A Framework for Dynamic Volume Rendering in Ptolemy II

Tiffany Crawford

EECS 290N Report

University of California at Berkeley
Berkeley, CA, 94720, USA

tiffcraw@eecs.berkeley.edu

Abstract

The framework for Dynamic Volume Rendering in Ptolemy II and the issues that arise with it are introduced in this paper. The motivation for volume rendering in Ptolemy II, is for the application of medical imaging. A toolkit for medical imaging was an initial goal of this work. However, the author slowly realized that having a deeper understand of the existing work, Ptolemy II and the API Java 3D was necessary. This paper attempts to develop a foundation from which the author can build upon to add the suggested functionality to Ptolemy II.

1 Introduction

Ptolemy II is a research software that is in development at the University of California, Berkeley. It is a package of java classes written and integrated in a way that addresses the needs and demands of embedded systems. This is done via Models of Computation (MoC) that define the semantics of the components, or actors that make up a system.

2 Background

2.1 Java 3D

Ptolemy II uses the Java API, Java 3D to create graphics. Java 3D is a unique API that has some overhead user learning and places constraints on the code design. A functional Java 3D program has two main components, a scene graph and a Canvas3D. The scene graph structure allows the user to describe components of a scene and how they relate to one another. For example, in defining the motion of an arm, the motion of the hand most have some motion relative to the hand in which it is attached to. Canvas3D is the window in which the scene created can be viewed.

Figure 1, is an example of a simple scene graph in Java3D, very similar to that used in the Ptolemy II graphics models. At the very top of all Java 3D scene graphs is the Universe, which represents everything a viewer sees in the world. At the next level, there is a locale which gives a frame of reference as to where the user exists in the world. Below the locale there are two BranchGroups.

The BranchGroup on the left leads to other Groups and/or Nodes that define objects that will be viewed in the space. Nodes are associated with NodeComponents. The ovals in the figure represent these components, they are not stand alone. Specifically in the volume rendering case, there is a Node that represents the 3D object, and then two NodeComponents, Geometry and Appearance. The path on the righ leads to the ViewPlatform. The ViewPlatform is the 'camera' of the scene graph. Once there is a scene and a camera in which to view it, the View tells what to render.

The importance of understanding Java 3D is important in this research because it places unique demands on the Model of Computation used for graphics models.

3 Related Work

There are two existing toolkits in Ptolemy II that provide actors and/or infrastructure that will be used in this work. An image processing library was done as a Master's Project by James Yeh, and the Graphics Toolkit was done in collaboration with a large contribution by C. Fong.

3.1 Image Processing Toolkit

The image processing toolkit takes advantage of the Java Advanced Imaging API to implement many of its actors. It

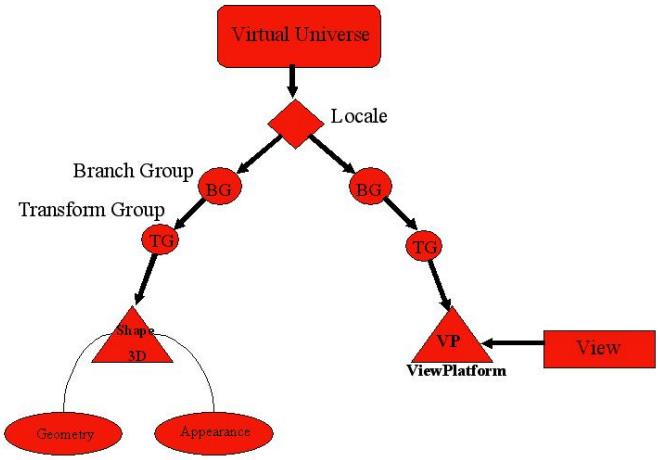

Figure 1. Java3D scene graph (Sun Microsystems)

defines a class JAIImageToken that creates a token that contains a javax.media.jai.RenderedOp. This class extends ImageToken an abstract class that creates an empty token that is compatible with java.awt.image. The importance of this will become apparent.

3.2 Graphics Toolkit

The graphics toolkit takes advantage of the Java3D API to implement many of its actors. Due to the nature of Java 3D, discussed above, this toolkit defines its own director that performs static scheduling. This director, the token SceneGraphToken, the actor ViewScreen, and a base class, GRShadedShape, will be very important in our discussion of volume rendering. A brief synopsis of their function and contribution follows.

3.2.1 GRDirector

The actors in this library do not have any meaning without an instantiation of GRDirector. GRDirector, is a director in Ptolemy II that extends StaticSchedulingDirector. It performs a topological sort of a directed acyclic actor graph. The calling of the prefire, fire and postfire methods of the actors in this graph are all called from within the director. Currently the director does not perform any sort of updates to its schedule. Once it begins firing the actors, it stays in this loop until it is time to exit the session.

3.2.2 ViewScreen

All GR domain actor graphs must contain a ViewScreen. The ViewScreen provides the Universe, Locale, BranchGroup, ViewPlatform and Canvas3D needed to create a Java3D scene. The other actors in the Ptolemy II actor graphs, act as Nodes and/or Transform Groups (see Background). In this design most of the scene graph is inside the ViewScreen. This makes the behavior of the actor graph someone intuitive, considering the other actors behave like nodes to be added to the scene. The constraint that arises that is not present in Java 3D, is the presence of only one "'Window"' to the world. In Java 3D you may have multiple.

3.2.3 GRSceneGraphToken

GRSceneGraphToken is a token type defined specifically for the GR domain. At least one output port of all actors that extend GRShadedShape, GRTransform and the input ports to both GRTransform and ViewScreen are compatible with this token type. References to nodes to be added to the scene graph are handled via this token.

3.2.4 GRShadedShape

GRShadedShape is a base class used to define 3D shapes in Ptolemy II. It is somewhat analogous to the triangle in Figure 1. The base class creates the NodeComponent, Appearance, in its initialize method, by calling createmodel(). In its fire method which is currently called once it sends a referencce to this node to be added to the scene graph. The NodeComponent, Geometry, is defined by the actors that extend GRShadedShape.

4 Platform

4.1 Data

DICOM (Digital Imaging and Communications in Medicine) is a data format developed by ACR and NEMA to provide a medical imaging standard [4]. It is becoming more and more widely used in medical imaging. The newer MRI machines automatically output DICOM images instead of the proprietary formats formerly used. The Medical Imaging Toolkit in Ptolemy will read in DICOM images and create an AWTImage token. AWTImageToken is a class in Ptolemy II that contains a java.awt.image.

4.2 ImageJ

The medical imaging toolkit takes advantage of a Java API. This API, ImageJ, was developed by Wayne Rasband at the National Institute of Health (NIH). ImageJ is an

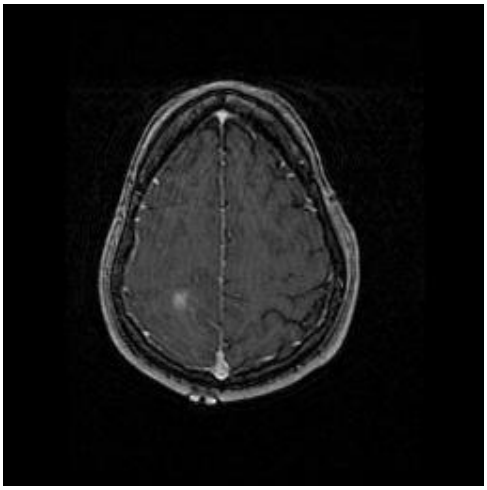

Figure 2. MRI Image encoded and converted to JPEG using ImageJ

open-source package of Java classes that read and analyze medical images. ImagePlus is a class within ImageJ whose subclasses can read different image formats including DICOM. ImagePlus implements a java.awt.Image interface. Figure 2, is a image read by this program.

4.3 GR Domain

Needless to say the rendering will be done using the GR MoC in Ptolemy II. As mentioned earlier, the GR domain has certain constraints placed upon it by Java 3D and other constraints due to its design. In building the platform, considering changes that may be neccessary to the existing structure need to be adressed.

4.3.1 GRDirector

The GRDirector, performs a static scheduling at initialization and keeps this schedule throughout the entire session. In the current design the fire method does not call any update method to update the schedule. In the new platform, at the beginning of each fire of the director an update method will be called to check the workspace graph for changes and if changes exist buildActorTable(), will be called again.
These changes will allow new data and/or actors to be added to the scene graph during a session and allow the ViewScreen to display it, without reinitialization.

4.3.2 ViewScreen

The current design of the GR domain requires one and only one ViewScreen to execute properly. As models are built it may be useful to have two ViewScreens in one session. This may be achieved via composite GR actors inside of another domain.

4.3.3 GRShadedShape

GRShadedShape will be the base class of Axis2DRenderer and Axis3DRenderer if that path is pursued. Java3D has a feature called capabilities. The capabilities vary according the the class being called, but for this discussion the capabilities deal with read and write access. The default is false, due to a more optimal implementation of the program with this setting.
Dynamic volume rendering requires write access and also the ability to update the reference to the node, so that the ViewScreen can display the update. GRShadedShape makes a scene graph connection in its fire method, and is never called again. The new design require for fire method to be called at each iteration. During fire() a test for a change in parameter will be perforemed and return true if changes have been made. A return of true will lead to a call of the proper update methods.

5 Future Work

5.0.4 ImagePlusReader

Extends ImageReader. Reads JPEG, TIFF, DICOM, BMP, and PGM file formats. The image is read in from a file via a parameter and has two outputs. One output will be able to produce multiple images if a file contains more than one image. This will be done via a parameter, stacksize. The second output will contain the image's metadata

5.0.5 Axis2DRenderer

This actor will produce a 3-D image from a stack of image slices. The details of how Axis2DRenderer will receive and handle its data are still being explored, however a probable method to present the idea follows. It may have two inputs, a single image or a stack of images, and one output, a 3-D image. One input will receive images from ImagePlusReader. The second input will receive the metadata from ImagePlusReader. The actor will use this metadata to properly form a 3-D output from the 2-D slices. Image can be updated, but will be redrawn

5.0.6 Axis3DRenderer

Another possible alternative to Axis2DRenderer which allows the image to be dynamically updated without being redrawn is Axis3DRenderer. This can perform volume rendering using a 3D texture. This method is preferable for

the above reasons however, a lot more computationally demanding and thus slower.

5.0.7 Image Segmentation

This actor will attempt to locate abnormalities within the image slices. This actor will have one input, a 2D slice, as in the output of ImagePlusReader. It will also have a set of parameters as needed.

6 Conclusion

A set of actors which will add to two existing Ptolemy II domains, is being created. The demands that some of the actors place on the Model of Computation and how they are being adressed are discussed. It is now hopeful that the framework and background provided will allow for the future work to be more approachable.

References

[1] Shuvra Bhattacharyya, et. al"Heterogeneous Concurrent Modeling and Design in Java (Volume 3:Ptolemy II Domains) ," C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, H. Zheng, " Technical Memorandum UCB/ERL M04/17, University of California, Berkeley, CA USA 94720, USA, June 24, 2004.
[2] James Yeh, " Image and Video Processing Libraries in Ptolemy II," Master's Report, Technical Memorandum No. UCB/ERL M03/52, University of California, Berkeley, CA, 94720, USA, December 16, 2003.
[3] C. Fong, S. Neuendorffer. Source code comments. ptolemy.domains.gr.kernel.GRDirector.
[4] http://medical.nema.org/dicom/2004/0401PU.PDF
[5] http://www.java3d.org