Asymptote: Interactive TEX-aware 3D vector graphics

John C. Bowman

Abstract

Asymptote is a powerful descriptive vector graphics language for technical drawing recently developed at the University of Alberta. It attempts to do for figures what (LA)TEX does for equations. In contrast to METAPOST, Asymptote features robust floating-point numerics, high-order functions, and a C++/Java-like syntax. It uses the simplex linear programming method to resolve overall size constraints for fixed-sized and scalable objects. Asymptote understands affine transformations and uses complex multiplication to rotate vectors. Labels and equations are typeset with TEX, for professional quality and overall document consistency.

The feature of Asymptote that has caused the greatest excitement in the mathematical typesetting community is the ability to generate and embed inline interactive 3D vector illustrations within PDF files, using Adobe's highly compressed PRC format, which can describe smooth surfaces and curves without polygonal tessellation. Three-dimensional output can also be viewed directly with Asymptote's native OpenGL-based renderer. Asymptote thus provides the scientific community with a self-contained and powerful TEX-aware facility for generating portable interactive three-dimensional vector graphics.

1 Introduction

Notable enhancements have recently been made in the TEX-aware vector graphics language Asymptote.[1] This article provides an overview of those advances made since the publication of articles in *TUGboat* that describe Asymptote's 2D [1] and 3D [2] typographic capabilities. Some of these advances were developed in preparation for and during TEX's 2^5 anniversary workshop in San Francisco. These improvements are contained in the current release (2.03) of Asymptote.

2 Batching of 3D TEX

A significant improvement was made in the processing of 3D TEX labels: their conversion into surfaces is now batched, resulting in much faster execution.

In two dimensions, Asymptote uses a two-stage system to position TEX labels within a figure. First, a bidirectional TEX pipe is used to query the width, height, and depth of a TEX string. This information is used to align the label within a TEX layer on top of a PostScript background. The PostScript background and TEX layer are linked together within a file that is then fed to TEX for final processing. In other words, in two dimensions all that Asymptote really does is prepare a TEX file, deferring typesetting issues to the external TEX engine.

Since TEX is inherently a two-dimensional program, the above scheme will clearly not work in three dimensions. As described in [2], Asymptote uses a PostScript interpreter to extract Bézier paths from the output of TEX+Dvips (or PDFTEX+Ghostscript). Previously, typesetting each three-dimensional label therefore required executing three external processes, drastically slowing down the processing of three-dimensional figures (particularly under the Microsoft Windows operating system).

In most instances, however, the deferred drawing routines [1, 2] do not need detailed Bézier path data in order to size figures, but only the three-dimensional bounding boxes of each label. The only exception is the case where a three-dimensional label needs to be manipulated (e.g. extruded or transformed), a case that in practice arises infrequently. In all other cases, the bounding box may be computed simply by transforming into three dimensions the two-dimensional bounding box reported *via* the bidirectional TEX pipe (which can process many thousands of TEX strings per second).

This allows the conversion of three-dimensional labels into Bézier paths to be deferred until the final conversion of a three-dimensional picture into a fixed-size frame, from which OpenGL calls or PRC code can then be generated. To distinguish the individual path arrays associated with each TEX string within the generated PostScript code, each string is typeset on a separate page. Batching TEX labels in this manner yields remarkable performance gains (typically a factor of two to five faster, depending on the number of TEX labels and the underlying operating system).

3 Billboard labels

By default, three-dimensional labels now behave like "billboards" that interactively rotate to face the camera (fig. 1). This default can be changed locally or globally:

```
import three;
settings.autobillboard=true; // default
currentprojection=perspective(1,-2,1);
draw(unitbox);
label("Billboard",X,red,Billboard);
label("Embedded",Y,blue,Embedded);
```

[1] Andy Hammerlindl, John Bowman, and Tom Prince, available under the GNU Lesser General Public License from `http://asymptote.sourceforge.net/`

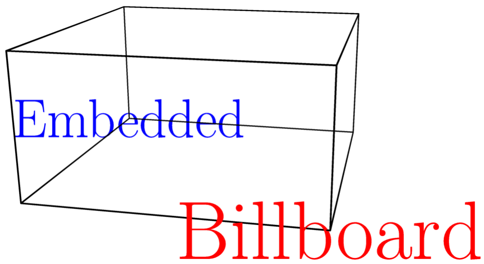


Figure 1: Billboard labels interactively rotate to face the camera, while embedded labels rotate with the picture.

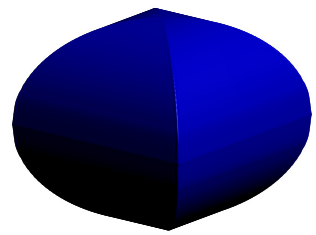
4 Rendering options

Using the latest PRC specification [3], Michail Vidiassov recently overhauled Asymptote's PRC driver, which was originally written by Orest Shardt. The most significant new feature, lossy PRC compression, allows one to produce much more compact 3D PDF files (typically smaller by a factor of two or more). Such specialized rendering options can be specified *via* the structure `render` defined at the beginning of module `three`. The real member `compression` of this structure can be used to set the desired compression value. The real variables `Zero=0.0`, `Low=0.0001`, `Medium=0.001`, and `High=0.01` represent convenient predefined compression values. The default setting, `High`, normally leads to no visible differences in rendering quality. However, when drawing the Bézier approximation to a unit sphere described in [6], PRC compression may create rendering artifacts at the poles and should be disabled:

```
import three;
draw(unitsphere,
     render(compression=Zero,merge=true));
```

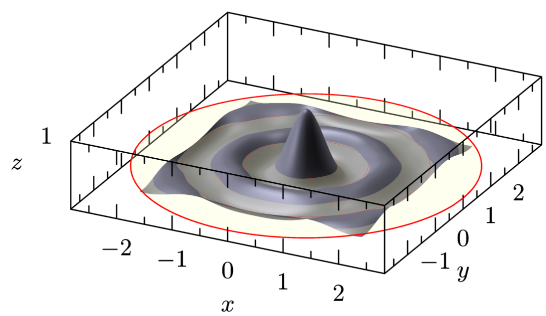
The `merge` argument here is a tri-state boolean variable; the value `true` causes nodes to be merged into a group before Adobe's fixed-resolution rendering mesh is generated. The choice `merge=default` causes only opaque patches to be merged, while the default setting `merge=false` completely disables merging. Patch merging results in faster but lower-quality rendering. It is particularly useful for rendering parametrized surfaces like the volume bounded by two perpendicular unit cylinders centered on the origin:

```
import graph3;
currentprojection=orthographic(5,4,2);
real f(pair z) {
  return min(sqrt(1-z.x^2),sqrt(1-z.y^2));
}
surface s=surface(f,(0,0),(1,1),40,Spline);
transform3 t=rotate(90,O,Z),
           t2=t*t, t3=t2*t;
transform3 i=xscale3(-1)*zscale3(-1);
draw(surface(s,t*s, t2*s,t3*s, i*s, i*t*s,
             i*t2*s, i*t3*s),blue,
     render(compression=Low,closed=true,
            merge=true));
```



This example also illustrates another PRC rendering option, the boolean member `closed`; specifying `closed=true` requests one-sided rendering, whereas the default value `closed=false` requests two-sided rendering.

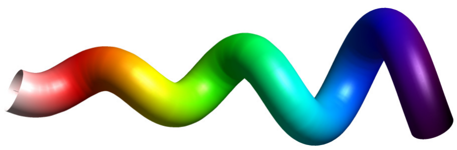
Asymptote now automatically generates a PRC model tree that reflects the object hierarchy, grouping patches together logically, as illustrated in the model tree for the PDF version of the graph below:



One can also manually begin and end a new group called `name` in the 3D model tree for a picture `pic`, using the render options in the structure `render`:

```
void begingroup3(picture pic=currentpicture,
                 string name="",
                 render render=defaultrender);
void endgroup3(picture pic=currentpicture);
```

Various geometric PRC primitives have also been implemented in the current PRC driver. For example, the primitives `drawPRCsphere`, `drawPRCcylinder`, and `drawPRCtube` are useful for drawing and capping compact representations of thick curves (tubes) on a picture. The algorithm for constructing circular tubes was first rewritten to use Oliver Guibé's splined representation of parametrized surfaces, using in the angular direction `splinetype periodic` scaled by $2a$, where the parameter $a = \frac{4}{3}(\sqrt{2}-1)$ is determined by requiring that the third-order midpoint of a cubic Bézier spline lie on the unit circle. This Bézier surface representation is used directly for OpenGL output. For PRC output, it is more efficient to extract from this representation a path that describes the tube center and another path lying on the surface of the tube. These two paths are then passed as arguments to the `drawPRCtube` primitive, which Adobe Reader then renders into a smooth tube, as shown below:



When drawing a three-dimensional dot, the rendering setting `sphere` allows the user to choose between the built-in PRC representation of a sphere (`PRCsphere`) or an efficient NURBS approximation (`NURBSsphere`) to a sphere using 10 distinct control points [4] (the 8-point version discussed in [5] leads to rendering artifacts at the poles). The default, `NURBSsphere`, generates slightly larger files but renders faster than the built-in PRC primitive.

Another new rendering option, which applies to both OpenGL and PRC output, is `labelfill`. Enabled by default, this option allows one to fill subdivision cracks in opaque unlighted (purely emissive) labels, thereby working around artifacts due to the suboptimal algorithms used in Adobe Reader.

5 SVG output

To support web usage, Asymptote now uses Martin Gieseking's excellent `dvisvgm` utility to generate SVG natively (as well as PostScript, PDF, and 3D PRC) vector graphics output. The setting `svgemulation` may be enabled to emulate unimplemented SVG features like Gouraud and tensor-patch shading; otherwise such elements will be replaced by PNG images.

6 Latexmk support

The latest version (1.18) of the `asymptote.sty` package, which allows one to embed Asymptote commands within a LaTeX file, supports both global and local values for the `inline` and `attach` options. It supports John Collins' excellent `latexmk` Perl script for updating only those figures that have changed since the last compilation. One may also specify an `\asydir` subdirectory for Asymptote figures.

7 Conclusion

The Asymptote enhancements described in this article have greatly increased the speed and usability of Asymptote, especially for large documents that contain many three-dimensional figures. They are the result of collaborations among many Asymptote users. In particular, I would like to acknowledge Andy Hammerlindl for designing and implementing much of the underlying Asymptote language, Orest Shardt and Michail Vidiassov for their exceptional work on the PRC driver, Olivier Guibé for his implementation of splined parametric surfaces, Philippe Ivaldi for his implementation of rotation-minimizing frames [7], and Will Robertson and Herbert Schulz for discussions at the TUG 2010 workshop regarding `asymptote.sty`. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada.

References

[1] John C. Bowman and Andy Hammerlindl. Asymptote: A vector graphics language. *TUGboat: The Communications of the TeX Users Group*, 29(2):288–294, 2008.

[2] John C. Bowman and Orest Shardt. Asymptote: Lifting TeX to three dimensions. *TUGboat: The Communications of the TeX Users Group*, 30(1):58–63, 2009.

[3] ISO/TC171/SC2. 3D use of product representation compact (PRC) format, 2009. `http://pdf.editme.com/files/PDFE/SC2N570-PRC-WD.pdf`.

[4] Kaihuai Qin. Representing quadric surfaces using NURBS surfaces. *Journal of Computer Science and Technology*, 12(3):210–216, 1997.

[5] Kaihuai Qin, Zesheng Tang, and Wenping Wang. Representing spheres and ellipsoids using periodic NURBS surfaces with fewer control vertices. *Computer Graphics and Applications, Pacific Conference on*, 0:210, 1998.

[6] Orest Shardt and John C. Bowman. Surface parametrization of nonsimply connected planar Bézier regions. *Submitted to Computer-Aided Design*, 2010.

[7] Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. Computation of rotation minimizing frames. *ACM Trans. Graph.*, 27(1):1–18, 2008.

◇ John C. Bowman
Dept. of Mathematical and Statistical Sciences
University of Alberta
Edmonton, Alberta
Canada T6G 2G1
`bowman (at) math dot ualberta dot ca`
`http://www.math.ualberta.ca/~bowman/`