# Java Enabled Opto-Electronic Learning Tools and A Supporting Framework

**Pratibha Gopalam, Alexander N. Cartwright,**
**Electrical Engineering**
**Bina Ramamurthy,**
**Computer Science and Engineering**
**University at Buffalo, State University of New York**

## Abstract

The use of multimedia tools over the World Wide Web is an extremely desirable instructional method. Unintentionally this has created a maze of online tutorials and demonstrations with huge amounts of information in disarray. In addition, Cognitive theories, like Active Learning and Experiential Learning, applicable to the engineering domain, sermonize modeling, problem resolution and problem visualization as the key elements in instruction. In this paper, we present some of our work on building user configurable Java Applets for education in photonics (lasers and optics). These include design Applets for laser principles that illustrate optical ray tracing systems, population inversion, and 3-D visualization of optical polarization.

This paper focuses on the design and implementation of user-controlled context based educational resources. A design-based learning experience using Java Applets and the multiple facets of the design and development of such a software system is described. Specifically, an initial object-oriented framework that emphasizes key elements of design like reusability, flexibility, modularity and extensibility is provided. This framework is developed for the design of educational Java Applets that provide user configurable simulation environments. The six key elements of the framework are: Components, Strategy objects, Singleton objects, Visitor objects, Toolkits and Containers. Moreover, methods to initialize Applets using design windows and HTML tags are presented. This framework provides basic guidelines for developing user-configurable Java simulation environments for use in any science or engineering field.

## Introduction

Research in engineering education has clearly identified the need for improving teaching styles to match the vast spectrum of learning styles [1,2]. R. M. Felder and L. K. Silverman, in their paper "*Learning and Teaching Styles in Engineering Education*" [1], provide an excellent discussion of learning and teaching styles. Felder clearly describes how specific teaching styles can be adopted to address various learning styles found among students [2]. These variations in learning are classified as 32 different possible learning styles. For clarity, we have repeated the summary table from pg. 675 of Felder's paper [1] in Table 1. It is important to realize that people do not neatly fall into any particular learning style but that there is a continuum of learning styles. For example, it is possible for both visual and verbal teaching to be equally effective for a

particular student. With these various learning and teaching styles identified, any instructional material structured to address all these learning styles would naturally prove very effective.

Today, there is additional pressure to present material in a dynamic visual appealing manner. Students, when taken collectively, expect a stimulating learning environment similar to that provided by television, video and computer games to which they have grown accustomed. Java Applets can be used as supplementary instructional material in traditional lecture style courses to allow instructors to present educational material in a more visually appealing manner. In this way, Java Applets allow for the incorporation of teaching styles not normally practiced.

**Relationship between Learning Styles and Java Applets**

In the language of Felder, the traditional lecture style of teaching (abstract/ verbal/ deductive/ sequential) incidentally addresses only the intuitive/deductive/reflective/sequential learning styles [Table 1]. Invariably, lecture style teaching is a mismatch with students that have other learning styles. To increase the impact of teaching on students, lecture style teaching should be coupled with active student participation with live demonstrations, and practical laboratory exercises with lots of scope for experimentation and reflective observation. Educational Java Applets can be used to introduce context-based case studies and encourage experimentation, which would appeal to the sensory/visual/inductive/active/global learners. Hence, we believe that a teaching technique using Java Applets in conjunction with traditional lectures would approach the ideal teaching style.

**Table 1: Dimensions of Learning and Teaching Styles (from Felder and Silverman (1988)). . The learning and teaching styles with an * benefit tremendously from Java Applets.**

| *Preferred Learning Style* | | *Corresponding Teaching Style* | |
|---|---|---|---|
| **Sensory*** | Perception | **Concrete*** | Content |
| Intuitive | | Abstract | |
| **Visual*** | Input | **Visual*** | Presentation |
| Auditory[3] | | Verbal | |
| **Inductive*** | Organization | **Inductive*** | Organization |
| Deductive | | Deductive | |
| **Active*** | Processing | **Active*** | Student Participation |
| Reflective | | Passive | |
| Sequential | Understanding | Sequential | Perspective |
| **Global*** | | **Global*** | |

Educational Java Applets, in their simplest form, can provide wonderful demonstrations and concrete examples of underlying principles and concepts. They help sensory learners that have a strong desire to see things working to understand the theory. They also provide an excellent mechanism for presenting concepts and principles using pictures, block diagrams and vivid simulation schematics, to favor visual learners. The Applet's graphical user interface allows the user to interactively change the behavior of the Applet and hence appeal to active learners. More importantly, by allowing learners to dynamically set parameter values and watch varying results, Applets appeal to the inductive style of learning. In addition, Applets can appeal to both global

and sequential learners by providing an overview of large-scale systems with links to detailed demonstrations of specific underlying fundamental properties [4]. Therefore, Applets provide a powerful means of supplementing existing classroom instruction to address learners that prefer sensory, visual, inductive, or global learning. Specifically, adding educational Applets to our contemporary lecture style curriculum will provide an adept teaching style that could match the learning styles of most students and provide each student with some educational experience that they enjoy.

Applets that demonstrate a specific concept are extremely useful for specific domains. For example, Chu R. Wie has compiled a number of excellent semiconductor educational Applets [5] and the associated framework for this type of Applet [6]. Here, we would like to extend the functionality of Applets to allow the users to design and simulate experiments and systems (user-configurable virtual laboratory Applets). Moreover, additional groups have realized the importance of developing user configurable simulation environments [7, 8].

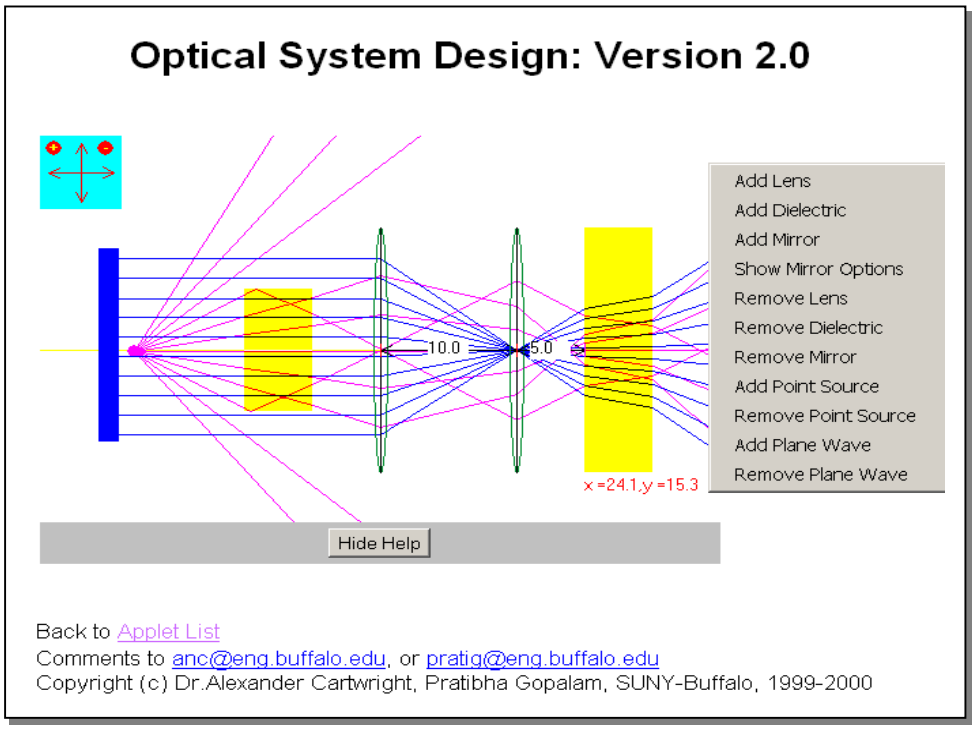


**Figure 1: Optical Design Applet showing graphical representation of optical components and sources. This applet is a design environment where students can add, remove, and modify optical components and observe the resulting ray tracing.**

## Example of User Configurable Design Applet: Optical Design System

Figure 1 shows a snapshot of our Optical Design Applet where all the optical components (lenses, mirrors, dielectrics) in the optical system are graphically represented. This Optical Design Applet engages learners in active experimentation because it lets users add and remove components like lenses, mirrors, dielectrics and light sources. It also lets users change various parameters of the components such as the focal length of the lenses, the radius of curvature of the mirrors, the location of these components, and the type of light sources. This simple user

configurable design applet allows a student to experiment with various configurations of lenses, mirrors, and sources before and after conducting optical experiments in a laboratory. In this way, students get a feeling for what to expect in laboratory, have a method for reproducing what they saw in laboratory, and appreciate how theory approximates actual conditions in a laboratory. Applets that give users total control for running virtual experiments provide a very conducive environment for encouraging and developing the questioning approach to understanding that is very important in engineering.

**Object Oriented Programming and Frameworks**

Programming in object-oriented languages, like Java, is often referred to as a means of making software systems modular, extensible and reusable. This is not the whole truth; object oriented languages merely help incarnate these concepts by supporting implementation features like inheritance, polymorphism, abstract classes and interfaces. Furthermore, object oriented languages are only tools, not a panacea for all distresses in software engineering. Programming software systems in an object-oriented language by no means guarantees them to be reusable and scalable. Simplicity, reusable abstraction and systematic organization together form the essence of object oriented software. These have never been easy to attain and continue to evade even experienced object oriented designers. New designers baffled by the degree of insight needed to get a design "right", tend to get frustrated with the amount of redesign and rework involved in developing robust and scalable software systems. Experience teaches designers not to solve every problem from first principles, but to reuse solutions that have worked in the past.

During the initial stages of our efforts at developing Java educational Applets, an entire new applet was developed from scratch for each new topic. This resulted in an enormous amount of time being wasted. In order to increase productivity, we started to directly adopt designs, protocols and implementations from our previous work. This was possible because of some good generic design strategies that were adopted in earlier Applet designs. In software engineering a number of design techniques have been identified to solve specific design problems and make object-oriented designs flexible, elegant and ultimately reusable. A designer who is familiar with such patterns can apply them immediately to design problems without having to rediscover them. These experiences have been recorded as Design Patterns [9]. Furthermore, we realized that the distinctive set of solutions (explained in detail later) that emerged in the course of our work followed some existing design patterns.

The collective use of these design patterns for developing applications takes us to the next level of organization; Frameworks [10]. A framework is an object oriented reuse technique that serves as the skeleton of an application. Frameworks have built-in flexibility and can be customized by an application developer to meet his/her needs. In other words, a framework is a reusable "semi-complete" application that can be specialized to produce custom applications [9]. Most importantly, frameworks help capture the design decisions and experiences that are necessary within domains for which the applications are being developed, engineering education being the domain of interest here. Once defined, frameworks help in the rapid development of similar applications with minimal effort. The benefits of following such an approach and developing a Framework have been well documented [11].

Our present understanding of teaching styles and awareness of design patterns and frameworks for software development allows us to set forth a suitable framework for developing user configurable educational tools. This framework will ensure that any user configurable educational tool developed will invariably adhere to guidelines identified for effective teaching styles. This framework is the compilation of the collective experiences from the fields of education and software engineering and will allow for easy development of powerful teaching aids in engineering education.

**Framework for Developing User Configurable Virtual Laboratory Applets**

Developing user-configurable virtual laboratory Applets has provided us much of the necessary experience to identify the key elements for this framework. This being only an initial definition of the framework, it is imperative that it is enhanced as a result of new observations during the course of future work and as the result of revisions from other educators using this framework. Figure 2 shows a block diagram of the framework that shows the various interactions between the elements. The six key elements we have identified are 1) Components that encapsulate data, 2) Strategy objects [9] to define interactions between Components, 3) Singleton objects [9] to define rules for the Components, 4) Visitor objects [9] to probe and change the status of Components, 5) a Toolkit [9] to perform routine tasks and 6) a Container to hold all these co-operating and interdependent elements. When using this framework to develop educational Applets, these elements should be developed through discussions between subject experts (instructors/professors) and the programmers implementing the specific Applet (professors/students/programmers).

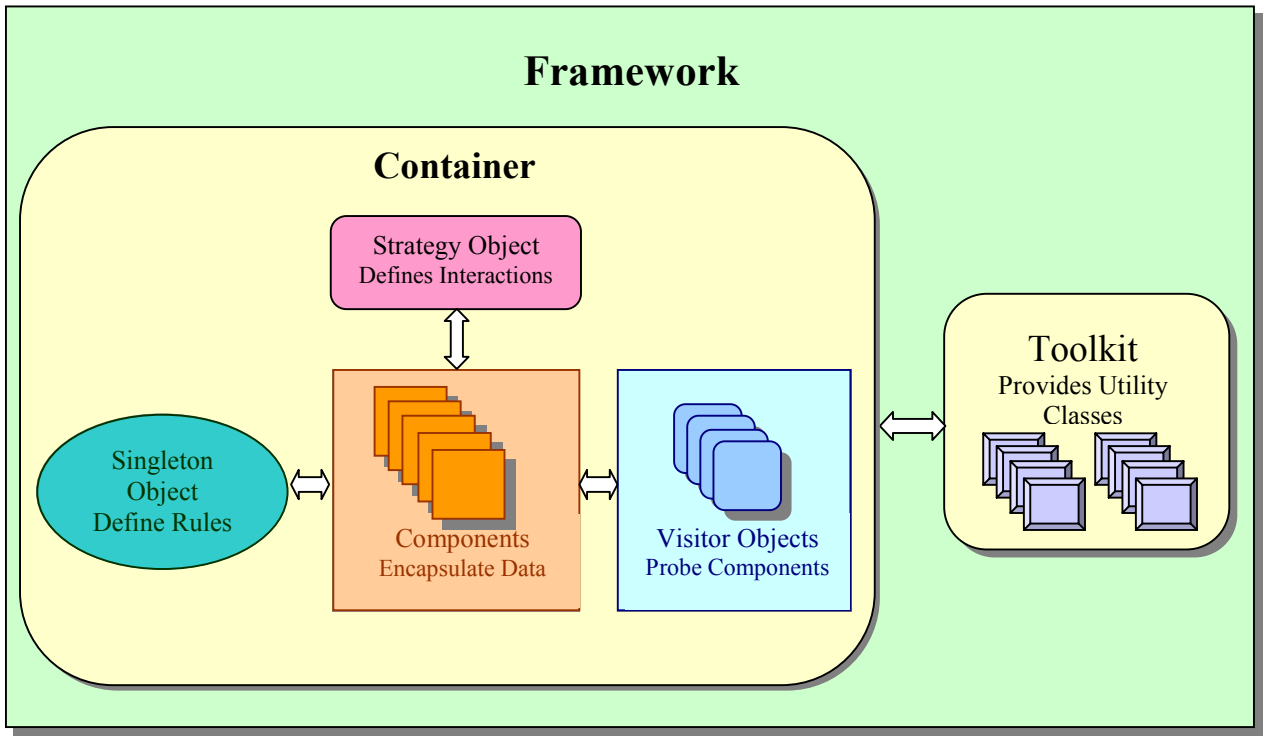


**Figure 2: User Configurable Virtual Laboratory Framework**

In the discussion that follows, we will assume some familiarity with object-oriented techniques, software design patterns, and technical terms from software engineering. We encourage readers

not familiar with these topics to refer to the appropriate references [9, 10, 11] for additional information. The essential elements for the framework for a user configurable design environment are:

1. **Components:** Components are simple elements that encapsulate the data associated with the object being modeled and typically have a graphical interface. This data can be accessed and manipulated programmatically through interface methods. The user interaction tools (pop-up menus, property lists and dialog boxes) use these interface methods to probe, change and monitor the data. There are two types of components: active and passive. Active components are components that are changed by interaction with other components (e.g., light sources). Passive components are those components that remain unchanged but can change other components behaviors (e.g., lenses). Students or programmers can develop components for the framework. These developers do not need to know everything about the system. For educators, this is where undergraduate students can effectively contribute to the development of educational Applet resources. As long as the instructor/lead person provides detailed specifications, it is relatively straightforward to develop these components.
2. **Strategy Objects**: These objects handle all the processing, the mathematical calculations, and the implementation for algorithms used in the design. They define the governing principles for the applications being developed, e.g., ray tracing matrix manipulation or solutions to rate equations. In other words, these objects define component-to-component interaction. These objects can in turn use the toolkit utilities (graphs, lines, data values, etc.) for presenting the results to the user. Strategy objects are quite complex and might require much more work from an implementation viewpoint. However, these are objects that are developed once for the lifetime of the application. They are essential to the framework, and the design of these objects requires much attention. Experienced software team members (programmers) are needed to develop these objects.
3. **Singleton Objects**: These are key elements that define the rules that uniformly apply to all components in a given scope. These provide global access to the information required by components conforming to these rules. All complying components have a reference to the Singleton object or are registered with this Singleton object. Therefore, any change in this Singleton object gets reflected in all components associated with that Singleton object. These objects can be used to define uniformity in distance, pressure, temperature, or color. Singleton objects also need to be developed by experienced software developers.
4. **Visitor Objects**: These are generic objects provided in the framework that can be called upon by the component objects to allow access to their data. These visitor objects provide a mechanism for accessing and changing the component's data. Thus, visitor objects allow dynamic run-time configuration of components in the applet. Furthermore, these objects can monitor and use the toolkit classes to plot changes in the components data. Generally, these visitor objects employ some graphical interface object, e.g. pop-up windows and dialog boxes, to set, get and display component specific data. Experienced program developers should develop visitor objects.
5. **Toolkit**: The toolkit is a set of related and reusable library classes designed to provide general-purpose functionality. It is comprised of utility items like graphs, calculators, integration routines, etc. In addition, users of the framework are free to create any

additional tools that are essential for their subject specific needs. In this manner the toolkit will continue to evolve over time. The choice of the developer of these tools depends on the level of complexity involved in the particular tool. In some cases, it might be sufficient to provide students with detailed specifications of a desired tool and allow them to develop it. However, experienced developers should be used for complex tools.

6. **Container:** A container integrates all the constituent elements and makes them aware of each other. In order for this to be accomplished, all contributing objects have to be registered with the container. The container defines how these objects collectively cooperate in any application. In addition, the container provides the placeholders for all participating objects in an application. A person well versed in the subject area of the educational applet (e.g., in our case photonics) should be responsible for the design the container.

The framework moves the responsibility, for the application's behavior and operation, away from the user of the framework into the framework. This is referred to as inversion of control [11]. The developer of any one key element does not need to have complete knowledge of how the total framework works. Elements, developed individually, are not required to actively react to any event in the framework. It becomes the responsibility of the framework to dispatch events to the appropriate elements and call upon them to react by invoking "hook methods"[11] on them. The framework achieves this by requiring that the user of the framework conform to standard protocols by implementing specific interfaces or abstract classes.

The framework requires that Applets provide design windows for specifying the initial system setup and some means of passing the values of the parameters associated with the components. In the present implementation of the framework, parameter passing is accomplished through the HTML APPLET tag. This scripting ability, through HTML, allows for statically configuring the context of the system or experiment (the user of the applet is not allowed to modify the system layout). From an educational viewpoint, this allows instructors to freeze the educational content and still allow students running the applet to use the interactive tools provided to dynamically change values and parameters.

**Example Implementation of the Framework**

It is important to realize that, to date, we have only developed a few systems that follow the guidelines in this framework. More importantly, the implementation of the entire framework still requires refinement before it can be reused by other educators that would like to develop similar user configurable virtual laboratory environments. However, with this preliminary definition of the framework, we are constantly working to develop a generic set of objects, non-subject specific, which can be readily dispersed through the WWW.

As an example of the implementation of the Framework we will explain the design and implementation of the Optical Design System. Figure 3 shows an example of the Optical Design System where users can watch optical rays tracing through various optical components. Users are allowed to add, delete and modify optical components in the system. To aid in the understanding of this prototype implementation of our framework, we will follow the same numbering scheme previously used above in the definition of the framework. Moreover, we have inserted a number

of labels in Figure 1 to clearly show the various elements of the framework. Names in italics in are the actual names used in the coding of the Optical Design System.

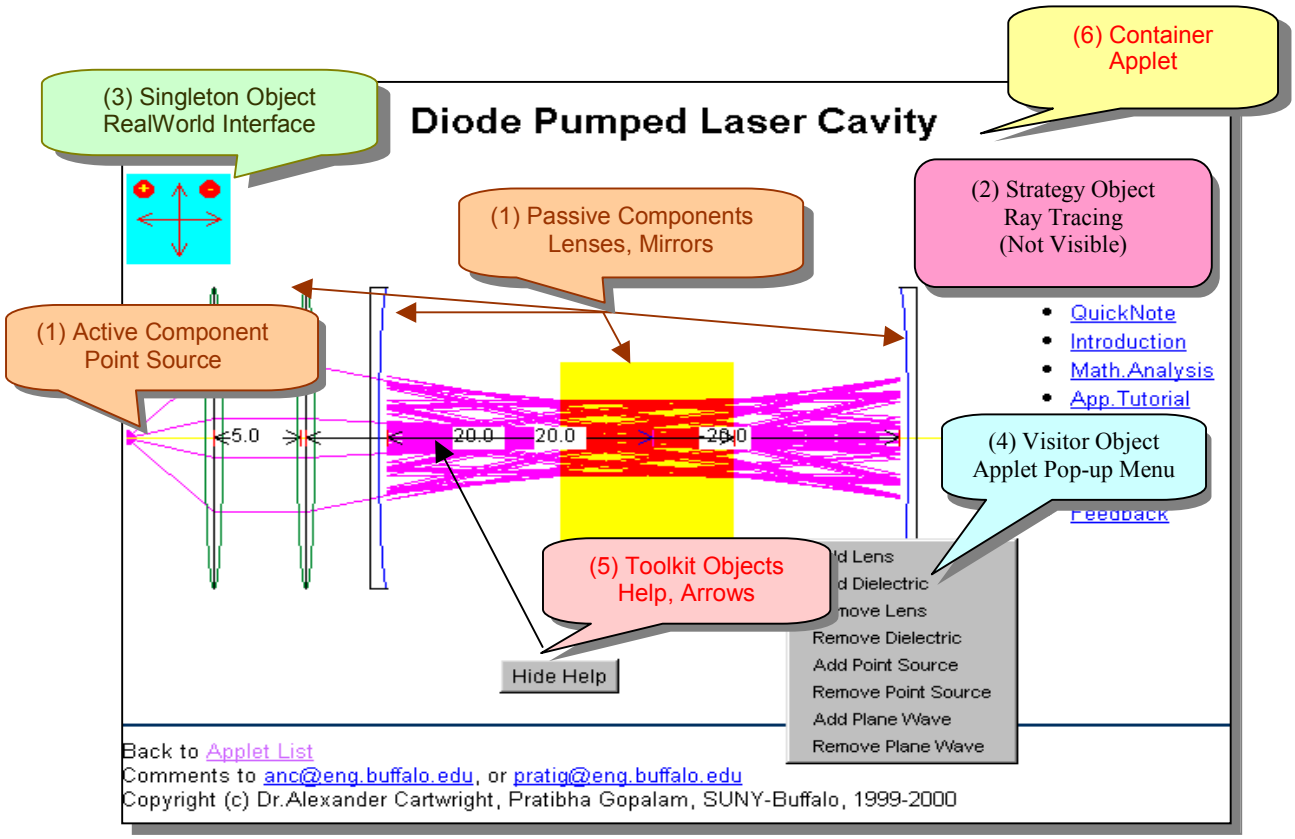


**Figure 3: An example configuration of the Optical Design Applet. This Applet demonstrates a diode pumped laser system using ray tracing. This applet was generated with HTML tags. Examples of the various elements of the framework are labeled.**

Specifically, the Optical Design system is made up of the following elements:

1. **Components**: The components in this applet are lenses, mirrors, and rays (which make up the sources). These components encapsulate data and provide interface methods for changing their properties. Every optical component is composed of an *ABCDObject* and an *OpticalInterface* object. In order to participate in container-to-component communication, every optical component implements the *OpticalComponent* interface. Furthermore, these components are responsible for updating their internal *ABCObject* as a result of user interactions. Ideally the responsibility to provide access to the component's data should be delegated to the Visitor objects provided in the framework. This provides a standard means for the container (applet) to interact with the components.
2. **Strategy Objects**: At the present time, we have incorporated the algorithm for ray tracing within the ray object. Ideally, the strategy for the interaction of the ray with a component should be in a separate Strategy object. Accordingly, the Strategy object is a separate entity in the Polarization of Light applet shown in Figure 4 [12].

3. **Singleton Objects**: The *RealWorld* object gives us a standard set of units for measuring distance (e.g., cms) in this applet. Changing the *RealWorld* object is accomplished by the *RealWorld* graphical interface (shown in the upper left hand corner of Figure 1 and Figure 3). The + and – locations on the *RealWorld* graphical interface allow users to zoom in and zoom out by scaling the pixel to distance variables. When this happens all objects in the applet repaint themselves using this updated coordinate system information. Moreover, the arrows on the graphical interface allow the user to scroll the applet in various directions, creating an expanded virtual layout region. The *RealWorld* object is fairly generic and has been reused in the polarization applet to represent units for distance measurement, and in an Energy Systems Applet to represent energy of a particular level (in electron volts). At the present time, every component in the Optical Design Applet uses a reference to the *RealWorld* object to update itself. However, a more recent implementation of this *RealWorld* object is using messaging and registration of components. In other words, a component registers itself with the *RealWorld* object. All registered components are notified of any changes in that *RealWorld* object by generated events. In this way, when the *RealWorld* is changed all of its corresponding components are made aware of the change. This causes an inversion of control where the behavior of the component is controlled by changes to this Singleton object. We have been made aware of another Java educational resource that provides similar functionality for treating temperature and pressure [7].
4. **Visitor objects**: In the Optical Design Applet there are Visitor objects that probe components to get and set specific state information. As shown in Figure 3, the addition and deletion of components etc., is handled by a pop-up menu object, which is a Visitor Object to the Applet. The properties of lenses and mirrors in the applet can also be modified using property list boxes that pop up when the user right clicks on the components.
5. **Toolkit**: We have developed a number of tools to provide for graphical user interfaces and complex mathematical computations (integration, 3D visualization, and graphing). Moreover, we have incorporated some excellent tools from NETLIB [13], and from the Ptolemy project (*Ptplot*) [14]. The ability to use existing resources, like those provided by the toolkit, is essential for ease of programming.
6. **Container**: The applet itself is the container in our Optical Design system. It recognizes lenses, mirrors, dielectric media and ray sources present in the system. It also knows how these optical sources interact with components and manages additions and deletions of components to the system.

Finally, these Applets have the additional benefit of being able to use HTML parameter tags with the applet to pass initial configuration information. This allows for developing various models by using a simple scripting language. Figure 4 shows the polarization applet simulating an optical intensity modulator. The example HTML code (bottom of figure) uses the *param* fields within an <applet></applet> HTML tag to initialize the applet.

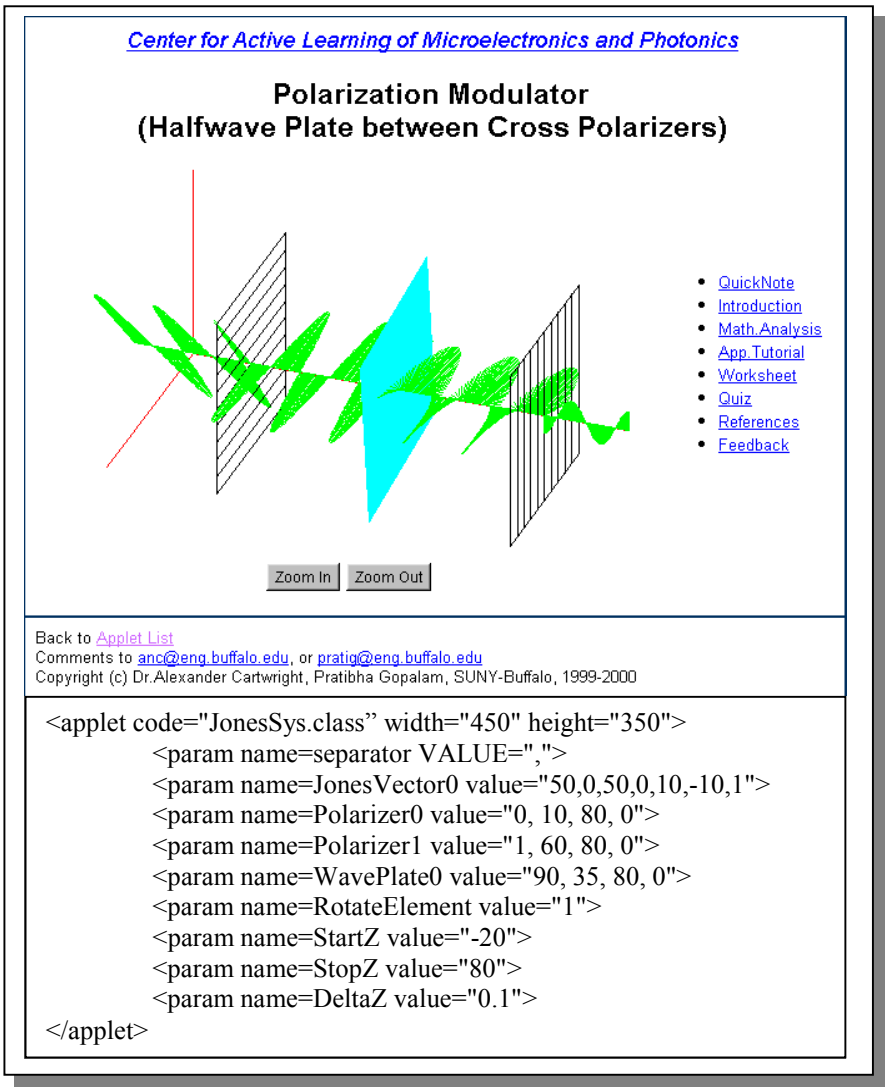


**Figure 4: The polarization of light Applet simulating a half-wave plate polarizer modulator is shown at the top of this figure. The HTML tags used to generate this applet are shown at the bottom. Note that the parameters passed to the applet determine the necessary data values for the components.**

## Conclusions and Future Directions

It is essential to continue this work to develop a generic, portable, set of objects for the proposed framework for use by other educators. This would be valuable to other educators interested in developing similar virtual design Applets and laboratory applications. Moreover, any Java educational resources developed using this framework can be readily incorporated into existing lecture style courses throughout the country. This framework helps in identifying generic patterns and techniques that are necessary in the development of any application independent of the targeted academic discipline. We are planning to adopt the software component technology using JavaBeans™ to further enhance the developed framework. Finally, this paper represents our preliminary attempt to standardize the development of user configurable virtual laboratory environments to serve as supplementary educational resources for various science and

engineering subjects. As such, we firmly believe that this framework will definitely evolve more rapidly as other groups contribute to this effort by attempting to use it.

## Acknowledgments

The authors would like to acknowledge the financial support of the National Science Foundation, Award #9950794 and Cartwright's National Science Foundation CAREER Award #9733720. We would also like to acknowledge the contributions of Xin Hu, Derek Hoiem and Philip Manijak in the development of some educational Applets and resources used in this work. Finally, we would like to acknowledge Chu R. Wie for his advice on the development of Java Applets.

## Bibliography

1. Felder, R. M., Silverman, L. K., "Learning and Teaching Styles in Engineering Education," *Engineering Education*, *78*(**7**), 674-681, April 1988.

2. R.M. Felder, "Reaching the Second Tier: Learning and Teaching Styles in College Science Education," *J. College Science Teaching, 23*(5), 286-290 (1993).

3. URL: http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Learning_Styles.html; R. M. Felder, Auditory was subsequently changed to Verbal.

4. Alexander N. Cartwright, Pratibha Gopalam, N. Liu, Z. Yuan, T. Tang and Chu R.Wie, "Context Based Educational Java Applets Using Consumer Products," American Society for Engineering Education Annual Conference, Session 2632, June 18-21(2000).

5. URL: http://jas2.eng.buffalo.edu; Java Semiconductor Resource, C. R. Wie.

6. Zhiyong Yuan, "Design, Implementation and Application of Framework in Java Educational Applets," Masters Thesis, University at Buffalo, Feb. 2001.

7. Bryan Mihalick, "Development of a standalone java-based molecular simulation environment", Masters Thesis, University at Buffalo, Feb. 2001.

8. URL: http://wings.buffalo.edu/eng/ce/kofke/applets/; Molecular Simulation Applets, David Kofke.

9. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns Elements of Reusable Object Oriented Design," Addison-Wesley (1995).

10. Ralph E. Johnson, "Frameworks = Components + Patterns", *Communications of the ACM*, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, Page 39, October 1997.

11. M. Fayad, D. C. Schmidt, "Object-Oriented Application Frameworks," *Communications of the ACM*, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, October 1997.

12. URL: http://www.ee.buffalo.edu/~camp/applets/photonics/JonesMatrix/modulator1.html; CAMP Website.

13. URL: http://www.netlib.org; Netlib Mathematical Tools for Java.

14. URL: http://ptolemy.eecs.berkeley.edu/java/ptplot/; Ptolemy Project

ALEXANDER N. CARTWRIGHT
Alexander N. Cartwright is an Associate Professor of Electrical Engineering at the University at Buffalo. In 1998, he received a NSF CAREER Award that supports his research on GaN based optoelectronic devices and his educational activities. In 2000, he was awarded a Department of Defense Young Investigator Award for research in piezoelectricity in III-N materials.

PRATIBHA GOPALAM
Pratibha Gopalam is a graduate student in Electrical Engineering, at the University at Buffalo. She is researching the framework development using software design patterns and JavaBeans component architecture. This framework is for expediting the creation of learning tools in Photonics using Java Applets .She received her undergraduate degree in Electronics and Communication Engineering from Bangalore University, India, in 1997. She worked as a software engineer for Hewlett Packard India Software Operations and Asea Brown Boveri Pvt. Ltd. before joining SUNY at Buffalo.

BINA RAMAMURTHY
Bina Ramamurthy is a Teaching Assistant Professor of Computer Science and Engineering at the University at Buffalo. Her research is focused on design of Java Technology based systems and Distributed Systems.