Alexandria - A Virtual Repository of Knowledge

Joseph M. Graham*[1] and Eva Comaroski†[1,2]

[1]Virtual Islands for Better Education (VIBE) University of New Orleans, Department of Biological Sciences (http://wiki.bio-se.info)
[2]Wizardry and Steamworks, Consulting Group, United Kingdom (http://was.fm)

August 13, 2013

Abstract

The Virtual Islands for Better Education (VIBE) is an allotment of OpenSim islands targeting a wide audience, starting with activities for K-12 education, colleges University-level education. Overall, the purpose of VIBE is to create learning tools in virtual worlds, including lesson plans and walk-throughs. To complement that effort, we have built a library that would conveniently offer the reference material to support the learning activities. We present Alexandria, an OpenSim-based virtual library that uses synchronization techniques based on OpenSim Scripting Language (OSSL) and server-side software in order to allow the stashing of educational and recreational material within virtual worlds.

1 Introduction

Stashing material in the real-world is expensive due to the general high-costs of space as well as the maintenance that is required to preserve the material. With the advances of technology, we have seen major benefits from stashing content digitally. Foremost, there are many cost-effective techniques (Lee, Slattery, Lu, Tang, & Mccrary, 2002) that allow the sound preservation of material, ranging from the ability to back-up the content to the fact that the overall lifetime of digital medium outstretches by far the lifespan of paperback material (Ranstam, 2008).

Studies (Ladd & Mize, 1983) have shown that learning, whilst being a personal form of development, also may convey the "social experience" where students get to interact with their peers, as well as being able to build connections

*joseph.graham.michael@gmail.com
†eva@was.fm

that will further help them in their career. To that end, the digital medium provides an excellent technical way to preserve, distribute and annex material but digital medium fails short when it comes to providing the social interaction between students.

In order to address these shortcomings, we have grafted the two worlds together - the factual dissemination of learning material with the social experience of a real-world library. The result is a Virtual World library that contains learning material for students, as well as providing a game-like multi-player experience that allows students to meet virtually in order to disseminate data together. Taking one step further and benefitting from the development of media delivery, we have additionally added recreational material to supplement the library.

Alexandria is built using the OpenSim (Overte, 2011) software which is a platform forked off Linden Lab's (Linden, 2003) code. The OpenSim platform allows one to create a virtual world, to which multiple clients can connect and then interact. Contrasting with other Virtual World building platforms, such as Unity3D (Wang et al., 2010), OpenSim allows the building of content directly within the world. This is done by uploading assets such as textures, animations and sounds and then applying the assets to in-world objects or to clothes. Internally, OpenSim pre-generates a few defaults, such as the sky, or the default ground - which can then be changed, as well as fully implementing the basic protocols for multiple clients to connect and interact. While building Alexandria, these defaults have allowed us to focus on the library, rather than re-implementing stereotypical game-related mechanisms.

2 Overview

The motivation behind creating a virtual library such as Alexandria is explained in the Motivation chapter 3. Further on in the "Technical Background" chapter 4 the underlying technical mechanisms for data synchronization and in-world display of literature are explained. An overview of the library is provided in the "Overview of Alexandria" chapter 5 where we informally describe the current outlook of the library. In the "Conclusions" chapter 6 a table is provided that sums up literature gathered so far. We finalize with a discussion that explains why the authors believe that Virtual World libraries could be a potential future asset that could easily measure up to real-world libraries.

3 Motivation

An old proverb advises to *"not judge a book by its cover"* and historically, that proverb relies on the lack of associative or contextual information (Medin & Schaffer, 1978) that a book cover holds in relation to its contents. Even in modern, real-world, large libraries, spanning hundreds of thousands of books, when a person reaches a bookshelf, they are presented with a wide palette of

book-choices by author. Although they may have been following classifications such as those already provided by the Dewey system (Dewey, 1965), it is still uncertain for an individual which book they should pick. In practice, without being able to read the whole book, the selection process is performed by associating the author names, or by looking over the cover, perhaps an illustration or a short biography of the author on the back - all of these selection criteria being quite tertiary to the literature held between the covers. In the end, due to the lack of associative information, libraries tend to be consulted only when an individual knows precisely what they are looking for. For example, the Dewey system is an exhaustive classification (Wiegand, 1998) that is apt to classify any type of literature and works great for finding something specific. Browsing a library with thousands of books becomes extremely difficult if an individual does not know precisely what they are looking for.

In that sense, Alexandria tries to address those shortcomings by providing much more than book covers, searching system or timeline-based triage. Alexandria is built by creating floating islands, each decorated and themed by the literature that is placed on those islands. It is thus very convenient to "just browse" the repository by following the decorations and artwork. For example, an individual looking for war-time journalism, will find a floating island decorated with sculptures of soldiers, a burning vehicle and perhaps a house on fire that contains the bookshelves with the corresponding literature.

Alexandria does not address data archival and preservation as much as it addresses the delivery and display of that data. Technically, as will be explained further on in this article, the technology of Alexandria uses a database as a backend (namely, MySQL but SQLite is also possible) so it inherits the qualities of that database. However, the improvement over other similar database-based libraries is the ability to include artistic hints and pointers in order to deliver thematic literature at a glance.

4 Technical Background

SecondLife (SecondLife, 2003), a product of Linden Labs, provides a pragmatic programming language (similar to LISP), called the Linden Scripting Language (LSL). While LSL caters for most of the necessities of object manipulation, the language has no constructs for storing large chunks of data within the Virtual World. In short, using LSL one is able to read data from a document format called Notecards, but one cannot write to Notecards. For building libraries such as Alexandria, this read-only limitation makes the bulk-upload of textual material prohibitive - short of having to manually copy and paste the contents of every document.

On the other hand, OpenSim allows C# scripting within the virtual world and additionally provides extensions to LSL, called the OpenSim Scripting Language (OSSL), that is able to generate Notecards. By programming in LSL and using the OSSL extensions, a framework has been created for Alexandria that allows the import of textual material within the Virtual World.

Nevertheless, the Notecard format created by Linden Lab is limited and only able to render plain ASCII text, and integrate landmarks or pictures. Another limitation is that Notecards that spam more than 255 bytes cannot be read by scripts. This is insufficient in the case where technical documents have to be stored on Alexandria. The uploads are thus split into two main categories: documents that do not include figures or that do not rely on extensive typography (such as mathematical renderings) are uploaded using the Notecard format, otherwise technical documents are converted to images first and then uploaded directly into Alexandria as textures.

4.1 Automatic Document Synchronization

A sketch of the upload and archival of Notecard-based documents, is illustrated in Figure 1. Alexandria is built using "themed" floating islands, where every island is decorated to the specifics of the type of literature that can be found on that island. For example, the American poets floating island is roughly a fusion between the Gothic style of Edgar Allan Poe, coupled with the rural style in the industrial era. Similarly, the ancient Greek literature island is decorated using statues and cultural-specific idiosyncrasies pertaining to the Greek islands.

On these themed-islands, bookcases are placed that are given the name of the author whose work they will contain. The underlying filesystem contains a folder that carries the same name of that bookcase. A synchronizer script, illustrated in Figure 1 scans the folder on the filesystem for documents.

Whenever a document is added or deleted from the named folder on the filesystem, the synchronizer script regenerates the bookcase by creating or deleting Notecards. Adding a new author is a task that consists in creating a folder on the filesystem named after the author and then creating a book-case on Alexandria, giving the bookcase the name of the author and adding the synchronizer script to the bookcase.

The "filesystem" in Figure 1 is to be thought as a generic filesystem that could have an underlying network or cloud-based storage. The synchronization script does not care about the lower-layers of IO, but is designed to be aware of folders and text-files. This gives great opportunities for collaboration, for example, we have been using Dropbox (Wu, Ping, Ge, Wang, & Fu, 2010) as a mounted share on the server that Alexandria runs on in order to allow several contributors to upload material into Alexandria. Once a bookshelf is created, named and the synchronization scripts are set-up, users can start to add documents that the synchronization script reads and automatically adds the documents to the corresponding bookshelf. Given the limited resources and personnel that VIBE has at its disposal, we have found that this method of building a library by crowd-sourcing (Huberman, Romero, & Wu, 2009; Brabham, 2013) was by far more feasible than hiring a dedicated staff. Other filesystems based on high-level IO abstraction, for example filesystems in userspace (FUSE, 2009) or distributed filesystems such as CODA (Kistler & Satyanarayanan, 1992) and even distributed systems (Lauvset, 2001), are also possible options for storage backends.

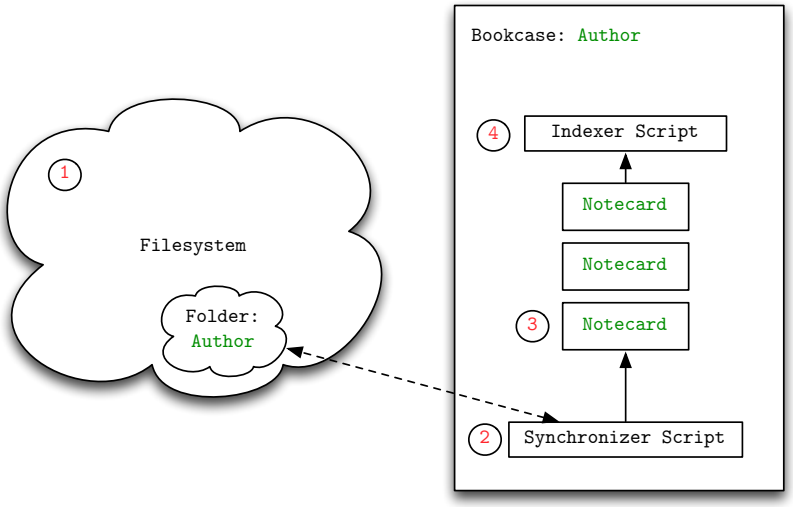

Figure 1: The filesystem contains several folders named after the author (1). These folders get processed from an in-world synchronization script (2), which reads the documents in the folder and generates Notecards within the bookcase (3). Additionally, and indexer script (4) reads the Notecards whenever an individual searches for a key-word.

4.2 Searching and Indexing

The indexer script illustrated in Figure 2 communicates with an in-world object called the "Oracle" in order to locate the position of bookcases in Alexandria whenever an individual (a human being, called an "agent" in Linden's terms) searches for a given keyword.

When an agent searches for keywords using the "Oracle", the query is relayed over a communication channel to all the bookcases on Alexandria. The indexer script returns a positive result if those keywords appear in the name, description or within the documents contained in the bookcase. This is actually performed by a weighted search, so that if an agent requests, for example, the name "Poe" the indexer script within the Edgar Allan Poe bookcase will be more likely to return a result in time than if the keyword was found within the documents.

The weighting is performed by racing the indexer scripts and by setting a timeout so that first the name of the bookcase is checked, then the description and if there is sufficient time left over before the query expires, matches within the document are returned as well. This ensures that only the most relevant bookcases are selected for a given user-supplied query. The concept of racing the bookcases using timeouts is inspired from race-conditions (Carr, Mayo, & Shene, 2001) in programs but made useful for the purpose of delivering relevant data (and, perhaps, a demonstration of what sleep-sort (Anonymous, 2011) inspired triage can accomplish).

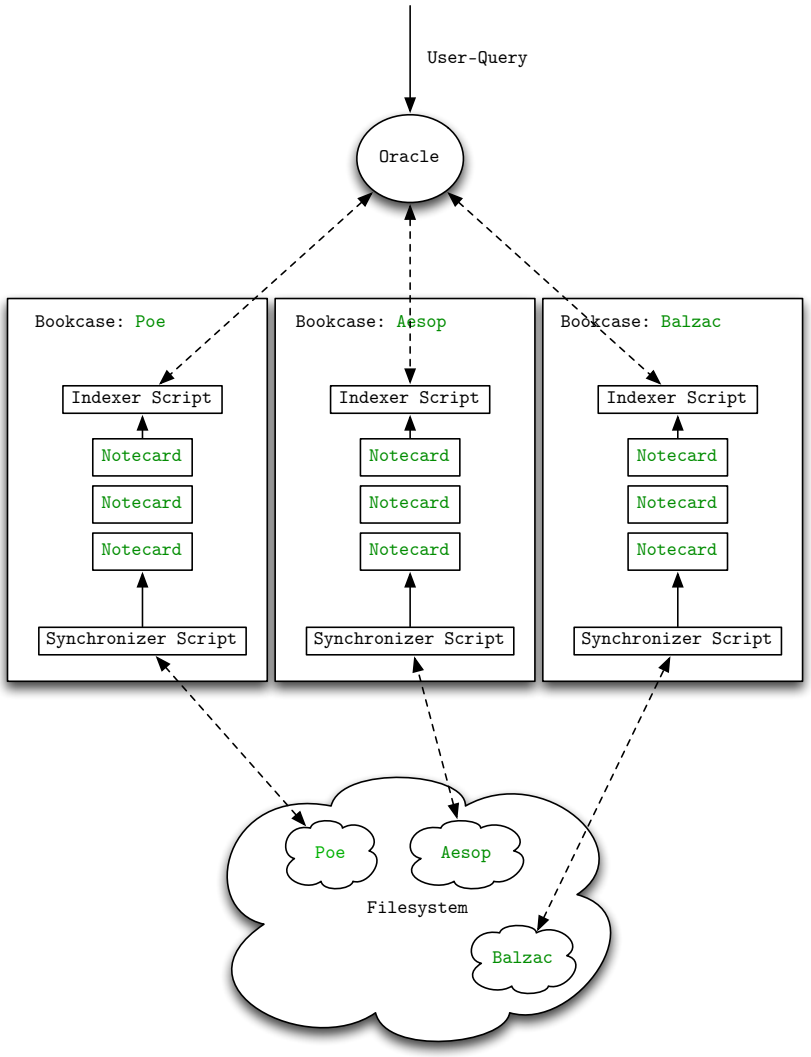

Figure 2: Whilst the Notecards are generated using the synchronizer script, the "Oracle" provides the means to query the bookcase whenever an agent searches for a keyword.

When an indexer script within a bookcase triggers a positive result for a given search-query, it sends the name and global coordinates of the bookcase to the "Oracle". The procedure of placing responses on shared communication channesl can be seen in UNIX terms as making use of data pipes. After the timeout, the "Oracle" organizes the responses from the bookcases and offers the agent a pop-up menu with the names of the bookcases. The agent can then

select an item from the menu and sit on the "Oracle" in order to quickly travel to that bookcases's location.

Additionally, all the bookcases contain a third script, a script called "giver", that hands out the Notecards whenever an agent clicks (or "touches", in Linden terms) a bookcase.

4.3 Programming Notes

The indexer, oracle and giver script are programmed in LSL while the synchronization script is written using C# and OSSL. The usage of different programming languages is intentional because OpenSim does not have C# or OSSL enabled by default. This is used to Alexandria's advantage because Alexandria can be forked into a development grid, where content is created and uploaded, and separately one can maintain a publicly accessible grid that only hands out and indexes the literature. Moving content between the development and the production grid is performed by OpenSim's region-wide export capabilities called OARs - essentially an OAR file is just a compressed tape-archive (IEEE, 1992) of assets.

4.4 Catastrophic Failure

The simple design that Alexandria uses for content addition secures the library with multiple levels of protection in the event of a catastrophic failure. First, the OAR system allows the creation of incremental backups, as well as running a mirrored-image of Alexandria on the display grid - thereby offering redundancy. Second, all the literature is stored in the cloud, so that Alexandria does not have a single point of failure (Armbrust et al., 2010). In the event that one of our contributors loses all their data, then that data will still be available on the cloud and on the machines of the contributors.

OpenSim can use either MySQL or SQLite as a storage backend. Within that database, all the assets are stored, along with the Notecards containing the literature. In the event that the database becomes corrupted, the documents are still present on the cloud and, in the event where a reconstruction is needed, the synchronization script will automatically regenerate the bookshelves without requiring human intervention. Using Dropbox for this purpose is perhaps not the best solution because if a participant chooses to delete files, then they will be deleted from all other machines. In Alexandria's case, all participants are considered trustworthy for the task of populating the library with literature.

5 Overview of The Library

Alexandria shows itself to be a tool capable of expanding and becoming a useful item for schools around the globe. The ability to perform backups, and restore items back into the grid when destroyed or removed by accident is invaluable in making sure that there is no catastrophic failure of the system, and that any

loss by a power outage is minimal. Given the virtual setting, the interaction between students and teachers may take place in a more comfortable environment that allows for an easy oversight of students using the system. Social interaction is as well, allowing groups of students to collaborate and learn in a safe and fun environment for them. Since Alexandria provides contextual clues to books, specific genres can be quickly located and used for research, reports and discussions. Furthermore, the Virtual World provides an user inventory, that allows them to take the books and visit other grids where lessons may be planned on. For example, a physics book can be taken from Alexandria and brought to a physics simulation, so that equations and principles are at hand. This eliminates the stress induced by having to search through a backpack for a specific book if forgotten in a dorm or classroom.

Alexandria's main landing point includes HyperGate access to two gate networks in OpenSim, and a link back to VIBE landing area. Since it is also a area for new users to get acclimatized, we have couches, books, and help is available for those new to virtual worlds.

Next to the landing area is a collection of popular books for browsing, so that they are easily accessible; especially, for those that are new to Virtual worlds, without getting too lost in the library. Also included is a teleport at each island along with at the art center and landing area for easy teleport between islands and areas. The art center houses a collection of works ranging from classics, such as Michelangelo, Da Vinci and up to more modern artists, such as Andy Warhol, and Picasso. A large overview of the Library is visible from the windows in the art gallery, as well as plenty of chairs to sit and look out over the main bay.

Under the art center, on a dock, is the tunnel to the underwater Science Fiction section, themed after Jules Verne's 20,000 leagues under the sea. Included with the Science Fiction, is a collection of game cheat codes and tips, arranged alphabetically, so that hopefully everyone will even be able to find something for their interest.

6 Conclusions

Alexandria contains collected literature from various sources, many literary works being supplied by the Gutenberg project (Gutenberg, 1971). At the time of writing the library spans multiple DDC classes, illustrated in Figure 3 that are displayed in either Notecard or tablet format. DDC sorting seems to be inadequate for sorting modern books by the very way it classifies them. Many derivates of literary genres such as Science Fiction, Fantasy, Teen Paranormal, Mystery, Crime Investigation, Children's Books, Self Help are not included in DDC. Each of these categories have stronger subdivisions, for example as a subset of Science Fiction such as Cyberpunk, Steampunk, and Retro book types. Under the DDC, all of these differing works and extremely different ideas are relegated to one section, Fiction, which differs by where they were written. Unfortunately, the regional classifications for DDC can be deemed unfair, consisting

Documents on Alexandria (DDC)	Number
French Satire and Humor	66
Literature	664
Poetry	4632
English Fiction	351
Ethics	269
Aristotelian Philosophy	1
Platonic Philosophy	1
Biographies	1
German Essays	90
Other Germanic Literatures	12
German Fiction	2
Fiction	243
Science and Religion	1
Genetics and Evolution	2
Drama	12
Total	6347

Figure 3: The number of documents present on Alexandria using the DCC classification at the time of writing. The poetry section appears to be the largest because the value refers to the number of poems rather than the number of authors.

of primarily European and American author sorting, dumping most of the rest of the world in large sections, for example "Asian authors", or just "Other". Many European authors are placed in Germanic sections although the country of origin may vary between Nordic countries. Another key fault with DDC is its combinations of fiction books, regardless of the content. An interesting point of confusion is the use of Romantic languages for use with only France, and Italian, a direct descendant of Latin is placed in Italian, however many european languages are spawned from Romantic or Latin roots, making that classification of romantic to only be used for France ambiguous. It seems obscure whether DDC attempts to distinguish between nationalities, or between languages, or whether the distinction is based on an attempt at cultural differentiation.

References

Anonymous. (2011). *Sleep sort.* `http://dis.4chan.org/read/prog/1295544154`.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... Zaharia, M. (2010, April). A view of cloud computing. *Commun. ACM*, *53*(4), 50–58. Retrieved from `http://doi.acm.org/10.1145/1721654.1721672` doi: 10.1145/1721654.1721672

Brabham, D. (2013). *Crowdsourcing*. Mit Press. Retrieved from http://books.google.ro/books?id=JgJWk6xY9RcC

Carr, S., Mayo, J., & Shene, C.-K. (2001, October). Race conditions: a case study. *J. Comput. Sci. Coll.*, *17*(1), 90–105. Retrieved from http://dl.acm.org/citation.cfm?id=772488.772504

Dewey, M. (1965). Dewey; Decimal classification and relative index. *New York; Lake Placid Club Education Foundation, 1965, 2 s..(2480) p. Tablas*.

FUSE. (2009). *Filesystem in userspace.* http://fuse.sourceforge.net/.

Gutenberg. (1971). *Project Gutenberg* [Digital Library]. http://www.gutenberg.org/wiki/Main_Page. Retrieved from http://www.gutenberg.org/wiki/Main_Page

Huberman, B. A., Romero, D. M., & Wu, F. (2009, December). Crowdsourcing, attention and productivity. *J. Inf. Sci.*, *35*(6), 758–765. Retrieved from http://dx.doi.org/10.1177/0165551509346786 doi: 10.1177/0165551509346786

IEEE. (1992). Ieee standards interpretations for ieee standard portable operating system interface for computer environments (ieee std 1003.1-1988). *IEEE Std 1003.1-1988/INT, 1992 Edition*. doi: 10.1109/IEEESTD.1992.106983

Kistler, J. J., & Satyanarayanan, M. (1992, February). Disconnected operation in the coda file system. *ACM Trans. Comput. Syst.*, *10*(1), 3–25. Retrieved from http://doi.acm.org/10.1145/146941.146942 doi: 10.1145/146941.146942

Ladd, G., & Mize, J. (1983). A cognitive-social learning model of social-skill training. *Psychol Rev*, *90*(2), 127-57.

Lauvset, K. A. J. (2001). Tos: Kernel support for distributed systems management. In *In proc. of the sixteenth acm symposium on applied computing, las vegas.* ACM Press.

Lee, K.-H., Slattery, O., Lu, R., Tang, X., & Mccrary, V. (2002, January). The State of the Art and Practice in Digital Preservation. *Journal of Research of the National Institute of Standards and Technology*, *107*(1), 93–106.

Linden. (2003). *Linden lab website.* http://wiki.bio-se.info.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological review*, *85*(3), 207–238.

Overte. (2011). *Opensim.* http://opensimulator.org/wiki/Main_Page.

Ranstam, J. (2008). Data handling, statistical computing, and archiving. *Acta Radiol*, *49*(10), 1137-9. Retrieved from http://www.biomedsearch.com/nih/Data-handling-statistical-computing-archiving/18608011.html

SecondLife. (2003). *Second life website.* http://www.secondlife.com.

Wang, S., Mao, Z., Zeng, C., Gong, H., Li, S., & Chen, B. (2010). A new method of virtual reality based on unity3d. In *Geoinformatics, 2010 18th international conference on* (pp. 1–5).

Wiegand, W. A. (1998). The" amherst method": The origins of the dewey decimal classification scheme. *Libraries & Culture*, 175–194.

Wu, J., Ping, L., Ge, X., Wang, Y., & Fu, J. (2010). Cloud storage as the infrastructure of cloud computing. In *Intelligent computing and cognitive informatics (icicci), 2010 international conference on* (pp. 380–383).