The ISDA Tools: Preserving 3D Digital Content

Kenton McHenry, Rob Kooper, Luigi Marini, Michael Ondrejcek

National Center for Supercomputing Applications

{kmchenry, kooper, lmarini, mondrejc}@ncsa.illinois.edu

http://isda.ncsa.illinois.edu

Abstract

In collaboration with the U.S. National Archives and Records Administration Division of Applied Research the Image, Spatial, and Data Analysis group at NCSA has developed a number of tools to aid in the preservation of digital records. In this paper we present these tools in the context of preserving 3D digital files. Tools such as the Conversion Software Registry, Software Servers, Polyglot, 3D Utilities, and Versus provide users with a scalable means of discovering and carrying out large file migration tasks in a manner that can take into account and quantify the unavoidable information loss that occurs when moving from one format to another. We present each of these tools and describe how they can be used.

Introduction

There are many different file formats to store any given content type. This is especially true in the case of 3D content where it seems that nearly every vendor of 3D software tends to come up with their own unique file format (Figure 1). In our evaluation of a number of popular 3D software packages we have documented over 144 different file formats [4]. Having many different formats for the same type of data is a problem for a couple of reasons. The first reason is a matter of accessibility in that having many formats makes it difficult to share data. If a particular format is obscure or only used by one particular software application then content created in that application might be difficult to share with users who do not have that software. The second reason is a matter of preservation. If a software vendor uses a proprietary format and does not make the format's specification open then if that vendor were to ever go out of business any content stored within that format could be potentially locked away forever. This latter situation has been known to occur.

One might argue that in an ideal world there would only be one file format for each type of content. Having one format, a format that has a standardized open specification, would make archiving that content much easier in that even if a viewer no longer exists in the future for that format a new viewer could be created based on the available specification. Determining what this format could be is a

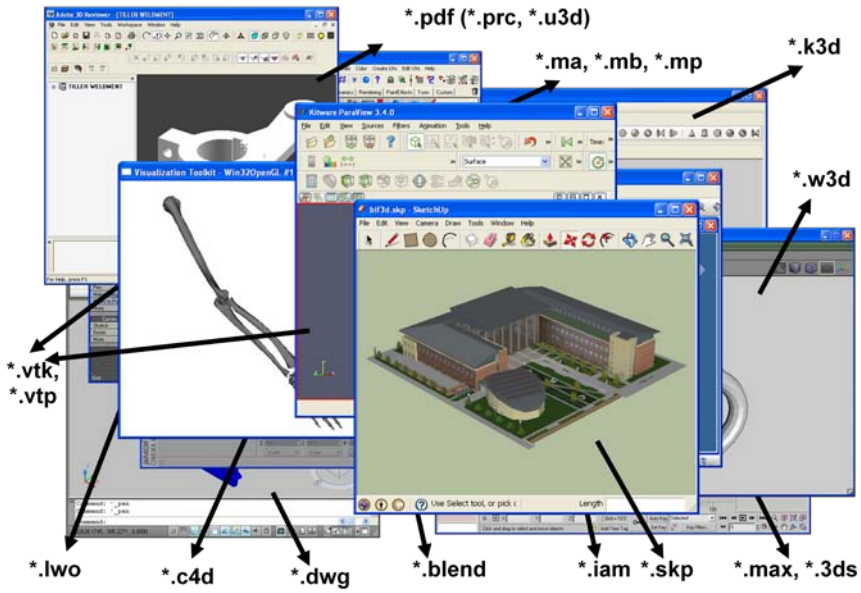

Figure 1. *Software vendors have a tendency to introduce new formats for the content that their software saves. This is especially true in the case of 3D content which has a large number of available formats today.*

problem. Even once this format is determined, converting from all the formats available today to that one format is also a problem.

As soon as the need for file format conversion arises we must begin to consider information loss. In Figure 2 we list several 3D formats along the types of information they each support. As shown here 3D files store a variety of attributes from geometry, to appearance, to scene properties and animation. Not all file formats support all attributes. Even within individual attributes information can be stored differently. For example the geometry of a model is often represented as either a faceted mesh, as parametric surfaces, as a boundary representation, or as constructive solid geometry. Converting from a format that supports one representation to one that supports another representation requires a conversion of the content itself. Some of these conversions are possible. For example B-Rep surfaces can be transformed to faceted meshes by a process called tessellation. Doing this will alias an otherwise continuous surface and depending on the sampling drastically change the file size. The tessellation process will result in a loss of information in that the inverse process, going from a faceted mesh to a B-Rep, will not result in the original content. In fact this inverse conversion is not trivial at all.

Keeping in mind that conversions will almost always result in some sort of information loss we can then begin the process of determining what an optimal file

Format	Geometry				Appearance				Scene				Animation
	Faceted	Parametric	CSG	B-Rep	Color	Material	Texture	Bump	Lights	Views	Trans.	Groups	
3ds	√	√			√	√	√	√	√	√	√		
igs	√	√	√	√	√						√	√	
lwo	√	√			√	√	√	√					
obj	√	√			√	√	√	√				√	
ply	√				√	√	√	√					
stp	√	√	√	√	√							√	
wrl	√	√			√	√	√	√	√	√	√	√	√
u3d	√				√		√	√	√	√	√	√	√
x3d	√	√			√	√	√	√	√	√	√	√	√

Figure 2. *While there are a variety of formats to store 3D content each can store information a little differently. Common attributes stored within 3D files are geometry, appearance, and scene structure. Each attribute can also be stored in a number of ways. Converting between formats that do not support the same type of information will result in some sort of information loss.*

format for long term preservation is. We define this format to be one that is standardized/open and results in as little information loss as possible when converted to from all other available formats. Such a format would have the best chance of keeping the bulk of an archives data accessible in the future.

Below we present a number of tools we have developed towards the end goal of empirically evaluating which format is optimal for long term preservation for a given content type. We present these tools individually as they possess useful qualities on their own right, separate from the overall goal of choosing an optimal file format.

3D Utilities

Our 3D Utilities are both a library and a collection of tools written in Java created for the purpose of accessing content within various 3D file formats. Like the NCSA Portfolio project before it, 3D Utilities provides a number of 3D file loaders. Unlike Portfolio which was built on Java3D and loaded content in a Java3D scene graph, 3D Utilities takes a far simpler and less restrictive approach. 3D Utilities uses an extremely simple polygonal mesh representation for all its 3D content. File loaders for various formats are created so as to parse a 3D file type and load its content into a polygonal mesh. If a file format does not store its 3D geometry as a mesh then it is up to the loader to convert between representations. This library of loaders can be used by any java application to load 3D content from a file for the purpose of rendering or manipulating it through the mesh data structure (which is nothing more than a list of vertices and faces connecting them).

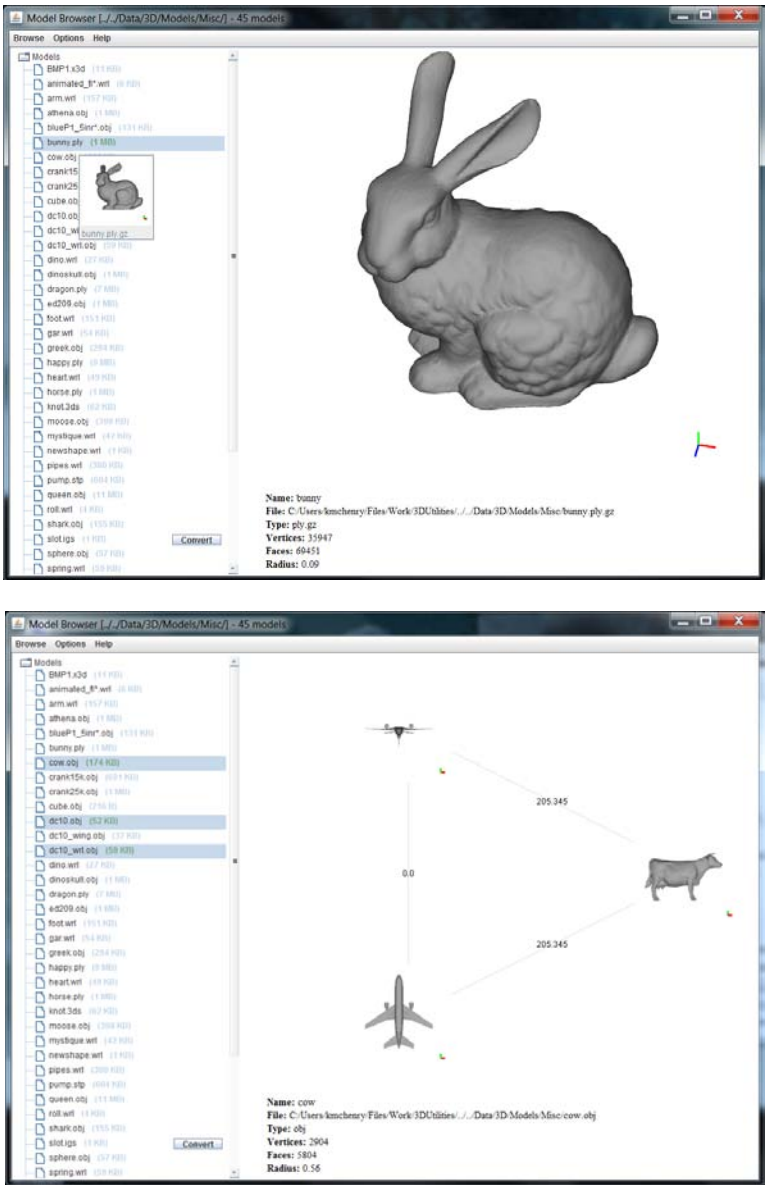

Figure 3. Top: *The Model Browser tool included as part of the 3D Utilities allows one to view 3D files under a given directory and manipulate data within a number of different 3D file formats.* ***Bottom:*** *If a user selections multiple files from the left hand side the 3D content from each file is displayed and compared to one another. Above 3 models are compared using the light fields measure [2], a measure that is invariant to rigid transformations such as rotations. Notice the two planes have a distance of 0 between them even though they are rotated differently.*

In addition to file loaders the 3D Utilities library also contains a library of mesh signatures. These mesh signatures act as a hash allowing one to compare two different 3D models and retrieve the most similar instance from a collection of 3D models. We have signatures implemented based on vertices statistics, polygonal face surface area [1], light fields [2], and spin images [3]. Each signature allows for different aspects of the models to be considered during a comparison and each is best suited under differing situations. These signatures can be used within Java code to compare two meshes loaded using the library of loaders.

3D Utilities also contains several tools: ModelViewer, ModelViewerApplet, ModelBrowser, and ModelConverter. The ModelViewer is both a class extending a JPanel, which can be used within code to display 3D content, and a standalone application to view content within 3D files. The ModelViewerApplet is an applet version of the ModelViewer allowing 3D content to be viewable from within a web browser. The ModelBrowser tool uses ModelViewers to provide a convenient way of viewing all 3D files under a given file system directory. As shown in Figure 3 (top) all 3D files found under a user specified directory are shown in the leftmost panel. From here a user can select one of the files to be displayed in the top right panel. This panel being an instance of a ModelViewer allows a user to use the mouse to change viewing directions and manipulate the model. In the bottom right panel metadata of the shown 3D model is displayed. If multiple files are selected in the left pane they are compared using a specified signature and shown simultaneously in the right pane with distances along edges that connect them. This can be used as a means of visualizing how the various signatures emphasize different aspects of a 3D model during comparison (Figure 3 bottom). The last tool, the ModelConverter, utilizes the 3D Utilities file loaders to perform format conversions. As all loaders load/save content to/from a mesh representation it is a simple matter to convert between formats by loading the content using one loader and saving it using another.

The Conversion Software Registry

As stated previously vendors have a tendency to create new file formats. For the sake of some level of portability applications often allow content to be imported/exported to and from a handful of other formats. This tendency allows many software applications to act as converters between different formats. As many formats are closed source and proprietary the software of the vendor responsible for the format is often the best means of getting into and out of that particular format. As many applications only support a small number of imports/exports it is also very likely that many desirable conversions from specific source formats to target formats won't be supported. However, it is very possible that multiple applications chained together can carry out conversions though intermediary formats that are not directly supported by any one application.

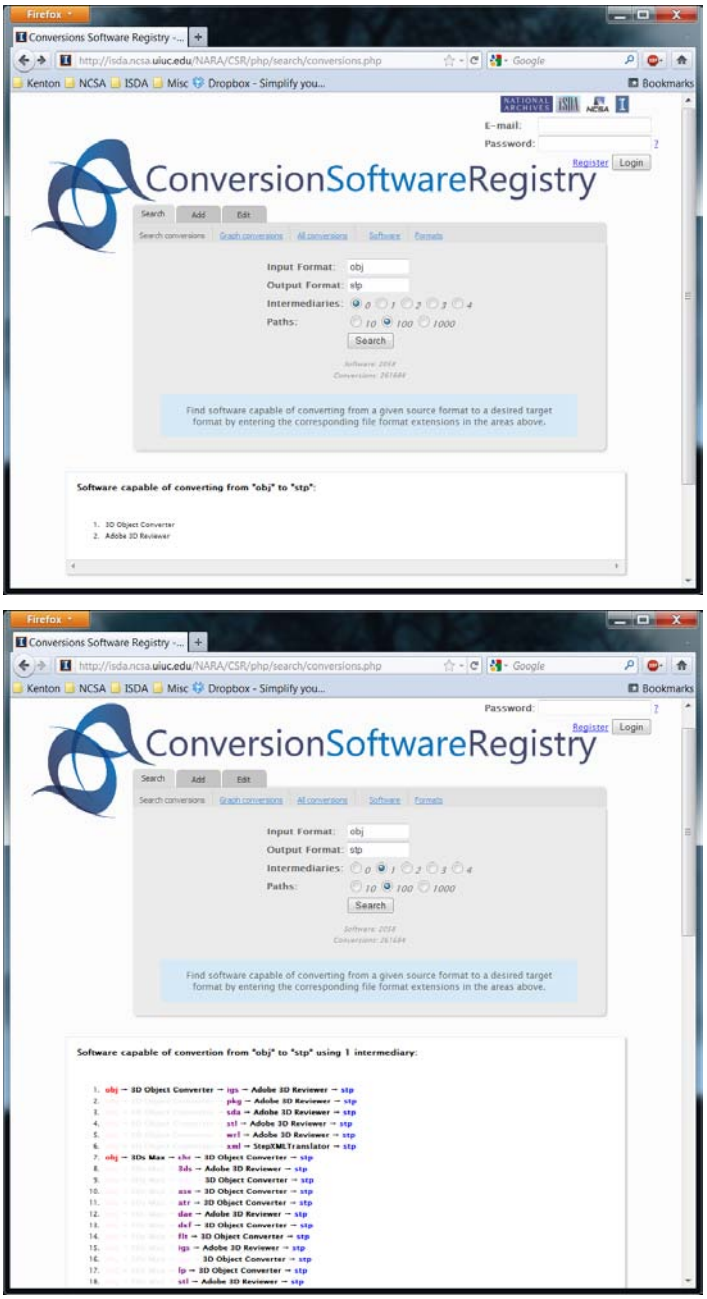

Figure 4. *The Conversion Software Registry.* ***Top:*** *A user searching for software based on supported for input format *.obj and output format *.stp.* ***Bottom:*** *A user searching for chains of software supporting this same input and output format, however, this time with one allowed intermediary format.*

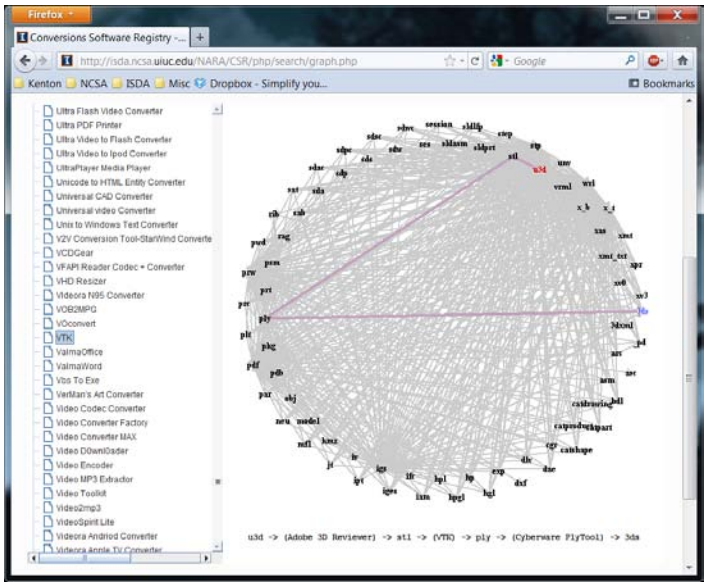

Figure 5. *A user can search the data within the CSR graphically. After selecting a number of applications to consider in the left the user can use the mouse to select a source and target file format. If a means of converting between the source and target file format exists the shortest chain of software capable of carrying it out will be displayed.*

This information, in terms of software inputs and outputs, is essential in order to identify software to carry out a specific conversion. Determining what software is needed for a particular conversion, however, can be difficult without direct access to the software (i.e. without purchasing the software). To aid users in this task we have created the Conversion Software Registry (CSR), an online database which indexes software based on the input and output formats they support [9]. As shown in Figure 4 (top) a user can type in the extension of a source format and target format to get back a list of applications that directly supports that conversion. By clicking a radio button below the query box a user can tell the system to consider conversions using a one or more intermediary formats. As shown in Figure 4 (bottom) many more options are presented by doing this.

The CSR also allows users to search through the space of possible conversions graphically (Figure 5). Applications selected in the left pane are represented in the right pane as an input/output graph. This graph has at its vertices file formats and directed edges connecting a pair of formats to represent an application capable of carrying out that particular conversion. A user can search for a conversion path between formats by using their mouse to select a source and target file format. An un-weighted shortest path algorithm is then used to find a conversion path with the least number of intermediary formats. Other types of supported queries include: finding all the formats reachable from a given source format, finding all the

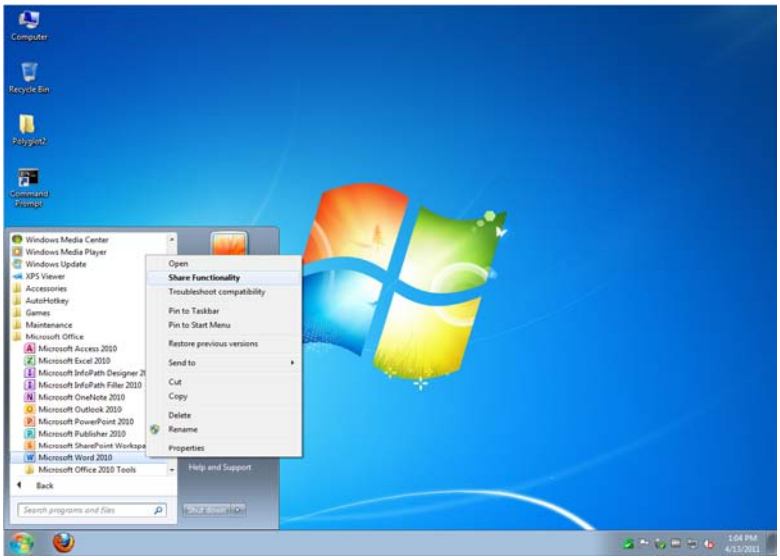

Figure 6. *Software functionality can be shared in a manner analogous to how folders are shared in the Windows operating system. Access to operations within software can be shared simply by right clicking on them and selecting share from the menu.*

formats that can reach a particular target format, and finding a set of file formats that can be commonly reached by a selected set of source formats.

Software Servers

Ideally we would like to be able to support format conversions within code (e.g. Java). However because of the many closed propriety formats in existence this is not at all feasible. As pointed out in the previous section, however, there is software available to carry out many conversions. A Software Server is a tool that attempts to bridge this gap between need and availability [4, 5, 7]. A Software Server running on a system allows functionality within locally installed software to be shared in a matter that is analogous to how Windows allows one to share a folder. To share functionality within software (e.g. the open and save operations within a particular program) one need only right click on the shortcut of the application and select "Share Functionality" from the menu (Figure 6). Once this is done information obtained from the shortcut is used to query the Conversion Software Registry which serves as a repository for wrapper scripts that are capable of automating functionality within command line and graphical interface driven software. These scripts can be written in any text based scripting language. We tend to use AutoHotKey [1]and Sikuli [10] as they allow for the scripting of graphical interfaces (making a large amount of software accessible from the Software

[1] http://www.autohotkey.com

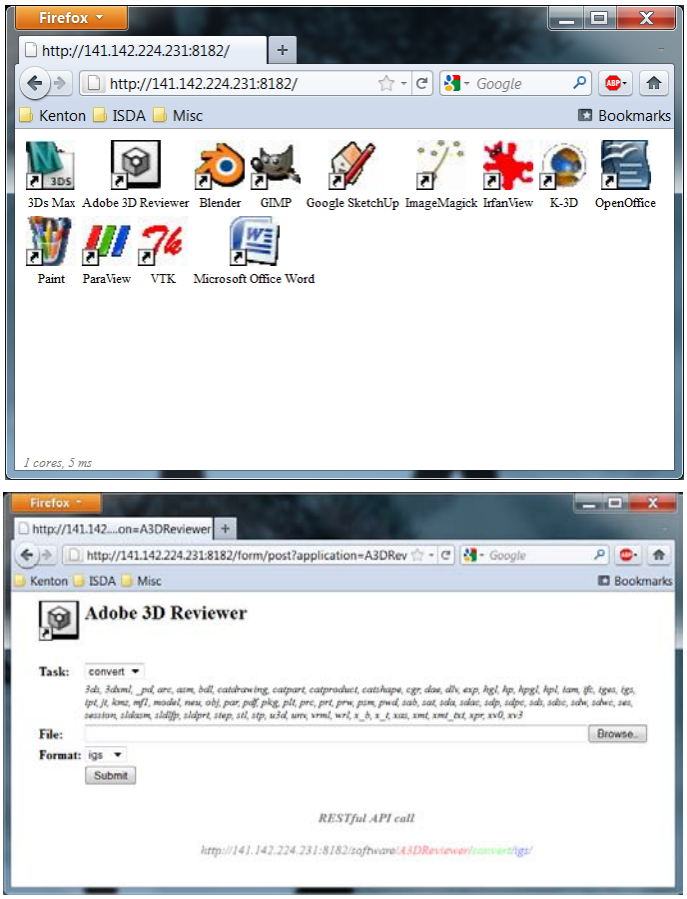

Figure 7. *Shared software functionality can be accessed from a browser by accessing the host address of the machine running the software server.* ***Top:*** *If accessed directly from a browser the software is presented in a manner that resembles a file manager from a modern OS with software represented by icons.* ***Bottom:*** *When an icon is clicked on the user is presented with a form that allows them to access the software functionality.*

Server). Wrapper scripts associated with the needed software are downloaded and configured to run on the local environment.

Once a Software Server is running shared software functionality can be accessed by simply pointing a web browser to the address of the hosting machine (Figure 7 top). When this is done a user is presented with a page that mimics modern file managers with a number of icons representing available software. When a user clicks on a particular piece of software they are presented with a form that allows them to call the remotely shared functionality from the browser (Figure 7 bottom). Tasks always take the form of a quadruple of the form: software, task, output format, input file. Though the form presents this in an easy to use graphical manner,

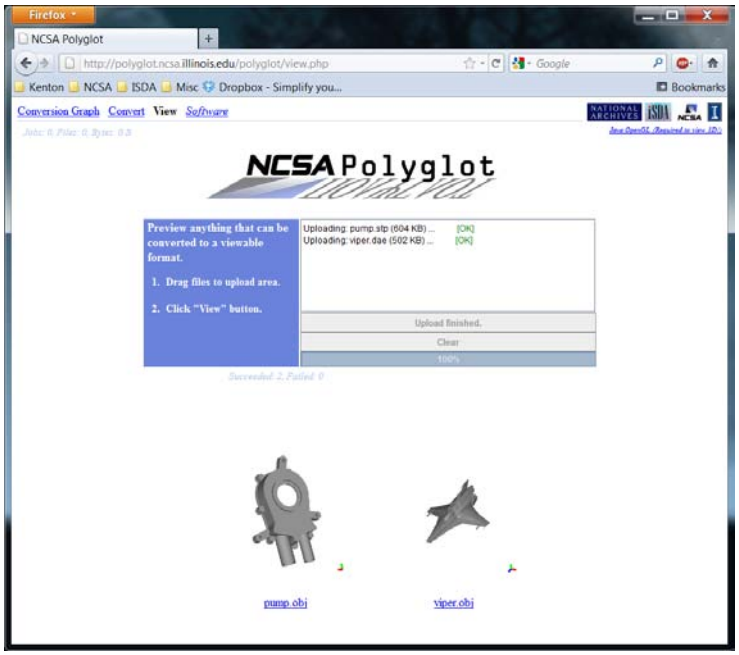

Figure 8. *A web interface to NCSA Polyglot. A user can drag and drop files to the large area in the middle. In this particular case the user does not select the desired output format as it is set automatically to one which can be viewed within the browser. When the users presses the "View" button a conversion path is executed across the underlying Software Servers to reach the desired target format. When completed the results are displayed in the area below.*

the strength of the Software Server is in the RESTful interface it provides to carry out these tasks. Specifically the same task can be carried out by posting a file to a URL on the host in the form of:

http://<host>:8182/software/<**Software**>/<**Task**>/<**Output**>/

where the final element of the quadruple, the input file, is the posted file. This simple, consistent, widely accessible interface to software allows software functionality to be used within new code as if it were a function within a library. Any language that supports accessing URLs can access this functionality, possibly wrapping it to look like a native library.

Polyglot

NCSA Polyglot [6] is a conversion service built on top of the previously described Software Servers. When Software Servers come online they begin to broadcast their existence. A Polyglot instance listens for these broadcasts. When a new Software Server is found Polyglot will query it for all shared software

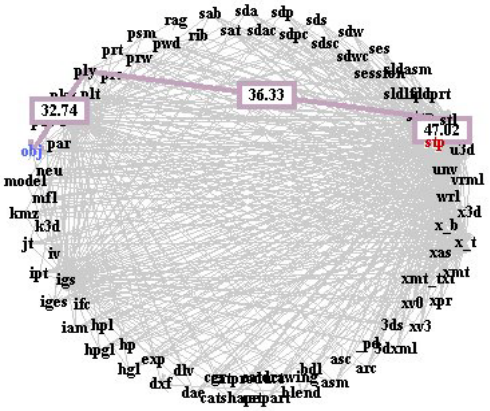

Figure 9. *A weighted input/output graph, where the weights represent information loss obtained using Versus on a sample set. A shortest weighted path algorithm can be used to find conversion paths that result in the least amount of overall information loss.*

functionality, looking for software with available input and output operations to carry out conversions. From this information Polyglot constructs an input/output graph. This graph is searched in order to carry out conversions across chains of software based on a given input format and desired output format.

Polyglot can be accessed through either a Java API or a web interface which allows a user to drag and drop a number of files from their local machine, select an output format, and carry out the coversions with the results being made available for download (Figure 8).

Versus

Versus is a framework/library of content to content comparison measures supporting raster images, vector graphics, 3D models (via the signatures from our 3D Utilities), and documents. These measures are used by Polyglot as a means of estimating the information loss incurred by converting from one format to another through a piece of software. Information loss is evaluated in Polyglot using a sample set of files in a format that can be directly loaded by one of the loaders in our 3D Utilities library. These files are converted to every reachable format within an input/output graph and then converted back to the original. We are forced to do execute through these A-B-A' conversion paths in order to compare the file's contents before and after the conversion. Each before and after file is compared using a set measure from Versus and the returned similarity is averaged along each edge representing software in the graph (Figure 9). This weighted graph can be used by Polyglot to identify conversions paths with minimal information loss. In addition this weighted graph can be used to derive an answer to the question of which format would be best suited for long term preservation (as described earlier).

Conclusion

The ISDA tools are a collection of libraries and software constructed for the purpose of providing solutions to problems in digital preservation. We have presented five of these tools: 3D Utilities, the Conversion Software Registry, Software Servers, Polyglot, and Versus. These and others are available from our site at http://isda.ncsa.illinois.edu as free open source software.

Acknowledgments

This research has been funded through the National Science Foundation Cooperative Agreement NSF OCI 05-25308 and Cooperative Support Agreement NSF OCI 05-04064 by the National Archives and Records Administration (NARA).

References

1. S. Brunnermeier and S. Martin, “Interoperability cost analysis of the U.S. automotive supply chain,” RTI International Research Publications, March 1999.

2. D. Chen, X. Tian, Y. Shen, and M. Ouhyoung, “On visual similarity based 3d model retrieval,” Eurographics Computer Graphics Forum, 2003.

3. A. Johnson and M. Hebert, “Using spin images for efficient object recognition in cluttered 3d scenes,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999.

4. K. McHenry, R. Kooper, M. Ondrejcek, L. Marini, P. Bajcsy, “A Mosaic of Software,” IEEE eScience, 2011.

5. K. McHenry, R. Kooper, L. Marini, P. Bajcsy, “Designing a Scalable Cross Platform Imposed Code Reuse Framework,” Microsoft eScience Workshop in San Francisco, CA, 2010.

6. K. McHenry, R. Kooper, and P. Bajcsy, “Towards a universal, quantifiable, and scalable file format converter,” The IEEE Conference on e-Science, 2009.

7. K. McHenry, R. Kooper, and P. Bajcsy, “Taking matters into your own hands: Imposing code reusability for universal file format conversion,” The Microsoft e-Science Workshop, 2009.

8. K. McHenry and P. Bajcsy, “An overview of 3d data content, file formats and viewers,” Technical Report ISDA08-002, 2008.

9. M. Ondrejcek, K. McHenry, and P. Bajcsy, “The conversion software registry,” The Microsoft e-Science Workshop, 2010.

10. T. Yeh, T. Chang, and R. Miller, “Sikuli: Using gui screenshots for search and automation,” UIST, 2009.