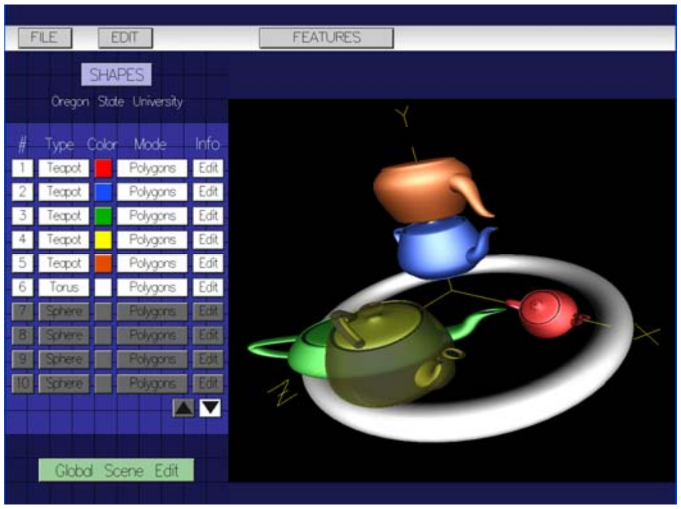
Shapes: Allowing K-12 Students to Work in 3D

Mike Bailey
Oregon State University

Steve Lukas
Soapbox Mobile, Inc.

Rozeanne Steckler
Oregon State University



#	Type	Color	Mode	Info
1	Teapot	🟥	Polygons	Edit
2	Teapot	🟦	Polygons	Edit
3	Teapot	🟩	Polygons	Edit
4	Teapot	🟨	Polygons	Edit
5	Teapot	🟧	Polygons	Edit
6	Torus	⬜	Polygons	Edit
7	Sphere	▨	Polygons	Edit
8	Sphere	▨	Polygons	Edit
9	Sphere	▨	Polygons	Edit
10	Sphere	▨	Polygons	Edit

Teapots Galore

Abstract

Students of all ages are motivated to learn to think and work in 3D. They see it in movies such as *The Incredibles* and *Chicken Little.* They see it in television shows such as *Max Steel* and *Jimmy Neutron*. They see it in their video games. It is an integral part of their world. It makes sense, then, that they can be taught the essence of 3D graphics by letting them create their own 3D scenes. This paper presents a program called *shapes*. It allows kids of all ages to create their own 3D scenes that are interesting and compelling. By gently and subtly weaving Cartesian coordinates throughout the program, the kids learn about X, Y, and Z without even knowing they are learning it.

Goals

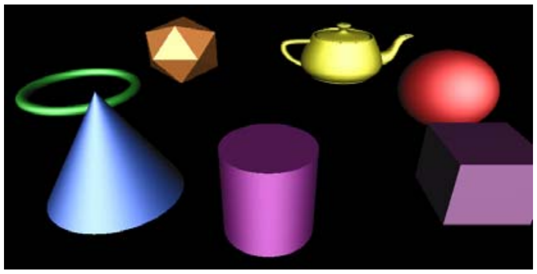
We had two goals in mind when creating *shapes*. One was to get K-12 students interested in computers as a means to create something. So much of kids' exposure to computers is in the form of video games, where too often the plot is to destroy something or someone. We were also trying to show a side of the computer as an aid to the creative process of building something. We wanted a program that we could point to and talk about possible careers in engineering, design, animation, game development, and art.

The second goal was to teach Cartesian coordinates. All school kids will see this eventually. We felt that we could show it to them early, and in a more enticing way than they would ever see in school. Kids seem to be thrilled when they learn how something really works. This would be our chance to sneak a little computer graphics math into their way of thinking.

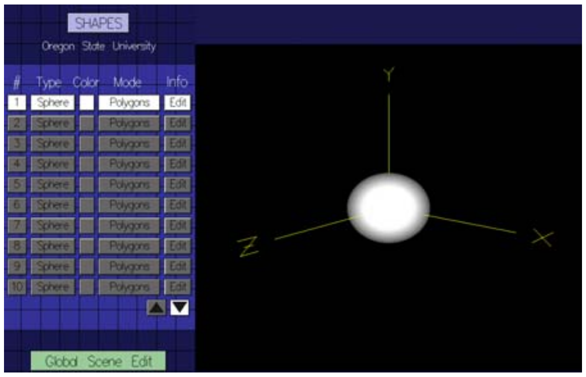
Using shapes

Shapes allows the user to select one of seven basic geometric primitives:

- Sphere
- Box
- Cylinder
- Cone
- Torus
- Icosahedron
- Teapot

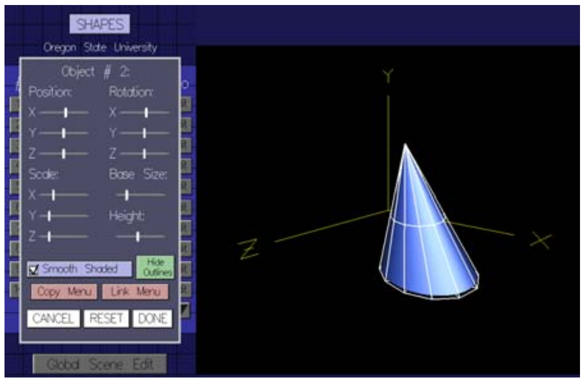


When selected, the object appears in white in a default size at the origin, as shown here. From there, it can have a color assigned to it. It can be toggled between wireframe and solid. Best of all, it can be edited.



The Default is a White Shape at the Origin

All objects in the scene have an Edit button. When pressed, the user is presented with a dialog box that allows the shape to be re-positioned in X, Y, and Z. The object can be rotated about any axis. It can also be non-uniformly scaled. Some of the objects have specific editing controls, such as the cone's base radius shown here.

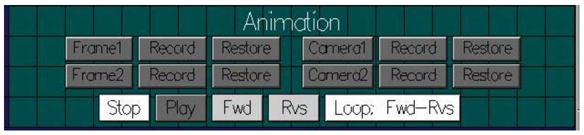


Editing a Cone

Editing controls are all buttons, sliders, and checkboxes. We created our own user interface widgets from OpenGL calls to guarantee both portability and consistency across operating system implementations.

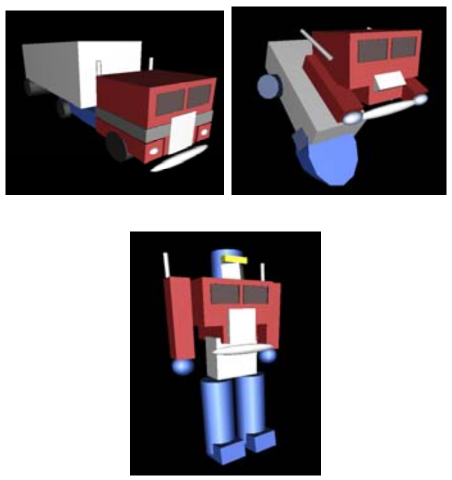
Animation

Animation comes about through interpolating two state vectors. A state vector is the set of all parameters that define the current scene. This interface is shown below.



Animation Menu

The user puts the scene in one state consisting of shapes, positions, rotations, scales, and colors, and hits the first Record button. She then places the scene in another state and hits the second Record button. Hitting the Play button then interpolates between the two, possibly forward only, possible forward-and-backwards. The following figures show a "transformer" animation – part robot, part truck.

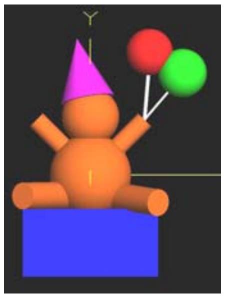


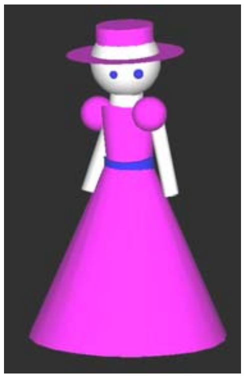
Animating Optimus the Transformer

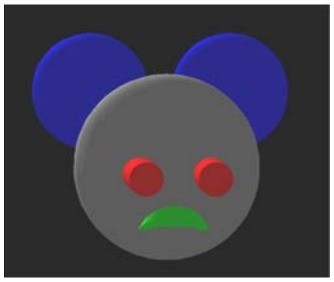
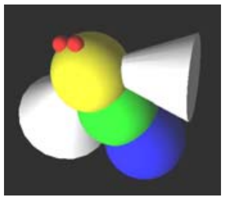
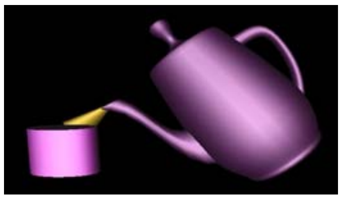
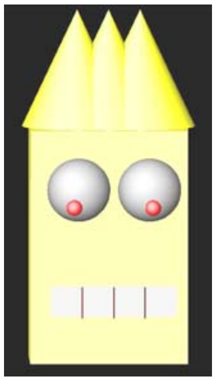
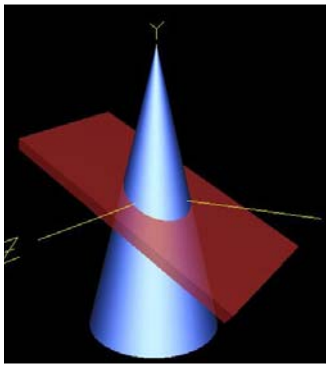
Some Scene Examples

These are some of our favorite scenes that have been created over the course of the last few years.

All were created by various students, with the exception of the last one which was created by one of the school teachers who was experimenting with using *shapes* to explain conic sections.



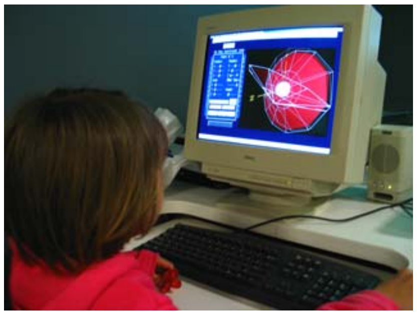


Examples of Scenes Created Using shapes

Experience

We have used the *shapes* program in a variety of outreach programs, including those for high school, middle school, lower school, and even Kindergarten. We have used it for both Girl Scout Computer Badge days and for the Boy Scout computer merit badge. We have used it for Take-Our-Children-To-Work-Day events.

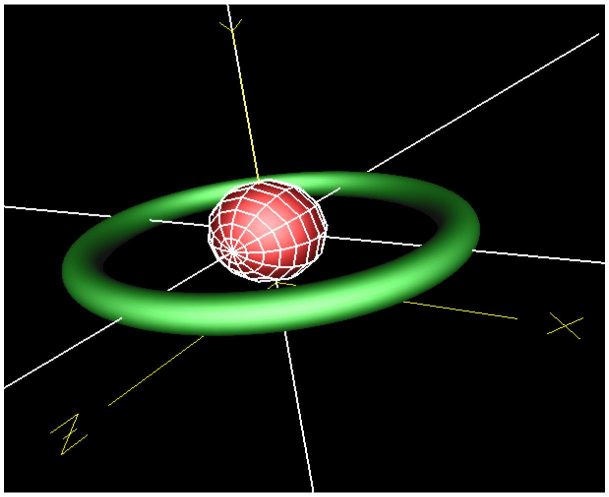


This photo shows a 6-year-old using *shapes*. Our experience is that any child above around age 5 can learn 3D because they are highly motivated to do so, and because *shapes'* simplicity makes it so effortless. The barrier to entry for the kids below age 5 does not appear to be the 3D, but not yet knowing how to read the words in the interface.

Understanding the concept of Cartesian coordinates does not seem to be a barrier to entry. Many times, students came in already knowing X and Y from school. In these cases, extending their understanding into Z is only a small jump. What about those who have never heard about any coordinate system? So far, we have never found that to be a problem. The kids are so motivated to want to make their own 3D scene that they don't hesitate to try. And, even if they are not completely able to predict in advance which direction X, Y, and Z are, they keep positioning or rotating in different directions until they get it right. After a while, they seem to "get" the coordinate system, without even realizing that they have learned it.

Junior-level Girl Scouts at a Computer Badge Day

The biggest problem the students seem to have with working in 3D is the same problem that adults have: the ambiguity of lining things up. The kids will meticulously place one 3D item on top of another, only to rotate the scene and find that the two shapes are widely separated. To alleviate this frustration, editing a shape automatically causes a large 3D cursor to be attached to the object being edited. In this way, the students can see what it will take to move this object to a specific location with respect to another, in this example, moving the red sphere to the same Y-level as the green torus. This seems to alleviate the bulk of the students' frustrations with 3D.



A 3D Cursor Aids in Alignment

Implementation

Shapes is written in C++ using OpenGL [Shreiner2005] for the graphics, and GLUT [Glut2006] for the window interface. Thus, shapes can be recompiled for Windows, Linux, UNIX, or Macintosh. The only specific operating system concession is that the Windows version uses *CFileDialog* from the Microsoft Foundation Classes for selecting files to save, load, or write an image into.

Conclusions

The best learning takes place when students teach the material to themselves, and don't even realize that they have learned something. This is the case here. Very young kids have made unexpectedly sophisticated scenes, while having no prior experience with coordinate systems or 3D.

Shapes seems to be especially effective for working with girls. Girls, in particular, seem to like the sense of building something, rather than shooting at it or blowing it up. Interestingly, we have not noticed any boy/girl differences in how well the kids can think and work in 3D. Conventional wisdom says that boys have better spatial reasoning than girls, but we haven't seen it. If that is true, then perhaps the right tools make up the gap.

We call this type of learning "Trojan Horse Education". The information ends up in the students' brains, but it snuck in disguised as fun. And, like the original Trojan Horse, the recipients do most of the work bringing it in.

Getting *shapes*

There are versions of shapes for Windows, Linux, and Macintosh. To get the shapes program and the documentation, go to:

`http://eecs.oregonstate.edu/~mjb/shapes`

References

GLUT2006 http://www.opengl.org/resources/libraries/glut.html

SHREINER2005 Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis, *OpenGL Programming Guide*, 5th Edition, Addison-Wesley, 2005.