Computer Graphics Courseware

T. Ullrich and D. W. Fellner

Institut für ComputerGraphik, TU Braunschweig, Germany

Abstract

A lot of courseware tools suffer from the almost mutually exclusive goals of ease of usability on the one hand and extensibility and flexibility on the other hand. In most cases the tools are either ready-to-use applications (e.g. a virtual lab) or complex tool sets which need a long period of domain-specific adjustment.

This paper presents the courseware environment AlgoViz which primarily addresses this problem. The AlgoViz project provides a software collection which is currently focused on the visualization of fundamental computer graphics algorithms and geometric modeling concepts. The intention is to build a collection of components, that can easily be combined to new applications.

Supporting a purely visual programming paradigm, AlgoViz offers the possibility to create new demonstration applications without having to write a single line of source code. To demonstrate its potential AlgoViz comes with a variety of examples already forming a valuable computer graphics tutorial.

Categories and Subject Descriptors (according to ACM CCS): K.3.1 [Computer Uses in Education]: Collaborative learning, Distance learning K.3.2 [Computer and Information Science Education]: Computer science education

1. Introduction

E-learning has become an important internet application which enriches teachings in several aspects. It uses visualization techniques which have always been a very important factor for the understanding of complex methods and algorithms. Especially in life sciences problems often show dynamic behavior and are difficult to describe. Therefore, videos, simulations, and interactive animations assist an instructor in teaching scenarios.

As the importance of visualizations is well known, a lot of toolkits have been developed to support lecturers and students [UFK*89]. Consequently, innumerable projects deal with the problem of producing high quality visualizations and simulations in an efficient way [KHS98], [SGH*], [LS01].

But the production of interactive components, video clips and multimedia-learning materials is extremely expensive; particularly, if the material is didactically prepared for this medium. Procedures that minimize these additional costs are therefore desirable.

Besides financial considerations other aspects also have to be taken into account. Most visualization modules are designed and implemented by a scientific research group to enhance their teaching [Bro02], [DKDB01]. They are designed, implemented and published for a specific purpose and are used in a particular context [KMS*], [Min]. An exchange of this context – e.g. an adaptation to one's own courseware materials, or improvements, extensions and bug fixes – is, in general, not intended.

Furthermore, the long-term maintenance of a project is not guaranteed due to high fluctuation of employees in higher education.

2. Generic building blocks

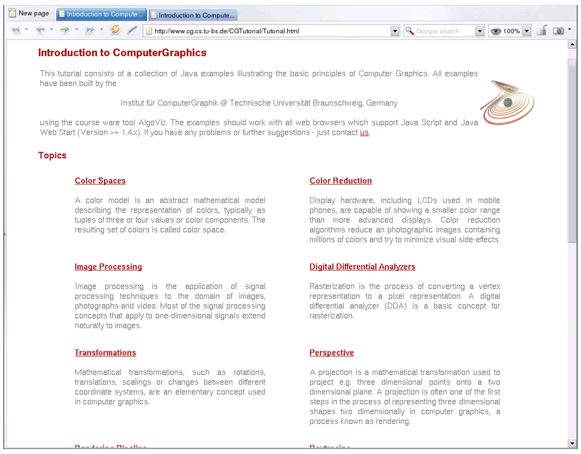
To overcome these disadvantages, we have developed a library of elementary components which can easily be combined to applications and we have also created various examples built from these library components to illustrate their usage.

The AlgoViz project provides visualization modules for fundamental computer graphics algorithms and geometric modeling concepts. The main idea is to provide a good set of small and flexible building blocks and let the user do the rest.

The project addresses two distinguished user groups: first, it supports lecturers, who want to visualize algorithms efficiently. Second, AlgoViz can be used as a framework for students within a drill-and-practice environment.

As this project focuses on high usability, not only its building block structure is in use, but also the following concepts and techniques.

- **Open Source:** The AlgoViz project will be published freely under an open source license without any costs for end users in the near future [FEJ03b], [FEJ03a].
  Based on recent feedback, we hope that the project will attract an active user group which might give – similar to other open source projects – support to new users.
- **Modularity:** All software parts are written in Java and have an object-oriented design. The granularity and the independence of the components simplify the creation of extensions and improvements [Fel91].
- **Simple Structures:** More important than modularity is a lean design with few dependencies. It allows a short period of vocational adjustment. Simple modifications and adaptations to one's own learning materials are doable with exceptionally little effort.
- **Many Examples:** To demonstrate the usage of our software components we are creating a computer graphics tutorial (see Figure 1) which will cover all fundamental computer graphics algorithms and geometric concepts needed within an introductional computer graphics lecture.


New page | Introduction to Compute... | Introduction to Compute...

http://www.cg.cs.tu-bs.de/CGTutorial/Tutorial.html

Introduction to ComputerGraphics

This tutorial consists of a collection of Java examples illustrating the basic principles of Computer Graphics. All examples have been built by the

Institut für ComputerGraphik @ Technische Universität Braunschweig, Germany

using the course ware tool AlgoViz. The examples should work with all web browsers which support Java Script and Java Web Start (Version >= 1.4.x). If you have any problems or further suggestions - just contact us.

Topics

Color Spaces

A color model is an abstract mathematical model describing the representation of colors, typically as tuples of three or four values or color components. The resulting set of colors is called color space.

Color Reduction

Display hardware, including LCDs used in mobile phones, are capable of showing a smaller color range than more advanced displays. Color reduction algorithms reduce an photographic images containing millions of colors and try to minimize visual side-effects.

Image Processing

Image processing is the application of signal processing techniques to the domain of images, photographs and video. Most of the signal processing concepts that apply to one-dimensional signals extend naturally to images.

Digital Differential Analyzers

Rasterization is the process of converting a vertex representation to a pixel representation. A digital differential analyzer (DDA) is a basic concept for rasterization.

Transformations

Mathematical transformations, such as rotations, translations, scalings or changes between different coordinate systems, are an elementary concept used in computer graphics.

Perspective

A projection is a mathematical transformation used to project e.g. three dimensional points onto a two dimensional plane. A projection is often one of the first steps in the process of representing three dimensional shapes two dimensionally in computer graphics, a process known as rendering.

Figure 1: *The CG tutorial homepage.*

- **JavaBean Concept:** Each software part of the AlgoViz project which implements a visualization component or an algorithm is written as a so-called JavaBean [Fla02]. Fulfilling the JavaBean standard, a component can be arranged with other components and can be modified on a purely visual basis. There is no need to write even a single line of code to create an application, only the elementary concepts of software design and a basic knowledge of programming paradigms are required.
  An example of such a building process without the need to code is given in the next section.

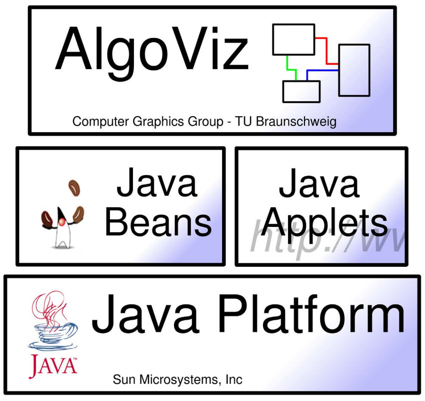


Figure 2: *Schematic AlgoViz overview.*

- **XML Serialization:** All JavaBeans automatically support XML serialization [McL01]. Applications and applets which were interpreted within a bean container and have been serialized instead of being compiled can be reloaded into the bean container for further modifications.
- **Easy To Publish:** Written in Java and designed to be used as a JavaApplet, a stand-alone application or via Java Network Launch Protocol (JNLP) as a network application all programs of the AlgoViz project can be published easily.

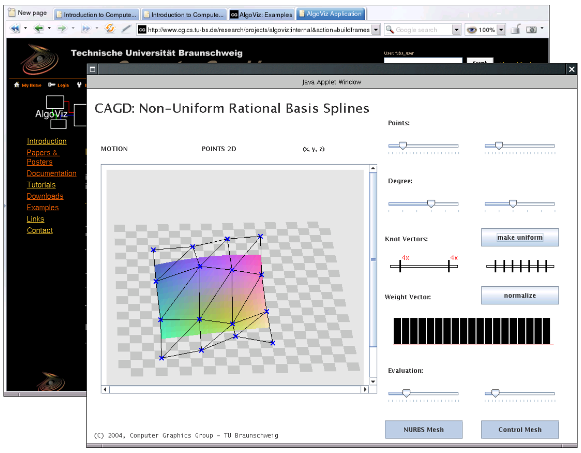


Figure 3: *An applet about CAGD created with AlgoViz.*

After this overview the AlgoViz project will now be presented in detail.

3. The AlgoViz Bean Collection

The most important part of the AlgoViz project is its collection of JavaBeans. Up to now the AlgoViz bean collection is focused on – but not restricted to – computer graphics topics.

These beans form the building blocks upon which the building block concept is based on. Besides some minor exceptions almost everything is implemented as a JavaBean. Unfortunately the term "JavaBean" is used with different meanings depending on the context. Within this paper a JavaBean is an architecture for the definition and reuse of Java software components with the primary goal to allow developers to create reusable components in a simple to use manner. Beans differ from simple class libraries in that they are intended rather for visual manipulation than for programming issues.

The spectrum of JavaBeans varies from simple GUI components (e.g. the simple label of the Java SDK called `JLabel`) to complex GUI components (e.g. the LaTeX renderer for mathematical formulas of the AlgoViz bean Collection) or non-GUI components (e.g. a mesh loader).

3.1. Creating New Applications

Using a JavaBean collection, applications can be developed in several ways. Of course, there exists the classical way of programming by writing source code. In this case the use of beans does not really differ from the use of any software library. For skilled and trained programmers classical programming will be the preferred usage.

Another possibility to use a bean collection is a bean container. A bean container is able to load up a set of JavaBeans and initialize them based on information loaded from a descriptor file. One of the first bean containers is Sun's BeanBox which has been replaced by its successor called BeanBuilder [Sun]. Other bean containers are GeoVISTAStudio [TG02] and JBeanStudio [Tak].

Common to all mentioned bean containers is the clear emphasis on visual programming. This concept realizes functional dependencies by state descriptions and data flow diagrams [CP02].

Using this technique it is possible to "write" an application without the need to write any source code. Everything is done by wiring events, event filters and data flows which can be seen in Figure 4. Therefore, the user does not need any programming skills of a specific programming language. Just a little background knowledge about object orientation, data flows and algorithms is sufficient.

Some integrated development environments (IDE) offer a way to combine both, the classical way of programming and visual programming; e.g. the Visual Editor project – a graphical plugin for Eclipse.

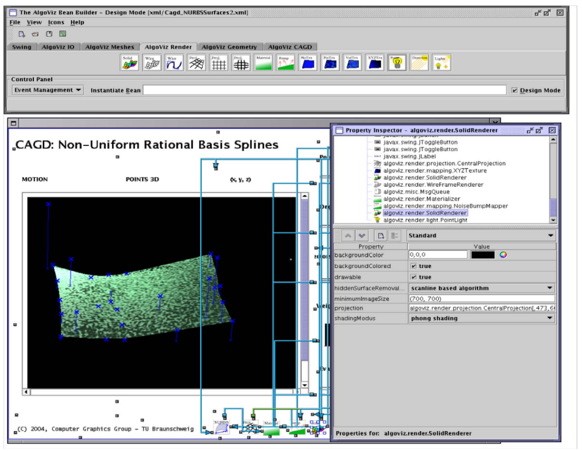

The AlgoViz Bean Builder - Design Mode [xml/Cagd_NURBSSurfaces2.xml]
File View Icons Help
Swing AlgoViz IO AlgoViz Meshes AlgoViz Render AlgoViz Geometry AlgoViz CAGD
Control Panel
Event Management
Instantiate Bean
Design Mode
CAGD: Non-Uniform Rational Basis Splines
MOTION
POINTS 3D
(C) 2004, Computer Graphics Group - TU Braunschweig
Property Inspector - algoviz.render.SolidRenderer
Standard
Property
Value
backgroundColor
0,0,0
backgroundColored
true
drawable
true
hiddenSurfaceRemoval
scanline based algorithm
minimumImageSize
(700, 700)
projection
shadingModus
phong shading
Properties for: algoviz.render.SolidRenderer

Figure 4: *A screen shot during the building process of the applet shown in figure 3.*

3.2. An Example at Full Length

To demonstrate the simplicity of visual programming the creation of a simple demonstration tool, illustrating the conversion of an RGB color into HSV representation, will be shown.

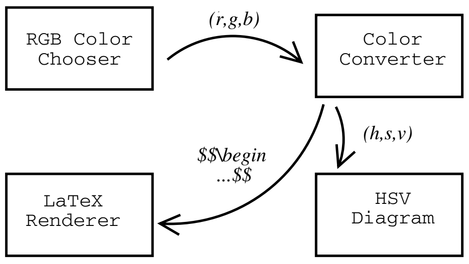


Figure 5: *The schematic structure of the example.*

According to the diagram in Figure 5 the application consists of four main components:

1. An RGB color chooser lets the user select a color in RGB representation.
2. The color converter takes an RGB triple and converts it into HSV representation according to the well-known conversion formula. In a non-teaching scenario this would be sufficient, but in our case we would like to have an illustration of the conversion step. Therefore each algorithm implemented in AlgoViz provides additional information and explanations.
   Just like the color converter, most beans demonstrating something based on a formula simply provide a LaTeX based explanation to use the possibility of displaying mathematical equations.
3. The result (the same color in HSV representation) is shown in a color diagram which uses an illustration of the

according color space which means that the HSV triple is rendered within a color pyramid.

4. The LaTeX Renderer is used to display the conversion equation and additional explanations.

In a visual programming environment the application can be built according to the diagram in Figure 5 without writing any source code. Figure 6 shows a screen shot during the creation process using the BeanBuilder. The data flow can be established using a simple wizard which can be seen in Figure 7.

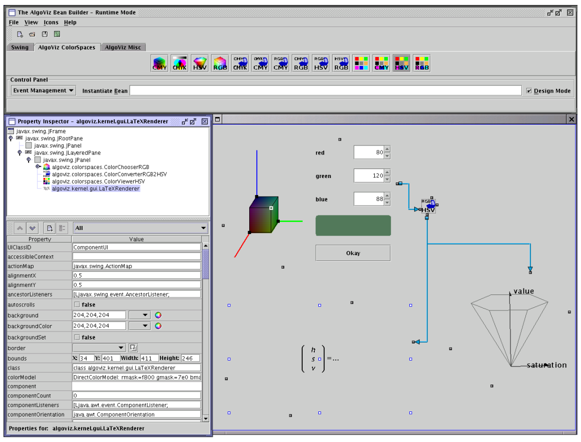

The AlgoViz Bean Builder - Runtime Mode
File View Icons Help
Swing AlgoViz ColorSpaces AlgoViz Misc
Control Panel
Event Management
Instantiate Bean
Design Mode
Property Inspector - algoviz.kernel.gui.LaTeXRenderer
red
green
blue
80
120
88
Okay
value
saturation
Properties for: algoviz.kernel.gui.LaTeXRenderer

Figure 6: *Creating the example application with the BeanBuilder. The screen shot shows the main window (top), the application to build (right) and the property editor (left).*

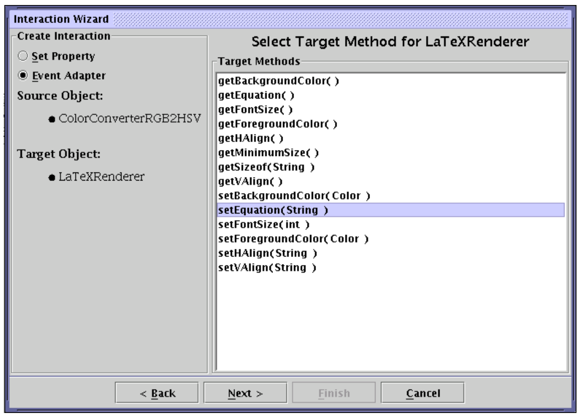
The similarity between the schematic diagram and the concrete realization is obvious. After a short period of domain-specific adjustment – especially in the usage of the preferred bean container – the building block concept used throughout the AlgoViz bean collection results in rapid development and, consequently, reduces the production time and cost in multimedia development.

3.3. Publishing Created Applications

Once a teaching unit has been created it may be published as a normal application which has to be downloaded and installed. This may be adequate if it is used quite often.

In other situations it might be appropriate not to install it but to start it via a network. In this case JavaApplets are a solution. However, due to problems with different browsers and platforms, it is not best choice. The Java Network Launch Protocol (JNLP), introduced by Sun, should be shortlisted. It avoids the problems raised by applets.

No matter what choice is taken, the AlgoViz bean collection supports each of them by using only those techniques and libraries which are available on all platforms and all supported distribution channels (applet, application, JNLP).



Interaction Wizard

Create Interaction

○ Set Property
● Event Adapter

Source Object:
● ColorConverterRGB2HSV

Target Object:
● LaTeXRenderer

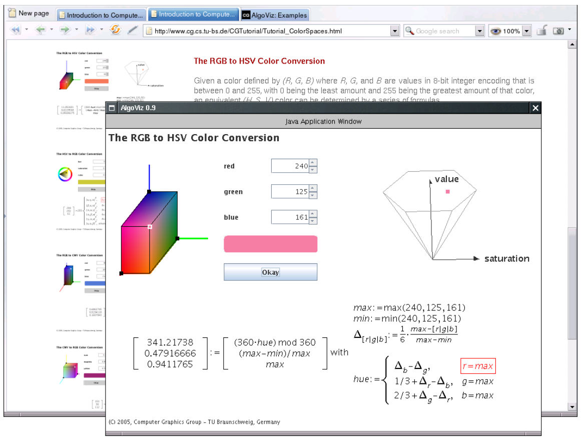
Select Target Method for LaTeXRenderer

Target Methods

```
getBackgroundColor( )
getEquation( )
getFontSize( )
getForegroundColor( )
getHAlign( )
getMinimumSize( )
getSizeof(String )
getVAlign( )
setBackgroundColor( Color )
setEquation(String )
setFontSize( int )
setForegroundColor( Color )
setHAlign(String )
setVAlign(String )
```

< Back | Next > | Finish | Cancel

Figure 7: *The Interaction wizard establishes data flow connections between beans. It filters possible connections and displays only those that have compatible types.*


New page
Introduction to Compute...
AlgoViz: Examples
http://www.cg.cs.tu-bs.de/CGTutorial/Tutorial_ColorSpaces.html
The RGB to HSV Color Conversion
AlgoViz 0.9
Java Application Window
The RGB to HSV Color Conversion
red
green
blue
240
125
161
Okay
value
saturation
(C) 2005, Computer Graphics Group – TU Braunschweig, Germany

Given a color defined by (R, G, B) where R, G, and B are values in 8-bit integer encoding that is between 0 and 255, with 0 being the least amount and 255 being the greatest amount of that color,

$max := \max(240, 125, 161)$
$min := \min(240, 125, 161)$

$$\begin{bmatrix} 341.21738 \\ 0.47916666 \\ 0.9411765 \end{bmatrix} := \begin{bmatrix} (360 \cdot hue) \bmod 360 \\ (max - min)/max \\ max \end{bmatrix} \text{ with } \Delta_{[r|g|b]} := \frac{1}{6} \cdot \frac{max - [r|g|b]}{max - min}$$

$$hue := \begin{cases} \Delta_b - \Delta_g, & \boxed{r = max} \\ 1/3 + \Delta_r - \Delta_b, & g = max \\ 2/3 + \Delta_g - \Delta_r, & b = max \end{cases}$$

Figure 8: *The ready-built example application published using Sun's JNLP technique.*

3.4. Usage of the AlgoViz Bean Collection

Despite the creation of new demonstration programs, other application areas are at hand.

Within a drill-and-practice environment, the bean collection can be used just like a class library. Students that use this library do not have to implement standard program frameworks – e.g. data input routines, matrix multiplications, user interfaces, etc. – and can concentrate themselves on the algorithms they should learn.

A totally different usage is offered by ready-built applications based on the AlgoViz bean collection. Our set of examples will form a computer graphics tutorial and will be extended until the fundamental algorithms and concepts of computer graphics science are well covered.

The tutorial can be utilized in a typical class room scenario as well as in a blended-learning scenario. Such a tutorial may be some kind of illustrated and animated table of contents showing the topics the lecture dealt with. This is useful for exam preparations and for students that join an advanced lecture without having attended the introduction.

3.5. Content of the AlgoViz bean collection

In order to create a complete computer graphics tutorial we strive to cover the fundamental algorithms of computer graphics and computer aided design. The following topics are already covered or will be covered in the near future.

- **Color Spaces:** This elementary topic is part of almost every introduction into computer graphics. Beans implementing color diagrams and color conversion routines visualize the connections between different color spaces.

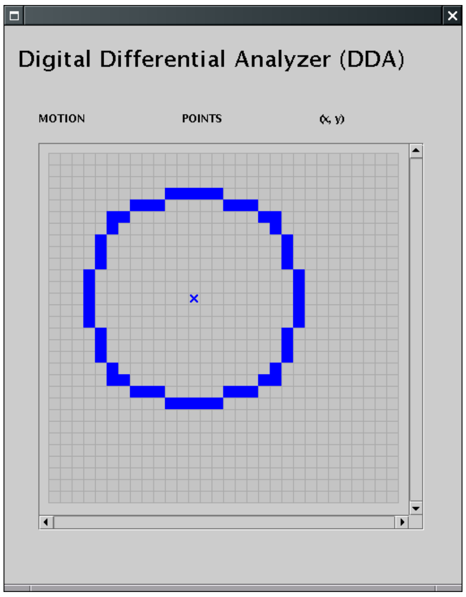


Figure 9: *Drawing a circle is a typical job for a DDA.*

- **Digital Differential Analyzers:** As a part of every two dimensional graphics library, digital differential analyzers (DDA) must not be missing.
- **Image Processing:** To explain image based algorithms or higher dimensional filter techniques, the AlgoViz bean collection provides beans which illustrate e.g. a wavelet transformation.

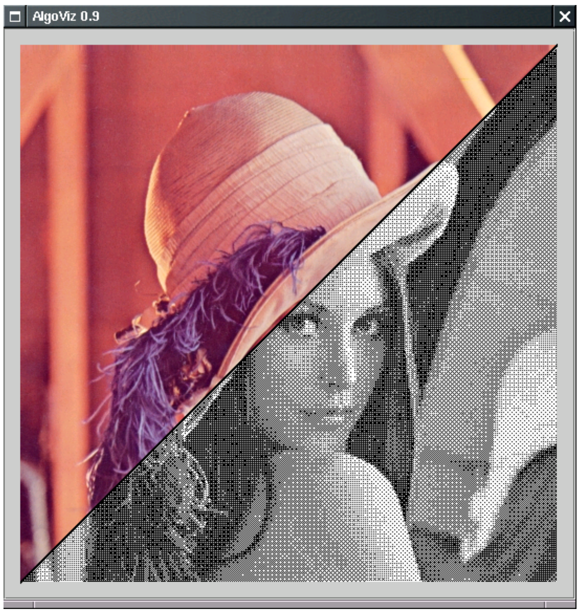


Figure 10: *The result of a simple dither algorithm.*

- **Perspective & 3D Transformations:** Computer graphics would be unimaginable without the basic concepts of perspective and transformations.

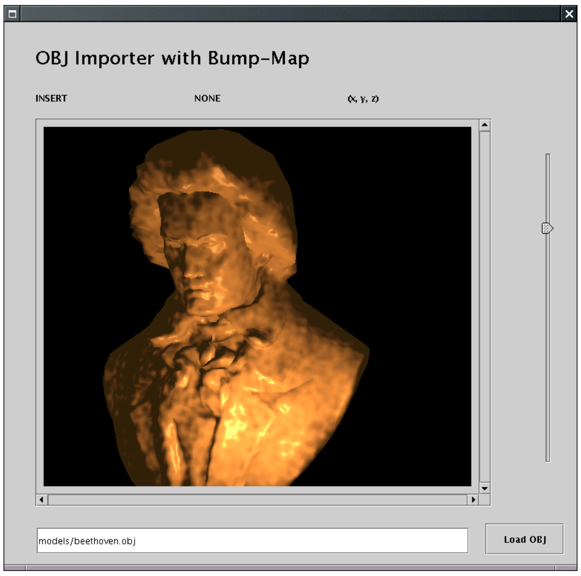


Figure 11: *A bump mapping illustration.*

- **Rendering Pipeline:** Today's rendering pipelines used by OpenGL or DirectX use techniques which can be illustrated with AlgoViz beans. These beans implement diverse shaders, lighting, bump mapping, etc.

- **Ray tracing:** A minimalistic ray tracer to bring out the main concept is in progress and will soon be ready.

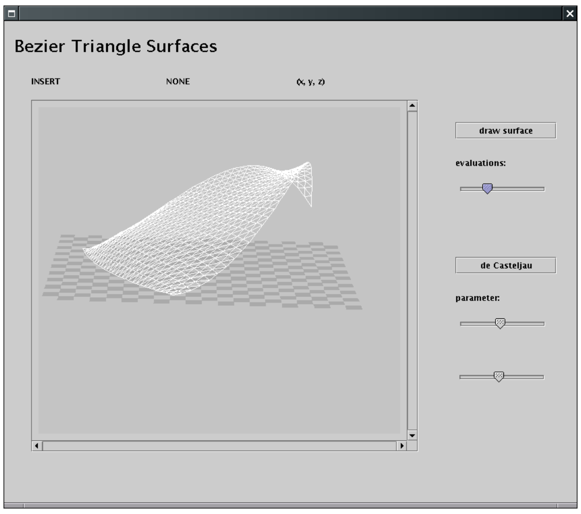


Figure 12: *A Bézier triangle surface.*

- **Curve and Surface Design:** Classical Computer-Aided Geometric Design (CAGD) topics like curve and surface design based on Bézier and B-Spline techniques are implemented in bean components for surface evaluation and degree elevation for example.
- **Subdivision Surfaces:** Another surface design topic that will be covered soon are subdivision surfaces which attracted significant attention within the last few years.
- **Computational Geometry:** Basic principles like triangulations or Voronoi diagrams are work in progress.

Beside these beans concerned with computer graphics algorithms and topics the AlgoViz bean collection also consists of software components without a special focus, nameable the LaTeX equation renderer, an image viewer, the matrix library and many more.

4. Results

Many courseware tools can be put into one of two categories. One the one hand there exists ready-to-use applications which offer almost no possibilities for enhancement, improvement or adaption to one's own courseware materials. On the other hand complex tool sets are available. These tools are very flexible and powerful. Unfortunately, they require an enormous time for domain-specific adjustment.

Our approach tries to solve this problem by providing both; a flexible tool set, which can be used in many contexts, and complete examples.

AlgoViz is a software environment which enriches computer graphics teaching. Due to its building block system it offers a very flexible environment for lecturers who want to visualize algorithms efficiently and who do not have the time to program everything from bottom up. The created applications can substantially enrich a lecture. Furthermore, the visualizations can be published rather effortlessly via internet using Sun's JNLP or JavaApplet technique.

5. Future Work

The AlgoViz project is currently focused on the visualization of fundamental computer graphics algorithms and geometric modeling concepts. A lot of topics are already covered by corresponding AlgoViz beans, but the completion of the bean set of basic algorithms is still has to be done. At the same time the expansion of the CG tutorial will be addressed.

In the near future AlgoViz will cover the most important, fundamental computer graphics algorithms and will be available at

```
http://graphics.tu-bs.de/
   research/projects/AlgoViz
```

as a framework for visualization issues in computer graphics education. The computer graphics tutorial will be published under the internet address

```
http://graphics.tu-bs.de/
   CGTutorial/Tutorial.html
```

References

[Bro02] BROWN UNIVERSITY COMPUTER GRAPHICS RESEARCH GROUP: *The Exploratories Project.* (online) http://www.cs.brown.edu/ exploratories/ home.html, 2002.

[CP02] CHI E. H., PALO ALTO RESEARCH CENTER: Expressiveness of the data flow and data state models in visualization systems. In *Proc. of the Advanced Visual Interfaces Conference* (2002), pp. 375–378.

[DKDB01] DROFENIK U., KOLAR J., DUIJSEN P. V., BAUER P.: New web-based interactive e-learning in power electronics and electrical machines. In *Conference Record of the 2001 IEEE Industry Applications Conference* (2001), vol. 3, pp. 1858–1865.

[FEJ03a] FIGUEIREDO F. C., EBER D. E., JORGE J. A.: Cgems - computer graphics educational materials server. In *30th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 03)* (2003).

[FEJ03b] FIGUEIREDO F. C., EBER D. E., JORGE J. A.: A refereed server for educational cg content. In *Proc. of European Association for Computer Graphics (Eurographics 2003)* (2003).

[Fel91] FELLNER D. W.: Object-oriented programming – does it help in computer graphics? *New Results and Trends in Computer Science* (1991), 132–151.

[Fla02] FLANAGAN D.: *Java in a Nutshell*. O'Reilly & Associates, 2002.

[KHS98] KLEIN R., HANISCH F., STRASSER W.: Web based teaching of computer graphics: Concepts and realization of an interactive online course. In *SIGGRAPH 98 Conference Proceedings* (1998).

[KMS*] KILIAN M., MOHS T., STRAUB R., BANGERT C., PRAUTZSCH H.: *CAGD-Applets – an interactive tutorial on geometric modeling*. (online) http://i33www.ira.uka.de/ applets/.

[LS01] LALEUF J. R., SPALTER A. M.: Create@brown, a component repository for learning objects: A progress report. In *Proceedings of ACM JCDL* (2001).

[McL01] MCLAUGHLIN B.: *Java & XML: Solutions to Real-World Problems*. O'Reilly & Associates, 2001.

[Min] MIN P.: *Computer Graphics Applets*. (online) http://www.cs.princeton.edu/ min/ cs426/ applets.html.

[SGH*] STRASSER W., GUMHOLD S., HANISCH F., HAUTH M., EHLERT A.: *Spielend Visualisieren*. (online) http://www.gris.uni-tuebingen.de/ projects/ vis/ titlepage.html.

[Sun] SUN MICROSYSTEMS: *The BeanBuilder*. (online) http://java.sun.com/ products/ javabeans/ beanbuilder/.

[Tak] TAKATSUKA M.: *JBeanStudio*. (online) http:// jbeanstudio.sourceforge.net.

[TG02] TAKATSUKA M., GAHEGAN M.: Geovista studio: A codeless visual programming environment for geoscientific data analysis and visualization. *The Journal of Computers & Geosciences* (2002).

[UFK*89] UPSON C., FAULHABER T., KAMINS D., LAIDLAW D., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics & Applications 4* (1989), 30–42.