AN ALTERNATIVE TO CONVENTIONAL METHODS IN DIGITAL-IMAGE VIEWING

Submitted by:__________________________________

Yetu A. Yachim

A Thesis Approved On

___________________________________
(Date)

by the Following Reading and Examination Committee:

___________________________________
Dr. Eric C. Rouchka, Thesis Director

___________________________________
Dr. Dar-jen Chang

___________________________________
Dr. Hichem Frigui

___________________________________
Dr. R. Eric Benson
ACKNOWLEDGEMENTS

I am indebted to those who have provided me with the technical knowledge and resources needed to complete this project: Dr. Eric Rouchka, Dr. Dar-jen Chang, and Dr. Hichem Frigui. To the friends, students and faculty alike, who have helped me through the Speed School undergraduate program I give many thanks. And, to my parents, without whose support I would not have realized any of my achievements, I offer my deepest gratitude.
ABSTRACT

Digital images have primarily been viewed using desktop applications. These types of programs attempt to load all of an image’s graphical data into memory in order to display the image in its entirety. The method is effective for images that are relatively small, but for large, high-resolution images, this tends to be slow and resource-exhaustive. Coupled with degraded system performance are the issues of storage and image access. Some newer programs make efforts to mitigate these problems, but few do so satisfactorily. The Ymage System was designed as an alternative to conventional approaches for managing and viewing images. The system provides high accessibility to data while minimizing demands on the machine from which the image is viewed. The design accomplishes these objectives and even the initial, low-performance implementation presented here can compete with existing systems. Further, it appears that the flexibility of the design lends itself to use for non-visual applications such as distributed image processing. It is possible that further development of the Ymage System could make a significant impact on the approach to the development of image-handling software in the future.
# TABLE OF CONTENTS

**APPROVAL PAGE** ................................................................. ii  
**ACKNOWLEDGEMENTS** ........................................................... iii  
**ABSTRACT** ........................................................................ iv  
**NOMENCLATURE** ................................................................. vi  
**LIST OF TABLES** ................................................................. vii  
**LIST OF FIGURES** ............................................................... viii  

## I  INTRODUCTION  
A. Digital Images ......................................................................... 1  
  A.1 Image File Storage ............................................................ 2  
  A.2 Image File Retrieval .......................................................... 5  
B. Applications of Digital Images ................................................. 5  

## II  EXISTING SOFTWARE  
A. Standalone/Desktop Model .................................................... 8  
  A.1 Picasa™ ........................................................................... 8  
  A.2 ImageQuery ....................................................................... 12  
  A.3 Photo Organizer Deluxe ..................................................... 14  
B. Client-Server Model ............................................................... 16  
  B.1 NASA World Wind ........................................................... 16  
  B.2 Zoomify ............................................................................ 17  

## III THE YMAGE SYSTEM  
A. System Architecture ............................................................ 20  
  A.1 Data Storage Component .................................................. 23  
  A.2 Data Access Component ................................................... 24  
  A.3 Visualization Component ................................................. 25  
B. Implementation ...................................................................... 26  
  B.1 Development Environment and Tools .................................. 26  
  B.2 Data Storage using PostgreSQL ........................................ 28  
  B.3 Apache HTTP for Data Access ......................................... 31  
  B.4 Java Applet Viewer .......................................................... 36  
C. Evaluation and Possible Improvements ................................... 38  

## IV RESULTS  

## V CONCLUSION  

**REFERENCES** ........................................................................ 46  
**APPENDIX A: Image Subdivision Process** ............................. 48  
**APPENDIX B: SQL Database-Schema Script** .......................... 50  
**APPENDIX C: Source Code Listing for Apache Module** .......... 53  
**APPENDIX D: Source Code Listing for Visualization Applet** .... 72  
**APPENDIX E: Images Tested** ................................................... 116
LIST OF TABLES

TABLE 1 - Image Upload Process .......................... 33
TABLE 2 - Images Tested ................................ 41
TABLE 3 - Desktop Application Load Times ............. 42
TABLE 4 - Ymage System Test Results .................. 42
TABLE 5 - Zoomifyer EZ™ Test Results ................. 43
LIST OF FIGURES

FIGURE 1 – Picasa™, searching for images ......................... 9
FIGURE 2 – Viewing a large image in Picasa™ ....................... 10
FIGURE 3 – Picasa™ image upload failure .............................. 11
FIGURE 4 – ImageQuery, searching for images by metadata ......... 13
FIGURE 5 – Photo Organizer, database designer ..................... 14
FIGURE 6 – Photo Organizer, organizing program .................... 15
FIGURE 7 – NASA World Wind ........................................ 17
FIGURE 8 – Zoomifyer EZ™, embedded Adobe® Flash viewer .... 18
FIGURE 9 – The composition of an image .............................. 21
FIGURE 10 – Client-Server design of the Ymage System ............ 22
FIGURE 11 – The role of the Data Access Component ............... 25
FIGURE 12 – Data Storage Schema ..................................... 29
FIGURE 13 – Java Applet Visualization Component ................. 37
I. INTRODUCTION

The use of digital images has become critical in many areas. In medicine, for example, there is heavy reliance on digital imagery for diagnosing physical health problems. To be of any use, these images must be very detailed, but this presents problems. Detailed images require a great deal of storage space and a management system that can handle large files. Given the satisfaction of these requirements, there is also the problem of viewing the image. Traditional visualization and manipulation programs seem to target relatively small images, so using them on larger files results in a significant reduction of system performance. It is also rare for these programs to provide a means of accessing images that reside in the storage devices of systems on which they are not immediately running. The development of a system that effectively handles the storage, access, and viewing problems described would be of great value to the many disciplines that rely on digital imagery. The aim of this project was to design such a system in order to facilitate the use of large images.

A. Digital Images

Digital images are numerical representations of visual information [1]. They can be formed by capturing and digitizing naturally existing visual data, or by generating digital data using computer programs. Digitizing an image involves mapping
visual data points to discrete numerical values on cells of a digital grid. These grid cells are called picture elements, or pixels and the collection of adjacent pixels, each representing a different point, composes an entire digital image. How closely the image resembles the original visual data depends on pixel density and pixel depth. The former is a measure of how many pixels represent a fixed visual region; the latter describes the possible variability between pixels. As pixel depth and pixel density increase, so does the quality of the image, but the higher the image quality, the greater the amount of space required to store the image. Herein lays the image quality versus image size trade-off. The dilemma has produced many different file formats, each attempting to provide the right balance for particular use of images [2]. Since systems that deal with digital images must inevitably deal with files, it is important to look into the factors involved in the storage and retrieval of image files.

A.1 Image File Storage

Digital images that are stored in files contain several key parts. The most important is the graphical data set which represents the true visual data; it can be in raster, geometric, or latent forms. Raster and geometric data have direct visual mappings and so they can be immediately rendered on a display device; latent data requires transformation into one of the other forms before rendering. In addition to the graphical data, image
files usually contain self-describing information in the form of signatures, or magic numbers, and version numbers. Metadata and non-graphical data may also be present in the file, but these are dependent on the specific file format and the program used to generate it.

The data contained in an image file must be ordered in some fashion. The arrangement of data can be chosen to maximize retrieval efficiency. To achieve this, metadata should be at the beginning of a the file, followed by the file identification, then graphical data which can be preceded or succeeded by the non-graphical data [2]. The placement of metadata first provides the file reader with any important information upfront that may influence the way in which the file is used and prevent unnecessary traversal through the remaining data. Since an understanding of the file’s structuring of graphical and non-graphical data is vital to reading it effectively, the identification should follow the metadata. It would also be reasonable to keep the graphical and non-graphical data in independent, contiguous blocks. This would eliminate the need to skip about the file in search of these data. Though this structuring would provide the most benefit in terms of ease and flexibility of reading, not all image files are organized in this way [2].

Regardless of data order, the contents of image files need to be present in some form. Various encoding schemes are used for this purpose. Text encoding represents a data set in human-
readable form that can be manipulated directly with simple text-editors. This encoding scheme usually yields very large files and requires a lot of processing before the display of the image is possible so it is not often used when storage space requires consideration. Character encoding presents data in a form that replaces equivalent text-encoding blocks with shorter data codes. This saves space, but reading this data requires software that understands these codes. Displaying images encoded in this way also requires processing. Binary encoding saves space and can be rendered directly, but the data is specific to a particular hardware platform and manipulating data of this type also requires special software. With all three encoding schemes, there are trade-offs usually between storage space and retrieval time. The application to which an image will be put dictates the selection.

Image data sets are often transformed to reduce the amount of space required for storage; this transformation is called compression. Several techniques are used to compress images [2]. Since the visual data that images represent are not random, analysis of an image can reveal features or patterns. One compression strategy involves exploiting these patterns by replacing them with succinct representations. This is particularly effective when features repeat throughout an image. Sometimes the data set that represents the image is too descriptive. In these cases, the precision of the image can be reduced by eliminating superfluous detail without losing any
significant information; this is a reduction in image resolution. Yet another compression method reduces the number of bits needed for representing the values in the data set and is the same as minimizing pixel depth. Many applications require very high quality images which, as mentioned above, require large raw storage space. To make the storage of these images practical, compression is an imperative.

A.2 Image File Retrieval

The retrieval of image data from a file is, in theory, relatively easy. For data that is logically structured, the process involves opening a stream to the file, obtaining the metadata and format identification information, and then reading the desired data.

For unstructured files that cannot be randomly accessed, the reading process may require traversal of a significant portion of the file before reaching the required data. Once this data is reached, transformations may also be needed before the image can be rendered. These transformations may include decompression, decoding, and rasterization. After extracting and processing the image data, it can be sent to a display device, such as a monitor for visual presentation.

B. Applications of Digital Images

The significance of the digital image is clear from its widespread and varied uses. Images can be used to represent many
different types of data and are heavily relied upon to help identify and solve different problems.

Digital images are commonly used like their analog counterparts for simple display purposes. They can be acquired through photography, generated with the aide of software, or scanned from film. Their visual appeal makes them useful for advertisements, art galleries, educational and instructive media, online consumer catalogs, and other non-analytical purposes.

A relatively recent use of digital images has been in the area of document storage. The representation of textual documents in digital image form is a useful way to preserve the contents of deteriorating analog documents. Not long ago, Google began an initiative to scan library books and these scans to a collection of searchable digital content [3]. Optical character recognition (OCR) technology provides the option of converting these images to texts for textual analysis purposes. The combination of visual detail without loss of textual information makes images useful for representing documents.

The true power of images lies in the analytical uses to which they can be put. The medical field makes extensive use of digital imagery in this way. Images produced by radiologists capture physical properties of the human anatomy through X ray, ultrasound, magnetic resonance, and positron emission [4]. These images can be analyzed visually or computationally to investigate anatomical features and physiological functions. The information
gathered from the analysis of these images is relied upon for diagnosing health problems and possible treatments.

Other scientists also use digital image. Geologists, for example, employ digital imaging to study the Earth. Data that they collect from seismic surveys (a sound reflection technique) can be compiled to describe geological formations digitally [5]. The physical attributes of an area can be determined and used for various studies and commercial purposes, such as land changes over time that may determine safe building areas or profitable oil-drilling sites.

Remote sensing, or satellite imagery, is an area in which one of the earliest applications of digital imaging was made. Satellite images capture surface and atmospheric information from space with reflective wave methods. A great number of images are captured and the data available in these images can be coupled with other information, such as geological data, to reveal geospatial details.

Digital imagery is an integral part of many existing disciplines and advances will naturally prove them relevant to emerging fields. The significance of digital images provides sufficient motivation to investigate how they can be better managed and more effectively used.
II. EXISTING SOFTWARE

The uses described in the previous section show that digital images are very important. Their varied uses attest to their flexibility as a medium of data representation. Though the problems that they help to identify and solve may require different analytical approaches, applications that deal with digital images share the common problem of file retention and access. Of the many programs written to deal with images, few adequately address the fundamental issues of storage and retrieval. The following sections examine some programs that do attempt to handle these issues.

A. Standalone/Desktop Model

The standalone model is common for applications that act on local data. Programs in this category are either fed the location of data or conduct searches for image data on local media. These programs usually do not address storage; rather, the focus is on organizing and presenting data in an organized form which differs from that of the window manager. Picasa™, ImageQuery, and Photo Organizer Deluxe are three examples.

A.1 Picasa™

Picasa™ is a Google application for organizing images [6]. In keeping with Google’s philosophy, Picasa™ searches local media for existing image files and presents the results in an organized
fashion (see Figure 1). When images are found, thumbnails are created on-the-fly and cached in DB2 databases. Information about the files is saved to eliminate unnecessary future searches. Images that are imported into Picasa™ from external media such as CD’s and DVD’s are copied to the local hard drives.

Along with file organizing, Picasa™ also provides visualization and image processing features. Clicking on a thumbnail triggers the display of a larger version of the image that can be cropped for export, more closely inspected by zooming, or otherwise manipulated. Documentation of its internals is not available, but it appears, from Figure 2, that Picasa™
loads only the part of an image is needed. This is likely to be a factor contributing to its high response times.

Picasa™ also has the ability to upload images to storage systems that Google maintains. Options are available to reduce upload time and storage size by compression and resizing. Without these modifications, the original images could be too large to store on the available space, as in the case demonstrated in Figure 3. It is unknown, here, whether Picasa™’s upload tool could not handle the size of that particular image or whether the problem was insufficient storage space. In either case, this exemplifies the size issue that must be considered when dealing with high resolution images.
Picasa™ is a useful tool for locating and viewing images that reside locally, but it is not ideal for several reasons. First, the organizational features are only on the surface. The implied structure of the data is not maintained outside of Picasa™. This introduces a dependency of the data on the program and is problematic. Should the program malfunction or become obsolete, the apparent organization of the data is not necessarily guaranteed. Second, Picasa™ only truly maintains images that are locally stored. While it permits uploads to its storage facilities, and thereby provides greater access to those images, it does not appear that there is any way of viewing the remote version of uploaded images through the client Picasa™.
software; these must be viewed through other services that Google provides. Upload to non-Google systems is also only possible through email -- a medium not well suited for transferring large images. Furthermore, storage space on Google servers requires yearly payment for capacity greater than 1024 megabytes. And, third, although performance is reasonable when viewing large (about 100-megabyte) images, there is still significant lag in responsiveness every time the image is viewed anew. At times, the lag is long enough to suggest that the application has frozen. These issues make Picasa™ inadequate for handling images of substantial size.

A.2 ImageQuery

ImageQuery was written by Armin Hanisch with an emphasis on retrieval [7]. ImageQuery takes SQL-like strings and uses them to search the local file system. These queries can be written to find images by metadata available in the file. ImageQuery neither handles image storage nor furthers the accessibility of image files, and only minimally addresses visualization. Searches return a set of thumbnails whose images can be viewed either with the internal viewer or an external program designated for the purpose. Figure 4 shows the results of a search and the specification of a query through the Query Wizard.
The program takes an interesting approach to file retrieval. Metadata in image files is underutilized and ImageQuery appears to be one of the few applications that try to fully capitalize on it. The fact that so little of the data is ever used, however, provides a counter-incentive to storing this data. A lack of the metadata’s availability in files then undermines the benefit that ImageQuery could provide. Consistent inclusion of metadata for files in storage would correct the problem. By adding a management component to enforce this inclusion, ImageQuery could be significantly improved.
A.3 Photo Organizer Deluxe

PrimaSoft’s Photo Organizer Deluxe is a commercial database-centered, image storage and organizing system [8]. It is composed of two interdependent programs. The first is a database designer that allows the development of a data storage scheme consisting of various field types (see Figure 5). The second program is an organizer and viewer that facilitates data entry and visualization.

![Photo Organizer, database designer](image)

FIGURE 5 – Photo Organizer, database designer
(a) Available options include database and server maintenance.
(b) GUI-based database designing.

The designer is a flexible program that simplifies the process of describing how the data is to be organized. One notable feature is the option of storing image files in the database directly or inserting a reference entry that points to the file. Retaining image files directly in the database ensures
that the data is present and organized as expected. The accompanying documentation warns, however, that the performance of the system may suffer if this option is used extensively [8]. This, and an inspection of system’s files, indicates that the database is not meant to handle files that are very large. It is also unclear whether the data in the database is immediately accessible outside of the designer and organizer programs.

Although Photo Organizer makes an effort to address the storage and retrieval issues, it does not do so satisfactorily. The developers of the system noted that there may be problems with large image files, but only mitigate the problem by allowing storage outside of the system. This undermines the management
control that the database should have over the data. File and web database export features are also provided to increase accessibility to data. However, the first is impractical for large images and the second is only available at a fee. The system is not ideal, but the idea of organizing images and accompanying data in a database may be the best approach to handling image storage and retrieval.

B. Client-Server Model

The client-server model is used for applications that require a high degree of access to data by users in different places. It answers the access problem by centralizing data and providing a mechanism to respond to external requests. This section describes two programs that approach image storage and retrieval by using the client-server model.

B.1 NASA World Wind

World Wind is an open-source application developed by NASA for viewing the entire Earth [9]. It downloads satellite data stored on Microsoft’s TerraServer database [10] and displays images for the areas being viewed. The client is designed for computers running Microsoft Windows. In Figure 6, World Wind is showing satellite images taken over Louisville, Kentucky and southern Indiana. Since the area is so large, the initial images are not detailed, but inward zooming reveals replaces these with images that represent the region in greater detail.
World Wind was written to handle geographical data, but its idea of loading parts of images on demand could be applied to other large images. The benefits are apparent. If only one image of high detail were to represent the entire Earth, an unreasonable amount of time and client storage space would be required to view it. Dividing the image into separate parts allows clients to request only the parts of the image that they need; this saves time and space.

B.2 Zoomify

Zoomify is a commercial system for viewing images [11] and is perhaps the system that best addresses the storage and retrieval needs of large images. Given an image, Zoomifyer EZ™ generates
several thumbnails, divides each thumbnail into tiles, and compresses each tile. These files can then be viewed through an Adobe® Flash movie embedded in a web page. The details of this visualization element are unknown, but the behavior of the viewer suggests that the generated thumbnails are viewed as tile-groups and that zooming changes the tile-group in focus. Thumbnails that represent higher levels of detail are larger and consist of more tiles. During visualization, only those tiles in view are downloaded; this maximizes performance by reducing the amount of data that must be transferred for any given tile-group. Figure 8 shows a “Zoomify-ed” image being viewed from the flash movie in a web-browser.

FIGURE 8 – Zoomifyer EZ™, embedded Adobe® Flash viewer
The approach is successful and relatively well implemented, but some design issues are still apparent. Image and metadata are stored on a server as flat files and the viewer must reside on this server. The corruption of these files could prevent proper visualization. And, since these are just flat files, data maintenance (back-up, error correction, etc.) cannot easily handled. A more robust design with the ability to ensure the quality of the data over time would be an improvement.
III. THE YMAGE SYSTEM

It appears that there does not yet exist a system that fully answers the demands of large digital image storage and retrieval, but from a study of current software such as system is possible. The programs examined in the previous sections present effective solutions to different parts of the problem. It is from these approaches and an understanding of their shortcomings that the Ymage System was designed.

At the heart of the design is an abstraction called the “ymage.” An ymage is the collection of data that describes the original image and any additional information that may relate to it.
As illustrated in Figure 9, ymages contain layers, metadata, and “other data.” Metadata describe layers and their properties while “other data” includes any supplementary information about an ymage. Layers are abstractions of the graphical data that represent scaled versions of the original image. Every ymage has at least one layer. The first (layer 0) represents the un-scaled original image; the second represents an image that is some factor smaller than the first; the third is that same factor smaller than the second, and so on. Every layer is divided into subsections called image blocks. For a given layer, all blocks are of equal dimensions but not necessarily the
same dimensions as the blocks of other layers. Representing an image in this way ensures flexible access to information that describes it.

A. System Architecture

The overall system consists of three components, one to address each of the three perceived problems (i.e. storage, retrieval, and visualization). The need for structured storage is satisfied by the Data Storage Component (DSC). Transfer and manipulation of image data is handled by the Data Access Component (DAC). The Visualization Component (VC) provides the logistics for accurate image and data display. Each of the components is a relatively independent subsystem; this modularity eases system maintenance and makes the system flexible.

FIGURE 10 – Client-Server design of the Ymage System

It is possible for modular systems to become inefficient if a clear protocol is not established. To reduce the likelihood of possible inefficiencies, communication paths between the
different components are fixed and strictly enforced. As shown in Figure 10, client software and the Data Storage Component only communicate with the Data Access Component which is at the center of communication. The DAC validates data entering storage and polices data requests made by the client. Enforcement of this communication scheme maximizes access to data without jeopardizing its integrity or laying undue burden on the DSP.

A.1 Data Storage Component

The purpose of the data storage system is to retain information about an image. This includes the graphical data that would be stored in the files. Since this component will not restrict the types of image files that it retains, there is no guarantee that the graphical data will be accompanied by the needed metadata, so this information is stored in separate records. Other desirable data as it relates to a stored image can be retained in this system as well (e.g. annotation and image ownership information). In addition to keeping all of the image-related data in a structured manner, this subsystem provides flexible, direct access to its data.

The requirements of this component are consistent with the function of relational database management systems (DBMS). Since a DBMS stores data in customizable schemas, the structure of the data can be defined to the needs of the Data Storage Component. Most DBMS's also provide a powerful data retrieval mechanism (e.g. SQL querying) that can answer requests and return data in
forms that differ from the schema. Data backup and maintenance are also incentives to use them, and additional features of a specific DBMS may introduce other benefits.

A.2 Data Access Component

The Data Access subsystem is the middleware component responsible for connecting the Data Storage Component to the client applications; all data storage and retrieval must be are through this component. The process of data storage begins first with the transfer of data to the DAC. After this component completes an analysis of the data, all of the necessary processing is undertaken (i.e. subdivision and compression), then the data is forwarded to storage (see Figure 11). This serves as a means of validating data integrity and ensures that only data which has been analyzed and properly processed is retained. This subsystem also handles requests from clients for data that is already in storage. By mediating access to the storage component, this subsystem can reduce the work load of storage system by capturing request statistics and making optimization decisions based on that information. This has the potential to speed up requests and overall system performance.
Since the images targeted by the Ymage System are large both dimensionally and with regard to storage requirements, compressing and sub-dividing is necessary in order to make them usable. Integrated into the Data Access Component are the means of handling these processing needs.

In order to make requests of the storage system on behalf of the clients, this component must understand client requests. A simple mechanism consisting of a small vocabulary of commands was devised. When a request is handled, image files are returned without modification and non-graphical data is returned without unnecessary formatting.

A.3 Visualization Component

The Visualization Component is responsible for displaying the ymage data. To accomplish this, the system makes requests for stored data through the Data Access Component and combines the
results obtained. Since the interface between the data and this component requires only knowledge of the simple vocabulary of Data Access commands, developing different types of this component is possible.

B. Implementation

This first implementation focuses on proving the feasibility of the Ymage System and developing code that can be easily understood for further development. With the future in mind, consideration was given to restrictions that may be imposed by the use of proprietary software, so external code and binaries were chosen from open-source projects. Because this version was implemented to ensure functionality, it is not a full-fledged high-performance system; section C describes some possible improvements. Nonetheless, this implementation demonstrates the idea behind the design and shows the feasibility of developing the Ymage System.

B.1 Development Environment and Tools

Cygwin is a Linux emulation environment for Microsoft Windows [12]. It provides access to much of the Linux API and allows the development and execution of Linux targeted software on machines running the Windows Operating System. It was chosen as the development environment for this implementation for two reasons. The first is that it increases the likelihood that code developed in this Windows platform would be portable to Linux. Cygwin was adopted, secondly, in order to leverage the wealth of available
open-source development tools and libraries enjoyed by the Linux community, again, while guaranteeing that the code runs on the Windows platform.

Since various components of the system were written in different programming languages, an integrated development tool that could support this with minimal overhead was needed. The open-source Eclipse SDK 3.2 was selected. In addition to providing pluggable modules for many languages, Eclipse simplifies development with visual aids and code-refactory tools that are invaluable for efficient development.

Several independent libraries were used to perform tasks that varied from loading image files and handling database connectivity to managing memory allocation and processing HTTP requests. The Developer’s Image Library (DevIL, formerly OpenIL) [13] was used for image file loading, manipulation, and saving. PostgreSQL’s libpq library [14] was used to handle the database connectivity. The Apache Portable Runtime Libraries (libapr and libaprutil) [15] provided platform independent implementations of memory management functions. Libapreq [15], another Apache Project library, provided a reliable means of managing HTTP request data.

The availability of Cygwin packages containing some of these libraries made them very easy to use, but some needed to be built from source. All of the libraries listed above, were developed with portability in mind, so they supported the use of “make” and the GNU Autotools: autoconf, automake, and libtool
The accessibility of these packages through Cygwin’s setup program made building and debugging the libraries relatively simple.

The environment and tools chosen were a great aide in the implementation process. Many of the programs that were needed, and most of the libraries, are available as Cygwin packages which are easy to install and have been thoroughly tested.

B.2 Data Storage using PostgreSQL

As previously mentioned, the Data Storage Component must provide a structured data retention mechanism that keeps data retrieval simple. Since relational database management systems are designed for this purpose, one was chosen to act as the storage component.

PostgreSQL is an open-source object-relational DBMS [14]. It is known for its platform independence, reliability, scalability, and compliance with existing standards. It was selected for these reasons and because of its ease of integration into the Cygwin environment.

A simple schema was designed to describe the structure and types of data that would be held by the storage system. The schema consists of four tables and some stored procedures. The conceptual model behind the schema’s design focuses on the idea of an “ymage.” As mentioned before, this is an abstraction which consists of data that describes an image.
Ymages in the data storage schema consist of several parts. The first is a description found in the ymage_desc table. This table has one entry per ymage and contains the name of the ymage, its owner, the number of layers it contains, and the width and height of the original image. The name is a unique identifier and primary key, so the ymage_name field of every entry must be different.

Every ymage has one or more layers. A layer is an entity that corresponds to a scaled version of the original image. Layer-N is a copy of the original image with a size that is \((\frac{1}{2})^N\) times that of the original. Layers are not explicitly described in the schema, but are nonetheless referenced for identification in the block_desc and ymage_block tables.
Blocks are sub-images that, when arranged in the proper order, fully represent the visual aspect of the original image. Each individual block is part of a layer and is uniquely identifiable by combining the ymage’s name, layer, row, and column. The properties of any particular block are a function of the ymage and layer to which the block belongs. The width and height of blocks belonging to a layer are the same, but may be different from those of other layers. The bounds of a block’s position (relative to other blocks) are also dependent on the layer to which the block belongs. The block_desc table holds information about each layer’s blocks: the dimensions of the graphical block data and size of the layer grid. The ymage_block table has the graphical data for each block in the “imagedata” field, the other fields identify the block.

The remaining table, ymage_annot, holds annotation data. The location of the annotation (pos_x, pos_y) is given with respect to the dimensions in the ymage_desc table. The “user” field identifies the category or person who is associated with the annotation. The “annotation” field contains the information of interest. The remaining fields are referential and provide a means of linking the data to its respective ymage while ensuring the uniqueness of each record.

Functions are included to ensure that changes in the table schema have a minimal impact on the Data Access Component. These functions are primarily for inserting into, and removing records from, the tables. The one exception is a utility function for
determining the next identification number for an insertion into the ymage_annot table.

B.3 Apache HTTP for Data Access

As with data storage, the Data Access Component requires a reliable, accessible, yet flexible implementation. Efficient file distribution capability is very important since this component will deal mostly with files. As possible implementation strategies were evaluated, it became apparent that the desired traits were already inherent in a class of software: web servers.

If a web server was to be used, however, it had to be customizable for the somewhat unorthodox purpose that it would serve. The Apache HTTP server appeared to be the suitable candidate. Its modularity and configurability meant that components which would be unnecessary or obstructive for the purposes of this component could be removed or replaced with those better suited to the needs of Data Access. Its open-source and platform independent nature suggested compliance with the development tools already selected. So the Apache web server (version 2.2) was chosen as the backbone on which the Data Access Component would be built.

None of the initially envisioned modifications to the server were required for this implementation. Instead, a module was developed to run as part of the default configuration. The architecture of the Apache server is rather complicated, but in short it consists of a minimalist core which loads functionality
onto itself dynamically at runtime [15]. This functionality takes the form of modules and can perform any tasks related to request processing. Some of those loaded by default handle a wide range of web server needs, from enforcing security to determining how to respond to requests for certain types of files. To convert Apache from a web server to the Data Access Component of the Ymage System, a custom module was written named “mod_ymage.”

The module was developed with a command/parameter structure in mind. File requests sent to the server are intercepted and extensions are checked against the string “ymg”. If the extension of the requested file matches “ymg”, mod_ymage takes control of the request and makes an attempt to service it, otherwise it signals to Apache that it does not wish to handle it and passes the request along to another module. In order to service the request, mod_ymage first has to determine what the request is asking; it does this by extracting, parsing, and analyzing the query string. It looks for a value with the key named “cmd”. If cmd is found, its value signifies the course of action to take; if not found, mod_ymage responds to the request with a message explaining the problem. There are four “cmd” values (or simply commands) to which mod_ymage responds; these are UPLOAD, DOWNLOAD, GET, and ANNOTATE. They correspond to functions that can be applied to ymage data.

UPLOAD is interpreted as a desire to send an image file to the server for processing and submission to the Data Storage system. When this command is given, mod_ymage expects to find an
accompanying image file that it can upload to a temporary location on the system’s local storage. If the file is not submitted as part of an HTTP POST request, mod_ymage rejects the command and returns a message indicating that the upload was not consistent with its expectations.

When a file is uploaded, it is processed with the goal of dividing the image into layers and blocks then ultimately constructing an ymage for storage. The process involves several steps.

<table>
<thead>
<tr>
<th>TABLE 1 – Image Upload Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Analysis</td>
</tr>
<tr>
<td>Image blocks must all be of the same dimension as dictated by the schema, so the image is evaluated and a simple algorithm is used to determine the best dimensions to use, given a target. The algorithm is described in Appendix A. This analysis is done on a per-layer basis since the dimensions of each layer are different.</td>
</tr>
<tr>
<td>Step 2: Block Extraction</td>
</tr>
<tr>
<td>With the block dimensions known, the original image is traversed and independent blocks are extracted and saved locally. This is done for each layer. Regardless of the original image file’s format, the extracted blocks are saved as JPEGs with parameters set for high compression. Metadata is collected during this step as well.</td>
</tr>
<tr>
<td>Step 3: Data Storage</td>
</tr>
<tr>
<td>With all of the required data retrieved, an ymage can be constructed. An entry is placed in the ymage_desc table as a reference to the new ymage; the ymage_name field is populated with the file request (i.e. the string preceding &quot;.ymg&quot; in the HTTP request). All image blocks and metadata are then sent to the database server.</td>
</tr>
</tbody>
</table>
A message indicating completion is returned upon success of all three steps. An appropriate failure message is returned otherwise.

Image blocks that reside in the Data Storage Component are retrieved with the DOWNLOAD command. Parameters are passed in order to identify the desired block; these parameters are “layer”, “row”, and “column”. Their values correspond to those in the ymage_block table and are submitted in the query string as key-value pairs. Mod_ymage queries the data in storage with these parameters and if the specified image block is found, a file representing the block is returned in response. A message indicating failure to find the data is returned otherwise.

Non-graphical data is obtained using the GET command. The “item” parameter must be specified with this command; it takes one of three values: YMAGELIST, METADATA, and ANNOTATION. Specifying YMAGELIST causes a comma-delimited list of ymage names from the ymage_desc table to be returned. Use of METADATA indicates a request for information from either the ymage_desc, block_desc, or ymage_annot tables. To select which of these three, the “schema” parameter must be assigned a value. When “schema” is has the value “YMAGE”, the “width”, “height”, and “num_layers” values from the ymage_desc table are returned in semicolon-delimited list of key-value pairs. When “schema” is assigned the value “BLOCK”, “width”, “height”, “num_rows”, and “num_columns” are returned from the block_desc table in the same format. Assigning “schema” the value of “ANNOTATION_USER” returns
a comma-delimited list of unique annot_user values. Assigning “schema” the value “ANNOTATION” retrieves the values from the “annotation” field of the ymage_annot table for a given user. This “schema” parameter must be accompanied by the “user” parameter to indicate which data set is desired. The response returned is a newline-delimited list of annotation records. Each record contains the location of the annotation as “x” and “y” key-value pairs separated by “||”. The content of the “annotation” field of the ymage_annot table is the last item in the returned string.

The last command, ANNOTATE, is used to request the placement of annotation data into the ymage_annot table. This command requires the inclusion of additional parameters, “x”, “y”, and “user”, which correspond to the pos_x, pos_y, and annot_user fields in the ymage_annot table, respectively. The command must be sent as an HTTP POST with the annotation data in the body of the request under the name “annotation”. Correct specification of parameters results in the placement of the data onto the database and a message indicating success. Failure to include any of the required parameters for this and all other commands causes mod_ymage to return a message indicating the missing parameter.

The Data Access Component’s simplicity and use of the HTTP protocol makes it possible for different clients to be developed that use it. A web browser, for example, can be used to request data from the system and in fact, much of the system’s debugging
was done in this way before a viewing program was implemented. Future clients can be developed with new technologies and without the need to modify the Data Access Component; these clients only require the ability to make HTTP requests and interpret the results.

B.4 Java Applet Viewer

Many different approaches can be taken with respect to the Visualization Component. Initially, a Mozilla Firefox plug-in was developed for the purpose. It fulfilled the basic needs, but when experimentation with more advanced functionality was desired, it became more practical to use Java. The applet that was implemented is capable of using all of the features of the Data Access Component and behaves like a black box, hiding the image request details behind a graphical user interface.
The applet sends HTTP requests to the Apache Data Access Component and updates widgets with the data that it receives. To view an image, the URL of the server running mod_ymage is entered and a list of available ymages is requested. The returned list is converted into a set of menu options from which an ymage name can be selected. Clicking on one of these menu items initiates the download and display of graphical data. To maximize the efficiency of viewing the ymage, display begins at the highest layer, i.e. the one which consists of a single image block. As the user zooms into the image, the applet determines whether to move to the next layer and initiate download requests for new image blocks or to resize the blocks that it already has.
When a layer is reached that is composed of more than one image block, the applet identifies which blocks are within the view port and downloads for display. Dragging the view port changes the visible region and image blocks which have not been downloaded are requested for subsequent display.

Upload features are also available in this applet. An image file can be located on the user’s local file system and previewed before upload. The preview image is a subsampled version of the original obtained using Java’s Image I/O API. While previewing, a rectangular region of the image can be selected for upload, should the whole image not be wanted.

The applet is independent of any particular Data Access server, so it can be used to view images from any server that runs an Image System Data Access Component. This flexibility can be attributed to the modularity of the overall system and maximizes data accessibility.

C. Evaluation and Possible Improvements
The implementation presented here is rather rudimentary. It is a model of how the system should generally work. As such, it does not fully exploit the strengths of the design. Improvements can be made to each subsystem in order for real performance gains to be realized.

Improvements can be made with respect to the Data Storage Component. Since PostgreSQL was available as a Cygwin binary package and other well known contenders such as MySQL were not,
the former was selected partly out of convenience. Past studies and current discussion have credited MySQL with being a faster relational database system and so an improvement in performance may be achieved in electing to use it. The schema used for this implementation would require only minor changes to ensure compliance with MySQL. As the performance degradation of this component becomes less of an issue, the schema can be expanded to more properly address security and data ownership concerns, system protocol enforcement, and inclusion of a greater assortment of data.

The Data Access Component is critical to the system’s overall performance since all data and requests are channeled through it. Improvements can be made to this component by exploiting the features and tweaking the operation of the Apache web server. Eliminating all modules that service demands inconsistent with the needs of the system and the hardware on which it is running would be likely to reduce overhead. Some of the modules loaded by default handle directory authorization, executing CGI scripts, and web authoring. These are not necessary for the Data Access to function. Along with removing such modules, a reduction in the number of database transactions by caching highly demanded data would also boost performance. The current implementation of this subsystem really consists of a web server with the ability to carry out functions consistent with the Ymage System. The ultimate goal is to develop this component
enough that it becomes a subsystem wholly dedicated to the servicing of the overall systems demands.

Improvements of the Visualization Component could be made by adopting the modularity that is apparent on the overall system level. Graphical display should be the sole purpose of the Visualization Component; in this way, the focus can be maintained on image quality. Optimizations could come in the form of using technologies which are better suited to fast, high-quality distributed visualization. Image uploads should be implemented in a client branch of the Data Access Component and eliminate the need to integrate visual demands with file transfer logistics.

For the practical purposes to which a system built from this architecture will be put, these changes would substantially improve usability. As a model, the present implementation demonstrates the possibilities of the design and is a respectable starting place for development of more rigorous software.
IV. RESULTS

Comparisons of the Ymage System (as implemented in this project) show that it is a more practical option most of the time than other commonly used digital image viewing software.

TABLE 2 - Images Tested

<table>
<thead>
<tr>
<th>Ref #</th>
<th>Image Name</th>
<th>Size</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P0-SC-QGFR146-Grina.tif</td>
<td>27.4 MB</td>
<td>4000 px</td>
<td>2400 px</td>
</tr>
<tr>
<td>2</td>
<td>P0-SC-QGFR142-Grml.tif</td>
<td>30.1 MB</td>
<td>3900 px</td>
<td>2700 px</td>
</tr>
<tr>
<td>3</td>
<td>BDM-P14-1-0009-HT559.tif</td>
<td>76.9 MB</td>
<td>4800 px</td>
<td>5600 px</td>
</tr>
<tr>
<td>4</td>
<td>BDNF-P12-1-0009-HT559.tif</td>
<td>98.9 MB</td>
<td>7200 px</td>
<td>4800 px</td>
</tr>
<tr>
<td>5</td>
<td>wt-E13-5-Brain-QGF033-HT551.tif</td>
<td>251 MB</td>
<td>15400 px</td>
<td>17100 px</td>
</tr>
</tbody>
</table>

*See Appendix E for images

Images from brain and retinal scans were used to test desktop applications, Zoomifyer EZ™, and the Ymage System. Table 1 lists the images and their properties. One important metric in determining how useful a program is for displaying an image is the time that it requires to do so. Load times are the time from when the program begins displaying a region to when all of the data for that region have been loaded and are visible. The load times of the desktop applications are compared in Table 3. Though Picasa™ out-performed the others for images ranging from 27 to 99 megabytes, it failed to load the 251 megabyte image at its highest resolution within the practical time limit of 10 minutes. The other programs loaded the images within a reasonable time,
but this was at the price of a significant performance reduction to all programs running on the computer.

<table>
<thead>
<tr>
<th>Image</th>
<th>Adobe Photoshop® CS2</th>
<th>GIMP2</th>
<th>Google Picasa2™</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 sec</td>
<td>3 sec</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>2</td>
<td>2 sec</td>
<td>3 sec</td>
<td>1 sec</td>
</tr>
<tr>
<td>3</td>
<td>6 sec</td>
<td>18 sec</td>
<td>2 sec</td>
</tr>
<tr>
<td>4</td>
<td>2 sec</td>
<td>33 sec</td>
<td>3 sec</td>
</tr>
<tr>
<td>5</td>
<td>21 sec</td>
<td>4 min 54 sec</td>
<td>&gt;10 min</td>
</tr>
</tbody>
</table>

Load times of the Ymage System were comparable, if not better than those of the desktop programs. However, improvements in load time were at the cost of segmentation and upload overhead (see Table 4). Although the combined segmentation and upload times for the Ymage System are significantly longer than desktop application load times, this cost must only be endured once and compensation is made in storage space savings.

<table>
<thead>
<tr>
<th>Image</th>
<th>Segmentation Time</th>
<th>Seg + Upload Time</th>
<th>Post-Seg Size</th>
<th>Load Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5 sec</td>
<td>25 sec</td>
<td>2.17 MB</td>
<td>2 sec</td>
</tr>
<tr>
<td>2</td>
<td>6 sec</td>
<td>26 sec</td>
<td>2.78 MB</td>
<td>1 sec</td>
</tr>
<tr>
<td>3</td>
<td>16 sec</td>
<td>49 sec</td>
<td>4.39 MB</td>
<td>2 sec</td>
</tr>
<tr>
<td>4</td>
<td>31 sec</td>
<td>1 min 32 sec</td>
<td>7.20 MB</td>
<td>1 sec</td>
</tr>
<tr>
<td>5</td>
<td>6 min 59 sec</td>
<td>13 min 54 sec</td>
<td>64.6 MB</td>
<td>1 sec</td>
</tr>
</tbody>
</table>

Zoomifyer EZ™ outperformed this implementation of the Ymage System with apparent load times of under one second for all images tested (see Table 5). Greater storage savings were also

Tests were conducted on a Dual AMD Athlon™ MP 2200+ (1.86 GHz processors) machine with 1GByte of PC2100 DDR RAM running Windows XP Professional with Service Pack 2.
realized with Zoomifyer EZ™ than with the Ymage System. Upload time to the web server for the former could not be accurately measured as there were several different, yet equally feasible, methods for completing the task; these times could be considered proportional to the size of the original image files.

<table>
<thead>
<tr>
<th>Image</th>
<th>Segmentation Time</th>
<th>Post-Seg Size</th>
<th>Load Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12 sec</td>
<td>1.86 MB</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>2</td>
<td>10 sec</td>
<td>2.37 MB</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>3</td>
<td>23 sec</td>
<td>3.63 MB</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>4</td>
<td>32 sec</td>
<td>6.06 MB</td>
<td>&lt;1 sec</td>
</tr>
<tr>
<td>5</td>
<td>4min 58 sec</td>
<td>51.2 MB</td>
<td>&lt;1 sec</td>
</tr>
</tbody>
</table>
V. CONCLUSION

The storage, retrieval, and visualization problems associated with the practical use of large, high resolution digital images can be resolved using a flexible, modular system design. Furthermore, there is sufficient open source-code to meet the requirements without the need of commercial software.

Conventional methods employed by the standalone applications proved to be inadequate for several reasons. First, the load times increased in proportion to image size. For images that are very large, the delay between the time an image was requested and the time it became visible was unreasonably long. And, second, the demand on system resources, in the form of storage space, system memory, and CPU use, rendered the viewing process and the computer unworkable. The time issue cannot be completely avoided; a great deal of data requires time to be processed. In the standalone applications this processing delay was observed every time the image was requested. With the Ymage System and Zoomifyer EZ™, there was a greater processing time realized, but it was only manifested once per image. During this time machine performance was affected to some degree; however, subsequent requests for the image were almost instantaneous regardless of image size. The reduction of required storage space also favored the Ymage System and Zoomifyer EZ™ over the client applications. It is apparent that the way in which these two
approach image storage and retrieval are an improvement to the approaches taken by the standalone programs.

The Ymage System performed comparably to Zoomifyer EZ™. This was surprising since the implementation of the former was not completed with much focus on performance optimization. Making use of the operational exploits described in the “Evaluation and Possible Improvements” section could potentially improve the Ymage System enough to ultimately equal Zoomifyer EZ™ in performance. The difference between the two systems would then be in data storage where the organization and maintainability of data in the Ymage System is an improvement over the other system.

The modularity of the Ymage System affords itself well to other applications. Since data retrieval in the system is independent of the Visualization Component, the system is extensible for non-visual functions. One interesting use would be for distributed image processing. Blocks could be obtained from different parts of an image by different machines and processed independently; the results could be combined later outside of the system or (with some modification to the Data Access Component) placed in Data Storage Component. The implications of organizing and providing access to image data as presented by the Ymage System has the potential to change the way in which images are viewed and manipulated. Novel approaches to image application design are sure to come from the options available through the architecture of this system.
REFERENCES


A.1 Image Subdivision Process

1. Load Image
2. Check if Image Dimensions Greater Than Target Dimensions
   - Yes: Compute Image Block Dimensions
   - No: Store Image Blocks and Metadata
3. If Image Dimensions Greater Than Target Dimensions, exit and go to Done.
4. Scale Image Down by 50%
5. Temporary Storage
6. Extract, Compress, and Save Image Blocks
7. Permanent Storage
8. Done
A.2 Algorithm for Computing Image Block Dimensions

1. Compare Image Dimension to Target Dimension. If Image Dimension is greater than Target Dimension, mark 'Done'.
2. If not, find the largest of the first 30 prime numbers that divide the Image Dimension evenly.
3. If a prime number is found, subtract 1 pixel from the Image Dimension and repeat the process.
4. If no prime number is found and the Image Dimension is not greater than the Target Dimension, divide the Image Dimension and repeat from step 1.
--- Database: ymage_db
--- DROP DATABASE ymage_db;
--- CREATE DATABASE ymage_db
---   ENCODING = 'SQL_ASCII'
---   TABLESPACE = pg_default;

--- Table: ymage_desc
--- DROP TABLE ymage_desc;
CREATE TABLE ymage_desc
(
    ymage_name character varying(128) NOT NULL,
    width integer NOT NULL,
    height integer NOT NULL,
    num_layers integer NOT NULL
--    CONSTRAINT pk_ymage_desc PRIMARY KEY (ymage_name)
)
WITHOUT OIDS;
ALTER TABLE ymage_desc
ADD CONSTRAINT pk_ymage_desc PRIMARY KEY (ymage_name);

--- Table: block_desc
--- DROP TABLE block_desc;
CREATE TABLE block_desc
(
    ymage_name character varying(128) NOT NULL,
    layer integer NOT NULL,
    width integer NOT NULL,
    height integer NOT NULL,
    num_rows integer NOT NULL,
    num_columns integer NOT NULL,
--    CONSTRAINT pk_block_desc PRIMARY KEY (ymage_name, layer),
    CONSTRAINT fk_block_desc FOREIGN KEY (ymage_name)
        REFERENCES ymage_desc (ymage_name) MATCH SIMPLE
        ON UPDATE CASCADE ON DELETE CASCADE
)
WITHOUT OIDS;
ALTER TABLE block_desc
ADD CONSTRAINT pk_block_desc PRIMARY KEY (ymage_name, layer);

--- Table: ymage_annot
--- DROP TABLE ymage_annot;
CREATE TABLE ymage_annot
(
    ymage_name character varying(128) NOT NULL,
    annot_user character varying(128) NOT NULL,
    annot_id bigint NOT NULL,
    pos_x integer NOT NULL,
    pos_y integer NOT NULL,
    annotation character varying NOT NULL
--    CONSTRAINT pk_ymage_annot PRIMARY KEY (ymage_name, annot_user, annot_id),
    CONSTRAINT fk_ymage_annot FOREIGN KEY (ymage_name)
        REFERENCES ymage_desc (ymage_name) MATCH SIMPLE
        ON UPDATE CASCADE ON DELETE CASCADE
)
WITHOUT OIDS;
ALTER TABLE ymage_annot
ADD CONSTRAINT pk_ymage_annot PRIMARY KEY (ymage_name, annot_user, annot_id);

--- Table: ymage_block
--- DROP TABLE ymage_block;
CREATE TABLE ymage_block
{  
ymage_name character varying(128) NOT NULL,
layer integer NOT NULL,
"row" integer NOT NULL,
"column" integer NOT NULL,
imagedata bytea NOT NULL,
-- CONSTRAINT pk_ymage_block PRIMARY KEY (ymage_name, layer, "row", "column"),
CONSTRAINT fk_ymage_block FOREIGN KEY (ymage_name, layer)
REFERENCES block_desc (ymage_name, layer) MATCH SIMPLE
ON UPDATE CASCADE ON DELETE CASCADE
DEFERRABLE
}  
WITHOUT OIDS;
ALTER TABLE ymage_block
ADD CONSTRAINT pk_ymage_block PRIMARY KEY(ymage_name, layer, "row", "column");

-- Function: get_next_annot_id(ymage character varying, user character varying)
-- DROP FUNCTION get_next_annot_id(ymage character varying, user character varying);
CREATE OR REPLACE FUNCTION get_next_annot_id(ymage character varying,
"user" character varying)
RETURNS bigint AS
'SELECT COALESCE(MAX(annot_id)+1,1) FROM ymage_annot WHERE ymage_name = $1 AND annot_user = $2'
LANGUAGE 'sql' VOLATILE;

-- Function: insert_ymage_desc(ymage_name character varying, width integer, height integer, num_layers integer)
-- DROP FUNCTION insert_ymage_desc(ymage_name character varying, width integer, height integer, num_layers integer);
CREATE OR REPLACE FUNCTION insert_ymage_desc(ymage_name character varying,
width integer, height integer, num_layers integer)
RETURNS void AS
'INSERT INTO ymage_desc(ymage_name,width,height,num_layers) VALUES($1,$2,$3,$4)'
LANGUAGE 'sql' VOLATILE;

-- Function: insert_block_desc(ymage_name character varying, layer integer, width integer, height integer, num_rows integer, num_columns integer)
-- DROP FUNCTION insert_block_desc(ymage_name character varying, layer integer, width integer, height integer, num_rows integer, num_columns integer);
CREATE OR REPLACE FUNCTION insert_block_desc(ymage_name character varying,
layer integer, width integer, height integer, num_rows integer, num_columns integer)
RETURNS void AS
'INSERT INTO block_desc(ymage_name,layer,width,height,num_rows,num_columns) VALUES($1,$2,$3,$4,$5,$6)'
LANGUAGE 'sql' VOLATILE;
-- Function: insert_ymage_annot(ymage_name character varying, annot_user character varying, pos_x integer, pos_y integer, annotation character varying)
-- DROP FUNCTION insert_ymage_annot(ymage_name character varying, annot_user character varying, pos_x integer, pos_y integer, annotation character varying);
CREATE OR REPLACE FUNCTION insert_ymage_annot(ymage_name character varying, annot_user character varying, pos_x integer, pos_y integer, annotation character varying)
  RETURNS void AS
  'INSERT INTO ymage_annot(ymage_name,annot_user,annot_id,pos_x,pos_y,annotation)
VALUES($1,$2,get_next_annot_id($1,$2),$3,$4,$5);'
  LANGUAGE 'sql' VOLATILE;
APPENDIX C

Source Code Listing for Apache Module

C.1 Module Header File (mod_ymage.h)

```c
#ifndef MOD_YMAGE_H_
#define MOD_YMAGE_H_

/* *** APACHE PORTABLE RUNTIME LIBRARY *** */
#include "apr.h"
#include "apr_buckets.h"
#include "apr_file_io.h"
#include "apr_general.h"
#include "apr_strings.h"
#include "apr_tables.h"
#include "apr_time.h"

/* *** APACHE HTTPD API *** */
#include "ap_config.h"
#include "httpd.h"
#include "http_config.h"
#include "http_log.h"
#include "http_protocol.h"
#include "http_request.h"

/* *** APACHE REQUEST LIBRARY *** */
#include "apreq_module.h"
#include "apreq_module_apache2.h"
#include "apreq_param.h"
#include "apreq_util.h"

/* *** POSTGRESQL *** */
#include "libpq-fe.h"

/* *** MISCELLANEOUS *** */
#include "math.h"
#include "bool.h"
#include "def_seg.h"

/* *** DATATYPES AND CONSTANTS *** */
typedef enum _YMG_VALIDATION
{
    YMG_VALID,
    YMG_INVALID
} YMG_VALIDATION;

typedef enum _YMG_FN_RESULT
{
    YMG_SUCCESS,
    YMG_MISSING_FIELD,
    YMG_NOT_AN_UPLOAD,
    YMG_ALREADY_EXISTS,
    YMG_DBCONN_ERROR,
    YMG_FILEOPEN_ERROR,
    YMG_MEMALLOC_ERROR,
    YMG_FILEREAD_ERROR,
    YMG_SUBPOOL_ERROR,
    YMG_DATAESCAPE_ERROR,
    YMG_FAIL_YMGDESC,
    YMG_BLOCK_NOT_FOUND,
    YMG_MISSING_LAYER,
```
YMG_MISSING_ROW,
YMG_MISSING_COLUMN,
YMG_MISSING_ITEM,
YMG_MISSING_SCHEMA,
YMG_MISSING_USER,
YMG_INVALID_SCHEMA_ARG,
YMG_INFO_NOT_FOUND,
YMG_NO_ANNOTATION,
YMG_MISSING_POSITION_ARG
} YMG_FN_RESULT;
/* *** FUNCTIONS *** */
/* Module Definition */
extern module ymage_module;
/* Module Hooks */
extern void ymage_register_hooks(apr_pool_t * /*pPool*/);
extern int ymage_type_checker(request_rec * /*pReq*/);
extern int ymage_handler(request_rec * /*pReq*/);
/* Utility Fns */
extern YMG_FN_RESULT yUpload(request_rec * /*pReq*/ , apreq_handle_t * /*apreq*/);
extern YMG_FN_RESULT yStore(request_rec * /*pReq*/ , const int[3] /*stats*/ , const
apr_array_header_t * /*arraylrc*/);
extern YMG_FN_RESULT yDownload(request_rec * /*pReq*/ , apreq_handle_t * /*apreq*/ ,
apr_bucket_brigade * /*imagedata*/);
extern YMG_FN_RESULT yGet(request_rec * /*pReq*/ , apreq_handle_t * /*apreq*/ ,
char ** /*result*/);
extern YMG_FN_RESULT yAnnotate(request_rec * /*pReq*/ , apreq_handle_t * /*apreq*/);
/* Text Response Fns */
extern void yPrintResponse(request_rec * /*pReq*/ , char * /*request_type*/ , char *
/*error_string*/ , char * /*string_color*/);
extern void yLogResponse(char * /*request_type*/ , char * /*error_string*/);
/* apxs build/install command */
/* --------------------------
* apxs2 -i -a -c
* -I/usr/include/apreq2 -I/usr/local/include/il
* mod_ymage.c filter.c handle.c def_seg.c
* -L/usr/local/lib
* -lapreq2 -llil -lilu -lllut -lpq
*/
#endif /*MOD_YMAGE_H*/
C.2 Module Source File (mod_yimage.c)

```c
#include 
#include "mod_yimage.h"

MODULE DEFINITION

module AP_MODULE_DECLARE_DATA
yimage_module =
{ 
STANDARD20_MODULE_STUFF,  /* my_dir_conf */
NULL,  /* my_dir_merge */
NULL,  /* my_server_conf */
NULL,  /* my_server_merge */
NULL,  /* my_cmds */
ymage_register_hooks, /* my_hooks */
};

GLOBAL VARIABLES

static YMG_VALIDATION BOOL_VALID_YMAGE_EXT = YMG_INVALID;
static char * STR_FILE_NAME = NULL;
static char * STR_FILE_EXT = NULL;
static char * STR_UPLOAD_PATH = NULL;

MODULE HOOKS

void yimage_register_hooks(apr_pool_t *pPool)
{
/* hook in the type checker */
ap_hook_type_checker(yimage_type_checker, NULL, NULL, APR_HOOK_MIDDLE);
/* hook in the main module handler */
ap_hook_handler(yimage_handler, NULL, NULL, APR_HOOK_MIDDLE);
}

/* Determine whether the request is to be handled by this module */
int yimage_type_checker(request_rec *pReq)
{
/* extract the filename and extension */
char* ext = strrchr(pReq->filename,'/'); ext++;
char* fn = ap_getword(pReq->pool,(const char **) &ext,'.');
STR_FILE_NAME = apr_pstrdup(pReq->pool,fn);
STR_FILE_EXT = apr_pstrdup(pReq->pool,ext);

/* assume that this module will not handle the request */
BOOL_VALID_YMAGE_EXT = YMG_INVALID;

/* if the extension is not correct do not handle this request */
if(!ext) return DECLINED;
```
else if(apr_strnatcasecmp(ext, "ymg") != 0)  return DECLINED;

/* the extension was correct, indicate that the request should be handled */
BOOL_VALID_YMAGE_EXT = YMG_VALID;

return OK;
}

/* Handle the request as part of the Ymage System */
* @param: pReq - pointer to the current request */
int ymage_handler(request_rec * pReq)
{
    /* if the file extension is not .ymg, do not handle */
    if(BOOL_VALID_YMAGE_EXT == YMG_INVALID) {
        return DECLINED; }

    /* use APREQ filter to process request query-string, post-data, etc */
apreq_handle_t *apreq = apreq_handle_apache2(pReq);

    /* determine how to handle the request */
apreq_param_t * param = NULL;

    /* obtain command string */
    if((param = apreq_args_get(apreq, "cmd")) != NULL)
    {
        switch(yUpload(pReq, apreq))
        {
        case YMG_MISSING_FIELD:
            yPrintResponse(pReq, "UPLOAD", "Missing upload field.", "#ff0000");
            return OK;
        case YMG_NOT_AN_UPLOAD:
            yPrintResponse(pReq, "UPLOAD", "Invalid upload request.", "#ff0000");
            return OK;
        case YMG_SUCCESS:
            break;
        default: break;
        }

        /* upload file from client to webserver */
        if(apr_strnatcasecmp(param->v.data, "UPLOAD") == 0)
        {
            switch(yUpload(pReq, apreq))
            {
            case YMG_MISSING_FIELD:
                yPrintResponse(pReq, "UPLOAD", "Missing upload field.", "#ff0000");
                return OK;
            case YMG_NOT_AN_UPLOAD:
                yPrintResponse(pReq, "UPLOAD", "Invalid upload request.", "#ff0000");
                return OK;
            case YMG_SUCCESS:
                break;
            default: break;
            }

            /* segment image and upload blocks to database server */
            char * dir = apr_pstrdup(pReq->pool, "/tmp/");
dir = apr_pstrcat(pReq->pool, dir, STR_FILE_NAME, NULL);
apr_dir_make(dir, APR_FPROT OS_DEFAULT, pReq->pool);

            int * stats = (int*)apr_palloc(pReq->pool, 3*sizeof(int));
apr_array_header_t * lrc = apr_array_make(pReq->
            pool, 10, sizeof(ILint));

            switch(ydsSegment(pReq, 256, STR_UPLOAD_PATH, dir, STR_FILE NAME, lrc, stats))
            {
            case ARRAYLRC_NULL:
                yPrintResponse(pReq, "SEGMENTATION", "Memory error.", "#ff0000");
                break;
            case INVALID_DIMENSION:
                yPrintResponse(pReq, "SEGMENTATION", "Invalid target dimension specified.", "#ff0000");
                break;
            case IMAGE_LOAD_FAIL:
                yPrintResponse(pReq, "SEGMENTATION", "Could not load the image.", "#ff0000");
                break;
            default: break;
            }
        }
    }
}
```c
ap_rprintf(pReq,"<br>%s", STR_UPLOAD_PATH);
break;
case DEFSEG_SUCCESS:
    switch(yStore(pReq,stats,lrc))
    {
    case YMG_ALREADY_EXISTS:
        yPrintResponse(pReq,"DBSTORE","Ymage already
exists.","#ff0000");
        break;
    case YMG_DBCONN_ERROR:
        yPrintResponse(pReq,"DBSTORE","Could not connect
to database server.","#ff0000");
        break;
    case YMG_FILEOPEN_ERROR:
        yPrintResponse(pReq,"DBSTORE","Could not open block
files.","#ff0000");
        break;
    case YMG_MEMALLOC_ERROR:
        yPrintResponse(pReq,"DBSTORE","Ran out of memory.",
"#ff0000");
        break;
    case YMG_FILEREAD_ERROR:
        yPrintResponse(pReq,"DBSTORE","Could not read from
block file.","#ff0000");
        break;
    case YMG_SUBPOOL_ERROR:
        yPrintResponse(pReq,"DBSTORE","Could not create a
subpool.","#ff0000");
        break;
    case YMG_DATAESCAPE_ERROR:
        yPrintResponse(pReq,"DBSTORE","Error escaping binary
data.","#ff0000");
        break;
    case YMG_FAIL_YMGDESC:
        yPrintResponse(pReq,"DBSTORE","Failed to add
ymg_desc record.","#ff0000");
        break;
    case YMG_SUCCESS:
        /*apr_dir_remove(dir,pReq->pool);*/
        yPrintResponse(pReq,"UPLOAD","Image uploaded to the
database sucessfully.","#008800");
        break;
    default: break;
    }
    break;
default: break;
}
}
/* download image block from database */
else if(apr_strnatcasecmp(param->v.data,"DOWNLOAD") == 0)
{
    apr_bucket_brigade * image = apr_brigade_create(pReq->pool, pReq->
connection->bucket_alloc);

    /* retrieve arguments for block download */
    switch(yDownload(pReq,apreq,image))
    {
    case YMG_MISSING_LAYER:
        yPrintResponse(pReq,"DOWNLOAD","Missing parameter: layer.",
"#ff0000");
        break;
    case YMG_MISSING_ROW:
        yPrintResponse(pReq,"DOWNLOAD","Missing parameter: row.",
"#ff0000");
        break;
    case YMG_MISSING_COLUMN:
        yPrintResponse(pReq,"DOWNLOAD","Missing parameter:
column.","#ff0000");
        break;
    case YMG_BLOCK_NOT_FOUND:
        yPrintResponse(pReq,"DOWNLOAD","Requested ymage block
not found.","#ff0000");
```

57
break;
case YMG_SUCCESS:
    /* put data on the output stream */
    pReq->content_type = "image/jpg";
    ap_pass_brigade(pReq->output_filters, image);
    break;
default: break;
}
}
else if(apr_strnatcasecmp(param->v.data,"GET") == 0)
{
    char * result = NULL;
    switch(yGet(pReq,apreq,&result))
    {
    case YMG_DBCONN_ERROR:
        yPrintResponse(pReq,"GET","Could not connect the the
database.","#ff0000");
        break;
    case YMG_MISSING_ITEM:
        yPrintResponse(pReq,"GET","Missing parameter: item.",
                        "#ff0000");
        break;
    case YMG_MISSING_LAYER:
        yPrintResponse(pReq,"GET","Missing parameter: layer.",
                        "#ff0000");
        break;
    case YMG_INFO_NOT_FOUND:
        yPrintResponse(pReq,"GET","Requested image info could not
be found.","#ff0000");
        break;
    case YMG_MISSING_SCHEMA:
        yPrintResponse(pReq,"GET","Missing parameter: schema.",
                        "#ff0000");
        break;
    case YMG_INVALID_SCHEMA_ARG:
        yPrintResponse(pReq,"GET","Invalid argument for parameter:
schema.","#ff0000");
        break;
    case YMG_MISSING_USER:
        yPrintResponse(pReq,"GET","Missing parameter: user.",
                        "#ff0000");
        break;
    case YMG_SUCCESS:
    /* put data on the output stream */
    pReq->content_type = "text/html";
    ap_rprintf(pReq,result);
    break;
    default: break;
}
}
else if(apr_strnatcasecmp(param->v.data,"ANNOTATE") == 0)
{
    switch(yAnnotate(pReq,apreq))
    {
    case YMG_NO_ANNOTATION:
        yPrintResponse(pReq,"ANNOTATE","No annotation provided.",
                        "#ff0000");
        break;
    case YMG_MISSING_USER:
        yPrintResponse(pReq,"ANNOTATE","Missing parameter: user.",
                        "#ff0000");
        break;
    case YMG_MISSING_POSITION_ARG:
        yPrintResponse(pReq,"ANNOTATE","Missing parameter: x and/or
y.","#ff0000");
        break;
    case YMG_DBCONN_ERROR:
        yPrintResponse(pReq,"ANNOTATE","Could not connect the the
database.","#ff0000");
        break;
case YMG_SUCCESS:
    yPrintResponse(pReq, "ANNOTATE","Annotation added successfully.", "\#008800");
    break;
default: break;
}

else
{
    yPrintResponse(pReq, ":", "NO COMMAND SPECIFIED", "\#ff0000");
}

/* Indicate that the request has been handled */
return OK;

/*=============================================================================*/
/* *** UTILITY FNS *** */
/* ------------------------------------------------ ---------------- */
/*=============================================================================*/

YMG_FN_RESULT yUpload(request_rec * pReq, aprreq_handle_t * apreq) {
    /* retrieve the posted file with name 'ymgfile' */
    apreq_param_t * param = apreq_body_get(apreq, "ymgfile");
    if (param == NULL) {
        return YMG_MISSING_FIELD;
    }
    else if(param->upload == NULL) {
        return YMG_NOT_AN_UPLOAD;
    }
    else {
        apr_bucket_brigade *bb; /* upload contents */
        bb = apr_brigade_create(pReq->pool, pReq->connection->bucket_alloc);
        apreq_brigade_copy(bb, param->upload);
        /* create a temporary file in the tmp directory */
        srand((int)time(NULL));
        apr_file_t * tmpfile;
        const char* tmpstrfn = apr_pstrdup(pReq->pool, STR_FILE_NAME);
        const char* tmpname = apr_pstrdup(pReq->pool, "/tmp/");
        char* tmnum = apr_itoa(pReq->pool, rand());
        tmname = apr_pstrcat(pReq->pool, tmname, tmstrfn, tmnum, NULL);
        STR_UPLOAD_PATH = apr_pstrdup(pReq->pool, tmpname);
        apr_file_open(&tmpfile, tmname, APR_OPEN_WRITE|APR_OPEN_CREATE| APR_OPEN_BINARY|APR_OPEN_TRUNCATE|APR_FPROT_UREAD| APR_FPROT_UWRITE|APR_FPROT_GREAD|APR_FPROT_GWRITE| APR_FPROT_WREAD|APR_FPROT_WWRITE,pReq->pool);
        apr_pool_cleanup_register(pReq->pool, tmpfile, (void*) apr_file_close, apr_pool_cleanup_null);
        apr_off_t wlen = 0;
        aprreq_brigade_fwrite(tmpfile, &wlen, bb);
        apr_file_close(tmpfile);
    }
    return YMG_SUCCESS;
}
/* Store the image blocks in the database 
* @param: pReq  - pointer to the request object 
* @param: stats - array containing width, height, number-of-layers respectively 
* @param: arraylrc - empty non-null array for storing layer, row, and column of each block 
*/

YMG_FN_RESULT
yStore(request_rec * pReq,const int stats[3],const apr_array_header_t * arraylrc) {
    PGconn * conn = NULL;
    PGresult *res = NULL;
    char conninfo[256];
    char query[4096];

    const char *hostip = "127.0.0.1";
    const char *database = "ymage_db";

    /* create subpool for this function, free at the end */
    apr_pool_t * subpool = NULL;
    apr_pool_create_ex(&subpool,pReq->pool,apr_pool_abort_get(pReq->pool),NULL);
    if(subpool == NULL) { return YMG_SUBPOOL_ERROR; }

    /* connect to database */
    sprintf(conninfo,"hostaddr=%s dbname=%s",hostip,database);
    conn = PQconnectdb(conninfo);
    if(PQstatus(conn) != CONNECTION_OK) {
        yLogResponse("DBSTORE","Couldn't connect to the database server.");
        PQfinish(conn);
        return YMG_DBCONN_ERROR;
    }

    /* check existance of ymage with this name */
    PQclear(PQexec(conn,"BEGIN"));
    sprintf(query,"SELECT * FROM ymg_desc WHERE ymg_name = '%s'",STR_FILE_NAME);
    res = PQexec(conn,query);
    PQclear(PQexec(conn,"COMMIT"));
    if(PQntuples(res) > 0) {
        yLogResponse("DBSTORE","The ymage already exists.");
        PQclear(res);
        PQfinish(conn);
        return YMG_ALREADY_EXISTS;
    }

    /* write the ymg_desc record */
    PQclear(PQexec(conn,"BEGIN"));
    sprintf(query,"INSERT INTO ymg_desc(ymg_name,width,height,num_layers) VALUES('%s',%d,%d,%d)",STR_FILE_NAME,stats[0],stats[1],stats[2]);
    res = PQexec(conn,query);
    PQclear(PQexec(conn,"COMMIT"));
    if(PQntuples(res) > 0) {
        PQclear(res);
        PQfinish(conn);
        return YMG_FAIL_YMGDESC;
    }

    /* send the data to postgresql */
    char filename[1024];
    int i = 0, layer = 0, row = 0, column = 0;
    apr_file_t * imagefile = NULL;
    unsigned char * imagebuffer = NULL;
    apr_size_t fsize = 0;
    int * max = apr_palloc(subpool,2*stats[2]*sizeof(int));

    /* begin transaction block */
PQclear(PQexec(conn, "BEGIN"));

/* defer constraints */
PQclear(PQexec(conn,"SET CONSTRAINTS fk_ymg_block DEFERRED"));

for(i = 0; i <= arraylrc->nelts - 3; i += 3) {
    /* extract binary data */
    imagefile = NULL;
    layer = ((int*)arraylrc->elts)[i];
    row = ((int*)arraylrc->elts)[i+1];
    column = ((int*)arraylrc->elts)[i+2];

    max[(2*layer)+0] = (int)fmax((float)max[(2*layer)+0],(float)row);
    max[(2*layer)+1] = (int)fmax((float)max[(2*layer)+1],(float)column);

    sprintf(filename,"/tmp/%s/%s_%d_%d_%d.jpg",STR_FILE_NAME,
            STR_FILE_NAME,layer,row,column);
    apr_file_open(&imagefile,filename,APR_READ|APR_BINARY,
                  APR_FPROT_UREAD|APR_FPROT_GREAD|APR_FPROT_WREAD,
                  subpool);  
    if(imagefile == NULL) {
        yLogResponse("DBSTORE","Couldn't open the ymage block file.");
        return YMG_FILEOPEN_ERROR;
    }
    apr_finfo_t finfo;
    apr_file_info_get(&finfo,APR_FINFO_SIZE,imagefile);
    fsize = finfo.size;

    /* create sub-subpool for the data that can be freed after use */
    sspool = NULL;
    apr_pool_create_ex(&sspool,subpool,apr_pool_abort_get(subpool),
                       NULL);
    if(sspool == NULL) { return YMG_SUBPOOL_ERROR; }

    imagebuffer = apr_palloc(sspool,fsize);
    if(imagebuffer == NULL) {
        apr_file_close(imagefile);
        return YMG_MEMALLOC_ERROR;
    }
    /* read, close, and delete the file */
    apr_file_read(imagefile,(void*)imagebuffer,&fsize);
    apr_file_close(imagefile);
    if(fsize != finfo.size) {
        return YMG_FILEREAD_ERROR;
    }
    /*apr_file_remove(filename,sspool);*/

    /* store binary data */
    ILubyte * edata = PQescapeBytea(imagebuffer,finfo.size,&fsize);
    if(edata == NULL) {
        PQfinish(conn);
        return YMG_DATAESCAPE_ERROR;
    }
    char * dataquery = apr_pstrcat(sspool,"INSERT INTO ymg_block(ymg_name,
                                        layer,"row","column",imagedata) VALUES('",STR_FILE_NAME,"",
                                        ",",apr_itoa(sspool,layer),
                                        ",",apr_itoa(sspool,row),
                                        ",
                                        apr_itoa(sspool,column),
                                        ",",edata,"/')",NULL);  
    res = PQexec(conn,dataquery);
    PQfreemem(edata);
    PQclear(res);
    apr_pool_clear(sspool);
}

/* store data info about each ymage block */
int w = stats[0]; int h = stats[1];
for(layer=0;layer<stats[2];layer++)
{
    sprintf(query,"INSERT INTO block_desc(yimg_name,layer,width,height,num_rows ,num_columns) VALUES('%s',%d,%d,%d,%d,%d','STR_FILE_NAME',layer, w/(max[(2*layer)+1]+1),h/(max[(2*layer)+0]+1),max[(2*layer)+0]+1, max((2*layer)+1)+1);
    res = PQexec(conn,query);
    yLogResponse("DBSTORE - CMD",query);
    PQclear(res);
    w /= 2; h /= 2;
}
/* end transaction block */
PQclear(PQexec(conn,"COMMIT"));

apr_pool_clear(subpool);
PQfinish(conn);
    return YMG_SUCCESS;
}
/* ----------------------------------------------- */
/* Download an image block from the database 
* @param: pReq   - pointer to the request object 
* @param: apreq  - pointer to request parser 
* @param: imagedata - pointer to bucket to be populated with binary image 
* */
YMG_FN_RESULT
yDownload(request_rec * pReq,apreq_handle_t * apreq ,apr_bucket_brigade * imagedata)
{
    PGconn * conn = NULL;
    PGresult *res = NULL;
    char conninfo[256];
    char query[1024];

    const char *hostip = "127.0.0.1";
    const char *database = "ymage_db";
    aprreq_param_t * param;
    int layer = -1, row = -1, column = -1;

    /* retrieve layer */ param = NULL;
    if((param = apreq_args_get(apreq,"layer")) != NULL)
    { layer = (int)apr_atoi64(param->v.data); }
    else
    { return YMG_MISSING_LAYER; }

    /* retrieve row */ param = NULL;
    if((param = apreq_args_get(apreq,"row")) != NULL)
    { row = (int)apr_atoi64(param->v.data); }
    else
    { return YMG_MISSING_ROW; }

    /* retrieve column */ param = NULL;
    if((param = apreq_args_get(apreq,"column")) != NULL)
    { column = (int)apr_atoi64(param->v.data); }
    else
    { return YMG_MISSING_COLUMN; }

    /* connect to database */
    sprintf(conninfo,"hostaddr=%s dbname=%s",hostip,database);
    conn = PQconnectdb(conninfo);
    if(PQstatus(conn) != CONNECTION_OK)
    {
        yLogResponse("DBDOWNLOAD","Couldn't connect to the database server.");
PQfinish(conn);
    return YMG_DBCONN_ERROR;
    }
/* request binary data */
PQclear(PQexec(conn, "BEGIN"));
sprintf(query, "SELECT imagedata FROM ymg_block WHERE ymg_name = '%s' AND layer = %d AND "row" = %d AND "column" = %d", STR_FILE_NAME, layer, row, column);
res = PQexec(conn, query);
PQclear(PQexec(conn, "COMMIT"));
if (PQntuples(res) < 1)
{
    PQclear(res);
PQfinish(conn);
    return YMG_BLOCK_NOT_FOUND;
}
size_t binarydatalen = 0;
unsigned char * binarydata = PQunescapeBytea(PQgetvalue(res, 0, 0), &binarydatalen);
/* pump data into a bucket brigade */
apr_bucket * bucket = apr_bucket_pool_create(binarydata, binarydatalen, pReq->pool, pReq->connection->bucket_alloc);
PQfreemem(binarydata);
APR_BRIGADE_INSERT_TAIL(imagedata, bucket);
PQclear(res);
PQfinish(conn);
return YMG_SUCCESS;

/* ------------------------------------------------ ---------------- */

/* Obtain metadata from the database server */
/* @param: pReq  - pointer to the request object */
/* @param: apreq - pointer to request parser */
/* @param: result - string pointer to contain requested info */
*/

YMG_FN_RESULT yGet(request_rec * pReq, apreq_handle_t * apreq, char ** result)
{
    PGconn * conn = NULL;
    PGresult * res = NULL;
    char conninfo[256];
    char query[1024];
    const char *hostip = "127.0.0.1";
    const char * database = "ymage_db";
    apreq_param_t * param;
    char * item;

    /* connect to database */
    sprintf(conninfo, "hostaddr=%s dbname=%s", hostip, database);
    conn = PQconnectdb(conninfo);
    if (PQstatus(conn) != CONNECTION_OK)
    {
        yLogResponse("DBGET","Couldn't connect to the database server.");
PQfinish(conn);
        return YMG_DBCONN_ERROR;
    }

    /* retrieve item */
    if (param = apreq_args_get(apreq,"item") != NULL)
    {
        item = apr_pstrdup(pReq->pool, param->v.data);
    }
    else
    {
        PQfinish(conn);
        return YMG_MISSING_ITEM;
    }

    /* obtain requested data */
    if (apr_strnatcasecmp(param->v.data, "YMAGELIST") == 0)
    {
        int i = 0;
    }
}
/* create and execute query */
PQclear(PQexec(conn,"BEGIN"));
sprintf(query,"SELECT ymg_name FROM ymg_desc");
res = PQexec(conn,query);
PQclear(PQexec(conn,"COMMIT"));
if(PQntuples(res) < 1)
{
    PQclear(res);
    PQfinish(conn);
    return YMG_INFO_NOT_FOUND;
}
*result = apr_psprintf(pReq->pool,"%s",PQgetvalue(res,0,0));
for(i=1;i<PQntuples(res);i++)
{
    *result = apr_psprintf(pReq->pool,"%s,%s",*result,
PQgetvalue(res,i,0));
}
else if(apr_strnatcasecmp(param->v.data,"METADATA") == 0)
{
    char * block;
    /* get the block field */
    if((param = apreq_args_get(apreq,"schema") ) != NULL)
    { block = apr_pstrdup(pReq->pool,param->v.data); }
    else
    { PQfinish(conn); return YMG_MISSING_SCHEMA; }
    /* determine whether to return block or full image data */
    if(apr_strnatcasecmp(param->v.data,"BLOCK") == 0)
    {
        char * layer;
        /* obtain layer */
        if((param = apreq_args_get(apreq,"layer") ) != NULL)
        { layer = apr_pstrdup(pReq->pool,param->v.data); }
        else
        { PQfinish(conn); return YMG_MISSING_LAYER; }
        /* create and execute query */
PQclear(PQexec(conn,"BEGIN"));
sprintf(query,"SELECT width, height, num_rows, num_columns FROM
block_desc WHERE ymg_name = '%s' AND layer = %s",
STR_FILE_NAME,layer);
res = PQexec(conn,query);
PQclear(PQexec(conn,"COMMIT"));
if(PQntuples(res) < 1)
{
    PQclear(res);
    PQfinish(conn);
    return YMG_INFO_NOT_FOUND;
}
    *result = apr_psprintf(pReq->pool,"%s;" width=%s;height=%s;num_rows=%s;num_columns=%s",
PQgetvalue(res,0,PQfnumber(res,"width")),
PQgetvalue(res,0,PQfnumber(res,"height")),
PQgetvalue(res,0,PQfnumber(res,"num_rows")),
PQgetvalue(res,0,PQfnumber(res,"num_columns")));
}
else if(apr_strnatcasecmp(param->v.data,"YMAGE") == 0)
{
    PQclear(PQexec(conn,"BEGIN"));
sprintf(query,"SELECT width, height, num_layers FROM ymg_desc WHERE
ymg_name = '%s',STR_FILE_NAME);
res = PQexec(conn,query);
PQclear(PQexec(conn,"COMMIT"));
if(PQntuples(res) < 1)
{
    PQclear(res);
PQfinish(conn);
    return YMG_INFO_NOT_FOUND;
}

*result = apr_psprintf(pReq->pool,
    "width=%s;height=%s;num_layers=%s",
    PQgetvalue(res,0,PQfnumber(res,"width")),
    PQgetvalue(res,0,PQfnumber(res,"height")),
    PQgetvalue(res,0,PQfnumber(res,"num_layers")));
}

else if(apr_strnatcasecmp(param->v.data,"ANNOTATION_USER") == 0)
{
    int i;

    PQclear(PQexec(conn,"BEGIN"));
    sprintf(query,"SELECT DISTINCT annot_user FROM ymg_annot WHERE
        ymg_name = '%s'",STR_FILE_NAME);
    res = PQexec(conn,query);
    PQclear(PQexec(conn,"COMMIT"));
    if(PQntuples(res) < 1)
    {
        PQclear(res);
        PQfinish(conn);
        return YMG_INFO_NOT_FOUND;
    }
    *result = apr_psprintf(pReq->pool,"%s",PQgetvalue(res,0,0));
    for(i=1;i<PQntuples(res);i++)
    {
        *result = apr_psprintf(pReq->pool,"%s,%s",*result,
            PQgetvalue(res,i,0));
    }
}
else
{
    PQclear(res);
    PQfinish(conn);
    return YMG_INVALID_SCHEMA_ARG;
}

else if(apr_strnatcasecmp(param->v.data,"ANNOTATION") == 0)
{
    int i;
    char * user;

    /* obtain layer */
    if((param = apreq_args_get(apreq,"user")) != NULL)
    { user = apr_pstrdup(pReq->pool,param->v.data); }
    else
    { PQfinish(conn); return YMG_MISSING_USER; }

    /* create and execute query */
    PQclear(PQexec(conn,"BEGIN"));
    sprintf(query,"SELECT pos_x, pos_y, annotation FROM ymg_annot WHERE
        ymg_name = '%s' AND annot_user = '%s'",STR_FILE_NAME,user);
    res = PQexec(conn,query);
    PQclear(PQexec(conn,"COMMIT"));
    if(PQntuples(res) < 1)
    {
        PQclear(res);
        PQfinish(conn);
        return YMG_INFO_NOT_FOUND;
    }
    *result = apr_psprintf(pReq->pool,"x=%s||y=%s||%s",
        PQgetvalue(res,0,PQfnumber(res,"pos_x")),
        PQgetvalue(res,0,PQfnumber(res,"pos_y")),
        PQgetvalue(res,0,PQfnumber(res,"annotation")));
    for(i=1;i<PQntuples(res);i++)
{ *result = apr_psprintf(pReq->pool,"%s\nx=%s||y=%s||%s",*result,
PQgetvalue(res,i,PQfnumber(res,"pos_x")),
PQgetvalue(res,i,PQfnumber(res,"pos_y")),
PQgetvalue(res,i,PQfnumber(res,"annotation")));
}
PQclear(res);
PQfinish(conn);
}

return YMG_SUCCESS;

YMG_FN_RESULT yAnnotate(request_rec * pReq,apreq_handle_t * apreq)
{
apreq_param_t * param = NULL;
if((param = apreq_body_get(apreq,"annotation")) != NULL)
{
/* obtain all arguments needed to build the query */
char * annotation = apr_pstrdup(pReq->pool,param->v.data);

if((param = apreq_args_get(apreq,"user")) == NULL)
{
return YMG_MISSING_USER;
}
char * user = apr_pstrdup(pReq->pool,param->v.data);

if((param = apreq_args_get(apreq,"x")) == NULL)
{
return YMG_MISSING_POSITION_ARG;
}
char * x = apr_pstrdup(pReq->pool,param->v.data);

if((param = apreq_args_get(apreq,"y")) == NULL)
{
return YMG_MISSING_POSITION_ARG;
}
char * y = apr_pstrdup(pReq->pool,param->v.data);

PGconn * conn = NULL;
PGresult *res = NULL;
char conninfo[256];
char * query = NULL;

const char *hostip = "127.0.0.1";
const char *database = "ymage_db";

/* connect to database */
sprintf(conninfo,"hostaddr=%s dbname=%s",hostip,database);
conn = PQconnectdb(conninfo);
if(PQstatus(conn) != CONNECTION_OK)
{
yLogResponse("DBANNOTATE","Couldn't connect to the database server.");
PQfinish(conn);
return YMG_DBCONN_ERROR;
}

/* create and execute query */
PQclear(PQexec(conn,"BEGIN"));
size_t lenin = strlen(annotation);
size_t lenout = 0;
unsigned char * annot_escaped = PQescapeBytea(annotation,lenin,&lenout);
query = apr_pstrcat(pReq->pool,""INSERT INTO ymg_annot(ymg_name,annot_user,
annot_id,pos_x,pos_y,annotation) VALUES('",STR_FILE_NAME,"','"user",""get_next_annot_id("",STR_FILE_NAME,"","",user,""),"",x,"",y,"",annot_escaped,"'"),NULL);
res = PQexec(conn,query);
PQclear(res);
PQfreemem(annot_escaped);
PQclear(PQexec(conn,"COMMIT"));
PQfinish(conn);

else
{
    return YMG_NO_ANNOTATION;
}

return YMG_SUCCESS;

/*********************************************/
/* *** TEXT RESPONSE FNS *** */
/* ------------------------------------------------ ---------------- */
/* Output HTTP response to the user. */
/* @param: pReq          - pointer to the current request */
/* @param: request_type  - name of current request process */
/* @param: error_string  - comment about problem or success */
/* @param: string_color   - color of error_string text */
/* */
void
yPrintResponse(request_rec * pReq,char * request_type,char * error_string,char * string_color)
{
    /*if(apr_strnatcmp(pReq->content_type,"text/html") != 0)
    { }*/
    pReq->content_type = "text/html"
    ap_rprintf(pReq,"<span><b>MOD_YMAGE</b> [%s]</span> <span style="color:%s;">%s</span>",request_type,string_color,error_string);

} /* ------------------------------------------------ ---------------- */
/* Output response to Apache error log. */
/* @param: request_type  - name of current request process */
/* @param: error_string  - comment about problem or success */
/* */
void
yLogResponse(char * request_type,char * error_string)
{
    /*apr_time_t ntime = apr_time_now();
    apr_time_exp_t stime; apr_time_exp_gmt(&stime,ntime);
    */
    fprintf(stderr,"MOD_YMAGE [%s]: %s",request_type,error_string);

} /* ------------------------------------------------ ---------------- */
/* *******************************************************/
C.3 Image Subdivision Header File (def_seg.h)

```c
#ifndef DEF_SEG_H_
#define DEF_SEG_H_

/* *** CONSTANTS *** */
#define THRESHOLD_FACTOR 0.75

/* *** APACHE PORTABLE RUNTIME LIBRARY *** */
#include "apr.h"
#include "apr_buckets.h"
#include "apr_file_io.h"
#include "apr_general.h"
#include "apr_strings.h"
#include "apr_tables.h"
#include "apr_time.h"

/* *** APACHE HTTPD API *** */
#include "ap_config.h"
#include "httpd.h"
#include "http_config.h"
#include "http_log.h"
#include "http_protocol.h"
#include "http_request.h"

/* *** OPENIL/DEVIL LIBRARY *** */
#include "IL.h"
#include "ILU.h"
#include "ILUT.h"

/* *** MISC. C UTILITY LIBRARIES *** */
#include "bool.h"

/* *** CONSTANTS / RETURN CODES *** */
typedef enum _DEFSEG_RESULT
{
    DEFSEG_SUCCESS,
    ARRAYLRC_NULL,
    INVALID_DIMENSION,
    IMAGE_LOAD_FAIL,
    MEMORY_ERROR
} DEFSEG_RESULT;

/* *** FUNCTIONS *** */
extern ILint ydsDivide(ILint /*dimension*/, ILint /*target*/);

extern DEFSEG_RESULT ydsSegment(request_rec * /*pReq*/, ILint /*target_dimension*/,
char * /*imagepath*/, char * /*outpath*/, char * /*savename*/, apr_array_header_t * /*arraylrc*/, int * /*stats*/);

#endif /*DEF_SEG_H_*/
```
C.4 Image Subdivision Header File (def_seg.c)

```c
#include "def_seg.h"

/**************************** ***************************************************/

/* Evenly divide dimension until it is at least as small as newmaxdim */
/* * @param: dimension - the original dimension */ *
/* @param: target - the goal dimension */ */

ILint
ydsDivide(ILint dimension, ILint target)
{
    /* use the first 30 prime numbers for division */
    const int PRIMES = 30;
    ILint tmp, original = dimension;
    ILint threshold = THRESHOLD_FACTOR * target;
    int i;

    /* divide down the dimension */
    while(dimension > target)
    {
        tmp = 0;

        /* find the largest divisor */
        for(i=0; i<PRIMES; i++)
        {
            /* check divisibility by prime number */
            if((dimension % prime[i]) == 0)
            {
                ILint newdim = dimension/prime[i];
                /* found a divisor that is not too big */
                if(newdim >= target) { break; }
                /* found a divisor that works, but might be too large, keep looking */
                else if(newdim >= threshold) { tmp = prime[i]; }
            }
        }

        /* divisor was found */
        if(i < PRIMES)
        {
            dimension = dimension / prime[i];
        }
        /* divisor was found and skipped */
        else if(i == PRIMES && tmp != 0)
        {
            dimension = dimension / tmp;
        }
        /* no first divisor was found, trim off one pixel and try again */
        else if(dimension == original)
        {
            dimension--;
        }
        /* couldn't divide by any more */
        else
        {
            break;
        }
    }

    return dimension;
}

/**************************** ***************************************************/

/* Create the image blocks and return a linked list of the file names */
```
DEFSEG_RESULT
ydsSegment(request_rec * pReq,ILint target_dimension, const char * imagepath, const char * outpath, const char * savename, apr_array_header_t * arraylrc, int * stats)
{
    ilInit();
    iluInit();
    ILuint image, img;
    ILint width, height, w, h;
    ILint row, col;
    ILint format, type;
    char tmppstr[256];
    int layer = 0;

    /* create a subpool */
    apr_pool_t * subpool = NULL;
    apr_pool_create_ex(&subpool,pReq->pool,apr_pool_abort_get(pReq->pool),NULL);
    if(subpool == NULL)
    {
        ilShutDown();
        return MEMORY_ERROR;
    }

    /* ensure arraylrc is valid */
    if(arraylrc == NULL) { return ARRAYLRC_NULL; }

    ILint target_width, target_height;
    target_width = target_height = target_dimension;

    /* return error on invalid dimension entry */
    if(target_dimension <= 0) { return INVALID_DIMENSION; }

    /* create image-names */
    ilGenImages(1,&image);
    ilGenImages(1,&img);

    /* load the file into memory */
    ilBindImage(image);
    if(!ilLoadImage(imagepath))
    {
        ilDeleteImages(1,&img);
        ilDeleteImages(1,&image);
        ilShutDown();
        return IMAGE_LOAD_FAIL;
    }
    ilConvertImage(IL_BGR, IL_UNSIGNED_BYTE);

    /* adjust target using aspect ratio */
    format = ilGetInteger(IL_IMAGE_FORMAT);
    type = ilGetInteger(IL_IMAGE_TYPE);
    width = ilGetInteger(IL_IMAGE_WIDTH);
    height = ilGetInteger(IL_IMAGE_HEIGHT);
    target_width *= (float)width / (float)height;
    stats[0] = width; stats[1] = height;

    /* adjust the utility image */
    ilBindImage(img);
    ilConvertImage(format,type);
    /* ilRegisterOrigin(IL_ORIGIN_UPPER_LEFT);
    ilSetInteger(IL_JPG_QUALITY,85);
    /* ilSetInteger(IL_PNG_INTERLACE,IL_TRUE); */
    /* ilSetInteger(IL_SAVE_INTERLACED,IL_TRUE); */

    /* obtain dimensions */
    w = ydsDivide(width,target_width);
    h = ydsDivide(height,target_height);

    /* generate image segments */
do
{
  /* resize the utility image */
ilBindImage(img);
iluScale(w,h,1);

Ilvoid * data = apr_palloc(subpool,w*h*3*sizeof(ILubyte));
if(data == NULL)
{
ilDeleteImages(1,&img);
ilDeleteImages(1,&image);
ilShutDown();
return MEMORY_ERROR;
}
for(row = 0; row <= height - h; row += h)
{
  for(col = 0; col <= width - w; col += w)
  {
    int r = row/h;
    int c = col/w;
    /* ilBlit(image,0,0,0,col,row,0,w,h,1); */
ilBindImage(image);
ilCopyPixels(col,row,0,w,h,1,IL_BGR,IL_UNSIGNED_BYTE,data);
ilSetPixels(0,0,0,w,h,1,IL_BGR,IL_UNSIGNED_BYTE,data);
sprintf(tmpstr,"%s/%s_%d_%d_%d.jpg",outpath,savename,layer,r,c);
    (*((int*)apr_array_push(arraylrc))) = layer;
    (*((int*)apr_array_push(arraylrc))) = r;
    (*((int*)apr_array_push(arraylrc))) = c;
ilSave(IL_JPG,tmpstr);
    /* use jpegtran to make progressive? ...mmm..probably not */
    char cmd[8192];
sprintf(cmd,"jpegtran -progressive %s > %s",tmpstr,tmpstr);
system(cmd);
  }
}
apr_pool_clear(subpool);
/* next level */
layer++;

/* scale image to dimensions for this level */
ilBindImage(image);
iluImageParameter(ILU_FILTER,ILU_NEAREST);
width *= 0.5; height *= 0.5;
iluScale(width,height,1);

/* obtain list of divisors */
w = ydsDivide(width,target_width);
h = ydsDivide(height,target_height);

/* generate zoom out levels until you've zoomed everything into view */
}while(w > (THRESHOLD_FACTOR*target_width) && h >
              (THRESHOLD_FACTOR*target_height));
apr_pool_destroy(subpool);
/* save level */
stats[2] = layer;
ilDeleteImages(1,&img);
ilDeleteImages(1,&image);
ilShutDown();
return DEFSEG_SUCCESS;
APPENDIX D

Source Code Listing for Visualization Applet

D.1 Ymage Class (Ymage.java)

```java
/*
 * @program: Ymage Viewer Application (LOGIC)
 * @author: Yetu Yachim
 * @date: Spring 2007
 */

public class Ymage {
    /* *** MEMBERS *** */
    protected int m_Width;
    protected int m_Height;
    protected int m_NumLayers;
    protected YmageBlock[] m_Blocks = null;

    /* *** CONSTRUCTORS *** */
    public Ymage() {
        m_Width = -1;
        m_Height = -1;
        m_NumLayers = -1;
        m_Blocks = null;
    }

    public Ymage(int width, int height, int layers) {
        setWidth(width);
        setHeight(height);
        setNumLayers(layers);
    }

    /* *** PUBLIC METHODS *** */

    //Set Accessor Methods
    public void setWidth(int width) {
        m_Width = width;
    }

    public void setHeight(int height) {
        m_Height = height;
    }

    public void setNumLayers(int numlayers) {
        m_NumLayers = numlayers;
    }

    //Get Accessor Methods
    public int getWidth() {
        return m_Width;
    }

    public int getHeight() {
        return m_Height;
    }

    public int getNumLayers() {
```
public void setBlocks(YmageBlock [] blocks) {
    m_Blocks = blocks;
}

public YmageBlock getBlock(int layer) {
    return m_Blocks[layer];
}

/* SUB CLASSES */
public static class YmageBlock {
    /* Protected Members */
    protected int m_Width;
    protected int m_Height;
    protected int m_NumRows;
    protected int m_NumColumns;

    /* CONSTRUCTORS */
    public YmageBlock() {
        m_Width = -1;
        m_Height = -1;
        m_NumRows = -1;
        m_NumColumns = -1;
    }

    public YmageBlock(int width, int height, int rows, int columns) {
       .setWidth(width);
       .setHeight(height);
        setNumRows(rows);
        setNumColumns(columns);
    }

    /* PUBLIC METHODS */
    // Set Accessor Methods
    public void setWidth(int width) {
        m_Width = width;
    }

    public void setHeight(int height) {
        m_Height = height;
    }

    public void setNumRows(int numrows) {
        m_NumRows = numrows;
    }

    public void setNumColumns(int numcolumns) {
        m_NumColumns = numcolumns;
    }

    // Get Accessor Methods
    public int getWidth() {
        return m_Width;
    }

    public int getHeight() {
        return m_Height;
    }

    return m_NumLayers;
}
public int getNumRows()
{
    return m_NumRows;
}

public int getNumColumns()
{
    return m_NumColumns;
}
D.2 YmageViewer Class (YmageViewer.java)

```java
import java.util.*;
import java.net.*;
import java.io.*;

public class YmageViewer
{
    /* *** MEMBERS *** */
    protected static YmageViewer m_This = null;
    protected volatile Ymage m_Ymage = null;
    protected volatile String m_CurrentYmageName = "";
    protected volatile int m_CurrentLayer = 0;
    protected volatile Hashtable m_ImageHash = null;
    protected int m_Zoom = 0;
    protected URL m_ServerURL = null;

    /* *** CONSTRUCTOR *** */
    private YmageViewer()
    {
        m_Ymage = new Ymage();
        m_ImageHash = new Hashtable();
        try { m_ServerURL = new URL("http://localhost:800/"); }
        catch(Exception ex) { }
    }

    public static YmageViewer getViewer()
    {
        if(m_This == null)
        {
            m_This = new YmageViewer();
        }

        return m_This;
    }

    /* *** PUBLIC METHODS *** */

    //Set Accessor Methods
    public synchronized void setCurrentYmageName(String ymagename)
    {
        m_CurrentYmageName = ymagename;
    }

    public synchronized void setCurrentLayer(int layernumber)
    {
        m_CurrentLayer = layernumber;
    }

    public synchronized void setServerURL(URL url)
    {
        m_ServerURL = url;
    }

    public synchronized void setZoom(int zoomfactor)
    {
        m_Zoom = zoomfactor;
    }

    public synchronized void setYmageBlocks(Ymage.YmageBlock [] blocks)
```

75
public synchronized void setYmageWidth(int width)
{
    m_Ymage.setWidth(width);
}

public synchronized void setYmageHeight(int height)
{
    m_Ymage.setHeight(height);
}

public synchronized void setYmageLayerCount(int numlayers)
{
    m_Ymage.setNumLayers(numlayers);
}

// Add Accessor Methods
public synchronized void addImage(Object imagename, Object image)
{
    m_ImageHash.put(imagename, image);
}

// Remove Accessor Methods
public synchronized void clearImages()
{
    /* Enumeration keylist = m_ImageHash.keys();
    while(keylist.hasMoreElements())
    {
        m_ImageHash.remove(keylist.nextElement());
    }
    keylist = null; */
    m_ImageHash.clear();
    System.gc();
}

// Get Accessor Methods
public String getCurrentYmageName()
{
    return m_CurrentYmageName;
}

public int getCurrentLayer()
{
    return m_CurrentLayer;
}

public URL getServerURL()
{
    return m_ServerURL;
}

public int getZoom()
{
    return m_Zoom;
}

public Ymage.YmageBlock getYmageBlock(int layer)
{
    return m_Ymage.getBlock(layer);
}

public Ymage.YmageBlock getCurrentYmageBlock()
{
    return m_Ymage.getBlock(getCurrentLayer());
}

public int getYmageWidth()
{
public int getImageHeight()
{
    return m_Yimage.getHeight();
}

public int getImageLayerCount()
{
    return m_Yimage.getNumLayers();
}

public Object getImage(Object imagename)
{
    return m_ImageHash.get(imagename);
}

//Utilities
public String [] getImageList() throws Exception
{
    String [] ymagelist = null;

    //establish a connection with the web server
    Socket socket = new Socket(getServerURL().getHost(),getServerURL().getPort());
    InputStream in = socket.getInputStream();
    OutputStream out = socket.getOutputStream();
    PrintWriter writer = new PrintWriter(new OutputStreamWriter(out),true);
    BufferedReader bufIn = new BufferedReader(new InputStreamReader(in));

    //send request
    writer.println("GET /0.ymg?cmd=GET&item=YMAGELIST HTTP/1.0\n");

    //obtain response
    StringBuffer result = new StringBuffer();
    String line; int skip = 0;
    while((line = bufIn.readLine())!=null)
    {
        if(skip++<6) continue;
        result.append(line);
    }
    in.close(); out.close();
    socket.close();

    //process and return response
    ymagelist = result.toString().split(",");

    return ymagelist;
}

public void loadYmage() throws Exception
{
    Socket socket = null;
    InputStream in = null;
    OutputStream out = null;
    PrintWriter writer = null;
    BufferedReader bufIn = null;

    StringBuffer result;
    String [] keyval;
    String [] tmp;

    //int count = 0; //make a maximum of 15 attempts
    //while(count++ < 15 && m_Ymage.getNumLayers() < 0)
    {
        //establish a connection with the web server
        socket = new Socket(getServerURL().getHost(),getServerURL().getPort());
        in = socket.getInputStream();
    }
```java
out = socket.getOutputStream();
writer = new PrintWriter(new OutputStreamWriter(out), true);
bufIn = new BufferedReader(new InputStreamReader(in));

//send request
writer.println("GET /"+getCurrentYmageName()+".ymg?
cmd=GET&item=METADATA&schema=YIMAGE HTTP/1.0\n");

//obtain response
result = new StringBuffer();
String line; int skip = 0;
while((line = bufIn.readLine()) != null){
    if(skip++ < 6) continue;
    result.append(line);
}

//parse response
keyval = result.toString().split(";");
for(int i=0;i<keyval.length;i++){
    tmp = keyval[i].split("=");
    if(tmp[0].compareToIgnoreCase("width") == 0)
        m_Ymage.setWidth(Integer.parseInt(tmp[1]));
    else if(tmp[0].compareToIgnoreCase("height") == 0)
        m_Ymage.setHeight(Integer.parseInt(tmp[1]));
    else if(tmp[0].compareToIgnoreCase("num_layers") == 0)
        m_Ymage.setNumLayers(Integer.parseInt(tmp[1]));
}
in.close(); out.close();
socket.close();

//create an array of block descriptors and instantiate them
Ymage.YmageBlock [] blocks = new Ymage.YmageBlock[m_Ymage.getNumLayers()];
int k = 0;
for(k=0;k<blocks.length;k++)
    blocks[k] = new Ymage.YmageBlock();

//count = 0; k--; //make a maximum of 15 attempts
//while(count++ < 15 && (blocks[k].getWidth()==-1 || blocks[k].getWidth()==-1|| blocks[k].getNumRows()==-1 || blocks[k].getNumColumns()==-1))
{
    for(int i=0;i<blocks.length;i++)
    {
        //establish connection with the webserver
        socket = new Socket(getServerURL().getHost(),getServerURL().getPort());
        socket.setTcpNoDelay(true);
        in = socket.getInputStream();
        out = socket.getOutputStream();
        writer = new PrintWriter(new OutputStreamWriter(out),true);
        bufIn = new BufferedReader(new InputStreamReader(in));

        //send request
        writer.println("GET /"+m_CurrentYmageName()+".ymg?cmd=GET &item=METADATA&schema=BLOCK&layer="+i+" HTTP/1.0\n");

        //obtain response
        result = new StringBuffer();
        String line; int skip = 0;
        while((line = bufIn.readLine()) != null)
```
if (skip++ < 6) continue;
result.append(line);

//parse response
keyval = result.toString().split(";");
for (int j = 0; j < keyval.length; j++)
{
    tmp = keyval[j].split("=");
    if (tmp[0].compareToIgnoreCase("width") == 0)
    {
        blocks[i].setWidth(Integer.parseInt(tmp[1]));
    }
    else if (tmp[0].compareToIgnoreCase("height") == 0)
    {
        blocks[i].setHeight(Integer.parseInt(tmp[1]));
    }
    else if (tmp[0].compareToIgnoreCase("num_rows") == 0)
    {
        blocks[i].setNumRows(Integer.parseInt(tmp[1]));
    }
    else if (tmp[0].compareToIgnoreCase("num_columns") == 0)
    {
        blocks[i].setNumColumns(Integer.parseInt(tmp[1]));
    }
}
in.close(); out.close();
socket.close();
//System.gc();
}
m_Ymage.setBlocks(blocks);
D.3 YmageUploader Class (YmageUploader.java)

```java
import java.awt.image.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.util.Iterator;
import javax.imageio.ImageIO;
import javax.imageio.ImageReadParam;
import javax.imageio.ImageReader;
import javax.imageio.ImageWriter;
import javax.imageio.stream.ImageInputStream;
import javax.imageio.stream.ImageOutputStream;
import java.util.Random;

public class YmageUploader
{

protected static YmageUploader m_This = null;
protected volatile File m_ImagePath = null;
protected int m_SubSampler = 1;
protected volatile int m_Progress = 0;

public static YmageUploader getUploader()
{
    if (m_This == null)
    {
        m_This = new YmageUploader();
    }
    return m_This;
}

public String getFullImagePath()
{
    try
    {
        return m_ImagePath.getAbsolutePath();
    }
    catch(Exception ex)
    {
        return null;
    }
}

public String getImageFileName()
```
```java
{ 
    try 
    { 
        String ext = m_ImagePath.getName().toString();
        return ext.split("\.")[0];
    } 
    catch(Exception ex) 
    { 
        return null;
    } 
} 

public String getImageFileExt() 
{ 
    try 
    { 
        String ext = m_ImagePath.getName().toString();
        return ext.split("\.")[1];
    } 
    catch(Exception ex) 
    { 
        return null;
    } 
} 

public Image getImage(Dimension max_dim, Rectangle region) 
{ 
    BufferedImage image = null;
    Iterator readers = ImageIO.getImageReadersByFormatName(getImageFileExt());
    ImageReader reader = (ImageReader)readers.next();
    InputStream in = null;
    ImageInputStream iis = null;
    try 
    { 
        in = (new URL("file:///"+getFullImagePath())).openStream();
        iis = ImageIO.createImageInputStream(in);
        reader.setInput(iis, true);

        int w = reader.getWidth(0);
        int h = reader.getHeight(0);
        ImageReadParam param = reader.getDefaultReadParam();
        if(max_dim != null) 
        { 
            m_SubSampler = Math.max(w, h);
            if(m_SubSampler == w && m_SubSampler > max_dim.width)
                { m_SubSampler = m_SubSampler / max_dim.width; }
            else if(m_SubSampler > max_dim.height)
                { m_SubSampler = m_SubSampler / max_dim.height; }
            else 
                { m_SubSampler = 1; }
            param.setSourceSubsampling(m_SubSampler, m_SubSampler, 0, 0);
        }

        if(region != null) 
        { 
            region.x *= m_SubSampler;
            region.y *= m_SubSampler;
            region.width *= m_SubSampler;
            region.height *= m_SubSampler;
            param.setSourceRegion(region);
            param.setSourceSubsampling(Math.max(m_SubSampler/4,1),
                                         Math.max(m_SubSampler/4,1),0,0);
        }

        image = reader.read(0, param);
        iis.close(); in.close();
    } 
    catch(Exception ex) 
    { 
```
image = null;
}

return image;
}

//Utilities
public int getProgress()
{
    return m_Progress;
}

public void setProgress(int progress)
{
    m_Progress = progress;
}

public String uploadImage(URL serverquery, Rectangle region) throws Exception
{
    if(m_ImagePath == null)
    {
        throw new Exception("No image specified");
    }

    m_Progress = 0;
    File uploadfile = new File(m_ImagePath.getPath());
    if(region != null)
    {
        BufferedImage image = null;
        ImageReader reader = (ImageReader)ImageIO.getImageReadersByFormatName(getImageFileExt()).next();
        InputStream in = null;
        ImageInputStream iis = null;
        ImageOutputStream ios = null;

        //Load the desired region
        in = (new URL("file:///" + getFullImagePath())).openStream();
        iis = ImageIO.createImageInputStream(in);
        reader.setInput(iis, true);
        ImageReadParam param = reader.getDefaultReadParam();
        region.x *= m_SubSampler;
        region.y *= m_SubSampler;
        region.width *= m_SubSampler;
        region.height *= m_SubSampler;
        param.setSourceRegion(region);
        image = reader.read(0, param);
        iis.close(); in.close();

        //Save the desired region to a file
        ImageWriter writer = (ImageWriter)ImageIO.getImageWritersByFormatName(getImageFileExt()).next();
        String newfile = m_ImagePath.getAbsolutePath().replaceAll(getImageFileName(), getImageFileName() + "_region" + (new Random()).nextInt());
        uploadfile = new File(newfile);
        ios = ImageIO.createImageOutputStream(uploadfile);
        writer.setOutput(ios);
        writer.write(image);
    }
    m_Progress = 10;

    HttpURLConnection conn = (HttpURLConnection)serverquery.openConnection();
    // now connection is configured write the queryFile contents to the
    // outputStream
    conn.setRequestMethod("POST");
    conn.setDoOutput(true);
    conn.setDoInput(true);
    conn.setUseCaches(false);
Long uploadsize = 0; uploadfile.length();
byte[] data = new byte[1000];
FileInputStream imagein = new FileInputStream(uploadfile);
int cnt = imagein.read(data);
while (cnt > 0) //== data.length)
{
    out.write(data, 0, cnt);
    uploadsize += cnt;
    m_Progress = 10 + (int)(90*(float)uploadsize/ (float)uploadfile.length());
    cnt = imagein.read(data);
    /*if((uploadsize % 100000) == 0)
    {
        System.gc();
        out.flush();
    }*/
}
//out.write(data,0,cnt);
out.writeBytes("\n--****4353--\n\n");
out.close();
imagein.close();

// read response and write to System.out
BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
StringBuffer response = new StringBuffer();
String line;
while((line = in.readLine()) != null)
{ response.append(line); }
in.close();
m_Progress = 100;

conn.disconnect();
return response.toString();
}
D.4 YmageAnnotator Class (YmageAnnotator.java)

```java
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.URL;

public class YmageAnnotator {
    /* *** MEMBERS *** */
    protected static YmageAnnotator m_This = null;
    protected String m_CurrentYmageName = "";
    protected URL m_ServerURL = null;

    /* *** CONSTRUCTOR *** */
    protected YmageAnnotator() {
    }

    public static YmageAnnotator getAnnotator() {
        if (m_This == null) {
            m_This = new YmageAnnotator();
        }
        return m_This;
    }

    /* *** PUBLIC METHODS *** */

    // Set Accessor Methods
    public synchronized void setCurrentYmageName(String ymagename) {
        m_CurrentYmageName = ymagename;
    }

    public synchronized void setServerURL(URL url) {
        m_ServerURL = url;
    }

    // Get Accessor Methods
    public String getCurrentYmageName() {
        return m_CurrentYmageName;
    }

    public URL getServerURL() {
        return m_ServerURL;
    }

    public String[] getUserList() {
        String[] users = null;
        Socket socket = null;
        InputStream in = null;
        OutputStream out = null;
        try {
            
```
//establish a connection with the web server
socket = new Socket(getServerURL().getHost(),getServerURL().getPort());
in = socket.getInputStream();
out = socket.getOutputStream();

PrintWriter writer = new PrintWriter(new OutputStreamWriter(out),true);
BufferedReader bufIn = new BufferedReader(new InputStreamReader(in));

//send request
writer.println("GET /"+getCurrentYmageName()+.ymg?cmd=GET&
item=METADATA&schema=ANNOTATION_USER HTTP/1.0 
";

//obtain response
StringBuffer result = new StringBuffer();
String line;
int skip = 0;
while((line = bufIn.readLine())!=null)
{
    if(skip++<6) continue;
    result.append(line);
}
socket.close();

//process and return response
if(result.indexOf(",")<0)
{ users = null; }
else
{ users = result.toString().split(","); }

catch(Exception ex) { }
finally
{
    try { in.close(); } catch(Exception ex) { }
    try { out.close(); } catch(Exception ex) { }
    try { socket.close(); } catch(Exception ex) { }
}

return users;
D.5 ImgPanel Class (ImgPanel.java)

/*
 * @program: Ymage Viewer Application (GUI)
 * @author: Yetu Yachim
 * @date: Spring 2007
 */
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;
import java.net.*;
import javax.imageio.*;
import javax.imageio.event.*;
import javax.imageio.stream.*;
import java.util.*;
import java.io.*;
public class ImgPanel extends JComponent
{
    /* *** MEMBERS *** */
    static final long serialVersionUID = 1;
    protected Image m_Image = null;
    protected Rectangle m_Rect = null;
    protected InputStream m_In = null;
    protected ImageInputStream m_Iis = null;
    /* *** CONSTRUCTORS *** */
    public ImgPanel() { }
    public ImgPanel(URL url) throws Exception
    {
        loadimage(url);
    }
    public ImgPanel(Image image) {
        m_Image = image;
    }
    /* *** PUBLIC METHODS *** */
    //Set Accessor Methods
    public void setImage(URL url) throws Exception
    {
        loadimage(url);
    }
    public void setImage(Image img) {
        m_Image = img;
    }
    public void setRect(Rectangle rect) {
        m_Rect = rect;
    }
    //Get Accessor Methods
    public Image getImage() {
        return m_Image;
    }
    //Utility Methods
    private void loadimage(URL url) {
        //System.out.println(url.getProtocol()+"://"+url.getHost()+":"+url.getPort());
        //return;
    }
}
Iterator readers = ImageIO.getImageReadersByFormatName("jpg");
ImageReader reader = (ImageReader)readers.next();
try {
    m_In = url.openStream();
    m_Iis = ImageIO.createImageInputStream(m_In);
    reader.setInput(m_Iis, true);
    //reader.addIIOReadUpdateListener(new ProgressiveListener(this));
    m_Image = reader.read(0);
    //repaint();
} catch (Exception ex) { }
finally {
    closeConnections();
}

public boolean closeConnections() {
    boolean ret = true;
    try { m_Iis.close(); } catch (Exception ex) { ret = false; }
    try { m_In.close(); } catch (Exception ex) { ret = false; }
    return ret;
}

public synchronized void paintComponent(Graphics g) {
    if (m_Image == null) { return; }
    int w = getWidth();
    int h = getHeight();
    g.drawImage(m_Image.getScaledInstance(w, h, Image.SCALE_FAST), 0, 0, w, h, this);
    g.drawImage(m_Img, 0, 0, getWidth(), getHeight(), this);
    if (m_Rect != null) {
        g.setXORMode(Color.ORANGE);
        g.drawRect(m_Rect.x, m_Rect.y, m_Rect.width, m_Rect.height);
    }
}

/* *** Interlace loading handler *** */
public class ProgressiveListener implements IIOReadUpdateListener {
  protected ImgPanel m_ImgPanel = null;
  int count = 0;
  public ProgressiveListener(ImgPanel imgpanel) {
    m_ImgPanel = imgpanel;
  }
  public void thumbnailPassStarted(ImageReader source, BufferedImage theImage, int pass, int minPass, int maxPass, int minX, int minY, int periodX, int periodY, int[] bands) {
  }
  public void thumbnailUpdate(ImageReader source, BufferedImage theImage, int minX, int minY, int width, int height, int periodX, int periodY, int[] bands) {
  }
  public void thumbnailPassComplete(ImageReader source, BufferedImage theImage) {
  }
  public void imageUpdate(ImageReader source, BufferedImage theImage, int minX, int minY, int width, int height, int periodX,
public void passStarted(ImageReader source, BufferedImage theImage, int pass, int minPass, int maxPass, int minX, int minY, int periodX, int periodY, int[] bands)
{
    //System.out.println("starting a pass");
}

public void passComplete(ImageReader source, BufferedImage theImage)
{
    //m_ImgPanel.setImage(theImage);
    //m_ImgPanel.update(m_ImgPanel.getGraphics());
}
public class YmageViewerGUI extends JApplet {

static final long serialVersionUID = 1;

protected YmageViewer m_AppViewer = null; // @jve:decl-index=0:
protected YmageUploader m_AppUploader = null; // @jve:decl-index=0:
protected YmageAnnotator m_AppAnnotator = null; // @jve:decl-index=0:
protected Thread m_DisplayThread = null; // @jve:decl-index=0:
protected Dimension m_SpSize = null;
protected Point m_StartPan = null;
protected boolean m_KillDisplayThread = false;

protected YmageUploader m_AppUploader = null;
private JButton btnOK = null;
private JTextField tfServerURL = null;
private JLabel lblURL = null;
private JPanel plConfirmSettings = null;
private JButton btnCancel = null;
private JButton btnOK = null;
private JButton btnUpload = null;
private JTextField tfYmgName = null;
private JProgressBar pbUploadProgress = null;
private JPanel plImageOverlay = null;
private JPanel plOverlay = null;
private JPanel plNoJitterMiniImage = null;
private JPanel plMiniImage = null;
private JPanel plCenterScroller = null;
private JMenu muAnnotation = null;
private JMenuItem miShowAll = null;
private JSeparator sepAnnotationMenu = null;
private JMenuItem miShowAll = null;
private JSeparator sepAnnotationMenu = null;
private JMenuItem miUpload = null;
private JMenu muApplication = null;
private JPanel pnAppletRoot = null;
private JPanel plViewer = null;
private JMenuBar mbMainMenu = null;
private JMenu muApplication = null;
private JPanel plSettings = null;
private YmageUploader m_AppUploader = null;
private YmageAnnotator m_AppAnnotator = null;
private YmageViewer m_AppViewer = null;
public YmageViewerGUI()
{
    super();
}
/**
 * This method initializes this
 * @return void
 */
public void init()
{
    this.setSize(640, 480);
    this.setJMenuBar(getMbMainMenu());
    this.setContentPane(getPnAppletRoot());
this.addHierarchyBoundsListener(new java.awt.event.HierarchyBoundsAdapter()
{
    public void ancestorResized(java.awt.event.HierarchyEvent e)
    {
        m_SpSize = new Dimension(getPlCenterScroller().getSize());
        HandleZoom(null);
    }
});

m_DisplayThread = new Thread()
{
    public void run()
    {
        m_DisplayThread.setPriority(Thread.MIN_PRIORITY);
        while (m_DisplayThread == Thread.currentThread() |
            m_KillDisplayThread)
        {
            try { ShowVisible(); Thread.yield(); } 
            catch (Exception ex) { } 
        }
    }
};
System.out.println("Terminated display thread properly");

/**
 */

public void start()
{
    m_AppViewer = YmageViewer.getViewer();
    m_AppUploader = YmageUploader.getUploader();
    m_AppAnnotator = YmageAnnotator.getAnnotator();
    m_SpSize = new Dimension(getPlCenterScroller().getSize());
    getTfServerURL().setText(m_AppViewer.getServerURL().toString());
    m_AppAnnotator.setServerURL(m_AppViewer.getServerURL());
    m_DisplayThread.start();
}

public void stop()
{
    //stop the display thread
    m_KillDisplayThread = true;
    //try to close all open connections
    System.out.println("Trying to close stream for "+getPlImage().
                    getComponentCount()+" panels.");
    try
    {
        int count = 0;
        JPanel panel = getPlImage();
        for (int i=0;i<panel.getComponentCount();i++)
        {
            try
            {
                if (((ImgPanel)panel.getComponent(i)).
                    closeConnections())
                {
                    System.out.println(++count+"thread(s) closed");
                }
            }
            catch (Exception ex) { } 
        }
    }
    catch (Exception ex) { } 
    System.out.println("Properly stopped applet thread");

    /**
private JPanel getPnAppletRoot()
{
    if (pnAppletRoot == null)
    {
        pnAppletRoot = new JPanel();
        pnAppletRoot.setLayout(new CardLayout());
        pnAppletRoot.add(getPlViewer(), getPlViewer().getName());
        pnAppletRoot.add(getPlSettings(), getPlSettings().getName());
        pnAppletRoot.add(getPlUpload(), getPlUpload().getName());
    }
    return pnAppletRoot;
}

/**
 * This method initializes plViewer
 * @return javax.swing.JPanel
 */
private JPanel getPlViewer()
{
    if (plViewer == null)
    {
        plViewer = new JPanel();
        plViewer.setLayout(new BorderLayout());
        plViewer.setName("plViewer");
        plViewer.setBackground(Color.BLACK);
        plViewer.add(getPlViewerOverlay(), BorderLayout.CENTER);
        plViewer.add(getPlImageInfo(), BorderLayout.SOUTH);
        plViewer.setDoubleBuffered(true);
        plViewer.addMouseWheelListener(new MouseWheelListener()
        {
            public void mouseWheelMoved(MouseWheelEvent e)
            {
                HandleZoom(e);
            }
        });
    }
    return plViewer;
}

/**
 * This method initializes mbMainMenu
 * @return javax.swing.JMenuBar
 */
private JMenuBar getMbMainMenu()
{
    if (mbMainMenu == null)
    {
        mbMainMenu = new JMenuBar();
        mbMainMenu.setBackground(Color.LIGHT_GRAY);
        mbMainMenu.setBorder(BorderFactory.createMatteBorder(0, 0, 1, 0,
                Color.GRAY));
        mbMainMenu.add(getMuApplication());
        mbMainMenu.add(getMuImage());
        mbMainMenu.add(getMuAnnotation());
    }
    return mbMainMenu;
}

/**
 * This method initializes muApplication
 * @return javax.swing.JMenu
 */
/*
private JMenu getMuApplication()
{
    if (muApplication == null)
    {
        muApplication = new JMenu();
        muApplication.setText("Application");
        muApplication.setMnemonic(KeyEvent.VK_A);
        muApplication.add(getMiAppSettings());
        muApplication.add(getMiRetrieveYmages());
    }
    return muApplication;
}
/**
 * This method initializes plSettings
 * @return javax.swing.JPanel
 */
private JPanel getPlSettings()
{
    if (plSettings == null)
    {
        FlowLayout flowLayout = new FlowLayout();
        flowLayout.setAlignment(FlowLayout.LEFT);
        plSettings = new JPanel();
        plSettings.setLayout(flowLayout);
        plSettings.setName("plSettings");
        plSettings.add(getPlServerSettings(), null);
        plSettings.add(getPlConfirmSettings(), null);
    }
    return plSettings;
}
/**
 * This method initializes miAppSettings
 * @return javax.swing.JMenuItem
 */
private JMenuItem getMiAppSettings()
{
    if (miAppSettings == null)
    {
        miAppSettings = new JMenuItem();
        miAppSettings.setText("Settings");
        miAppSettings.setMnemonic(KeyEvent.VK_S);
        miAppSettings.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                ((CardLayout)getPnAppletRoot().getLayout()).
                show(getPnAppletRoot(), (String)getPlSettings()
                .getName());
            }
        });
    }
    return miAppSettings;
}
/**
 * This method initializes miRetrieveYmages
 * @return javax.swing.JMenuItem
 */
private JMenuItem getMiRetrieveYmages()
{
    if (miRetrieveYmages == null)
    {
        miRetrieveYmages = new JMenuItem();
        miRetrieveYmages.setText("Retrieve Ymage List");
        miRetrieveYmages.setMnemonic(KeyEvent.VK_R);
    }
miRetrieveYmages.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent e)
    {
        try
        {
            //retrieve a list of available ymages
            String[] ymages = m_AppViewer.getYmageList();

            //remove the previous list of ymages
            int compcount = getMuYmage().getMenuComponentCount();
            for(int i=0;i<compcount-2;i++)
            {
                getMuYmage().remove(getMuYmage().
                                    getMenuComponent(2));
            }

            //create an actionlistener
            ActionListener actionhandler = new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    YmageViewer ctl = m_AppViewer;
                    try
                    {
                        //free up some memory by removing previously
downloaded images
                        ctl.clearImages();

                        //get info about the new image
                        ctl.setCurrentYmageName(((JMenuItem)e.getSource()).
                                                getName());
                        ctl.loadYmage();

                        //update the image info
                        getTfYmageInfo().setText(" Name: "
                                                +ctl.getCurrentYmageName()
                                                +" Width: "+ctl.getYmageWidth()
                                                +" Height: "+ctl.getYmageHeight());
                        ctl.setCurrentLayer(ctl.
                                            getYmageLayerCount()-1);

                        //get list of annotation users for this
                        ymage
                        m_AppAnnotator.setCurrentYmageName
                        (m_AppViewer.getCurrentYmageName());
                        try
                        {
                            String[] userlist = m_AppAnnotator.
                                                getUserList();
                            int compcount = getMuAnnotation().
                                            getMenuComponentCount();

                            for(int i=0;i<compcount-2;i++)
                            {
                                getMuAnnotation().remove(
                                                  getMuAnnotation().
                                                  getMenuComponent(2));
                            }

                            //populate the Annotation menu with
                            the list of available ymages
                            for(int i=0;i<userlist.length;i++)
                            {
                                JMenuItem mi = new
                                JMenuItem(userlist[i]);
                                mi.setName(userlist[i]);
                            }
                        }
                    }
                }
            }
        }
    }
});
//bind the handler to each ymage's menu item
getMuAnnotation().add(mi);
} }catch(Exception ex) { }

//handle the UI stuff
getSpViewer().setVisible(true);
DrawGrid();

URL url = new URL(ctl.getServerURL().toString() + "/" + 
ctl.getCurrentYmageName() + ".ymg?cmd=DOWNLOAD&layer=" + 
(ctl.getYmageLayerCount() - 1) + "&row=0&column=0";

Image image = 
ImageIO.read(url.openStream());

int w = image.getWidth(null);
int h = image.getHeight(null);

//int borderthickness = 4;
float scalew = (float)getPlMiniImage().getWidth() / (float)w;
float scaleh = (float)getPlMiniImage().getHeight() / (float)h;

if((scalew * w) < getPlMiniImage().getWidth() && (scalew * h) < 
getPlMiniImage().getHeight())
{ w *= scalew; h *= scalew; }
else { w *= scaleh; h *= scaleh; }

getPlMiniImage().removeAll();
ImgPanel imgpl = new ImgPanel();
imgpl.setImage(image.getScaledInstance 
(w-4, h-4, Image.SCALE_SMOOTH));
imgpl.setSize(new Dimension(w-4,h-4));
imgpl.setPreferredSize(new Dimension
(w-4,h-4));

getPlNoJitterMiniImage().setVisible(true);
getPlMiniImage().add(imgpl);
getPlMiniImage().updateUI();

{(CardLayout)pnAppletRoot().getLayout() .show(pnAppletRoot(),(String)
getPlViewer().getName());}
catch(Exception ex)
{
    JOptionPane.showMessageDialog
    (pnAppletRoot(), ex.getMessage(),
    "Error: Get Ymage Info", 
    JOptionPane.ERROR_MESSAGE);
    }
}

//populate the Ymage menu with the list of available ymages
for(int i=0;i<ymages.length;i++)
{ JMenu mi = new JMenuItem(ymages[i]);
    mi.setName(ymages[i]);
// bind the handler to each ymage's menu item
mi.addActionListener(actionhandler);
muYmage().add(mi);

catch(Exception ex)
{
    JOptionPane.showMessageDialog(getPnAppletRoot(),
        ex.getMessage(),"Error: Get Ymage List",
        JOptionPane.ERROR_MESSAGE);
}
.(*CardLayout)getPnAppletRoot().getLayout()).
    show(getPnAppletRoot(),(String)getPlViewer().getName());
}

return miRetrieveYmages;

/**
 * This method initializes muYmage
 * @return javax.swing.JMenu
 */
private JMenu getMuYmage()
{
    if (muYmage == null)
    {
        muYmage = new JMenu();
        muYmage.setText("Ymage");
        muYmage.add(getMiUpload());
        muYmage.add(getSepYmageMenu());
        muYmage.setMnemonic(KeyEvent.VK_Y);
    }

    return muYmage;
}

/**
 * This method initializes miUpload
 * @return javax.swing.JMenuItem
 */
private JMenuItem getMiUpload()
{
    if (miUpload == null)
    {
        miUpload = new JMenuItem();
        miUpload.setMnemonic(KeyEvent.VK_U);
        miUpload.setText("Upload");
        miUpload.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                ((CardLayout)getPnAppletRoot().getLayout()).
                    show(getPnAppletRoot(),(String)getPlUpload().
                        getName());
            }
        });

        return miUpload;
    }

    return miRetrieveYmages;
}

/**
 * This method initializes sepYmageMenu
 * @return javax.swing.JSeparator
 */
private JSeparator getSepYmageMenu()
{
    if (sepYmageMenu == null)
    {

sepYmageMenu = new JSeparator();

return sepYmageMenu;

/**
 * This method initializes plUpload
 * @return javax.swing.JPanel
 */
private JPanel getPlUpload()
{
if (plUpload == null)
{
plUpload = new JPanel();
plUpload.setLayout(new BorderLayout());
plUpload.setName("plUpload");
plUpload.setBackground(new Color(255, 204, 153));
plUpload.add(getPlUploadButtons(), BorderLayout.NORTH);
plUpload.add(getPlUploadImage(), BorderLayout.CENTER);
plUpload.add(getPlRegionSelectTools(), BorderLayout.SOUTH);
}
return plUpload;
}

/**
 * This method initializes plServerSettings
 * @return javax.swing.JPanel
 */
private JPanel getPlServerSettings()
{
if (plServerSettings == null)
{
GridBagConstraints gridBagConstraints = new GridBagConstraints();
gridBagConstraints.fill = GridBagConstraints.VERTICAL;
gridBagConstraints.weightx = 1.0;
lblURL = new JLabel();
lblURL.setText("URL");
plServerSettings = new JPanel();
plServerSettings.setLayout(new GridBagLayout());
plServerSettings.setBorder(BorderFactory.createTitledBorder(null,
"Server", TitledBorder.DEFAULT_JUSTIFICATION,
TitledBorder.DEFAULT_POSITION, new Font("Dialog", Font.BOLD, 12), new Color(51, 51, 51)));
plServerSettings.setPreferredSize(new Dimension(180, 60));
plServerSettings.add(lblURL, new GridBagConstraints());
plServerSettings.add(getTfServerURL(), gridBagConstraints);
}
return plServerSettings;
}

/**
 * This method initializes tfServerURL
 * @return javax.swing.JTextField
 */
private JTextField getTfServerURL()
{
if (tfServerURL == null)
{
    tfServerURL = new JTextField();
    tfServerURL.setPreferredSize(new Dimension(140, 20));
}
return tfServerURL;
}

/**
 * This method initializes plConfirmSettings
 * @return javax.swing.JPanel
 */
private JPanel getPlConfirmSettings()
{
private JPanel getPlConfirmSettings()
{
    if (plConfirmSettings == null)
    {
        plConfirmSettings = new JPanel();
        plConfirmSettings.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(0, 4, 0, 0);
        plConfirmSettings.add(getBtnOK(), gbc);
    }
    return plConfirmSettings;
}

/**
 * This method initializes btnOK
 * @return javax.swing.JButton
 */
private JButton getBtnOK()
{
    if (btnOK == null)
    {
        btnOK = new JButton();
        btnOK.setText("Confirm");
        btnOK.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btnOK.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                try
                {
                    m_AppViewer.setServerURL(new URL(getTfServerURL().getText()));
                }
                catch (Exception ex)
                {
                    JOptionPane.showMessageDialog(getPnAppletRoot(), ex.getMessage(),
                    "URL Error", JOptionPane.ERROR_MESSAGE);
                }
                ((CardLayout)getPnAppletRoot().getLayout()).show(getPnAppletRoot(),
                (String)getPlViewer().getName());
            }
        });
    }
    return btnOK;
}

/**
 * This method initializes btnCancel
 * @return javax.swing.JButton
 */
private JButton getBtnCancel()
{
    if (btnCancel == null)
    {
        btnCancel = new JButton();
        btnCancel.setText("Cancel");
        btnCancel.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btnCancel.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
            }
        });
    }
    return btnCancel;
}
try {
    getTfServerURL().setText(m_AppViewer.getServerURL().toString());
} catch (Exception ex) { }

{(CardLayout) getPnAppletRoot().getLayout()).show(getPnAppletRoot(), (String) getPlViewer().getName());
}

return btnCancel;

/**
* This method initializes plYmageInfo
* @return javax.swing.JPanel
*/
private JPanel getPlYmageInfo() {
    if (plYmageInfo == null) {
        plYmageInfo = new JPanel();
        plYmageInfo.setLayout(new BoxLayout(getPlYmageInfo(), BoxLayout.X_AXIS));
        plYmageInfo.setBorder(BorderFactory.createMatteBorder(1, 0, 1, 0, Color.GRAY));
        plYmageInfo.setPreferredSize(new Dimension(600, 20));
        plYmageInfo.setSize(new Dimension(600, 20));
        plYmageInfo.add(getTfYmageInfo(), null);
    }
    return plYmageInfo;
}

/**
* This method initializes spViewer
* @return javax.swing.JScrollPane
*/
private JScrollPane getSpViewer() {
    if (spViewer == null) {
        spViewer = new JScrollPane();
        spViewer.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
        spViewer.setViewportView(getPlImageOverlay());
        spViewer.addMouseWheelListener(new MouseWheelListener() {
            public void mouseWheelMoved(MouseWheelEvent e) {
                HandleZoom(e);
            }
        });
        spViewer.getHorizontalScrollBar().addAdjustmentListener(new AdjustmentListener() {
            public void adjustmentValueChanged(AdjustmentEvent e) {
                HandleViewArea(e);
            }
        });
    }
    return spViewer;
}
spViewer.getVerticalScrollBar().addAdjustmentListener
{
    new AdjustmentListener()
    {
        public void adjustmentValueChanged(AdjustmentEvent e)
        {
            HandleViewArea(e);
        }
    }
};
spViewer.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
spViewer.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER);
spViewer.setVisible(false);
}
return spViewer;
}/**
 * This method initializes tfYmageInfo
 * @return javax.swing.JTextField
 */
private JTextField getTfYmageInfo()
{
    if (tfYmageInfo == null)
    {
        tfYmageInfo = new JTextField();
        tfYmageInfo.setFont(new Font("Dialog", Font.PLAIN, 11));
        tfYmageInfo.setEditable(false);
        tfYmageInfo.setBackground(Color.lightGray);
        tfYmageInfo.setBorder(BorderFactory.createEmptyBorder());
    }
    return tfYmageInfo;
}
/**
 * This method loads the visible image blocks into the scrollable panel
 */
private synchronized void ShowVisible()
{
    YmageViewer ctl = m_AppViewer;
    int w = ctl.getCurrentYmageBlock().getWidth();
    int h = ctl.getCurrentYmageBlock().getHeight();
    Rectangle vis = getPlImage().getVisibleRect();

    int lc = vis.x / w;
    int tr = vis.y / h;
    int rc = (vis.x + vis.width) / w;
    int br = (vis.y + vis.height) / h;

    for(int r=0;r<=br-tr;r++)
    {
        for(int c=0;c<=rc-lc;c++)
        {
            try
            {
                int tmpr = r + tr; int tmpc = c + lc;
                ImgPanel imgpl = (ImgPanel)ctl.getImage(ctl.
                getCurrentYmageName() + "_" + ctl.getCurrentLayer() +
                "x" + tmpr + "x" + tmpc);
                if(imgpl.getImage() == null)
                {
                    imgpl.setImage(new URL(ctl.getServerURL().toString() +"/" +
                    ctl.getCurrentYmageName() + ".ymg?cmd=DOWNLOAD&
layer="+ctl.getCurrentLayer()+"&row="+tmpr+
"&column="+tmpc));
    HandleZoom(null);
}
} catch(NullPointerException ex) { }
} catch(IIOException ex) { }
} catch(Exception ex)
{
    JOptionPane.showMessageDialog(getPnAppletRoot(),
    ex.getMessage(),"ERROR: "+ex.getClass(),
    JOptionPane.ERROR_MESSAGE);
}
}
getPlMiniImage().repaint();
/**
 * This method sets up the scroll viewport and renders the visible image blocks
 */
public void DrawGrid()
{
    YmageViewer ctl = m_AppViewer;
    try
    {
        //try to close all open connections
        System.out.println("Trying to close stream for "+getPlImage().
        getComponentCount()="+panels.");
        try
        {
            int count = 0;
            JPanel panel = getPlImage();
            for(int i=0;i<panel.getComponentCount();i++)
            {
                try
                {
                    if(((ImgPanel)panel.getComponent(i)).
                    closeConnections())
                    {
                        System.out.println(++count+"thread(s)
                        closed");
                    }
                }
                catch(Exception ex) { }
            }
        }
        catch(Exception ex) { }
        int nr = ctl.getCurrentYmageBlock().getNumRows();
        int nc = ctl.getCurrentYmageBlock().getNumColumns();
        int w = ctl.getCurrentYmageBlock().getWidth();
        int h = ctl.getCurrentYmageBlock().getHeight();
        int r, c;
        getPlImage().removeAll();
        getPlImage().setPreferredSize(new Dimension(w*nc,h*nr));
        getPlImage().setSize(new Dimension(w*nc,h*nr));
        ((GridLayout)getPlImage().getLayout()).setRows(nr);
        ((GridLayout)getPlImage().getLayout()).setColumns(nc);
        ((GridLayout)getPlImage().getLayout()).setHgap(0);
        ((GridLayout)getPlImage().getLayout()).setVgap(0);
        String ymagename = ctl.getCurrentYmageName();
        int layer = ctl.getCurrentLayer();
        for(r=0;r<nr;r++)
        {
for(c=0;c<nc;c++)
{
    ImgPanel image = null;
    if((image = (ImgPanel)ctl.getImage(ymagename+'_layer'+x+r+'x'+c)) == null)
    {
        image = new ImgPanel();
        image.setPreferredSize(new Dimension(w,h));
        image.setSize(new Dimension(w,h));
        image.setMinimumSize(new Dimension(w,h));
        image.setPreferredSize(new Dimension(w,h));
        ctl.addImage(ymagename+'_layer'+x+r+'x'+c,(ImgPanel)image);
    }
    getPlImage().add(image);
}
HandleZoom(null);
getPlImage().validate();
}
catch(Exception ex)
{
    //JOptionPane.showMessageDialog(null,ex);
}
/**
 * This method Draws the rectangle in representing the view area
 * *
 */
private void HandleViewArea(AdjustmentEvent e)
{
    try
    {
        JScrollBar hsb = getSpViewer().getHorizontalScrollBar();
        JScrollBar vsb = getSpViewer().getVerticalScrollBar();
        ImgPanel imgpl = ((ImgPanel)getPlMiniImage().getComponent(0));
        float vvis = (float)hsb.getVisibleAmount() /
                    (float)getPlImage().getWidth();
        float hvis = (float)vsb.getVisibleAmount() /
                    (float)getPlImage().getHeight();
        int x = Math.round(hsb.getValue() * imgpl.getWidth() /
                        (float)hsb.getMaximum());
        int y = Math.round(vsb.getValue() * imgpl.getHeight() /
                          (float)vsb.getMaximum());
        int w = Math.round(imgpl.getWidth()*vvis);
        int h = Math.round(imgpl.getHeight()*hvis);
        imgpl.setRect(new Rectangle(x,y,w-1,h-1));
    }
    catch(Exception ex) { }
}
/**
 * This method handles zooming as a mouse scroll wheel listener
 * *
 */
private void HandleZoom(MouseWheelEvent e)
{
    YmageViewer ctl = m_AppViewer;
    try
    {
        int w = ctl.getCurrentYmageBlock().getWidth();
        int h = ctl.getCurrentYmageBlock().getHeight();
        if(e != null)
        {
            if(e.getWheelRotation() < 0)
            {
                if(ctl.getZoom() == 9 && ctl.getCurrentLayer() > 0)
{ ctl.setCurrentLayer(ctl.getCurrentLayer()-1);
    ctl.setZoom(0);
    DrawGrid();
    return;
}
else
{ ctl.setZoom(ctl.getZoom()+1); }
else
{
    if(ctl.getZoom() == 0 && ctl.getCurrentLayer() <
   ctl.getYmageLayerCount() - 1)
    {
        ctl.setCurrentLayer(ctl.getCurrentLayer()+1);
        ctl.setZoom(10);
        DrawGrid();
        return;
    }
    else if(ctl.getZoom() > 0)
    {
        ctl.setZoom(ctl.getZoom()-1);
    }
}
}
float scale = 1.0f + (0.1f*ctl.getZoom());
w *= scale; h *= scale;
((GridLayout)getPlImage().getLayout()).setHgap(0);
((GridLayout)getPlImage().getLayout()).setVgap(0);
getPlImage().setPreferredSize(new Dimension(w*ctl.getCurrentYmageBlock().getNumColumns(),
h*ctl.getCurrentYmageBlock().getNumRows()));
getPlImage().setSize(w*ctl.getCurrentYmageBlock().
.getNumColumns(),h*ctl.getCurrentYmageBlock().
.getNumRows());
getPlImageOverlay().setPreferredSize(getPlImage().
.getPreferredSize());
getPlImageOverlay().setSize(getPlImage().getSize());
Dimension dp = getPlImage().getSize();
Dimension dn = new Dimension(m_SpSize);
if(dp.width < dn.width)
{ dn.width = dp.width; }
if(dp.height < dn.height)
{ dn.height = dp.height; }
getSpViewer().setPreferredSize(dn);
getSpViewer().setSize(dn);
getPlCenterScroller().updateUI();
}
catch(Exception ex) { }
}
flowLayout.setAlignment(FlowLayout.LEFT);
flowLayout.setVgap(3);

lblFilePath = new JLabel();

lblFilePath.setText(" Image File: ");
plUploadButtons = new JPanel();
plUploadButtons.setBorder(BorderFactory.createLineBorder(Color.lightGray, 1));
plUploadButtons.setLayout(flowLayout);
plUploadButtons.setPreferredSize(new Dimension(0, 30));

plUploadButtons.add(lblFilePath, null);
plUploadButtons.add(getTfUploadImagePath(), null);
plUploadButtons.add(getBtnUploadBrowse(), null);
plUploadButtons.add(getTfYmgName(), null);
plUploadButtons.add(getBtnUpload(), null);

} return plUploadButtons;

/**
 * This method initializes plUploadImage
 * @return javax.swing.JPanel
 */
private JPanel getPlUploadImage()
{
    if (plUploadImage == null)
    {
        plUploadImage = new JPanel();
        plUploadImage.setLayout(new GridBagLayout());
    }
    return plUploadImage;
}

/**
 * This method initializes tfUploadImagePath
 * @return javax.swing.JTextField
 */
private JTextField getTfUploadImagePath()
{
    if (tfUploadImagePath == null)
    {
        tfUploadImagePath = new JTextField();
        tfUploadImagePath.setPreferredSize(new Dimension(220, 22));
        tfUploadImagePath.setEditable(false);
    }
    return tfUploadImagePath;
}

/**
 * This method initializes btnUploadBrowse
 * @return javax.swing.JButton
 */
private JButton getBtnUploadBrowse()
{
    if (btnUploadBrowse == null)
    {
        btnUploadBrowse = new JButton();
        btnUploadBrowse.setText("Browse...");
        btnUploadBrowse.setPreferredSize(new Dimension(87, 20));

        btnUploadBrowse.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btnUploadBrowse.addActionListener(new java.awt.event.ActionListener()
                {
                    public void actionPerformed(java.awt.event.ActionEvent e)
                    {
                        if (getFcUploadFile().showOpenDialog(getPlUpload()) == JOptionPane.OK_OPTION)
                        {
                        }
                    }
                });
    }
    return btnUploadBrowse;
}
File nf = getFcUploadFile().getSelectedFile();
getTfUploadImagePath().setText(nf.getPath());
m_AppUploader.setImage(nf.getPath());
getTfYmgName().setText(m_AppUploader.getImageFileName());
getBtnRegion().setText("View Full Image");
getFlUploadImage().removeAll();
}
}
return btnUploadBrowse;
/**
* This method initializes fcUploadFile
* @return javax.swing.JFileChooser
*/
private JFileChooser getFcUploadFile()
{
    if (fcUploadFile == null)
    {
        fcUploadFile = new JFileChooser();
        fcUploadFile.setFileHidingEnabled(true);
        fcUploadFile.setDialogTitle("Select Image for Upload");
        fcUploadFile.addChoosableFileFilter
        {
            new FileFilter()
            {
                public String getDescription()
                {
                    return "Images Files";
                }
                public boolean accept(File f)
                {
                    if (f.isDirectory()) { return true; }
                    String extension = Utils.getExtension(f);
                    if (extension != null)
                    {
                        if (extension.equals(Utils.tiff) ||
                            extension.equals(Utils.tif) ||
                            extension.equals(Utils.gif) ||
                            extension.equals(Utils.jpeg) ||
                            extension.equals(Utils.jpg) ||
                            extension.equals(Utils.png))
                            return true;
                    }
                    return false;
                }
            },
        }
        return fcUploadFile;
    }

    public static class Utils
    {
        public final static String jpeg = "jpeg";
        public final static String jpg = "jpg";
        public final static String gif = "gif";
        public final static String tiff = "tiff";
        public final static String tif = "tif";
        public final static String png = "png";
        public static String getExtension(File f)
```java
String ext = null;
String s = f.getName();
int i = s.lastIndexOf('.');
if (i > 0 && i < s.length() - 1)
{
    ext = s.substring(i+1).toLowerCase();
}
return ext;
}

public static String getExtension(String s)
{
    File f = new File(s);
    return getExtension(f);
}

/**
 * This method initializes btnRegion
 *
 * @return javax.swing.JButton
 */
private JButton getBtnRegion()
{
    if (btnRegion == null)
    {
        btnRegion = new JButton();
        btnRegion.setText("View Full Image");
        btnRegion.setPreferredSize(new Dimension(160, 20));
        btnRegion.setSize(new Dimension(160, 20));
        btnRegion.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btnRegion.setEnabled(true);
        btnRegion.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                if (btnRegion.getText().compareTo("View Selected Region") == 0)
                {
                    try
                    {
                        Rectangle region = RegionSelector.getSelection();
                        BufferedImage image = (BufferedImage)
                            m_AppUploader.getImage(new Dimension(450, 300), region);
                        JScrollPane tmpsp = new JScrollPane();
                        try
                        {
                            tmpsp.setPreferredSize(((ImgPanel)getPlUploadImage().getComponent(0)).getSize());
                            tmpsp.setSize(((ImgPanel)getPlUploadImage().getComponent(0)).getSize());
                        }
                        catch (Exception ex) { }
                    }
                    catch (Exception ex) { }
                }
            }
        });
    }
    return btnRegion;
}
```
getPLUploadImage().removeAll();
ImgPanel impl = new ImgPanel();
impl.setImage(image);
impl.setPreferredSize(region.getSize());
impl.setSize(region.getSize());
tmpsp.setViewportView(impl);
Dimension imgsize = (Dimension)region.getSize().clone();
imgsize.width = Math.min(tmpsp.getSize().width, imgsize.width+UIManager.getInt("ScrollBar.width")+2);
imgsize.height = Math.min(tmpsp.getSize().height, imgsize.height+UIManager.getInt("ScrollBar.height")+4);
tmpsp.setPreferredSize(imgsize);
tmpsp.setSize(imgsize);
GridBagConstraints gbc = new GridBagConstraints();
gbc.anchor = GridBagConstraints.CENTER;
gbc.weightx = 1;
gbc.weighty = 1;
getPLUploadImage().add(tmpsp);
getPLUploadImage().updateUI();
btnRegion.setText("View Full Image");
}
catch(Exception ex)
{
   JOptionPane.showMessageDialog(getPlUpload(), ex.getMessage(), "Error: "+ex.getClass(), JOptionPane.ERROR_MESSAGE);
}
else if(btnRegion.getText().compareTo("View Full Image") == 0)
{
   if(getTfUploadImagePath().getText().length() < 5)
   {
      JOptionPane.showMessageDialog(getPlUpload(), "Please select an image.", "No image selected", JOptionPane.ERROR_MESSAGE);
      return;
   }
   try
   {
      BufferedImage image = (BufferedImage)m_AppUploader.getImage(new Dimension(450,300),null);
      getPLUploadImage().removeAll();
      ImgPanel impl = new ImgPanel();
      impl.setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
      impl.setImage(image);
      int w = image.getWidth();
      int h = image.getHeight();
      Dimension size = new Dimension(w,h);
      getPLUploadImage().add(impl);
      impl.setPreferredSize(size);
      impl.setSize(size);
      impl.addMouseListener
(RegionSelector.getRegionSelector
(impl));
impl.addMouseMotionListener
{RegionSelector.getRegionSelector
(impl));
RegionSelector.clearSelection();
getPlUploadImage().updateUI();
}
catch(Exception ex)
{
	JOptionPane.showMessageDialog
{getPlUpload(), ex.getMessage(),
"Error:"+ex.getClass(),
JOptionPane.ERROR_MESSAGE);
}
btnRegion.setText("View Selected Region");
});
return btnRegion;
}

public static class RegionSelector implements MouseListener, MouseMotionListener
{

protected static RegionSelector m_RegionSelector = null;
protected static ImgPanel m_Panel = null;
protected static Point m_Pressed = null;
protected static Point m_Released = null;

private RegionSelector() { }

public static RegionSelector getRegionSelector(ImgPanel panel)
{
	nif(m_RegionSelector == null)
	{ m_RegionSelector = new RegionSelector(); }

	m_Panel = panel;
	nreturn m_RegionSelector;
}

public static Rectangle getSelection()
{
	nif(m_Pressed == null || m_Released == null) { return null; }


Rectangle rect = new Rectangle();
rect.x = Math.max(Math.min(m_Pressed.x, m_Released.x),0);
rect.y = Math.max(Math.min(m_Pressed.y, m_Released.y),0);
rect.width = Math.max(m_Pressed.x, m_Released.x) - rect.x;
rect.height = Math.max(m_Pressed.y, m_Released.y) - rect.y;

return rect;
}

public static void clearSelection()
{
	m_Pressed = m_Released = null;
}

public void mouseEntered(MouseEvent e) { }
public void mouseExited(MouseEvent e) { }
public void mouseClicked(MouseEvent e) { }
public void mouseMoved(MouseEvent e) { }
public void mousePressed(MouseEvent e)
{
	nif(m_Pressed == null)
	{ m_Pressed = new Point(); }
	nif(m_Released == null)
	{ m_Released = new Point(); }
```java
m_Pressed.setLocation(e.getX(), e.getY());
m_Released.setLocation(e.getX(), e.getY());

public void mouseReleased(MouseEvent e)
{
    m_Released.setLocation(e.getX(), e.getY());
}

public void mouseDragged(MouseEvent e)
{
    Graphics2D g = (Graphics2D) m_Panel.getGraphics();
    if (g == null) return;
    int x1 = m_Pressed.x;
    int y1 = m_Pressed.y;
    int x2 = e.getX();
    int y2 = e.getY();
    m_Panel.update(g);
    g.setXORMode(Color.ORANGE);
    g.setStroke(new BasicStroke(2));
    g.drawLine(x1, y1, x1, y2);
    g.drawLine(x1, y2, x2, y2);
    g.drawLine(x2, y2, x2, y1);
    g.drawLine(x2, y1, x1, y1);
}

/**
 * This method initializes plRegionSelectTools
 * @return javax.swing.JPanel
 */
private JPanel getPlRegionSelectTools()
{
    if (plRegionSelectTools == null)
    {
        FlowLayout flowLayout = new FlowLayout();
        flowLayout.setAlignment(FlowLayout.RIGHT);
        flowLayout.setHgap(5);
        flowLayout.setVgap(1);
        plRegionSelectTools = new JPanel();
        plRegionSelectTools.setBorder(BorderFactory.createLineBorder(Color.lightGray, 1));
        plRegionSelectTools.setLayout(flowLayout);
        plRegionSelectTools.setPreferredSize(new Dimension(114, 24));
        plRegionSelectTools.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        plRegionSelectTools.add(getPbUploadProgress(), null);
        plRegionSelectTools.add(getBtnRegion(), null);
    }
    return plRegionSelectTools;
}

/**
 * This method initializes btnUpload
 * @return java.awt.Button
 */
private JButton getBtnUpload()
{
    if (btnUpload == null)
    {
        btnUpload = new JButton();
        btnUpload.setText("Upload");
        btnUpload.setHorizontalAlignment(SwingConstants.CENTER);
        btnUpload.setPreferredSize(new Dimension(73, 20));
        btnUpload.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btnUpload.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                // Code for action performed
            }
        });
    }
    return btnUpload;
}
```
Thread progressthread = new Thread()
{
    public void run()
    {
        m_AppUploader.setProgress(0);
        int progress = 0;
        JProgressBar pb = getPbUpload
        Progress();
        pb.setIndeterminate(false);
        while((progress = m_AppUploader.
            getProgress()) < 99)
        {
            pb.setValue(progress);
            pb.setString(progress + "%");
            Thread.yield();
        }
        pb.setIndeterminate(true);
        pb.setString("Waiting for
response...");
        while(m_AppUploader.getProgress() !=
100)
        {
            pb.setString("Complete");
            pb.setValue(100);
            pb.setIndeterminate(false);
        }
    }
};

Thread uploadthread = new Thread()
{
    public void run()
    {
        JEditorPane htmlpane = new
        JEditorPane();
        htmlpane.setEditable(false);
        htmlpane.setContentType("text/html");
        htmlpane.setBackground(new
        Color(238,238,238));
        String ymgname = getTfYmgName().
        getText();
        StringBuffer upstr = new
        StringBuffer
        (m_AppViewer.getServerURL().
        toString());
        upstr.append(ymgname + ".ymg");
        upstr.append("?cmd=UPLOAD");
        String response = "";
        try
        {
            if(getBtnRegion().getText().compareTo
("View Selected Region") == 0)
            { RegionSelector. clearSelection(); }
            getBtnRegion().setVisible(false);
            getBtnUploadProgress().setVisible
            (true);
            getBtnUpload().setEnabled(false);
            getBtnUploadBrowse().setEnabled
            (false);
            getTfYmgName().setEditable(false);
            response = m_AppUploader.uploadImage
(new URL(upstr.toString()),
RegionSelector.getSelection());
            htmlpane.setText(response);
        } catch
        {
            JOptionPane.showMessageDialog
            (getPlUpload(), htmlpane,
"Upload request: " + ymgname,
JOptionPane.
INFORMATION_MESSAGE);
        }
    }
};
```java
try {
    uploadthread.start();
    progressthread.start();
} catch (Exception ex) {
    m_AppUploader.setProgress(100);
    JOptionPane.showMessageDialog(getPlUpload(), ex.getMessage(),
            "Error: "+ex.getClass(), JOptionPane.ERROR_MESSAGE);
} finally {
    getBtnRegion().setVisible(true);
    getPbUploadProgress().setVisible(false);
    getBtnUpload().setEnabled(true);
    getBtnUploadBrowse().setEnabled(true);
    getTfYmgName().setEditable(true);
}
```
private JPanel getPlImage() {
    if (plImage == null) {
        plImage = new JPanel();
        plImage.setLayout(new GridLayout());
    }
    return plImage;
}

/**
 * This method initializes plImageOverlay
 *
 * @return javax.swing.JPanel
 */
private JPanel getPlImageOverlay() {
    if (plImageOverlay == null) {
        plImageOverlay = new JPanel();
        plImageOverlay.setLayout(new OverlayLayout(plImageOverlay));
        plImageOverlay.add(getPlOverlay());
        plImageOverlay.add(getPlImage());
        plImageOverlay.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent e) {
                m_StartPan = new Point(e.getX(), e.getY());
            }
            public void mouseReleased(java.awt.event.MouseEvent e) {
                m_StartPan = null;
            }
        });
        plImageOverlay.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
            public void mouseDragged(java.awt.event.MouseEvent e) {
                if (m_StartPan == null) { return; }
                JScrollBar hsb = getSpViewer().getHorizontalScrollBar();
                JScrollBar vsb = getSpViewer().getVerticalScrollBar();
                hsb.setValue(hsb.getValue() + (int)(m_StartPan.x - e.getX()));
                vsb.setValue(vsb.getValue() + (int)(m_StartPan.y - e.getY()));
            }
        });
    }
    return plImageOverlay;
}

/**
 * This method initializes plOverlay
 *
 * @return javax.swing.JPanel
 */
private JPanel getPlOverlay() {
    if (plOverlay == null) {
        plOverlay = new JPanel() {
            public void paintComponent(Graphics g) {
                /*g.setXORMode(Color.white);
                ImageViewer ctl = m_AppViewer;
                double zf = (1.0f+(ctl.getZoom()*0.1f));
                int ly = Math.max((int)Math.pow(2, ctl.getYmageLayerCount() - ctl.getCurrentLayer)}*/
        }
    }
    public void paintComponent(Graphics g) {
        /*g.setXORMode(Color.white);
        ImageViewer ctl = m_AppViewer;
        double zf = (1.0f+(ctl.getZoom()*0.1f));
        int ly = Math.max((int)Math.pow(2, ctl.getYmageLayerCount()) - ctl.getCurrentLayer

    public void mouseDragged(java.awt.event.MouseEvent e) {
        if (m_StartPan == null) { return; }
        JScrollBar hsb = getSpViewer().getHorizontalScrollBar();
        JScrollBar vsb = getSpViewer().getVerticalScrollBar();
        hsb.setValue(hsb.getValue() + (int)(m_StartPan.x - e.getX()));
        vsb.setValue(vsb.getValue() + (int)(m_StartPan.y - e.getY()));
    }
    };
    return plImageOverlay;
}
g.fillOval((int)(100*zf*ly),(int)(100*zf*ly),10,10);
g.drawString("Annotation",(int)(100*zf*ly),(int)(100*zf*ly));*

plOverlay.setOpaque(false);
plOverlay.setLayout(null);
return plOverlay;

/**
 * This method initializes plViewerOverlay
 * @return javax.swing.JPanel
 */
private JPanel getPlViewerOverlay()
{
if (plViewerOverlay == null)
{
    plViewerOverlay = new JPanel();
    plViewerOverlay.setLayout(new OverlayLayout(plViewerOverlay));
    plViewerOverlay.add(getPlViewOver());
    plViewerOverlay.add(getPlCenterScroller());
}
return plViewerOverlay;

/**
 * This method initializes plViewOver
 * @return javax.swing.JPanel
 */
private JPanel getPlViewOver()
{
if (plViewOver == null)
{
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.fill = GridBagConstraints.NONE;
    gbc.weightx = 1; gbc.weighty = 1;
    gbc.gridheight = 1; gbc.gridwidth = 1;
    gbc.insets = new Insets(0, 0, 10, 10);
    gbc.anchor = GridBagConstraints.SOUTHEAST;
    plViewOver = new JPanel();
    plViewOver.setLayout(new GridBagLayout());
    plViewOver.setOpaque(false);
    plViewOver.add(getPlNoJitterMiniImage(), gbc);
}
return plViewOver;

/**
 * This method initializes plNoJitterMiniImage
 * @return javax.swing.JPanel
 */
private Panel getPlNoJitterMiniImage()
{
if (plNoJitterMiniImage == null)
{
    plNoJitterMiniImage = new Panel();
    plNoJitterMiniImage.setLayout(new BorderLayout());
    plNoJitterMiniImage.setPreferredSize(new Dimension(180, 120));
    plNoJitterMiniImage.setSize(new Dimension(180, 120));
    plNoJitterMiniImage.setVisible(false);
    plNoJitterMiniImage.add(getPlMiniImage(), BorderLayout.CENTER);
private JPanel getPlMiniImage() {
    if (plMiniImage == null) {
        FlowLayout flow = new FlowLayout();
        flow.setHgap(0);
        flow.setVgap(0);
        plMiniImage = new JPanel();
        plMiniImage.setLayout(flow);
        plMiniImage.setPreferredSize(new Dimension(176, 116));
        plMiniImage.setBorder(BorderFactory.createLineBorder(Color.gray, 1));
        plMiniImage.setPreferredSize(new Dimension(176, 116));
        plMiniImage.setLayout(flow);
    }
    return plMiniImage;
}

/**
 * This method initializes plCenterScroller
 */
private JPanel getPlCenterScroller() {
    if (plCenterScroller == null) {
        GridBagConstraints gridBagConstraints1 = new GridBagConstraints();
        gridBagConstraints1.weighty = 1.0;
        gridBagConstraints1.gridx = 0;
        gridBagConstraints1.gridy = 0;
        gridBagConstraints1.fill = GridBagConstraints.NONE;
        gridBagConstraints1.anchor = GridBagConstraints.CENTER;
        gridBagConstraints1.weightx = 1.0D;
        plCenterScroller = new JPanel();
        plCenterScroller.setLayout(new GridBagLayout());
        plCenterScroller.add(getSpViewer(), gridBagConstraints1);
    }
    return plCenterScroller;
}

/**
 * This method initializes muAnnotation
 */
private JMenu getMuAnnotation() {
    if (muAnnotation == null) {
        muAnnotation = new JMenu();
        muAnnotation.setText("Annotation");
        muAnnotation.setMnemonic(KeyEvent.VK_A);
        muAnnotation.add(getMiShowAll());
        muAnnotation.add(getSepAnnotationMenu());
    }
    return muAnnotation;
}
private JMenuItem getMiShowAll()
{
    if (miShowAll == null)
    {
        miShowAll = new JMenuItem();
        miShowAll.setText("Show All");
    }
    return miShowAll;
}

/**
 * This method initializes sepAnnotationMenu
 * @return javax.swing.JSeparator
 */
private JSeparator getSepAnnotationMenu()
{
    if (sepAnnotationMenu == null)
    {
        sepAnnotationMenu = new JSeparator();
    }
    return sepAnnotationMenu;
}
APPENDIX E

Images Tested

1. P0-SC-QGFR146-Grina.tif

2. P0-SC-QGFR142-Grml.tif