MONASH University

Exploring Second Life as a Learning Environment for Computer Programming

Atul Sajjanhar

B.Eng.(SGSITS, Indore), Grad. Dip. App. Sc.(Deakin), M.Comp.(Monash), Ph.D(Monash)

A thesis submitted for the degree of Master of Education at

Monash University in 2017

Faculty of Education

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: [redacted]

Name: Atul Sajjanhar

Date: 27 June 2017

Acknowledgement

I would like to extend a big thank you to my Principal Supervisor Dr. Julie Faulkner for her guidance and support. I am very thankful to Dr. Hazel Tan for her feedback and extensive comments on the draft versions of the thesis.

I am thankful to my amazing friends I met during the weekends and evenings spent in the teaching and learning space at the Clayton Campus of the Faculty of Education. This includes Angela, Daniel, Lainie and Huiling. I thank my parents for their support and encouragement.

I thank Monash University for the opportunity to study the Master of Education. Since I am working fulltime, it would have been very difficult to complete the thesis without the flexibility offered at Monash University.

Table of Contents

List of Tables

List of Figures

Abstract

The premise of the study is that novice programmers find it challenging to grasp computer programming concepts because of the abstract nature of these concepts. This study gauges the perceptions of novice computer programmers about Second Life (SL) as a digital learning environment for concept understanding in computer programming. In this research, I focus specifically on SL because it is the most mature (robust and feature-rich) of 3D virtual world platforms, which is reflected by its high usage figures compared with other competing platforms. SL is widely used by educators as a learning environment because it offers, within a constructivist frame, the use of learner-centered pedagogies for designing curriculum.

The overarching research question that steers the study is: *To what extent do IT students find SL helpful in understanding programming concepts?* An ancillary research question evaluates the usability of SL. The concept of software usability is based on the principles of human-computer interaction; usability is a measure of usefulness (that is, effectiveness in accomplishing tasks) and ease-of-use (that is, user-friendliness) of a software application. This research question serves to distinguish the usability of the tool, namely SL, from the instructional process.

I adopt a case study as the research methodology, and conduct a small-scale study. The participants in the study possessed novice programming skills, and they engaged in activities designed for achieving the learning objectives. Qualitative and quantitative data collected from multiple sources of evidence were interpreted and analyzed to seek responses to the research questions.

Generally, the findings of the study were found to strongly support the use of SL for novice programmers to learn programming, and the usability of SL. The findings suggest positive support offered by SL although the findings cannot be used to make firm conclusions due to the small-scale nature of the study. Dissemination of findings will potentially lead to more effective pedagogies to communicate abstract concepts in contexts other than computer programming, and thus have even wider implications.

Chapter 1

Introduction

1.1. Background

Computer Programming is recognized by educators as an inherently complex intellectual activity (Teague, 2011). Failure rates in first-year computer programming courses have been a contentious issue, with higher education institutions reporting figures as high as 40% (Han & Beheshti, 2010; Lang, McKay, & Lewis, 2007).

Learning computer programming requires a hierarchy of skills (Sloane & Linn, 1988), 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, Fonseca, Morgado, & Martins, 2009). Low level computer programming skills are used to generate syntactically correct computer programs, or articulate conceptual solutions in a programming language. The focus of this research is on supporting the learning of high level computer programming skills. Such skills require a comprehension of abstract programming concepts which tend to be difficult to grasp because they do not have any real-life equivalent model (Dunican, 2002).

Failure to grasp computer programming skills can lead to de-motivation 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, Doyle, & Huggard, 2004). In recent years, there has been a significant drop in the number of students enrolling in computer science courses; attrition from these courses also remains high (Beaubouef & Mason, 2005; Lang, McKay, & Lewis, 2007; Soh, Samal, & Nugent, 2007; Teague, 2011).

In this chapter, I discuss the challenges in teaching computer programming in higher education institutions. I reflect on my professional experience, in my capacity as a university lecturer in IT, which motivated me to undertake this study. In the following sections, the research questions are articulated and the road map of the thesis is outlined.

1.2. Motivation

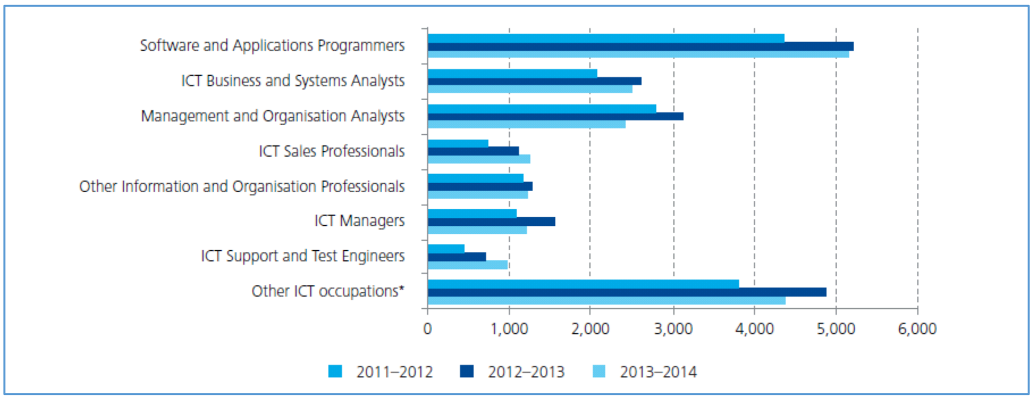
As a Lecturer in Information Technology at an Australian university, I have been involved in teaching courses related to programming and software development for almost 15 years. These courses consistently have high failure rates. Close analysis of the results, finds that most students who fail the unit appear to give up because they do not complete formal assessment tasks, or do not attempt the tasks at all. On the other hand, students who pass the unit usually pass with good marks which is reflected in the skewed normal distribution of results. The underpinning issue appears that novice programmers find it challenging to grasp key concepts related to programming. I have an appreciation of the findings of Buck and Stucki (2001), and Lister and Leaney (2003) who state that pedagogical approaches for most computer programming courses lack emphasis on programming concepts. These findings corroborate my teaching experience because the courses I have taught focus on software deliverables which inherently emphasize syntactical learning objectives.

I am also motivated by anecdotal evidence from discussions with students and lecturers which reflects an inherent apprehension amongst a large majority of students to study programming curriculum. These views are often unsubstantiated with concrete evidence, and emerge from long-held negative perceptions of programming as a profession. The inability to successfully complete a programming course is an opportunity lost to pursue a potentially lucrative career. However, attitudinal shifts will need broad systemic adoption of changes in curriculum and is outside the scope of this study. I am, therefore, also motivated by practical and intellectual goals of this research, as addressed by Maxwell (2013). The practical goal of this research is focused on accomplishing an objective which is stated as follows: subject to the findings of this study being positive, I would make a case for using the proposed pedagogical approaches in introductory computer programming courses at the university where I teach. The intellectual goal of this research refers to gaining insight into pedagogical approaches which may be adopted to counter the challenges faced by novice programmers to understand computer programming concepts.

1.3. Significance of the Study

The number of Australian graduates with Information and Communications Technology (ICT) qualifications has declined significantly since the peak achieved in early 2000s (Gliddon, 2012). ICT is an umbrella term which includes a range of areas, namely, computer science, information technology, and information systems. The attrition from ICT courses is an indicator of systemic failure in curriculum delivery. This shortfall in ICT graduates in Australia, is further exacerbated by lucrative offers from Silicon Valley based companies. It is important that higher education institutions in Australia aim to increase the future pipeline of ICT graduates.

The significance of the ICT workforce in Australia cannot be understated. According to a report jointly published by the Australian Computer Society and Deloitte Australia (Australian Computer Society, 2015), Australia’s digital economy represented 5% of the GDP in 2013-14. Despite off-shoring of ICT work, the demand for ICT professionals in Australia is robust. Australia’s ICT workforce in 2014 was 600,000 and is expected to increase to 700,000 by 2020. In recent years, 10,000 temporary skilled migration visas (category 457) have been issued annually to recruit ICT professionals from overseas, to fill the skills shortage. The net migration of ICT workers into Australia, is shown in Figure 1.1. As mentioned above, ICT covers a range of areas but the majority of ICT professionals recruited from overseas fill positions in the area of software development and programming which is an integral component of an IT course, and indeed the focus of this study.



Note. Excludes ICT industry admin and logistics support for which breakdowns are unavailable; electronic trades and professional data is for all industries

Figure 1.1. Net Migration of ICT Workers into Australia

From "Net Migration of ICT Workers into Australia" by Department of Immigration and Border Protection (https://www.border.gov.au/about/reports-publications/research-statistics/statistics/). In the public domain.

The significant number of migrant IT workers arriving into Australia each year, is indicative of the immediacy of skills shortage in the area of software development and programming.

The underlying problem which needs to be addressed, I argue, is systemic failure in curriculum delivery of computer programming courses in higher education institutions. Addressing this underlying problem will reduce the failure rates in programming courses, and alleviate the attrition rates of students from IT courses. This will result in more students successfully completing IT courses.

1.4. Scope of the Study

The range of skills which need to be acquired for computer programming, and the relevant affordances in learning environments form the basis of this study. For instance, according to extant literature, collaboration is an effective approach for computer programming (Guzdial et al., 1996; Menchaca, Balladares, Quintero, & Carreto, 2005) and learning generally. Collaboration is significant because it stimulates learning, promotes feelings of belonging to a team, encourages creativity, facilitates communication and increases personal satisfaction in relation to achievement (Casamayor, Amandi, & Campo, 2009). Collaborative environments can offer important support to students in their activities for computer programming. According to Newman, Griffin, and Cole (1989), collaboration in problem solving provides not only an appropriate pedagogy but also promotes reflection, a mechanism that enhances the learning process. Some environments which have been used to teach introductory computer programming include: ALICE which was developed by Dann, Cooper, and Pausch (2000), JELIOT which was developed by Ben-Bassat Levy, Ben-Ari, and Uronen (2003), BlueJ which was developed by Kölling, Quig, Patterson, and Rosenberg (2003), and RAPTOR which was developed by Carlisle, Wilson, Humphries, and Hadfield (2005). 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 makes these environments conducive for teaching high level computer programming skills. However, the inherent design drawback of these environments is the lack of the facility for collaboration amongst learners. Having established the significance of collaboration, I explore virtual worlds because they provide a collaborative learning environment which afford a contextual, embodied experience and have the potential to offer student engagement aligned with real-world experiences (Sajjanhar, 2012; Sajjanhar & Faulkner, 2014). There are several vendors which provide a Multi-User Virtual Environment (MUVE) or virtual world environments for teaching/learning. Second Life (secondlife.com) is a 3D virtual world developed by Linden Lab. Another popular 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 research, I focus specifically on Second Life (SL) because it is the most mature (robust and feature-rich) of virtual world platforms, which is reflected by its high usage figures compared with other competing platforms (Dalgarno, 2010; Warburton, 2009). SL is widely used in education (cf. Journal of Virtual Worlds Research, 2009). SL is used in medical and health education (Boulos, Hetherington, & Wheeler, 2007), and is used for teaching languages, including Arabic (Kern, 2009) and Chinese (Henderson, Grant, Henderson, & Huang, 2010). It is also used for researchers to collaborate (Novak, 2010). The pedagogical potential of SL is further addressed in the literature review in Chapter 2.

SL provides a collaborative learning environment thereby enhancing opportunities for student engagement. SL has a proprietary scripting language, namely, Linden Scripting Language (LSL), which is used to control the behavior and interactivity of virtual objects. LSL can be used to manifest real-world concepts in the virtual world. Hence, LSL can be used to acquire higher level computer programming skills such as problem analysis and conceptualization. Second Life provides a means for educators to use various pedagogies to enhance learning of computer programming. In this study, I allude to educational psychology and sociocultural pedagogies to construct a holistic proposal for learning computer programming concepts. The proposal takes into account the drawbacks of existing approaches. I analyze and evaluate the data emerging from my empirical study. The study is conducted in an Australian university and the participants are students studying introductory computer programming.

1.5. Research Questions

The aim of the study is to construct an enhanced learning environment for IT students' understanding of programming concepts. The participants in the study will have novice programming skills and the overarching research question which will steer the study is: *To what extent do IT students find SL helpful in understanding programming concepts?* This research question is henceforth referred to as RQ1. The study will gauge the perceptions of novice computer programmers about SL as a learning environment for concept understanding in computer programming.

An ancillary research question, henceforth referred to as RQ2, will evaluate the usability of SL: *How usable do participants find the SL environment?* The concept of software usability is based on the principles of human-computer interaction; usability is a measure of usefulness (that is, effectiveness in accomplishing tasks) and ease-of-use (that is, user-friendliness) of a software application. This research question, then, will serve to distinguish the role of the tool, namely SL, from the instructional process.

1.6. Thesis Outline

The thesis comprises five chapters. An overview of the following chapters is provided below.

Chapter 2: Literature Review

In Chapter 2, I review relevant learning theories and pedagogical approaches which relate to my proposed study. I comment on the approaches that education researchers have used in order to construct my own study.

This chapter reviews and critiques existing approaches for teaching computer programming courses in higher education institutions. I address the role of sociocultural concepts such as the Zone of Proximal Development and scaffolding. I examine the broad categories of teacher-centered and learner-centered approaches. Further, I describe and critique commonly used learning environments for teaching computer programming and discuss the affordances of SL for student engagement. I explore the pedagogical potential of SL in the light of approaches that education researchers have used. I identify the potential and the limitations of SL as an immersive environment for engaging students in computer programming. Further, I address persuasive computing tools, and the role of such tools in the learning domain.

Chapter 3: Methodology and Design

In Chapter 3, I justify the adoption of case study methodology for my study. I provide an overview of the empirical study and describe a mixed-methods approach for gathering data. I describe the empirical activities, designed for the participants, which are used as sources of evidence to assess understanding of high-level programming concepts amongst novice programmers.

Chapter 4: Findings

In Chapter 4, I describe the data collected from multiple data sources. I analyze and interpret the data to respond to the research questions within the context established through the literature review. Qualitative data are used to draw conclusions about RQ1. Quantitative data are used to address RQ2.

Chapter 5: Conclusion

In Chapter 5, I conclude the thesis by summarizing the findings of the research in the context of the research questions, and discuss the strengths and limitations of the study. I suggest approaches for future research directions and draw conclusions from the study.

Chapter 2

Literature Review

2.1. Introduction

In this chapter, I review and critique existing approaches for teaching computer programming courses in higher education institutions, addressing the importance of pedagogy and the learning environment. The learning theories discussed in Section 2.2, inform the pedagogical approaches discussed in Section 2.5. I emphasize the constructivist learning theory which has been the dominant learning theory since the 1980s. Widely accepted benchmarks for student engagement are described in Section 2.3. In Section 2.4, I discuss the taxonomy of learning objectives. Identifying the learning objectives is important to design appropriate learning activities. The activities used in this study are formulated in the next chapter. In Section 2.5, I review and critique pedagogical approaches which are categorized as teacher-centered and learner-centered approaches. In reviewing the pedagogical approaches, I stress the value of learner-centered approaches which are grounded in the constructivist learning theory. In Section 2.6, I describe commonly used learning environments for teaching computer programming. I critique these learning environments based on pedagogical limitations such as the lack of collaborative facilities, and the lack of a contextual embodied experience. These limitations constrain the level of student engagement in learning activities. I propose using a virtual world environment to overcome the limitations of existing learning environments, in the context of computer programming. In Sections 2.7, I give an introduction of virtual world environments, and discuss Second Life (SL) in the context of educational settings. In Section 2.8, I explain the building blocks of SL which are eventually used by me to develop a virtual classroom for conducting an empirical study to seek answers for the research questions. SL is addressed from a pedagogical perspective in Section 2.9; I explore the pedagogical potential of SL by commenting on the approaches that researchers have used. Further, I describe the potential and the limitations of SL as an immersive environment for engaging students in order to construct my own study. In Section 2.10, I address conceptual frameworks for designing persuasive computing tools which are used to influence attitude and behavior, and the role of such tools in the learning domain. In Section 2.11, I draw conclusions by summarizing the directions of this study which are informed by extant literature.

2.2. Learning Theory

Although learning is an important topic in psychology, it is difficult to define. A widely accepted definition of learning as a psychological phenomenon is: *a relatively permanent change in behavioral potentiality that occurs as a result of reinforced practice* (Kimble, 1961, p. 6). From a socio-cultural perspective, it is situated practices which emphasizes the inter-dependence of social and individual processes for the co-construction of knowledge (John-Steiner & Mahn, 1996). Learning theories are conceptual frameworks to explain the learning process. Learning theories have evolved to reflect changes in societal attitudes and expectations of education. Education researchers have proposed learning theories based on the influence of factors such as cognition, environment, and background of the learner.

Behaviorist learning theory was popular amongst psychologists in the early 19th century. The theory was founded by Pavlov, Skinner, and Thorndike (Mergel, 1998). This theory assumes that the learner is passive and responds to external stimuli. According to this theory, reinforcement and punishment can be used to change the behavior of the learner. The theory has been widely criticized because it lacks comprehensiveness, and fails to explain complex human behavior.

A widely accepted learning theory is the constructivist theory. Proponents of constructivist learning believe that knowledge cannot be acquired mechanistically thereby discounting the orderly, predictable, and controllable view of the universe as being adequate to explain the active role of the student, and the social interaction in educational settings (Liu & Matthews, 2005). The guiding principles for constructivist learning are as follows (Hein, 1991, p. 3).
Active process: the learner needs to engage in the learning process because it is not the passive acceptance of knowledge. Since learning is a mental process, it should engage the mind although it may be complemented by physical activities.
Social activity: in traditional learning the learner is not encouraged to interact socially; education is seen as a relationship between the learner and the material to be studied. Constructivist learning regards social interaction as essential to learning.
Contextual: the learner needs to understand how the knowledge can be used. Hence, learning cannot be isolated from the learner's prior knowledge and experience. To gain new knowledge, it is important to construct a path for the learner based the learner's prior knowledge and experience.
Motivation: a key component of constructivist learning is that the learner should be motivated. Making the learning process active, social and contextual motivates the learner.

The theory of multiple intelligences was developed by Howard Gardner in 1983 (Howard, 1983), and is widely represented in constructivist learning. According to the concept of multiple intelligences, human intelligence is multi-dimensional rather than the traditional view that intelligence is a construct characterized by a measure of IQ. The role of constructivist learning is to engage these multiple intelligences to create new cognitive structures. Hence, it is challenging for the teacher because the teacher needs to understand the current level of cognitive development of each student. In constructivist learning approaches, learners build on their existing knowledge by applying prior knowledge and experience rather than passively absorbing what is presented to them (Hadjerrouit, 2008; Teague, 2011). According to this pedagogy, learning is achieved by integrating prior knowledge and experience with new knowledge and context (Vrasidas, 2000).

Jean Paiget and Lev Vygotsky are two eminent figures in the development of constructivist learning theory. However, there exists a polarization between the Piaget's and Vygotsky's paradigms. Piaget stresses the importance of active participation of students; this variant of constructivist learning is referred to as the cognitive constructivist learning (Felix, 2005). Other prominent theorists affiliated with cognitive constructivist learning are Bruner, Ausubel and van Glasersfeld. On the other hand, Vygotsky stresses the importance of social interaction; this variant of constructivist learning is referred to as social constructivist learning (Felix, 2005). Other prominent theorists who are associated with social constructivist learning are Kuhn, Greeno, Lave, Simon and Brown. The social constructivist paradigm stresses the importance of social interaction and feedback in shaping the learning experience (Pear & Crone-Todd, 2002). The cognitive and social paradigms of constructivist learning theory are elaborated below.

2.2.1. The Cognitive Paradigm

Cognitive constructivist learning focusses on the intrapersonal process of individual knowledge construction. The theory is inspired by the belief that learning happens in the mind of the individual when the individual makes sense of the material. Hence, the focus is on individual construction of knowledge rather than co-construction of knowledge. This paradigm of learning theories stresses the cognitive aspects of the learning process. According to Piaget (1977), the learning process is an adjustment of ideas, as a result of interacting with the environment. This adjustment of ideas is driven by cognitive structure; students of the same age and same cultural background may have different cognitive structures. Piaget argued that cognitive development precedes learning. Piaget defined four distinct stages of cognitive development although the rate of cognitive development may vary amongst learners. Piaget's theory has been criticized by researchers who claim that the cognitive development cannot be divided into distinct stages because skills of learners develop in different ways. Applying Piaget's theory to learning programming needs the learner to actively interact with practical programming tools by engaging in activities designed by the teacher. On the other hand, the teacher needs to make a special effort to cater for different levels of cognitive development of learners. Therefore, the teacher needs to assess the level of cognitive development of students and design suitable activities.

Psychology studies have shown that students have personal beliefs regarding their own intelligence; these are referred to as self-theories. Many researchers have found that self-theories play a significant role in academic success (Dweck, 2000). According to psychologists, most people lean towards one of two theories: they either perceive their intelligence is static if they have a *fixed mindset*, or they perceive that their intelligence is malleable if they have a *growth mindset* (Katira, Williams, Wiebe, Miller, Balik, & Gehringer, 2004). Students with a fixed mindset are likely to exhibit a helpless response to challenges and experience a decrease in self-esteem when faced with such challenges. On the other hand, students with a growth mindset are likely to welcome challenges and maintain their self-esteem in the face of such challenges. Students with a malleable mindset believe that failure results from lack of effort rather than lack of intellectual ability. According to Yorke, and Knight (2004, p. 29-30), teachers should "appreciate the significance of self-theories for student learning; be able to infer whether students are inclined towards fixedness or malleability; and possess strategies for encouraging 'fixed' students to move towards malleability". Employing suitable pedagogies and practical programming tools are essential to facilitate the transition of students from a fixed mindset to a malleable mindset.

Cognitive constructivist learning approaches are popular for Maths and Science teaching. However, these approaches have not been very influential in computer programming. Van Gorp and Grisson (2001, p. 249-250) have identified activities for constructivist based programming teaching, as described below.
Code walkthroughs: In this activity, the computer program is provided to students. They need to examine the program and determine the outcome.
Code writing: In this activity, students are given small problems, and they need to write computer programs to solve the problems.

Code debugging: In this activity, students are provided with computer programs which have errors. Students need to find the errors; the errors may be logical or syntactic in nature.

Scaffolded code authoring: In this activity, students are supplied with partially completed computer programs; they are tasked to complete the computer programs by inserting appropriate code snippets.

The activities described above focus on active engagement of students but do not adequately address the social interaction element of constructivist learning. Hence, the activities can be categorized in the cognitive constructivist paradigm. The activities listed above are simple but significant to teach computer programming concepts (Van Gorp & Grisson, 2001). Constructivist classrooms are realized by implementing the activities listed above in rich development environments (described in Section 2.6).

2.2.2. The Social Paradigm

Russian psychologist Lev Vygotsky's (1978) social development theory examines learning as a social activity that takes place in a specific cultural environment. According to Vygotsky, "knowledge is constructed through active mental processes but is greatly influenced by one's social and cultural environment". In contrast to Piaget's theory, Vygotsky argued that social interaction, within the cultural context, is required for cognitive development to reach full potential. Vygotsky (1980) introduced the Zone of Proximal Development (ZPD). ZPD is the difference between what students are capable of achieving independently, and what they are capable of achieving with assistance from teachers and more capable peers. Tasks assigned to students can be broadly divided into three categories, namely, known tasks, attainable tasks and unattainable tasks. Vygotsky argued that learning is most effective when students are pushed slightly beyond their capabilities into ZPD, by accomplishing attainable tasks. Piaget (1977), who was a proponent of the cognitive theory of learning, also argued that optimal education required the learner to experience mildly challenging tasks although he stressed the importance of interaction with the environment. According to Dweck (2000), ZPD is most effective for students with a malleable view of their intelligence. Dweck (2000) suggests that students with a fixed mindset can be transformed to a growth mindset if the transformation is facilitated by the teacher. In order to achieve this transformation, the teacher should praise the efforts of students for attempting attainable tasks, and restrain from complimenting students for accomplishing known tasks. The concept of scaffolding is proposed by Wood, Bruner, and Ross (1976). Scaffolding is the support and guidance that teachers provide to students to complete tasks. It is a process teachers use to make sure their students can complete their tasks with progressively less support. ZPD and scaffolding complement the need for activation of prior knowledge to understand programming concepts. It is relevant in rich learning environments which afford learners the ability to build a community with their peers and teachers. The feedback, whether positive or negative, has a profound effect in shaping the learning experience.

Liu, and Tsai (2008, p. 634-637) identified the typical interactions that may exist between learners engaged in social constructivist learning. The patterns of interactions are classified as below.

Centralized knowledge exchange: In this type of interaction, there exists a learner who is the knowledge center; this person has a deep level of understanding, and actively responds to questions raised by peers. Other peers may also have a deeper level of understanding but they do not actively respond to questions raised.

Distributed knowledge exchange: In this type of interaction, each learner contributes to the interaction by raising questions and responding to questions.

Group development impediment: In this type of interaction, there is limited knowledge exchange. Knowledge exchange between peers is constrained by lack of interaction although most learners have a deep level of understanding.
Ability impediment: This type of interaction also has limited knowledge exchange. Knowledge exchange between peers is constrained by lack of interaction which is because the learners lack a deep level of understanding.
Partial knowledge exchange: This type of interaction fails to achieve the full potential of knowledge exchange. Few learners do not contribute to interactions. Interactions are restricted to few learners, and learners typically have significantly different levels of understanding.

According to Felix (2005), social constructivist learning promises the best results but requires the teacher to invest a significant amount of time to handle the inherent social and conceptual challenges. Combining the social and cognitive constructivist learning is a practical way to manage associated time constraints. A key aim of these learning theories is to enhance student engagement in learning activities. Therefore, in the next section, I discuss the findings of the National Survey of Student Engagement (NSSE). I used the findings of NSSE to inform the learning activities in this study. Teaching and learning approaches are addressed in Section 2.5 - these approaches draw upon learning theories discussed in this section.

2.3. Student Engagement Benchmarks

In this study, I investigate student engagement in learning activities. In order to gain insight into student engagement, I review the findings of the National Survey of Student Engagement (NSSE). NSSE is a widely accepted research initiative to measure student engagement (Robinson & Hullinger, 2008). NSSE measures the dimensions of student engagement on the basis of the *Seven Principles of Good Practice in Undergraduate Education* (Chickering & Gamson, 1987). These seven principles of good teaching are based on extensive research of the experiences of students and teachers. Below, I briefly address these seven principles.
Encourage contact between students and faculty: building rapport between students and teachers is important. Technology provides more opportunities for students and teachers to communicate.
Develop reciprocity and cooperation among students: Social aspects of learning is important to promote learning, as discussed in Section 2.2. Cooperation instead of competition achieves better learning outcomes.
Encourage active learning: according to this principle, students should take responsibility for their learning. This is challenging because of the background of passive learning of many students.
Give prompt feedback: feedback is a powerful tool in the learning process. Although there are different forms of feedback, a learner should be given meaningful information of their accomplishments, and areas which need improvement.
Emphasize time on task: the focus of this principle is that time should be spent effectively for learning activities
Communicate high expectations: according to this principle, high expectations from students will improve the performance of the students.
Respect diverse talents and ways of learning: different people learn in different ways. According to this principle, diversity should be embraced and fostered for improved learning outcomes.

The seven principles of good teaching are used as guidelines by NSSE. Although the NSSE was created for on-campus education, the principles on which it is based have been widely applied to the off-campus context as well (Robinson & Hullinger, 2008). NSSE is built on five benchmarks (Robinson & Hullinger, 2008, p. 103-105) which are addressed below. I draw relationships between the NSSE benchmarks and the learning environment used in this study.

Level of academic challenge: Level of academic challenge (LAC) refers to the academic effort made by students for activities such as studying, reading, writing, and preparing for class (Kuh, 2003). Development of mental capacities is an important component of this benchmark; it includes five levels of mental activities, namely, memorization, analysis, synthesis, making judgment, and application (Robinson & Hullinger, 2008).

Active and collaborative learning: Active and collaborative learning (ACL) refers to efforts of the students to contribute to class discussions, work with other students, and engage in other class activities (Kuh, 2003).

Student interaction with faculty members: Student–faculty interaction (SFI) relates to the nature and frequency of contact that students have with their faculty. Contact includes faculty feedback and discussion of grades and assignments, ideas, careers, and collaborative projects (Kuh, 2003). It is reported by Robinson, and Hullinger (2008) that the most common form of interaction is faculty feedback.

Enriching educational experience: An enriching educational experience (EEE) involves the development of the person to learn to work effectively with people from different backgrounds and enables the use of technology to facilitate collaboration (Kuh, 2003).

Supportive campus environment: Robinson, and Hullinger (2008) deemed supportive campus environment (SCE) criterion not relevant for the evaluation of student engagement in an online environment; however a virtual campus which is reflective of the physical campus reinforces the institutional spirit (Jennings and Collins, 2007) and the related support structure offered to students.

The learning environment used in this study is Second Life (SL). As an online technology, SL has the potential to meet the LAC, ACL and SFI benchmarks, recommended by NSSE. First, online technologies have been shown to positively impact student engagement benchmarks (Duderstadt, Atkins, & Houweling, 2002; Kuh, 2003). For instance, online technologies stimulate students to use higher order skills which provisions the capacity to meet the LAC benchmark. Second, online technologies are credited with ease of communication which improves their capabilities in the ACL and SFI benchmarks; having a supportive virtual campus facilitates the initiation of students especially if they lack an IT background.

As a virtual environment, SL has the potential to meet the SCE and EEE benchmarks, recommended by NSSE. First, the embodied experience in SL contributes significantly to achieving the EEE benchmark by improving interpersonal skills of inhabitants. Second, SL makes it feasible to build a virtual campus which is reflective of the physical campus; this contributes to the SCE benchmark.

As described above, SL has the potential to meet all the student engagement benchmarks. The advantages of SL compared with traditional online learning environments is the range of avenues it affords to address the student engagement benchmarks. These facilities of SL are addressed in Section 2.8.

2.4. Taxonomy of Learning Objectives

In Section 2.2, I discussed learning activities based on cognitive constructivist learning approaches. In the context of computer programming, the learning activities are categorized as code walkthroughs, code writing, code debugging and scaffolded code authoring (see Section 2.2). However, these categories of activities do not reflect the complexity of the activities. If novice programmers are assigned complex learning activities, they will be unable to complete the activities. Similarly, if advanced programmers are assigned simple learning activities, they will not achieve significant learning outcomes. Therefore, the complexity of the learning activities should be commensurate with the current level of cognition of the learner. Researchers have proposed taxonomies for classifying what students are expected to learn as a result of instruction.

Bloom's taxonomy is used as a model for categorizing aspects of learning into hierarchical levels (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956) based on complexity. If higher levels of learning are achieved by students, the instruction may be regarded as being successful. The taxonomy of learning, therefore, can be used as a holistic measure of the effectiveness of instruction. Bloom’s original taxonomy provides six hierarchical levels which are ordered from simple to complex and from concrete to abstract; each level builds on the level below. The six levels (Krathwohl, 2002, p. 213) are briefly described below.
Knowledge level refers to the ability to remember facts, concepts or principles;
Comprehension level refers to the ability to understand what students know and the ability to explain their understanding;
Application level refers to the ability to use their knowledge in a unique event;
Analysis level refers to the ability to break knowledge into components parts;
Synthesis level refers to the ability to create new knowledge;
Evaluation level is at the top of the hierarchy. This level refers to the ability to make judgments about something using some criteria.

Several weaknesses have been identified in Bloom’s taxonomy of cognitive development. First, the categories are semantically inconsistent - the knowledge category embodies a noun/noun-phrase and a verb/verb-phrase to reflect the subject matter and the cognitive development process while the other categories embody a verb/verb-phrase only (Krathwohl, 2002). Another weakness of Bloom’s taxonomy is that the hierarchical levels are frequently inverted. For example, in some scenarios achieving the *knowledge* level is more complex than achieving the *analysis* level.

A revised Bloom’s taxonomy was proposed to resolve this semantic anomaly. In the revised taxonomy, the noun and the verb aspects form two separate dimensions, namely, the knowledge dimension and the cognitive process dimension. The knowledge dimension includes the categories: factual, conceptual, procedural and metacognitive. The cognitive process dimension includes the categories: remember, understand, apply, analyze, evaluate and create. Bloom’s revised taxonomy (Anderson, Krathwohl, & Bloom, 2001) is widely used in curriculum design to strike a balance among lower levels of cognitive development (such as remember, understand, apply) and higher levels of cognitive development (such as analyze, evaluate and create) (Johnson & Fuller, 2006).

The premise of the study is that the participants are enrolled in an introductory programming course, and can be regarded as novice programmers (see Chapter 1). Therefore, learning activities should be designed to achieve more accessible levels of cognitive development, including remember, understand and apply, as discussed above. In the following two sections, I address the roles of pedagogies, and programming learning environments, in curriculum delivery with the general aim that students achieve the desired learning outcomes.

2.5. Pedagogical Approaches

Pedagogical approaches describe how learning theories are applied for teaching and learning. Hence, pedagogical approaches are grounded in principles of learning theories, discussed in Section 2.2. A range of pedagogies are available for educators to draw upon. No single approach is suitable for all circumstances and all contexts. Pedagogies provide different research lenses through which to interpret and gain insight into complex learning issues. Educators can use multiple pedagogical approaches to assist them to make sense of and respond to different circumstances and contexts. In this section, I have reviewed existing literature for pedagogical methods in general, and in computer programming courses. The pedagogies are divided into teacher-centered and learner-centered approaches, as described below.

2.5.1. The Teacher-Centered Approach

In traditional pedagogies, learning is the process of knowledge transfer from the teacher to the student. The teacher assumes responsibility for what is taught, how it is taught, and how learner performance is measured. In this model, the instruction frequently occurs with the whole class; instruction seldom occurs with individuals. The typical modes of instruction are lectures and textbooks. Listening to lectures and reading textbooks are the modes for knowledge transmission and rote-learning enhances knowledge that is retained. Lectures and textbooks are still the most effective ways of presenting a large amounts of material. According to this approach, rote-learning is essential to transform understanding into automated skill that is making the information available to the mind without conscious effort. However, rote-learning is weakest in providing motivation because of the lack of conceptual aids to learning. Further, low levels of engagement are linked to low levels of cognitive development, as discussed in Section 2.4. In this learning approach, instruction typically takes place in a classroom, and learning environments do not play a key role. This approach is widely regarded as reductionist because of the lack of cognitively powerful instructional strategies such as collaborative work and peer-assessment (Cuban, 1983; Relan & Gillani, 1997). Cuban (1983), found “a seemingly stubborn continuity in teacher-centered instruction despite intense reform efforts to move classroom practices toward instruction that was more learner centered”. As learning environments have evolved, criticism of this approach has escalated because researchers have questioned the ability to provide the learners with an authentic learning experience. A rich learning environment is deemed essential for intrinsically meaningful experiences of learning which are embedded in cognitive, social and cultural contexts (Relan & Gillani, 1997).

2.5.2. The Learner-Centered Approach

In learner-centered classrooms, students are placed at the center of the classroom organization to address their individual needs and learning styles. The transition from teaching the entire class to addressing the needs of individual learners requires extensive planning, and is tightly linked to teachers’ beliefs, attitudes and practices. This approach is driven by the goal to achieve deeper levels of understanding beyond skill and drill approaches. This model of learning has two main underlying assumptions: first, they challenge students to solve authentic problems; second, they assume that knowledge is best acquired in information rich settings. In this section, I discuss popular learner-centered approaches, namely, problem-based learning and experiential learning.

Problem-Based Learning

Problem-Based Learning (PBL) is based on a constructivist learning theory. It is an innovative learner-centric instructional method. In PBL, learning is focused on solving authentic problems, in other word, problems that professionals in the discipline would encounter. Authentic problems are distinctly different from well-defined exercises. The scope of exercises is limited; it involves practicing certain techniques to learn fragmented pieces of knowledge. Solving authentic problems aims to provide a holistic understanding of topics by driving the learning experience. PBL is grounded in the philosophy of John Dewey (1916) that learning is based on discovery and guided by mentoring rather than passive transmission of knowledge.

PBL was first developed in medical education in response to criticism that medical students were inadequately prepared to handle clinical situations (Hung, Jonassen, & Liu, 2008). PBL has become popular across a wide range of disciplines in higher education. Effectiveness of PBL has been reported in computing education (O'Grady, 2012). The outcomes of PBL match well with constructivist view of learning because it is characterized by the learner achieving a deeper level of understanding, the learner being motivated, the learner developing communication and meta-cognitive skills. As discussed in Section 2.2, the activities for teaching computer programming which are consistent with cognitive constructivist learning approach are: code walkthroughs, code writing, code debugging, and scaffolded code authoring. Further, collaboration is significant from the social constructivist learning perspective. According to Gallagher (1997, p. 338-340), the key components of PBL are as follows.

Initiating instruction with an ill-structured problem – Students are given an ill-structured problem; it is ambiguous and unclear. It can be solved in many ways and the solution can change as new information comes to light. An authentic problem or a real world problem, is established. The cognitive demands is the thinking required to solve an authentic problem is consistent with the cognitive demands of the discipline in which instruction is given.

Student as the stakeholder – In PBL, students take ownership of the problem and the process used to develop a solution. PBL breaks the traditional model of teaching and emphasizes student centered education instead of teacher centered education. PBL provides a form of apprenticeship for the students. According to Shin, Jonassen, and McGee (2003), while working with ill-structured problems, students are more likely to develop a significant body of content knowledge and make meaningful use of it when necessary; consciously regulate their own thinking, particularly when selecting and evaluating strategies and monitoring progress towards a solution; and develop defensible, evidence-driven arguments for the solutions they think are preferable under different circumstances.

Teacher as a coach – The role of the teacher in PBL is to challenge the thinking of the students and steer away from procedural approach to problem solving. The teacher uses meta-cognitive questions such as, "What is going on here" and "What further information is required", to familiarize students with the problem. Teachers prompt students to answer these questions and take ownership of the problem. These questions are also used to steer the students in the right direction.

When embedded into a PBL model, students learn to use rules of argumentation, experimentation, problem finding, and analysis (Gallagher, 1997; Gallagher, Stepien, & Rosenthal, 1992). As early as first grade, children students can learn peer tutoring and metacognitive reasoning through PBL (Shamir, Zion, & Spector-Levi, 2008).

Experiential Learning

Experiential learning (EL) is a PBL instructional approach. It is commonly defined as learning from life experience. The aim of experiential learning is to engage learners in direct experience (Wrzesien & Raya, 2010) or in-context action, with the aim to activate prior knowledge. EL is characterized with experience and reflection. Existing research shows that activation of prior knowledge contributes to the learning of new information (Schmidt, De Volder, De Grave, Moust, & Patel, 1989; Spires & Donley, 1998). Schmidt et al. (1989) found that by having prior knowledge activated, students were better able to come up with explanatory statements for the phenomenon being discussed. According to McCarthy, and McCarthy (2006) experiential activities are among the most powerful teaching and learning tools available. Experiential learning includes programs like internships, fieldwork and classroom experiential exercises.

Experiential learning as proposed by Kolb (1984) consists of a cycle of four stages: concrete experience, reflect on experience, abstract conceptualization and an active experience. The four stages reflect the four abilities that learners must have: they must be able to involve themselves in new experiences; they must be able to reflect on their experiences; they must be able to conceptualize their observations; and they must be able to make decisions and solve problems. Kolb uses the works of John Dewey (1916), Jean Piaget (1977) and Kurt Lewin (1939) to substantiate his proposed model.

The pedagogies described in this section can be used to inform the design of learning activities. However, adoption of a pedagogy is constrained by the learning environment. A learning environment may facilitate the adoption of some pedagogies to a greater extent than other pedagogies. In the next section, I discuss popular learning environments for computer programming.

2.6. Learning Environments for Computer Programming

According to Hwang, Wang, Hwang, Huang, and Huang (2008), programming learning environments play a vital role in students' cognitive development (see Section 2.2) in programming by providing methods to enrich the learning experiences. In traditional programming learning environments, such as classrooms, achieving higher levels of cognitive development in programming is constrained by the lack of practical tools to support learning activities. In Section 2.2, I discussed categories activities for constructivist based programming teaching, namely, code walkthroughs, code writing, code debugging, and scaffolded code authoring. These activities and peer-interaction can help students to achieve higher levels of cognitive development in computer programming (Fallows & Chandramohan, 2001). However, such activities need to be implemented within appropriate learning environments. In this section, I review learning environments which use learner-centric approaches for computer programming education. Most programming learning environments do not have the complexity of features found in fully-fledged programming environments used by professional programmers. Some of these environments are described below in the categories of animated programming environments, and visual programming environments.

2.6.1. Animated Environments

Animated programming environments guide students to learn the dynamics of program execution by animating the program execution. The advantages of animation in computer programming education have been extensively documented. Sykes (2007) conducted extensive qualitative and quantitative investigations to evaluate the effectiveness of such environments to learn programming and found that they are effective for novice programmers because of the immediacy and highly visual nature of the feedback. Examples of programming environments which support animation are TANGO which was developed by Stasko (1990), and ANIMAL which was developed by Röβling, and Freisleben (2002).

ALICE is a 3D visual environment developed by Cooper, Dann, and Pausch (2000) at Carnegie Mellon University. ALICE extends visualization to a 3D environment. ALICE is built on top of the programming language Python (http://www.python.org). In Alice, virtual objects populate a virtual world; the appearance and behavior of these objects can be manipulated by writing simple scripts. ALICE is used to teach problem-solving by the application of programming constructs.

Recent technological advances in virtual worlds such as Active World and Second Life have seen their emergence as learning environments. These environments support programmable in-world objects, and the behavior of these objects is used to provide visual feedback about program execution. The immersive nature of such environments bundled with programmability of artifacts and in-world communication channels has far reaching impact on personalized learning and learner autonomy. Such environments can provide meaningful experiences of learning which are embedded in cognitive and social contexts.

2.6.2. Visual Programming Environments

In this category, I discuss environments which are purpose-built for visualization of the program schema; the program schema is constituted of the building blocks of the program and the functionality afforded by each piece of the program. Tools in this category include those aimed for learning object-oriented paradigm of programming. Object-oriented programming is a complex activity and has been the focus for research of many computer science educators. Below, I discuss the tools which have visualization capability for the schema of object-oriented programs.

BlueJ is a pedagogical tool used to teach computer programming using Java programming language. The tool aims to address the misconceptions about object-oriented programming paradigm. The tool uses visualization of class structures to facilitate understanding of object-oriented programming concepts such as inheritance (Kölling, Quig, Patterson, & Rosenberg, 2003). The main advantage of visual programming environments is the abstraction of concepts (DeClue, 1996; Henrïquez, 2001; Stasko, Dominigue, Brown, & Price, 1998). A similar tool aimed at teaching Java programming language is DrJava.

The visual environment of BlueJ allows students to interact with existing classes without writing any code. This allows them to understand the concepts of object-oriented programming. Hagan and Markham (2000) used Problem-Based Learning to evaluate BlueJ. They used BlueJ to teach an introductory programming course at Monash University. They found that students perceived BlueJ to be helpful because of the visual nature of the environment. Hagan, and Markham

(2000) observed that using BlueJ alleviated confusion between concepts of classes and objects which was common in earlier years when BlueJ was not used in the course.

Other tools which provide class structure visualization are Rational Rose (IBM Rational Rose, 2003), and Borland's JBuilder (Romero, Cox, Du Boulay, & Lutz, 2003). However, these tools are purpose built for professional developers and lack the ease-of-use needed for introductory teaching. The drawback of BlueJ and other tools in this category is that they are tied to Java or another programming language.

RAPTOR (Carlisle, Wilson, Humphries, & Hadfield, 2005) is a visual programming environment for teaching object oriented programming. It uses a combination of Unified Modelling Language (UML) and flowcharts to generate programs. The programs generated within the RAPTOR environment can be executed visually within the environment.

In this section, I introduced a range of environments which are available for programming, and these environments can be used for learning activities. However, these environments do not support collaboration. According to the Zone of Proximal Development (ZPD) and scaffolding (see Section 2.2), collaboration is important to enhance student engagement in learning activities. In the next section, I introduce virtual world environments for implementing collaborative leaning activities in the context of computer programming.

2.7. Introduction of Second Life

In this section, I provide an introduction of Second Life (SL) to highlight its capabilities which are distinct from traditional programming learning environments (see Section 2.6). The capabilities of SL also support the achievement of student engagement benchmarks, described in Section 2.3. The notion of student engagement is based on the assumption that learning is influenced by how a student participates in educationally purposeful activities (Coates, 2005). Learning Management Systems (LMS) such as Blackboard, Desire to Learn and Moodle are used extensively for curriculum delivery in attempts to engage students. Grant and Clerehan (2011) note that LMS lacks embodied experiences. Contemporary educational theory emphasizes the need to have contextual embodied experience for student engagement (Migdalek, 2002).

There are several vendors which provide Multi-User Virtual Environment (MUVE) which afford a contextual embodied experience and have the potential to offer student engagement aligned with real-world experiences. A typical MUVE provides an immersive experience by enabling avatars, an avatar being the virtual representation of a human. It also contains virtual representations of real world objects. An avatar can view and interact with in-world objects. MUVE provides an interactive environment with capabilities for avatar-avatar and avatar-object interactions.

Second Life (secondlife.com) is a 3-dimensional (3D) virtual world environment. Another popular vendor of 3D virtual world is Active Worlds (www.activeworlds.com). River City Project and Active Worlds Educational Universe (AWEDU) are examples of projects for educational purposes, which are developed in Active Worlds. Second Life is the most mature of virtual world platforms which is reflected by its high usage figures compared with other competing platforms (Warburton, 2009; Dalgarno, 2010). According to Warburton (2009), SL combines the object-centric and egocentric aspects. Ultimately, what the residents of SL and other virtual worlds do so well is provide a reason (we can call them social objects) around which people can connect together and strive to continue those connections. SL is widely used in education (cf. Journal of Virtual Worlds Research, 2009). SL was initiated in medical and health education (Boulos, Hetherington, & Wheeler, 2007); it has been used for teaching languages, including Arabic (Kern, 2009) and Chinese (Henderson et al., 2010). It has also been used by researchers to collaborate (Novak, 2010). Nergiz Kern is an English teacher and teacher-trainer with extensive experience in SL. She observes that the immersive and collaborative nature of SL appealed to her (Kern, 2009). In the next section, I describe the building blocks of SL which are used to realize a virtual classroom, and indeed implement my empirical study.

2.8. Building Blocks in Second Life

Second Life (SL) is a 3-dimensional (3D) virtual world developed by Linden Lab in 1999. For SL to be effective, it should be able to accomplish the tasks performed in traditional learning environments. SL uses a client or viewer as the browsing software. The viewer needs to be installed in the user’s computer. The viewer receives data from the SL server and renders the 3D virtual environment. The viewer provides a graphical interface to use the functions of SL. The official viewer supplied by Linden Lab is the SL viewer. SL viewer code is released as open source which allows developers outside of Linden Lab to offer customized viewers, referred to as third-party viewers.

While providing features of traditional learning environments, SL provides opportunities for game-based learning and cooperative learning. According to Warburton (2009), opportunities exist for experiential learning and performative learning. As a means of illustrating the potential of SL to be used as a virtual classroom, and for implementing the empirical study, I develop an experimental land parcel – the land parcel being an analogy of real-world land, within SL. The land parcel provides space for in-world activities but it needs to be furnished with tools to engage students in learning activities. Building blocks of a virtual classroom in SL are described in the subsections below. I use these building blocks to realize a virtual classroom but these can also be used to develop land parcels in other contexts. Davis (2009) has proposed categories of tools in traditional learning environments. In the subsections below, I use these categories of tools to articulate the key affordances of SL, in the educational context.

2.8.1. Linden Scripting Language

Linden Scripting Language (LSL) is the proprietary scripting language of SL. Behavior and interactivity of in-world objects is controlled by LSL. Editing LSL is used for customization of the appearance/behavior of objects and enhancement of the functionality; resident-developed libraries can also be used for this purpose. LSL can be used to manifest real-world concepts in the virtual world. Hence, LSL can be used to acquire higher level computer programming skills such as problem analysis and conceptualization.

Esteves, Fonseca, Morgado, and Martins (2009) noted that a drawback of SL for novice programmers is the complexity of LSL. Although LSL is powerful, an inherent characteristic is the complexity of the syntax. Using a complex syntax to develop teaching activities for beginners is counter-intuitive. Third-party tools (tools not developed by Linden Labs) with a visual interface can be used to write simple LSL code. Using such third-party tools circumvents exposure to LSL, and is a suitable option for novice programmers. I discuss this further in the research design (see Section 3.3).

2.8.2. Avatars and Bots

In order to become a SL resident, one needs to create a virtual representation of him/her, namely, avatar. An avatar is referred to as the digital alter ego in the virtual world (Boulos, Hetherington, & Wheeler, 2007). An avatar is customizable; the clothes, hair, skin color, eyes can be selected to portray the in-world persona of an individual. A teacher needs to carefully decide the appearance of his/her avatar in order to make the students feel most comfortable. An avatar enters the SL virtual world by teleporting to a region using the SLurl (SL URL) which is used to identify each location uniquely. Avatars can also navigate by walking, flying and riding on virtual vehicles. For a specific location, an avatar may have the role of a visitor or land owner. Land owner has privileges to add, remove, and edit objects in their land parcel. The privileges of a visitor are granted by the land owner.

SL allows the creation of a bot account similarly to a human-controlled avatar account. A bot is script operated and allows user-interaction, role-playing and automation. Some typical uses of bots in SL are: to detect spammers/abusers; to invite residents to a group; and to greet residents (Linden Research Inc., 2011).

2.8.3. Acquiring Virtual Land

Virtual land is required to conduct educational activities such as lectures and discussions. Virtual land in SL is purchased directly from Linden Labs or on auction from other users who are selling their land. SL has an in-world unit of currency, namely, Linden dollars (L$) which is readily exchangeable to US dollars; it is used to purchase land and other goods/services. Land parcel available for purchase is either developed or undeveloped. Developed land has buildings and objects but costs more. Objects in developed land can be edited or removed by the owner. Undeveloped land is a vacant plot which needs to be developed from scratch according to personal taste of the owner; although cheaper, it requires substantial time and effort and the additional cost of purchasing objects (see Section 2.8.4). A land parcel can be designated a sandbox. A sandbox is a parcel of land which is available to SL developers for testing. A sandbox may be owned by Linden Lab or by a resident and is designated for public or private use. A sandbox is a land parcel which affords development privileges to users. Sandboxes are periodically cleared, removing any objects that users might have been left behind. 152 institutions are listed in the SL education directory (Linden Research Inc., 2013); these include accredited colleges, universities, and schools which own land in SL. The distribution by region, of institutions using SL, is shown in Figure 2.1.

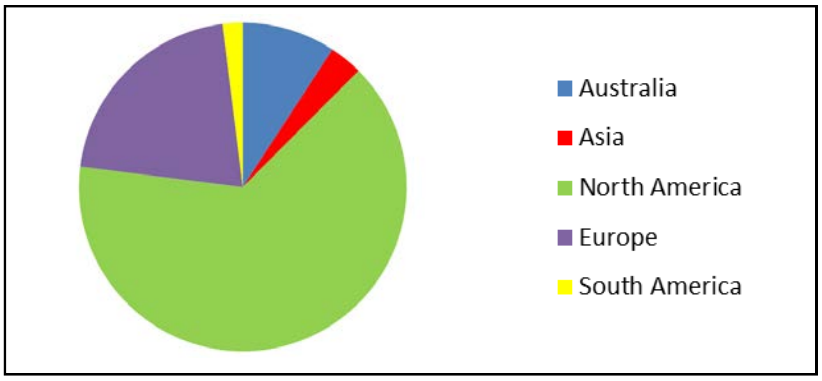


Figure 2.1. Distribution by Region of Institutions Using SL

Majority of the institutions with SL presence are located in USA; institutions in UK and Germany have majority SL presence in Europe; Singapore and Taiwan lead the tally in Asia. Common uses and characteristics of land occupied by institutions are addressed in the next section.

2.8.4. Inventory

Organizational tools such as calendar and files storage are key functional requirements of a learning platform. These tools are readily available on the internet; however, having an in-world presence makes them an integral part of SL – it prevents users' from having to access disparate systems.

SL Marketplace provides commonly used objects which can be purchased using Linden dollars (L$) which is the SL currency. For example, SL Marketplace returned 268 results for "calendar" search. Some are full-fledged calendar systems and some provide an in-world presence of calendar systems in the outside world, for example, an in-world presence of Google calendar. Similarly, SL Marketplace has many objects which support storage and archival systems; typical functions include: categorizing objects by their permissions; searching objects by applying filters on name, creator etc. Once an object is purchased, it will appear in the SL inventory. The user may modify the behavior of the object by editing the script (LSL) associated with the object.

2.8.5. Content Delivery

In this section, I describe a space in the acquired virtual land which is purpose-built for course delivery. This space is used to implement the activities for the empirical study. The space is described in this section assuming that the space has prefabricated buildings. Content can be delivered within the proposed space by using media streaming of content, displaying web content, and displaying slideshows.

Media Streaming and Web Content: A variety of media can be displayed on land parcels. This includes movies, audio, and web pages. The default media player object in the SL inventory (see Section 2.8.4) supports quicktime movies (ending in mov or mp4). Media from the local computer can be streamed into SL media player by using a streaming server; however, it is constrained by the upload bandwidth. Hence, it is best to use a streaming relay provider server to provide content to the in-world media player. The media can be streamed so that all residents view the same content by using the rtsp protocol; alternatively, streaming can be done on-demand; hence, each resident will view different content by setting the protocol to http (Linden Research Inc., 2014). In Figure 2.4, I furnished a virtual room with a media screen showing a quicktime movie. For streaming to work, quicktime needs to be installed in the local computer.

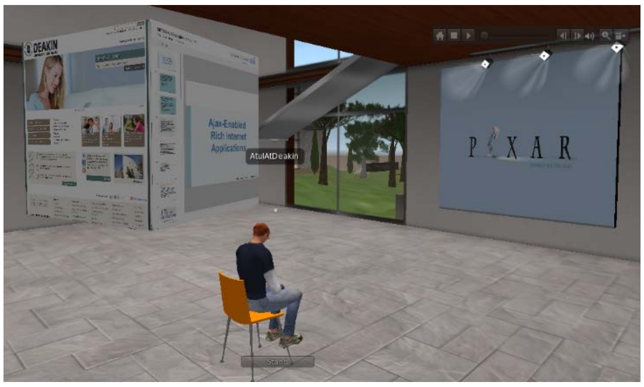


Figure 2.2. A Cube Prim with Different Media on Each Face and a Media Player Screen

Web content can be displayed on a prim. SL uses the Mozilla browser engine to display webpages. In Figure 2.2, a prim is shown – it is a cube with one face displaying a webpage.

Slide Presentation: There are several web sites which provide slide presentation hosting service, for example, Google Docs and Slideshare. Slide presentation from Google Docs is shown on the face of a prim in Figure 2.2.

An alternative approach for slide presentation is to develop it in-world. Each slide needs to be converted to an image file of a type which conforms to SL requirements. Each image is uploaded to the inventory at a cost of L$10. LSL is used to navigate through the images by advancing slides, going back and resetting to the first slide. Alternatively, several objects in SL Marketplace provide the functionality to create a slide presentation from a list of images (in the inventory). The slideshow is displayed in-world on any prim.

2.8.6. Collaboration

Collaboration is significant in problem-based learning which in turn is grounded in constructivist learning theory. Collaboration is required in virtual learning environments to accomplish group tasks. Collaboration is also significant because it stimulates learning, promotes feelings of belonging to a team, encourages creativity, eases communication and increases achieved personal satisfaction (Casamayor, Amandi, & Campo, 2009).

Collaboration is manifested in three main ways. First, avatars (described in Section 2.8.2) can communally or on-demand access course materials using appropriate in-world media objects to: listen and/or view streamed data. Second, avatars can interact with in-world objects. The in-world objects can be educational (for example, shared whiteboard) or social games requiring single player or multiple player participation. Third, avatars can voice/text chat with neighboring avatars (owned by student or teacher) or scripted agents (bots). Opportunities for in-world collaboration in SL are addressed below.

Text and Voice Chatting: SL provides text and voice (VoIP) chatting. Voice attenuates with increase in in-world distance from the speaker. SL provides the capability to log nearby chats and voice recordings. Chat logs and voice recordings can be maintained as described in (Linden Research Inc., 2008). Text and voice chat can happen between human-controlled avatars or scripted agents (bots). Grant and Clerehan (2011) use text chat in an assessment task for students of Chinese language: the assessment task requires student-operated avatars to engage in text chat with teacher-operated avatars and bots (automated non-playing characters). Kern (2009) notes that: *It* [SL text chat] *does display Arabic letters but they are not connected so that it is almost impossible to read.* A flash program in (Text Conversion) converts Arabic into SL compatible format. It is a functional but tedious work-around.

Shared whiteboard: A shared whiteboard is also tested for discussion purposes. While text and voice chatting are integral part of SL, shared whiteboard needs to be purchased with SL.

Groups: SL provides the capability of having groups. As in real-world, groups in SL are used for collaborative learning. Groups are implicit or explicit. Students can work implicitly in groups by going to a secluded space in SL to avoid interfering with other groups. Implicit groups do not use the group features of SL; such groups may be regarded as self-managed. SL allows the formation of explicit groups; these groups can have ownership of land and objects. The search tool in SL viewer or SL website is used to search groups; it returned 1717 groups (in June 2012) for the query term "education" and 2238 groups (in June 2012) for the query term "university". An inhabitant may join an existing group or create a new group.

> *Joining a group:* There are two kinds of groups - *open* groups which anyone can join; *closed* groups which requires an invitation to join. Some groups have a joining fee in L$ which is determined by the group owner.
>
> *Creating a group:* It costs L$100 to create a group. A group is required to have at least two members. The group owner decides whether a group is *open* or *closed* and if there is a joining fee. The owner can invite others to join a group and also determine the roles of other group members; for instance, a group owner can grant a group member the ability to invite others.

Lounge: Lounge is designed for avatars to unwind. It is a good place to socialize and chat about general interest topics. Virtual plasma screens can be realized in the lounge to watch movies. Plasma screens, chairs and other decorative objects need to be purchased from the marketplace. Other objects of interest in the lounge area are dartboards, pinball machine, RSS newsreader. Many creative objects are available from the Marketplace for a few L$ or even free.

2.8.7. Monitoring

Monitoring in SL is important to alleviate disruptions to teaching and learning activities. In this section, I list typical activities which warrant monitoring, and describe approaches to implement monitoring in SL.

Activities that require monitoring are listed below.

Avatars using obscenities in text/voice chat: it should be discouraged to maintain a conducive environment for the inhabitants;

Collusion amongst students to complete assessment tasks: students should be warned when such activity is detected or suspected;

i. Pirates in SL: these need to be detected to avoid intellectual property theft. Rogue programs called Copybots have the capability to copy in-world objects and marginalize the value of an owner's virtual land (Quarmby, 2008). Third-party viewers may be malicious or vulnerable to copybot attacks.
ii. Copyright infringement: copying course materials such as lecture notes, presentations and other resources is likely to infringe copyright laws.
iii. Monitoring is also required to identify cases of unsatisfactory progress such as not reading the course material, not participating in discussions, a group that has not finished an allocated task etc.
iv. SL provides tools which can be used as mitigating controls for activities listed above. These are discussed below.
v. Group moderation: SL group moderator has the capability to moderate a group's voice and text chat. The group moderator can exercise the option to disable text chat or mute voice chat of a group member.

SL provides the capability to log nearby chats and voice recordings. Chat logs and voice recordings can be maintained as shown in (Linden Research Inc., 2008). It is appropriate to use a bot account for these purposes in order to have an uninterrupted record of the data. Chat logs and voice recordings can be vital evidence for in-world misdemeanors including academic misconduct such as collusion. The logs/recordings can be automatically/semi-automatically analyzed to resolve disputes or take disciplinary actions.

i. Copyright infringement can be enforced by SL or owners of intellectual property.

   Enforcement by SL: An owner of intellectual property in SL can report copyright infringement to SL administrator which can lead to the cancellation of the infringer's account under Terms of Service (Second Life Terms of Service). Quarmby (Quarmby, 2008) notes that mere cancellation of account may not be a strong disincentive to repeat the behaviour because the infringer can enter SL with another account.

   Enforcement by owner: An owner may enforce copyright infringement by initiating legal proceedings. However, laws to punish in-world transgressions lack precedence and worldwide standardisation.
ii. Agents can be used for in-world monitoring. Log files are used to build students' profile which includes attributes such as learning style (reflective, sequential etc.), collaboration profile, and progress made in the course (Casamayor, Amandi, & Campo, 2009). Agents use Web Usage Mining (WUM) techniques (Srivastava, Cooley, Deshpande, & Tan, 2000) to draw inferences from student profiles and the teacher is alerted to unusual patterns. Examples of unusual patterns are: strongly sequential student does not perform tasks according to the order specified in the task plan; reflective student finishes the reading very quickly etc. (Casamayor et al., 2009). Once alerted, the teacher has the opportunity to intervene preemptively.

In the next section, I describe the pedagogical potential of SL realized through the building blocks described in this section, and give examples of SL projects which have been implemented in higher education institutions to facilitate engagement of students.

2.9. Pedagogical Applications of Second Life

Effective pedagogies for computer programming involve learner-centric approaches, namely, problem-based learning and experiential learning which are inspired by constructivist learning theory (see Section 2.5). Here, it is relevant to address the pedagogical underpinnings of Second Life (SL). A constructivist approach to learning focuses on the process of creating and sharing artifacts in SL (Girvan, Tangney, & Savage, 2013). This can be realized in SL because it offers the affordance to build objects (Dickey, 2003). SL is suitable for social constructivist approach (see Section 2.5) because it offers communication tools in an immersive environment (see Section 2.8). These tools may be used for collective learning of a group wherein individual learning is also a by-product of collective learning.

In this section, I summarize the learning benefits of SL which have been realized by higher education institutions. In general, these learning benefits of SL are achieved by implementing principles of constructivist learning to enhance learner engagement. Additionally, some of these implementations provide means for social interaction to facilitate knowledge exchange between peers. Few examples from SL wiki (Linden Research Inc., 2013) are included below.

At *Ohio University, USA*, an educational game is used in SL to learn about the impact of fast food on health. The game allows players to experiment with different eating styles at simulated fast-food restaurants. The game will assess the short-term and long-term health impact of the choices made by the players. Players achieve a high score if they make healthy choices.

At *San José State University, USA*, an educational virtual world is used in SL for cardiac auscultation training. The virtual clinic allows students to test their skills at identifying different types of heart murmurs based on sound.

At *Texas Wesleyan University in Fort Worth, USA*, a virtual lab, namely Gene Pool, was created in SL to learn about genetics; it comprises simulated experiments, tutorials and videos.

At the *University of California, Davis, USA*, a virtual world is used in SL to educate people about schizophrenic hallucinations.

HealthInfo Island is used to provide health information services to SL residents. This project comprises of three virtual buildings, namely, a medical library, a consumer health library, and a patient advocacy center. It provides meeting spaces groups and hosts collaborative events and activities.

At the *University of Plymouth, Devon, UK*, Virtual Neurological Education Centre (VNEC) is used in SL for participants to learn about the symptoms of neurological diseases. In VNEC, users are able to select a range of neurological symptoms (motor, sensory, and balance) which are animated by their avatar.

At the *University of Kansas, USA*, a virtual long-term care (LTC) facility is used in SL by the School of Nursing; in the virtual LTC, students can interact with staff who role-play as nurse.

At the *University of Wisconsin, Oshkosh College of Nursing, USA*, a virtual learning center is used in SL to learn about public health (PH). Students participate in PH exercises by learning the role of a sanitarian, and inspecting virtual bars and restaurants.

There are not many instances of the use of SL for teaching computer programming. Educational communities are growing rapidly in SL, as reported above. Researchers have reported enhanced learning experiences in SL. The examples given above use the affordances of SL for teaching. Some facilities in the 3D virtual environment are available in other collaborative environments; these include visualization and text/voice chats. The uniqueness of the learning experience in 3D virtual worlds is attributed to role-playing manifested by avatars, animation of objects and social interaction in an immersive environment.

Given the significant role that activation of prior knowledge plays in learning, my study proposes to activate students' prior knowledge of computer programming by tasking them to build artifacts. The reason for choosing SL as the programming learning environment is two-fold: first, the collaborative nature of SL inherently supports the principles of ZPD and scaffolding; second, the realization of artifacts, albeit virtual, brings physical meaning to abstract programming concepts. SL provides the facilities to create collaborative problem-based scenarios thus justifying the adoption of problem-based learning. In PBL, students take ownership of the problem. According to Esteves et al. (2009), a visual project or a project which animates the program execution is deemed suitable in SL because it will engage and motivate students. On the other hand, a non-visual project in SL will not leverage the affordances of SL; it will fail to motivate students because of lack of visual feedback from program execution.

Experiential learning is a PBL instructional approach. As discussed in Section 2.5, it has a cycle of four stages, namely, concrete experience, reflect on experience, abstract conceptualization and an active experience (Kolb, 1984). Girvan, Tangney, and Savage (2013) have mapped stages of experiential learning to the stages in creating programmable artifacts in SL. The activity of creating programmable artifacts requires programmable tools. The proprietary programming tool in SL is not suitable for novice programmers owing to complexity of use and associated steep learning curve. Add-on thirty party tools are deemed more suitable for learning programming. Girvan et al. (2013) used the third-party tool called SLurtle (programmable turtles in SL) as the programming tool for creating artifacts; the mapping can be generalized for SL third-party programming tools as shown in Figure 2.3.

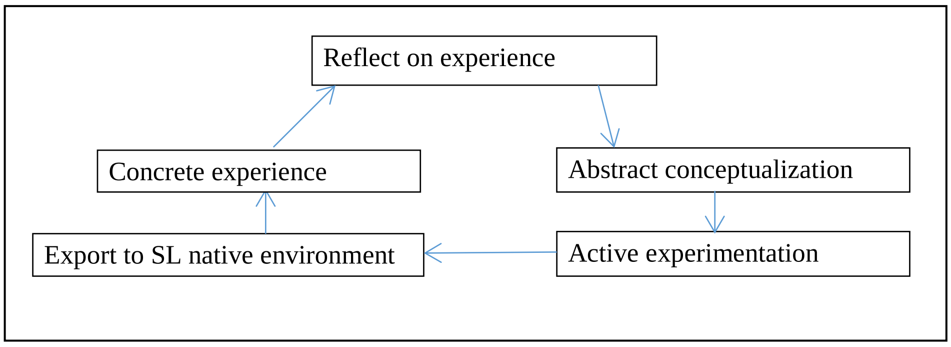


Figure 2.3. Stages of Experiential Learning and Stages to Create Programmable Artifacts

Adapted from "SLurtles: Supporting constructionist learning in Second Life", Girvan, C., Tangney, B., Savage, T (2013), Computers and Education, Vol. 61, 115–132.

According to Girvan et al. (2013), the experiential learning cycle of Figure 2.3 provides learners with concrete experience which they can observe and reflect upon. The learners can reassess the code by comparing the animation at the concrete experience stage with the original plan. The current approaches in computer programming education guides the research methodology of this study which is discussed in this chapter.

In this section, I addressed the affordances of SL for education. I adopt SL for the learning activities (described in the next chapter) because SL is a promising environment for enhancing student engagement, and achieving the desired learning outcome which is to understand programming concepts. Based on literature review, it is observed that SL provides a fun and interesting approach for educators and students alike. The concern of technology being overwhelming is alleviated by the fact that a large number of potential users, namely, university students, are digital natives and familiar with virtual worlds in role-playing games such as World of Warcraft.

2.10. Interface Design

Thus far, I have reviewed literature relevant to answering the main research question: *To what extent do IT students find SL helpful in understanding programming concepts?* In this section, I review literature relevant to the second research question: *How usable do participants find the SL environment?* As discussed in Section 2.2, learning is a psychological phenomenon which changes behavior. Below, I address conceptual frameworks applied to designing computing products which affect behavior. These conceptual frameworks are applied to the design of websites, mobile apps and indeed learning environments, to influence people in various ways. Further, I address evaluation of usability which is grounded in the theories of interface design.

2.10.1. Interface Design Theories

In this section, I address the conceptual elements of interface design in the context of affordances of learning environments. These approaches are focused on principles of human-computer interaction (HCI) and the computing tools are referred to as persuasive technologies because like human persuaders, these technologies employ persuasion which is defined as a “non-coercive attempt to change attitudes or behaviors” (Fogg, Cuellar, & Danielson, 2009, p.110). According to Oinas-Kukkonen (2013), persuasive technologies aim to persuade humans by leveraging the design of software. According to Fogg (2003a), persuasive technologies not only have advantages over traditional media but also humans. Fogg noted the advantages of persuasive technologies over humans are: they are more persistent than humans, offer greater anonymity, manage huge volumes of data, use many modalities for interaction, scale easily, and handle sensitive issues. Fogg identified learning as one of the domains in which persuasive technologies have the potential to play a role. With the emphasis on behavior, the theoretical constructs in designing persuasive computing tools are inherently adopted from the field of psychology.

Persuasive System Design (PSD) proposed by Oinas-Kukkonen and Harjumaa (2008) is a widely accepted model to incorporate persuasiveness into the user interface. The PSD model builds on theoretical constructs in psychology, such as The Theory of Reasoned Action (TRA) proposed by Fishbein and Ajzen (1975). TRA explains the motivational factors which determine behaviors. TRA is widely used to influence the design of persuasive computing tools. The PSD model’s persuasion context consists of features which originate from Fogg’s work in persuasive technology (Fogg, 2003b). These software features are divided into four categories, namely, support for the primary task, computer-human dialog, perceived system credibility, and social influence. Here I address each category of software features in the PSD model. First, software features which provide support for the primary task are features which help the user to complete the main activities; examples of software features in this category are verbal praise, timely suggestions to the user, and virtual rewards. Second, software features relating to computer-human dialog aim at ease of completing the activities; an example of this software feature is a wizard which guides the user through an activity. Third, the user perception of system credibility can be enhanced by software features such as exemplars of completed activities, and endorsements. Fourth, software features relating to social influence motivates the user by leveraging social interaction; examples in this category are audio and text chat.

According to the PSD model, some or all of these features may be implemented by a persuasive technology. PSD model is widely used in the design of systems for lifestyle intervention - such systems are referred to as Behavior Change Support Systems (BCSS). The desired outcome of a typical BCSS is: healthy eating, quit smoking, or exercising regularly. PSD has not been used widely for education although it can potentially be used for education. Second Life (SL) provides a learning environment which can potentially deliver software features in each of the categories identified in the PSD model. SL, therefore, is deemed suitable from the perspective of its design. In the next section, I address the evaluation of software in the context of its design.

2.10.2. Evaluation of Usability

Above, I introduced theoretical constructs in human behavior change, which are used to design persuasive computing tools. However, it is challenging to measure change in behavior. Hence, evaluation of persuasive computing tools is difficult, and in fact an open research problem (Oinas-Kukkonen, 2013). In this section, I address the evaluation of interface design based on user perception about the learning value of interface design. Technology Acceptance Model (TAM) proposed by Davis (1989) is widely used for evaluation of interface design. Theoretical grounding of TAM is the theory of reasoned action (TRA) proposed by Fishbein and Ajzen (1975). TRA (discussed in Section 2.10.1) is adapted by TAM to explain the acceptance of technology. The premises 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 premises, Davis (1989) proposed two constructs, namely, perceived usefulness (PU) and perceived ease-of-use (PEOU) as indicators of intent to use a technology, namely, technology acceptance (TA); he used an experiment with email and graphics to validate TAM. TAM constructs have been validated in a number of other studies, for example, Matheison (1991), Adams, Nelson, and Todd (1992), and Chau (1996). TAM has been expanded in TAM2 and TAM3 (Venkatesh & Davis, 2000; Kim & Park, 2012) to evaluate usability of new albeit specific technologies. I use TAM to evaluate the usability of Second Life (SL). The evaluation of SL, based on TAM constructs, is used to inform the data collection in the context of the second research question.

2.11. Conclusion

In this chapter, I have reviewed literature in order to gain an understanding of constructivist learning theory and pedagogical approaches in relation to computer programming. I have addressed the significance of constructivist learning and discussed pedagogical approaches based on principles of constructivist learning, namely, PBL and experiential learning. In PBL, learners build on their existing knowledge by applying prior knowledge to solve real-world problems (Hadjerrouit, 2008). Experiential learning focuses on the process of creating and sharing artifacts which are personally meaningful (Girvan, Tangney, & Savage, 2013). I provided an overview of learning paradigms to achieve a higher level of cognitive development in computer programming students. Research recommends constructivist learning as an optimal pedagogy for computer programming education. Literature argues that learner-centric approaches are effective to achieve a higher level of cognitive development for novice programmers. In computer programming education, instructional approaches are realized using instructional environments, namely, programming environments. The literature review reflects that programming environments which have capabilities of animation of program execution and visualization of program schema, are effective for programming instruction. I have addressed the alignment between affordances of SL and constructivist pedagogies. In line with improved pedagogical approaches in computer programming, extant literature suggests that SL provides a constructivist environment for building skills and knowledge. In this study, I propose to explore how SL might support the learning of programming concepts.

In the context of the design of SL, I reviewed conceptual frameworks applied to designing computing products to influence attitudes and behavior; I focused on the Persuasive System Design model which is widely accepted. I identified Technology Acceptance Model as a suitable approach to evaluate the usability of SL, and seek responses to the second research question about the usability of SL.

In the next chapter, I discuss the research methodology and research design of my study. I will design activities within SL environment for the empirical study, using principles of constructivist learning. These activities will be the sources of evidence for data collection. In Chapter 4, I will analyze and interpret the data to seek answers for the research questions.

Chapter 3

Research Design

3.1. Introduction

In this chapter, I investigate suitable research methodologies for my proposed study. Further, I elaborate the research design by describing the participants, the setting, and methods for data-collection and analysis. I explain the design of activities used for data gathering.

3.2. Research Methodology

Research methodology is defined as “a theory of how the inquiry should proceed” (Schwandt, 2007, p. 193). Sandelowski (2000) believes that classical or standard methodologies should be used to guide the study, hence, researchers should be encouraged to use ideas from these methodologies. Similarly, Patton (2002, p. 68) recommends adopting the stance of “methodological tolerance”. On the other hand, according to Thorne (2008), a single methodology should be adopted to avoid dilution of standards for research practice.

Action Research (AR) methodology is deemed suitable to determine and evaluate innovative technology adoption for education (Warden, Stanworth, Ren, & Warden, 2013). The essence of AR is encapsulated within its name - it represents action and research, or in other words practice and theory (McKay & Marshall, 2001). AR is an interventionist approach. In AR, the educator works collaboratively with the students to realize change in the context of the problem by using a set of structured activities, as discussed in problem-based learning and experiential learning in Chapter 2. AR is a cyclical process in which continual feedback is used by the researcher, to modify the activities. The emphasis of AR is to successively enhance instructional design in each iteration, based on evaluation of prior cycles. Warden, Stanworth, Ren, and Warden (2013) employed AR to investigate synchronous learning in distributed environments (SLIDE). In their study, Warden et al. (2013) progressed through the iterations by evaluating results in each iteration and identifying key learning points; accordingly, techniques were slightly modified and applied in the next iteration. Through AR, limitations related to hardware, software and human factors were identified in each iteration of their study. They noted that tools like WebCT and Blackboard are limited by their emphasis on asynchronous learning. On the other hand, virtual environments have the potential to evolve social interactions into interactions for synchronous learning. The significance of virtual environments is realized by noting that gaming in virtual environments is more popular than watching movies in cinemas; failing to use virtual environments for education will risk marginalization. Virtual environments like SL have rich possibilities for implementing SLIDE. Despite the perceived suitability of AR, I discounted this methodology because of time constraints. AR requires the study to be implemented over an extended period, to accommodate multiple iterations of evaluation. I consider AR to be a suitable approach for future work, aimed at corroborating and extending the findings of this study (see Section 5.4).

The small-scale nature of the study did not warrant the use of AR methodology. Instead, I selected case study research methodology because it is suitable for an exploratory study (Yin, 2011). Case study research is widely used by researchers in real-life issues and problems. A case study is defined by Yin (1984, p. 23) "as an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used". A case study requires that the *case* or the unit of analysis is defined; a case study entails assessment of the case. A case may involve multiple sources of evidence to gather data; the collected data are qualitative and/or quantitative. In my study, I define the case as the cohort of participants undertaking the same activities in SL, with the aim to achieve an understanding of abstract programming concepts. The participants are students enrolled in introductory computer programming courses at a higher education institution. I design empirical activities to be undertaken by the participants, which will potentially drive the desired change, namely, better understanding of programming concepts.

Owing to the inherent limitations of time and resources, I conducted a pilot study to evaluate Second Life (SL) in the context of the research questions. A pilot study is a small-scale study and typically used as a preliminary study before implementing a full-scale study. Due to the nature of a pilot study, the findings cannot be used to make firm conclusions. The findings of the pilot study are used to draw tentative conclusions which can potentially be generalized by conducting an extensive study in the future.

Reliability of research is an important issue. Research is characterized as reliable if other researchers are able to obtain similar findings when the operations and procedures of the research inquiry are repeated. The challenge is that data from different researchers might not converge because of the nature of empirical activities in which data are collected from real-life events. Techniques are applied to establish the reliability of the research by ensuring the soundness of the data gathered. Reliability is increased by establishing meaningful parallelism across multiple data sources. The data sources are discussed in the next section.

In the next section, I discuss the design of the empirical study. I will elaborate on the nature of the learning activities which will be used for data collection. I describe the multiple data sources in order to address the reliability of the research. Further, I will address the analysis and interpretation of the data, to shed light on the research questions.

3.3. Research Design

In this section, I discuss the design of the proposed study to investigate the role of SL, in the context of teaching programming concepts to novice programmers. I adopt a case study research methodology (see Section 3.2). I address the research design by describing the participants, the SL facilities, and the sources of evidence for data-collection and analysis.

3.3.1. Participants

The invitation to participate was distributed by email to IT students enrolled in an introductory programming course at an Australian university. The total number of invitees was around 100; and with an anticipated response rate 20%, I expected 20 invitees to respond to my invitation. Background of each respondent was established based on 5 criteria about gender, current course, and IT experience (see Table 4.1). The background was used to assign each participant to an appropriate group, as discussed in Section 4.2. After being assigned a group, each participant was expected to spend 45 minutes to participate in the study, as detailed in the following table:

Table 3.1. Activities for Participants

Activity and Allocated Time	**Activity Description**
Gain familiarity of SL through instructions provided by me 15 minutes	I instructed the participants in order to prepare them to undertake tasks in SL. Instructions comprised a video tutorial and a demonstration. - Each participant watched a video tutorial about Second Life. The duration of the video was about 7 minutes. The video focused on navigation within SL, and voice/text chat to interact with other virtual beings, namely, avatars. - I gave a demonstration to each participant about editing and testing code within SL. The duration of the demonstration was about 7 minutes. During the demonstration, I encouraged the participants to use voice/text chat for the purposes of collaboration (with other participants and/or with me) while subsequently undertaking tasks in SL.
Undertake tasks in SL 20 minutes	At the time of the study, the participants were studying a computer programming course at university, and I assumed that their course led the participants to have exposure, albeit limited, to programming concepts of variables, conditional statements and iterations. Participants were expected to use their knowledge of computer programming concepts to undertake tasks in SL that were designed by me. Small groups of four participants' attempted the tasks synchronously to allow for collaboration. As discussed above, each participant was assigned to a group, based on the background of the participant, which was established beforehand. Handouts (Appendix I) were distributed to guide the participants through the activities.
Complete a questionnaire 10 minutes	A questionnaire (Appendix II) was used for data collection to seek responses for the two research questions. After undertaking the tasks, participants completed a questionnaire.

3.3.2. Second Life Practices

After gaining familiarity with SL through instructions provided by me, the participants undertook tasks in SL (as shown in Table 3.1). Below, I describe the programming tool for undertaking SL tasks; the collaboration facility in SL which is available to participants; and an overview of the tasks.

Programming Tool in SL: A programming tool in SL is needed to undertake the tasks. Linden Scripting Language (LSL) is the native (embedded) programming tool in SL as discussed in Section 2.8. However, LSL involves a steep learning curve for novice computer programmers (see Section 2.8). Barriers to engagement are lowered by providing a low-floor tool, in other words, a tool which has a moderate learning curve. Scratch for Second Life (S4SL) was designed by Rosenbaum (2008) as a low-floor and high-ceiling (powerfully expressive) programming environment 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 a virtual object in SL to add behavior and interactivity to otherwise static objects (Pellas & Peroutseas, 2016). I used S4SL to develop tasks for the proposed study. The code generated for each activity using S4SL is discussed in Section 3.4.

Collaboration Facility in SL: While undertaking the tasks, the SL collaboration facility was available to participants, thus enabling, participants to interact with others in real-time using voice and/or text chat (see Chapter 2). SL eliminates the need to post messages in another environment (and await a response) thereby promoting continuity. An avatar can listen-to/view messages posted by another avatar; however, avatars need to be within range to be able to communicate. Range is a pre-set virtual threshold distance; avatars further apart than the range are unable to communicate. Avatars can communicate by voice, subject to the avatar posting the message being within range of the avatar that is listening. Similarly, avatars within range can communicate by using text messages which appear in a pop-up element. Participants may participate actively by posting and/or responding-to messages; the messages may be text or voice. Alternatively, participants may participate passively by viewing/listening to messages posted by avatars within range.

The SL Tasks: Participants undertook tasks in small groups of four; synchronous participation allowed collaboration. The composition of each group is important because participants may need to seek the assistance of other members of their group to complete the tasks. Each group engaged in the tasks, at a time which was mutually convenient to me and each group member. I assigned the same three main activities to each participant within each group. Hence, each participant was tasked with the same activities, comprising one activity each for the programming concepts of variables, conditional statements and iterations. Each activity was estimated to take 7 minutes. For the activities, SL objects were provided by me in the SL inventory, and a handout with short questions (see Appendix I) was given to each participant. Each SL object had embedded code which was generated in S4SL (as discussed above). The participants needed to study the code embedded within the SL objects, and answer the questions in the handout. Some activities required the code to be edited by the participants. Each activity is elaborated in Section 3.4.

3.3.3. Mixed-Method Approach for Data Gathering

In this section, I describe the sources of evidence used for gathering data. The data gathered were quantitative and qualitative; this is defined as a mixed-method approach (Creswell, 2003). A Mixed-method approach enables the researcher to draw on the strengths of both a quantitative approach and a qualitative approach.

Qualitative research is exploratory research. Qualitative research is defined as “a type of educational research in which the researcher relies on the views of participants; asks broad, general questions; collects data consisting largely of words (or text) from participants; describes and analyses these words for themes; and conducts the inquiry in a subjective, biased manner" (Creswell, 2008, p. 46). This theory maintains that the researcher's subjectivity is central to the research findings. Hence, the researcher's experience and knowledge will influence the findings. It is used to gain an understanding of underlying reasons, opinions and motivations (Wyse, 2011). It is used to uncover trends in thoughts and opinions and gain a deep understanding of the problem. A Qualitative approach is defined as a situated activity that locates the observer in the world, and it consists of a set of interpretive, material practices that make the world visible (Denzin & Lincoln, 2005). In qualitative research the researcher subjectively interprets the data to discover the findings. The strength of this approach is its ability to emphasize the viewpoint of the researcher. Hara (1995) states that the qualitative research approach in education is able to encompass interpersonal, social, and cultural contexts of education more fully than the quantitative research approach. This approach is able to account for psychological dimensions of humans which are impossible to represent in quantitative data. Education research deals with complex issues which can be effectively addressed by qualitative research because of its potential to offer rich and wide-ranging descriptions.

On the other hand, quantitative research is "a type of educational research in which the researcher decides what to study; asks specific, narrow questions; collects quantifiable data from participants; analyses these numbers using statistics; and conducts the inquiry in an unbiased, objective manner" (Creswell, 2008, p.46). A Quantitative approach pursues facts based on statistical evidence. In this approach, the researcher is viewed as an objective observer, and the research results are independent of the researcher's personal viewpoint. In summary, this approach emphasizes the discovery of existing facts by using neutral scientific knowledge (Hara, 1995). Below, I describe the three sources of evidence, namely, a questionnaire, observation of participants, and SL chat logs used for gathering data.

3.3.3.1. Questionnaire

Questionnaires are widely used in education research for gathering quantitative and qualitative data. Questionnaires allow the researcher to use identical questions to obtain responses from participants (O'Toole & Beckett, 2010). After undertaking the SL tasks, data are collected from the participants as shown in Table 3.1. A questionnaire (see Appendix II) was used as one source of evidence for data collection to elicit views of participants, relevant to the research questions. The questionnaire has the following parts:

- Parts 1 and 2 of the questionnaire has 11 Likert-scale questions related to usability of SL. The questions were adopted with adaption from the Technology Acceptance Model (TAM) proposed by Davis (1989). TAM is widely used for evaluating acceptance of end-user computing technologies, as discussed in Section 2.10.
- Part 3 of the questionnaire has 4 open-ended questions about the participants' experience in undertaking the SL tasks. The data was used to answer the first research question: *To what extent do IT students find SL helpful in understanding programming concepts?*

3.3.3.2. Observation of Participants

Observation is one of the preferred tools for qualitative research to collect data (Mackenzie & Knipe, 2006; Merriam, 1998; Yin, 2009). Nicholls, Mills, and Kotecha (2014) stated that observation might be adopted "alongside other data-collection methods to focus on different aspects of the issues being researched or to support or extend the other evidence collected" (p. 251). I collected qualitative data about the participants, by observing them within SL, while they undertake tasks in SL. Participants synchronously undertook tasks in SL, in small groups of up to four at a time. In order to collect data, my avatar joined each group in SL. While each group undertook the tasks, I took notes about the interactions in SL, and the extent to which the objectives of the tasks are achieved by each participant. In my observations, I articulated two distinct categories of interactions within SL. First, interactions of the participants with the SL controls, including accessing objects from the SL inventory, and using SL to initiate dialog with other participants. Second, I took notes of text and voice communications between participants.

3.3.3.3. Second Life Chat Logs

I configured SL to log (text and voice) chat data which was generated from collaboration amongst participants while they undertook the tasks. The chat logs constituted the third source of evidence. These were used to verify the data gathered from observation.

3.3.4. Data Analysis

By performing data analysis, I addressed the ultimate research purposes, namely, personal curiosity considerations, and achieving practical and intellectual outcomes, as discussed in Section 1.2. Data Analysis is defined as “close engagement with one’s data, and the illumination of their meaning and significance through insightful and technically sophisticated work” (Antaki, Billig, Edwards, & Potter, 2003, p. 30). To seek answers for the research questions, I reflected on and analyzed data from the three sources of evidence, namely, the questionnaire, the chat logs and the observation data, as described in Section 3.3.3. I used qualitative and quantitative approaches to analyze data.

I analyzed qualitative data to seek an answer for the first research question: *To what extent do IT students find SL helpful in understanding programming concepts?* Qualitative data were obtained from open-ended questions in the questionnaire, the chat logs and the observation notes. The data from these three sources varied vastly in the degree of structure. Responses to open-ended questions in questionnaires typically lacked elaboration making them challenging to interpret. Observation notes were recorded by me - I attempted to make these as descriptive as possible. In my capacity of the researcher, I needed to be objective in my description of the observation, and moderate the personal reflection during the interpretive process. The third source of data, namely, chat logs, captured spoken words and text messages exchanged between participants while they engaged in activities. The main purpose of the chat logs is for triangulation.

In order to answer the main research question, I identified patterns in each source of data. According to Stake (2010), the activity of analyzing qualitative data is an extension of the kind of analysis we do in everyday life. I brought my critical thinking skills, which have been developed as an educator in computer programming, to interpret the data. I used my analytical skills to understand the experience of the participants. While analysis is focused on the participants, it is desirable to generalize the findings from qualitative research (Bazeley, 2013). Generalizing the findings addresses the practical element of research purposes, as discussed in Section 1.2.

Quantitative data collected from Likert-scale questions in Parts 2 and 3 of the questionnaire were analyzed to inform the second research question: *How usable do participants find the SL environment?* I used Technology Acceptance Model constructs of perceived usefulness (PU) and perceived ease-of-use (PEOU) to analyze the data (Davis, 1989), as described below.

Technology Acceptance Model: Technology Acceptance Model (TAM) proposed by Davis (1989) is widely used for evaluating acceptance of end-user computing technologies (see Section 2.10.2). TAM constructs of perceived usefulness (PU) and perceived ease-of-use (PEOU) are used to explain technology acceptance. PU and PEOU are comprised of items shown in Table 3.2. I adapted PU and PEOU to evaluate SL as a learning environment for computer programming concepts. The relevant questions (see Appendix II) are adopted from TAM constructs of Table 3.2; however, the item on *Job Performance* is not included because it is not deemed relevant for the study.

Table 3.2. 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

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.

$$\propto = \frac{N.\bar{c}}{\bar{v} + (N-1).\bar{c}} \qquad (1)$$

Here N is equal to the number of items, $\overline{C}$ is the average inter-item covariance among the items and $\overline{\upsilon}$ equals the average variance. Cronbach's α of 0.8 is considered high enough in most social science experiments (Carmines & Zeller, 1979). This is used in Chapter 4.

3.3.5. Ethical Considerations

Ethical considerations are an important part of any social research projects involving human participants. Hall (2008) and Sieber (1998) highlighted three important ethical considerations which should be addressed by the researcher, during the planning stages, to adhere to external and institutional standards. The ethical considerations I took into account are described below.

Voluntary informed consent: Invitation to participate was accompanied with a plain language statement to explain the expected involvement of participants (see Appendix III). The participants were informed that were are not obliged to participate, and the outcomes of the study would not affect the grades of participants. I informed participants that they could participate in their own time and deciding not to participate would not affect their marks, or their relationship with the researchers.

Privacy and confidentiality: I take responsibility for the confidentiality of the participants and will comply with a strict code of ethics to report and archive the collected data. Data will be held securely in password-protected computer files and deleted after 5 years. Identity of individuals is not be revealed in this thesis or in any other articles published from the research.

Risk assessment: I assessed the risk posed to participants, and classified the study as 'low risk' because it was expected to cause only minor inconvenience or discomfort to the participants.

For this study, I addressed the ethical considerations described above in the human ethics application to seek approval for conducting social research. I attempted to mitigate the potential conflict of interest, owing to my dual role of teacher and researcher, by not having assessable tasks in the study. I obtained a *Human Ethics Certificate of Approval* from Monash University Human Research Ethics Committee (see Appendix IV).

3.4. Activities for Empirical Study

In order to address the research questions, SL was evaluated for its effectiveness in helping to learn computer programming concepts. In the remainder of the chapter, I address details of the particular ways that SL works to scaffold student learning about programing.

The empirical study was realized in a virtual classroom which was implemented using the building blocks of SL described in Section 2.8, and leveraging the pedagogical potential of SL described in Section 2.9. The activities for the empirical study were informed by the cognitive paradigm and the social paradigm of the learning theory, discussed in Section 2.2. Robins, Rountree, and Rountree (2003), proposed a three-pronged comprehensive approach to teach/learn programming, addressing knowledge required to design, generate and evaluate programs. In this empirical study, I focused on reinforcing programming concepts rather than comprehensive programming. Therefore, the proposed activities did not require participants to generate programs. The focus of the learner was directed away from programming language syntax, and directed towards evaluating programs or reviewing programs, and giving comments which is similar to the approach adopted by Lister and Leaney (2003). In this study, I also emphasized collaboration amongst participants to complete the activities. Since the participants were novice programmers, the learning activities were designed to achieve lower levels of cognitive development, as articulated in Bloom’s revised taxonomy of learning objectives (see Section 2.4).

The staircase example of Rosenbaum (2008) was used to develop activities for the empirical study. Rosenbaum (2008) used Scratch for Second Life (S4SL) to create Linden Scripting Language (LSL). LSL code within an object is activated when the object is touched by an avatar, and a staircase is created. The final resting position of the object is at the end of the staircase. A screenshot of the staircase example of Rosenbaum (2008) is shown in Figure 3.1.

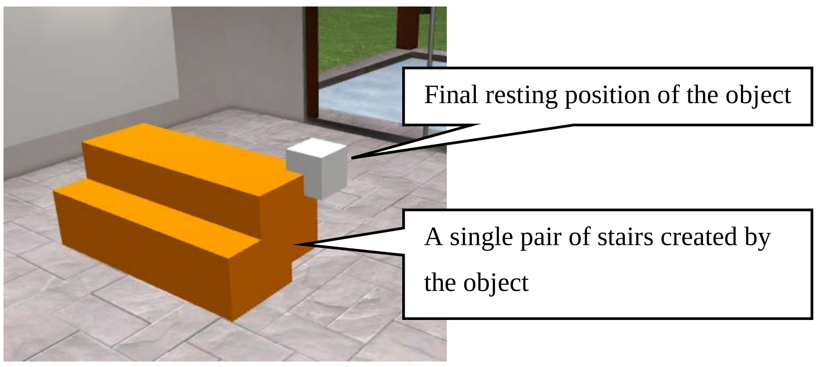

Final resting position of the object
A single pair of stairs created by the object

Figure 3.1. Staircase is Created When the Object is Touched by an Avatar

As described in Section 3.3.2, S4SL is a low-floor and high-ceiling (powerfully expressive) programming tool. I decided to use S4SL because the participants did not have extensive programming experience. Students at a large Australian university were invited to participate in the empirical study about the use of SL as a learning environment for computer programming. The empirical study was implemented for the three scenarios of variables, iteration statements and conditional statements. The S4SL code for the staircase example (described above) was modified for each scenario. For the different scenarios, objects with relevant code were provided to the participants in the SL inventory (see Section 2.8). These objects could be materialized in SL by dragging them from the SL inventory to the virtual classroom (referred to as *rezzing*). Participants could examine the behavior of an already rezzed object by clicking on the object. The code within an object exhibits the behavior which is referred to as *visualization*. Behavior of an object could be visualized with or without editing the code within the objects. Some activities required the code to be edited, while other activities did not require the code to be edited. During attempting each activity, participants were required to complete short questions in the printed handouts (see Appendix I) which were distributed to them before they commenced the activities. These questions served to reinforce the directions given to the participants.

The tasks were informed by cognitive constructivist learning approach and categorized as a code walkthrough and scaffolded code authoring, as proposed by Van Gorp and Grisson (2001) (see Section 2.2). Three sources of evidence, namely, a questionnaire, SL log files, and observation of participants were used for data collection, as described in Section 3.3. Participants engaged in tasks related to the three scenarios of variable, iteration statement, and conditional statement, as described in the following subsections.

3.4.1. Variable

In computer programming, a variable is a label which is assigned to data. After a variable is created, it can be used to manipulate the data within the computer program. Variables are very powerful because they can be used to exchange data between programs or segments of programs (referred to as modules). Here, I use an example to explain the role of variables in a program. Let us say, there is a program to compute the interest payable on a loan. The program can compute the interest payable when two data are supplied to it, namely, the loan amount and the rate at which interest is charged. Assuming this program is used by a corporation, the two data will vary for different clients and therefore the program will need to be customized for each client. However, this can be avoided and the program can be streamlined by using a variable for each data. Values can be assigned to these variables using a separate code segment, without having to modify the core program which is dedicated to computing the interest rate. This is shown in the schematic below.

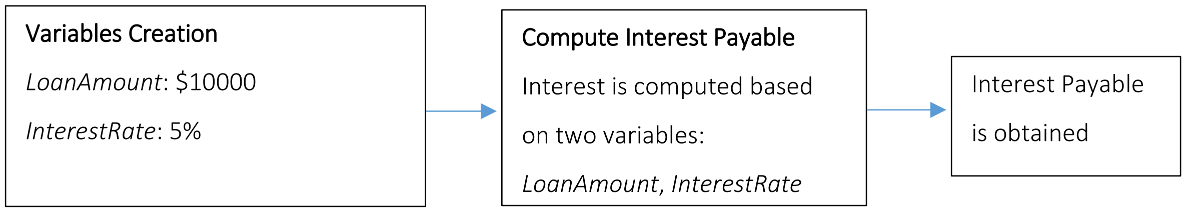


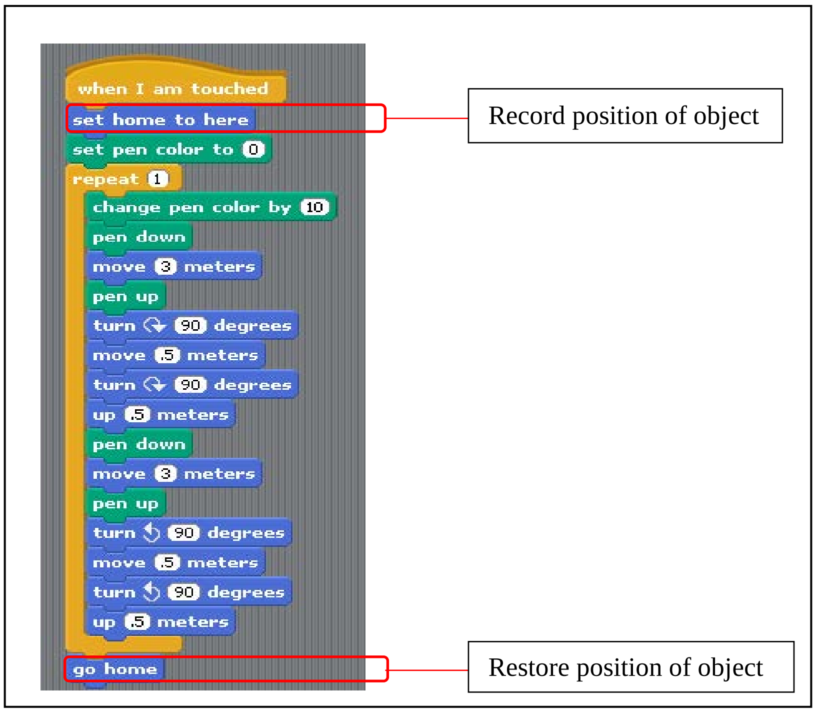
Figure 3.2. Using Variables

The schematic above shows two code segments which are coupled together to compute the interest payable. Compute Interest Payable program segment computes the interest payable with two variables. Avoiding the use of concrete values in the Computing Interest Payable streamlines the program by building program segments which are loosely dependent on each other. In the context of this example, the two program segments (Variables Creation and Compute Interest Payable) are referred to as modules, and the programming style is referred to as modular programming. Modular programming is significant to achieve reusability of code by eliminating redundancy is the program. In this example, redundancy is eliminated by reusing Compute Interest Payable segment for different values of LoanAmount and InterestRate. Modular programming reduces the expense incurred for maintaining code because there is no redundancy, and minimal code needs to be maintained. Modular programming is a good programming practice and it relies heavily on variables.

I used a code walkthrough (see Section 2.2) to implement the activity related to a variable. Hence, in this activity the participant needed to examine code, and its behavior. The participant did not need to write or edit code. Two pieces of code used for the activity were supplied in two objects in the SL inventory. Participants could place an object in the virtual world by dragging it from the inventory into the virtual classroom. The participants were required to examine the two codes, and answer the questions in the handout (Appendix I, Activity 1). The first piece of code was identical to Rosenbaum's staircase example, described above. The second piece of code used a variable, and the pseudocode is as below.

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

S4SL code for the pseudocode is shown in Figure 3.3.



```
when I am touched
set home to here
set pen color to 0
repeat 1
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move .5 meters
    turn ↻ 90 degrees
    up .5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move .5 meters
    turn ↺ 90 degrees
    up .5 meters
go home
```

Figure 3.3. Restore Position of Object After Staircase is Created

Two statements highlighted in Figure 3.3 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. The behavior exhibited by the code is that after completion of the staircase, the virtual object returns to the position it had at the beginning. S4SL code generated from Figure 3.3 was embedded in an object in SL. The behavior of the code is shown in the screenshot in Figure 3.4.

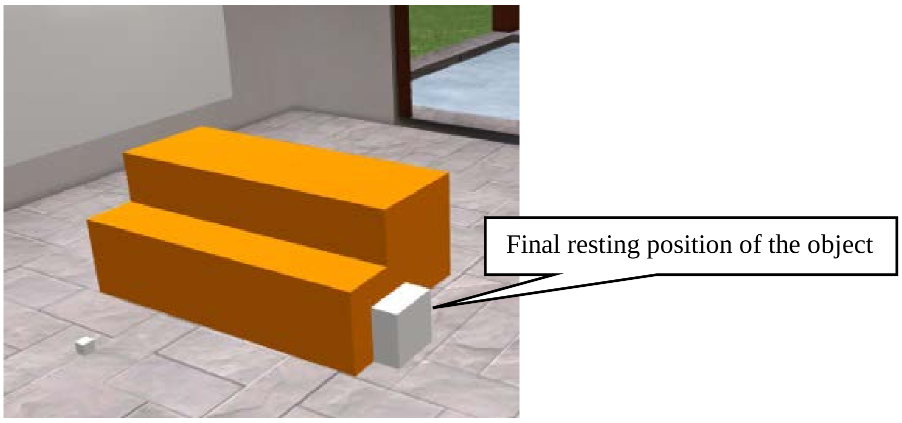


Figure 3.4. Visualization of Scenario #1

The main difference between the codes of Figure 3.1 and Figure 3.4 is in the final resting position of the object; in other words, the resting place of the object after the creation of the staircase. In Figure 3.1, the object rests at the end of the staircase; in Figure 3.4, the object returns to its initial position after creating the staircase. A variable is used to realize the return of the object to its initial position.

3.4.2. Iteration Statement

In computer programming, an iteration statement is used to repeatedly execute a code segment. Iteration statement is very powerful because it reduces redundancy in programming. Assume a program which needs to repeatedly execute a program segment; in this case, the program segment which needs to be repeated can be embedded within an iteration statement. The concept of an iteration statement is explained in the schematic below.

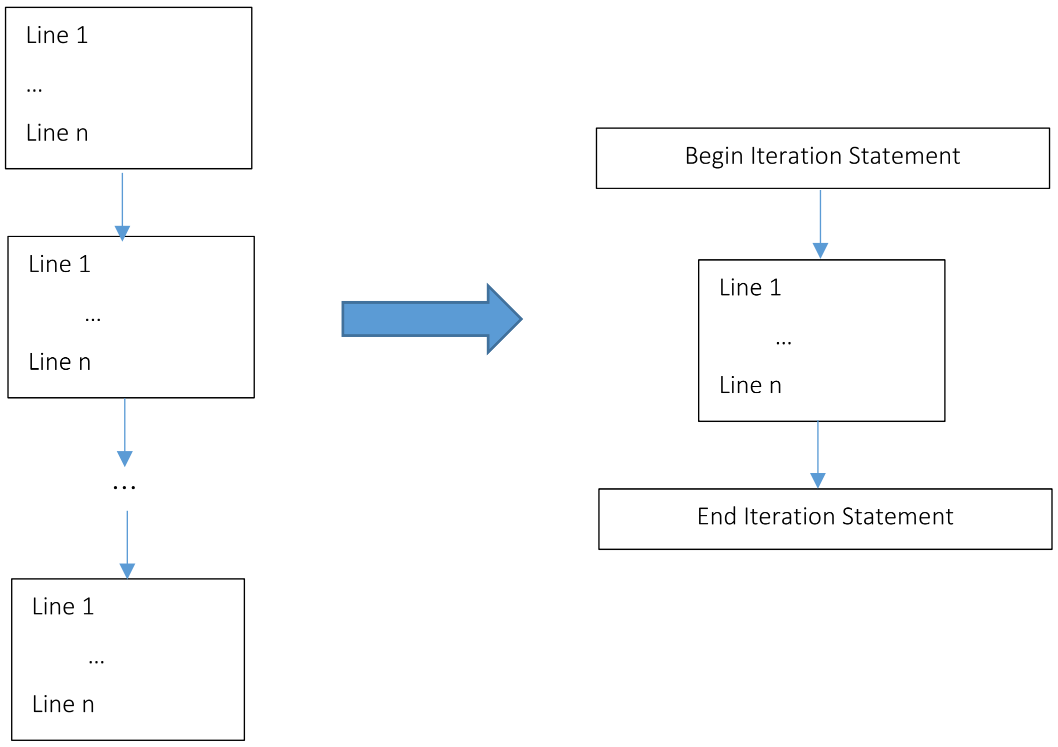


Figure 3.5. Iteration Statement

On the left side of Figure 3.5, the code segment comprising Line 1 to Line n, is repeated multiple times; this approach does not use an iteration statement. The schematic on the right side of Figure 3.5 is equivalent to the schematic on the left but it uses an iteration statement. On the right, the code segment which needs to be repeated is embedded within an iteration statement. The iteration statement will repeatedly execute the code comprising of Line 1 to Line n. The number of iterations can be specified for the iteration statement. Using an iteration statement clearly reduces the redundancy in the coding style, and streamlines the coding process.

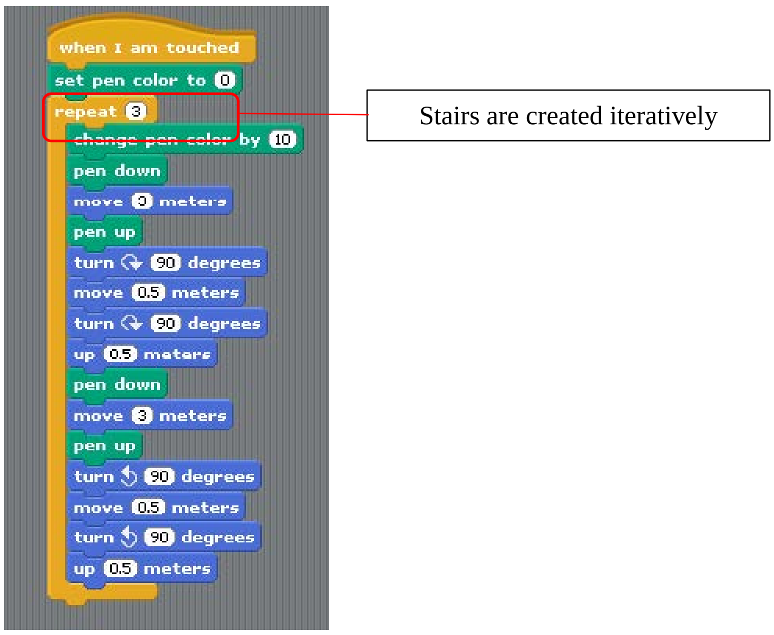
The pseudocode for an iteration statement is shown below. The first line has a variable called counter which controls the number of times the iterations are performed. For instance, if initialvalue is 1 and finalvalue is 10, the statements 1 through statement k will execute 10 times over, assuming that the counter will increment by 1 after each iteration; the iterations will stop when the counter becomes equal to the finalvalue.

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

I used scaffolded code authoring (see Section 2.2) to implement the activity related to an iteration statement. Hence, in this activity, the participants were required to edit a small piece of code, and answer the questions in the handout (Appendix I, Activity 2). The code used in the activity is based on staircase example (Rosenbaum, 2008), described above. In the staircase example of Rosenbaum (2008), when an object is touched, it creates a single pair of stairs, as shown in Figure 3.1. In this activity, the code had to be edited to create multiple pairs of stairs which was achieved by embedding the staircase creation code with an iteration statement. The pseudocode for implementing this scenario is the following. The number of pairs of stairs will depend on finalvalue in the pseudocode below.

```
for variable = 1 to finalvalue
    create single pair of stairs
end iterations
```

S4SL shown in Appendix I (Activity 2) was made available to participants, through an object in the SL inventory. The participants could place the object in the virtual world, by dragging it from the inventory into the virtual classroom. Participants were required to edit the code, as shown in Figure 3.6.



```
when I am touched
set pen color to 0
repeat 3
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move 0.5 meters
    turn ↻ 90 degrees
    up 0.5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move 0.5 meters
    turn ↺ 90 degrees
    up 0.5 meters
```

Figure 3.6. Create a Staircase Using an Iteration Statement

In Figure 3.6, the number of stairs is controlled by the counter supplied to the repeat statement. The behavior of the code is exhibited in the screenshot shown in Figure 3.7.

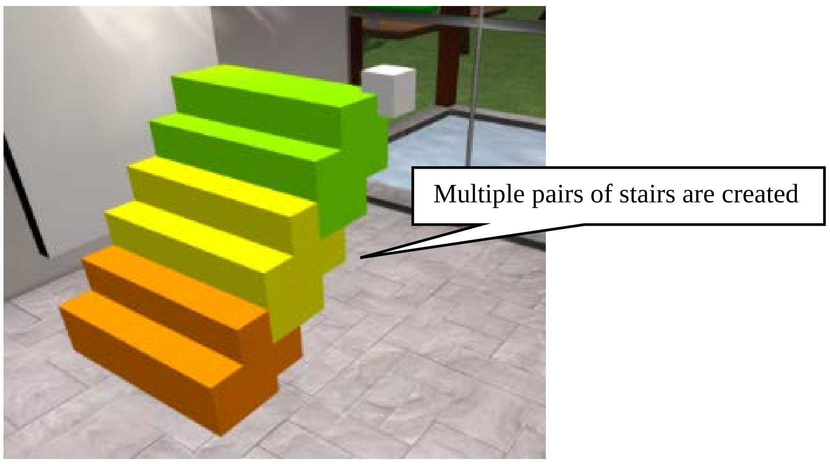


Figure 3.7. Visualization of Scenario #2

The main difference between the codes of Figure 3.1 and Figure 3.7 is in the number of stairs created. In Figure 3.1, a single pair of stairs is created; in Figure 3.7, multiple pairs of stairs are created. An iteration statement is used to realize the creation of multiple pairs of stairs, shown in Figure 3.7.

3.4.3. Conditional Statement

A conditional statement is used to execute a program segment, if a predetermined condition exists. A conditional statement is used to determine if a certain condition holds good; if the condition holds good, a set of statements is executed otherwise the set of statements is ignored. This is shown in the schematic below.

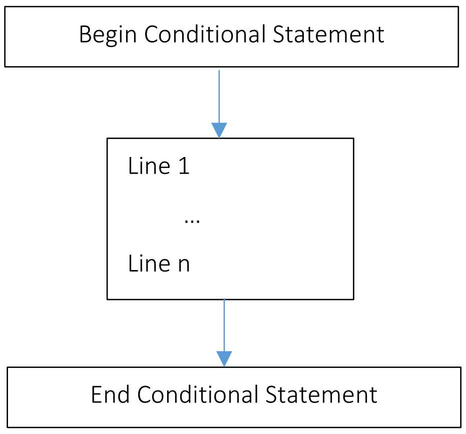


Figure 3.8. Conditional Statement

The code segment comprising `Line 1` through `Line n`, is only executed if the conditional statement decides that the predetermined criteria exists. Here, I use an example to explain the concept of a conditional statement. Let us say that a program is required by a corporation to send out promotional emails to the clients. Further, let us assume that the promotional emails are required to be sent to clients aged over 65 years. In other words, the promotional emails need to be sent conditional to the client age being more than 65 years. A programming solution can be implemented by using a conditional statement. The conditional statement will check the age of each client, and send out the email if the client matches the criterion. The pseudocode for a conditional statement is shown below. The segment from statement 1 through to statement k is executed based on the existence of a condition which is tested in the first line.

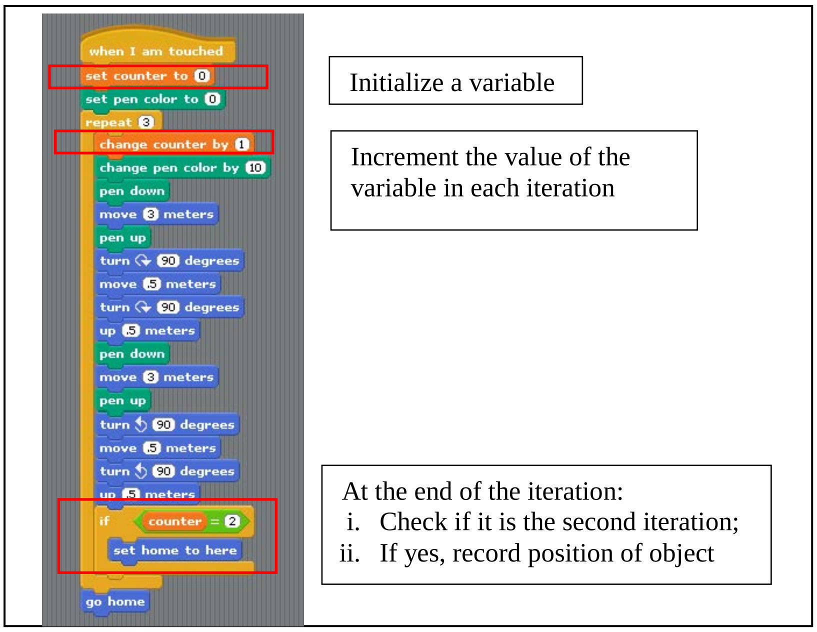
```
if (condition exists)
    statement 1
       . . .
    statement k
end conditional statement
```

Similarly to the activity for an iteration statement, I used scaffolded code authoring (see Section 2.2) to implement the activity related to a conditional statement. Hence, in this activity, the participant were required to edit a small piece of code, and answer the questions in the handout (Appendix I, Activity 3). The code used in the task was based on Rosenbaum's staircase example, described above. In this activity, when a virtual object was touched, it created multiple staircases similar to Figure 3.7; however, after completion of the staircases, the virtual object returned to the position it had at the end of the second iteration. The task used the concepts of variables and iteration statements, discussed in Sections 3.4.1 and 3.4.2. Additionally, it used the concept of a conditional statement. The pseudocode for implementing this activity 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)
6                then (remember the current position)
7            end conditional statement
8        end iterations
9    object returns to the position which was stored in line 6
```

In the pseudocode above, a conditional statement is used in line 5. The conditional statement terminates in line 7. Hence, the statement in line 6 is executed if a predetermined condition holds true.

S4SL shown in Appendix I (Activity 3) was made available to participants, through an object in the SL inventory. The participants could place the object in the virtual world, by dragging it from the inventory into the virtual classroom. Participants were required to edit the code to insert the conditional statement, as shown in Figure 3.9. When the object was touched by an avatar, the script responded by creating a staircase. After the staircase was created, the object returned to the position it held at the end of the second iteration.



```
when I am touched
set counter to 0
set pen color to 0
repeat 3
    change counter by 1
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move .5 meters
    turn ↻ 90 degrees
    up .5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move .5 meters
    turn ↺ 90 degrees
    up .5 meters
    if counter = 2
        set home to here
go home
```

Initialize a variable
Increment the value of the variable in each iteration
At the end of the iteration:
i. Check if it is the second iteration;
ii. If yes, record position of object

Figure 3.9. Return the Object to a Predetermined Position

The visualization of the code in Figure 3.9, is shown in Figure 3.10. The main differences between the codes of Figure 3.1 and Figure 3.10 are: first, in Figure 3.1, a single pair of stairs is created, and multiple pairs of stairs is created in Figure 3.10; second, the final resting position of the object, in other words, the position of the object after the completion of the stairs, is different. This functionality is achieved by using a conditional statement.

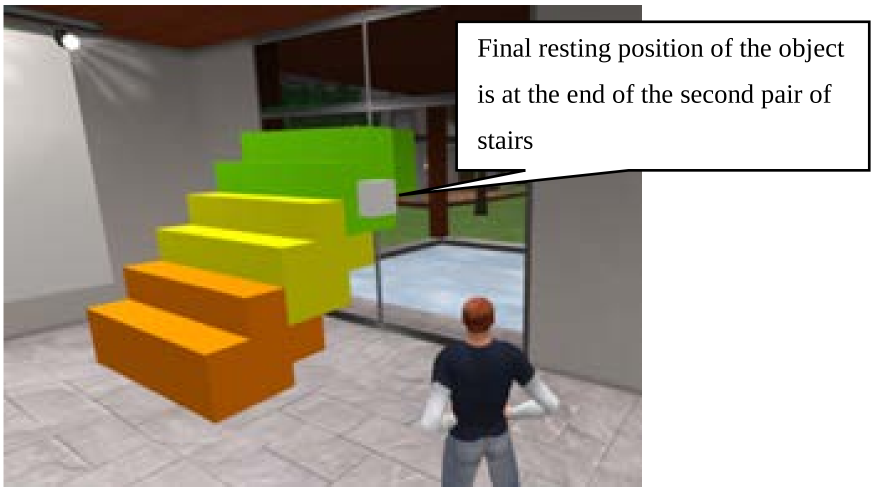

Final resting position of the object is at the end of the second pair of stairs

Figure 3.10. Visualization of Scenario #3

3.5. Conclusion

In this study, I considered but discounted the use of Action Research because of time constraints. The adopted research methodology, namely, case study, is used by me to address the research questions: *To what extent do IT students find SL helpful in understanding programming concepts; and how usable do participants find the SL environment?* Further, I described the sources of evidence for a mixed method approach to collect data. I adopted a learner-centered approach to engage participants in the learning activities. I observed the actions of the participants and also provided in-world assistance by adopting the role of the coach. To assess SL in the context of the research questions, tasks categorized as code walkthrough and scaffolded code authoring were implemented in-world as described in Section 3.4. The activities were designed to encourage the participants to use prior knowledge, and work collaboratively to solve authentic problems. Participants documented their experience by completing a questionnaire. Besides the questionnaire, the sources of evidence are the SL log files and observation data. The data collection and data analysis are described in the next chapter. In the next chapter, I will analyze the data to address the research questions.

Chapter 4

Findings

4.1. Introduction

Data were gathered from an empirical study designed in Second Life (SL) which is an elaborate learning environment, as described in Chapter 2. Activities for the empirical study were described in Chapter 3. Broadly, constructivist learning was adopted for the empirical study. More specifically, a social constructivist paradigm (see Chapter 2) was used because of the emphasis on social interaction. In line with this paradigm, each participant was assigned a group. Prior to engaging in activities, the participants were inducted into the SL environment. Each participant was required to complete the activities individually although they were encouraged to seek assistance from, and provide assistance to, other members of the group. A total of 20 minutes was allocated to these activities. I advised the participants to spend between six and seven minutes on each of the three activities. If a participant completed an activity before other members of the group, he/she was encouraged to offer assistance to others until everybody completed the current activity.

In Section 4.2, I address the background of the respondents which was used to assign each participant to a group. Following a short induction, participants engaged in the empirical study which used a mixed-method approach for data gathering, as discussed in Section 4.3. Quantitative data collected from the questionnaire are presented in Section 4.4. Qualitative data collected from open-ended questions in the questionnaire and observation notes are presented in Section 4.5. A discussion of the findings in the context of the research questions is presented in Section 4.6. The chapter is concluded in Section 4.7.

4.2. Background of Respondents

Informed consent was sought from students enrolled in introductory programming courses. I solicited expressions of interest to participate in the study, from students enrolled in introductory programming courses. I corresponded by email with each respondent to establish his/her background from 5 questions about the gender, current IT course, and IT experience; it served as an external factor which influenced the data gathered. The background data were used to identify prior skills of participants which could influence their capability to complete the activities used in the empirical study. After establishing the background of the respondents, each participant was assigned a group so that each group had the capabilities which gave them a good chance to complete the activities.

Table 4.1 summarizes the background of the twelve people who consented to participate in the study. I used the data in Table 4.1 to determine the composition of the groups. When constituting groups, I aimed for each group to have the desirable capabilities required to complete the activities which would enable participants to understand concepts readily. In the context of this study, I identified that the capabilities which each group should possess are: the ability to effectively interact with the SL environment which is achieved by having familiarity with SL or similar virtual world environment; and the ability to actively interact with group members for knowledge exchange and knowledge construction.

Table 4.1. Background of Participants

Name*	Gender	Gaming Experience	Currently Enrolled Course	Prior Learning in IT	Programming skills
Adam	Male	No	Master of IT	0 months	None
Aaron	Male	No	Bachelor of IT	0 months	None
Shane	Male	Yes	Bachelor of IT	0 months	None
Kate	Female	No	Bachelor of IT	0 months	None
Gavin	Male	No	Bachelor of IT	12 months	HTML and CSS
Chris	Male	No	Master of IT	0 months	None
Jenny	Female	No	Master of IT	0 months	None
Guy	Male	Yes	Bachelor of IT	6 months	Web design
Sam	Male	No	Bachelor of IT	0 months	None
Rohan	Male	No	Bachelor of IT	0 months	None
Jane	Female	No	Bachelor of IT	0 months	None
Liz	Female	No	Bachelor of IT	0 months	None

* Pseudonyms are used to comply with the ethical requirements of the project as discussed in Chapter 3.

Below, I elaborate each aspect of the background of participants and identify capabilities of individuals, such as: IT skills acquired from prior learning, and familiarity with virtual world environments albeit in the context of extra-curricular activities such as gaming and virtual tourism. I draw relationships between the capabilities possessed by individual participants and the desirable capabilities of each group. The existence of these capabilities in each group will potentially enhance the groups' chances of successfully completing the activities.

4.2.1. Programming Skills from Prior Learning

In order to answer the main research question, I needed to collect data from novice programmers who engaged in empirical activities, and completed a survey. Therefore, I proceeded to recruit participants who were novice programmers, and exclude respondents who possessed significant programming skills. This question gathered data about prior formal learning, and sought to assess computer programming skills acquired from prior learning. The purpose of the question was to help me to preclude potential participants, if they possessed significant programming skills because data gathered from such participants would not be relevant to answer my primary research question: *To what extent do IT students find SL helpful in understanding programming concepts?* I restricted participation to students who were recently enrolled in a programming course, and have not undertaken any prior studies in programming, so that the data are relevant to this research question.

Gavin and Guy reported possessing programming skills but I did not disqualify them from participation because their skills were not core programming skills. The skills possessed by Guy and Gavin, namely, HTML and CSS, and Web design, did not cover programming concepts which formed the basis of the SL activities.

4.2.2. Generic IT Skills from Prior Learning

Similarly to the above question, this question also gathered data about prior formal learning but sought to assess IT skills other than computer programming. I established the criteria for generic IT skills as: a person who had undertaken an IT course albeit a non-programming course, prior to the current course. Participants identified as possessing generic IT skills would be distributed amongst different groups because they will be in a position to offer assistance to their peers, enhance intra-group interaction, and ultimately improve their group's chances of successfully completing the activities.

Most participants (10 of 12) reported prior learning in IT of 0 months because they had not undertaken any formal instruction in IT, prior to commencing the current IT course. Hence, these participants did not have any academic background in IT, prior to commencing the current IT course at University. These participants did not study IT in high school. Gavin and Guy completed an academic course in IT but their courses did not include computer programming. Hence, Gavin and Guy possessed prior formal learning in IT but were novice computer programmers. The data led me to establish that Gavin and Guy possessed generic IT skills and perhaps more confidence in the context of the activities which would enable them to assist other members of the group. I therefore assigned Gavin and Guy to different groups to facilitate active intra-group interaction.

4.2.3. Gaming Experience

As SL is a 3D virtual world immersive environment, prior experience in similar gaming environments would likely affect the experience of the participants when attempting SL activities. These data were used to distribute participants with gaming experience between the groups with the aim that each group can benefit from the inherent skills of gamers.

Two participants, namely, Shane and Guy had gaming experience in virtual world environments; other participants did not have exposure to any virtual world environment. I used these data to ensure that Shane and Guy were assigned to different groups with the aim to enhance interaction within each group.

4.2.4. Gender

I collected data about gender because it is relevant with regards to the exposure and/or interest of participants in 3D virtual world immersive games. The commonly held perception is that computer games are played predominantly by males. Data about gender were complemented with data about gaming experience (see Section 4.2.3) to draw relationships between gender, computer gaming experience and familiarity of SL virtual world environment.

The gender data were used to achieve authentic gender distribution in each group; hence, the gender ratio of the participants is reflected in the gender ratio of each group. There were eight male and four female participants which reflects the typical gender distribution in IT courses. Two male students, namely, Shane and Guy have gaming experience in virtual world environments, and none of the female participants have exposure to any virtual world environment. I use this data to ensure uniformity in gender ratio amongst the different groups with the aim to enhance interaction within each group

4.2.5. Course Currently Studying

This question was used to gather data about the IT course the participants were currently studying. While all participants were recruited from an IT school at an Australian university, they were studying introductory programming courses at different levels, namely, undergraduate and postgraduate levels. The data are relevant because postgraduate students are more mature and likely to have better inter-personal communication skills which can be leveraged to enhance the level of intra-group interaction.

Data showed that three participants were enrolled in Master of Information Technology (MIT) and nine were enrolled in Bachelor of Information Technology (BIT). Participants studying the MIT did not have IT background because their undergraduate degree was in a non-IT area. The three postgraduate students, namely Adam, Chris and Jenny, reported that they studied non-IT courses in their undergraduate studies. Adam and Chris reported having completed business courses; Jenny reported that she completed an arts degree in history. I used the data obtained for this question for the composition of groups with the aim to achieve active interaction within each group.

4.2.6. Composition of Groups

First, I used data obtained in Section 4.2.1 to assess prior programming skills of participants, and determined that no respondent should be excluded persons based on existing programming skills. Second, I used data based on Section 4.2.2 - Section 4.2.5 to determine the composition of each group. In creating the groups, I analyzed the background of participants and carefully considered the capabilities which each group should possess to achieve the best learning experience. As discussed above, I identified that each group should possess: the ability to effectively interact with the SL environment which is achieved by having familiarity with SL or similar virtual world environment; and the ability to actively interact with group members for knowledge exchange and knowledge construction. Gavin and Guy possessed generic IT skills and therefore were assigned to different groups. Shane and Gavin had prior experience in virtual worlds which was achieved through gaming, and therefore were assigned to different groups. There were four female participants and therefore at least one female was assigned to each group. There were three participants who were postgraduate students and therefore one was assigned to each group. The final composition of the groups was as below.

Group 1: Adam (postgraduate), Sam, Kate (female), and Gavin (IT experience, gamer).

Group 2: Aaron, Shane (gamer), Jenny (female, postgraduate), and Liz (female).

Group 3: Guy (IT experience), Chris (postgraduate), Jane (female), and Rohan.

Each group engaged in the activities designed for the empirical study. In the next section, I describe the approach adopted for data collection.

4.3. Data Collection

In order to avoid distraction by features of SL, and content within SL, participants were inducted into the SL environment prior to engaging in activities. In the induction, I asked the participants to view a short video, in their own time - the video provided an overview of SL with a focus on the functionality required to complete the activities; this video is created by University of North Carolina (University of North Carolina, 2010). This video focused on materializing an object in the virtual world by dragging it from the inventory to the virtual world (referred to as *rezzing*), examining the behavior of the object, and viewing the code embedded within the object. The video also demonstrated means of written and verbal communication in SL. After participants finished viewing the video, I explained the activities to be undertaken. As explained in Chapter 3, the three activities undertaken by participants related to high level programming concepts of variables, iteration statements and conditional statements. The activities required rezzing of objects which were made available in the SL inventory. Printed handouts (Appendix I) were also distributed to participants before they commenced the activities. The activities required that the participants to examine and/or edit the code embedded within objects, and answer questions in the handouts. I encouraged the participants to interact with other participants by responding to questions, and asking questions.

Activities within SL were used for data collection. Linden Scripting Language (LSL) is the native scripting language (programming tool) within SL. The complexity of LSL deterred me from using it in the activities. Using LSL would shift the focus of the activities to syntactical details, thereby detracting from the research focus of programming concepts. Therefore, I used a visual tool, namely, Scratch for Second Life (S4SL), for the participants to undertake the activities. Adoption of S4SL (described in Chapter 3) is consistent with the research focus of the study to assess gain in understanding of programming concepts rather than the syntactical aspects of a programming language. Knowledge of programming concepts, once gained, can be applied to different programming languages. Quantitative data and qualitative data were gathered from the empirical study, as described below.

4.4. Quantitative Data Results

In this section, I present quantitative data gathered from the questionnaires. The Technology Acceptance Model (TAM) was used to design the questionnaire for gathering quantitative data. TAM is widely used in IT diffusion research, as discussed in Chapter 3. Quantitative data were gathered for two relevant factors in the TAM model, namely, perceived ease of use (PEOU) and perceived usefulness (PU). Data from the Likert-scale questions in the questionnaire are summarized in Tables 4.2 and 4.3. The Likert scale questions in Table 4.2 and Table 4.3 have five options: Strongly Agree (SA), Agree (A), Neutral (N), Disagree (D) and Strongly Disagree (SD). Mean and Standard Deviation is computed by assigning values in a 5-point scale to each option: 5 to SA, 4 to A, 3 to N, 2 to D and 1 to SD. It is safe to assume that in providing responses, participants compare the PEOU and PU of SL with other programming environments used by them. In their current programming curriculum, participants were using Microsoft Visual Studio .Net (VS .Net) Integrated Development Environment (IDE). Therefore, it may be assumed that VS .Net IDE is the benchmark. Discussion and analysis of the data are presented in the context of the research questions and the literature. The items in Table 4.2 are adopted from TAM constructs discussed in Chapter 3.

Table 4.2. Perceived Ease of Use

Questions in respect to SL activities	SA	A	N	D	SD	Mean	Std.Dev
1. Easy to learn	41.7%	41.7%	16.7%	0%	0%	4.25	0.754
2. Clear and understandable	50%	41.7%	8.3%	0%	0%	4.33	0.888
3. Easy to become skillful	16.7%	50%	25%	8.3%	0%	3.75	0.866
4. Easy to use	25%	33.3%	41.7%	0%	0%	3.83	0.834
5. Controllable	25%	58.3%	16.7%	0%	0%	4.08	0.668
6. Flexible	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

Each item in Table 4.2 is addressed below.

Easy to learn: A highly positive response was received with ten of twelve participants (n=10, 83.4%) indicating SA and A. Two participants indicated N (n=2, 16.7%). Nobody indicated D or SD.

Clear and understandable: In the questionnaire, I paraphrased this question as: SL tasks makes the programming concepts clear and understandable. A highly positive response was received with eleven of twelve participants (n=11, 91.7%) indicating SA or A. No one indicated D or SD. The activities were straightforward and I was careful not to include too many requirements which would potentially confuse the participants.

Easy to become skillful: In the questionnaire, I paraphrased this question as: It is easy to become skillful in programming tasks in SL/S4SL. While most participants indicated a positive response, one of the twelve participants (n=1, 8.3%) indicated D; no other question in the table got a D or SD. I attribute this to the background of this participant – he had limited IT experience and no prior exposure to a virtual world environment. Participants who did not have prior IT experience or exposure to virtual world environment may not have achieved adequate in-world experience to acquaint themselves with the in-world paradigm and fully appreciate this item. This item achieved the lowest mean value (3.75), owing to the background of few participants.

Easy to use: The question was used to assess whether SL is an easy to use environment for attempting programming tasks. The participants gave a positive feedback; however, the result did not reflect strong support for SL as an easy to use environment. On closer analysis of results, only two participants, Guy and Shane, indicated SA. According to my observation notes, Shane had prior experience in virtual world computer games. It is interesting to note that Guy, who did not have prior exposure to virtual world computer games indicated SA for this question. In my perception, this is possibly related with Guy's prior IT experience. Participants who did not have prior IT experience or exposure to virtual world environment may need more in-world experience to appreciate the ease of use characteristic of SL.

Controllable: In the questionnaire, I paraphrased this question as: SL is a controllable environment for attempting programming tasks. A highly positive response was received with ten of twelve participants (n=10, 83.4%) indicating SA or A. *Flexible*: Everybody indicated either SA or A (n=12, 100%). The data represents a strong perception that SL provides a flexible environment for learning programming concepts.

The items in Table 4.3 are adopted from TAM constructs discussed in Chapter 3; however, the item on *Job Performance* is not included because it is not deemed relevant for the activities.

Table 4.3. Perceived Usefulness

Questions in respect to SL tasks.	SA	A	N	D	SD	Mean	Std.Dev
7. Work more quickly	33.3%	58.3%	8.3%	0%	0%	4.25	0.621
8. Increased Productivity	50%	33.3%	16.7%	0%	0%	4.33	0.778
9. Effectiveness	58.3%	41.7%	0%	0%	0%	4.58	0.515
10. Makes job easier	16.7%	41.7%	41.7%	0%	0%	3.75	0.753
11. Useful	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

Similarly to Table 4.2, the Likert scale questions in Table 4.3 have five options: SA, A, N, D and SD. Each item in Table 4.3 is addressed below.

Work more quickly: In the questionnaire, I paraphrased this question as: I can perform the programming tasks quickly in SL/S4SL. A highly positive response was received with eleven of twelve participants (n=11, 91.7%) indicating SA and A.

Increased Productivity: In the questionnaire, I paraphrased this question as: SL improves my learning efficiency of programming concepts. A highly positive response was received with ten of twelve participants (n=10, 83.4%) indicating SA and A.

Effectiveness: In the questionnaire, I paraphrased this question as: SL is effective for learning programming concepts. This complemented the above question about productivity. The question received a 100% response (n=12) for SA and A.

Makes job easier: In the questionnaire, I paraphrased this question as: SL makes it easier to learn programming concepts compared with other environments. Participants who do not have prior IT experience or exposure to virtual world environment may need more in-world experience to fully appreciate this item. This item achieved the lowest mean value (3.75), owing to the background of few participants.

Useful: In the questionnaire, I paraphrased this question as: SL is useful for learning programming concepts. The question received a highly positive feedback with ten of twelve participants (n=10, 83.4%) indicating SA or A. One participant (n=1, 8.7%) indicated D for this question. No other question in the table received a response of D or SD.

Overall a good response was received for the TAM constructs (on average participants indicated Agree) despite few items in the Likert-scale questions achieving a low mean value. I used the reliability procedure in SPSS to compute Cronbach's α. A value of 0.868 was obtained for the 11 Likert-scale questions. As discussed in Chapter 3, this value of Cronbach's α reflects high internal consistency reliability amongst TAM constructs, in social science experiments.

4.5. Qualitative Data Results

Qualitative data presented in this section were gathered from responses to open-ended questions in the questionnaire, and my observation notes. Section 4.5.1 details students' reflections based on open-ended questions in the questionnaire. I have addressed my observation notes, and log file data in Section 4.5.2 and Section 4.5.3 respectively.

4.5.1. Students' Reflections

The open-ended questions from the questionnaire and the corresponding responses are detailed below.

Q: Did you find the tasks interesting?

The question sought response about the level of engagement in the activities. Participants gave positive comments about the interesting nature of the tasks developed for the empirical study. One participant simply commented: "The activities are interesting". This is a succinct response which did not provide further insight about the activities. One participant commented: "It is more interesting than just write down or read code". The comment broadly addresses the holistic approach adopted in this empirical study with SL as the enabling environment. The comment reinforces the importance of code walkthroughs and scaffolded code authoring in the context of constructivist leaning as addressed in Chapter 2. The rich features of SL are noted by another participant who said: "I have been doing programming for two months and I am amazed to see what we can do with programming". One participant addressed the visual nature of the activities by saying: "It is easy to learn programming by visualization". This is a significant comment because visualization is extensively used in programming environments as discussed in Chapter 2. The adoption of visualization in pedagogical approaches was addressed in Chapter 3. The comments in the questionnaire responses are influenced by the individual experiences of the participants while engaging in the activities which are purposefully designed for novice programmers. The comments reaffirm the suitability of SL for novice programmers. SL could potentially be used for intermediate and advanced level programming curriculum too; however, the activities were not designed to explore this. Generally, comments were positive about the potential of SL for teaching computer programming. Two participants indicated that SL is suitable for novice programmers by saying: "it very useful for beginners who want to clear the concepts of programming"; and "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?

All participants unanimously agreed that the virtual presence of the teacher was useful. One participant commented: "At the time of the activity, I had doubts and the virtual presence of the teacher clarified my doubts". The virtual presence of the teacher can provide cues to the participants to complete the activities. Peers may also assist in completing in activities - insight into the role of peers is sought from the next question. However, the role of the teacher is fundamentally different because the teacher is able to adopt a considered approach based on the concepts of zone of proximal development (ZPD), and scaffolding, as discussed in Chapter 2. Other comments along similar lines were:

> 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; Virtual presence of the teacher is helpful as teacher can always guide or give the solution if any doubt or problem is there; It is really good to have virtual presence of someone who can guide; 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; Yes, I think it is helpful. When I have questions, I can chat with the virtual presence of the teacher.

In a real-life implementation of curriculum delivery, the teacher's presence will be limited by time constraints. Although the teacher may not be able to help each participant individually, the presence of the teacher is expected to have a moderating influence in the virtual classroom and avoiding the 'tyranny of freedom' due to lack of structure (Schwartz, 2000).

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

The question sought response about the role of peers in accomplishing the tasks. The role of building a community with peers is described in Chapter 2. As noted above, the teacher may not be available all the time. If the teacher is unavailable, peers with a deeper level of understanding will be able to assist peers. It will result in the students building on the knowledge, skills and experience of their peers. Peer interaction is important to develop a higher level of cognitive understanding in programming. In-world resources such as videos will also be of assistance when the teacher and peers are unavailable. Participants perceived peer interaction as a positive influence. One participant commented: "It is really interesting to learn with peers in this environment". Another participant voiced support for peer interaction by stating: "Yes, SL provides a good social community. Peers could learn programming and share what they learn". Another participant commented: "Virtual presence of peers may distract the learner from his/her task". In collaborative environments, self-moderation by participants is important. In a real-life implementation of SL for curriculum delivery, this concern can be mitigated by reminding participants that the interaction needs to be relevant to the curriculum. This can be enforced by the teacher by reviewing SL log files which provide audio recordings of in-world conversations.

Q: Any other comments

The question sought feedback from participants about issues which were not articulated elsewhere in the questionnaire. One participant stated: "It can explain more programming concepts visually and we can learn fast. It is interesting to learn here". The comment addressed the visual nature of the activities. In this study, all activities use visualization of the staircase. When the avatar touches an object, a staircase is created as discussed in Chapter 3. Different staircases are created for objects associated with each activity. Visualization makes it possible for participants to observe if their interpretation of the code is accurate. If the participant is unable to explain the behavior of the object by examining the code, the object can be deleted and materialized (rezzed) again, and this process is repeated until the correct interpretation of the code is achieved. Relating the object behavior to the code, reinforces what participants have learnt. Extant literature is available about empirical studies to determine the educational value of visualization. Despite the intuitive appeal of visualization and its proven educational value, it has not been adopted widely in computer science education because of the time and effort required for creating and conducting such activities (Hundhausen, Douglas, & Stasko, 2002). In general, SL has the potential to use a number of pedagogies for designing learning activities. However, this is a small-scale study and findings cannot be used to make firm conclusions. Further, an empirical study with teacher participants is required to draw tentative conclusions about this issue. One participant stated: "SL is a good platform for understanding programming concepts as well as for entertainment". The "entertainment" opportunities afforded by SL are likely to be a distraction; however, it may serve to encourage students to visit the environment frequently and become acquainted with using the controls within the environment. The teacher needs to be careful about exposing students to inappropriate content such as adult movies and gambling. One participant commented: "It is really useful to understand programming fundamentals but implementing or understanding the logic is all up to the learners". Although the comment appears skeptical, it reflects that the ultimate aim of a pedagogical approach is to transfer knowledge to the learner with the aim that the learner is able to apply the knowledge. In problem-based learning, the student needs to take ownership of the problem. A participant stated: "I have not experienced SL but after doing the tasks, I am interested in SL". Another participant stated: "I think this is good for students who pay attention to details otherwise they are not able to identify the difference between objects". The comment reflects the concern that the activities may not be fully understood. The premise for the success of a problem-based approach is that the problem is understood by the learner. As discussed in Chapter 2, the teacher needs to play the role of the coach to ensure that the problem is understood by the students. The teacher should design authentic problems. Using the principles of experiential learning is also useful in this context.

4.5.2. Observation Notes

Besides the open-ended questions in the questionnaire, qualitative data were also gathered from my observation notes. The purpose of observation notes is to establish parallelism between multiple sources of data to ensure reliability through a robust design, as discussed in Chapter 3. The activities are completed by the participants in groups of four. The background of participants to ensure that the core features of constructivist learning existed for each group, as discussed in Section 4.2. I observed and made notes as each group attempted the activities in SL. While all groups were able to complete the activities successfully, there were differences in the performance of each group, and in the experience of individuals within each group. Observation data for each group are described and discussed below. I use the terminology proposed by Liu and Tsai (2008) to identify the different patterns of peer interaction within each group, as described in Chapter 2.

Group 1: The group was comprised of Adam, Sam, Kate and Gavin. Based on the background of participants which is established in Section 4.2, Gavin was relatively more experienced in this group because of prior IT experience (see Table 4.1). During the in-world activities, Gavin stated that he had experience in virtual world computer games; the SL environment therefore was a familiar paradigm for Gavin. Adam, Shane and Kate did not have prior IT experience (see Section 4.2). At the onset, Gavin could be regarded as the only peer with a deeper level of understanding. Gavin acquired a peer-leadership role. In this role, Gavin assumed an authority figure, and other group members frequently requested his assistance. He was motivated, confident and announced his actions in-world for the convenience of the group members. Gavin maintained a positive tone about the activities and encouraged his peers through the activities. Group 1 completed each activity before the allocated time of six minutes. Analysis of the observation data shows that much of the interaction was between participants; a small number of messages were exchanged between me and the participants. In fact, the communication between participants was skewed with most communication being explicitly directed to Gavin. This allowed me to observe passively. The group progressed relatively smoothly through the activities. I noted that there was a high degree of peer-interaction which is the premise for social constructivist approach, as described in Chapter 2. Interactions in this group exhibit characteristics of centralized knowledge exchange with Gavin being the knowledge center. While some peers may have a deeper level of understanding as articulated in the taxonomy of learning objectives (see Section 2.4), it is the level and quality of interaction which ultimately determines if there is knowledge exchange between the peers.

Group 2: The group was comprised of Aaron, Shane, Jenny and Liz. All members of this group did not have prior IT experience. Members of the group posted messages about the general nature of the tasks such as: *How do I begin,* and *where is the inventory?* Such messages, referred to as indirectly task-focused, were recorded at the early stages of the activity; messages thereafter were directly task-focused. Group 2 was slower than Group 1 although members completed each activity within the allocated time of six minutes. Members of the group were competent and were able to complete the activities with minimal guidance; however, there was less peer-interaction in Group 2, compared with Group 1. The majority of the interaction was between me and the participants. Although messages were not explicitly directed to me, the participants were not forthcoming in responding. I assisted the participants individually which slowed the progress of the entire group. I applied principles of ZPD and scaffolding (see Chapter 2) to provide support and guidance but refrained from spelling out the solution. An ability impediment described the pattern of interaction between the group members.

Group 3: The group was comprised of Guy, Chris, Jane and Rohan. In this group, Guy is relatively more experienced. Chris, Jane and Rohan do not have prior IT experience. Group 3 took longer than the allocated time for completing activity #1. Guy was first to complete activity #1, followed by Jane and Chris; they completed within the allocated time. Rohan took a little bit longer to complete the activity; he finished in 7 minutes and 20 seconds. My avatar reminded Rohan that he had exceeded the allocated time; however, I encouraged him to continue attempting the activity. Guy, Jane and Chris waited for Rohan to complete #activity 1 so that all the group members could progress to the next activity simultaneously. All members completed activity #2 and activity #3 within the allocated time. According to my observation notes, Rohan exceeded the allocated time to complete #activity 1 because he was unable to observe the behavior of the object which appeared to be obscured from view, by the staircase. The final resting position of the object needed to be observed, to complete the task sheet. I informed Rohan that the object was obscured by the staircase and I guided him to change the in-world orientation so the object came into view. After completing #activity 1, Rohan was comfortable with changing the orientation because he used this facility in activity #2 and activity #3. Hence, the delay in completing #activity 1 was caused by the lack of familiarity with the controls, in the SL viewer. The pattern of interaction in this group was distributed knowledge exchange because all members posted questions and responded to questions.

4.5.3. Log files

The log files contained text and verbal chat messages exchanged between participants. I reviewed the log files with the aim to identify and address anomalies in the observation notes. I found that the log files were consistent with the observation notes addressed in the above section. Although log files did not provide any new findings, they served to triangulate data which is an essential part of qualitative data analysis.

4.6. Discussion

This section moves from the specific data collection findings to a more general discussion about the findings in relation to my research questions. Each research question is addressed below.

4.6.1. Research Question 1

Research Question 1: To what extent do IT students find SL helpful in understanding programming concepts?

The research question aims to gauge the perceptions of novice computer programmers about SL as a learning environment for concept understanding in computer programming. Participants recruited for the pilot study were novice programmers and they engaged in activities designed for the pilot study. Multiple sources were used for data collection, namely, questionnaire and observation notes. The views and opinions of the participants inform the results of the pilot study. An analysis of the data in the context of this research question is presented below.

Analysis of qualitative data from the pilot study show that participants responded positively to the activities used in the empirical study - the activities were designed to challenge and intrigue the participants with the ultimate aim to achieve a high level of engagement resulting in building new knowledge based on existing knowledge. Using Vygotsky's (1980) concept of Zone of Proximal Development (ZPD), I was mindful to avoid making the activities too difficult which could potentially overwhelm the participants, and eventually result in low levels of engagement. ZPD and scaffolding (discussed in Chapter 2) were realized by me (in the role of the teacher), and the more capable peers which led to knowledge exchange in a social constructivist approach. I noticed a high level of motivation and engagement by participants. In terms of the complexity of the three activities, no single activity stands out as being more complex than the others. Participants who completed one activity quickly, also completed other activities quickly; similarly, if participants took longer to complete an activity, they took longer to complete other activities.

From sources of evidence, two additional themes emerge in relation to the experience of participants in engaging in group activities within SL for the purpose of knowledge exchange and building. These are: first, familiarity of participants with the environment; second, the collaboration amongst participants – this includes the level of interaction and the nature of interaction. These are addressed below.

Familiarity with any learning environment will inherently have a learning curve. Gaining familiarity will enhance the confidence in using the learning environment. Generally, the participants had limited familiarity of SL because of the limited time spent in-world. Limited familiarity with SL also meant that participants were curious and wanted to experiment with the controls. Few participants attempted to fly within SL which proved to be a distraction to others. As the participants worked their way through the activities, their actions became more deliberate because of common aspects in each activity with which they became increasingly familiar - these aspects were rezzing of objects from the inventory (as described earlier), and viewing the code associated with objects. Lack of familiarity with these aspects resulted in an exchange of messages which were not directly related to the tasks. However, the frequency of such messages, referred to as indirect task-focused messages, declined substantially between the first and the second activity which was consistent with the participants expanding their comfort-zone within the SL environment. In a real-life implementation for curriculum delivery, more training is recommended for students prior to engaging them in learning activities. Familiarity with the environment will increase focus of participants in attempting the tasks, and fluency in completing the tasks. In a real-life curriculum delivery setting, rules about the appropriateness of messages should be established to discourage off-task messages. Familiarity with the environment should be assessed and used as a criterion for composition of the groups to ensure that each group has the necessary skills to complete the learning activity, as discussed in Section 4.2.

Interaction amongst participants: Here, I address aspects of the level of interaction, and the nature of interaction, between participants.

The level of interaction – significant levels of interaction are important in social constructivist learning to enable knowledge exchange. The interaction initiated by an individual may be personality based. Various approaches can be adopted to encourage peer-interaction. However, it is inevitable that the interaction between participants is different for each group based on group dynamics emerging from the rapport, inter-personal skills and background. In a real-life curriculum delivery setting, participants can be assessed for initiating and participating in discussions, and SL log files can be reviewed for assessment purpose – the approach will discourage participants from lurking passively which would be ineffective for knowledge exchange and building.

The nature of interaction - Meaningful interaction amongst the participants for knowledge exchange should be relevant, and learner-centered. These aspects of the interaction are addressed here. For relevant interaction, measures need to be adopted to minimize indirect task-focused, and off-task exchange of messages. The interaction can be directed towards task-focused messages by giving ownership of the problem to the participants. Indirect task-focused messages can be minimized by ensuring that the participants understand the problem. Lack of familiarity may also result in indirect task-focused interaction, as discussed above. The interaction in some groups may be primarily between the teacher and learner, as observed for Group 2. Lack of learner-centered interaction is likely to be the case when all members in the group lack a deeper level of understanding and are thereby unable to assist peers. On the other hand, a deeper level of understanding does not guarantee the success of the group because skills to initiate interaction are important.

In order to facilitate learner-centered interaction, it is important to ensure appropriate composition of groups, as discussed in Section 4.2. If the interaction is primarily with the teacher, the teacher needs to adopt the role of a coach and encourage participants to interact with peers for knowledge exchange. Few participants were distracted by the inherent characteristics of virtual worlds such as off-task social interactions, and in-world entertainment. Approaches for mitigating the drawbacks of SL are suggested in Chapter 5.

4.6.2. Research Question 2

Research Question 2: How usable do participants find the SL environment?

This is an ancillary research question to evaluate the usability of SL. It is the extent to which SL facilitates an understanding of computer programming concepts. I adopted the TAM construct (see Section 2.10.2) to answer this research question. Data gathered from the questionnaire and collated in Section 4.3 were drawn upon to answer this research question by evaluating PEOU and PU. Cronbach's α of 0.868 is obtained for the Likert-scale questions, which indicated a high level of internal consistency reliability of the items in PEOU and PU. Hence, the questionnaire is seen as a reliable measurement instrument.

The responses to Likert-scale questions shown in Table 4.2 and Table 4.3 paint a positive picture of SL with regards to PEOU and PU. While the nuances of each criteria used to assess PEOU and PU are addressed in Section 4.4., it is worthwhile to reiterate that none of the respondents responded with "Strongly Disagree" to any of the Likert-scale questions. Therefore, the participants viewed SL as easy to use, and had an appreciation of the facilities of SL in the context of the activities.

4.7 Conclusion

Generally, the quantitative and qualitative data gathered from the pilot study strongly support the use of SL for novice programmers to learn programming, and the usability of SL. The empirical study was framed by a social constructivist approach; hence, the study implicitly reflects positively on the adoption of this pedagogical approach. SL is a virtual world environment and prior experience in such a paradigm cannot be assumed. Prior experience in a virtual world environment or lack of such experience has a strong influence in shaping the perceptions of participants. In the final chapter, I give an overview of the study and discuss the limitations of the research. I suggest implications of the findings and conclude the thesis.

Chapter 5

Conclusion

5.1. Introduction

Understanding of computer programming concepts is challenging because of the abstract nature of these concepts. My motivation for the study originated from my appreciation of this problem in the role of a Lecturer in Information Technology with extensive experience in teaching computer programming. The research purposes of this study were: first, personal curiosity from the perspective of a Lecturer in Information Technology; and second, achieving practical and cognitive outcomes. The practical outcome is related to driving the desired change, namely, better understanding of programming concepts by adoption of the proposed approach to teach introductory programming. The intellectual outcome is related to dissemination of findings which will potentially lead to more effective learning.

With these research purposes in mind, I conducted a literature review and established the premise of the study that novice programmers find it challenging to grasp computer programming concepts because of the abstract nature of these concepts. The significance of the problem links to the high fail rates in higher-education computer programming courses and ultimately the high rates of attrition from computer programming related disciplines. Through extant literature, I established the role of virtual world environments in learner-centered pedagogical approaches. In the context of computer programming, I described the pedagogical potential of Second Life (SL) and hypothesized its role for teaching programming concepts to novice programmers. I articulated two research questions in order to direct the study: *To what extent do IT students find SL helpful in understanding programming concept; and how usable do participants find the SL environment?*

I conducted an empirical study, based on learner-centered pedagogical approaches, to seek answers for the research questions articulated above. I developed a virtual classroom in SL using the building blocks of SL (see Section 2.8). I used this virtual classroom to conduct the empirical study. The empirical study employed a problem-based approach to engage novice programmers in learning activities. The complexity of the learning activities was commensurate with achieving low levels of cognitive development, as articulated in Bloom's revised taxonomy of learning objectives (see Section 2.4). Activities were undertaken by the participants in a collaborative manner, and I adopted the role of a coach to provide support and guidance using principles of Zone of Proximal Development (ZPD) and scaffolding. In Chapter 4, I analyzed the data from the empirical study and presented the findings of the study in the context of the research questions. I found that the activities challenged and supported the thinking of the participants. In this chapter, I conclude the thesis by providing an overview of the study, discussing the limitations of the study, and providing directions for future work.

5.2. Overview

In the initial chapter, I identified the research problem and articulated the research questions. In Chapters 2, a thorough literature review was conducted to identify a suitable pedagogy. An appropriate research methodology, for the purpose of collecting meaningful data, was identified in Chapter 3. I adopted learner-centered pedagogy, and a case study as the research methodology. A case study approach offered me an empirical investigation which is suitable for the real-life nature of the problem being investigated. In a case study, the data are strongly related to the experiences of participants. A case study is not used to establish patterns or themes unlike other approaches such as experiments or carefully structured questionnaires. Hence, a case study was suitable to investigate typical and exceptional experiences of participants. Learner-centered pedagogical approaches strengthens the study because it is characterized with a focus on solving authentic problems, and a high-level of learner engagement. As discussed in Chapter 2, despite the popularity of constructivist approach, it has not been widely used in programing (Ben-Ari, 1998; Van Gorp & Grisson, 2001). In this empirical study, I designed constructivist activities within SL, for the participants to understand programming concepts. As discussed in Section 2.3, the choice of SL as the learning environment was based on its potential to meet the student engagement guidelines. My adopted approach was based on the activation of prior knowledge for reinforcing concepts, knowledge exchange, and building new knowledge. I adopted twofold measures to achieve a high-level of engagement from participants. First, I adopted a learner-centered approach to facilitate active participation. Second, I leveraged the visual nature of the activities to enhance engagement of participants. In order to answer RQ1, I designed activities based on programming concepts of conditional statements, iterations and variables because of the significance of these concepts; extant literature (see Chapter 2) shows that novice programmers have difficulty in understanding these concepts. Since the learning activities were designed for novice programmers, the expected learning objectives were to achieve lower levels of cognitive development, including remember, understand and apply as described by Bloom's revised taxonomy of learning objectives (see Section 2.4). For the processes of data collection, data reporting and data archiving, I took measures to comply with ethical considerations in accordance with *Human Ethics Certificate of Approval* which was obtained from Monash University Human Research Ethics Committee (MUHREC). As discussed in Section 3.3, I adopted a mixed-methods approach for data gathering to seek answers for the research questions. In order to enhance the reliability of the findings, I used multiple sources of evidence for data collection, namely, questionnaires, observation data and log files.

In order to answer RQ1, I used qualitative data obtained from questionnaires, observation data and log files. I used my experience as an educator in computer programming and thematic analysis drawn from literature, to interpret and analyze the data. In order to answer RQ2, I adopted TAM constructs to evaluate Perceived Usefulness and Perceived Ease of Use, of Second Life. TAM has been widely adopted for evaluation of usability of technology as detailed in Chapter 2. Quantitative data obtained from questionnaires were used for evaluation of the usability of SL. I used the reliability procedure in SPSS to compute Cronbach's α (see Section 3.3). A value of 0.868 was obtained for the 11 Likert-scale questions. As discussed in Chapter 3, this value of Cronbach's α reflects high internal consistency reliability amongst TAM constructs, in social science experiments.

Multiple sources of data inherently increase the reliability of the findings. However, findings from this study cannot be generalized because of the small-scale nature of the study. This is further addressed in the next section.

5.3. Limitations

The data from the pilot study were interpreted and analyzed to evaluate SL in the context of RQ1 and RQ2. Findings of the study, described in Chapter 4, suggest that: first, the proposed activities in SL add value for novice programmers; second, SL is usable. Despite the positive nature of the findings, the study has limitations arising from constraints related to time and resources. In this section, I address the limitations in the context of each research question.

The limitations could potentially impact the accuracy of the findings in relation to RQ1. First, participants had limited exposure to SL. Therefore, the participants might not have fully appreciated the learning potential of SL because the study represents a snapshot. On the other hand, participants may have been unaware of perceived challenges in using SL – for example, in Section 4.5, it was noted that one participant lost view of an object in the virtual world and was unable to use SL controls effectively to re-gain view of the object. The likelihood of change in attitudes about SL affects the accuracy of the data. Second, while a case study has strengths, it has inherent limitations owing to the small number of participants. Having more participants would improve the reliability of the data and enable generalization of findings. Third, it should also be considered that I had a dual role in this research; besides being the researcher, I am the teacher of the participants. Being the teacher of participants has a two-fold impact – it may influence the data obtained from participants although the participants were informed that the study did not involve assessable tasks, and participation was voluntary (see Section 3.3.5); further, being the teacher of the participants may influence my interpretation of the data despite me trying to be objective during the interpretive process (see Section 3.3.4). Fourth, although findings reflect that participants have positive attitudes towards the proposed approach for teaching programming concepts to novice programmers, the participants' grasp of concepts is not conclusive because the study did not include formal assessment tasks. Hence, the evaluation was limited. Fifth, the level of interaction between participants was restricted to their group; hence, experiences in one group were not shared with other groups. Sixth, although visualization within SL received a good response from participants, the time and effort intensive nature of creating and conducting learning activities is not addressed in the study. Seventh, few participants were distracted by the inherent characteristics of virtual worlds such as off-task social interactions, and in-world entertainment. In Section 4.5, it was noted that a participant commented about the potential distraction from the virtual presence of peers. However, in a real-life implementation of SL for curriculum delivery, this concern can be mitigated by reminding participants that the interaction needs to be relevant to the curriculum. This can be enforced by the teacher by monitoring SL log files of in-world activities and interactions, as discussed in Section 2.8.

The study also has limitations in relation to RQ2. First, as with RQ1, limited exposure of participants to SL implies that the views and perceptions of the participants may not be well-informed because the study represents a snapshot. Second, data collection from more participants is likely to improve the robustness of the findings and enable working towards generalization of findings. Third, TAM model used for evaluating the usability of SL relies on user perceptions rather change measuring change in behavior, as discussed in Section 2.10. Measuring change in user behavior, however, will need knowledge of human psychology which will need the study to be implemented as a multi-disciplinary exercise.

5.4. Future Directions

In the future, I suggest that the study be extended to negotiate the limitations described in Section 5.3. In Section 5.3, I identified that a limitation of the study is the SL inexperience of the participants. As discussed in Section 5.3, lack of familiarity with SL controls affects the accuracy of data about user experience because of the likelihood of change in attitudes. This limitation can be overcome in a future study, by giving participants more in-world experience prior to engaging them in SL activities. However, potential participants will need to commit more time to gain familiarity with SL, unless they have obtained experience in SL in other contexts. Using SL over an extended period of time, will improve the robustness of data because the likelihood of change in attitudes will diminish. Further, familiarity with SL will alleviate indirect task related interaction amongst participants, and the participants can focus on the learning activities. Another limitation of the study is the small number of participants. A case study typically has few participants for qualitative data collection. On the other hand, the nature of quantitative data does not need the researcher to spend extended periods of time to interview individual participants; therefore, it is feasible to improve the reliability of the quantitative data by collecting data from more participants. In a future study, quantitative data may be collected from more participants thereby providing a deeper and broader sense of the usability of SL.

I also suggest that learner-centered interaction be broadened by adopting communal constructivist approach. In communal constructivist approach, the learner-centered interaction is not limited to participants within the group; hence, participants in one group are able to interact with participants from other groups. A communal constructivist approach requires that each group creates artifacts which become available to other groups. The aim of inter-group interaction is to transfer knowledge between groups (Tangney, FitzGibbon, Savage, Mehan, & Holmes, 2001). This approach will potentially enhance learning outcomes.

In this study, I discounted the use of Action Research (AR) because of time constraints. In future work, I propose to use AR methodology because it is suitable to determine and evaluate innovative technology adoption for education (Warden, Stanworth, Ren, & Warden, 2013), as discussed in Section 3.5.

5.5. Conclusion

In this study, I proposed and evaluated a constructivist approach for teaching computer programming concepts to novice programmers, using SL as the enabling platform. Analysis of data collected from the pilot study shows that the proposed approach is effective in teaching programming concepts to novice programmers. The findings also suggest that SL is a usable platform. Based on the findings of the study, I am confident that the proposed approach can deliver learning outcomes in curriculum delivery of introductory programming courses. The caveat of the findings is that prior experience of students in virtual world environments will potentially enhance the learning outcomes. Therefore, educators should assess informal use of SL (or other virtual world environments) by the students, before adopting SL for curriculum delivery.

I will disseminate the findings of this study in education journals and conferences, and recommend the adoption of the approach for teaching introductory programming at higher education institutions. Dissemination of findings will potentially lead to more effective pedagogies to communicate abstract concepts in contexts other than computer programming, and thus have even wider implications.

References

Adams, D. A., Nelson, R. R., and Todd, P. A. (1992). Perceived usefulness, ease of use and usage of information technology: a replication, MIS Quarterly, 16(2), 227-247.

Anderson, L. W., Krathwohl, D. R., and Bloom, B. S. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, Allyn and Bacon.

Antaki, C., Billig, M., Edwards, D., and Potter, J. (2003). Discourse analysis means doing analysis: A critique of six analytic shortcomings, retrieved 25 October 2015 from
https://dspace.lboro.ac.uk/2134/633

Avison, D. E., and Wood-Harper, A. T. (1991). Conclusions from action research: The Multiview experience. In *Systems thinking in Europe*, Springer US, 591-596.

Australian Computer Society (2015), Australia's Digital Pulse, Deloitte Access Economics, retrieved 15 October 2015 from
https://www.acs.org.au/content/dam/acs/acs-publications/02062015-Australias-Digital-Pulse-FINAL.PDF

Bazeley, P. (2013). Qualitative data analysis: Practical strategies. Sage.

Beaubouef, T. B. and Mason, J. (2005). Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. Inroads – SIGCSE Bulletin 37 (2).

Ben-Ari, M. (1998). Constructivism in computer science education. The Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, 257-261.

Ben-Bassat Levy, R., Ben-Ari, M., and Uronen, P. A. (2003). The Jeliot 2000 program animation system. Computers and Education, 40(1), 1-15.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., and Krathwohl, D. R. (1956). Taxonomy of educational objectives: Handbook I: Cognitive domain. New York: David McKay.

Boulos, M. N. K., Hetherington, L. and Wheeler, S. (2007). Second Life: an overview of the potential of 3-D virtual worlds in medical and health education, Health Information and Libraries Journal Vol. 24, No. 4, 233-245.

Buck, D. and Stucki, D. (2001). JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum. In: Proceedings of SIGCSE Technical Symposium on Computer Science Education, Charlotte NC, USA, ACM Press, 16–20.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M. (2005, February). RAPTOR: a visual programming environment for teaching algorithmic problem solving. In ACM SIGCSE Bulletin, Vol. 37, No. 1, 176-180.

Carmines, E. G., and Zeller, R. A. (1979). Reliability and validity assessment, Sage Publications.

Casamayor, A., Amandi, A. and Campo, M. (2009). Intelligent assistance for teachers in collaborative e-learning environments, Computers and Education, Vol. 53, Issue 4, 1147–1154.

Chau, P. Y. K. (1996). An Empirical Assessment of a Modified Technology Acceptance Model, Journal of Management Information Systems, Vol. 13, No. 2,185-204.

Chickering, A. W., and Gamson, Z. F. (1987). Seven Principles for Good Teaching in Undergraduate Education, American Association for Higher Education (AAHE) Bulletin 39, 3-7.

Coates, H. (2005). The value of student engagement for higher education quality assurance. Quality in Higher Education, 11(1), 25-36.

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. In Journal of Computing Sciences in Colleges, Vol. 15, No. 5, Consortium for Computing Sciences in Colleges.

Creswell, J. W. (2003). Research design: qualitative, quantitative, and mixed methods approaches (3rd Ed.). Thousand Oaks: Sage Publications.

Creswell, J. W. (2008). Educational research: planning, conducting, and evaluating quantitative and qualitative research (3rd Ed.). Upper Saddle River, N.J.: Pearson/Merrill Prentice Hall. Denzin and Lincoln, 2005

Cuban, L. (1983), How did teachers teach, 1890–1980, Theory into Practice, 22 (3), 160–165.

Dalgarno, B., Lee, M.J.W., Carlson, L, Gregory, S and 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., and Pausch, R. (2000, July). Making the connection: programming with animated small world. In ACM SIGCSE Bulletin, Vol. 32, No. 3, 41-44.

Davis, B. G. (2009). Tools for Teaching, Second Edition, Wiley Publishers.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly, Vol. 13(3), 319-340.

DeClue, T. (1996). Object-orientation and the principles of learning theory: A new look at problems and benefits. Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education.

Denzin, N. K., and Lincoln, Y. S. (2005). The Sage handbook of qualitative research. Sage Publications.

Department of Immigration & Border Protection (2015), Net Migration of ICT Workers into Australia, 25 April 2016, retrieved from
https://www.border.gov.au/about/reports-publications/research-statistics/statistics/

Dewey, J. (1916). Democracy and Education: An Introduction to the Philosophy of Education. New York, NY: Simon & Schuster, The Free Press.

Dickey, M. D. (2003). Teaching in 3D: Pedagogical affordances and constraints of 3D virtual worlds for synchronous distance learning. Distance education, Vol. 24, No. 1, 105-121.

Duderstadt, J., Atkins, D., and Houweling, D. (2002). Higher education in the digital age: Technology issues and strategies for American colleges and universities. Westport, CT: Praeger.

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses.

Dweck, C. S. (2000). Self-theories: Their role in motivation, personality, and development. Psychology Press.

Esteves, M., Fonseca, B., Morgado, L., and Martins, P. (2009). Using Second Life for Problem Based Learning in Computer Science Programming, Journal of Virtual Worlds Research, Vol. 2, No. 1, 3-25.

Fallows, S. and Chandramohan, B. (2001). Multiple approaches to assessment: reflections on use of tutor, peer and self-assessment. Teaching in Higher Education, Vol. 6, No. 2, 229-246.

Felix, U. (2005). E-learning pedagogy in the third millennium: the need for combining social and cognitive constructivist approaches. ReCALL, 17(01), 85-100.

Fishbein, M., and Ajzen, I. (1975). Belief, attitudes, intention, and behavior. An introduction to theory and research. Massachusetts: Addison-Wesley.

Fogg, B. J., Cuellar, G., and Danielson, D. (2009). Motivating, influencing, and persuading users: An introduction to captology. Human Computer Interaction Fundamentals, 109-122.

Fogg, B. J. (2003a). Persuasive technology: using computers to change what we think and do. Morgan Kaufmann Publishers.

Fogg, B. J. (2003b). Prominence-interpretation theory: Explaining how people assess credibility online. In CHI'03 extended abstracts on Human factors in computing systems, ACM, 722-723.

Gallagher, S. A. (1997). Problem-based learning: Where did it come from, what does it do, and where is it going?. Journal for the Education of the Gifted, 20(4), 332-362.

Gallagher, S. A., Stepien, W. J., and Rosenthal, H. (1992). The effects of problem-based learning on problem solving. Gifted Child Quarterly, 36(4), 195-200.

Gliddon, J. (2012). Australia's ICT graduate crisis, itNews, retrieved 12 September 2015 from http://www.itnews.com.au/news/australias-ict-graduate-crisis-322882

Grant, S and Clerehan, R (2011). Finding the discipline: Assessing student activity in Second Life. Australasian Journal of Educational Technology, 27(5), 813–828.

Guzdial, M., Kolodner, J. L., Hmelo, C., Narayanan, H., Carlson, D., Rappin, N., Hübscher, R., Turns, J., and Newstetter, W. (1996). Computer support for learning through complex problem-solving. Communications of the ACM, Vol. 39, No. 4, 43-45.

Girvan, C., Tangney, B., Savage, T (2013). SLurtles: Supporting constructionist learning in Second Life, Computers and Education, Vol. 61, 115–132.

Hadjerrouit, S (2008), Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study, Informatics in Education, 2008, Vol. 7, No. 2, 181–210

Hagan, D., and Markham, S. (2000). Teaching Java with the BlueJ environment. In Proceedings of Australasian Society for Computers in Learning in Tertiary Education Conference, retrieved 10 December 2015 from http://www.ascilite.org/conferences/coffs00/

Hall, R. (2008). Applied social research: Planning, designing and conducting real-world research. Macmillan Education.

Han, J., and Beheshti, M. (2010). Enhancement of computer science introductory courses with Mentored Pair Programming. Journal of Computing Sciences in Colleges, Vol. 25, No. 4, 149-155.

Hara, K. (1995). Quantitative and qualitative research approaches in education. Education, 115(3), 351.

Hein, G. (1991).The Museum and the Needs of People. CECA (International Committee of Museum Educators) Conference, 15-22 October, Jerusalem Israel, retrieved 10 March 2015 from http://www.exploratorium.edu/education/ifi/constructivist-learning

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

Henrïquez, L.M. (2001). Software visualization: An overview. Upgrade, Vol. 11, No. 2, 4-7.

Howard, G. (1983). Frames of mind: The theory of multiple intelligences. NY: Basics.

Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. Journal of Visual Languages and Computing, Vol. 13, No. 3, 259-290.

Hung, W., Jonassen, D. H., and Liu, R. (2008). Problem-based learning. Handbook of research on educational communications and technology, Vol. 3, 485-506.

Hwang, W-Y., Wang, C., Hwang, G-J., Huang, Y-M. and Huang, S. (2008). A web-based programming learning environment to support cognitive development. Interacting with Computers, Vol. 20, No. 6, 524-534.

IBM Rational Rose (2003). Visual Modeling with Rational Rose, retrieved 10 June 2013 from http://www.rational.com/products/rose/index.jsp

Jenkins, T. (2001). The motivation of students of programming. In Proceedings of ITiCSE 2001: The 6th annual conference on innovation and technology in computer science education, 53–56.

John-Steiner, V., and Mahn, H. (1996). Sociocultural approaches to learning and development: A Vygotskian framework. Educational psychologist, 31(3-4), 191-206.

Johnson, C. G., and Fuller, U. (2006). Is Bloom's taxonomy appropriate for computer science, In Proceedings of the 6th Baltic Sea conference on computing education research: Koli Calling, ACM, 120-123.

Journal of Virtual Worlds Research (2009). Pedagogy, Education and Innovation in Virtual Worlds.

Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S., and Gehringer, E. (2004, March). On understanding compatibility of student pair programmers. In ACM SIGCSE Bulletin, Vol. 36, No. 1, 7-11, ACM.

Kim, J., and Park, H. A. (2012). Development of a health information technology acceptance model using consumers' health behavior intention. Journal of medical Internet research, 14(5), retrieved 27 June 2017 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3510715/

Kimble, G. A. (1961). Conditioning and learning. Appleton-Century-Crofts Hilgard and Marquis', USA.

Kolb, D. A. (1984). Experiential learning: Experience as the source of learning and development. Prentice-Hall, New Jersey, USA.

Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. (2003). The BlueJ system and its pedagogy. Computer Science Education, 13(4), 249-268.

Kern, N. (2009). Starting a Second Life, retrieved 25 May 2012 from http://slexperiments.edublogs.org/2009/03/03/starting-a-second-life/

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview, Theory into Practice, 41:4, 212-218.

Kuh, G. D. (2003). What We're Learning About Student Engagement From NSSE: Benchmarks for Effective Educational Practices, Change: The Magazine of Higher Learning, 35:2, 24-32.

Lewin, K. (1939). Field theory and experiment in social psychology: Concepts and methods. American journal of sociology, 44(6), 868-896.

Lang, C., McKay, J., and Lewis, S. (2007). Seven Factors that Influence ICT Student Achievement. ACM SIGCSE Bulletin, 12th annual SIGCSE conference on Innovation and Technology in Computer Science Education 39 (3).

Linden Research Inc. (2008). Second Life Bot, retrieved 15 May 2012 from http://wiki.secondlife.com/wiki/Video_Tutorial/Record_voice_chat_and_sounds

Linden Research Inc. (2011). Second Life Bot, retrieved 15 May 2012 from http://wiki.secondlife.com/wiki/Bot

Linden Research Inc. (2013). Second Life Education, retrieved 15 May 2012 from http://wiki.secondlife.com/wiki/Second_Life_Education

Linden Research Inc. (2014). Streaming Video in Second Life, retrieved 15 May 2012 from http://wiki.secondlife.com/wiki/Streaming_Video_in_Second_Life

Lister, R. and Leaney, J. (2003). First year programming: let all the flowers bloom. Proceedings of the fifth Australasian conference on Computing education, Adelaide, Australia, 221-230.

Liu, C. H., and Matthews, R. (2005). Vygotsky's Philosophy: Constructivism and Its Criticisms Examined. International Education Journal, 6(3), 386-399.

Liu, C. C., and Tsai, C. C. (2008). An analysis of peer interaction patterns as discoursed by on-line small group problem-solving activity. Computers and Education, 50(3), 627-639.

Mackenzie, N., and Knipe, S. (2006). Research dilemmas: paradigms, methods and methodology. Issues in educational research, 16(2), 193-205.

Mathieson, K. (1991), Predicting use intentions: comparing the technology acceptance model with the theory of planned behaviour, Information Systems Research, 2(3), 173-191.

Maxwell, S. E. (2013), Methodological issues in planning and interpreting replication studies (Invited address), American Psychological Association, Honolulu, USA.

McCarthy, P. R., and McCarthy, H. M. (2006). When Case Studies Are Not Enough: Integrating Experiential Learning Into Business Curricula. Journal of Education for Business, 81(4), 201-204.

McKay, J., and Marshall, P. (2001). The dual imperatives of action research. Information Technology and People, 14(1), 46-59.

Menchaca, R., Balladares, L., Quintero, R., and Carreto, C. (2005). Software engineering, HCI techniques and Java technologies joined to develop web-based 3D-collaborative virtual environments. In Proceedings of the 2005 Latin American conference on Human-computer interaction, ACM, 40-51.

Mergel, B. (1998). Instructional design and learning theory, retrieved 25 November 2014 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.645.7122&rep=rep1&type=pdf

Merriam, S. B. (1998). Qualitative research and case study applications in education. Jossey-Bass, San Francisco, California, USA.

Migdalek, J (2002). Performing English: The classroom as rehearsal space. Prospect, 17(2), 53-61.

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

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

Nicholls, C. M., Mills, L., and Kotecha, M. (2014). Observation. In J. Ritchie, J. Lewis, C. M. Nicholls and O. Rachel (Eds.), Qualitative research practice: a guide for social science students and researchers, 2nd ed., Thousand Oaks: Sage Publications, 243-268.

Norman K. Denzin, and Yvonna S. Lincoln. (2005). The Sage handbook of qualitative research. Sage. Hall.

Novak, T. P. (2010). eLab City: A Platform for Academic Research on Virtual Worlds, Journal of Virtual Worlds Research, Vol. 3, No. 1, 3-33.

O’Grady, Michael J (2012). Practical problem-based learning in computing education. ACM Transactions on Computing Education (TOCE) 12.3: 10.

Oinas-Kukkonen, H. (2013). A foundation for the study of behavior change support systems. Personal and ubiquitous computing, Vol. 17, No. 6, 1223-1235.

Oinas-Kukkonen, H., and Harjumaa, M. (2008). A systematic framework for designing and evaluating persuasive systems. Persuasive Technology, Springer Berlin Heidelberg,164-176.

O'Toole, J., and Beckett, D. (2010). Educational research: creative thinking and doing. South Melbourne, Victoria: Oxford University Press.

Patton, M. Q. (2002). Qualitative research and evaluation methods (3rd ed). Thousand Oaks: Sage Publications Oaks: Sage Publications.

Pear, J. J., and Crone-Todd, D. E. (2002). A social constructivist approach to computer-mediated instruction. Computers and Education, Vol. 38, No. 1, 221-231.

Pellas, N., and Peroutseas, E. (2016). Gaming in Second Life via Scratch4SL: Engaging high school students in programming courses. Journal of Educational Computing Research, 54(1), 108-143

Piaget, J. (1977). The role of action in the development of thinking. In Knowledge and development, Springer USA, 17-42.

Quarmby, B. (2008). Pirates Among the Second Life Islands – Why You Should Monitor the Misuse of Your Intellectual Property in Online Virtual Worlds, ExpressO, retrieved 25 May 2012 from http://works.bepress.com/ben_quarmby/2

Relan, A., and Gillani, B. B. (1997). Web-based instruction and the traditional classroom: Similarities and differences. Khan, Vol. 62, 41-46.

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer science education, Vol. 13, No. 2, 137-172.

Robinson, C. C. and Hullinger, H. (2008). New Benchmarks in Higher Education: Student Engagement in Online Learning, Journal of Education for Business, Vol. 84, No. , 101-109.

Romero, P., Cox, R., Du Boulay, B., and Lutz, R. (2003). A survey of external representations employed in object-oriented programming environments. Journal of Visual Languages and Computing, Vol. 14, No. 5, 387-419.

Jennings, N. and Collins, C. (2007). Virtual or Virtually U: Educational Institutions in Second Life, International Journal of Human and Social Sciences, Vol. 2, No. 3, 180-186.

Rosenbaum, E. (2008). Scratch for Second Life. In Proceedings of the International Conference of the Learning Sciences-ICLS, 144-152.

Rößling, G., and Freisleben, B. (2002). ANIMAL: A system for supporting multiple roles in algorithm animation. Journal of Visual Languages and Computing, Vol. 13, No. 3, 341-354.

Sajjanhar, A. (2012). Virtual Worlds for Student Engagement, Creative Education, Vol. 3, Special Issue, 796-801.

Sajjanhar, A., and Faulkner, J. (2014). Exploring second life as a learning environment for computer programming, Creative Education, 5(1), 53-62.

Sandelowski, M. (2000). Focus on research methods combining qualitative and quantitative sampling, data collection, and analysis techniques. Research in nursing and health, Vol. 23, 246-255.

Schmidt, H. G., De Volder, M. L., De Grave, W. S., Moust, J. H. C., and Patel, V. L. (1989). Explanatory models in the processing of science text: The role of prior knowledge activation through small-group discussion. Journal of Educational Psychology, Vol. 81, No. 4, 610–619.

Schwandt, T. A. (2007). The Sage dictionary of qualitative inquiry. Sage Publications.

Schwartz, B. (2000). Self-determination: The tyranny of freedom. American psychologist, Vol. 55, No. 1, 79.

Shamir, A., Zion, M., and Spector-Levi, O. (2008). Peer tutoring, meta-cognitive processes and multimedia problem-based learning: The effect of mediation training on critical thinking. Journal of Science Education and Technology, Vol. 17, 384-398.

Shin, N., Jonassen, D. H., and McGee, S. (2003). Predictors of well-structured and ill-structured problem solving in an astronomy simulation. Journal of research in science teaching, Vol. 40, No. 1, 6-33.

Sieber, J. E. (1998). Planning ethically responsible research. In L. Bickman and D. J. Rog (Eds.), Handbook of applied social research methods, Thousand Oaks: Sage Publications, 127-156.

Sloane, K. and Linn, M.C. (1988). Instructional conditions in Pascal programming classes. In R. Mayer, Ed. Teaching and Learning Computer Programming: Multiple Research Perspectives. Lawrence Erlbaum Associates, Hillsdale, N.J. 207-235.

Soh, L. K., Samal, A. and Nugent, G. (2007). An integrated framework for improved computer science education: Strategies, implementations, and results. Computer Science Education, 17(1), 59-83.

Spires, H. A., and Donley, J. (1998). Prior knowledge activation: Inducing engagement with informational texts. Journal of Educational Psychology, Vol. 90, No. 2, 249–260.

Stake, R. E. (2010). Qualitative research: Studying how things work. Guilford Press.

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

Stasko, J., Domingue, J., Brown, M. H., and Price, B. A. (1998). Software Visualization: Programming as a Multimedia Experience, MIT Press, Cambridge, USA.

Stasko, J. T. (1990). Tango: A framework and system for algorithm animation. Computer, Vol. 23, No. 9, 27-39.

Street, C. T., and Meister, D. B. (2004). Small business growth and internal transparency: The role of information systems. MIS quarterly, Vol. 28, No. 3, 473-506.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. Journal of Educational Computing Research, Vol. 36, No. 2, 223-244.

Holmes, B., Tangney, B., FitzGibbon, A., Savage, T., and Mehan, S. (2001). Communal Constructivism: Students constructing learning for as well as with others. Technology and Teacher Education Annual, Vol. 3, 3114-3119.

Teague, D. (2011). Pedagogy of Introductory Programming: A People-First Approach, MIT (Research) Thesis, Queensland University of Technology.

Thorne, S. (2008). Interpretive description. Walnut Creek, CA: Left Coast Press.

University of North Carolina (2010). Second Life in Education, retrieved 11 March 2012 from https://www.youtube.com/user/UNCChapelHill/

Van de Ven, A. H., and Poole, M. S. (1995). Explaining development and change in organizations. Academy of management review, Vol. 20, No. 3, 510-540.

Van Gorp, M. J., and Grisson, S. (2001). An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming" Computer Science Education, Vol. 11, No. 3, 247-260.

Venkatesh, V., & Davis, F. D. (2000). A theoretical extension of the technology acceptance model: Four longitudinal field studies. Management science, 46(2), 186-204.

Vrasidas, C. (2000). Constructivism versus objectivism: Implications for interaction, course design, and evaluation in distance education. International Journal of Educational Telecommunications, Vol. 6, No. 4, 339-362.

Vygotsky, L. (1978). Mind in Society: The Development of Higher Psychological Processes. Translated by Michael Cole. Harvard University Press, Boston, USA.

Vygotsky, L. S. (1980). Mind in society: The development of higher psychological processes. Harvard University Press, Massachusetts, USA.

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, Vol. 40, 414–426.

Warden, C. A., Stanworth, J. O., Ren, J. B., and Warden, A. R. (2013). Synchronous learning best practices: An action research study. Computers and Education, Vol. 63, 197-207.

Wood, D. J., Bruner, J. S., and Ross, G. (1976). The role of tutoring in problem solving. Journal of Child Psychiatry and Psychology, Vol. 17, No. 2, 89-100.

Wrzesien, M., and Raya, M.A. (2010). Learning in serious virtual worlds: evaluation of learning effectiveness and appeal to students in the E-Junior project. Computers and Education, Vol. 55, No.1, 178–187.

Wyse, S. E. (2011). What is the difference between qualitative research and quantitative research? Retrieved 15 April 2015 from
http://www.snapsurveys.com/blog/what-is-the-difference-between-qualitative-research-and-quantitative-research/

Yin, R. K. (1984). Applied social research methods series Case study research: Design and methods. Thousand Oaks: Sage Publications.

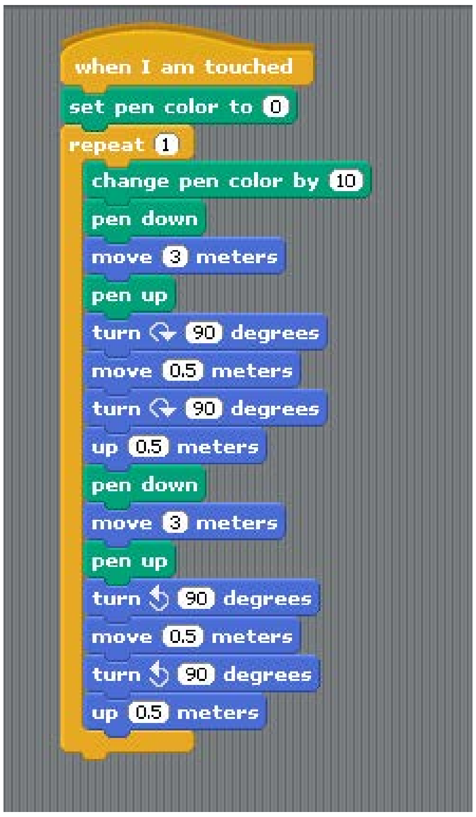
Yin, R. K. (2009). Case study research: design and methods (4th Ed.). Thousand Oaks: Sage Publications.

Yin, R. K. (2011). Applications of case study research. Sage Publications.

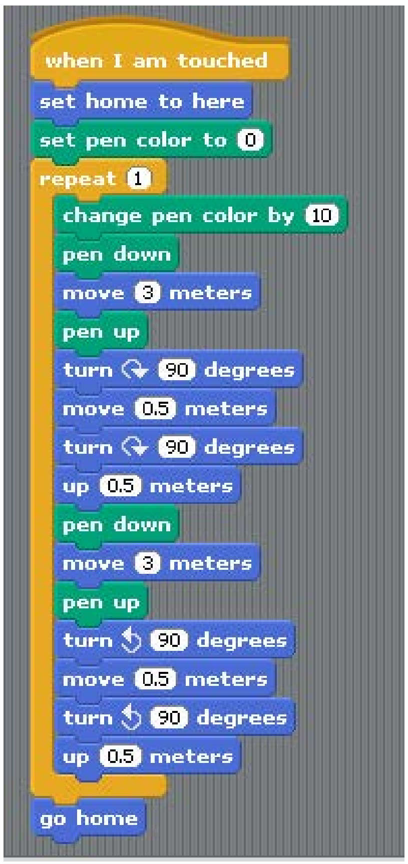
Yorke, M., and Knight, P. (2004). Self-theories: some implications for teaching and learning in higher education. Studies in Higher Education, Vol. 29, No. 1, 25-37.

Activity 1

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



```
when I am touched
set pen color to 0
repeat 1
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move 0.5 meters
    turn ↻ 90 degrees
    up 0.5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move 0.5 meters
    turn ↺ 90 degrees
    up 0.5 meters
```



```
when I am touched
set home to here
set pen color to 0
repeat 1
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move 0.5 meters
    turn ↻ 90 degrees
    up 0.5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move 0.5 meters
    turn ↺ 90 degrees
    up 0.5 meters
go home
```

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

1. Examine the code and identify the difference in the code.

____

____

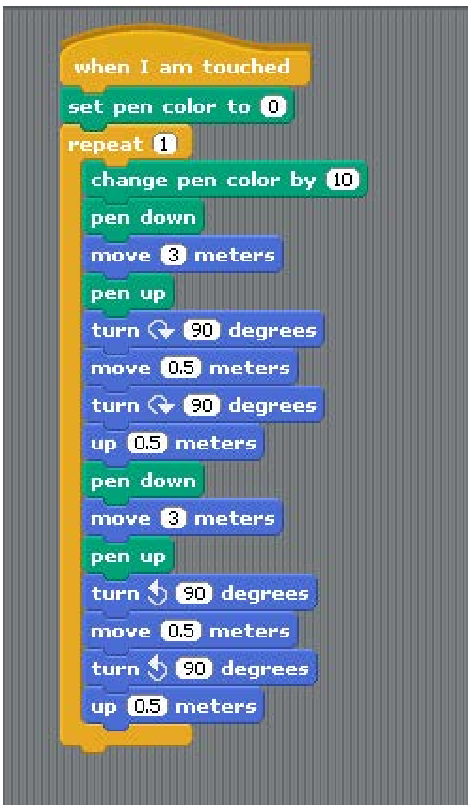
2. How is the difference in the code, reflected in the visualization?

____

____

Activity 2

S4SL code in object Activity 2-Object A is shown below.



```
when I am touched
set pen color to 0
repeat 1
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move 0.5 meters
    turn ↻ 90 degrees
    up 0.5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move 0.5 meters
    turn ↺ 90 degrees
    up 0.5 meters
```

Use object Activity 2-Object A from the *Second Life Inventory* to visualize the piece of code, and answer the questions below.

1. Modify the code so that the object creates three pairs of stairs.

____

____

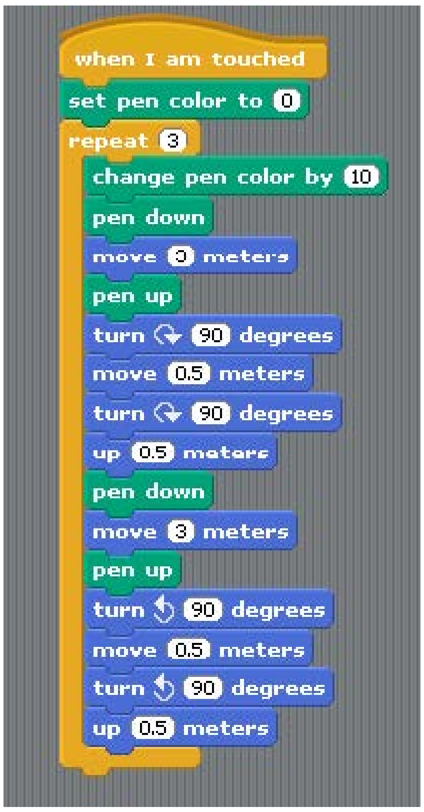
2. How is the difference between the original and modified code reflected in the visualization?

____

____

Activity 3

S4SL code in object Activity 3-Object A is shown below.



```
when I am touched
set pen color to 0
repeat 3
    change pen color by 10
    pen down
    move 3 meters
    pen up
    turn ↻ 90 degrees
    move 0.5 meters
    turn ↻ 90 degrees
    up 0.5 meters
    pen down
    move 3 meters
    pen up
    turn ↺ 90 degrees
    move 0.5 meters
    turn ↺ 90 degrees
    up 0.5 meters
```

Use object Activity 3-Object A from the *Second Life Inventory* to visualize the piece of code and answer the questions below.

1. Modify the code so that the object returns to the position at the end of the second pair of stairs.

____

____

2. How is the difference between the original and modified code reflected in the visualization?

____

____

1. Perceived Ease Of Use					
Answer the following questions in respect to SL/S4SL programming tasks including: • tasks demonstrated by the researcher; • tasks attempted by you.	SA	A	N	D	SD
1. *Easy to learn* SL/S4SL provides an easy to learn approach for programming concepts	○	○	○	○	○
2. *Clear and understandable* SL/S4SL tasks makes the programming concepts clear and understandable	○	○	○	○	○
3. *Easy to become skillful* It is easy to become skillful in programming tasks in SL/S4SL	○	○	○	○	○
4. *Easy to use* SL/S4SL is an easy to use environment for attempting programming tasks	○	○	○	○	○
5. *Controllable* SL/S4SL is a controllable environment for attempting programming tasks	○	○	○	○	○
6. *Flexible* SL/S4SL provides a flexible environment for learning programming concepts	○	○	○	○	○

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

2. Perceived Usefulness					
Answer the following questions in respect to SL/S4SL programming tasks: • tasks demonstrated by the researcher; • tasks attempted by you.	SA	A	N	D	SD
1. Work more quickly I can perform the programming tasks quickly in SL/S4SL	○	○	○	○	○
2. Increased Productivity SL/S4SL improves my learning efficiency of programming concepts	○	○	○	○	○
3. Effectiveness SL/S4SL is effective for learning programming concepts	○	○	○	○	○
4. Makes job easier SL/S4SL makes it easier to learn programming concepts compared with other environments	○	○	○	○	○
5. Useful SL/S4SL is useful for learning programming concepts	○	○	○	○	○

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

3. Open-ended questions	
1. Did you find the tasks interesting?	
2. Do you expect the virtual presence of the teacher to be helpful for completing activities in this environment?	
3. Do you expect the virtual presence of the peers to be helpful for completing activities in this environment?	
4. Any other comments	

PLAIN LANGUAGE STATEMENT AND CONSENT FORM

TO: Participant

Plain Language Statement

Date:

Full Project Title: Exploring Second Life as a Learning Environment for Computer Programming

Principal Researcher: Dr Atul Sajjanhar

You are invited to take part in this research project. Participation in any research project is voluntary. If you do not wish to take part you are not obliged to. Deciding not to participate will not affect your relationship to the researchers or to Deakin University. Once you have read this form and agree to participate, please sign the attached consent form. You may keep this copy of the Plain Language Statement.

The purpose of this research is to investigate the effectiveness of Second Life (SL) as a learning environment for computer programming concepts. The collated views of academics and students will be published in peer-reviewed conference/journal articles. No individual will be able to be identified in any publication.

If you agree to take part in the project, you should fill out a questionnaire. The questionnaire asks about your opinion of SL as a learning environment for computer programming concepts. There are also some general questions about you, to help interpret the information that you give. The questionnaire would normally take up to 45 minutes of your time to complete. You are encouraged to participate in this study during your own time. All student participants will be given \$30 Kmart® gift vouchers.

Approval to undertake this research project has been given by the Human Ethics Advisory Group (HEAG), Faculty of Science, Engineering and Built Environment, Deakin University. If you have any complaints about any aspect of the project, the way it is being conducted or any questions about your rights as a research participant, then you may contact: The Manager, Office of Research Integrity, Deakin University, 221 Burwood Highway, Burwood Victoria 3125, Telephone: 9251 7129, Facsimile: 9244 6581; research-ethics@deakin.edu.au. Please quote Reference Number STEC-8-2013-SAJJANHAR

If you require further information or if you have any problems concerning this project, you can contact principal researcher. The researcher responsible for this project is: Dr Atul Sajjanhar, Email: atuls@deakin.edu.au; [redacted]

PLAIN LANGUAGE STATEMENT AND CONSENT FORM

TO: ***Participant***

Consent Form

Date:

Full Project Title: Exploring Second Life as a Learning Environment for Computer Programming

Reference Number: STEC-8-2013-SAJJANHAR

I have read and I understand the attached Plain Language Statement.

I freely agree to participate in this project according to the conditions in the Plain Language Statement.

I have been given a copy of the Plain Language Statement and Consent Form to keep.

The researcher has agreed not to reveal my identity and personal details, including where information about this project is published, or presented in any public form.

Participant's Name (printed) ……………………………………………………………………

Signature ………………………………………………………… Date ……………………………

Please mail or fax this form to:

Dr Atul Sajjanhar
School of Information Technology,
Deakin University, Burwood, VIC 3125
[redacted]

PLAIN LANGUAGE STATEMENT AND CONSENT FORM

TO: Participant

Withdrawal of Consent Form

(To be used for participants who wish to withdraw from the project)

Date:

Full Project Title: Exploring Second Life as a Learning Environment for Computer Programming

Reference Number: STEC-8-2013-SAJJANHAR

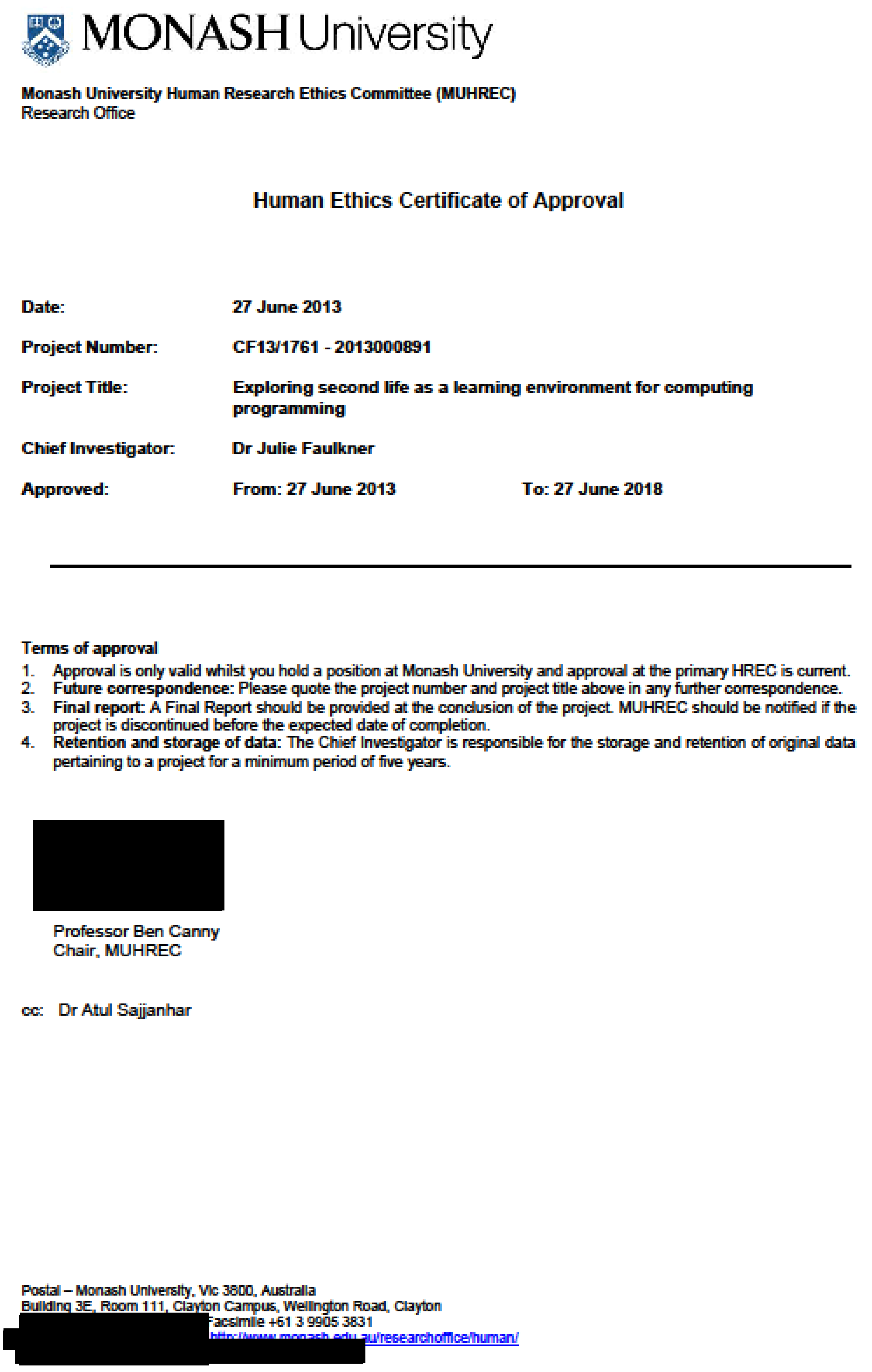
I hereby wish to WITHDRAW my consent to participate in the above research project and understand that such withdrawal WILL NOT jeopardise my relationship with Deakin University.

Participant's Name (printed) …………………………………………………….

Signature ………………………………………………………………. Date ……………………

Please mail or fax this form to:

Dr Atul Sajjanhar
School of Information Technology,
Deakin University, Burwood, VIC 3125
[redacted]



MONASH University

Monash University Human Research Ethics Committee (MUHREC)
Research Office

Human Ethics Certificate of Approval

Date:	**27 June 2013**
Project Number:	**CF13/1761 - 2013000891**
Project Title:	**Exploring second life as a learning environment for computing programming**
Chief Investigator:	**Dr Julie Faulkner**
Approved:	**From: 27 June 2013** **To: 27 June 2018**

Terms of approval

1. Approval is only valid whilst you hold a position at Monash University and approval at the primary HREC is current.
2. **Future correspondence:** Please quote the project number and project title above in any further correspondence.
3. **Final report:** A Final Report should be provided at the conclusion of the project. MUHREC should be notified if the project is discontinued before the expected date of completion.
4. **Retention and storage of data:** The Chief Investigator is responsible for the storage and retention of original data pertaining to a project for a minimum period of five years.

[redacted]

Professor Ben Canny
Chair, MUHREC

cc: Dr Atul Sajjanhar

Postal – Monash University, Vic 3800, Australia
Building 3E, Room 111, Clayton Campus, Wellington Road, Clayton
[redacted] Facsimile +61 3 9905 3831
http://www.monash.edu.au/researchoffice/human/