lead to intentions,

which generate behaviors. The relationship between belief, attitude, intent, and behavior is adapted by TAM.

The premise of TAM are: first, users tend to use technology if they believe that it will help them perform better; second, potential users who believe in the usefulness of the technology may still reject it if they find that it is not easy to use (Chen, Chiu, & Wu, 2010). Based on these premise, Davis (1989) proposed two constructs, namely, perceived usefulness (PU) and perceived ease-of-use (PEOU) as indicators of intent to use a technology i.e. technology acceptance (TA); he used an experiment with email and graphics to validate TAM. Constructs proposed in TAM have been further validated in a number of studies, for example, Matheison (1991), Adams, Nelson and Todd (1992), and Chau (1996). TAM constructs are comprised of items listed below for PU and PEOU.

PU and PEOU (**Table 1**) are adapted to evaluate SL/S4SL as a learning environment for computer programming concepts; the adapted items are shown in the Data Analysis section (Section 3.3). Job Performance item in PU is regarded irrelevant; hence, it is removed from the questionnaire.

Cronbach's α is used as a measure of internal consistency reliability of the items in PU and PEOU. High values of alpha are used as evidence that the items have an underlying correlation. Cronbach's alpha is computed as below.

$$\alpha = \frac{N \cdot \bar{c}}{\bar{v} + (N-1) \cdot \bar{c}} \quad (1)$$

Here N is equal to the number of items, $\bar{c}$ is the average inter-item covariance among the items and $\bar{v}$ equals the average variance. Cronbach's α of 0.8 is considered high enough in most social science experiments (Carmines & Zeller, 1979).

Experimental Setup

Faculty members and students at a large Australian university were invited to participate in an empirical study about the use of SL as a learning environment for computer programming. The potential participant did not need prior experience of SL, although some programming knowledge was required. Student participants were given a $30 Kmart ™ voucher as incentive to participate.

Robins et al. (2003) proposed a three-pronged comprehensive approach to teach/learn programming, addressing knowledge required to design, generate and evaluate programs. Here, we focus on reinforcing programming concepts rather than comprehensive programming. Therefore, the proposed experiments do not require students to generate programs. The focus is removed from programming language syntax and directed towards evaluating programs i.e. reviewing programs and giving comments (Lister & Leaney, 2003). The empirical study implicitly addresses knowledge of program design.

The staircase example of Rosenbaum (2008) is used to develop experiments for the empirical study. Rosenbaum (2008) uses S4SL to create LSL; the script is embedded in an object in SL. When the object is touched by an avatar, it creates a staircase. The final resting position of the object is at the end of the staircase. Visualization of the staircase example of Rosenbaum (2008) is shown in **Figure 2**.

Experiments to address programming concepts of *variables*, *iterations* and *conditional statements* are described below. Visualization is used to confirm that the correct solution is achieved for each problem and multiple attempts may be required to obtain the correct solution.

Table 1.
TAM constructs.

Perceived Usefulness	*Perceived Ease Of Use*
Work more quickly	Easy to learn
Job Performance	Clear and understandable
Increased Productivity	Easy to become skillful
Effectiveness	Easy to use
Makes job easier	Controllable
Useful	Flexible

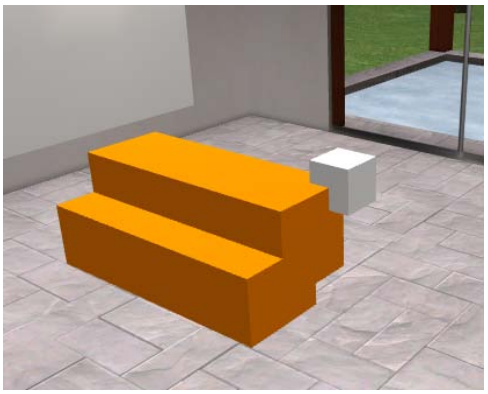


Figure 2.
Staircase is created when the object is touched by an avatar.

Variables

Variables are used for temporary storage of data. A variable is demonstrated by defining a scenario in which after completion of the staircase, the virtual object returns to the position it had at the start. The psuedocode for implementing this scenario is the following:

variable (namely, *variable_location*) is used to record the position of the object
create a pair of stairs
object is returned to position specified by *variable_location*

Two blocks of code highlighted in **Figure 3** demonstrate the use of variables. *set home to here* assigns the current coordinates of the object to a variable; *go home* will return the object to the location stored in the variable. LSL generated from **Figure 3** is embedded in an object in SL. When the object is touched by an avatar, the script responds by creating a staircase; after creating the staircase the object returns to its initial position as shown in **Figure 4**.

Iteration Statement

An iteration statement is used in computer programming to implement an iterative task. The psuedocode for iteration statement is:

```
for variable = initialvalue to finalvalue
   statement 1
...
   statement k
end
```

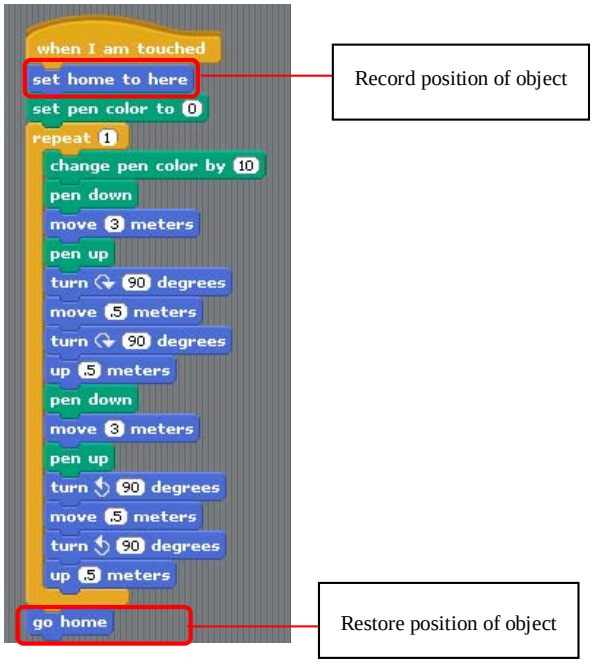


Figure 3.
Restore position of object after staircase is created.

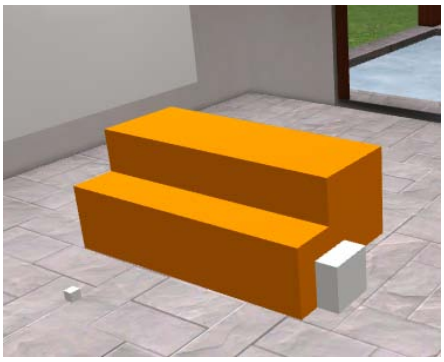


Figure 4.
Visualization of the experiment.

In the staircase example of Rosenbaum (2008), when an object is touched, it creates a single pair of stairs (**Figure 2**). In this section, multiple pairs of stairs are created. The psuedocode for implementing this scenario is the following:

for variable = initialvalue to finalvalue
  create single pair of stairs
end

The number of pairs of stairs will depend on *initialvalue* and *finalvalue* in the psuedocode above. The scenario is implemented in S4SL by using *repeat* as shown in **Figure 5**. The number of stairs is controlled by the *counter* supplied to the *repeat* statement.

LSL generated from **Figure 5** is embedded in an object in SL. When the object is touched by an avatar the script responds by creating a staircase. Visualization of the experiment is shown in **Figures 6** and **7**.

Conditional Statement

A conditional statement is executed when a predefined con-

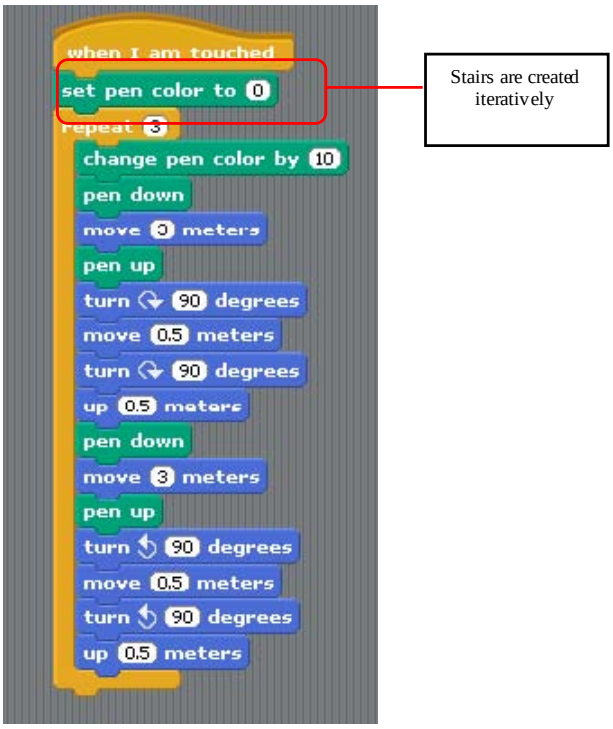


Figure 5.
Create a staircase using an iteration statement.

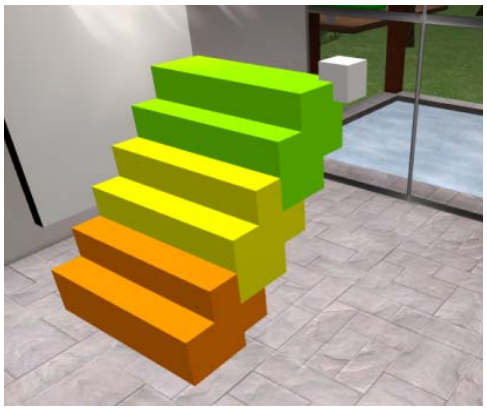


Figure 6.
Visualization of the experiment.

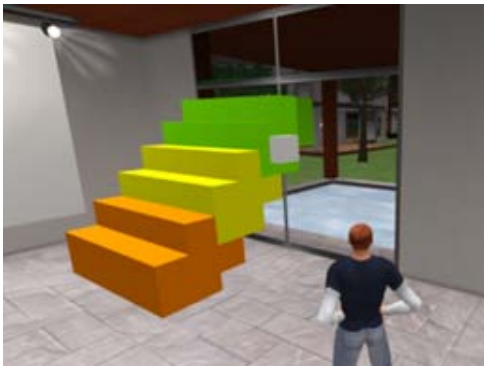


Figure 7.
Visualization of the experiment.

dition holds good. The psuedocode for conditional statement is:

if (condition exists) then (execute statements)

Hence, an expression is evaluated to decide if a pre-defined condition holds good; if the condition holds good, a set of statements is executed otherwise the set of statements is ignored.

We consider a scenario which is an enhancement of the problems in Sections 3.2.1 and 3.2.2. In this scenario, when a virtual object is touched, it creates a staircase similar to Section 3.2.2; however, after completion of the staircase, the virtual object returns to the position it had at the end of the second iteration. The psuedocode for implementing this scenario is the following:

1. variable_counter is initialized to 0
2. for variable_loop = 1 to finalvalue
3. variable_counter is incremented
4. create single pair of stairs
5. if (variable_counter equals 2)

then (remember the current position)

6. end
7. object returns to the position which was stored in line 5

LSL generated from **Figure 8** is embedded in an object in SL. When the object is touched by an avatar, the script responds by creating a staircase. After the staircase is created, object returns to the position it held at the end of the second iteration.

To assess SL in the context of the research question, tasks were implemented in-world based on the experiments described in Section 3.2. The tasks are included in the Appendix. Participants were required to complete these tasks in-world. It was possible to observe the actions of the participant's avatar in-world and also provide in-world assistance. Participants documented their experience by completing a questionnaire. The questionnaire comprised four parts: Part 1 obtained background of the participant regarding their programming experience; Parts 2 and 3 had Likert-scale questions which were adopted with adaptation from the TAM questionnaire; Part 4 was used for qualitative data collection and comprised open-ended questions about the participant's experience.

Data Collection and Analysis

A total of 12 people participated in this study. They were all post-graduate students with 75% males and 25% females. Data from completed questionnaires is analyzed to derive conclusions about SL as a learning environment for computer programming concepts. Results from the Likert-scale questions in the questionnaire are summarized in **Tables 2** and **3**.

Cronbach's α of 0.868 is obtained for the 11 Likert-scale questions. Some feedback from open-ended questions in the questionnaire is given below.

Q: Did you find the tasks interesting?

i. The activities are interesting and very useful for beginners who want to clear the concepts of programming.

ii. I have been doing programming for two months and I am amazed to see what we can do with programming.

iii. It is easy to learn programming by visualization.

iv. It is more interesting than just write down or read code.

v. It is user friendly for beginner to learn programming

Q: Do you expect the virtual presence of the teacher to be helpful for completing activities in this environment?

i. At the time of the activity, I had doubts and the virtual presence of the teacher clarified my doubts.

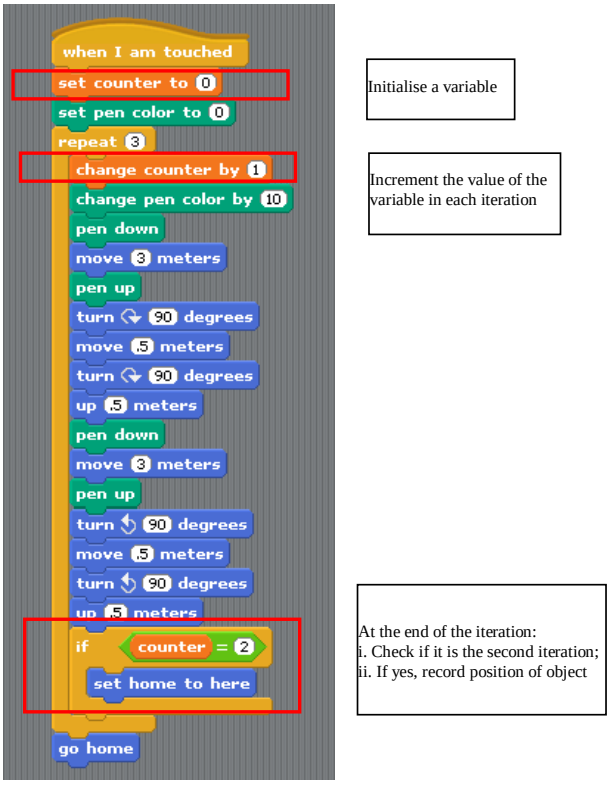


Figure 8.
Record the position of the object at the end of the second iteration; return the object to this location after all stairs are created.

ii. I do favor the concept of a teacher because sometimes you need help when you are stuck and programming needs some help in one way or the other

iii. Virtual presence of the teacher is helpful as teacher can always guide or give the solution if any doubt or problem is there.

iv. It is really good to have virtual presence of someone who can guide.

v. Virtual presence of teacher could help me to improve my skills because I am not nervous when I encounter the virtual presence of the teacher.

vi. Yes, I think it is helpful. When I have questions, I can chat with the virtual presence of the teacher.

Q: Do you expect the virtual presence of the peers to be helpful for completing activities in this environment?

i. It is really interesting to learn with peers in this environment.

ii. Virtual presence of peers may distract the learner from his/her task.

iii. Yes, SL provides a good social community. Peers could learn programming and share what they learn.

Q: Any other comments

i. It can explain more programming concepts visually and we can learn fast. It is interesting to learn here.

ii. SL is good platform for understanding programming concepts as well as for entertainment.

iii. It is really useful to understand programming fundamentals but implementing or understanding the logic is all up to the learners

iv. I have not experienced SL but after doing the tasks, I am

Table 2.
Perceived ease of use.

Questions in respect to SL/S4SL tasks.	SA	A	N	D	SD	Mean	Std Dev
1). *Easy to learn* SL/S4SL provides an easy to learn approach for programming concepts	41.7%	41.7%	16.7%	0%	0%	4.25	0.754
2). *Clear and understandable* SL/S4SL tasks makes the programming concepts clear and understandable	50%	41.7%	8.3%	0%	0%	4.33	0.888
3). *Easy to become skillful* It is easy to become skillful in programming tasks in SL/S4SL	16.7%	50%	25%	8.3%	0%	3.75	0.866
4). *Easy to use* SL/S4SL is an easy to use environment for attempting programming tasks	25%	33.3%	41.7%	0%	0%	3.83	0.834
5). *Controllable* SL/S4SL is a controllable environment for attempting programming tasks	25%	58.3%	16.7%	0%	0%	4.08	0.668
6). *Flexible* SL/S4SL provides a flexible environment for learning programming concepts	41.7%	58.3%	0%	0%	0%	4.42	0.515

SA: Strongly Agree; A: Agree; N: Neither Agree or Disagree; D: Disagree; SD: Strongly Disagree.

Table 3.
Perceived usefulness.

Questions in respect to SL/S4SL tasks.	SA	A	N	D	SD	Mean	Std Dev
7). *Work more quickly* I can perform the programming tasks quickly in SL/S4SL	33.3%	58.3%	8.3%	0%	0%	4.25	0.621
8). *Increased Productivity* SL/S4SL improves my learning efficiency of programming concepts	50%	33.3%	16.7%	0%	0%	4.33	0.778
9). *Effectiveness* SL/S4SL is effective for learning programming concepts	58.3%	41.7%	0%	0%	0%	4.58	0.515
10). *Makes job easier* SL/S4SL makes it easier to learn programming concepts compared with other environments	16.7%	41.7%	41.7%	0%	0%	3.75	0.753
11). *Useful* SL/S4SL is useful for learning programming concepts	58.3%	25%	8.3%	8.3%	0%	4.33	0.984

SA: Strongly Agree; A: Agree; N: Neither Agree or Disagree; D: Disagree; SD: Strongly Disagree.

interested in SL.

v. I think this is good for students who pay attention to details otherwise they are not able to identify the difference between objects.

Conclusions and Future Work

The research question addressed in this paper is the extent to which SL facilitates an understanding of computer programming concepts. An empirical study, based on problem-based learning and experiential learning approaches was conducted. Participants completed a questionnaire comprising Likert-scale questions and open-ended questions. Responses to Likert-scale questions (adopted from TAM questionnaire) paint a positive picture of SL with regard to PEOU and PU in the context of the research question. Cronbach's α of 0.868 is obtained for the Likert-scale questions; it indicates a high level of internal consistency reliability of the items in PEOU and PU. Responses to open-ended questions show that participants responded positively to the experiments; however, few participants got distracted by the inherent characteristics of virtual worlds, e.g., in-world social interactions; in-world entertainment is also a likely distraction.

In the future, it is proposed that constructivist learning be incorporated to enhance the learning experience. In constructivist approach, learners build on their existing knowledge by applying prior knowledge to solve real-world problems (Hadjerrouit, 2008). This approach to learning focuses on the process of creating and sharing artefacts which are personally meaningful (Girvan, Tangney, & Savage, 2013). Girvan & Savage (2010) have argued the alignment between affordances of SL and constructivist pedagogies. SL tools which afford the construction of persistent objects use features of this pedagogy (Girvan, Tangney, & Savage, 2013). In line with improved pedagogical approaches in computer programming, preliminary data suggest that SL provides a constructivist environment for building skills and knowledge.

Acknowledgement

We acknowledge that this paper is financially supported by Parallel and Distributed Computing Lab., School of Informa-

tion Technology, Deakin University, Australia.

REFERENCES

Adams, D. A., Nelson, R. R., & Todd, P. A. (1992). Perceived usefulness, ease of use and usage of information technology: A replication. *MIS Quarterly, 16,* 227-247. http://dx.doi.org/10.2307/249577

Ben-Bassat Levy, R., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education, 40,* 1-15. http://dx.doi.org/10.1016/S0360-1315(02)00076-3

Boulos, M. N. K., Hetherington, L. & Wheeler, S. (2007). Second Life: an overview of the potential of 3-D virtual worlds in medical and health education, *Health Information & Libraries Journal, 24,* 233-245. http://dx.doi.org/10.1111/j.1471-1842.2007.00733.x

Britzman, D. (2003). Practice makes practice: A critical study of learning to teach. New York State University: New York Press.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: A visual programming environment for teaching algorithmic problem solving. *ACM SIGCSE Bulletin, 37,* 176-180. http://dx.doi.org/10.1145/1047124.1047411

Carmines, E. G., & Zeller, R. A. (1979). Reliability and validity assessment. *Sage University Paper 17.* Beverly Hills: Sage Publications.

Casamayor, A., Amandi, A., & Campo, M. (2009). Intelligent assistance for teachers in collaborative e-learning environments. *Computers & Education, 53,* 1147-1154. http://dx.doi.org/10.1016/j.compedu.2009.05.025

Chau, P. Y. K. (1996). An empirical assessment of a modified technology acceptance model. *Journal of Management Information Systems, 13,* 185-204.

Chen, M.-P., Chiu, C.-H., & Wu C.-C. (2010). Instructional simulations for teaching high school computer science concepts: A technology acceptance perspective. *IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning,* 216-218.

Dalgarno, B., Lee, M. J. W., Carlson, L, Gregory, S., & Tynan, B. (2010). 3D immersive virtual worlds in higher education: An Australian and New Zealand scoping study. Ascilite Sydney, 269-280.

Dann, W., Cooper, S., & Pausch, R. (2000). Making the connection: programming with animated small world. *ACM SIGCSE Bulletin, 32,* 41-44. http://dx.doi.org/10.1145/353519.343070

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology, *MIS Quarterly, 13,* 319-340. http://dx.doi.org/10.2307/249008

Dickey, M. D. (2003). Teaching in 3D: Pedagogical affordances and constraints of 3D virtual worlds for synchronous distance learning. *Distance education, 24,* 105-121. http://dx.doi.org/10.1080/01587910303047

Duch, B. J. (2001). Models for problem-based instruction in undergraduate courses. *The Power of Problem-Based Learning,* 39-46.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2009). Using second life for problem based learning in computer science programming. *Journal of Virtual Worlds Research, 2,* 3-25.

Fishbein, M., & Ajzen, I. (1975). Belief, attitude, intention, and behavior: An introduction to theory and research. Reading, MA: Addison-Wesley.

Guzdial, M., Kolodner, J. L., Hmelo, C., Narayanan, H., Carlson, D., Rappin, N., Hübscher, R., Turns, J., & Newstetter, W. (1996). Computer support for learning through complex problem-solving. *Communications of the ACM, 39,* 43-45. http://dx.doi.org/10.1145/227210.227600

Girvan, C., & Savage, T. (2010). Identifying an appropriate pedagogy for virtual worlds: A communal constructivism case study. *Computers & Education.* http://dx.doi.org/10.1016/j.compedu.2010.01.020

Girvan, C., Tangney, B., & Savage, T. (2013). SLurtles: Supporting constructionist learning in Second Life. *Computers & Education, 61,* 115-132. http://dx.doi.org/10.1016/j.compedu.2012.08.005

Hadjerrouit, S. (2008). Towards a blended learning model for teaching and learning computer programming: A Case Study. *Informatics in Education, 7,* 181-210.

Henderson, L., Grant, S., Henderson, M., & Huang, H. (2010). University students' cognitive engagement while learning in a Virtual World. *Australian Computers in Education Conference,* 6-9 April, Melbourne.

Hwang, W.-Y., Wang, C., Hwang, G.-J., Huang, Y.-M., & Huang, S. (2008). A web-based programming learning environment to support cognitive development. I*nteracting with Computers, 20,* 524-534. http://dx.doi.org/10.1016/j.intcom.2008.07.002

Jenkins, T. (2001). The motivation of students of programming. *Proceedings of ITiCSE 2001: The 6th Annual Conference on Innovation and Technology in Computer Science Education (*pp. 53-56). New York: ACM.

Journal of Virtual Worlds Research (2009). Pedagogy, education and innovation in Virtual Worlds.

Kolb, D. A. (1984). Experiential learning: Experience as the source of learning and development. Upper Saddle River, NJ: Prentice-Hall.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Computer Science Education, 13,* 249-268. http://dx.doi.org/10.1076/csed.13.4.249.17496

Kern, N. (2009). Starting a Second Life. http://slexperiments.edublogs.org/2009/03/03/starting-a-second-life/

Lister, R., & Leaney, J. (2003). First year programming: Let all the flowers bloom. *Proceedings of the Fifth Australasian Conference on Computing Education* (pp. 221-230). Adelaide.

Mathieson, K. (1991). Predicting use intentions: Comparing the technology acceptance model with the theory of planned behaviour. *Information Systems Research, 2,* 173-191. http://dx.doi.org/10.1287/isre.2.3.173

Menchaca, R., Balladares, L., Quintero, R., & Carreto, C. (2005). Software engineering, HCI techniques and Java technologies joined to develop web-based 3D-collaborative virtual environments. *Proceedings of the 2005 Latin American conference on Human-computer interaction* (pp. 40-51). New York, NY: ACM Press. http://dx.doi.org/10.1145/1111360.1111365

Miliszewska, I., & Tan, G. (2007). Befriending computer programming: a proposed approach to teaching introductory programming. *Journal of Issues in Informing Science & Information Technology, 4,* 277-289.

Newman, D., Griffin, P., & Cole, M. (1989). The construction zone: Working for cognitive change in school. New York: Cambridge University Press.

Novak, T. P. (2010). eLab city: A platform for academic research on virtual worlds. *Journal of Virtual Worlds Research, 3.*

Record voice chat and sounds. (2012). http://www.screencast.com/users/Featured/folders/Featured/media/cf26872f-fa66-418b-89e8-47d9f3a13b02

Robins, A., et al. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education. 13,* 137-172. http://dx.doi.org/10.1076/csed.13.2.137.14200

Rosenbaum, E. (2008). Scratch for second life. *Proceedings of the International Conference of the Learning Sciences-ICLS,* Utrecht, The Netherlands: ICLS., 144-152.

Sajjanhar, A. (2012). Virtual worlds for student engagement. *Creative Education, 3,* 796-801.

Savery, J. R. (2006). Overview of problem-based learning: Definitions and distinctions. *Interdisciplinary Journal of Problem-based Learning, 1,* 9-20. http://dx.doi.org/10.7771/1541-5015.1002

Second life bot. (2012). http://wiki.secondlife.com/wiki/Bot.

Sloane, K., & Linn, M. C. (1988). Instructional conditions in Pascal programming classes. In R. Mayer (Ed.), *Teaching and learning computer programming: Multiple research perspectives* (pp. 207-235). Hillsdale: Lawrence Erlbaum Associates.

Stamouli, I., Doyle, E., & Huggard, M. (2004). Establishing structured support for programming students. *34th Annual Conference on Frontiers in Education, 2,* F2G - 5-9.

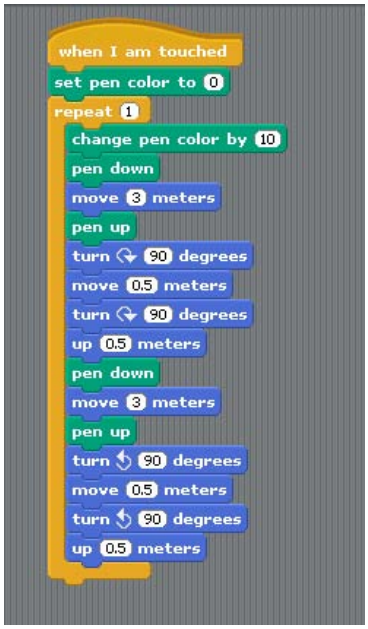
Torp, L., & Sage, S. (2002). *Problems as possibilities: Problem-based learning for K-16 education* (2nd ed.). Alexandria, VA: Association for Supervision and Curriculum Development.

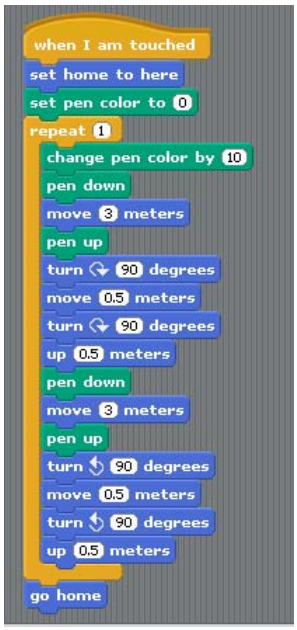
Warburton, W. (2009). Second Life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. *British Journal of Educational Technology, 40*, 414-426. http://dx.doi.org/10.1111/j.1467-8535.2009.00952.x

Wrzesien, M., & Raya, M. A. (2010). Learning in serious virtual worlds: Evaluation of learning effectiveness and appeal to students in the E-Junior project. *Computers & Education, 55*, 178-187. http://dx.doi.org/10.1016/j.compedu.2010.01.003

Appendix—Experiment 1

S4SL code in objects Task 1-Object A and Task 1-Object B is shown below.





Use objects Task 1-Object A and Task 1-Object B from the *Second Life Inventory* to visualize the two pieces of code and answer the questions below.

1. Identify the difference in the code.

_____

_____

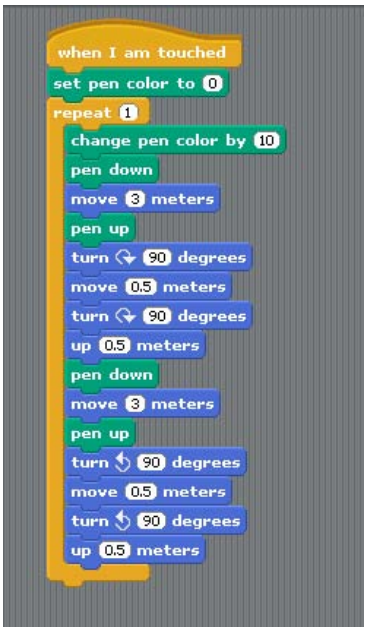
2. How is the difference reflected in the visualization?

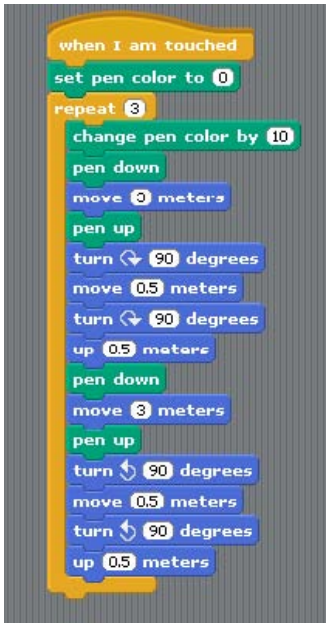
_____

_____

Appendix—Experiment 2

S4SL code in objects Task 2-Object A and Task 2-Object B is shown below.





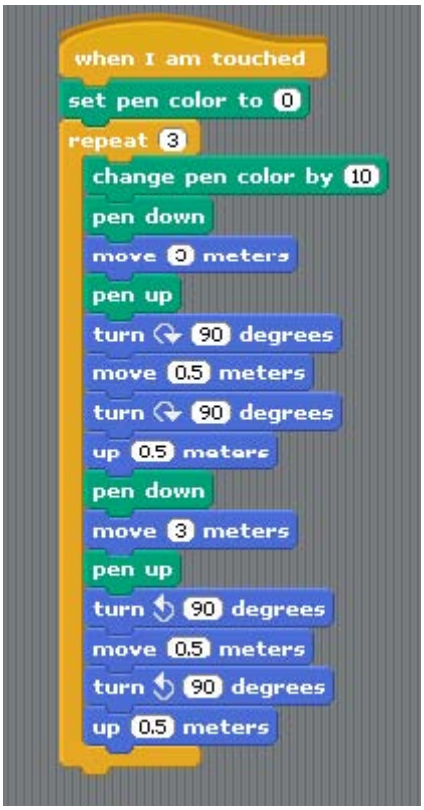
Use objects Task 2-Object A and Task 2-Object B from the *Second Life Inventory* to visualize the two pieces of code and answer the questions below.

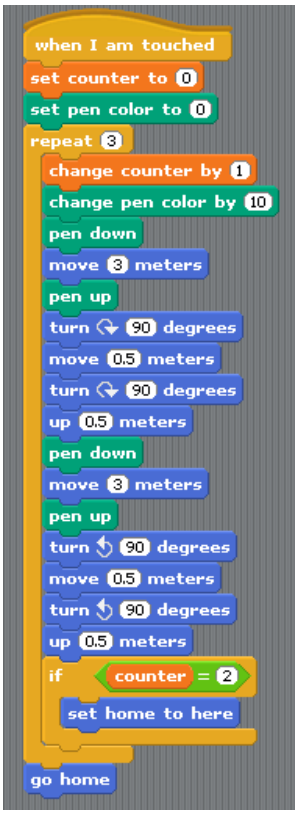
1. Identify the difference in the code.

2. How is the difference reflected in the visualization?

Appendix—Experiment 3

S4SL code in objects Task 3-Object A and Task 3-Object B is shown below.





Use objects Task 3-Object A and Task 3-Object B from the *Second Life Inventory* to visualize the two pieces of code and answer the questions below.

1. Identify the difference in the code.

2. How is the difference reflected in the visualization?