Overview of the Goals and Present Status of the Graphics Teaching Tool Project

Dana Tenneson

Department of Computer Science

Brown University

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Department of Computer Science at Brown University

Signature (Prof. Andries van Dam, Project Advisor) Date

Abstract:

Proficiency with professional graphics software is a common requirement for many non-technical as well as technical occupations. However, there are few pedagogical tools for teaching fundamental computer graphics concepts outside of computer science. This report describes the Graphics Teaching Tool (GTT), a pedagogical graphics program for teaching these fundamental concepts.

Inspiration:

The application of computers as visual tools is a widespread phenomenon. As such, people in many occupations outside of traditional computer engineering need to be proficient with graphical computer tools. Photographers and graphic designers use Adobe's Photoshop to manipulate images. Business professionals use Microsoft's PowerPoint to create their meeting presentations. Artists and engineers alike use 3D modeling packages in their trades. A basic understanding of the concepts of computer graphics is a useful stepping-stone towards understanding the potentially overwhelming complexity of these standard graphical systems. Even people who don't actively create computer visualizations often need to be able to interpret visual representations of data generated by computers. A basic knowledge of computer graphics is becoming part of IT literacy. Anticipated use of the GTT should address some of the goals stated in recent reports calling for increased IT literacy and better education of the IT workforce [PITAC, PCAST 1997] by helping educators better teach graphics concepts.

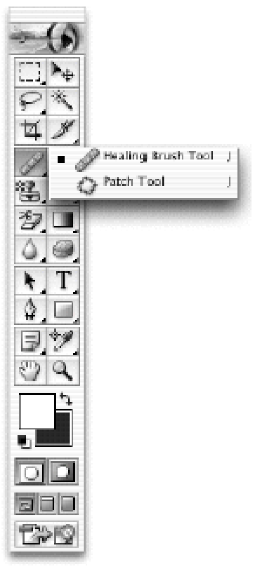


Figure 1: The Adobe Photoshop 7.0 toolbar contains 22 tools, 17 of which have pull-down options. This is a lot of options at first glance.

The form of this project is inspired by learning science research which has repeatedly found that hands-on, constructivist learning experiences leads to better learning outcomes than traditional lecture and reading [Jonassen 2003]. Searches of online educational software repositories and related conference literature (e.g., SIGGRAPH educational program abstracts) did not reveal any attempts in this direction for the targeted non-majors audience.

Related Work:

Standard texts in computer graphics have been published for both technically proficient audiences [Foley 1992, Hearn, 1996] and artists [Spalter 1999] but do not take advantage of the interactive learning possible with today's computers. Tutorials that detail the features and complex options of professional graphics packages are commonly available, but mostly gloss over the underlying graphical concepts or take them for granted [Adobe]. They present step-by-step instructions on how to perform specific common tasks, but leave the user without a solid foundation to solve other problems. Many algorithm visualizations and other types of hands-on tools, such as Exploratories [Exploratories], have been developed to teach graphical concepts, but are chiefly designed for computer science students. The mathematical and algorithmic intensity makes them largely unsuitable for people outside of their target audience. At the other extreme, children's applications such as Creative Writer and Kid Pix do include smaller feature-sets than professional applications, but such programs don't explicitly present any of the underlying graphics concepts [Druin 1998]. A middle ground between these various forms of graphics education—one in which a small feature set is combined with pedagogical instruction—is necessary to teach graphics fundamentals.

Overview:

The Graphics Teaching Tool is a graphical creation environment that combines multiple types of computerized visual data into a single working space. It currently contains a 2D Raster paint-type module and a 3D Geometric modeling-type module. A 2D Geometric vector graphics program is planned to be included and a 3D Voxel environment may be included as well. These various data types are layered together in a single panel instead of being broken into separate specialized applications. This facilitates the user's ability to explore and contrast the capabilities of the different data types as well facilitating use in a classroom setting since students don't need to start multiple resource-heavy professional applications.

The GTT's pedagogical dialogs provide a means to "look under the hood" of the data types and their tools. Each pedagogical dialog explores a fundamental concept in computer graphics without requiring knowledge of programming. The Data Inspection Tool allows the user to click any visually represented object and examine its underlying data. Some tools require the user to experiment with the nature of the tool and design their own personal version. For example, instead of coming with the standard circular brush tool common to paint programs, the GTT requires the user to design the alpha footprint of the brush thereby creating their own custom tool. All such pedagogical dialogs contain hypertext document areas for information relevant to the presented concept and walkthroughs of using the dialog. This facilitates exploration of the application and the basic graphical concepts without direct classroom instruction.

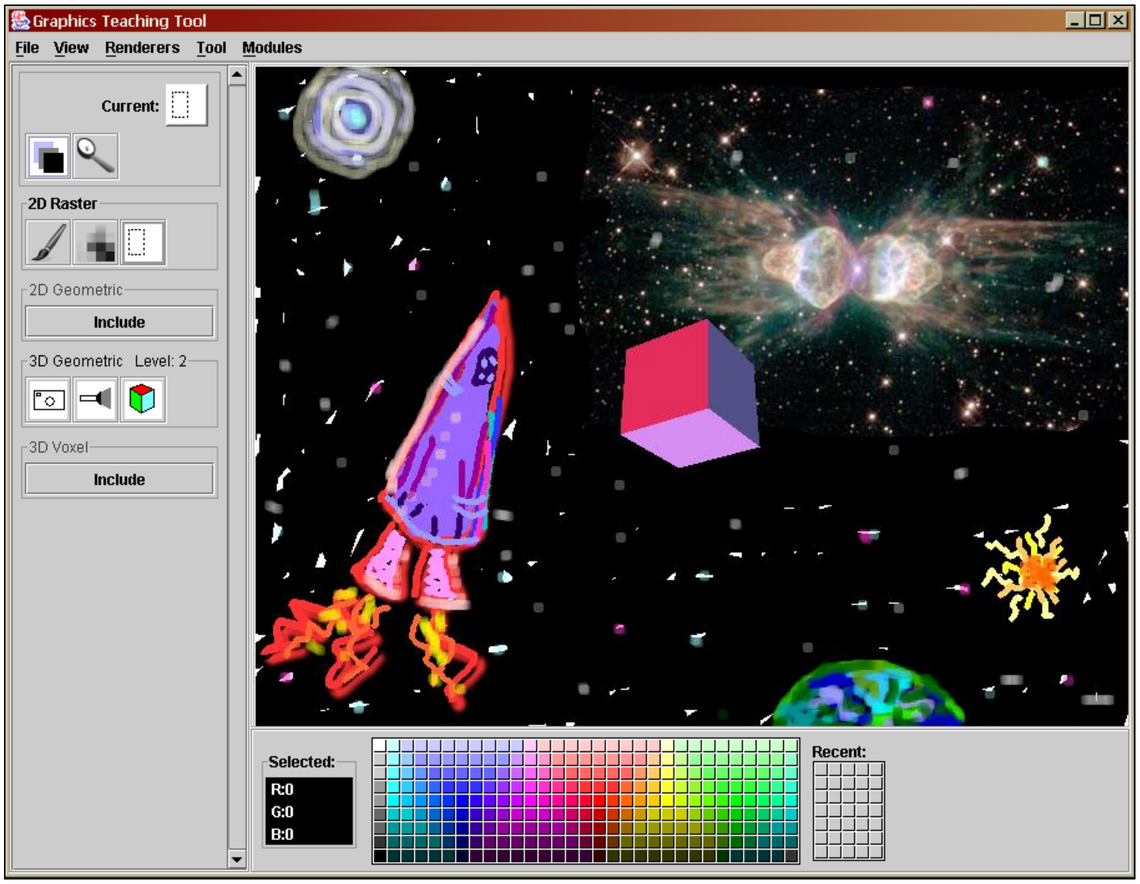


Figure 2: The GTT Main Interface with 2D Raster and 3D Geometric modules loaded.

Design Goals:

The GTT is intended to be a free, low-overhead, expandable educational tool that can produce non-trivial graphical scenes. It is both a Java applet and application so that classrooms can choose which way to run the program based on their needs. The GTT is more efficiently run through local installation of the application but can be run over the web as an applet with less technical administration. The GTT's module architecture (described below) minimizes memory overhead in order to place less of a burden on older computers and facilitate expandability. For pedagogical purposes, the GTT attempts to provide most of the functionality of a suite of professional graphical applications through a greatly reduced set of tools that illustrate the basic concepts of each data type. The usefulness of the GTT environment for assignments and class projects is enhanced by import and export features that let student tie in GTT work with commercial programs. One can import .tif files, such as the Hubble space image in Figure 1, and users can also export the final scene as raster data for further editing in a a professional program.

Module Architecture:

The GTT consists of several separate libraries wrapped into jar file modules. Each module contains a segment of the application's functionality. There are four categories of module, Essential, Module, Renderer, and Toolkit. The Essential category applies to the module GTT_Main which contains two libraries. The first is the core application library containing the user interface and module architecture. The second is the Basics library containing classes not specific to the GTT such as mathematical utility functions and generic UI elements. The Module category applies to the libraries which house the core of each graphics data type such as 2D Raster. The Renderer category is for optional renderers added to Modules in addition to the default renderer built into the Module. The Toolkit category is for external data manipulation tools developed for a Module but not part of the Module. For example, a developer could create a paint fill tool to add the 2D Raster Module without modifying the Module source.

When run in application mode, the GTT's Plugin Loader scans the home directory of the application and identifies which jar files are part of the GTT. Using information stored in their manifest files, the modules can be registered with the application, but not consume overhead in the Java Class Loader until the user chooses to load the module. The user's saved preferences from a previous session may be set to automatically load previously active modules into the system at startup. When the GTT is run in applet mode, the same process takes place with the exception that modules are identified by name in a text file on the web server instead of automatically identified.

This modular system allows independent developers to produce and release need-specific additions to the GTT. For example, Constructive Solid Geometry Tool could be added to the 3D Geometric module for an architecture class or an entire Constructive Solid Geometry Module could be created. This new module could be placed in the application directory with the core modules and automatically become available upon restart. This specialized plugin system allows developer flexibility and user simplicity not present in more standard Java pluggable architectures founded on separate library jar files being installed at a system level.

Core Functionality:

The GTT provides a core set of functionality separate from the data-specific tools. The core tools are the Data Inspection Tool and the Layer Sorting Tool. The core menu actions are the capability to import and export data, save and load personalized workspaces, and choose a data renderer. A native file format and workspace management will be included in the future. Color picking is currently included in the core with a limited palette of colors.

The Data Inspection Tool dialog changes appearance based on the type of data queried. Therefore, each Data Inspection Tool dialog is described in each data type's functionality section.

The Layer Sorting Tool allows the user to change the order of the data type layers. Each data type gets only one layer to prevent confusing situations where objects of the same data type do not interact correctly because they are in different layers. However, each data type may have its own internal layering structure. This is critical to 2D Geometric data which is, by its nature, layered. Other data types use internal layering for providing overlays of selected areas or other data specific diagrams.

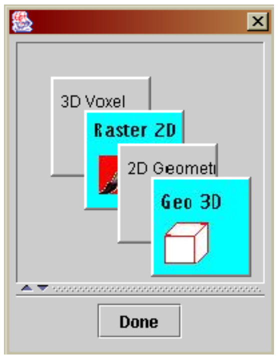


Figure 3: The Layer Sorting Tool.

Image import attempts to assign an incoming file to the proper data type by allowing the different data type libraries to bid in an auction for the privilege to import the file. Some files may be importable by various data types. Such a file would receive different magnitudes of bids according to the different data types' abilities to import the file. For example, the 2D Geometric Module would properly import an acrobat file but an approximation could be made with 2D Raster. In this case, if both Modules are present, both would place bids, but the 2D Geometric Module would outbid the 2D Raster. Whichever present Module believes it can best import the data wins the bid. Currently, 2D Raster is the only layer that can import data and it recognizes files according to the Java Image IO library. Image export occurs by taking a snapshot of the current layered display and it is exported with the Java Image IO library.

The custom tools that users create in pedagogical dialogs define their workspace. Individual tools can be saved and loaded to allow tool sharing. Additionally, the entire workspace can be saved and loaded. This process attempts to occur automatically at application start and exit to provide easy access. During the course of learning from the GTT, the user will go from having a generic workspace with few pre-defined tools to a customized workspace with tools largely designed by the user.

The ability to set the renderer for each kind of data reinforces the concept that the underlying data is different from its representation. Each Module has a default fast renderer that is used during animated updates of the data such as the middle of a brush stroke or the rotation of a cube. The user can select a different kind of renderer to render the data once the animation has completed. Possible 3D Geometric renderers could be based on ray tracing, non-photorealistic rendering, etc. Less obviously, 2D Raster data can be rendered in alternate ways as well. The GTT currently contains an ASCII text renderer for the 2D Raster data as an example alternate renderer.

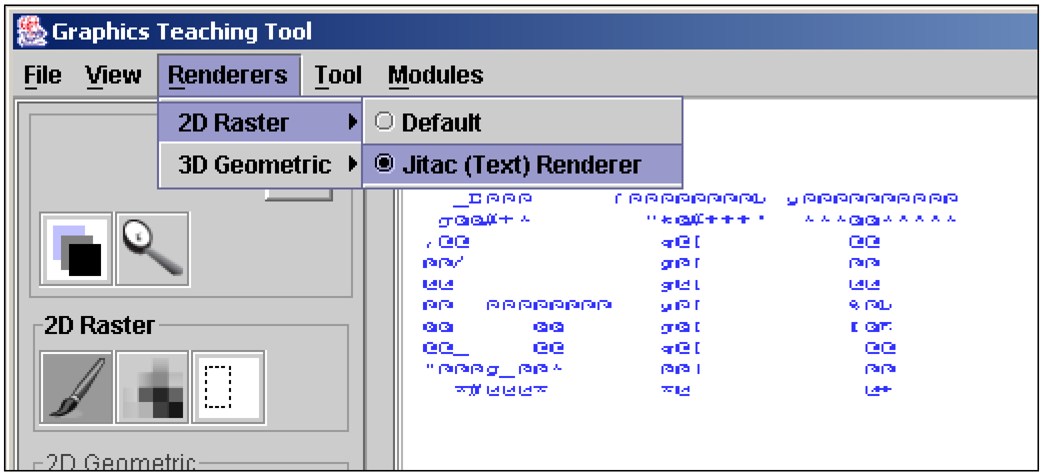
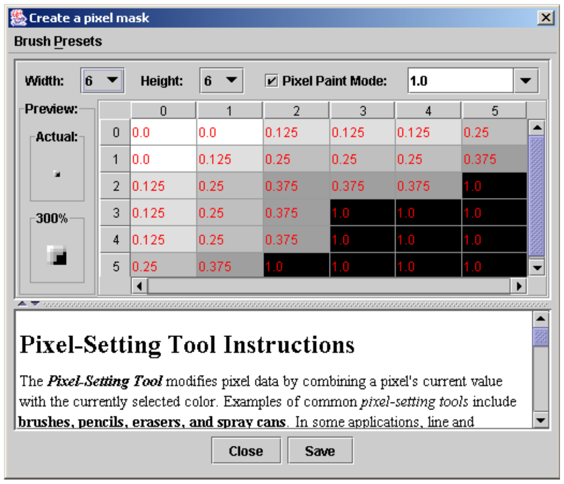


Figure 4: The example Jitac ASCII renderer for 2D Raster data.

2D Raster Functionality:

The wide array of tools in standard paint programs were reduced to three simple tool concepts for the 2D Raster module, namely pixel-setting, filtering, and selection. Pixel values can be set using a Pixel-Setting Tool which acts much like a simple brush. Pixel values can be modified based on nearby pixel values with a Filter Tool. Pixels can be isolated for selective filtering and movement using a Selection Tool. Common paint tools which are absent from this list include the fill tool whose effect can be slowly reproduced with the Pixel-Setting Tool and various shape and text tools whose functionality is better suited to 2D Geometric data. The Data Inspection Tool dialog for 2D raster is a grid of the pixel RGBA values.

The Pixel-Setting Tool uses an alpha mask to apply a color to the raster data. A user designs their own mask by choosing the dimensions of the mask and then either choosing a preset brush type from a menu or filling in the alpha values in a table. To facilitate the fast creation of custom masks, values in the table can be not only edited manually with the keyboard, but also selected from a combo box of selections. A 'Pixel Paint Mode' of the table also allows the user to select an alpha value and rapidly apply that value by painting across the desired cells. Future versions of the Pixel-Setting Tool will include the ability to select which Porter-Duff alpha compositing rule the mask will use. This will make tools like erasers and Photoshop-style airbrushes possible.



Width: 6 Height: 6 ✓ Pixel Paint Mode: 1.0

	0	1	2	3	4	5
0	0.0	0.0	0.125	0.125	0.125	0.25
1	0.0	0.125	0.25	0.25	0.25	0.375
2	0.125	0.25	0.375	0.375	0.375	1.0
3	0.125	0.25	0.375	1.0	1.0	1.0
4	0.125	0.25	0.375	1.0	1.0	1.0
5	0.25	0.375	1.0	1.0	1.0	1.0

Pixel-Setting Tool Instructions

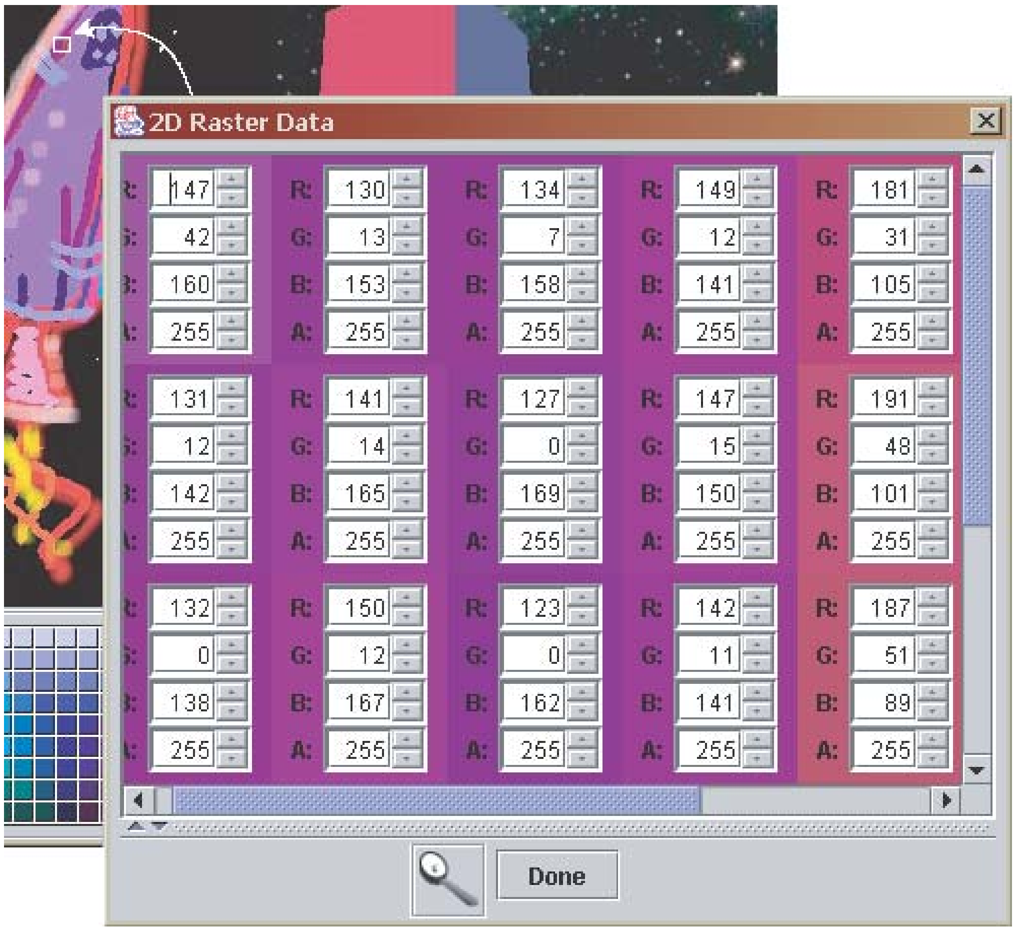
The ***Pixel-Setting Tool*** modifies pixel data by combining a pixel's current value with the currently selected color. Examples of common *pixel-setting tools* include **brushes, pencils, erasers, and spray cans**. In some applications, line and

Figure 5: The Pixel Setting Tool

The Filter Tool creates kernel filters using much of the same interface as the Pixel-Setting Tool. Procedural filters such as scaling were avoided due to their complexity. The Filter tool comes with built-in options for creating constant blur and constant sharpen filters with the intention that students should create their own complex blur and sharpen filters along with other kernel filters.

The Selection Tool allows the user to designate an area of the 2D raster data to be affected by a filter or moved to another part of the raster layer. The Selection Tool comes in four preset varieties. The three common selection types can be made that are rectangular, ellipsoid, freehand polygonal. Lastly, a constructive area geometry selection tool allows a user to define a selection by "painting" over an area with a small circular brush. Once a selection is made, the area can be dragged around the environment and dropped elsewhere. Filters are applied only to the selected area. The user can delete selected data by pressing the Delete key.

The 2D Raster Data Inspection Tool dialog is a grid of the pixel RGBA values for the immediate area around the mouse click (shown in the image by a pulsing rectangle). Spinners allow the user to manually change a pixel's RGBA values. The design for this tool was inspired in part by the old Apple MacPaint "FatBits" feature, which let users zoom into a section of their work and edit the individual pixels.



R: 147	R: 130	R: 134	R: 149	R: 181
G: 42	G: 13	G: 7	G: 12	G: 31
B: 160	B: 153	B: 158	B: 141	B: 105
A: 255	A: 255	A: 255	A: 255	A: 255
R: 131	R: 141	R: 127	R: 147	R: 191
G: 12	G: 14	G: 0	G: 15	G: 48
B: 142	B: 165	B: 169	B: 150	B: 101
A: 255	A: 255	A: 255	A: 255	A: 255
R: 132	R: 150	R: 123	R: 142	R: 187
G: 0	G: 12	G: 0	G: 11	G: 51
B: 138	B: 167	B: 162	B: 141	B: 89
A: 255	A: 255	A: 255	A: 255	A: 255

Figure 6: The Data Inspection Tool dialog examines pixel RGBA values in an area of a raster image.

3D Geometric Functionality:

Three tools are currently implemented for the 3D Geometric data type. Objects can be created and translated with the Object Tool. The virtual camera can be moved with the Camera Tool. Lights can be created, moved, and colored with the Light Tool. The Data Inspection Tool dialog presents a scene graph of the 3D scene for the user to manipulate.

The Object Tool currently presents the user with the ability to create cubes, spheres, and cylinders. An object design dialog is planned that will allow a user to design tessellated objects. 3D file imports will be linked to the Object tool so that objects can be loaded from disk. The object is initially placed at the origin of the scene. Clicking and dragging the object can translate it along the X and Y axes with the mouse scroll wheel controlling Z-axis movement.

The Camera Tool ties mouse movements to the parameters of the camera. In Ken Perlin's Handy Dandy Pure-Java1.0 3D Renderer, the 3D library currently being used as the update renderer, the location, focal length, and field of view define the camera. The camera is always facing the origin. The library authors are considering changing this restriction in the near future and the GTT would then be able to expand the Camera Tool options. With the direction mode selected in the Camera Tool, mouse dragging moves the camera on a sphere around the origin. With the frustum mode selected, mouse dragging changes the field of view and the focal length.

The Light Tool allows the user to design a light. Lights are directional lights tied to the location of the camera due to restrictions in the 3D library. The Light Tool dialog allows the user to turn a light on or off, change the color of the light, and change the angle of the light. As the angle of the light changes, a diagram is overlaid on the 3D scene to show the new orientation of the light relative to the origin.

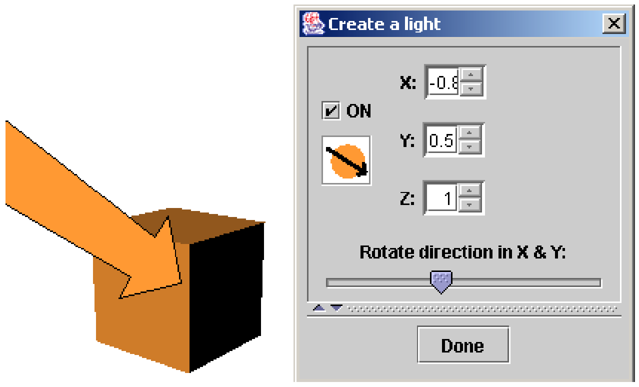
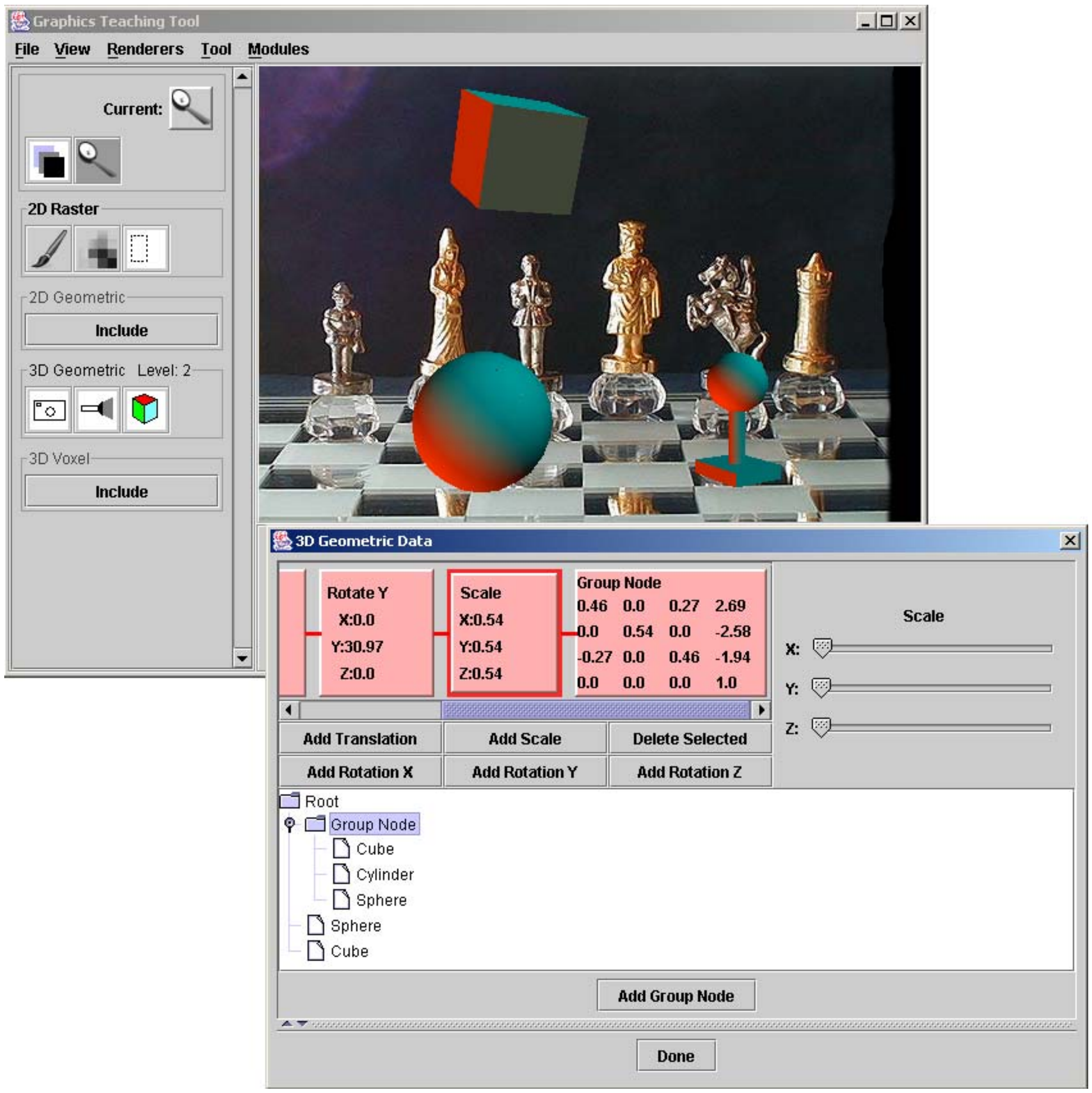


Figure 7: The Light Tool creates a directional light.

The Data Inspection Tool dialog for 3D Geometric data presents a scene graph of the 3D data. For user simplicity, the groupings and transformations are separated into different sections of the display. A tree shows the group structure of the scene where the user can drag objects and group nodes to change the scene structure. The currently selected node and the transformations between it and its parent are depicted at the top of the dialog. The user can insert new transformations, delete existing transformations, and reorganize the order of the transformations. The transformation's key values, rather than its matrix represent, each transformation type. For example, a rotation is depicted as the degree of rotation around the axis instead of as a matrix of sines and cosines. In the future, the user will be able to switch modes between seeing the values and seeing the matrices. The currently selected node has its composite matrix depicted at the end of the chain of transformations.



Rotate Y
X:0.0
Y:30.97
Z:0.0

Scale
X:0.54
Y:0.54
Z:0.54

Group Node

0.46	0.0	0.27	2.69
0.0	0.54	0.0	-2.58
-0.27	0.0	0.46	-1.94
0.0	0.0	0.0	1.0

- Root
  - Group Node
    - Cube
    - Cylinder
    - Sphere
  - Sphere
  - Cube

Figure 8: The 3D Geometric Data Inspection Tool dialog modifies the scene graph of the 3D scene.

Core Organization:

The core of the GTT is the MainPanel. This JPanel contains all of the UI components except for the menu bar and acts as a switchboard for components that need to access components outside of their container hierarchy. Within the MainPanel are objects to control tool selection, color selection, and the main canvas.

The ToolPanel manages tools and sorts them according to Module data type. When a Module is disabled, the tool panel contains an “include” button for the Module. The ToolPanel also relays notification of Tool selection to other core classes.

The ColorPanel contains the swatch panel from the JColorChooser and a component displaying the RGB values of the currently selected color. Expansion of the ColorPanel to contain pedagogical dialogs on color theory may be included at a later date.

The CanvasPanel layers the various data types together. It contains a CanvasLayer for each data type which is rendered according to the assigned LayerRenderer. The CanvasPanel is responsible for registering itself for mouse and key events that the currently selected tool wants and for being able to detect which layer a click point corresponds to.

Module Organization:

Within each Module, there is a class that implements the Module interface. This Module is the class that is initially instantiated upon loading. It acts as a producer of other module classes and as the routing switchboard for the module elements that need to communicate with each other. Tools and Renderers appended to Modules during module loading are assigned pointers to their parent Module upon Module instantiation. Via the Module, any class can access the Module's data set and CanvasLayer. If necessary, the CanvasLayer can then be used to access the MainPanel.

External Libraries:

The GTT currently uses two external libraries. The 3D Geometric Module uses Ken Perlin's Handy Dandy Pure-Java1.0 3D Renderer as the default renderer. The 2D Raster Module has an example external module renderer, which uses Jitac (Java Image To ASCII Converter) to provide text renderings of the raster data.

As the name states, Ken Perlin's library provides a 3D renderer using only Java code in a Java 1.0 compliant applet. The library was modified to add support for rendering as a lightweight JComponent with transparency and to allow scene graph changes after instantiation. This library was chosen over Java3D as the default 3D library for several reasons. In particular, its size, continuing support, and pure-Java codebase make it superior to Java3D for the GTT. It does lack the vast functionality of Java3D, but contains the fundamentals for a pedagogically oriented application.

The Jitac library by Konrad Reick renders images as ASCII text using Sun's Jimi (Java Image Manipulation Interface) library. The Jitac Renderer wraps Jitac's ASCII production routines outputting the colored HTML to a JLabel instead of a file.

Other Coding Considerations:

Because the goal for the GTT includes real classroom use and hopefully additions by others, great care was taken to make the code as easy to read and re-use as possible. Interface widgets adhere to standard Swing programming practices [Sun] and the code is in the process of being extensively documented internally with JavaDoc and externally with code explanation documents.

Project Statistics:

The GTT, excluding the external libraries, currently consists of 98 Files and 17,928 lines of code.

Future Work:

As stated above, the 2D Raster and 3D Geometric Modules are almost complete and the 2D Geometric Module will be started over the summer. The GTT is currently slated to have an initial release this winter so that it can be used in a 300-student introductory course for non-majors at Brown University (CS2) next spring and probably also in one or more courses at neighboring Johnson and Wales University, a vocational school that has started a rapidly expanding School of Technology. A NSF CCLI grant proposal to fund continued development of the GTT will be produced this summer. Finally, assessment will be crucial to the continued success of this effort. Although the GTT does not have the multitude of features available in professional production-level software, the GTT will let students solve real-world design and image-creation problems in a range of courses. According to learning theories such as those expounded in [Jonassen 2003] this will make the tool much more effective than canned demos and tutorials because it will allow for "authentic" assessment strategies of students' real solutions to real-world imaging and design problems.

Acknowledgements:

I'd like to thank Anne Spalter for her many hours spent working with me giving support, feedback, and design help on most aspects of the GTT, Andy van Dam for useful, concise feedback on the project and getting me into a project with an educational aspect, and the people at Sun Microsystems for providing The Java Tutorial which was constantly on my desktop or in my lap during development.

References:

Adobe: *Classroom in a Box* and other materials. http://www.adobe.com/education/main.html.

Druin, 1998: Druin, Allison (Editor). *The Design of Children's Technology*. Morgan Kaufmann, October 1998.

Exploratories: http://www.cs.brown.edu/exploratory.

Foley 1992: J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practic.*, Addison-Wesley, 1992.

Hearn 1996: Hearn, Donald and Pauline M Baker. *Computer Graphics: C Version*. Prentice Hall, 1996.

Jonassen, 2003: Jonassen, David, Jane Howland, Joi Moore, and Rose M. Marra. *Learning to Solve Problems with Technology: A Constructivist Perspective*. Merrill Prentice Hall, 2003.

PCAST, 1997: The PCAST (Presidents Committee of Advisors on Science and Technology) Report to the President on the Use of Technology to Strengthen K-12 Education in the United States, March 1997. http://www.ostp.gov/PCAST/k-12ed.html.

PITAC: (The Presidents Information Technology Advisory Committee)
http://www.hpcc.gov/ac/report/.

Spalter 1999: Spalter, Anne. *Computers in the Visual Arts*, Addison Wesley, 1999

Sun: *Sun's Java Documentation Site*. http://developer.java.sun.com/developer/infodocs/.