Introducing Zoomify Image

Adam Smith

Zoomify Image is a mature product for easily publishing large, high-resolution images on the Web. End users view these images with existing Web-browser software as quickly as they do normal, downsampled images. A Flash-based Zoomifyer client asynchronously streams image data to the Web browser as needed, resulting in response times approaching those of desktop applications using minimal bandwidth. The author, a librarian at Cornell University and the principal architect of a small, open-source company, worked closely with Zoomify to produce a cross-platform, open-source implementation of that company's image-processing software and discusses how to easily deploy the product into a widely used Web-publishing environment. Limitations are also discussed as are areas of improvement and alternatives.

Zoomifyer from Zoomify (www.zoomify.com) enables users to view large, high-resolution images within existing Web-browser software while providing a rich, interactive user experience. A small Zoomifyer client, authored in Macromedia Flash, is embedded in an HTML page and makes asynchronous requests to the server to stream image data back to the client as needed. By streaming the image data in this way, the image renders as quickly as a normal, downsampled image, even for images that are gigabytes in size. As the user pans and zooms, the response time approaches that of desktop applications while using the smallest possible bandwidth necessary to render the image. And because Flash has 98.3 percent browser saturation, viewing "Zoomified" images is seamless for most users and allows them to view images interactively in much greater detail than would otherwise be practical or even possible.[1]

Zoomify Image (sourceforge.net/projects/zoomifyimage) was created at Cornell University in collaboration with Zoomify to create an open-source, cross-platform, and scriptable version of the processing software that creates the image data displayed in a Zoomifyer client. This work was immediately integrated into an innovative content-management system that was being developed within the Zope Application Server, a premier Web application and publishing platform. Authors in this system can add high-resolution images just as they normally add downsampled images, and the image is automatically processed on the server by Zoomify Image and displayed within a Zoomifyer client. Zoomify Image is now in its second major release on Source Forge and contains user contributed software to easily deploy it in other environments such as PHP.

Zoomifyer has been used in a number of applications in many fields, and can greatly enhance many research and instructional activities. Applying Zoomifyer to digital-image collections is obvious, allowing libraries to deliver an unprecedented level of detail in images published to the Web. New applications also suggest themselves, such as serving high-resolution images taken from tissue samples in a medical lab or using Zoomifyer in advanced geospatial image applications, particularly when advanced client features such as annotations are used.

The Zoomifyer approach also has positive implications for preservation and copyright protection. Zoomify Image generates cached derivatives of master image files so the image masters are never directly accessed in the application or sent over the Internet. Image data are stored and transmitted to the client in small chunks so that end users do not have access to the full data of the original image.

Deploying Zoomify Image

Dependencies and winstallation

Zoomify Image was designed initially to be a faithful, cross-platform port of Zoomify's image-processing software. It was developed in close cooperation with Zoomify to provide a scriptable method for invoking the image-preparation process for Zoomifyer clients so this technology could be used in more environments.

Zoomify Image is written in the Python programming language and uses the third-party Python Imaging Library (PIL) with JPEG support, both of which are also open source and cross-platform. It has been tested in the following environments:

- Python 2.1.3
- PIL 1.1.3

and

- Python 2.4.3
- PIL 1.1.4

Installers for Python and PIL exist for all major platforms and can be obtained at python.org and www.pythonware.com/products/pil.

The installation documentation that comes with PIL will help you locate the appropriate JPEG libraries if they are missing from your system. For MacOSX, you can find pre-built binary installers for Python, PIL and Zope at sourceforge.net/projects/mosxzope.

Adam Smith (ajs17@cornell.edu) is a Systems Librarian at Cornell University Library, Ithaca, New York.

The "EZ" version of the Zoomifyer client, a Flash-based applet with basic pan and zoom functionality, is packaged with Zoomify Image for convenience so the software can be used immediately once installed. The EZ client is covered by a separate license and can be easily replaced with more advanced clients from Zoomify at www.zoomify.com. (A description of how to upgrade the Zoomifyer client is included in this paper.)

After Python and PIL with JPEG support are installed, download the Zoomify Image software from sourceforge.net/projects/zoomify-image and decompress it.

Using Zoomify Image from the command line

Begin exploring Zoomify Image by invoking it on the command line:

```
python <your_path>/ZoomifyFilePr
ocessor.py <your_image_file>
```

Or, to process more than one file at a time:

```
python <your_path>/ZoomifyFile
Processor.py <image_1> <image_2>
<image_3>
```

The file format of the images input to Zoomify Image are typically either TIFF or JPEG, but can be any of the many formats that PIL can read.[2] An image called "test.jpg" is included in the Zoomify Image distribution and is of sufficient size and complexity to provide an interesting example.

During processing, Zoomify Image creates a new directory to hold the converted image data in the same location as the image file being processed. The name of this directory is based on the file name of the image being processed, so that, for example, an image called "test.jpg" would have a corresponding folder called "test" containing the converted image data used by the Zoomifyer client. If the image file has no file extension, the directory is named by appending "_data" to the image name, so that an image file named "test" would have a corresponding directory called "test_data." If the process is re-run on the same images, any previously generated data are automatically deleted before being regenerated.

Zoomify provides substantial documentation and sample code on its Web site that demonstrates how to use the data generated by Zoomify Image in several environments. User-contributed code is bundled with Zoomify Image itself, further demonstrating how to dynamically incorporate this conversion process into several environments. An example of the use of Zoomify Image within the Zope Application Server is given.

Incorporating Zoomify Image into the Zope Application Server

The popular Zope Application Server contains a number of built-in services including a Web server, FTP and WebDAV servers, plug-ins for accessing relational databases, and a hierarchical object-oriented database that uses a file-system metaphor for storage. This object database provides a unique opportunity to incorporate Zoomifyer into Zope seamlessly.

To use Zoomify Image with Zope, the distribution must be decompressed into your Zope Products directory. For versions 2.7.x and up, this is at:

```
<zope_instance_home>/Products/
```

In Zope versions prior to the 2.7.x series, the Products directory is at:

```
<zope_home>/lib/python/
Products/
```

Restart Zope and now within the Web-based Zope Management Interface (ZMI), the ability to add Zoomify Image objects appears. After selecting this option, a form is presented that is identical to the form used for adding ordinary Image objects within Zope. When an image is uploaded using this form, Zope automatically invokes the Zoomify Image conversion process on the server and links the generated data to the default Zoomifyer client that comes with the distribution. If the image is subsequently edited within ZMI to upload a new version, any existing conversion data for that image are automatically deleted, and the new conversion data are generated to replace them, just as when invoked on the command line.

Again, the uploaded image can be in any format that Zope recognizes as having a content-type of "image/..." and that PIL can read. The only potential "gotcha" in this process is that in the versions of the Zoomifyer client the author has tested, Zoomify Image objects that have file names (in Zope terminology, the file name is the object's "id" property) with extensions other than ".jpg" are not displayed properly by the Zoomifyer client. So, when uploading a TIFF image, for example, the id given to the Zoomify Image object should either not contain an extension, or it should be changed from image.tif to something like image_tif. This bug has been reported to Zoomify and may be fixed in newer versions of the Flash-based viewing software at the time of publication.

To view the image within the Zoomifyer client, simply call the "view" method of the object from within a browser. So, for a Zoomify Image object uploaded to:

```
http://<your_domain>/test/test.jpg
```

go to this URL:

```
http://<your_domain>/test/test.
jpg/view
```

Or, to include this view of the image within a Zope Page Template

(ZPT), simply call the tag method of the Zoomify Image just as you would a normal Image object in Zope. So, in a ZPT, use this:

<tal:block replace="here/test.jpg/tag" />

It is possible that the Zoomify Image conversion process will not have had time to complete when someone tries to view the image. The Zoomify Image object will attempt to degrade gracefully in this situation by trying to display a downsampled version of the image that is generated part way through the conversion process, or, if that is also not available, finally informing the user that the image is not yet ready to be viewed. This logic is built into the tag method.

To add larger images more efficiently, or to add images in bulk, the Zoomify Image distribution contains detailed documentation to quickly configure Zope to accept images via FTP or WebDAV and automatically process them through Zoomify Image when they are uploaded.

Finally, the default Zoomifyer client can be overridden by uploading a custom Zoomifyer client into a location where the Zoomify Image object can "acquire" it, and giving it a Zope id of "zoomifyclient.swf".

How it works

To be viewed by a Zoomifyer client, an image must be processed to produce tiles of the image at different scales, or tiers. An XML file that describes these tiles is also necessary. Zoomify Image provides a cross-platform method of producing these tiled images and the XML file that describes them.

Beginning at 100-percent scale, the image is successively scaled in half to produce each tier, until both the width and height of the final tier are, at most, 256 pixels each. Each tier is further divided into tiles that are, at most, 256 pixels wide by 256 pixels tall, as seen in figure 1.

These tiles are created left to right, top to bottom. Tiles are saved as images with the naming convention indicated in figure 2.

The numbering is zero-based, so that the smallest tier is represented by one tile that is at most 256 x 256 pixels wide with the name "0-0-0.jpg."

Tiles are saved in directories in groups of 256, and those directories also follow a zero-based naming convention starting with "TileGroup0." Lower-numbered tile groups contain lower-numbered tiles, so 0-0-0.jpg is always in TileGroup0.

Zoomifyer clients understand this tile-naming scheme and only request tiles from the server that are necessary to stitch together the portion of the image being viewed at a particular scale.

Limitations

Zoomify Image was developed to meet two goals:

1. to provide a cross-platform port of the Zoomifyer con-

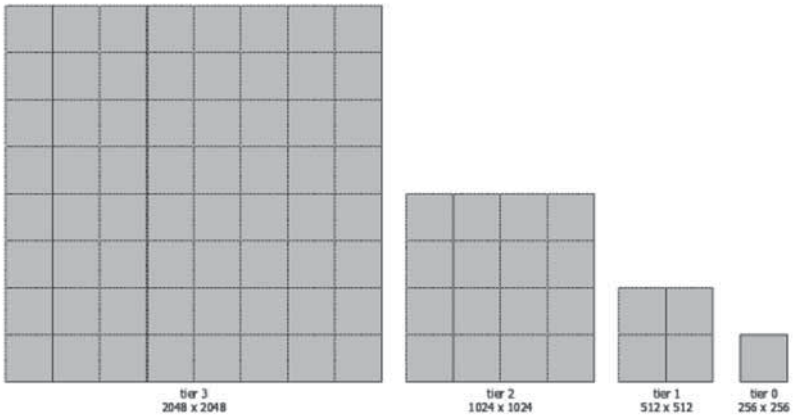

Figure 1. Tiers and tiles for a 2048 x 2048 pixel image

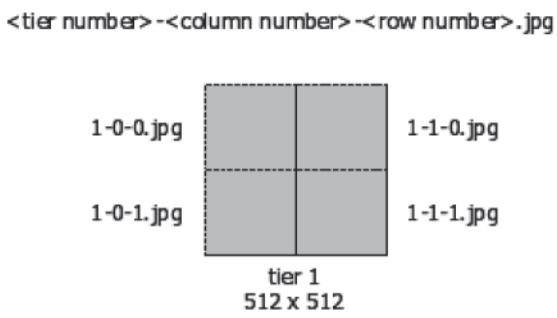

Figure 2. Tile image naming scheme

verter for use in UNIX/Linux systems, and
2. to make the converter scriptable, and ultimately integrate it into open-source content-management software, particularly Zope.

This Zoomifyer port was written in Python, a mature, high-level programming language with an execution model similar to Java. Although Zoomify Image continues to be optimized, compared to the official Zoomify conversion software, it is slower and more limited in the sizes of images it can reasonably process. Anecdotally, Zoomify Image has been used effectively on images hundreds of megabytes large, but significant performance degradation has been reported in the multi-gigabyte range.

Because of these limitations in Zoomify Image, the official Zoomify image-processing software is recommended for converting very large images manually in a Windows or Macintosh environment. The Zoomify Image product is recommended in the following circumstances:

- The conversion must be performed on a UNIX/Linux machine.
- The conversion process must be scriptable, such as for batch processing or being run dynamically.
- Images sizes are not in the multi-gigabyte range.

If a scriptable, cross-platform version of the Zoomifyer converter is needed, but performance is an issue, several things can be done to extend the current limits of the software. Obviously, upgrading hardware, particularly RAM, is effective and relatively inexpensive. Running the latest versions of Python and PIL will also help. Each new version of Python makes significant performance improvements, and this was a primary goal of version 2.5, which was released in September 2006.

The author believes that the current weak link in the performance chain is related to how Zoomify Image is loading image data into memory with PIL during processing. In the current distribution, a Python script contributed by Gawain Avers, which is based partially on the Zoomify Image approach, uses ImageMagick instead of PIL for image manipulation and is better able to process multi-gigabyte images. The author would like to add the ability to designate the image library at runtime in future versions of Zoomify Image.

Future development

Beyond improving the performance of the core-processing algorithm, the author would also like to explore opportunities for more efficiently processing images within Zope, such as spawning a background thread for processing images so the Zope Web server can immediately respond to the client's image-submission request. The author would also like to improve the tag method to display data more flexibly in the Zoomifyer client and ensure consistent behavior with Zope's default Image tag method. Finally, Zoomify Image could also benefit from the addition of a simple configuration file to control such runtime properties as image quality and which third-party image-processing library to use, for example.

Conclusion

Zoomify Image is mature, open-source software that makes it possible to publish large, high-resolution images to the Web. It is designed to be convenient to use in a variety of architectures and can be viewed within existing browser software. Download it for free, begin using it in minutes, and explore its unique possibilities.

References

1. Adobe Systems, Macromedia Flash Player Statistics, http://www.adobe.com/products/player_census/flashplayer/ (accessed March 1, 2007).

2. PythonWare, *Python Imaging Library Handbook: Image File Formats*, http://www.pythonware.com/library/pil/handbook/formats.htm (accessed Aug. 6, 2006).

Resources

Macromedia Flash Player Statistics (http://www.adobe.com/products/player_census/flashplayer/) (accessed Jan. 2, 2007).

Python Imaging Library (PIL) (http://www.pythonware.com/products/pil/) (accessed Jan. 2, 2007).

Python Programming Language Official Web site (http://www.python.org/) (accessed Jan. 2, 2007).

Zoomify Image (http://sourceforge.net/projects/zoomifyimage/) (accessed Jan. 2, 2007).

Zoomify (http://www.zoomify.com/) (accessed Jan. 2, 2007).

Zope Community (http://www.zope.org/) (accessed Jan. 2, 2007).

Zope installers for MacOSX (http://sourceforge.net/projects/mosxzope/) (accessed Jan. 2, 2007).