Controlling Scientific Instruments Using JAVA on LINUX

Christine Beaty

Controlling Scientific Instruments Using JAVA on LINUX

C. Beaty
Advised by Dr. Christian Stoeckl

Laboratory for Laser Energetics
University of Rochester
250 East River Road
Rochester, NY 14623

Abstract

Motivated by the increased use of the Linux operating system at LLE, a program has been built in that environment to control the operation of CCD cameras. CCD cameras are used in several diagnostics, including UV spectrometers and streak cameras, that analyze the 60 of the OMEGA laser. This program, written in Java, allows users with little or no knowledge of the intricacies of CCD camera command logic to set up the camera and acquire data in the form of an image. Among the issues examined during programming lay the optimization of reading the image, leading to faster acquisition of the image, and the synchronization of threads in a multi-threaded program, which ensures that the camera control program will adequately respond to the user. The program was created using special care to facilitate easy reuse of the code for similar applications without requiring considerable modification.

Introduction

The Laboratory for Laser Energetics (LLE) at the University of Rochester conducts inertial confinement fusion (ICF) experiments with the sixty-beam OMEGA laser. Several of the diagnostics used in OMEGA both to measure the laser performance and in the target experiments are dependent upon the images acquired by CCD (Charged Couple Device) cameras. Therefore, controlling CCD cameras is vital to the success of the experiments at LLE.

Several versions of software for controlling CCD cameras have been developed by scientists at LLE in the past, all designed for use in the Microsoft Windows operating system. However, with an increasing interest in the Linux operating system at LLE, it was interesting to explore if software to operate CCD cameras could be created in the Linux environment. Since the Java programming language has several built-in features useful for software development, such as multi-threading, it was logical to write the software in that language.

Figure 1
CCD camera

Uses of CCD Cameras at LLE

CCD cameras are utilized in a variety of areas at LLE, including ultra-violet (UV) spectroscopy, X-ray diagnostics, and streak cameras. The images acquired by the cameras in use in the UV spectrometers capture the spectral lines of a laser beam. These spectral lines are then used to analyze the wavelength of the beam. In this manner, the different

wavelengths of the sixty beams in OMEGA can be analyzed.

The CCD cameras are also crucial when coupled with a streak tube and implemented as streak cameras in the generic streak camera platform. Light from a laser beam passes through the streak tube and is converted to a beam of electrons through the use of a photocathode. Any change in the intensity of the laser beam changes the intensity of the electron beam, so the electron beam is an accurate copy of the laser beam. This electron beam is then swept across a phosphor plate, causing it to glow. The changing intensities of the electron beam cause the glow to reflect the intensity variations of the laser beam. The CCD camera on the other side of this phosphor plate then photographs the glowing phosphor and, in effect, the intensity of light in the laser beam. Images from the camera can then be analyzed to measure the time history of the laser beam.

Program Function

The purpose of the CCDControl program is to provide an easy-to-use interface between the user and the CCD camera. The camera itself is connected to a device driver in the computer through a fiber-optic cable. The computer inside the camera recognizes certain letters as commands to perform a specified task. For example, sending the command "J" to the camera causes the camera to prepare itself to receive 32 configurations from the user's computer, and the command "L" causes the camera to send those configurations back to the user's computer. The CCDControl program automatically sends the specific commands to initialize the camera when it begins running, and other commands are sent during the course of the run as the user wishes.

The user, therefore, does not need to have any knowledge about which command to use to perform a task.

During the process of programming, several minimum values for the proper functioning of the camera were discovered. For example, the camera cannot safely process exposure times fewer than ten milliseconds (ms) for a timed image acquisition. Any exposure time less than ten ms may result in a failure to acquire the image, so an exposure time limit was encoded to ensure that the camera operates correctly at all times. In addition, monitoring the remaining exposure time, which is done by sending a command at regular intervals, cannot be done faster than every 50 ms and has to stop 200 ms before the end of the exposure. Smaller values may cause the camera to function incorrectly when acquiring the image.

User Interface

The GUI (Graphical User Interface) of the CCDControl program was based upon an earlier program written in Java that operates in Windows, with a few additions. It was meant to be as user-friendly as possible, with clear intentions of how it should be utilized. It provides important and

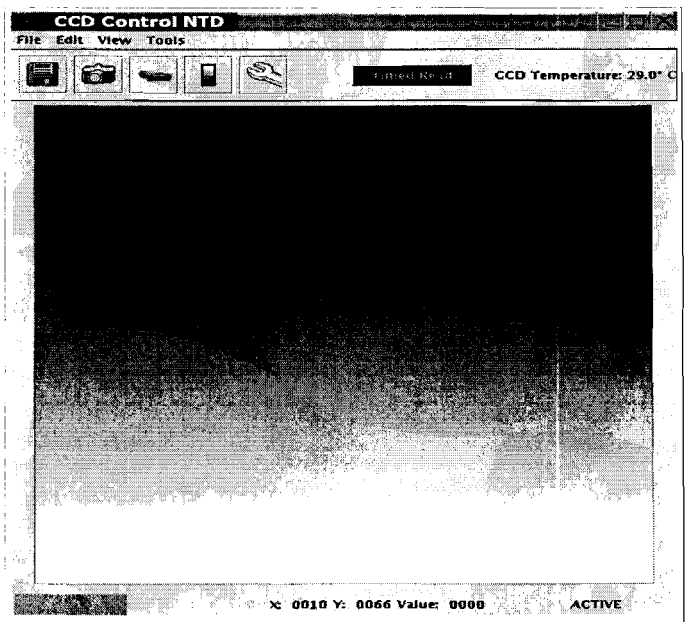

Figure 2
CCDControl GUI

real-time information regarding the status of the camera and each image acquisition, so the user knows exactly what the camera is doing. The temperature of the CCD is displayed in the GUI and automatically updated every second, and progress bars show the remaining exposure time and percent of the image read. The values of the 32 configurations and readout parameters are also displayed in a panel in the GUI to allow for easy user alteration. Any applied changes are then stored and used for subsequent initializations.

Program Structure

One advantage that the Java language brought to the program was the ability to create reusable code through the layering of the classes, allowing for easy future

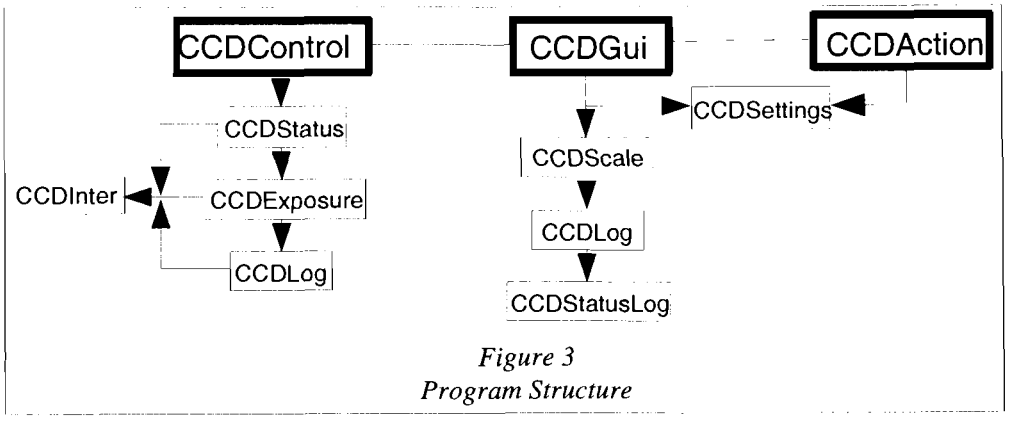

Figure 3
Program Structure

modification. Since very little of the program is hardcoded, the classes are more versatile. The layered classes can easily be reused for other, similar applications without requiring major alterations in the code. The organization of the individual classes increases code readability and makes program comprehension much easier. Understandable code is crucial for a program which may undergo future modifications, and the layering provides for easy alteration of one class without drastically affecting the others around it.

The majority of the programs within CCDControl were written in Java; however, the methods of the interface that controls the camera needed to be written using the Java Native Interface and C because Java cannot interact directly with devices. There are several layers of software, beginning with the device driver which interacts directly with the camera, followed by the interface in C and Java, and then the CCDControl program.

Optimization

One of the key issues in designing this program was the efficiency at which the image could be acquired and displayed. Since the image is acquired in a grid of 2200 by 2200 pixels, and the typical size of a computer monitor is 1280 by 1024 pixels, it was clear that the image had to be reduced in size. It was determined that the image should be reduced in size to 550 by 550 pixels, or one-sixteenth of the original size, for test readings. This was done by a process called undersampling, displaying only every fourth pixel in every fourth row.

Several algorithms were tested to find the most efficient method of condensing the image, as it is often the case with CCD cameras that the image needs to be displayed very soon after the reading commences. The method that is used in the program takes approximately six ms per image, a reasonably small amount of time. The original algorithm

Algorithm A

```
int l, j=0, i=0;
while (i<550*550) {
   l = i + 550;
   for (int k = i, k<l;k++) {
      image[k] = ImgArray[j];
      j = j + 4;
   }
   i = i + 550;
   j = j + 2200 * 3;
}
```

Algorithm B

```
int k = 0;
for (int i=0; i<2200*2200; i++) {
   if ((i/2200)%4 == 0 &&  i%4 ==  0) {
      image[k] = ImgArray[i];
     k++;
   }
}
```

Figure 4
Undersampling Algorithms

tested took about 166 ms per image, so the speed of the final algorithm was increased by a factor of 28. This remarkable increase is due to the number of operations performed. The first algorithm, algorithm "B" in the above table, has two divisions in each of 4840000 (2200 by 2200) iterations. Because division is a very time-consuming operation and there were so many divisions per run of the algorithm, the time required for the algorithm to complete its task was much greater than that of the final algorithm, algorithm "A." The latter has two multiplication operations, which are less time-consuming than divisions, in each of 550 iterations and one addition and one multiplication in each of 302500 iterations. Not only does algorithm "A" avoid the use of divisions, it also has fewer loops to go through, which both contribute to the remarkable increase in efficiency. Thus, the user of the program significantly benefits from the optimization of the undersampling process.

Synchronizing Threads

Another issue examined in the development of this program was the synchronization of the threads that accessed the camera at different points in the program. Because the camera is a resource that many parts of the program have to share, it was necessary to synchronize the running of these threads so as to ensure that every thread could execute as planned and the camera wouldn't have any problems functioning as the user intended.

If the threads were not synchronized, several potential problems would arise. The CCD camera is a device that can only handle one task at once, whether that task be

reading the image, sending the status, or configuring itself to the users' specifications. If the camera is assigned one task, and then assigned another task without having first completed the first, serious problems can arise that range from sending back wrong data to freezing the system. In the diagram, potential problem spots are circled. The first and third circles indicate points at which the camera has been given two commands, "Expose Camera" and "Get Status", without having the chance to reply to the first command. These errors could result in incorrect data being sent back. The second circle illustrates two problems. One problem is that the camera has been commanded to send back two replies (Reply and Status) without first getting the command to acquire the status. The second problem is much more serious, as it affects the acquisition of the image. The camera has been designed so that the command to read the camera must be sent directly after the command to expose the camera. However, in the diagram, the camera has been directed to "Send Status" in between the "Send Reply" and "Read Camera." This could easily result in the freezing of the system or other disruptive events.

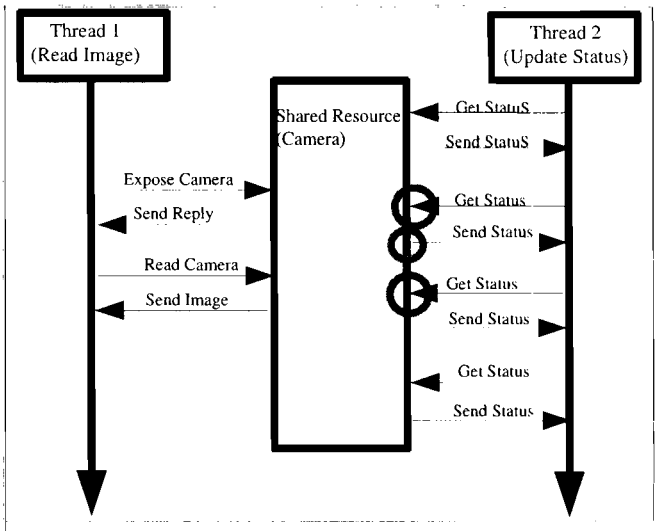

Figure 5
Threads Without Synchronization

Synchronization, however, prevents such errors from occurring by allowing only one set of commands to be sent to the CCD camera at one point. The synchronizing logic blocks all other threads from accessing the camera until the camera has finished. This

blocking is represented by the shaded oval. When the "Reserve Camera" command is sent, a shield of sorts is constructed, and the commands from the "Update Status" thread cannot get at the camera. Only after the camera has finished its task is the camera released and another set of commands may be sent by a different thread and a new task performed. Thus, the "Update Status" thread resumes. This solves the problems posed by the first and third circles in the above diagram, and some additional logic solves the problem of exposing and reading in the second circle.

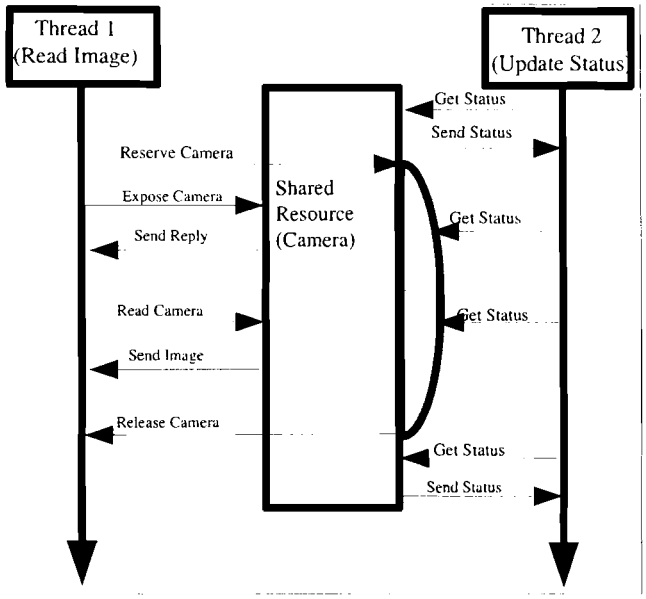

Figure 6
Threads With Synchronization

Advantages

The advantages of the CCD Control program include real-time status updates and increased flexibility in manipulating the acquisition of the image. The temperatures of the CCD and the backplate, as well as the vacuum chamber pressure, are newly acquired and displayed every second, and progress bars indicate remaining exposure time and acquisition time each time an image is acquired. These serve to keep the user informed as to the camera's current status.

Several methods were employed that increase the user's ability to manipulate the image acquisition. A panel was constructed that holds a few of the most used settings in

one place. This allows for easier modification of the settings, as the user does not need to know the number of the specific configuration or parameter that holds the value of the desired setting.

The scaling process was also updated to keep in place with new technologies in the CCD cameras. Older models of the camera sent image data back in 12-bit format, while the newer ones have changed to 16-bit. To accommodate this, the scaling was changed from 12-bit to 16-bit, the highest value (represented by white) changing from 4096 (2^{12}) to 65536 (2^{16}). Because of this rise in values, the process of selecting a value was made somewhat easier and more exact with the addition of text boxes where the user can input the desired scale values, instead of depending on the accuracy of the slider.

Conclusion

The successful development of a Linux-based program to control CCD cameras is of great assistance to diagnostic scientists at LLE. Scientists now have the ability to choose which operating system to use to control CCD cameras, which is of importance as Linux use increases. The CCD Control program allows users with little knowledge of the individual commands sent to the camera to interact with the CCD camera efficiently and effectively. The program has been tested in the Generic Streak Camera Platform (GSCP) and will undergo further improvements in the future before its implementation.

Acknowledgments

First and foremost, I would like to extend my greatest thanks to Dr. Christian Stoeckl for advising me on this project. Without a doubt, it is due to his expertise and guidance, as well as his remarkable patience and dedication, that this was completed. I would also like to thank Dr. R. Stephen Craxton for giving me the opportunity to participate in this program.