An Interactive Tutorial System for Java

Eric Roberts
Stanford University
eroberts@cs.stanford.edu

ABSTRACT

As part of the documentation for its library packages, the Java Task Force (JTF) developed an online tutorial system that enables teachers and students to explore the resources provided by the Task Force in a highly interactive style. The individual pages that make up the tutorial often include demonstration programs that the reader can experiment with while remaining on the same web page as the explanatory text. Although the original motivation for developing that tutorial system was to document the JTF packages themselves, the structure is general enough for teachers to design their own tutorials and interactive demonstrations. This paper describes the structure of that tutorial system and illustrates its use. In addition, the paper describes some work-in-progress that will make it possible to create interactive lecture demonstrations from PowerPoint™.slides.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: computer science education, curriculum.

General Terms

None.

Keywords

Computer science education, CS1, teaching libraries, Java.

1. INTRODUCTION

The ACM Java Task Force was created in 2004 to develop a set of pedagogical resources that will make it easier to teach Java in first-year computer science courses [8]. The primary deliverables of that effort are a set of library packages that address those problems identified by the community as the most significant challenges involved in teaching Java to an introductory audience [1]. In addition to those packages, the Task Force has also produced several software tools that support the teaching of Java in other ways. This paper describes an online tutorial system that was initially developed to document the Java Task Force libraries, but which also makes it possible for instructors to develop their own interactive tutorials for students.

2. APPLETS AS A TUTORIAL PARADIGM

Ever since its introduction in 1995, educators have recognized that Java represents an attractive environment for the development of interactive teaching materials, primarily because of its integration with the web through the applet mechanism. The 1997 and 1998 ITiCSE conferences convened working groups to develop Java as a resource for creating visualizations and interactive animations for instructional use [3, 7]. The 1997 report offers the following assessment of the opportunities that Java provides:

> Visualization has long been an important pedagogical tool in CS education. The widespread use of the Web and the introduction of Java, with its ability to present interactive animated applets and other types of animation, all provide opportunities to expand the availability of visualization-based teaching and learning. [3]

Other authors have reported on similar attempts to use Java as a framework for developing teaching materials. The SIGCSE'98 proceedings, for example, include no fewer than five papers on the use of Java as a framework for supporting interactive, web-based tutorials and animations [2, 4, 5, 6, 9].

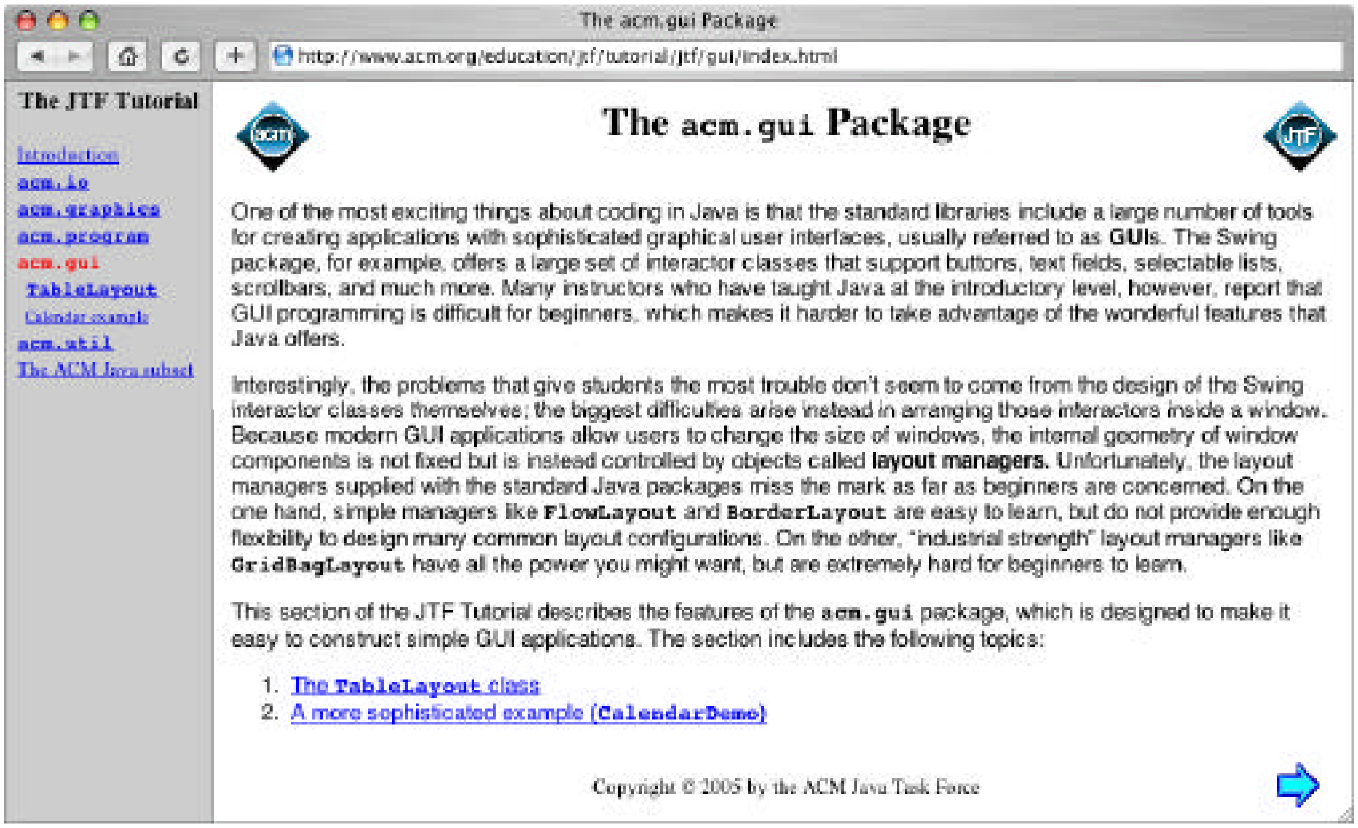
In recent years, the difficulty of maintaining compatibility among applets running in an ever-expanding array of browsers has led to some decline in the popularity of the applet paradigm. For reasons discussed at length in the January 2006 release of the JTF Project Rationale [1], however, the Task Force is convinced that applets continue to provide an appropriate framework for making software applications available on the web. The Task Force, moreover, has undertaken a major effort to eliminate compatibility problems, as described in the report.

3. OVERVIEW OF THE JTF TUTORIAL

The purpose of this section is to give readers a feel for how the online JTF tutorial system operates—admittedly in the noninteractive style that a paper copy imposes. Figure 1 shows the top-level page for the chapter on the **`acm.gui`** package, which supports the development of simple graphical user interfaces or GUIs. The browser window is divided into three frames. The frame along the left is the **navigation frame,** which contains an outline of the tutorial as a whole, expanded to show the detailed structure of the current section. The majority of the right-hand side of the window is occupied by the **text frame,** which provides the text description for that page. As you can see from the example, the text frame can use the full repertoire of HTML features; the sample page includes images for the ACM and JTF icons, multiple fonts to enhance readability, an ordered list identifying the subsections, and hyperlinks that connect the section names to the appropriate pages. The third frame, which is called the **control bar,** appears at the bottom of the window and includes the copyright notice and an arrow link to the next page.

Figure 2 shows a slightly more interesting page that appears later in this section of the tutorial. As you can see from the expanded outline in the navigation frame, this slide is the fourth slide in the **`TableLayout`** subsection of the **`acm.gui`** package description. More importantly, this page shows a sample applet

Figure 1. Sample page from the JTF Tutorial



The acm.gui Package

One of the most exciting things about coding in Java is that the standard libraries include a large number of tools for creating applications with sophisticated graphical user interfaces, usually referred to as **GUIs**. The Swing package, for example, offers a large set of interactor classes that support buttons, text fields, selectable lists, scrollbars, and much more. Many instructors who have taught Java at the introductory level, however, report that GUI programming is difficult for beginners, which makes it harder to take advantage of the wonderful features that Java offers.

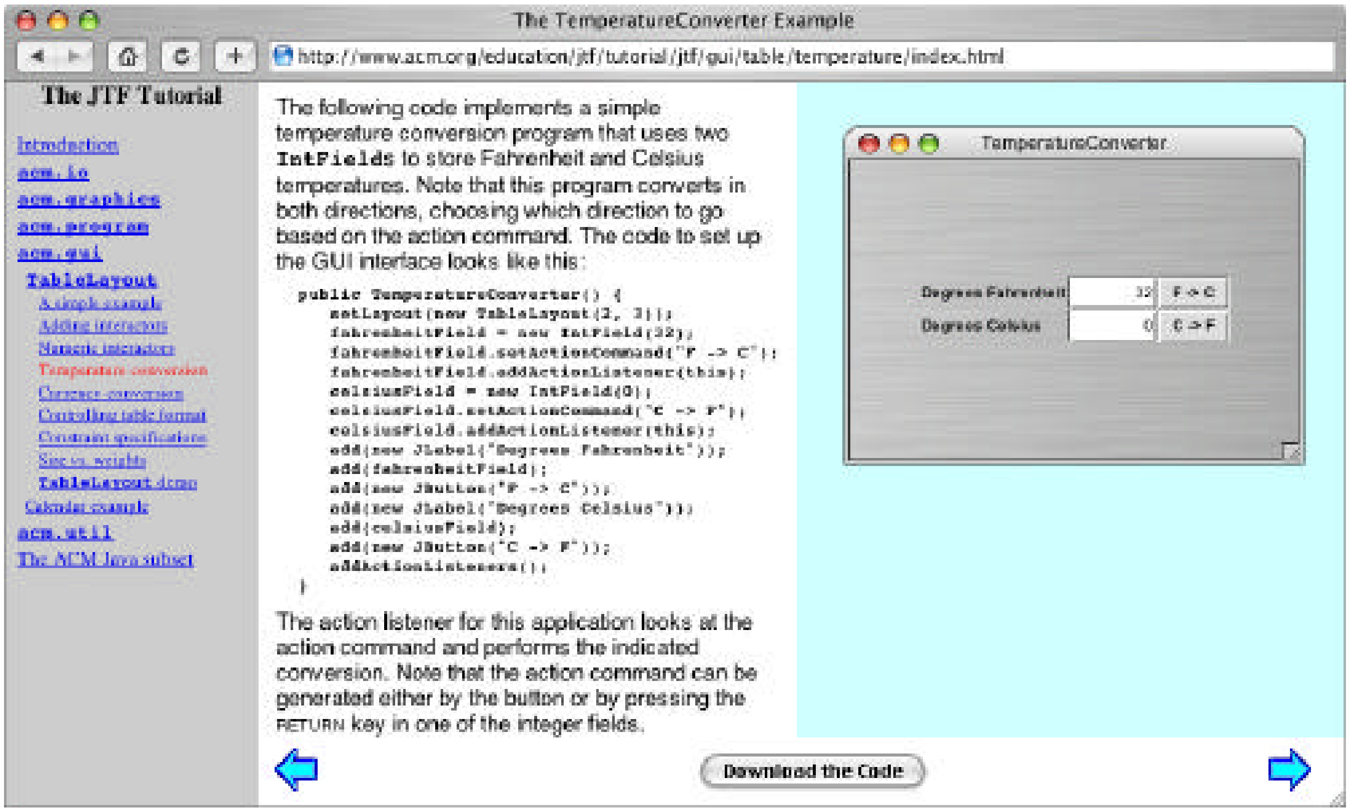
Interestingly, the problems that give students the most trouble don't seem to come from the design of the Swing interactor classes themselves; the biggest difficulties arise instead in arranging those interactors inside a window. Because modern GUI applications allow users to change the size of windows, the internal geometry of window components is not fixed but is instead controlled by objects called **layout managers.** Unfortunately, the layout managers supplied with the standard Java packages miss the mark as far as beginners are concerned. On the one hand, simple managers like `FlowLayout` and `BorderLayout` are easy to learn, but do not provide enough flexibility to design many common layout configurations. On the other, "industrial strength" layout managers like `GridBagLayout` have all the power you might want, but are extremely hard for beginners to learn.

This section of the JTF Tutorial describes the features of the `acm.gui` package, which is designed to make it easy to construct simple GUI applications. The section includes the following topics:

1. The `TableLayout` class
2. A more sophisticated example (`CalendarDemo`)

Copyright © 2005 by the ACM Java Task Force

Figure 2. Sample tutorial page showing embedded applet



The following code implements a simple temperature conversion program that uses two `IntFields` to store Fahrenheit and Celsius temperatures. Note that this program converts in both directions, choosing which direction to go based on the action command. The code to set up the GUI interface looks like this:

```
public TemperatureConverter() {
   setLayout(new TableLayout(2, 3));
   fahrenheitField = new IntField(32);
   fahrenheitField.setActionCommand("F -> C");
   fahrenheitField.addActionListener(this);
   celsiusField = new IntField(0);
   celsiusField.setActionCommand("C -> F");
   celsiusField.addActionListener(this);
   add(new JLabel("Degrees Fahrenheit"));
   add(fahrenheitField);
   add(new JButton("F -> C"));
   add(new JLabel("Degrees Celsius"));
   add(celsiusField);
   add(new JButton("C -> F"));
   addActionListeners();
}
```

The action listener for this application looks at the action command and performs the indicated conversion. Note that the action command can be generated either by the button or by pressing the RETURN key in one of the integer fields.

running in a **demo frame** that appears to the right of the text frame. As you can see if you look at the web-based version of the JTF Tutorial, the demo program is fully active. You can type a temperature into the Fahrenheit or Celsius field and then convert that temperature to the other scale either by hitting RETURN or by clicking the appropriate button. Finally, the control bar now contains a button that downloads the code for the applet running in the demo frame.

Although it may not be obvious at first glance, the demo frame contains more than the `TemperatureConverter` program itself. If you simply included an `applet` tag for this program in the HTML code for the frame, the applet would appear in a simple rectangular box, without the title bar and controls. Adding the simulated frame is not merely (if you will) "window dressing" but actually plays an important pedagogical role. In this section of the tutorial in particular, it is very important for readers to understand how the layout changes when the user resizes the window. The resize control in the lower right and the maximum/minimize buttons on the title bar are active and resize the window inside the demo frame. Resizing the window signals the layout manager for the applet that it needs to reconstruct the layout of the window, making it easy for the user to see exactly what happens as the size of the window changes.

The look and feel of the simulated frame adjusts to match the system on which it is running. The demo applet shown in Figure 2 shows how things look when the browser is running on a Macintosh. On a Windows platform, the style of the applet frame changes to conform to the standard appearance of the Windows environment.

4. DESIGNING YOUR OWN TUTORIALS

To see how the tutorial package might be tailored to support your own customized tutorial, the first thing you need to understand is what parts of the tutorial page are generated by the tutorial author and what parts are generated automatically. The short answer is that the author is responsible only for the text frame. Everything else is generated automatically. The navigation frame, the control bar, the demo frame, and the index file that creates the frame structure are all generated from the text pages by running the `CreateTutorial` application on the directory tree containing the specifications of the various text frames.

The `CreateTutorial` application, of course, needs to have a certain amount of information in order to create the HTML files that represent the page. In particular, there must be some way for the `CreateTutorial` application to determine the order of individual pages at a particular level of the hierarchy, the name of the class (if any) to run as an applet, and the location from which to download the code. This information is provided by adding a `slide` tag to the HTML file for the text frame. Because this tag is not recognized as part of HTML, browsers will ignore it. The `CreateTutorial` application, on the other hand, can scan files for this tag and use the attributes to generate the additional files needed to support the tutorial operation.

The attributes attached to the `slide` tag depend on the role of that page. For the `TemperatureConverter` slide, this tag must contain the information needed to identify the both the `.jar` file for the applet and the name of the main class. In this specific instance, the `slide` tag looks like this:

```
<slide archive="JTFTutorial.jar"
       code="TemperatureConverter.class"
       zip="TemperatureConverter.zip">
```

The `archive` and `code` attributes are the same as those found in a traditional `applet` tag and let the `CreateTutorial` application know how to start the demo program. If these attributes are missing, no demo page appears. Similarly, the `zip` attribute tells `CreateTutorial` where to find the downloadable code. It is this attribute that triggers the creation of the button in the control bar that downloads the code.

The tutorial page shown in Figure 1 has no demo program but is the root of a directory section with three subsidiary pages. The HTML file for this page must specify the labels for the navigation bar and the directory names of the subsidiary pages, as follows:

```
<slide
  pageref="<code>TableLayout</code>:table"
  pageref="Calendar example:calendar">
```

Here, each of the `pageref` attributes indicates a subsidiary page that descends from the page containing the `pageref` entries. The portion of the `pageref` attribute that precedes the colon specifies the tag to be included in the navigation frame. Note that this section of the entry can contain embedded HTML tags, such as the ones in this example that set tokens like `TableLayout` in code font. The portion of the `pageref` attribute that follows the colon represents the name of a subdirectory that includes the slides for this subsidiary page. These subsidiary pages may further subdivided into lower level pages to any level of nesting.

Thus, to create a tutorial of your own, all you need to do is

1. Create HTML pages for the text frames of any pages you want to include in your tutorial. To do so, you can either write your own HTML files or use page-creation tools to assist in that process. For example, popular applications like Microsoft Word™ allow you to save files as HTML, which makes it particularly easy for web novices to create these pages.
2. Arrange these HTML pages into a directory hierarchy that is appropriate for the logic and flow of the material.
3. Edit the HTML files so that they include the `slide` tags that provide the metainformation needed to create the tutorial, as illustrated by the earlier examples.
4. Run the `CreateTutorial` application to create the complete set of HTML files.

5. CREATING LECTURE DEMO APPLETS

Although being able to create online tutorials will presumably be useful to some instructors, that capability alone may not prove sufficient to entice a large number of people to use this system. The HTML-based approach faces two significant impediments:

1. Online tutorials are not as common as lecture demonstrations. Almost all faculty teaching introductory computer science courses give lectures and can benefit enormously from having interactive demonstration programs to accompany the lecture presentation. Creating online reference material is in some sense "extra."
2. Creating a lecture demonstration is easier than writing a web tutorial, primarily because the tools are more effective. Even with web-authoring tools, designing a web page tends to require more effort than creating a presentation using Microsoft PowerPoint™. In addition, the number of instructors who can use PowerPoint effectively is far larger than the number who feel comfortable with web-authoring tools. In the modern university, PowerPoint is an essential tool that more or less everyone learns.

Unfortunately, however, PowerPoint doesn't really fit the bill for illustrating Java programs. Although it is a fine tool for creating slides that move the viewer through a sequence of concepts, it lacks the ability to run Java programs within a presentation. And although third-party plugins exist to support coordinated displays of HTML pages and PowerPoint slides, those plugins are difficult to use effectively and do not run on all of the major platforms.

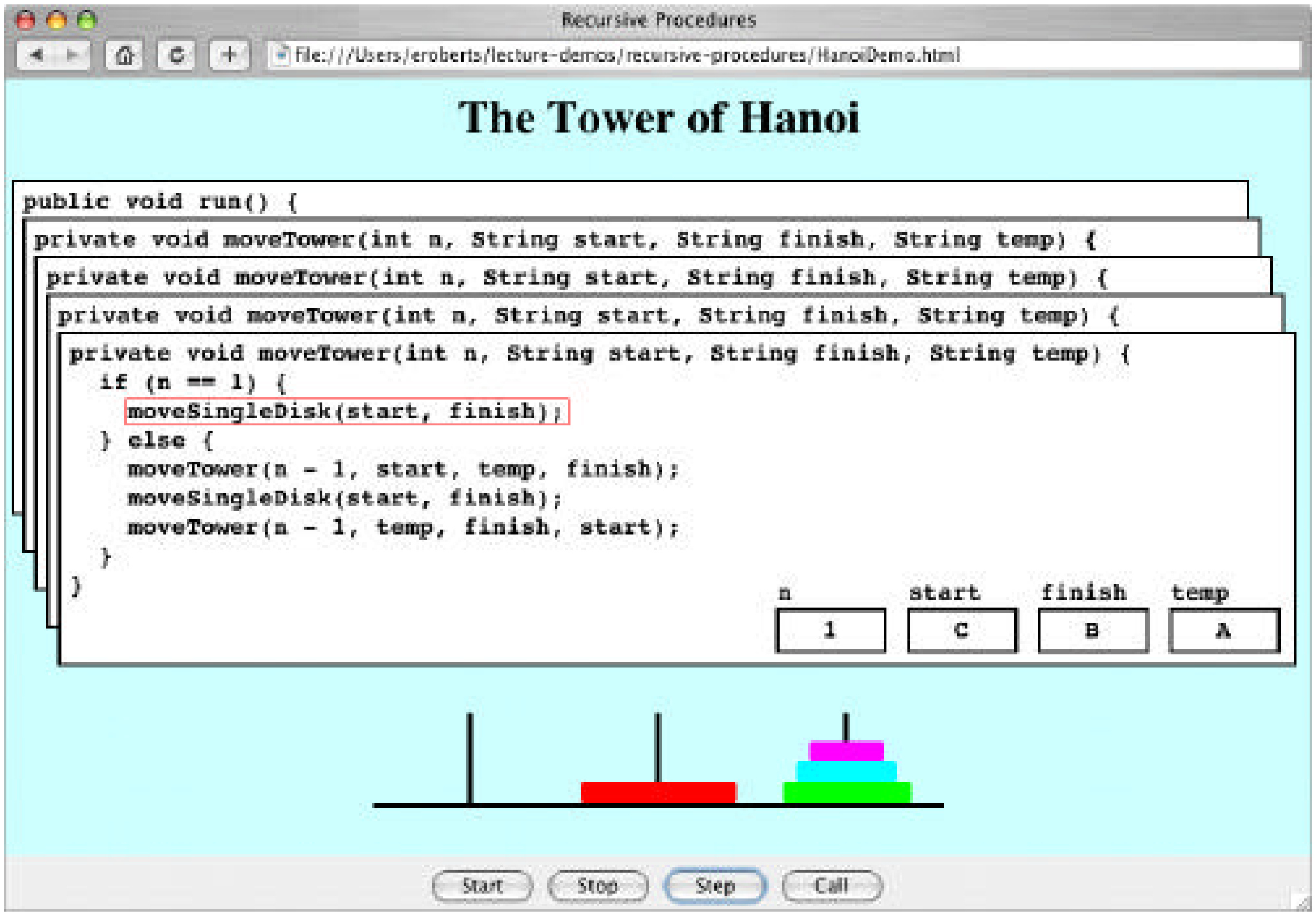
After we finished implementing the web-based tutorial system, the natural question that arose was whether it would be possible to extend its functionality to support the creation of lecture slides as well. Java itself is certainly more than capable of supporting such a presentation system, but the fundamental challenge is finding a way to leverage existing PowerPoint expertise so that users can create such displays without learning a new authoring paradigm.

In seeking to determine whether such a system was feasible, we initially considered—and then abandoned—several potential strategies:

- *Capturing the operation of an applet as a succession of images or digital video.* In many ways, the simplest approach to support displaying a program in a PowerPoint presentation is to capture the succession of images as the program is run and then to drop that set of images into a standard PowerPoint presentation. The downside of this strategy is that the demonstration is entirely "canned" in the sense that it is impossible for the user to control the operation of the demo in the way that running an actual program would permit. For certain styles of program animation, this loss of user control greatly diminishes the effectiveness of the demonstration.
- *Converting the PowerPoint presentation to HTML and then adding in the appropriate applet tags.* The fact that modern implementations of PowerPoint already include an option to save a presentation as HTML suggests that it might be possible to convert an existing PowerPoint presentation to HTML form and then postprocess the HTML file to add any demonstration applets the author wished to include. This strategy, however, turns out to be unworkable. When PowerPoint converts a presentation to HTML, it does so by generating screen images of each slide and then creating an HTML file that simply loads each image as a separate page. Such a mechanism clearly offers no flexibility to edit the individual components of a slide.
- *Converting each slide in the PowerPoint presentation to a web page by parsing the PowerPoint data file and creating an HTML page that duplicates its effect.* Although this strategy initially seems to offer considerably more flexibility than the preceding one, it fails because PowerPoint and HTML have entirely different models for how information is displayed on the screen. The most obvious difference is that PowerPoint allows the user to control the position of each structure on the screen; web browsers, by contrast, lays out the entire presentation dynamically by interpreting the HTML. Equally important, however, is the fact that HTML offers no mechanism for implementing even the simplest PowerPoint effects, such as making new items appear on the slide each time the mouse is pressed. Being able to duplicate at least the ability to sequence through a slide is essential if users are to accept such a mechanism.

What we ended up doing deciding to do instead was to convert the entire PowerPoint presentation to an applet. Figure 3 illustrates how the PowerPoint-to-applet mechanism appears when it is running in a browser. The title on the page was simply the title on

Figure 3. Lecture demonstration page with an embedded applet


Recursive Procedures
file:///Users/eroberts/lecture-demos/recursive-procedures/HanoiDemo.html
The Tower of Hanoi
Start
Stop
Step
Call

```
public void run() {
private void moveTower(int n, String start, String finish, String temp) {
private void moveTower(int n, String start, String finish, String temp) {
private void moveTower(int n, String start, String finish, String temp) {
private void moveTower(int n, String start, String finish, String temp) {
  if (n == 1) {
    moveSingleDisk(start, finish);
  } else {
    moveTower(n - 1, start, temp, finish);
    moveSingleDisk(start, finish);
    moveTower(n - 1, temp, finish, start);
  }
}
```

n	start	finish	temp
1	C	B	A

a PowerPoint slide, and the entire rest of the page is simply the **`HanoiDemo`** applet running with the same background color.

The first step in creating the applet version of the PowerPoint presentation is to parse the binary data file to create a data structure that drives a more general presentation applet. That system is currently under development. Its success is not dependent on implementing the full range of PowerPoint effects, since most of the available effects are not actually used (or are certainly not essential) in an effective presentation. The prototype implementation interprets only text boxes, bullet lists, and images, limiting the available effects to having an item appear when the mouse is clicked. Later versions will support a wider range of PowerPoint features, most notably the ability to group items together to form a single unit.

The key additional feature is that the PowerPoint-to-applet translation system looks for a particular kind of labeled box in the presentation and converts that to an applet insertion, using the boundary of the box to indicate the location of the applet and the label to identify which applet to run. Those embedded applets are run in the context of the larger applet that implements the PowerPoint presentation as a whole. In the presentation shown in Figure 3, for example, the entire slide reason below the title was taken up by an applet box whose label indicated that the region was to be replace by the **`HanoiDemo`** applet.

6. CONCLUSION

The JTF Tutorial system allows potential users of the Task Force packages to learn the details of those packages in an environment that offers a high degree of interactivity. That system, moreover, can be easily tailored to support custom tutorial packages developed by individual instructors.

In addition to web pages, the JTF Tutorial system is currently being extended to support the creation of lecture demonstration applets from simple PowerPoint presentations. Initial experiments with this approach show significant promise, although it remains to be seen how effective this strategy will prove in practice.

REFERENCES

1. ACM Java Task Force. Project rationale. Second public draft, January 2006.
2. Lewis Barnett, Joseph F. Kent, Justin Casp, and David Green. Design and implementation of an interactive tutorial framework. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA, February 1998.
3. Joseph Bergin, Thomas L. Naps, et al. Java resources for computer science instruction. Report of the ITiCSE'98 Working Group on Curricular Opportunities of Java Based Software Development, SIGCSE Bulletin, September 1997.
4. Christopher Boroni, Frances Goosey, Michael Grinder, and Rockford Ross. A paradigm shift! The Internet, the web, browsers, Java and the future of computer science education. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA, February 1998.
5. David Cole, Roger Wainwright, and Dale Schoenefeld. Using Java to develop Web based tutorials. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA, February 1998.
6. Herbert Dershem and Peter Brummund. Tools for Web-based sorting algorithms. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA, February 1998.
7. Thomas Naps, et al. Using the WWW as the delivery mechanism for interactive, visualization-based instructional modules. Report of the ITiCSE'97 Working Group on Visualization, SIGCSE Bulletin, September 1997.
8. Eric Roberts. Resources to support the use of Java in introductory computer science. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, VA, March 2004.
9. Susan H. Rodger and Willard C. Pierson. Web-based animation of data structures using JAWAA. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA, February 1998.