EXTENDING THE USE OF VIRTUAL WORLDS AS AN EDUCATIONAL PLATFORM
Network Island: An Advanced Learning Environment for Teaching Internet Routing Algorithms

John McCaffery, Alan Miller and Colin Allison
School of Computer Science, University of St Andrews, North Haugh, St Andrews, Scotland, U.K.

Keywords: Virtual world, OpenSim, MRM, Education, Interactive, Graph theory, Multi user virtual environment, MUVE.

Abstract: Virtual worlds provide a rich platform for supporting exploratory education. Their ability to bring together multimedia, programmability, interactivity and enhanced presence in a distributed 3D virtual environment makes them an excellent basis for interactive learning. This paper outlines work done in the virtual world OpenSim to create a learning environment for teaching the core algorithms which underpin Internet routing. This work demonstrates the power of virtual worlds to serve as a platform for developing 3D learning scenarios. To achieve this it was necessary to move beyond the limitations of the traditional virtual world scripting paradigm. This meant developing a system that allowed the power of high level software development to be added to the framework of a virtual world. Using OpenSim's Mini Region Modules, an API has been developed which allows for code written externally and compiled to software libraries to be imported into OpenSim via a scripting mechanism while the server is live. This mechanism has been used to develop a graph theory based visualisation tool that is fully situated within a virtual world. This visualiser is then used to demonstrate interactive simulations of Link State and Distance Vector routing algorithms. The mechanisms developed serve to highlight just how powerful virtual worlds can be as a development platform and how this power can be harnessed for education.

1 INTRODUCTION

Virtual worlds provide an intuitive 3D environment where users are represented by avatars (OpenSim, 2010; LindenLabs, 2010b; Worlds, nd; Smith et al., 2003). This presence is engaging. Users are able to, and like, interacting with each other. This in turn provides natural support for group work and collaboration. The environment is programmable. It is able to support a rich set of interactions, which allows it to be used in a variety of different ways. It is a multimedia platform; video, sound, pictures animations can all be supported (Weber et al., 2007). Thus a rich set of educational resources can be brought together and accessed through a 3D virtual world browser. The resources can be navigated by avatars moving around a 3D space, much as a group of students might explore an art gallery or a museum that they had exclusive access to. Content within 3D worlds like Second Life (SL) is created by residents. This extends beyond the ability to design and build things (Weber et al., 2007), for example to shape terrain, construct buildings and design clothes. Through the Linden Scripting Language (LSL) (LindenLabs, 2010a; Heaton, 2007) and other mechanisms users are able to program their world making it a truly interactive experience. This programmability has been used to ask students to create machines which demonstrate Dijkstra's Shunting Algorithms (Perera et al., 2009) or control the movement of a virtual turtle using simple commands entered into the chat box (Clavering and Nicols, 2007).

The novelty of this project is the scope of the system it builds. Using the multimedia and programming features of virtual worlds for education has produced interesting and valuable work before (Chodos et al., 2010; Bellotti et al., 2008; Ritzema and Harris, 2008; Gollub, 2007; Livingstone, 2007). These projects are often interactive however the scripted content of the learning environment is generally fairly simple. One of the reasons for this is the scripting environment in which programmers must work. The scripting mechanism developed for Second Life, LSL, was devel-

McCaffery J., Miller A. and Allison C..
EXTENDING THE USE OF VIRTUAL WORLDS AS AN EDUCATIONAL PLATFORM - Network Island: An Advanced Learning Environment for Teaching Internet Routing Algorithms.
DOI: 10.5220/0003342402790284
In *Proceedings of the 3rd International Conference on Computer Supported Education* (CSEDU-2011), pages 279-284
ISBN: 978-989-8425-49-2
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

oped for a specific environment. This environment is one where all servers are to be run by one commercal company. This means limiting user power to write scripts which could potentially damage other user's experience is a necessity. The result is a strong scripting language with very defined limits. LSL must be compiled in world from a text editor provided by the system client. Scripts are attached to specific objects in world and only have the power to affect the object they are attached to, not the larger scene graph. Any cross object communication must be done via the chat mechanism. Scripts are also given arbitrary constraints such as placing time restrictions on how often scripts can change an object's position or spawn new objects. These restrictions artificially limit what can be achieved in developing applications for virtual worlds.

The contribution of this work is to demonstrate how, within an open source context, the limitations of virtual world scripting can be overcome. It does this by building an educational system of greater scope than has previously been attempted. The new design pattern allows systems to be built in an efficient and maintainable manner. This project shows a methodology whereby code can be developed in an efficient manner, making full use of standard IDE tools and good abstraction techniques. This code can then plug into a virtual world and become part of a learning environment. The fact that this code was developed externally, using an IDE means that the scope of projects developed using this technique can be greatly increased compared to what can feasibly be attempted working only with in world scripting mechanisms. This project essentially creates a 'best of both worlds' situation. That is to say the developer gets all the benefits of in world scripting in a virtual world combined with all the benefits of working in a full IDE. The developer can write their code in an IDE, compile it and then restart it in world without having to restart the server itself.

The system which was developed is a graph theory visualisation tool which has been used to create an interactive sandbox for use in teaching internet routing. The two algorithms which the system visualises are the version of Dijkstra's Shortest Path algorithm (Nvrat, 2004) used in Open Shortest Path First (OSPF) (Peyer et al., 2009; Kurose and Ross, 2009; Cormen et al., 2001) routing and a generic Distance Vector routing algorithm (Kurose and Ross, 2009). These two algorithms are appropriate because they are two of the most widely used routing algorithms on the Internet today. They are also appropriate because they both function very differently. That it is possible to develop a system which is general enough to visualise both algorithms shows that a well abstracted and fully de-coupled system can be developed and then plugged in to a virtual world.

This paper starts by outlining the system that was developed. It then goes into more detail on how the algorithms are visualised. There is then an analysis of problems faced when developing within a virtual world and the options available. Lastly the architecture used to develope the system is discussed.

2 CONTROLLING SIMULATIONS

Network Island is based around a tool which allows students to explore the behaviour of a routing algorithm. This tool is designed to add a new perspective to learning about routing, supplementing diagrams in textbooks or printed lecture notes. In the real world it is not possible to physically place down routers and link them up and then watch packets flow between them and algorithms run across them. In a virtual world it is possible and that is what this project brings to life. Users can create a topology, start sending a stream of packets across the network then alter the topology and see, in realtime, how the flow of packets alters in response to the changes. By making the system interactive and responsive and utilising the central metaphor of virtual worlds, that of the avatar, users are given a hands on experience of what is, in the real world, a completely abstract concept.

When the user logs in to the system what they see is a control panel and any network topology that is already in place. The topology consists of nodes linked together by links. The user is then able to use the control panel to highlight how the algorithms work from the point of view of any node they wish. This allows them to observe the steps the algorithm takes as it calculates routing paths across the network. The user is also able to send packets across the network and observe how changes made by the routing algorithm affect the route the packets take.

As well as running the algorithms on the topology that already exists the user is also able to alter the topology. They are free to add nodes, create or remove links and change the weights of the links that exist. The design of the interface is intended to be as simple as possible so the user can get instant feedback on how changes they have made to the network topology affect the routing. The goal is that users are free to watch how the algorithms work, develop hypothesis about how changing the topology will change the algorithms then make those changes and see if their hypothesis is correct.

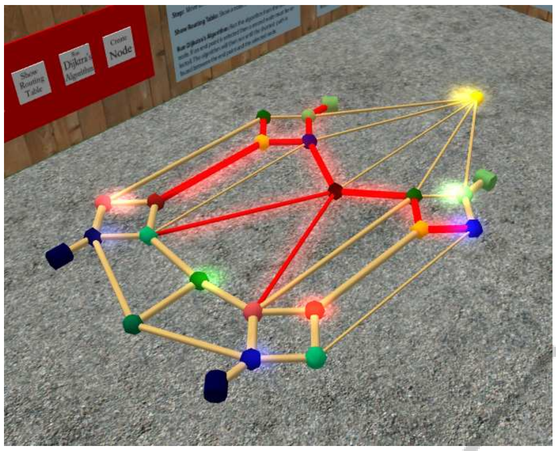

Figure 1: Dijkstra's Algorithm midway through calculating shortest paths from the black node.

3 THE ALGORITHMS

The two algorithms that the system visualises are Dijkstra's Shortest Path First algorithm (Moy, 1998) as it is used in Open Shortest Path First routing and the Distance Vector routing algorithm (Malkin, 1998). The system highlights how the algorithms are working with respect to a user specified node. Because both algorithms are very different what constitutes highlighting how the algorithm is working is also markedly different.

3.1 Dijkstra's Shortest Path First Algorithm as used in OSPF

Dijkstra's Shortest Path First Algorithm forms one half of the OSPF protocol. In OSPF each node maintains an internal graph representing the topology of the network. This internal graph is built up from information contained in packets which every node floods through the system. These packets detail their sender's links and their weights. These internal graphs are used to run Dijkstra's Shortest Path Algorithm which calculates the fastest paths from the node which is running the algorithm to every other node in the system that it knows about. These paths are then used to get the link along which the node should route packets for any given target node.

When Dijkstra's Algorithm is running in a node (known as the root node) at any given time the algorithm has a node which it has selected as the 'current' node. It then probes the links out from the current node. The path chosen between a current node and the previous current node is guaranteed to be the shortest back towards the root node. Probing a link involves checking the weight of the link and the state of the node at the other end of the link. If the node at the other end has not yet been probed it is added to a list known as the 'tentative list'.

The algorithm is visualised by highlighting the current node, highlighting the link to the neighbour of the current node which is currently being probed, highlighting the nodes in the tentative list and highlighting the shortest paths so far calculated. Highlighting is done either by setting the colour of a link or by making a node glow. Figure 1 shows Dijkstra's Algorithm running. When the algorithm is triggered to run it cycles through the steps with the root node being the node the user selected to highlight the algorithm from.

The flooding mechanism which forms the other half of the OSPF protocol is also visualised, albeit in a simplified manner. Whenever a change is made to the network topology (i.e. a link is added or removed) the two affected nodes (the nodes at either end of the link) send out flood packets along all of their links. These packets are visualised for the user and continue to be flooded until all nodes have received at least one flood packet. They serve to visually highlight the fact that the flooding mechanism is a crucial part of the OSPF algorithm.

3.2 Distance Vector Routing

The distance vector algorithm differs from the fully OSPF Shortest Path algorithm as it is a distributed algorithm. Each packet that is received forms part of the algorithm itself, rather than simply providing the information on which the algorithm will run. How the algorithm works is that whenever the network topology changes the nodes involved in the change send out packets to all their neighbors. These packets contain distance vectors which map the distances to all the nodes the sender knows how to route to. When these packets are received the sender's distance vector is analysed against the recipient's distance vector and if necessary the recipient changes their distance vector. If the recipient makes a change it then sends out its modified distance vector to all its neighbours and the process continues until all nodes have assimilated the new information.

To visualise the algorithm the packets containing the distance vectors are animated. Users can make changes to the topology and watch as the changes propogate out through the network. A highlight mechanism is used to facilitate understanding of how the receipt of distance vector packets alter the way that routing occurs. When the user chooses to highlight the algorithm what happens is the shortest path from

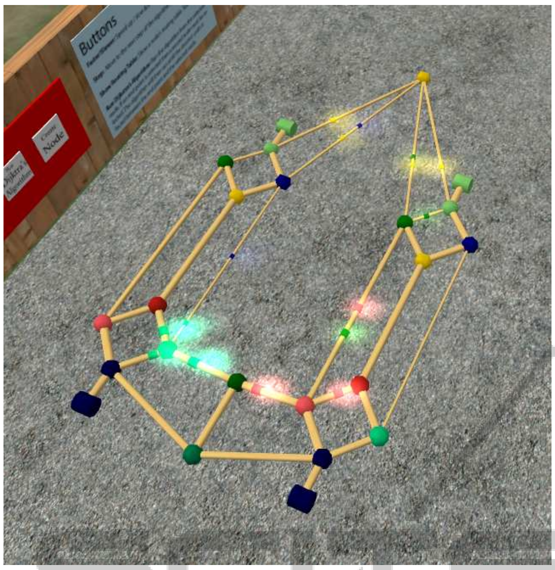

Figure 2: Distance Vector update packets moving across a topology after the removal of a central node.

the selected node to every other node is highlighted. This means that the user can then go and change the topology, watch the packets signalling the change in topology radiate out and see how the arrival of these packets goes on to affect the highlighted shortest paths. Figure 2 shows update packets that are starting to propagate through the system after the removal of a central node in the topology.

4 PROGRAMMING VIRTUAL WORLDS

One of the reasons virtual worlds are so interesting from an education point of view is their programmability. Not only can users change the shape of the world around them and add new geometry and structures into the world they can also attach code to the objects in-world. The fact that scripting behaviour of objects in-world is an integral concept to virtual worlds is what this project harnesses to create an enhanced learning environment. By approaching learning about Internet routing algorithms from a virtual world perspective rather than an animation or pure simulation perspective the fundamental concept of users having direct control over their environment is harnessed to give users control over the simulation.

The major challenge was the level of programmability that specific virtual worlds provide for users and administrators. Enabling untrusted users to write and execute code in a distributed environment may be problematic. As soon as all users are given the power to execute their own code within the larger framework of the distributed application they are given the ability to break the application.

When deciding how to face this challenge developers of virtual worlds are left with a choice. On the one hand they can give the user the power to create complex and involved systems within the environment they have created. This makes for great possibilities but also leaves the system vulnerable to exploitation. Alternatively the developers can limit users to develop within very controlled parameters which limit what the user can do but also limit their power to harm other user's experience.

SL, as it is available to the public, is an example of the latter design decision. If you connect to SL you are connecting to a server hosted by Linden Labs. This server is configured to put hard limits on the amount of damage you can do to the system as a whole. If you wish to develop within SL you must use LSL and accept its boundaries. These boundaries then constrain the complexity of the systems you can design.

OpenSim has a different service model for providing virtual worlds. It is a highly configurable open source virtual worlds platform which allows multiple instances to be connected together to create a virtual world that scales. Administrators are free to create their own server, configure it how ever they wish and have users connect to it. As such, where with SL all servers are run by Linden Labs and have the same security settings, in OpenSim it is up to individual admins to configure their server. This means any OpenSim server can be placed at any point on the continuum between tightly controlled to limit malicious behaviour or very free to give users, or a single group of users, great power over the environment. Because OpenSim puts the running of the server in the hands of the administrator it becomes the admin's choice how the server is configured for security. If the administrator wants to create a server where all users have access to highly powerful scripting tools such as MRM they can. Alternatively they can nominate a few users as developers and give them access to the more powerful tools while limiting general user's ability to alter the system. OpenSim supports LSL which is built into SL but it also supports other programming mechanisms. Being open source the code can be modified any way the administrator wishes. This could be directly, changing the source code that exists already, or it could be by adding in features in a modular manner. OpenSim provides four modes for programming content.

1. **Sandbox Scripting.** LSL is an example of this, users write LSL scripts with very defined con-

straints in-world. On one level this is powerful, in that it allows in-world objects to be programmed to react to events in a rich and varied way. However the creation of complex systems is difficult and time consuming. Access to system resources is controlled making the creation of moderately time sensitive applications impractical. Linden Scripting language was originally developed by Linden Labs to be the scripting language used in SL. It has since been implemented in OpenSim.

2. **Internal API Scripting.** An example of this are MRMS and the alternative scripting mechanism they provide. Where LSL is a custom designed scripting language developed to be written in-world and to manipulate the object the script is placed in MRMs are a C# API. Code is written in C# using an API which gives access to the world's scenegraph. This code can be written in-world in the same way as LSL, using a text editor built into the client. This means at one level it can be used as a straight substitute for LSL. The only difference being that MRMs allow code to manipulate any object in the scene, not just the object the script is running in. As well as allowing code to be written in-world in the same way as LSL MRMs also allows external libraries to be loaded in and referenced by its in-world scripts.
3. **External API Development.** This is a system whereby a small piece of code, plugged directly into OpenSim then links to an external library which runs as an MRM. This project uses a custom designed system for this functionality[1]. The programmer writes minimal code in-world, all they do is specify a configuration file. This in turn triggers an external library which goes on to load up a series of libraries defined in the configuration file. This system allowing code to be written in an IDE, compiled in an IDE then run in-world without a restart of the server.
4. **Region Modules.** a more heavy weight approach to system development. The programmer has open access to OpenSim internals and writes a module which is plugged directly into OpenSim. The problem here is that the absence of a structured API means that changes in OpenSim are likely to cause Region Module to break. Also writing region modules means that in order for the module to be recompiled the server must be restarted. Region modules are more suited to changing the way OpenSim itself works and adding major features than for scripting the behaviour of in-world objects.

[1]There is a freely available, open source, plugin for OpenSim which does this called MRM loader available. However at the time of writing it is in the early stages of development and has two major drawbacks. 1. It does not allow any arguments to be passed in to the loaded module, 2. A flaw in the design of the system means that the current version does not allow listeners to be added to the world. This means that no user interaction can happen in a system developed using MRM loader. This is why a proprietary solution was developed.

The relationship between the above programming modes is shown in Figure 3 which is based on a diagram from (Deem, 2009).

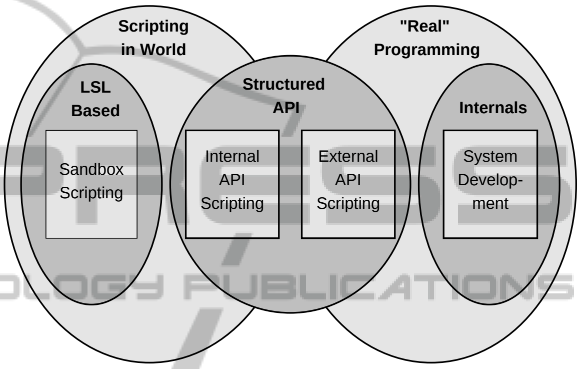

Figure 3: Different methods of programming content in OpenSim.

5 ARCHITECTURE

The architecture of the system is designed to fully exploit the ability to reference externally compiled libraries from in-world scripts. There is a library, the scripts library, which can be dropped into the main OpenSim directory and referenced by any script. Initialising an instance of an object with the location of a config file hands over control from the in-world script to the externally compiled library. The instantiated class can then dynamically instantiate a bootstrap class from a library whose location is specified from the configuration file. By changing what configuration file the class is initialised with different systems or configurations can be loaded. The bootstrap class must adhere to an interface defined in the scripts library but the files can be compiled and placed anywhere in the file system. These files can be re-compiled while the system is running in-world and the in-world system can then be restarted and the newly compiled code loaded. This mechanism means that a developer can use an external IDE to develop a system. The developer is able to make changes to the system, re-compile and then type a command in-world to restart the system and see the changes appear.

This is a powerful system as the developer is able to utilise all the power of multiple libraries, high level programming and IDE development tools and at the same time behave in-world as if they were just writing and re-loading a script, never having to restart the server.

6 CONCLUSIONS

Network island is an example of the scope of project that can now be developed within a virtual world platform. It is a full, dynamic, adaptive, interactive, distributed system that gives users a tool to help them understand an abstract concept. It demonstrates that not only can something like this be developed but it can be developed at a level of abstraction which allows educators to go in and write plug ins for the system to alter it depending on what they need to teach. This project shows the scope which is available when developing in a virtual world and leverages relatively young technology to look at a new direction which can be taken with these environments. By developing the system as essentially a plugin which can then support further plug ins it shows how a system can be put together which works within virtual world's tradition of easy development and simple scripting but has considerably more power behind it than has previously been available.

REFERENCES

Bellotti, F., Berta, R., De Gloria, A., and Zappi, V. (2008). Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds. In *DIMEA '08: Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, pages 77–84, New York, NY, USA. ACM.

Chodos, D., Stroulia, E., Boechler, P., King, S., Kuras, P., Carbonaro, M., and de Jong, E. (2010). Healthcare education with virtual-world simulations. In *SEHC '10: Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*.

Clavering, R. S. and Nicols, A. R. (2007). Lessons learned implementing an educational system in second life. In *BCS-HCI '07: Proceedings of the 21st British HCI Group Annual Conference on People and Computers*, pages 19–22, Swinton, UK, UK. British Computer Society.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition.

Deem, M. (2009). Maimed leech. http://maimedleech.com/.

Gollub, R. (2007). Second life and education. *Crossroads*, 14(1):1–8.

Heaton, J. (2007). *Introduction to Linden Scripting Language for Second Life*. Heaton Research, Inc.

Kurose, J. F. and Ross, K. W. (2009). *Computer Networking: A Top-Down Approach*. Addison-Wesley Publishing Company, USA, 5th edition.

LindenLabs (2010a). The lsl portal. http://wiki.secondlife.com/wiki/LSL_Portal.

LindenLabs (2010b). Second life. http://www.secondlife.com.

Livingstone, D. (2007). Second life education workshop at the second life community convention, san francisco, august 20, 2006. *eLearn*, 2007(3):4.

Malkin, G. (1998). RIP Version 2. RFC 2453 (Standard). Updated by RFC 4822.

Moy, J. (1998). OSPF Version 2. RFC 2328 (Standard). Updated by RFC 5709.

Nvrat, P. (2004). Review of "algorithm design: foundations, analysis and internet examples" by michael t. goodrich and roberto tamassia. john wiley & sons, inc. 2001. *SIGACT News*, 35(2):14–16.

OpenSim (2010). Opensimulator. http://opensimulator.org.

Perera, I., Allison, C., Nicoll, J. R., and Sturgeon, T. (2009). Towards successful 3d virtual learning - a case study on teaching human computer interaction. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for Internet Technology and Secured Transactions*, pages 1–6.

Peyer, S., Rautenbach, D., and Vygen, J. (2009). A generalization of dijkstra's shortest path algorithm with applications to vlsi routing. *J. Discrete Algorithms*, 7(4):377–390.

Ritzema, T. and Harris, B. (2008). The use of second life for distance education. *J. Comput. Small Coll.*, 23(6):110–116.

Smith, D. A., Kay, A., Raab, A., and Reed, D. P. (2003). Croquet - a collaboration system architecture. *c5*, 00:2.

Weber, A., Rufer-Bach, K., and Platel, R. (2007). *Creating Your World: The Official Guide to Advanced Content Creation for Second Life*. Wiley, Indianapolis, IN.

Worlds, A. (n.d.). Active worlds. http://www.activeworlds.com/.