# Computer Science Education in Virtual Worlds

November 5, 2009

**Benjamin Kearns**
bmk33@student.canterbury.ac.nz
**Department of Computer Science and Software Engineering**
**University of Canterbury, Christchurch, New Zealand**

**Supervisor: Dr Tim Bell**
tim.bell@canterbury.ac.nz

## Abstract

The Computer Science Unplugged project has developed a variety of activities that allow children to be taught computer science concepts without using computers. Currently users in some environments and users with disabilities are not able to participate in some of the activities. In order to solve this problem, we investigate using the activities implemented in a virtual world. In this research, we discuss development of an implementation of the Turing test activity, the practical issues we encountered while deploying it to a local school and the general lessons to be learned for people deploying such activities. Finally we compare its effectiveness against the traditional Computer Science Unplugged activities.

# Contents

# 1 Introduction

The number of new students enrolling in computer science degrees is at a critical low[3]. Currently there are very few high school students who know what exactly a computer scientist is, and what sort of tasks computer scientists perform. Most school level computer science classes focus on teaching programming and give the wrong impression to students. The common misconception is that the typical computer scientist is a male who sits down programming in front of a computer all day, every day with little to no interaction with other people [20]. This is far from the truth, with many jobs not even requiring programming skills.

The Computer Science Unplugged project aims to help address this misconception by developing activities which can be taken in to schools, in which the children have fun and will learn what computer science is and what computer scientists do, all without ever touching a computer [2, 1]. In order to encourage more students to study it at University, the project also helps to increase general awareness of computer science as a career. One example of the kinds of activities performed is an activity where there are "islands" which the children must run between, where at each island they get given a choice of two other islands to which they can travel to from that island, and they must try and make it from the starting island to treasure island. This activity is designed to teach students about state machines, in a fun, non-technical, way.

While these sorts of activities do work well [11], in some environments they also have certain limitations. Firstly, students with disabilities find many of the activities either difficult or impossible, and secondly many schools are not fortunate enough to have as much space as is needed to perform these activities, especially in populated cities. For example in Japan some schools are in high rise buildings and have no room outside for the students.

The goal of this research is to explore development of these activities inside a virtual environment, where these limitations can be overcome. Additionally we also develop activities in a virtual world, which can overcome some of the limitations of the real world in order to teach the concepts better.

We have split the research up in to two distinct components, first there is the implementation and evaluation of an activity in a virtual environment. Secondly, more of a practical issue, is how to deploy the activities to schools so that they will work in as many environments as possible. This is important as many schools have computer systems which are very locked down and restricted and in order for us to have the project used as much as possible, we must attempt to mitigate these issues so that the system can be used with the least amount of effort possible.

# 2 The Turing Test Activities

As stated above, this work builds heavily on the Computer Science Unplugged project [2, 1], and investigates the effectiveness of teaching computer science concepts in virtual worlds compared to teaching the same concepts off-line, without the involvement with computers.

The goals of the Computer Science Unplugged project are[17]:

1. Increase interest in computer science.
2. Increase students' perception that computer science is a challenging and intellectually stimulating field.
3. Show students that there is a difference between computer science and programming.
4. Increase the number of females who choose to study computer science.

While it is possible to teach young students programming[7], the Computer Science Unplugged project does not have any activities which require programming or even teach programming concepts.

As a case study, the topic which we decided to teach the students about was the Turing test[18]. The Turing test is a competition which is run, in which a number of judges sit down in front of a text only interface and have conversations with either a computer or a real person and have to try and tell which they are talking to. If a machine can carry out an intelligent conversation with a real person, and that person can not tell that it is not a real person, then that machine is said to have passed the Turing test. This topic gets students thinking about artificial intelligence and the issues surrounding it and we decided to implement this activity in a virtual world as we thought it would allow us to demonstrate the concept in an easier and more natural way than on paper.

In the original Computer Science Unplugged activity [1] the students have a fixed set of questions (see Appendix A) which they are able to ask and one child plays the role of a computer and has a fixed set of responses to those questions and another person is allowed to answer free-form (and therefore plays the role of a person). These two people are placed in a separate room, and two runners run backwards and forwards to those people, asking them the appropriate questions and replying with their responses. The students must then try and figure out which of them has the script (and is pretending to be the computer), and which of them is answering naturally. When performing the activity at a school, we made a slight variation on the original activity and only had one runner, so that as many students as possible could involved with deciding which questions would be best to ask (which is at the heart of the activity.)

In the online version of the activity we had a number of students enter a virtual world. The students were placed on an island and were placed in one of three main social areas. We also put a number of characters that are controlled by an artificial intelligence engine into the world, in these areas. The students then had to walk around and try and write down the names of the computer controlled characters. Practically we had to have the computer controlled characters stand still, and we also had to ask some of the students to stand still, as implementing a realistic "movement" engine is a very difficult problem[9] and was out of the scope of this research. Incorrect movement would be an instant give away that an avatar is not real. Further details of the implementation are given in the next section.

Before we started each activity (both online and offline), we started by asking some general questions to the students, and asked them to raise their hands if they thought computers were intelligent, and then prompting for answers why. We then followed this by "How can well tell if computers are intelligent?". After the activity, we asked students to raise their hands if they still thought computers were intelligent, and gave them a survey to fill out. We finally would explain to them, in simple terms, that computer scientists and philosophers are still debating whether or not computers will ever be truly intelligent.

# 3 The Online Activity

## 3.1 Technical background

One option we considered was to implement the activity in SecondLife as it is a well-known and free virtual world which is able to be downloaded from Linden Labs, and there is currently much research being conducted in to using SecondLife for educational purposes[4, 22, 21]. There has also been work on teaching computer science concepts in SecondLife, such as programming languages[6] and state machines[13]. It allows the content to be aimed at different levels of experience, distance learning[16], face to face style interaction, and interactive visual demonstrations which are not possible in a traditional setting[8]. There is also work being done on integrating virtual environments with traditional learning management systems. One such project is Sloodle[12], which integrates Moodle and SecondLife. Systems such as this, allow courses to be taught using a mixture of a virtual world and an online course management system based in a web browser. The integration includes features such automatic synchronisation of lecture notes between SecondLife and Moodle, synchronised blogging, synchronised polls and quizes and a 3D course-work drop box. There has been some work on people with disabilities using SecondLife, although the previous work work mostly focuses on teaching social skills or skills relating to how to cope with the real world [5, 19] to disabled students, while this work makes previously inaccessible activities accessible to users with disabilities.

There are, however, a number of issues associated with using SecondLife, the most major one being that people under the age of 18 must remain in a separate, isolated area, and adults who enter there must be educators, have police checks done on them, and must be in the United States. Additionally, people under the age of 13 are not allowed in the area at all, meaning this approach would be limited to high schools only. Some parents might also be uncomfortable allowing their children in to an "unrestricted" virtual world on the Internet. These factors completely rule out using SecondLife as an option for students under eighteen in countries other than the United States.

Another project, OpenSIM releases an open source version of the SecondLife server, which can be run on either a server on the school's network, or an external server on the Internet. If OpenSIM is run on a server by the school then it allows the entire environment to be recreated inside a school's network, without any connection to the Internet needed, thereby avoiding all of the associated risks with unknown people being in the environment and firewall issues. The standard SecondLife client can be used to connect to OpenSIM servers, and a number of additional clients have been created based on the official SecondLife client's source code which have various additional features.

The first half of this research was dedicated to creating a Computer Science Unplugged activity in the OpenSIM environment. In the second half of this research we looked in to the practical issues of deploying this sort of system to intermediate and high schools.

## 3.2 Installation and Deployment

There are a number of problems present to deploying a system like this to schools. This section first discusses the advantages of running it in an off-line environment. We then discuss the specific issues which we had when deploying the system to a school, which was a local Intermediate school containing students between eleven and fourteen years of age in Christchurch, New Zealand. Finally, this section goes in to additional problems which may be encountered at other schools.

Almost all schools have some sort of firewall and Internet filtering system in place[10]; it would be unacceptable for any large school which can not closely monitor all access to not have a firewall and filtering system in place. This, however, causes issues when trying to deploy and use new applications on the network. This means that if we were to develop the activities in Linden Lab's SecondLife grid, then some schools would have difficulties connecting to the main SecondLife grid because the firewall would not let them out. The firewall at the school we tested it at was blocking outgoing connections to specific ports and not blocking based on IP addresses. This is typical of schools. For large schools with a full time system administrator, this can be worked around more easily, providing there are no policies preventing it, however many smaller schools do not have dedicated system administrators and instead outsource the job meaning that it may take some time for exceptions to the firewall rules to be set up. Additionally many schools use their ISP to do filtering and firewalling of their Internet connection creating further administrative issues. With one survey[10] reporting that only 10% of primary schools, and 23% of high schools manage their own firewalls, this is something which needs to be thought about. Practically there is very little risk in opening up these ports on the firewall, however some administrators may not see the merits in opening them up and may have defined policies which prevent it.

A further problem with connecting to the public SecondLife grid is that police checks need to be run on any person wanting to enter the under 18 area. This police check can only be performed in the United States, making it very impractical for schools or Universities outside the United States to set up an area there.

The other public grid which we could deploy activities to would be the Ongens OpenSIM [15] grid, which is primarily run by Otago University, with assistance from the University of Canterbury, the University of Auckland and Telecom New Zealand. The Ongens grid is primary run for research and education purposes, so it makes sense to consider this as an option, however this option is still subject to firewall issues as well as parents and teachers being anxious about letting children go in to a public environment online. However, if the firewall issues are worked around, and policies and monitoring are put in place to ensure the students' safety online, then this would still be a viable option, as it does not require any technical knowledge for teachers other than installing the SecondLife client. This option would also allow groups of students from different schools to participate in the activities together. We have not investigated this option any further as the environment in which we were testing had an external firewall which blocked most outgoing traffic.

One way to address the issues is to deploy our own isolated OpenSIM environment to schools, although this has many issues as well. Irrespective of how easy the installation process is made, teachers will still need to set up and maintain their own OpenSIM environment which is not going to be as easy as logging in to a system which is already set up. However, the biggest issue we encountered when deploying OpenSIM ourselves was that OpenSIM is still very much under development, and as such has a number of bugs. The developers also seem to put a focus on getting the software working under Linux and put less of an emphasis on Windows. This makes sense as they are mostly targeting system administrators who want to run OpenSIM grids on the Internet or within a large network, and not a smaller network or a school. An example which demonstrates both the developmental nature and the emphasis the developers put on Linux is there was on bug which kept recurring where under Windows the server would not send down the models and skins for avatars, and as such they would appear as white glowing blobs. It looks abnormal and is not very practical to have a conversation between two glowing white blobs in a virtual world. Strangely enough, this issue either disappeared or occurred a lot less frequently when the OpenSIM server was run under Linux using the MySQL database backend. In principle this should not happen, as it is written in C# and the same executable is run under both Linux and Windows.

This bug alone forced us to run our experiments using a laptop running the OpenSIM server under Linux, using the MySQL database as a backend. Unfortunately this is not an option for most schools as teachers would not have the experience or skills to be able to set up and configure a full database engine or Linux. This, however, may be possible at some schools which have system administrators which do have those skills, but a large amount of the target audience of these activities are intermediate schools, which are typically smaller and will not have dedicated IT staff. One potential solution to this problem would be to preconfigure a Live CD environment which teachers can "plug and play" on their laptops. This will

save the teachers from having to set up Linux or a database engine on their computers and would work for setting up the server in most cases. There would however be some configurations and hardware where this is unlikely to work, although the number of these cases will be fairly small. Hopefully as OpenSIM is developed further, it will be possible to provide an easy installer which can be installed on a Windows machine which does not have these bugs, and does not require a separate database engine.

Another issue which we had was that in order to specify which server to use, command line options must be passed to the SecondLife viewer executable. It is undesirable to ask students to open up a command line and pass in these command line parameters. Firstly it is tedious to guide students through this process as many students will need individual help, and secondly, some schools may have locked down launching the command line or random executables. In order to combat this problem, we developed a small application which broadcasts packets on the network requesting the address of the server, and then launches the client, telling it to connect to the appropriate server.

This small application was developed in C#, which caused an unforeseen issue: the computers did not have the .NET runtime installed on them, causing the application to crash at start-up. The options to solve this solution were:

1. Rewrite the application in C++.
2. Recompile the application using the Mono project's[14] executable creator
3. Write the wrapper in a small interpreted language, and bundle the script and the executable for the interpreter.

Initially we tried the second option, as it seemed quick and easy, however as additional problems arose (see below,) the third option quickly became the best, as it allowed quick and rapid development of the scripts on the lab computer, rather than having to recompile the application after every change.

The first problem that we encountered when trying to run it in the school environment was that the computers were fairly locked down. One thing which did not occur to us however, is that by default on systems running Windows XP Service Pack 2, the firewall is enabled which blocks all unknown applications. The accounts were limited, non-administrator accounts, so it was not able to alert the user and allow them to unblock the application. This meant that the launcher application we had written was not able to receive the packets which it needed to figure out where the server was located. The only option for working around this issue was to go around every computer in the lab and disable the local firewall. Some larger schools may have central administration of this, which would make this task trivial. In practice schools should not need local firewalls enabled on each computer individually as they should have a firewall at the perimeter of the network anyway, however Microsoft does enable it by default on Windows XP Service Pack 2 and above, and it may help to prevent malware spreading in some cases.

These firewall issues, along with the lack of the .NET runtime on the systems forced us to move to consider other options for the launcher application. Rewriting the application in C++ would have been time consuming and error prone, so this option was ruled out. The "round trip" time for developing the application using Mono's executable maker was rather high, as it required us to compile the application twice in different environments. Also the application did not look native and did not feel very user friendly. The last option, the one we used, was to write a Ruby based launcher script. This allowed us to quickly test whether the packets were getting past the firewall, and make other small changes to the script, on the lab computer we were testing it on.

The next problem we had was how to distribute the client software to each of the computers. We considered three approaches to solve this problem:

1. Use flash drives.
2. Use a CD or DVD.
3. Run it from a network share.

While doing the initial tests, we ran the client software from a flash drive and this worked well, however buying 30 flash drives for a single class activity will be fairly expensive in some countries, so we decided to look in to other ways of distributing the software to the other computers.

One solution we tried was to run the SecondLife client from a CD and it became apparent fairly early on that this was impractical. The SecondLife client software requires a lot of random access to files, something which CDs can not do fast, as it is loading the software. When testing the software, we stopped it loading after 10 minutes of waiting.

The final method we tried was to run it off a shared network drive. This also failed as the Windows XP console does not support running software from a network, preventing the launcher script from running, effectively preventing this option.

The end solution to this problem was to go around each computer in the room and install the software on to the hard drive. This was a very tedious process, but it was the only practical way of doing it. In the future, if some schools adopted to run the Computer Science Unplugged online activities as regular events, then they would be able to leave the software permanently installed on the computers, and build it in to the install image they use for setting up the computers.

Other schools may have automated software which allows software installation similar to this in an automated way, effectively making this the most practical option, and other versions of Windows may not have the same problems with running command line software from a network, and Unix based systems (such as Mac OS and Linux) certainly do not have this issue. Although if the software were to be run from a network, the network speed and hard drive access speed of the server may become issues. Further testing would be needed.

Once we got a classroom full of students, we found that the computer we were using for the server, which was our own laptop, was significantly underpowered to run both the OpenSIM server and the bots. While we had enough processing power for the 5-10 person testing which we did, when we had around 40 people (including bots), the system running the server ran out of RAM and started using swap space, effectively preventing the system from working and it was also randomly dropping client connections. Our laptop was fairly low end (2GHz with 1.5GB of RAM). To combat this problem, instead we took in a computer with much higher resources (Quad Core 2.4GHz, with 7GB of RAM). This was sufficient to run the system. It turns out this issue was because the bots have really high system requirements (between 150MB and 200MB of RAM each). The bots themselves are fairly simple code-wise, so it must have been either due to one of the libraries they utilise, or the Mono Runtime (which is the .NET runtime for Linux) must have a really large overhead. Further investigation would be needed to determine the cause of this.

## 3.3 Development

We developed two activities in the virtual world, however we only ended up deploying one, for the reasons given below. During development, we encountered a number of difficulties. The first difficulty we encountered was that in order to make an engaging and high quality activity for children, moderate level skills are required in scripting and modelling, and while the author is proficient with programming, the graphics and 3D design work were a significant hurdle.

We initially started developing an online version of the state machine activity[1]. In the original activity, students ran to a person representing an island where they would ask for path A or B and the person representing the island would tell them where to go to next. From this they would build up a map of possible paths until eventually they got to the treasure island and finished the game. In the online version of the activity, a number of ideas were raised.

The first idea was to have the students swim between islands. However, without adequate modelling skills these would have to simply be islands with sticks on them, saying the island's name. The second idea was to simply have sticks coming out of the ground. Both of these had two issues, firstly, if we wanted to enforce the route the students took then a significant amount of fairly complex scripting would have been involved. Secondly, and most importantly, sticks in the ground are not particularly engaging.

Another idea was to have a house the students could walk through which was modelled on a state


Figure 3.1: The original state machine house.

machine, where each room would represent a node, and there would be two doors or gateways in each room, going to adjacent nodes. The biggest problem with this approach was that even as we were developing it, we were feeling fairly claustrophobic when walking around, and we assume that if we were feeling uncomfortable then some of the students would too. Figure 3.1 shows a screen shot of this house.

To mitigate this issue, we could have developed the house instead as an outdoor maze with hedges, however this has further issues. The biggest one being that the students would fly for the pure novelty of flying, and the moment the look down they would see the layout of the maze defeating the purpose of the activity.

Due to these issues, we stopped work on developing the state machine activity and instead decided to work on developing an activity to demonstrate the Turing test.

Implementing an activity to demonstrate the Turing test in a virtual world was a bit more exciting than the idea of implementing an activity to demonstrate state machines, as it allowed us to do something which is not possible off-line: have natural conversations with real bots. As stated above, the idea of the activity was to have a group of students enter the virtual world and a group of bots, and the students have to identify who the bots are and who the real people are. This is a much more realistic demonstration of the Turing test than having people running back and forward with pieces of paper. In order to overcome the limitations with having to model the world and creating a more engaging environment for the children, we instead downloaded a free island called Educasim, which was designed for children in educational settings. This island can be downloaded from http://www.rexxed.com/2009/06/educasim-a-world-for-educational-use/. The island had four main centres on it, and we had the students and bots spawn at one of three of these. One was a house, and two were stage-like places with large domes and seats. This island, with some of the bots and students' avatars present, can be seen in figure 3.2.

In order to minimise the additional factors which would allow the students to recognise whether the avatar they were communicating with was a bot, we had to impose a few additional restrictions upon the environment. Firstly, as stated above, we could not implement realistic movement of the bots, so we had to ask some students to stand still. Secondly, we had to randomly generate names for both the students and the bots. This task was performed by the start up script mentioned earlier. Practically we thought that some

Figure 3.2: A screen shot of the experiment.

girls having male names and some of the boys having female names might have been an issue, but it turned out that the students did not care at all.

# 4 Results of the Activities

In this section, we will discuss the results and outcomes of the experiment. First we will discuss the set up of the experiment, then we will explain our observations while running the experiments, and thirdly we discuss outcomes of the survey.

## 4.1 Experiment

The experiment was run across four classes at a local intermediate school. All participants were in year 8, meaning they ranged from twelve to fourteen years of age, and were in average streamed classes, meaning they were of average intelligence for their age. We had two classes which were asked to perform the original Computer Science Unplugged activity, and two which were asked to participate in the activity in the virtual world. In total, we had 49 students participate in the original activity and 50 students participate in the virtual world. As stated above, the server was run on a system with a Quad Core 2.4GHz and 7GB of RAM, and the systems the students used were fairly underpowered school systems, with 256MB of RAM and 1GHz processors.

## 4.2 Observations

The first class we had did the original activity, they were very well behaved, almost everyone participated in the activity and overall appeared to be learning a lot.

The second class also did the original activity. They were not as well behaved, very few of the students were on task, with many running around and not at their desks. Overall the activity appeared to be very unproductive. At one point we had four runners for one team, when we were running it with just one. We suspect two factors caused this: they had a reliever for that day and their PE time was cancelled so they could participate in the experiment.

The third class was in the computer lab, again they seemed less well behaved, with many running and flying around the virtual world (see below). Again, they did not seem very focused on the activity. The activity for this class was being run on the last block for the day, and was also after lunch. Unexpectedly, there were some students who had a large amount of difficulty using the SecondLife client's interface.

The final class was again in the computer lab, and was fairly well behaved, although still not as well behaved as the first class in the classroom. They seemed to be learning more than the first class in the computer room but the results show that they also did not learn significantly more. Like the other class, there was also a lot of confusion about how to use the interface.

Both classes in the computer labs seemed to enjoy themselves a lot more than the classes in the classrooms and the data supports this observation (see below). The students in the computer labs however were however a lot more distracted and at one point all of the students in the first class decided they would fly up and sit on the roof of the house. In both classes, there were also students who modified the terrain and were building new objects and structures. In figure 4.1 you can see hills the students created in the landscape. All of the students in both classes very quickly discovered that the easiest way to identify if someone was a bot was simply to ask them if they are a computer, as the bots would respond "Would being a computer matter to you if I were metal instead of flesh?", which is not a response any person who is that age would

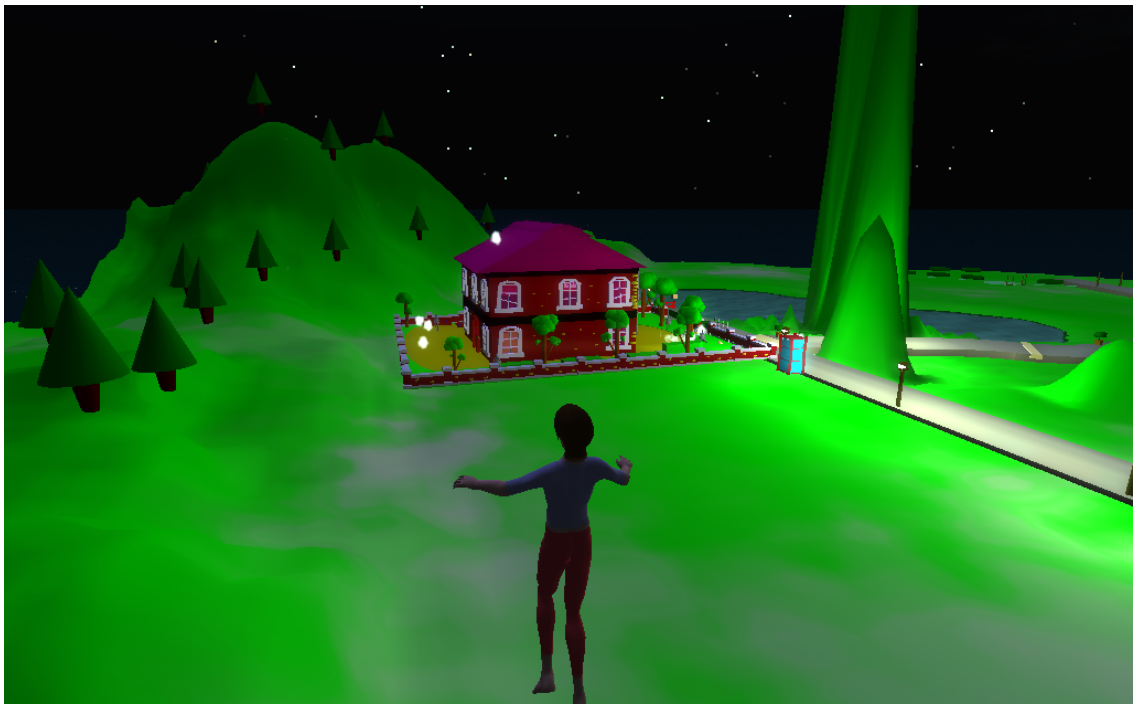

Figure 4.1: Evidence of distraction - students creating hills in the land.

give. Another problem in both classes was that a number of the students did not see the "Instant Message Received" alert in the corner of the screen, and so would not respond to the other students who were messaging them, and a lot of students were also using global chat rather than sending instant messages to specific people (despite instructions on how to do it properly). In both classes, we had about 40 people in the world (including bots), yet the bots did not have many people talking to them. In the first class, there were 10 conversations with the bots in total, and in the second there were 15. These numbers are very low when you consider the number of students in the world. The final observation about the classes in the computer labs was that almost everybody in the class used text language, which of course confused the natural language processing engine in the bot. Interestingly though, some students thought that everyone speaking text language must have been a bot, while others saw it as a give-away that the person they were talking to was a person.

In all of the classes, it was interesting to note that when we asked if the children thought that computers were intelligent at the beginning almost all students raised their hands, and at the end there would always be fewer students raise their hand. In some classes, at the start around 25 students would raise their hands, and at the end less than 10 students would raise their hand.

## 4.3 Surveys

As stated above, after the activity, we gave the students a survey to fill out. The survey consisted of the following questions:

1. Gender
2. A range of questions to determine their computer usage
3. How much did you enjoy the activity?
4. If you needed to be able to tell if someone you're chatting to online is a real person, or a computer pretending to be a person, how would you do so?

5. What sort of questions are most effective?
6. What did you enjoy most about the activity?
7. What did you enjoy least about the activity?
8. After performing the activity, would you be interested in working developing Artificial Intelligence?

In order to assess the survey, we subjectively assessed how much the student had learned from their answers to questions four and five and gave them a ranking of 1, 2 or 3 - 1 meaning they did not understand at all and three meaning they understood the Turing test perfectly. For the purposes of assessing the survey, for the last question, if the child answered "Yes" then we gave them a three, if they answered "Maybe" we gave them a two, and if they answered "No" then we gave them a one.

Despite the initial observations overall the students did not learn significantly more or less in either the off-line activity or the online one ($t=-0.689$, $df=96$, $p=0.49$). This is shown in figure 4.2. This however does not mean that the activity is pointless, as we need to consider it against our original goals, but it does mean that other factors would be used to determine which mode of delivery is best. This is described further in the next section.

Figure 4.2 also shows that irrespective of whether the students were in a class which was more focused and better behaved they learned the same amount as students who were in classes which misbehaved ($t=0.323$, $df=96$, $p=0.75$).

The classes which did the activity online did enjoy the activity significantly more than those that did it off-line ($t=-3.34$, $df=96$, $p=0.0012$). This is also shown in figure 4.2.

It seems natural that if the students enjoy the activity more, then they are more likely to remember the activity and are more likely to consider it as a path when choosing what to study at University and high school. However there was no correlation between how much the students enjoyed the activity and whether they were likely to consider it as a career option. This is discussed further in the Future Work section.

It was also interesting to see that a number of students thought outside of the scope of the original activity and when asked how one could tell the different between a computer and a human, they put "Ask to see them on webcam", or (more worryingly) "Ask to meet in person". Although the original Turing test states "the interrogater cannot demand practical demonstrations" [18], it is good to see that the students were actively thinking about it.

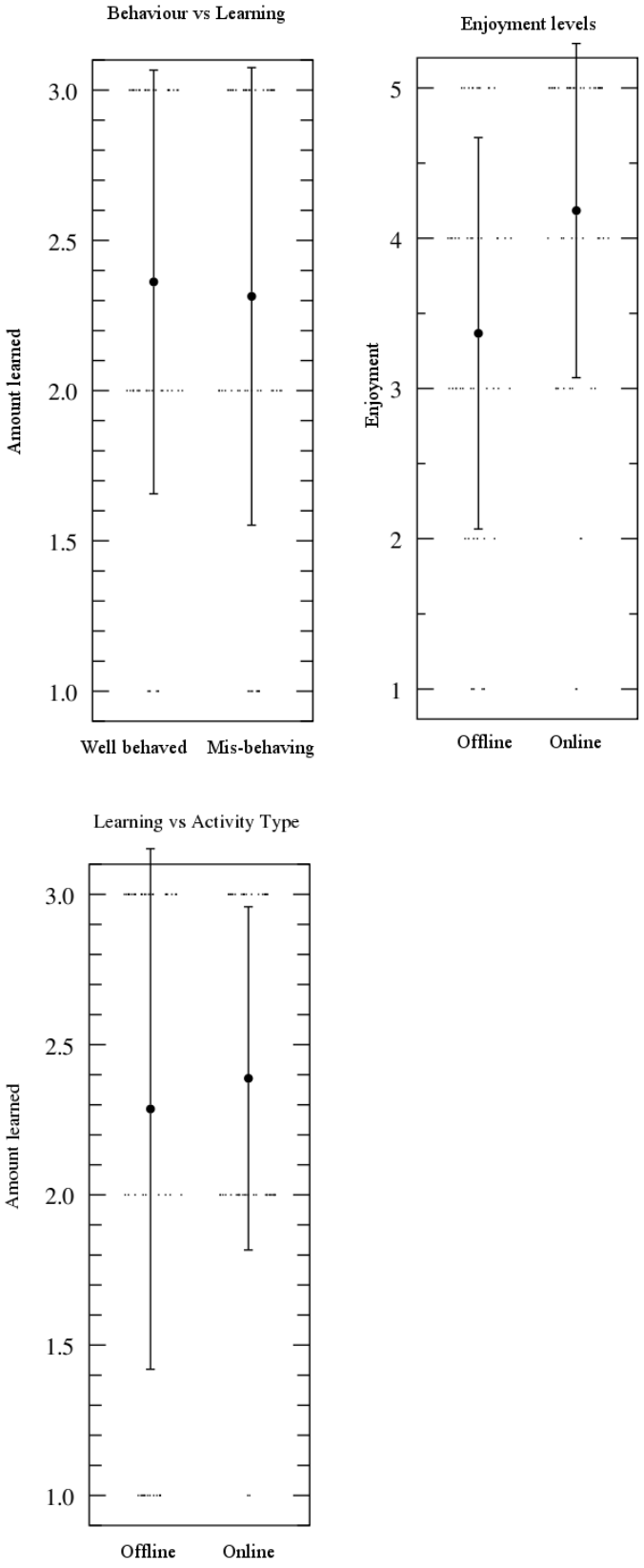


Figure 4.2: The results of the experiment.

# 5 Discussion

## 5.1 Is online worth it?

So should Computer Science Unplugged go online? In order to answer this question we must consider what we want to achieve with the project.

One of the major goals of the Computer Science Unplugged project was to allow teachers without a background in computer science to teach basic computer science concepts to classes of school children. Given the number of problems that we had while trying to deploy the system to the school, it seems unlikely that we will be able to develop a guide or system which would allow the majority of teachers, without any background in computer science or general system administration skills, to deploy the system without assistance.

If the goal of the activity was to teach the students basic computer science concepts, then given that the students in the experimental group did not learn significantly more or less from performing the activity, it seems like the amount of effort to set up the online activity is not worth it. However, if the goal of the activity is to raise awareness of computer science as a study path and to attract students to this path, then again maybe a different approach should be considered.

Finally, if the overall goal of the activity was to make Computer Science Unplugged more accessible to students in countries with less room outside, or students with disabilities, then we have shown that the concepts are taught just as effectively, so if the deployment or firewall issues can be overcome then this is a viable option in these cases.

## 5.2 Future Work

### 5.2.1 Alternative Approaches

One possible alternative approach is to instead of deploying the system to schools, we instead organise field trips in to Universities or other larger institutions, such as large high schools, for the following reasons:

1. The environment would be able to be set up and tested in advance.
2. There are dedicated system administrators who have skills and experience setting up server software and configuring client systems.
3. The computers are likely to have higher specifications than those at most high schools or intermediate schools.
4. They often have systems in place to allow automated installation and configuration of software which would help prevent a number of the issues we encountered.
5. Once the environment is set up, many classes could be run through with very little additional administrative effort.
6. More time would be able to be spent on a field trip, as it takes some time for the students to log in and to become familiar with the environment, and more activities could be ran through over the course of a morning rather than just one in a single afternoon class.

One final point, which is not made in the above list, is that students are much more likely to remember a field trip, rather than an exercise which is held in their own computer lab at school. This is probably the most important point, because as we have already shown, running the exercise off-line is just as effective as running it online, so the entire point of the activity is now solely to attract students to computer science, and for that they must have a fun and memorable experience, and a field trip is much more suited to these.

An investigation would need to be conducted to investigate whether bringing school students in to a University to have them do the activities is worth the effort involved, and to identify any issues which may arise from that. It may turn out that the additional hype created by the field trip results in them learning less.

We should also investigate whether simpler modes for delivering activities within computer labs in schools are more or less effective. By this, we mean things such as flash games or other applications designed to teach the students the same concepts, which do not require the same high specifications as a virtual world, and are also a lot less distracting. An example of one such activity would be an application where we pair up students with another person to chat to, some of which will be their peers and other will be bots, and then have them talk to them in a "chat room" style interface rather than a virtual world. These could be hosted online and in a web browser which would eradicate almost all issues with networking and system requirements, and allow even easier deployment. However, it remains to be seen if this would be as effective, fun and memorable as doing the activities in a virtual world, and again, we would need to assess the effectiveness of this approach against the other approaches and the original goals of the project.

Another possible path option is to use a virtual world other than SecondLife. Although SecondLife is the virtual world of choice, there are many other 3D virtual worlds which may have easier interfaces and give fewer opportunities for distraction.

Some schools already have SecondLife installed on the school computers. In these cases, setting up and configuring the software will be trivial, and hopefully the software itself would be less of a novelty for the students. In this case the virtual world may be less of a game to the students and more of a learning tool, and they may end up being a lot more focused. It would also save many of the set up and configuration issues that we encountered. Further work would still be necessary to investigate whether there is still any benefit to teaching the concepts using the virtual world. On this note, in some states or countries there may already be actively used OpenSIM grids. We may want to further investigate loading our environment and activities in to these grids, if they exist.

Finally, we would want to implement further activities in the SecondLife environment, as some of the activities may be able to better demonstrate concepts than we can do in real life. For example it may really help demonstrate the concept of how quickly powers of two grow - something which is very difficult in a classroom, as anything above 16 or 32 becomes very tedious. Additionally, Ritzema *et al* have shown that the amount students learn differs depending on the individual activity within the virtual world[16].

### 5.2.2 Other work

As mentioned above, no correlation was found between whether the students enjoyed the activity and whether they were likely to consider computer science as a career option. However the result of a small survey does not necessarily reflect long term decisions of the students and whether they will choose to study computer science later, so it would be worth doing a similar activity and identifying if running the activity (off-line and online) with a group of students increases their chances of studying computer science at high school or University.

# 6 Conclusion

Using computers for teaching computer science concepts in virtual worlds to children was not worth the effort in the environment in which we were experimenting. Setting up the virtual world was very difficult and troublesome, even for us, who have a significant amount of skill in setting up and configuring computers. It is therefore going to be even more difficult to ask a teacher to set up the environment, unless the process becomes simpler. We found it was no more effective to use a virtual world than to use traditional classroom techniques so although the students found using the virtual world significantly more enjoyable, the trade off between the effort required and benefits needs to be very carefully considered. Despite these difficulties, it still may be a great option for students with disabilities and in other environments where a large amount of space is difficult to get to perform the traditional activities, also, if a school has high end computers and skilled IT staff, or has SecondLife already installed, then it still may be worth the effort. In the future we will need to investigate the effectiveness of deploying the activity at a school which already has either SecondLife already set up or skilled technical staff, setting up the environment in advance at a University, or developing online games which teach the same concepts, inside a web browser.

# Bibliography


[1] T. Bell, I.H. Witten, and M. Fellows. *Computer Science Unplugged: Off-line activities and games for all ages*. Citeseer, 1999.

[2] T. Bell, I. Witten, and M. Fellows. Computer science unplugged. *URL http://www.csunplugged.org/*, 2009.

[3] Lillian (Boots) Cassel, Andrew McGettrick, Mark Guzdial, and Eric Roberts. The current crisis in computing: what are the real issues? In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 329–330, New York, NY, USA, 2007. ACM. ISBN 1-59593-361-1. doi: http://doi.acm.org/10.1145/1227310.1227426.

[4] Richard Stephen Clavering and Andrew Robert Nicols. Lessons learned implementing an educational system in second life. In *BCS-HCI '07: Proceedings of the 21st British HCI Group Annual Conference on HCI 2008*, pages 19–22, Swinton, UK, UK, 2007. British Computer Society. ISBN 978-1-902505-95-4.

[5] M.S. Conklin. 101 uses for Second Life in the college classroom. *Games, Learning and Society*, 1, 2005.

[6] M. Esteves, B. Fonseca, L. Morgado, and P. Martins. Using Second Life for Problem Based Learning in computer science programming. *Journal of Virtual Worlds Research*, 2(1), 2009.

[7] Daniel Frost. Fourth grade computer science. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 302–306, New York, NY, USA, 2007. ACM. ISBN 1-59593-361-1. doi: http://doi.acm.org/10.1145/1227310.1227417.

[8] Rachel Gollub. Second life and education. *Crossroads*, 14(1):1–8, 2007. ISSN 1528-4972. doi: http://doi.acm.org/10.1145/1349332.1349334.

[9] Philip Hingston. A turing test for computer game bots. In *IEEE Transactions on Computational Intelligence and AI in Games, Vol. 1, No. 3*. IEEE, To appear 2009.

[10] S. Kitchen, S. Finch, and R. Sinclair. Harnessing Technology schools survey 2007. *National Centre for Social Research*, 2007.

[11] Lynn Lambert and Heather Guiffre. Computer science outreach in an elementary school. *J. Comput. Small Coll.*, 24(3):118–124, 2009. ISSN 1937-4771.

[12] D. Livingstone and J. Kemp. Integrating web-based and 3d learning environments: Second life meets moodle. *Next Generation Technology-Enhanced Learning*, 8.

[13] D. Marghitu, T. Bell, B. Kearns, B. Ward, and K. Stephen-Pope. Using Virtual Worlds to Engage Typical and Special Needs Students in Kinesthetic Computer Science Activities: A Computer Science Unplugged Case Study. 2009.

[14] I. Novell. The Mono Project, 2006.

[15] M. Purvis. Distributed Services over Broadband Networks at ONGENS.

[16] T. Ritzema and B. Harris. The use of Second Life for distance education. 2008.

[17] Rivka Taub, Mordechai Ben-Ari, and Michal Armoni. The effect of cs unplugged on middle-school students' views of cs. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 99–103, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-381-5. doi: http://doi.acm.org/10.1145/1562877.1562912.

[18] A.M. Turing. Computing machinery and intelligence.

[19] Lucia Vera, Gerardo Herrera, and Elias Vived. Virtual reality school for children with learning difficulties. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 338–341, New York, NY, USA, 2005. ACM. ISBN 1-59593-110-4. doi: http://doi.acm.org/10.1145/1178477.1178541.

[20] Sarita Yardi and Amy Bruckman. What is computing?: bridging the gap between teenagers' perceptions and graduate students' experiences. In *ICER '07: Proceedings of the third international workshop on Computing education research*, pages 39–50, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-841-1. doi: http://doi.acm.org/10.1145/1288580.1288586.

[21] Y. Zhao and L. Wu. Second Life: a New Window for E-learning.

[22] Q. Zhu, T. Wang, and Y. Jia. Second life: A new platform for education. In *First IEEE International Symposium on Information Technologies and Applications in Education, 2007. ISITAE'07*, pages 201–204, 2007.

# A Questions for the offline Turing activity

This appendix contains the questions for the offline Turing activity, as well as the answers that the person pretending to be a computer gave. The questions which can be asked are in bold, and the computer's answers follow.

1. **What is the name of Bart Simpson's baby sister?** I can't remember.
2. **What do you think of Roald Dahl?** He writes funny books.
3. **Are you a computer?** Are you a computer?
4. **What is the next number in the sequence 3, 6, 9, 12, 15?** 18
5. **What do you think of nuclear weapons?** Nuclear weapons are very dangerous and should not be used.
6. **What is 2 x 78?** 166 (This is deliberately incorrect!)
7. **What is the square root of 2?** 1.41421356237309504878
8. **Add 34957 to 70764.** Wait for about 20 seconds before giving the answer . . . 105621.
9. **Do you like school?** Yes, I like school.
10. **Do you like dancing?** Yes, I like dancing.
11. **What day is it today?** Give the correct day of the week.
12. **What time is it?** Give the correct time.
13. **How many days are there in February in a leap year?** 2000 and 2004 are leap years.
14. **How many days are there in a week?** Seven.
15. **For which country is the flag a red circle on a white background?** I dont know.
16. **Do you like to read books?** Yes, I like to read books.
17. **What food do you like to eat?** Im not hungry, thanks.