Using MuPAD and JavaView to Visualize Mathematics on the Internet

M. Majewski
Zayed University, UAE
majewski@mupad.com

K. Polthier
Zuse Institute Berlin (ZIB)
polthier@zib.de

Abstract Mathematics education strongly benefits from the interactivity and advanced features of the Internet. The presentation of mathematical concepts on the Internet may go far beyond what we could demonstrate in traditional mathematics textbooks. In this paper we demonstrate, in a number of examples, the additional insight into complex mathematical concepts that can be gained from 3D interactive visualization embedded into web pages. Mathematical visualization is improving our teaching environments and the communication between teachers and students.

We combine two mathematical software systems—MuPAD as a development platform and JavaView for computation and online visualization of interactive mathematics experiments. We discuss the practical aspects of online publications and show some technical details about how to develop mathematical experiments on your own. The conference presentation will demonstrate MuPAD and JavaView components in live-action.

1. Introduction

There is no doubt about the future role of the Internet in education. For many teaching and research institutions Internet is the main vehicle for publishing and discussing scientific concepts, doing experiments and cooperate with the global academic community. If we wish to be successful in progressing with this concept, it is very important to develop the most efficient tools for the online communication of scientific ideas, and high quality materials that can be used through the Internet.

Mathematics, with its abstract ideas and concepts, is one of the disciplines where visualization can significantly simplify and intensify the learning process. Creating online models to visualize mathematical concepts is certainly a challenging task. We must mention two major difficulties — first, the translation of abstract mathematical ideas into computational descriptions and, second, the technical difficulties of implementing such descriptions. Many educators are able to develop a computational description of a mathematical idea. They must choose the objects to be represented and how they will look on the computer screen, the type of animation and other features to be used. However, the technical knowledge can be a significant barrier even for very experienced teachers. Usually they need to be familiar with some programming languages or computer packages that can be used for developing online models. They have to know how to publish their models and finally about software limitations on the user side. The need for such technical knowledge often prevents many educators from developing their own online materials.

Let us concentrate for a while on the nature of the Internet and see how this environment can influence development and publishing of online visualizations. Everyone knows that Internet means millions of individual computers, servers, cables and other hardware, connected in a huge network. The advantage of such a network is that everybody can access anything that was left for him on one of the servers. Usually we do not need any sophisticated and expensive software on our local computers to access and use the online resources. Information on the Internet is distributed, which means that various pieces related to the same topic can be located on different servers. This leads to an interesting opportunity of developing online courses based on distributed resources.

One of the limitations of such environment is the variable connection speed of accessing information. Some parts of the network are very fast and while others are too slow to transmit documents even with simple images. Another feature of this environment is the diversity of the operating systems on the local computers. End users currently use various versions of MS Windows, Linux clones, Mac OS and many other operating systems. Finally, on their local computers, the end users may have very different software, different Internet browsers and different fonts installed. Their machines can be very fast or very slow.

All the mentioned features give us some hints how online applications should be designed. Taking care of all these aspects is important for online visualization of mathematics as we will transmit large sets of data, perform intensive online calculations and produce interactive images in real time. Here are some aspects, which are essential for web-based applications:

1. The size of used files must be very small.
2. Complex applications should have a modular design. Additional parts are downloaded only when required during an experiment.
3. Applications should run on a majority of end-user computers, web browsers and operating systems.
4. Applications should deliver the major and unique components but try to use as much as possible resources of the client computer.
5. Applications should have enough power to deal not only with static pictures but also with complex calculations and the manipulation of 2D, 3D images and animations.

A number of authors have analyzed selected tools for the use on the Internet to visualize mathematical concepts (see for example [1],[3],[4]). From their observations it follows that applications written in the Java programming language fulfill all the above mentioned constraints (see [3],[7]). Java applets can be very modular and additional classes can be downloaded from the server only when needed. With Java, we are able not only to manipulate graphics but also build complex computing engines. Finally, a Java runtime system is often pre-installed, or can be installed directly from Sun web site (`www.java.com`), for most computers and operating systems. Because of the principles of Java, the same Java applet is able to run on many different operating systems and web browsers.

2. The conceptual model of JavaView

JavaView is a software for computation and visualization in mathematics. The software is written in the Java programming language. The class libraries are well-documented and allow the development of custom applications and experiments. JavaView is free and its class libraries can be download from `www.javaview.de`.

An important feature of JavaView is the smooth integration as a 3d viewer for graphics and animations, as well as a geometry processing engine, into popular Computer Algebra Systems (CAS) like Maple, Mathematica, Matlab and MuPAD. For example, JavaView is an official Maple Powertool. Figure 1 shows a JavaView applet that uses a webMathematica server to compute a complex surface. The JavaView clipping algorithm is used to explore the surface. Figure 2 shows JavaView applet displaying Maple plot on the home page of the Maple Powertools.

In the simplest case, JavaView is used to display a graphic produced by the CAS. This provides the CAS with a state of the art viewing environment that allows interactive scaling, translation, rotations in 2D, 3D, and 4D, picking vertices, configuring material colors, adding textures and transparency. More advanced applications use JavaView to build interactive components and animations in web pages or use the visualization capabilities of JavaView to explore complex graphics, for example by clipping parts of the geometry. From inside Mathematica, full scripting of JavaView is possible, based on Mathematica J/Link package. This works bidirectional, i.e. JavaView events are able to invoke Mathematica computations, which then change the content shown by JavaView.

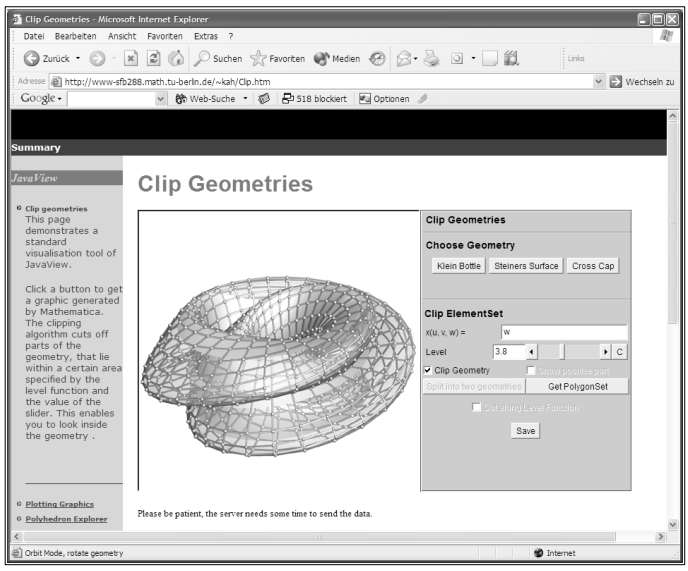


Fig 1. JavaView applet using webMathematica server

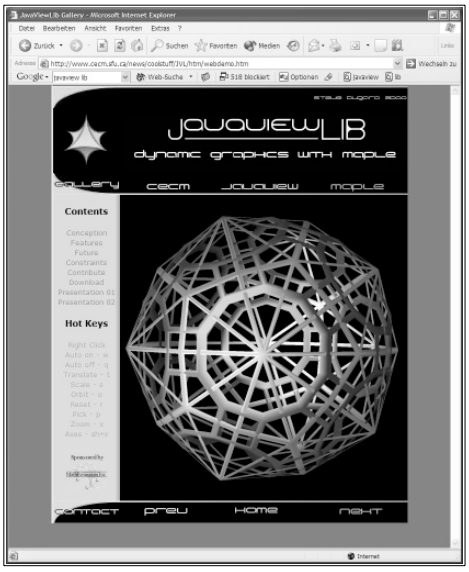


Fig. 2. Maple plot in JavaView

Additionally, webMathematica provides the technology to query a Mathematica kernel from a Java applet over the web. This enables web pages equipped with JavaView applets to access the whole Mathematica computation capabilities. On the other hand, webMathematica users who want to build web interfaces to their Mathematica solutions can use JavaView as a convenient tool to enrich their web pages with high quality visualizations and interactive elements.

This is important to mention the particular relation between JavaView and MuPAD. Both, MuPAD and JavaView, use XML format to describe its graphics. Many features of graphical objects implemented in MuPAD are the same, or very similar to those implemented in JavaView. The scene, camera and light concepts are the same in both programs. Therefore, geometries developed in MuPAD and saved to JavaView format, with some exceptions, behave and look the same in JavaView like they were looking in MuPAD.

2.1 JavaView – an overview

JavaView can be used in different context. For example, it can be used as a standalone application on a local computer, or in the form of specialized applets embedded into web pages. Such applications allow an interactive calculation, display and manipulation of 2D and 3D geometries. The code of JavaView is highly optimized to reduce file size while still having very sophisticated functionality. Like with many other Java applets, the web browser on the user side downloads only the necessary classes and model files. Therefore, the download time can be much shorter than for other types of online applications.

When downloading a web page with an embedded JavaView applet, the user may even be not aware that he is dealing with something different than a web page with static pictures. However, the image produced by JavaView on the web page can be rotated, scaled, moved, animated, etc. Easily accessible menus contain hundreds of operations allowing us to perform complex experiments with our models. Figure 3 shows a very basic 3D model displayed by JavaView.

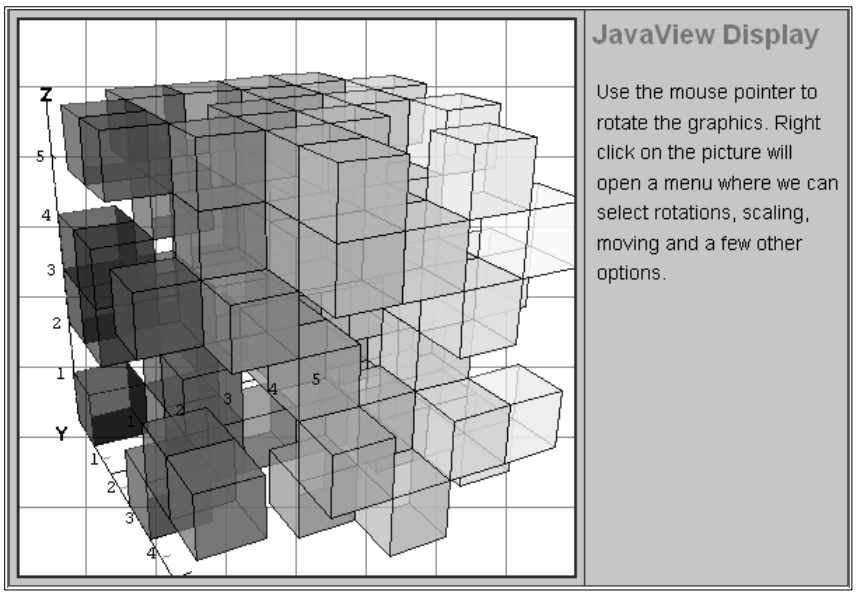


Fig. 3. A simple JavaView model embedded into a web page

A more complex JavaView model may contain additional menus, toolbars, panels or sliders. An example of such model is shown in figure 4. Here the list-browser applet of JavaView allows the user to select from a user-specified list of precomputed geometry models.

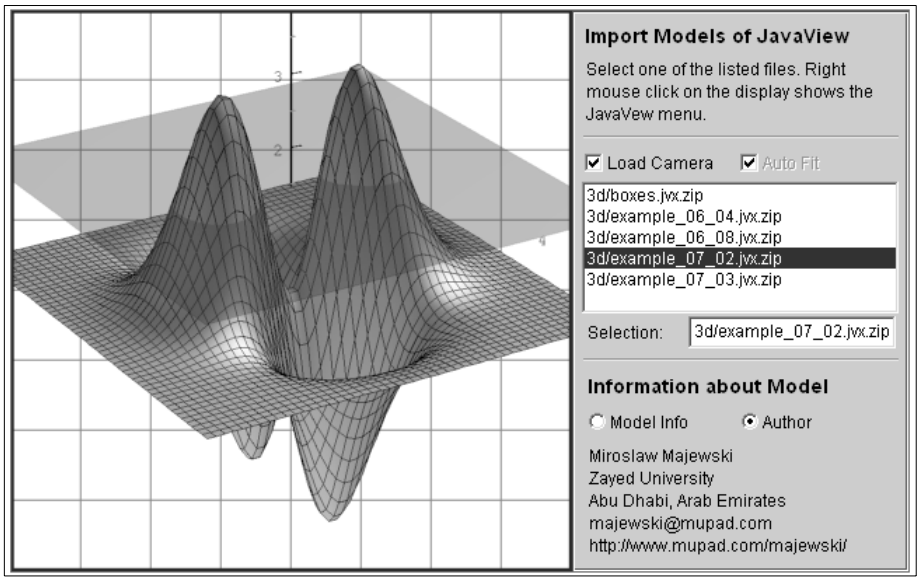


Fig. 4. The JavaView list-browser applet

In the following sections of this paper, we will show how users can create their own JavaView models and add them to a web page.

2.2 JavaView Lite, JavaView full and applications

JavaView is delivered as a set of jar archives. Each archive contains a collection of Java classes in a compressed form. The libraries are well documented, allowing the development of own applets and applications based on the high-level classes and methods available in JavaView. The user guide

provides an introduction to programming in Java and the usage of the classes of JavaView. The reference documentation provides detailed comments for all classes and methods. The documentation was automatically generated from the JavaView source code with the javadoc utility.

The basic archive JavaView.jar contains packages for the 3D display, geometry classes, import and export loaders, linear algebra classes, and some other sub packages. The optional archive jvx.jar provides extended geometry classes, a number of powerful workshop classes, and additional loaders. A second optional archive vgpapp.jar contains a set of JavaView applications and introductory tutorials for programmers.

For the efficient inclusion of precomputed geometry in interactive images, respectively applets in online documents, the JavaView distribution contains a special archive, namely, the jvLite.zip archive which is drastically optimized for download. This tiny lite version is about 15% of the size of the original archive JavaView.jar. jvLite.zip was automatically generated using the JAX tool of IBM by a sophisticated analysis of the Java byte code and by removal of all interface classes from JavaView.

2.3 Programming library

The public interface of JavaView allows the use of the library for own Java applications and applets. The spectrum of applications that can be created ranges from interactive educational applets for online courses to computational services for research and industrial applications as well as mathematical experiments that can easily be published on the web.

The JavaView library offers:

- Data structures to represent geometries
- Algorithms for geometric modeling
- A numeric and a linear algebra package
- A framework to build animations
- Classes that handle picking and camera events

The framework to build own applications in JavaView is called a project. For programmers, projects simplify the integration of modules with the JavaView environment by providing an easy access to display windows, animation support and handling of display and camera listeners. A project is a full-fledged application and similar to a Java applet. For example, a project often provides the setup of a mathematical experiment including different Java classes, panels and dialogs, and HTML descriptions. In comparison to Java applets, JavaView projects provide a more flexible functionality by deriving from an own superclass of jv.project.PjProject. Whereas Java applets are solely designed to run inside an HTML page. JavaView projects may be created in an applet but may as well be invoked by another project. Since each project has a well-defined interface for its configuration, this allows to reuse the same project in different applets, where each time the project is configured differently.

3. Producing and publishing JavaView models on the web

There are a number of methods for producing JavaView models. In this article we concentrate on the most recent development—producing models with MuPAD and exporting them to JavaView. Although some other possibilities are still discussed, we will concentrate only on using the native JavaView format JVX. JVX files are text files following XML syntax. One can read them like any other text files and edit them by hand. MuPAD graphics properly exported to JVX contains all necessary information. Therefore, we can simplify our activities to produce graphics in MuPAD, exporting it to JVX and adding to a web page. In some situations we may wish to modify the final JVX file in JavaView so it will contain specific features.

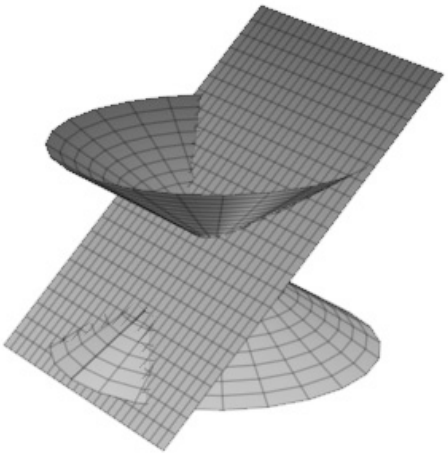
It is important to note that JavaView models produced this way will be mostly limited to displaying, customizing and animating 3D geometries and scenes. However, much sophisticated functionality of such models can be obtained by using or writing specialized applets extending functionality of JavaView.

3.1 Producing graphs and geometries in MuPAD

Let us start with a very simple example and show step by step how such example can be created in MuPAD, saved as JVX files. The goal of this example is to produce a picture of a cone and show the shape of the curve formed by the intersection of the cone with a plane.

We can start by declaring all necessary components in MuPAD. At this stage we do not bother about colors, lines and other parameters of our objects. We just create a cone using spherical coordinates and a plane using its functional description. In the final output we force constrained scaling and we remove the coordinate system. Here is all necessary MuPAD code and the resulting picture.

```
Cone := plot::Spherical([u, v, 1], u = -2*PI..2*PI, v = 0..2*PI):
Plane := plot::Function3d(x+1, x=-5..5, y=-5..5):
plot(Cone, Plane, Scaling=Constrained, Axes=None)
```



Such an image can be enhanced using VCam, which is a MuPAD tool. We can improve the quality of the surface, apply some additional corrections to it, change the transparency of the cone or the plane, and so on.

Now, we are ready to export our image to JavaView. In VCam, in `File` menu we choose the `Export` option and then from a long list of possible formats we choose JavaView Geometry (*.JVX). In MuPAD version 3.1, we can also save current display setting file *.JVD.

Finally we attach both produced files to a web page using very simple HTML code:

```
<applet archive=jars/javaview.jar,jars/jvx.jar
    width=400 height=350 code=javaview.class>
   <param NAME="model" VALUE="conical.jvx">
   <param NAME="displayFile" VALUE="conical.jvd">
   <param NAME="control" VALUE="Hide">
   Loading parametric surface.
</applet>
```

The final result on the web page may look like the one in the figure 5. We made it very simple so users can concentrate only on the shape of the intersection, i.e. the conical curve. A number of other changes and experiments can be done directly using applet menus (right click on the applet area). While

displaying this model on a web page we can rotate or translate any of these two objects separately. This way students can observe how the shape of the conical section changes depending on the location of the plane and its slope.

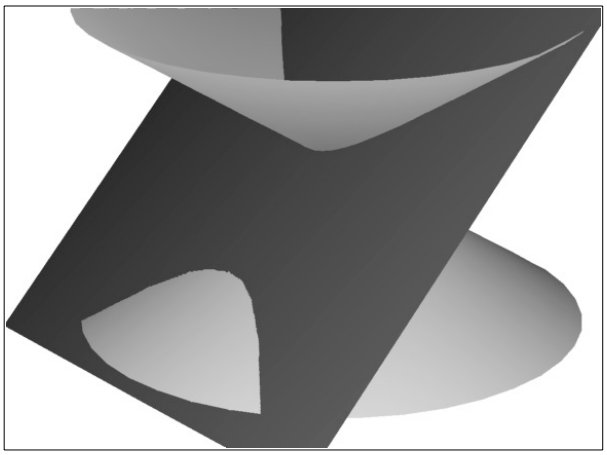


Fig. 5. Final JavaView model on the web page

3.2 Customizing models in JavaView

As we have said before, JavaView can be used also as modeling tool. By executing JavaView on a local computer we get into a very sophisticated modelling environment where we can manipulate, modify our models and save them in many different formats for future use.

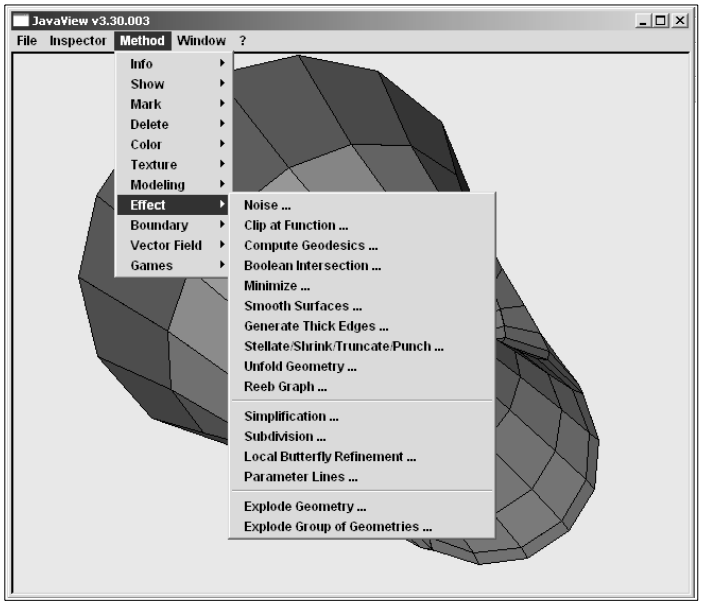


Fig. 6. JavaView Effects menu

The JavaView application contains sophisticated algorithms for geometry processing and visualization. Each algorithm has an individual dialog that provides control over the relevant parameters. The algorithms can be selected from the menus `Modeling` and `Effect`. Among the included tools for geometric modeling are:

- Mesh simplification
- Global and local subdivision schemes

- Mesh optimization using several curvature-based criteria
- Boolean operations on a set of curves and surfaces
- Feature preserving surface smoothing
- Cliping a geometry at an arbitrary level function
- Stellate, punch, shrink or truncate the faces of the mesh
- Visualize techniques for vector fields on surfaces
- Various surface coloring techniques based on scalar fields such as curvature
- Computation of topological properties such as Reeb Graph
- Scalar field analyzer
- ... (many other)

Figure 7 demonstrates the JavaView subdivision functionality. The left image shows an initial, roughly discretized surface. The user may choose an adequate subdivision method from a collection of ten different schemes. The image in the middle shows the refined surface. Still the mesh of the initial surface is available as a control grid to model the refined surface. The vertices of the control grid can be interactively dragged with the mouse. The image on the right shows the resulting surface after some modeling has been performed.

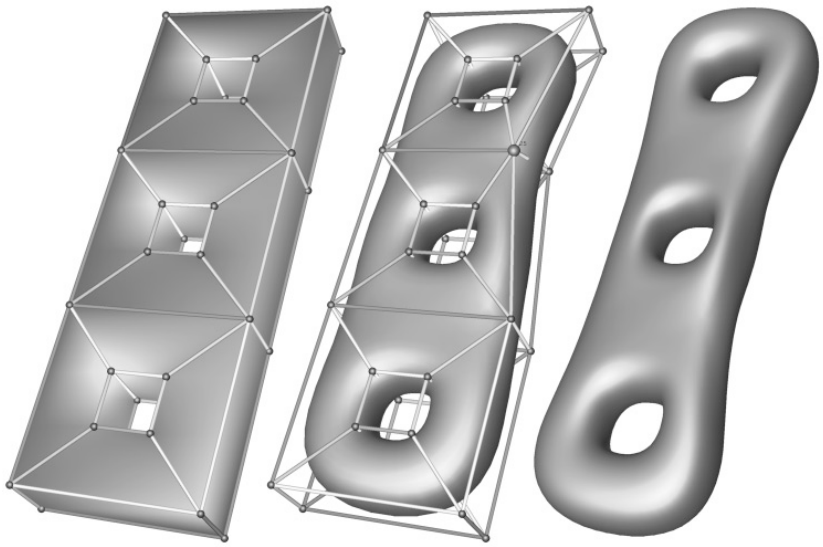


Fig. 7. Initial surface of genus 3 created with JavaView

4. Selected JavaView applications for school use

JavaView contains a number of specific and very interesting educational applications. Many of them are published on the JavaView web site at `www.JavaView.de` in the section entitled Geometric Surfaces. Here we can analyze a large group of predefined surfaces and change the parameters of their graphs. The enclosed figure 8 shows JavaView application with a crossection of the Kuen surface so one can see how it looks inside.

Another interesting application from a school point of view is an applet to display and experiment with solids. In this applet we can experiment with a number of famous solids. We can display them in highly attractive form where edges of applets are shown as tubes with a given number of sides and a given radius.

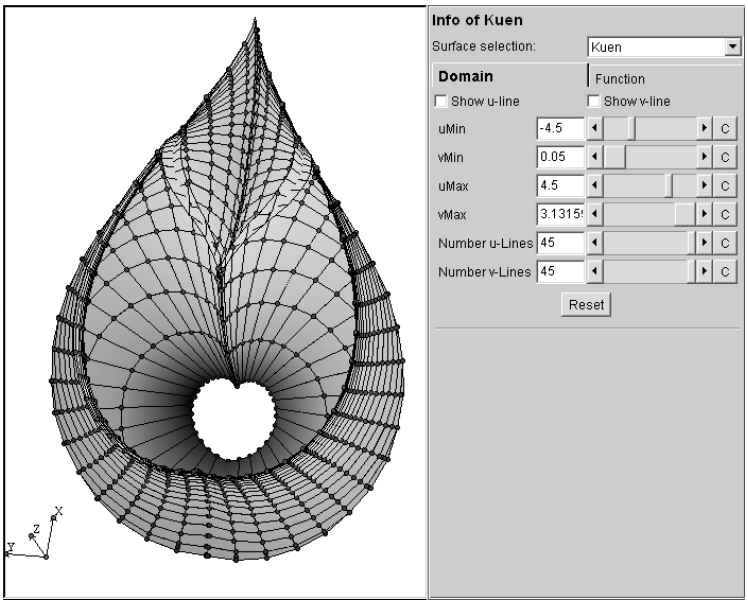


Fig. 8. Crossection of the Kuen surface in JavaView

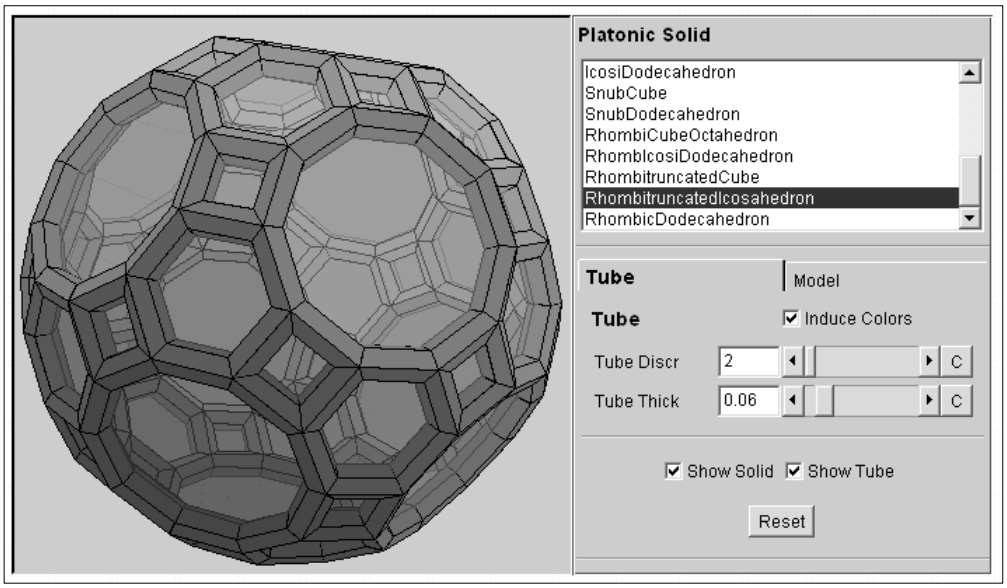


Fig. 9. JavaView applet to explore Platonic Solids

In a short paper, it is impossible to describe all the existing JavaView applications. In fact some of them may disappear in the future or may be replaced by newer versions. There are many JavaView applications that are not displayed on the JavaView web site but can be downloaded and explored on a local computer. For example, the tutorials section (see jv-tutor.zip file) contains a number of interesting examples which are very useful in the school practice. The figure below shows the L-system JavaView applet. This applet can be used for exploring properties of L-systems and a number of other activities. For example, the picture shows creation of a grid-like fractal. One of interesting school activities is to investigate how the pattern develops, invent formulae for the number of vertices in each step of iteration, the size of the grid, the number of edges in the grid, etc.

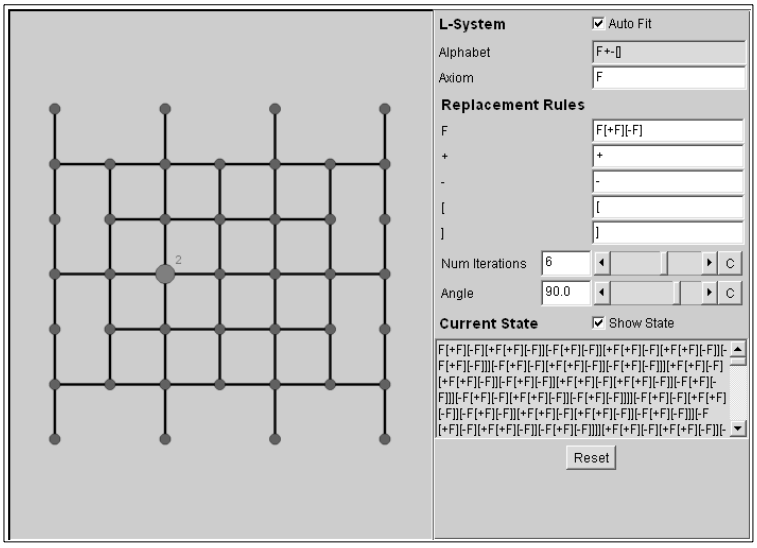


Fig. 10. JavaView applet to explore L-systems

Finally, let us mention the most recent, somewhat experimental, additions to JavaView applications. This is large collection of JavaView applets for exploring gravity. These applets can be very useful as an experimental online laboratory for learning physics.

7. Conclusions

The diversity of JavaView applications presented in this paper demonstrates how useful JavaView can be for mathematics teachers, mathematics students and scientists. We can use it to display mathematical concepts, build interactive mathematical experiments and add interactive component to our online courses. Creating JavaView models for MuPAD, Maple or Mathematica users can be a very easy task provided that they got some experience in using their CAS. Java programmers are able to extend functionality of JavaView and build custom applications using JavaView classes. The number of web sites using JavaView models proves that JavaView is already an appreciated tool in academic community.

8. Literature

1. Bauslaugh B., Cannings C.L., Nicholson K.: *An Intutitive Approach to Elementary Mathematics on the Web*, in Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.
2. Caprotti O., Cohen H., Sterk H., *OpenMath Technology for Interactive Mathematics Documents*, in Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.
3. Cervone D.P., *The StageTools Package for Creating Geometry for the We*b, in Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.
4. Husch L.: *Visual Calculus—Development and Tools*, in Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.
5. Joswig M., Polthier K.: *EG-Models—A New Journal for Digital Geometry Models*, Springer Verlag 2002, in Multimedia Tools for Communicating Mathematics.
6. Polthier K., Khadem S., Preuß E., Reitebuch U.: *Publication of Interactive Visualizations with* JavaView, in Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.