A Webware for Computer Graphics Education

Mustafa Sanver, Erik Gillespie, Li Yang

Abstract

This paper presents live and interactive webware for online learning of computer graphics concepts. A list of demos is provided. Each demo presents a concept in computer graphics by showing a 3D real world scene beside a 2D rendering scene with a list of graphics functions. Each demo allows users to interactively change the parameters and the order of execution of these graphics functions. Changing the parameters of the functions will produce the 2D rendering result from the 3D real world scene. The visual effects of user interaction will be reflected immediately in the 3D real world scene and the 2D rendering result. The webware was written by using the GL4Java library that provides native OpenGL binding for Java. Nate Robin’s well-known demos were implemented. These include translation, projection, light effect, texture mapping, and so on. New demos were also developed with pedagogical considerations in mind to emphasize the differences between model transformation and view transformation. Although the webware is designed for computer graphics learning the methodology is generic and can easily be applied to other disciplines or courses that require heavy visual presentation. This webware reflects our long-term efforts to develop web-based course material to show principles and techniques in computer science in an interactive way. We did this by having the related algorithms run live in the background and allowing students to interact with them in a web browser.

Introduction

Computer technology and the Internet are rapidly evolving and changing the way people do things in many disciplines including higher education in computer science. How computer education may benefit from technology is a main concern of this paper. Many methods are proposed for effective teaching and learning. This paper will discuss the convergence of technology and computer science education. We will focus on an introductory computer graphics course. Webware created for this purpose will be presented in detail.

The Internet and the World-Wide Web have many built-in advantages that make them ideal vehicles to convey knowledge to students. Among these advantages are their multimedia capabilities, support for user interaction, physical location independence, and ubiquity of access. Most importantly, the web presents an opportunity for hands-on experience.

An increasing number of enrollments in computer science and rapid changes in computer technology indicate the great demand for distance learning and online self-learning. Compared to traditional teacher-oriented classroom learning, technology-based learning has advantages. First, it is good for slow learners and classroom-talk-syndrome students. Second, it increases productivity of students by bringing multiple sources of information to the student through the
Internet. Lastly, students can directly control the pace at which they learn.

These features of technology-based learning make the design of course content very important. In the design of online web-based course content, a powerful way of conveying knowledge to students is through visualization. In the design of courseware with visualization the process of developing effective, well-focused visualization paradigms is very important. First of all, courseware design must follow the purpose of easy learning. Second, it must be interactive in order to provide hands-on learning to students. Third, it must be self-explanatory. Fourth, different kinds of learners and different levels of difficulty should be considered. Fifth, the order of presentation must be well designed at the most basic level and extend throughout the most comprehensive aspects.

One of the key issues in turning technological instructional methods into student-centered ones is the introduction of interaction into the learning process [5]. Students should get a chance to try a demonstration. Good courseware should enable students to have their own practice in learning an algorithm or a technique by entering parameters to the algorithm, submitting inputs, or even loading their own data files into the demonstration.

One objective of this paper is to provide an innovative visualization methodology for computer science education in an interactive web-based learning environment. With the above consideration in mind, we believe that the best way to show principles and techniques in computer science is to have the related algorithms running in the background and to allow students to interact with them in a web browser. The important advantages of interactive visualization over other computer based teaching tools are that they enable the user to interact in real time with the subjects to conceptualize relations that are not apparent from a less dynamic representation. Interactive web-based teaching with rich visualization of content has great value in computer science education because of its illustrative and interactive nature, seamless integration of the subject technology into education, and instant, wide availability.

We have chosen computer graphics as an example course because of its demand for visualization and the technical challenge of 3D rendering. Specifically, we have chosen CS527 - Computer Graphics at Western Michigan University as an example course that is expected to benefit from the supplementation of web-based tutorials. Educators have given the major points of the philosophy of the first computer graphics course [10]. Some of these points are: (1) Computer graphics is inherently 3D and courses should be also; (2) Computer graphics is intrinsically visual, and even the most technically-oriented graphics practitioner must be aware of the visual effects of algorithms. Unlike algorithms in other areas of computer science, algorithms in computer graphics must be considered not only for time and memory usage, but also for their visual effects; (3) Computer graphics education should be interactive. In summary, computer graphics is a particular subject that primarily depends on geometric perception and physical interaction and every effort should be made by educators to enhance this ability.

Although, nice interactive computer graphics demos exist [6,8], most of these are written in C and cannot run directly in a web browser. Our demos can be run everywhere and anytime. Tutorial sites that teach graphics concepts also exist on the Internet, but they are designed for business presentations. They are good for narrative, general-purpose online lectures, but they are not capable of illustrating important techniques interactively in computer science courses where principles, which the design of a tutorial is based on, need to be illustrated.
For two years we have consistently encouraged students who took CS527 to create web-based interactive demos as course projects, which are supported by a WMU Teaching and Learning with Technology grant. These interactive demos are designed for undergraduates taking their first course in computer graphics. For our CS527 students, developing the tutorials gives them interesting projects in an exciting area, requiring creativity and imagination as well as knowledge and systematic thinking. To ensure the quality and consistency of results, the CS527 instructor gives students a basic idea of computer graphics concepts and encourages students to add additional features that they think are useful to learning. Such student involvement and design flexibility have led us to revise and maintain the quality and persistence of the webware over time. The result is a set of web-based interactive 3D demonstrations that show concepts and techniques in computer graphics. The demos serve as alternative sources of information and offer another mode of learning to augment the classroom lectures.

Bringing OpenGL tutorials to the web so that students can access them within a web browser anytime and anywhere is the initial purpose of these demos. Many OpenGL demos are available for example, the popular Nate Robins’ OpenGL demos [8], which are written in C. Our demos differ in several ways. First, because our demos run directly in a web browser, which is available on almost every computer, no special software is needed. Second, we believe that the best way for students to learn computer graphics and OpenGL is to try each OpenGL function interactively. We have designed our demos in such a way that functions and parameters can be interactively modified, and the visual effects will reflect the changes immediately. Third, we have designed our demos with pedagogical considerations in mind. For example, we emphasize the differences between model transformation and view transformation even though they have no differences and are kept in a single model-view transformation matrix in OpenGL. These demos are designed as supplementary material to in-class lecture. Students are instructed on which demo to try after each class to reinforce what they have learned in class. Fourth, we use Java’s components to increase the user-friendliness of the demos while maintaining the originality of the demos. Fifth, we improved the demo algorithms considerably by using OpenGL display lists so that data files of geometric objects are only downloaded once. Lastly, we created and added a new set of fonts for small sized texts. This is because at the time we developed our demos GL4Java did not support glut font functions and the commonly available raster font could not satisfy our needs.

This paper discusses the design methodology and the visualization techniques used in our demos. The demos cover following major concepts of computer graphics: transformation, projection, light effects, material effects, fog effects, and texture mapping. Each demo gives students an opportunity to interact with a set of related OpenGL functions. We also developed a combo demo with enriched functionality by increasing the number of functions available to users in one demo. This variety of demos is grounded in incremental pedagogy and takes into consideration users’ different learning styles. These demos rely on novel techniques for demonstrating computer graphics concepts, principles, and algorithms: they run changes in the background while visualizing those changes on the screen.

The Design Methodology

A major advantage of interactive computer-aided learning is that it allows a student to proceed at his/her own pace, motivated by a curiosity about what happens interactively and “the need
to know” principles. Using demos like ours lets remote learners be active. Therefore, our
design was guided by the need to hold students’ attention, to allow students’ autonomy, and
to advance their creativity \[^2\].

One effective method to attract students’ attention is to use rich and creative visualization. We
have ported the Nate Robins’ OpenGL demos \[^8\], which were originally written in C, to Java.
The result is a set of OpenGL demos that can run instantly in a web browser. Each demo is
designed to show a concept or technique together with a list of supported OpenGL functions.
Following the style of the original Nate Robins demos, we have designed each demo so that its
user interface has three major windows: (1) a 3D real world scene of geometric objects
together with lights and a camera indicating what actually happens in the 3D real world, (2) a
2D image rendered on the back plane of the camera, and (3) a list of OpenGL functions.

The 3D world scene is modeled and processed by the OpenGL functions to produce the 2D
rendered image. When a student moves an object, changes the arguments of the OpenGL
functions, or changes the order of execution of the OpenGL routines, the final image will
reflect the change instantaneously. For example, Figure 1 is a screen snapshot of the
transformation demo. All other demos have a similar interface.

The interface of each demo provides three different views: a world view, a screen view, and a
command view. The world view presents objects as they exist in the 3D real world. This view
exposes the camera/eye position (denoted by x, y, z axes) in the scene, the viewing frustum,
and the direction of the light. The screen view exhibits the 2D rendered image. The 3D
graphic models were adapted from Nate Robins’ demos. The Viewpoint Corporation \[^11\]

![Figure 1. A screenshot of the Light demo in a web browser.](image-url)
originally provided the models to him. The command view provides a list of relevant OpenGL functions with their parameters so that a user can change the parameters and see how such changes affect the visual effects on the screen. When the mouse is over an argument’s value, an explanation message will appear at the bottom of the command view. To change an argument’s value, the user needs only to click the left mouse button and drag the mouse. More importantly, the order of these OpenGL functions is also changeable. Changing the order of OpenGL functions in a program may produce totally different results. In our demos these changes are reflected immediately in both the world-space view and the screen-space view. When the user changes the size of the web browser, these demos are capable of resizing themselves automatically in accordance with the size of the web browser.

Visualization Techniques

Delivering 3D live demos in a web browser is a technical challenge because the current web browsers have no built-in support for 3D rendering. A widely used approach for developing online web-based tutorials is to use Java applets. However, Java itself doesn’t support OpenGL functions intrinsically. Instead the binding of Java to OpenGL through a Java Native Interface is required. There are on-going developments for this non-standard interface. J Sparrow [4] and JavaGL’s GL4Java [3] are two such APIs. In our demos we have chosen GL4Java due to its easy installation and wide availability. GL4Java supports most of the glut library functions even though glut’s font library functions have not been implemented as of this paper’s submission. GL4Java is available for MS Windows and Unix/Linux OSs. Our webware requires that the GL4Java plug-in be installed from GL4Java’s website [3].

The Demos
Educators have pointed out [9,10] that geometry is a major part of an introductory computer graphics course. Aside from geometry, computer graphics involves light and surfaces and algorithms to simulate their interplay. ACM/IEEE Curricula 91 [1] suggests that introductory graphics courses need to include material about coordinate systems and transformations, material about light and surface properties, and material about the distinction between the ways various algorithms present light and surfaces visually. Based on such suggestions, we have chosen to develop the following list of demos:

An Introductory "Hello World" demo. The first demo of a series, this demo is intended as an introduction to OpenGL programming. Gasket and Gasket3D are example OpenGL demos. Fig. 2 shows a screenshot of the Gasket3D demo.

Transformation. Translation, scale, and rotation are model transformation functions used to position and orient the models. This demo includes geometric transformations, such as translation, rotation, and scaling, as well as the concatenation of these transformations. The user is able to change the values of the arguments of glTranslate(), glScale(), and glRotate() functions. Understanding the order of transformations is one of the most important parts in computer graphics. The order of these functions may also be changed.

Projection. Projection determines how objects are projected onto the screen. Specifying a projection is similar to choosing a lens for a camera. Students found that projection is a difficult topic because its transformation matrix is not affine. The Projection demo shows orthogonal, oblique, and perspective projections. Figure 3 is a screenshot of this demo.
A default light is used in the scene in order to make the object viewable. The direction of the light is from the camera’s center toward its z direction. The light position changes when the camera moves. Each view has its own popup menu. Clicking with the right mouse button will trigger these menus. In this demo the user is allowed to choose from three different projection functions: glFrustum(), gluPerspective() and glOrtho(), and to change the values of their arguments. The pop-up menu in the command view is also shown in Figure 3.

**Illumination and Shading.** Interaction with the light sources is the best way to demonstrate illumination and shading. In the light source demo, the location of the light can be changed and the visual effects of the changing light position can be seen immediately. Material properties affect the deflection and absorption of light. In the light-material interaction demo, we have created several different materials to be applied to an object to reveal the material effects. The light property and the material property of an object may also be changed.

**Texture mapping.** Texture mapping is difficult for students because it involves mapping one coordinate set onto another and other issues such as resampling and aliasing. This demo shows what happens to the final image when a 2D texture is applied and its various parameters are changed. Figure 4 provides an example screenshot of the texture-mapping demo.

![Figure 4. A screenshot of the Texture Mapping demo](image-url)

**Fixing bugs in Nate Robins’ original demo.** We have also fixed bugs found in Nate Robins’ original Light Position demo. In the original demo, as indicated in Figure 5 (left), after the order of glLightfv() and gluLookAt() are swapped in such a way that glLightfv() is called first, the coordinates of the light source are in reference to the center reference point (which is marked with an e in the world-space view and whose coordinates are marked by ‘center’ in the command manipulation window). This rendering is indeed wrong because the location of the light source should reference the position of the camera (the origin of the current coordinate). The second bug in the original demo is that the light effect is not reflected in the screen view. Figure 5 has two snapshots: one from the original demo (left) and the other from our web-based demo (right). The white lines on the pictures denote the direction and position of the light. The left picture demonstrates that the light position changes with e (the center...
reference point used in gluLookAt()). It also demonstrates that the illumination on the world-space view is not reflected in the screen-space view (the screen-space view still displays the bright side of the object). These two bugs have been fixed in our demo as shown in Figure 5 (right).

Figure 5. The light position problems in the original demo (left) are fixed in our demo (right).

Combo demo. 3D geometric transformation is one of the most confusing topics for first-time students. In teaching the computer graphics course, we have found that students are often confused when they have to consider both the world coordinate and the view coordinate. Nate Robins’ demos (and also the OpenGL API) do not make a clear distinction between the two coordinates because these two transformations behave in the same way and are
mathematically described by a single model-view matrix. From a pedagogical point of view, we believe this concept is easier to understand when these two transformations are separated. It is for this reason that we have designed and developed a combo demo. Figure 6 provides a screenshot of the demo. In the world view the world coordinates are marked by x-y-z and the view coordinates are marked by u-v-n. We have made the demo so that glTranslatef(), glRotatef(), and glScalef() will manipulate the geometric object rather than the camera. glLookAt() will manipulate the camera. Again, the argument values are changeable. Through student feedback we know that our combo demo gives students a much better picture of the concepts of geometric transformations when compared to the original Nate Robins demo.

Conclusion

Web-based teaching and learning has many advantages over traditional education. Visualization techniques provide an effective way to attract the student’s attention in learning activities that are self-directed, experimental, and personalized for the autonomous learner. We have developed a set of web-based interactive computer graphics demo as supplemental material for students who take an introductory computer graphics course. Demos in the webware have realized our design objective: OpenGL demos are presented through the World Wide Web. These demos expand the capacity of online learning of the computer graphics course. They are good illustrations of what can be done with OpenGL. The informal feedback from students is positive; they indicated the demos as a useful self-learning tool. Although students primarily use the presented demos in order to reinforce the principles learned in class, these demos are also useful for distance learners. These demos represent a generic methodology in tutorial design using web-based visualization that can easily be adapted to other subjects. Lastly, the development of these demos has provided good projects for students who took this course. Future development includes adding new demos for the following concepts: hidden surface removal, area filling, light reflection from mirror, and curves and solid surfaces. There will also be modifications on the command view of current demos to allow for easier user interaction.

Readers are invited to try the OpenGL demos presented in this paper. These demos are available at http://www.cs.wmich.edu/~yang/tlt/GL4Java/.

References

MUSTAFA SANVER is a Ph.D. student in the Department of Computer Science, Western Michigan University. His interests include visualization of databases, distributed databases, and security for transactions over the Internet. He is a member of the ACM and the Honor Society of Phi Kappa Phi.

ERIK GILLESPIE received his Bachelor’s degree from Western Michigan University in December 2002. He has a keen interest in database, graphics, and web applications, but his true devotion lies in low-level programming on embedded and non-PC platforms. He is particularly fond of and is an enthusiastic hobbyist for the Game Boy Advance.

LI YANG is an associate professor in the Department of Computer Science, Western Michigan University. His research interests include knowledge discovery, information visualization, pattern recognition, and databases. He is a senior member of IEEE and a member of the ACM.