

Lessons Learned Implementing an Educational System in Second Life

Richard Stephen Clavering

Computing Department
Lancaster University
InfoLab21, LA1 4WA

richard@clavering.me.uk

Andrew Robert Nicols

Computing Department
Lancaster University
InfoLab21, LA1 4WA
+44 7751 204885

andrew@nicols.co.uk

ABSTRACT

Second Life is an online 3D virtual environment that offers interesting potential for use in education due to its widespread availability, flexibility, and its use of standard platforms and input devices. Given a broad design brief for a nine-week masters' student project of using Second Life for education, we explored a range of potential ways of using the environment, and designed and implemented a 3D turtle-graphics system. In this paper we present our findings together with a reflection on both the constraints that Second Life places on the range of educational uses worth pursuing, and the specific issues likely to be faced by researchers creating other such systems.

Categories and Subject Descriptors

K.3.0 [Computer and Education]: Computer Uses in Education – *computer assisted learning (CAI)*

H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems – *artificial, augmented and virtual realities*

General Terms

Design, Human Factors.

Keywords

Second Life, Virtual Learning Environment, Education, Collaborative Virtual Environments, Virtual Reality, Teaching.

1. INTRODUCTION

Second Life has gained both considerable popularity and widespread attention in the media. Of particular interest within academia are its potential education uses, which we investigated in a nine-week project as part of our Masters in HCI. Rather than exploring social uses within education [1], or uses for groups with special needs such as those with learning difficulties [3], we looked at using it for learning and supporting teaching with a broader audience. We first explored the breadth of the design space by producing a range of conceptual designs, and then developed one of them – a 3D version of a turtle-graphics system (see section 4) – into a

working prototype. While the turtle-graphics prototype is not particularly novel, the experience allows us to offer researchers interested in using Second Life insight in three main areas: a detailed view of what Second Life is and is not, and how it differs from other virtual environments (see section 2); the types of design that are likely to be feasible (section 3); and design constraints that only became apparent during implementation, which we think are generally applicable but can be ameliorated (section 5). Section 6 presents our conclusions: that Second Life can be an effective platform for rapid prototyping of 3D interfaces despite its flaws, but that the limitations imposed by the system will make it hard to go from a prototype to a high-quality finished product.

2. SECOND LIFE

Second Life is an online three-dimensional virtual world, accessed via a custom software client available on multiple platforms, and with over seven million registered user accounts world-wide (though fewer regular users). Unlike online role-playing games, to which it is often compared, there are no set objectives, and the world is controlled to a much greater degree by the users. At a high level, the rules of the world are fixed: the Havok Physics Engine¹ provides detailed and fairly realistic physics simulation within the world, including enforcing aspects of real-world physics that need not apply to a virtual world, for instance buildings in Second Life cannot be larger inside than outside. However, the range of buildings, vehicles, avatars and objects users see are created primarily by the users themselves. There is an in-world currency, the Linden Dollar, which is convertible at no cost to and from US dollars. Linden Dollars can be used to buy areas of land in the virtual world, on which they can create their own environments (called “sims”). Buildings and other objects can either be bought from other users, or constructed from scratch at no cost other than the time spent. Users retain all intellectual property rights for objects they create: when selling them to other users they may individually control whether the buyer will be able to resell, edit, or create copies of the object.

The wide range of complex objects seen in Second Life are created by users linking together varying numbers of “prims” – basic (primitive) 3D shapes such as cubes, spheres, and prisms, which can be freely resized and rotated, and the appearance set to either a colour or texture image. The “material” a prim is made of can also be chosen from a pre-set list (including steel, wood, etc.) and this influences how the Havok physics engine treats the prim, as do a number of special properties for such things as a prim not being subject to gravity, being intangible, and whether the item is temporary (and therefore automatically

© Richard S. Clavering & Andrew R. Nicols, 2007
Published by the British Computer Society
Volume 2 Proceedings of the 21st BCS HCI Group
Conference

HCI 2007, 3-7 September 2007, Lancaster University, UK
Devina Ramduny-Ellis & Dorothy Rachovides (Editors)

¹ <http://www.havok.com> – Last retrieved 23rd May 2007

deleted soon after creation). Prims can be grouped together to make more complex shapes and have a number of objects and scripts added to their contents. Scripts are written in Linden Scripting Language (LSL), which is described by its creators as a cross between Java and C. LSL can be used to control objects, transfer currency between users, buy and sell objects, and to manage a range of other in-world activities. A number of features are built into the language to limit the potential load on servers, thereby enabling a fast and usable system for all users.

Avatars are used to represent the user in Second Life and many aspects of their appearance are highly configurable. Whilst the environment and avatar are both 3D objects within the Second Life World, some aspects of the user interface use conventional 2D dialogue windows overlaid on the 3D view. These include dialogues for performing activities such as localised chat, instant messaging, and search, and to display additional information such as the users' inventory and in world maps. Whereas objects can be built in the 3D world, it is not possible to modify the 2D elements of the interface or create new such elements. Input to both the 2D and 3D parts of the interface is via a standard keyboard and mouse, rather than specialised devices such as data gloves often found in virtual reality systems.

3. EXPLORING SECOND LIFE FOR EDUCATIONAL PURPOSES

Since Second Life became available to the public in 2003, it has found a growing popularity within the education sector. While many treat it as a game, it has encountered increased support from both educators and students who use it for a wide range of reasons, from running courses and classes in the environment to running research-based studies.

To improve the availability of Second Life for education, Linden Lab have provided a scheme named "Campus: Second Life", which allows college-level students in semester-long periods to borrow special areas of the environment for class-work and research. A second separate grid to be used solely by teenagers and schools, named "Teen Second Life" has been created. It attempts to provide a safer environment where strict regulation is enforced by Linden Lab employees. Educators are also invited to run classes and experiments within the Campus environment and, as of spring 2006, nineteen such events were running within the campus².

One of the most popular uses of Second Life in conjunction with education is as a social networking tool, enabling users to interact with one another and improve group dynamics, particularly within long-distance courses [1].

The ideas we considered were smaller in scope, due to the limited time available to us. They included a range of activities such as educational games and puzzles; models to explain scientific theories or engineering principles; and educational tools such as collaborative mind-mapping systems. Many of these are difficult to implement effectively because of unchangeable aspects of the Second Life environment, as explained below. For those ideas which were feasible, we often found that there was no obvious benefit in using Second Life rather than creating a tool in the physical world or using a 2D computer interface. A prime example of this were the educational games, and using Second Life for some of the

activities presently supported by virtual learning environments, such as distributing class notes and information, providing further sources of information and testing of students abilities. Similarly, we discarded the idea of a Second Life equivalent of a university open-day, in which potential students could explore a replica of the university campus, interact with staff and students, and find information about courses: course information is already readily available and more easily readable on the web. Interaction with other people, perhaps the most valuable part of such events, is much less rich in Second Life due to the restriction to text-based communication, and the lack of expressiveness of users' avatars. Additionally, requirements of virtual campuses have already been researched extensively [2].

We hoped to use Second Life's built-in Havok physics engine to implement educational tools, specifically simulations of real-world physics principles, reminiscent of the exhibits often found in science museums and similar to high-school science experiments. Examples might include electromagnetism, Newton's laws of motion, and conservation of momentum. While we believe that these would be worthwhile uses of the environment, and would give more feedback of the processes in action, the physics modelled by the Havok engine cannot be changed (e.g. by altering how collisions work, or the relative effects of gravity), even within private land owned by a user, and cannot be relied upon to be realistic enough for teaching science. Although the physics engine can be manipulated to have individual objects not subject to certain rules (such as gravity, or collision), there is no way to effect more detailed changes such as running in slow-motion.

While it might be possible to implement the physics for a simulation using LSL, there remain other issues restricting the usefulness of Second Life for such purposes. The inability to alter the available views of objects (e.g. to produce a cross-sectional diagram of an object) further restrict the environment for these uses, as does the difficulty of providing an interface to control the simulation: the 2D parts of the Second Life interface are not significantly extensible and lack the potential for either manipulation or customisation. Further, providing controls as Second Life objects would make for awkward interaction requiring the user to keep manipulating their field of view (or even moving their avatar) to get access to the controls as they moved it around the simulation.

A further idea we considered was to make use of the collaborative and graphical elements built in to Second Life by developing a multi-user mind-mapping tool. This tool was designed to enable users to share their ideas in a different, more creative manner. Although technically possible, we concluded that it could not be completed within the project time.

In the next section we present the idea we ultimately selected to develop: a 3D turtle-graphics system (shown in figure 1), selected for its proven educational value and the feasibility of its implementation.

4. THE TURTLE-GRAPHICS SYSTEM

Turtle-graphics is a simple model for drawing, using a metaphor of a turtle dragging its tail in the sand and thereby forming a line-art picture. The turtle is restricted to moving forward, turning, and raising or lowering its tail to fit with the metaphor; the commands to do so are usually issued using the Logo programming language. This combination has long been used in schools to teach children about computers and procedural thought (through procedural programming). Our idea was to implement a turtle-graphics system within Second

² <http://forums.secondlife.com/showthread.php?t=88521> – Last retrieved 24th May 2007

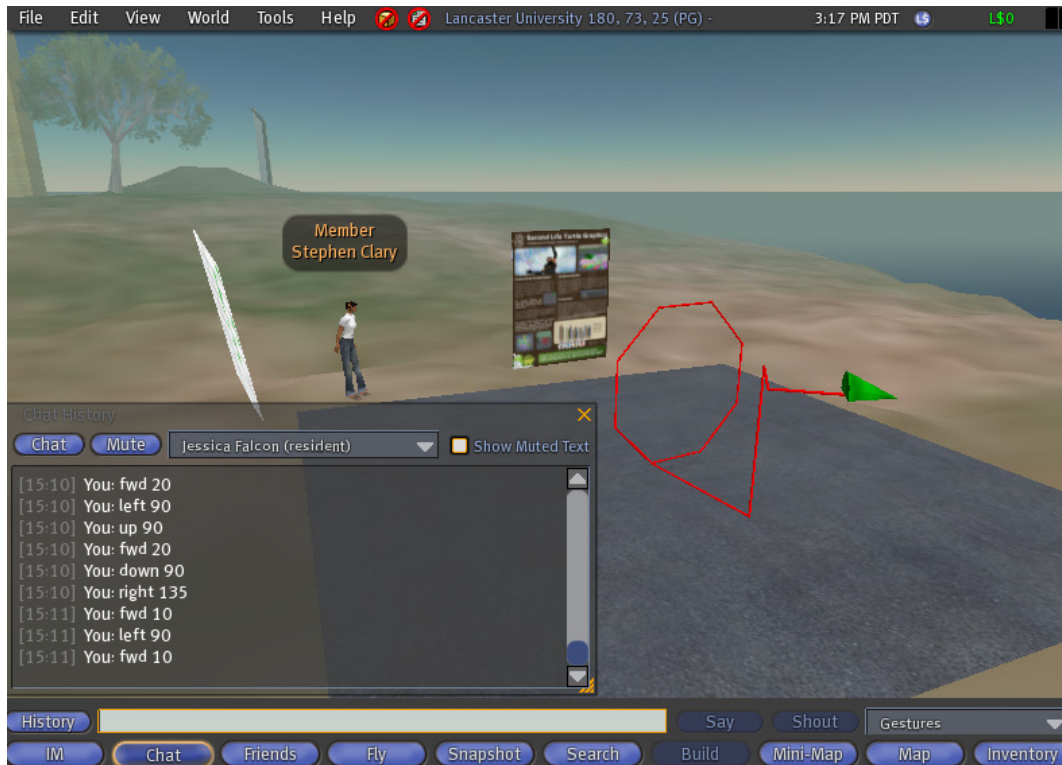


Figure 1. The turtle drawing in three dimensions with the chat window / interactive shell shown.

Life, and extending it to allow movement in all three dimensions (further breaking the metaphor, which was already stretched by the transition from physical turtle-shaped robots to on-screen turtles bearing little resemblance to their namesake). As well as the obvious benefits of being able to teach 3D graphics and geometry, and improving spatial thinking generally, we expected that using a 3D environment would make for a more challenging and engaging experience – hopefully improving the students’ knowledge and skill retention. While time constraints prevented us from carrying out a thorough evaluation, the informal procedure we used did tend to support this.

The design and implementation of the turtle were largely conflated due to our lack of time and of detailed understanding of the capabilities of the platform. As implemented, the system consists of a platform/plane object which the turtle draw on in 2D (and is needed as an origin for the coordinate system which the turtle moves relative to). The turtle is represented as a simple half-cone, which is the simplest possible way of fully indicating the orientation (a full turtle model could be substituted given more time). Pictures are composed of a series of cylindrical objects left behind the turtle, analogous to the lines found in a 2D system. Code is necessarily distributed across all three classes of object.

To control the turtle we envisaged the classic combination of an interactive console for immediate control, and a separate method for writing and saving more complex procedures (such as a full code editor). The latter proved impossible to implement (as explained in the next section), but we were able to implement a limited console. The commands expected from 2D turtle graphics were used, including those for calling and defining procedures. Additionally we provided an undo command, and two additional rotation commands to allow

movement in the third dimension. Due to time limits, we did not provide a full expression language and mathematical operations (due to the increased complexity of parsing input).

We had hoped to experiment with features to further the 3D geometry teaching goals, such as showing axes for the different coordinate spaces introduced each time the turtle turned, and being able to have the system display angles between arbitrary line segments, but limited time prevented this.

5. CHALLENGES IN DESIGNING THE TURTLE

In designing and implementing the turtle we met several classes of problem that would be relevant to other researchers. The most significant of these, which are explained in greater detail below, are: the lack of overall documentation of the system, and the corresponding need to implement while still learning about the system; the lack of extensibility of the 2D elements of the Second Life interface; problems caused by the mechanisms for limiting resource usage; and the mixed benefits of built-in features such as animation.

As when designing for any new platform, it was difficult to develop a detailed initial design for the turtle prior to starting coding, due to our limited knowledge of the platform's capabilities and limitations. However this was more marked than when (e.g.) switching between different toolkits for 2D desktop applications, since there is no standard set of widgets for 3D interfaces. Instead one must experiment with Second Life's graphical tools for creating prims, and read the documentation for LSL. The documentation lacks any overview of what kinds of actions are possible, instead simply providing descriptions of individual library functions and some language concepts such as the first-class events, and the resource-limiting

systems. LSL is a wholly procedural rather than object-oriented language – prims can communicate only through limited-length string messages (not via method calls). This results in a standard library that mixes expected basic functions such as those for array manipulation, and advanced functions for particle effects and for directing the physics engine. Overall, this leads to a style of development where the initial design is very abstract, with the details being decided only as implementation proceeds.

Being unable to extend the overlaid 2D elements of the Second Life interface proved to be the most significant problem in implementing our turtle, since entering Logo commands (either interactively or for defining procedures) is naturally suited to a 2D interface. Alternatives such as displaying the text on an in-world screen or board object would force the user to keep adjusting the camera or moving their avatar to switch between viewing their code and viewing the turtle. However, we found we could instead *repurpose* existing 2D overlays for our needs, and believe this could be a generally useful technique. In the case of the turtle this consisted of reusing the in-world chat system as a console for commanding the turtle: the turtle listens for all chat messages from nearby avatars, and attempt to parse them as Logo commands. The built-in chat history window in Second Life then functions as the command history of the interactive shell in a conventional turtle graphics implementations.

Repurposing interface elements in this way produces less than ideal results. As an example, providing good feedback for errors in Logo commands issued via the chat system is difficult because other users of Second Life may also be chatting near the turtle without intending to control it. We had to choose between issuing error messages for any message that wasn't a valid Logo command (including other users' conversations), which clutters the chat history, or silently ignoring errors, making it difficult for novice users to understand their mistakes. We briefly tried requiring all commands to be prefixed with "turtle" to distinguish them, but found to be too tedious for non-trivial use.

We also hoped to repurpose the existing script-editing window in Second Life for writing longer procedures or programs to control the turtle. It initially appeared that this should be feasible, at the acceptable cost of forcing users to use LSL syntax rather than Logo, but with the benefit of syntax colouring and free syntax error messages. However, running turtle scripts produced this way proved impossible, since LSL lacks any way of either evaluating code stored in a string, or executing a script by dragging it onto an object such as the turtle.

Resource-usage limits imposed by LSL, while admittedly necessary to prevent malicious or poorly-written scripts from interfering with all others on the same server, also cause considerable difficulties. Firstly, each script has a strict limit on the amount of memory it can use, and exceeding this usually results in stack-heap collision errors. This resulted in the "undo" command for our turtle having to retain only a very short history of commands. Considerable problems were also caused by built-in delays in certain library functions, including a few hundred milliseconds for each object instantiated, moved, resized or rotated. Delays also make coordination between different objects difficult (due to using message passing rather than blocking function calls), requiring manual delays to be inserted to keep the turtle's movements synchronised with the

creation of line-segments. In addition to delays, there is a second resource limitation mechanism called "energy", which accumulates over time and is used by some function calls. This appears only to be important for large objects (relative to avatar's sizes) that are "physical" (i.e. affected by gravity, and other rules of the Havok engine), and so was not relevant in our turtle system, and likely would not be in other educational tools.

The delay in built-in functions also allows Second Life to incorporate animation into movements and rotations: as objects are moved or resized using LSL, they are smoothly animated across the display. Whilst reducing the amount of work for a programmer, this introduces the possibility for additional confusion to the user. As an example, if the user would like to turn the turtle right by 270 degrees, Second Life will control the animation and instead turn it left by 90 degrees, potentially confusing users as to the current orientation of the turtle.

6. CONCLUSIONS

Overall, while Second Life does provides a rapid development environment for educational tools like our turtle, the constraints of the system such as the lack of extensibility of the 2D parts of the interface make the final designs of such tools suboptimal. Many of the built-in features such as animation and a detailed physics engine are a mixed blessing, since they cannot be controlled in detail, and some cannot be avoided. However, as Second Life continues to evolve this situation may improve.

The range of truly worthwhile educational applications of Second Life remains an open question. The efficacy of social uses of Second Life is hard to assess, which led us to instead explore uses involving small numbers of users, and largely ignoring the presence of an avatar. Many of the obvious ideas in this area such as explanatory interactive 3D models of Newton's laws of motion are difficult to implement well because of the lack of control of how objects are viewed, and how the physics engine operates, which we do not expect to change. Additionally, the use of keyboard and mouse interaction, while making the system more widely available, does exclude uses where tactile feedback is important.

7. ACKNOWLEDGMENTS

Our thanks to Nick Day and Can Zhao who were also members of the team designing the turtle, and to Corina Sas for her role as course director.

8. REFERENCES

- [1] Childress, M., and Braswell, R. Using Massively Multiplayer Online Role-Playing Games for online learning. *Distance Education*, 27, 2 (Aug. 2006), 187-196.
- [2] Prasolova-Førland, E. Analyzing place metaphors in different educational CVEs: Lessons learned. In *Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA '05)* (Porto, Portugal, December 14-16 2005). IADIS Press, 2005, 325-332.
- [3] Vera, L., Herrera, G., and Vived, E. Virtual reality school for children with learning difficulties. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE '05)* (Valencia, Spain June 15-17). ACM Press, New York, NY, USA, 2005, 338-341.