Graphics

Asymptote: Lifting TEX to three dimensions

John C. Bowman and Orest Shardt

Abstract

Asymptote, a modern successor to the METAPOST vector graphics language that features robust floating-point numerics, high-order functions, and deferred drawing, has recently been enhanced to generate fully interactive three-dimensional output. This data can either be viewed with Asymptote's native OpenGL-based renderer or internally converted to Adobe's highly compressed PRC format for embedding within a PDF file. Asymptote thus provides the scientific community with a self-contained and powerful TEX-aware facility for generating portable interactive three-dimensional PDF files.

1 Introduction

The descriptive vector graphics language Asymptote[1] was developed to provide a standard for drawing mathematical figures, just as TEX and LATEX have become the standard for typesetting equations in the mathematics, physics, and computer science communities [1]. For professional quality and portability, Asymptote natively generates PostScript, PDF, and PRC vector graphics output. The latter is a highly compressed 3D format that is typically embedded within a PDF file and viewed with Adobe Reader.

In both two and three dimensions, consistent fonts and equations should be used in the graphics and text portions of a document. This implies that labels must be typeset directly by TEX. This article provides an overview of the major advances in the current version (1.82) of Asymptote that allow it to extract and lift Bézier font descriptions generated by TEX and Dvips into 3D, using efficient algorithms for partitioning planar regions into nondegenerate Coons patches [3]. Together with 3D generalizations of the METAFONT path operators and a method for computing twist-free tubes and arrowheads, these algorithms provide the 3D foundation of Asymptote.

2 Bézier surfaces

A major recent advance in Asymptote is the ability to embed Bézier surfaces as interactive PRC content within a PDF file, as illustrated in Fig. 1.[2] In contrast, the version of U3D supported by Adobe can only render surfaces described by polygons and hence is not a suitable vector graphics format.

Figure 1: An interactive 3D PDF of a Bézier surface representation of the Utah Teapot.

3 Three-dimensional TEX

TEX produces output in a special device independent format (DVI). While this output can be easily turned into PostScript, one needs a way of extracting Bézier curves that describe properly kerned font characters. Asymptote does this by overloading the PostScript `/show` operator, as described in Appendix A. Special care was required to handle the filled rectangles that TEX uses to draw square root symbols and fraction bars. The resulting exact 2D vector representation of the original TEX input is treated by Asymptote as an array of paths to be filled with the PostScript nonzero winding number fill rule.

The routine `bezulate` described in Figs. 2 and 3, along with the nondegenerate patch splitting algorithms described in [3], is used to convert the resulting Bézier paths to Bézier surfaces. These surfaces are then output in the PRC format, along with a rendered preview image for noninteractive viewing and printing. Using these techniques, Asymptote is then able to typeset the Gaussian integral in Fig. 4 as an interactive 3D diagram.

4 Thick lines in 3D

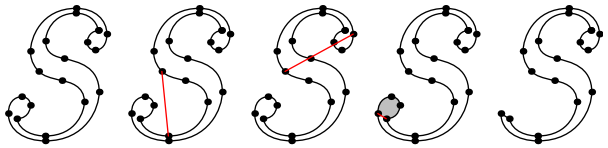
Figure 5 depicts capped thick lines and Asymptote's five (METAPOST-inspired) path connectors [2]:

```
-- .. & --- ::
```

for the following path, when lifted to the x–y plane:

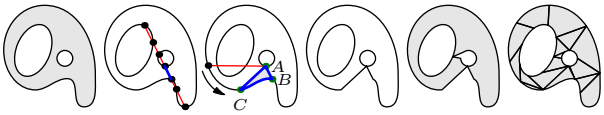
[1] Andy Hammerlindl, John Bowman, and Tom Prince, available under the GNU Lesser General Public License from `http://asymptote.sourceforge.net/`.

[2] An interactive PDF version of this article may be found at `http://asymptote.sourceforge.net/articles/`.



(a) (b) (c) (d) (e)

Figure 2: The `bezulate` algorithm. Starting with the original curve (a), several possible connections between nodes separated by 3 or 2 segments are tested. Connections are rejected if they do not lie entirely inside the original curve. This occurs when the midpoint is not inside the curve (b), or when the connecting line segment intersects the curve more than twice (c). If a connecting line passes both tests, the shaded section is separated (d) and the algorithm continues with the remaining path (e).



(a) (b) (c) (d) (e) (f)

Figure 3: Splitting of non-simply connected regions into simply connected regions. Starting with a non-simply connected region (a), the intersections between each curve and an arbitrary line segment from a point on an inner curve to the outer curve are found (b). Consecutive intersections of this line segment, at points A and B, on the inner and outer curves, respectively, identify a bounded region. Such a region can be found by searching along the outer curve for a point C such that the line segment AC intersects the outer curve no more than once, intersects an inner curve only at A, and determines a region ABC between the inner and outer curves that does not contain an inner curve. Once such a region is found (c), it is extracted (d). This extraction merges the inner curve with the outer curve. The process is repeated until all inner curves have been merged with the outer curve, leaving a simply connected region (e) that can be split into Bézier surface patches. The resulting patches and extracted regions are shaded in (f).

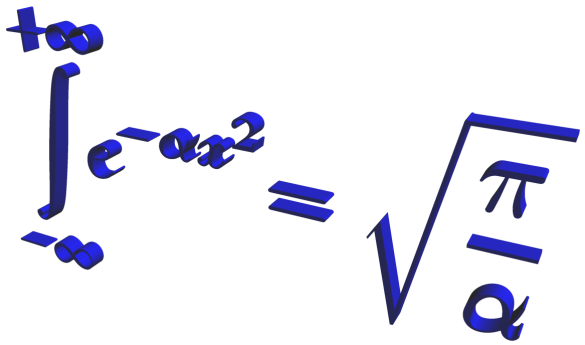


Figure 4: The Gaussian integral lifted to 3D.

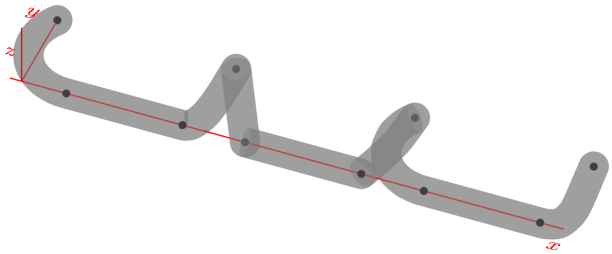


Figure 5: Interactive 3D diagram illustrating thick capped lines, opacity, and the five Asymptote path connectors.

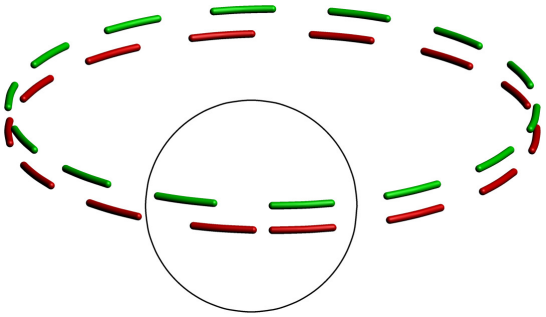


Figure 6: Comparison of arc length adjusted (green) and unadjusted (red) 3D dashed lines.

```
(0,10)..(5,0)---(18,0)::{(0,1)}(20,10)
&(20,10)..(25,0)--(38,0)::{(0,1)}(40,10)
&(40,10)::(45,0)---(58,0)..{(0,1)}(60,10).
```

Hemispheres are aligned at discontinuous junctions of Bézier segments. Disks, hemispheres, or closed cylinders can be used to cap the ends of a Bézier curve, according to the specified PostScript line cap.

Just as in 2D, the on-off duty cycle pattern for generating dashed lines can be automatically adjusted slightly to fit the path arc length evenly, as illustrated in Fig. 6.

A modification of Asymptote's adaptive thick line routine, contributed by Philippe Ivaldi and based on the rotation minimizing frame algorithm described by Wang [4], can be used to construct a tube of arbitrary (noncircular) cross section. For example, Fig. 7 was created by rotating the Greek letter π along a curve describing a trefoil knot.

Jens Schwaiger used similar methods to design a 3D version of Asymptote's `labelpath` function for typesetting text along curves and surfaces, as illustrated in Fig. 8.

5 Arrowheads in 3D

Arrows are frequently used in illustrations to draw attention to important features. We designed curved

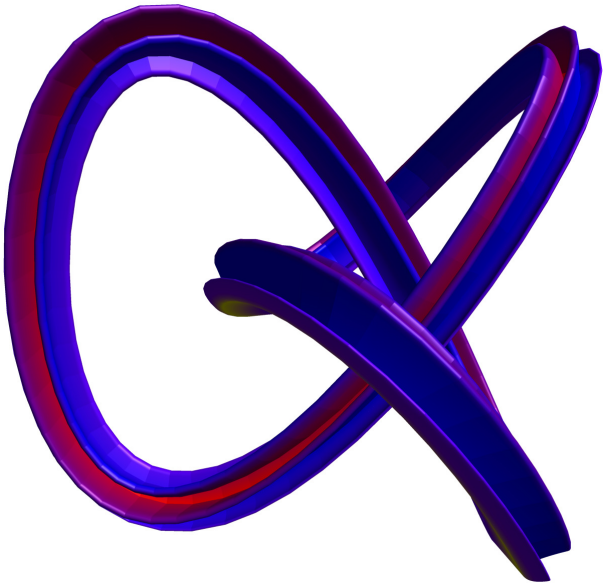


Figure 7: A trefoil knot drawn with Asymptote's arbitrary cross section `tube` module.

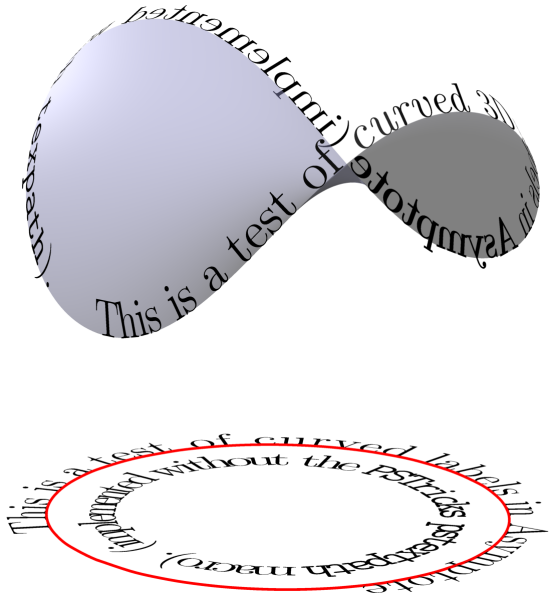


Figure 8: Illustration of curved labels drawn with the `labelpath3` module.

3D arrowheads that can be viewed from a wide range of angles. For example, the default 3D arrowhead was formed by bending a cone around the tip of a Bézier curve using the same algorithm as is used for constructing thick lines. Planar arrowheads derived from 2D arrowhead styles are also implemented; they are oriented by default on a plane perpendicular to the initial viewing direction. Examples of these arrows are displayed in Figs. 9 and 10. An engineering drawing that uses planar arrows is displayed in Fig. 11.

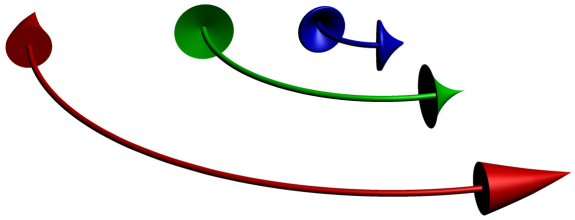


Figure 9: Predefined 3D revolved arrowheads: (blue) `TeXHead3`; (green) `HookHead3`; (red) `DefaultHead3`.

6 Double deferred drawing

Journal size constraints typically dictate the final width and height, in PostScript coordinates, of a 2D or projected 3D figure. However, it is often convenient for users to work in more physically meaningful coordinates. This requires *deferred drawing:* a graphical object cannot be drawn until the actual scaling of the user coordinates (in terms of PostScript coordinates) is known [1]. One queues a function to do the drawing only once the overall scaling is known. Asymptote's high-order functions provide a flexible automatic sizing mechanism: either or both of the 3D model dimensions and the final projected 2D size may be specified. This requires two levels of deferred drawing, a first pass to size the 3D model and a second pass to scale the resulting picture to fit the 2D size specification.

Deferred drawing allows one to draw a fixed-sized object at a scaled coordinate. The following code shows how to draw circles with 5mm radii at each vertex of a unit cube, independent of the overall picture scaling (cf. Fig. 12):

```
import three;
size(4cm);
currentprojection=orthographic(5,4,2);

void Circle(triple c, pen p) {
  picture pic;
  draw(pic,scale3(5mm)*unitcircle3,p);
  add(pic,c);
}

path3[] g=unitbox;
draw(g);

for(path3 p : g)
  for(int i=0; i < length(p); ++i)
    Circle(point(p,i),red);
```

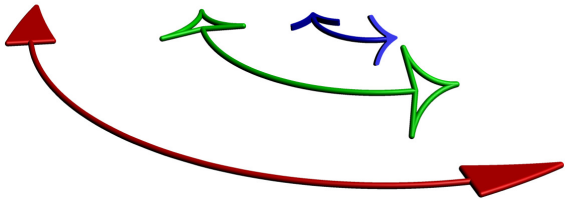


Figure 10: Predefined planar curved arrowheads: (blue) `TeXHead2`; (green) `HookHead2`; (red) `DefaultHead2`.

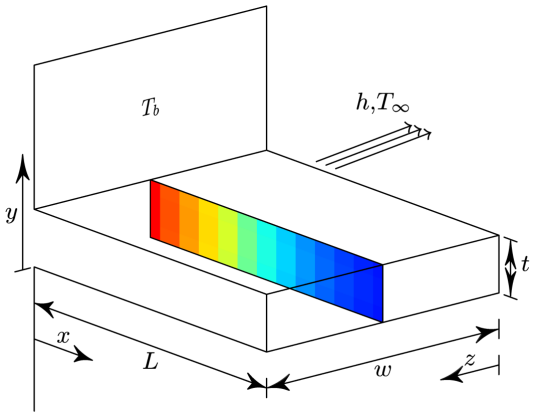


Figure 11: Temperature distribution in a cross section of a heat fin.

7 Interactive 3D Graphs

An important application of 3D TEX is in scientific graphing. The following code generates the interactive 3D surface in Fig. 13.

```
import graph3;
import grid3;
import palette;

currentprojection=orthographic(0.8,0.7,1.5);
size(225pt,0,IgnoreAspect);

real f(pair z) {
  return cos(2pi*z.x)*sin(2pi*z.y);
}
surface s=surface(f,(-1/2,-1/2),(1/2,1/2),20,
                  Spline);
draw(s,mean(palette(s.map(zpart),Rainbow())),
     black);
xaxis3(Label("$x$",0.5),Bounds,InTicks);
yaxis3(Label("$y$",0.5),Bounds,InTicks);
zaxis3(Label("$z$",0.5),Bounds,-1,1,
       InTicks(trailingzero));
grid3(XYZgrid);
```

In Fig. 14, a 3D interactive plot of the surface of the function $\Gamma(z) = \int_{0^+}^{\infty} e^{-t}t^{z-1}\,dt$, extended analytically to the complex plane, emphasizes its poles at the origin and at negative integers. This was produced with the Asymptote code:

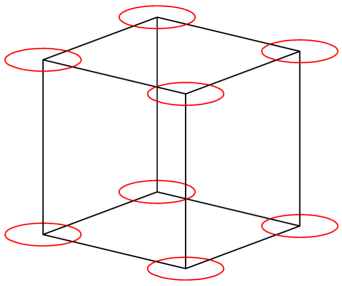


Figure 12: Example of double deferred drawing.

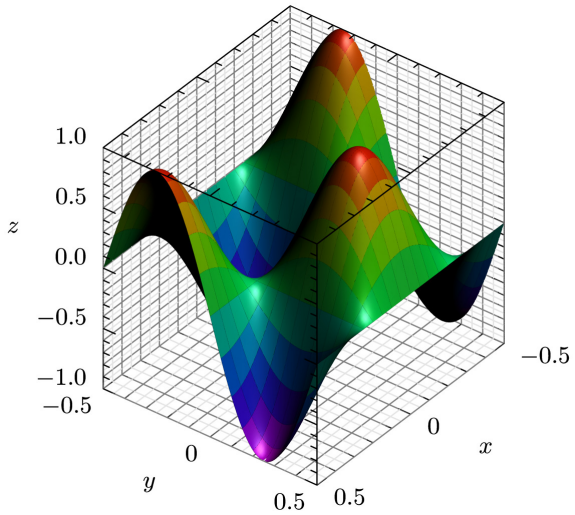


Figure 13: An interactive surface plot with elevation coloring.

```
import graph3;
import palette;
size(225pt,0,IgnoreAspect);
currentprojection=orthographic(1,-1.8,1);

real X=4.5; real M=abs(gamma((X,0)));
pair Gamma(pair z) {
  return (z.x > 0 || z != floor(z.x)) ?
    gamma(z) : M;
}
real f(pair z) {return min(abs(Gamma(z)),M);}

surface s=surface(f,(-2.1,-2),(X,2),60,Spline);

real Arg(triple v) {
  return degrees(Gamma((v.x,v.y)),warn=false);
}

s.colors(palette(s.map(Arg),Wheel()));
draw(s);
```

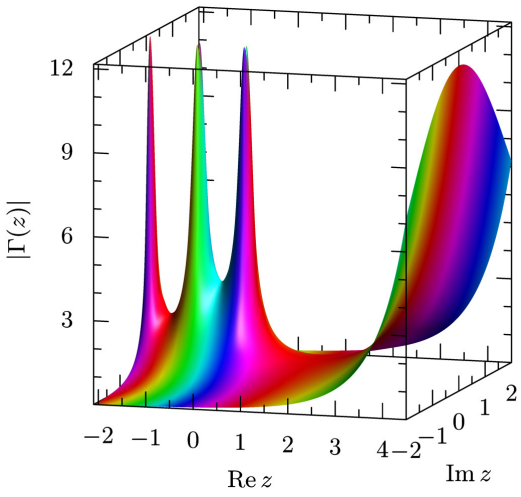
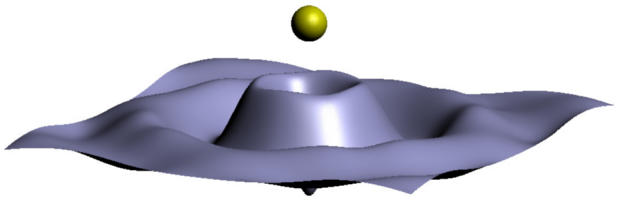


Figure 14: Surface plot of $\Gamma(z)$ in the complex plane, using an RGB color wheel to represent the phase. Red indicates real positive values.

```
xaxis3("$\mathop{\rm Re} z$",Bounds,InTicks);
yaxis3(Label("$\mathop{\rm Im} z$",align=Y-3Z),
       Bounds,InTicks);
zaxis3(rotate(90)*"$|\Gamma(z)|$",
       Bounds,InTicks);
```

8 Inline 3D PDF animations

Inline 3D PDF movies like the one below can be embedded with the help of the LaTeX `animate.sty` package. Unlike 2D inline PDF movies, each frame of a 3D movie is currently pre-rendered by Asymptote to a specified resolution, in order to resolve hidden surfaces correctly.



9 Future directions

There are still a number of applications (including the above animation) where vector PostScript or non-interactive PDF output of 3D scenes would be desirable. For example, Adobe Reader currently cannot generate and print high-resolution renderings of 3D objects.

PostScript is a 2D language that supports only Bézier splines and surfaces, which are shape invariant under affine (orthographic) projection but not perspective projection. In contrast, nonuniform rational B-splines are invariant even in the presence of perspective distortion since they are Bézier curves in a projective space described by homogeneous coordinates. Although PostScript is only a 2D language, vector graphics projections of Bézier surfaces are nevertheless possible using tensor product patch shading and hidden-surface splitting along approximations to the visible surface horizon.

We plan to implement partial prerendering of 3D manifolds to allow 3D scenes to be described within a 2D language like PostScript, without giving up on a vector (scalable) description. The idea is to extend Asymptote's 3D picture structure to segment and sort Bézier surfaces to resolve hidden surfaces correctly in the projected PostScript output. This will require the development of new algorithms for approximating intersections of Bézier surfaces and curves with each other. In collaboration with Troy Henderson and L. G. Nobre, we also plan to investigate techniques for optimally approximating nonuniform rational B-splines by Bézier curves through the addition of new control points. This will allow 2D projections of Bézier curves and surfaces to be well described as vector graphics objects in PostScript.

In the near future, we plan to provide JavaScript support for stationary billboards that always face the camera, as well as PRC animations.

As an aside, let us return to the issue regarding implicit equation solving raised in [1]. Unlike METAFONT and METAPOST, Asymptote does not currently have the notion of a `whatever` unknown. It was pointed out in [1] that the most common uses of `whatever` in METAPOST are probably more clearly written using explicit functions like `extension`. One METAPOST user recently asked us whether there is an elegant way to construct the circumscribed circle of a triangle, centered at the intersection point of two perpendicular bisectors. Indeed, the METAPOST code:

```
beginfig(1)
  path tri;
  u := 1in;
  tri := (origin--(1,0)--(2,1)--cycle) scaled u;
  z0 = (point 0.5 of tri) + whatever *
         (direction 0.5 of tri rotated 90);
  z0 = (point 1.5 of tri) + whatever *
         (direction 1.5 of tri rotated 90);
  dotlabel(btex etex, z0);
  draw fullcircle scaled
    (2*abs(z0-point 0 of tri)) shifted z0;
  draw tri withcolor red;
endfig;
end
```

can be written elegantly in Asymptote:

```
unitsize(1inch);
```

```
path tri=(0,0)--(1,0)--(2,1)--cycle;
pair z1=point(tri,0.5);
pair z2=point(tri,1.5);
pair z0=extension(z1,z1+I*dir(tri,0.5),
                  z2,z2+I*dir(tri,1.5));
dot(z0);
draw(circle(z0,abs(z0-point(tri,0))));
draw(tri,red);
```

Perhaps this example will help motivate hesitant METAPOST users to migrate to Asymptote, allowing them to take full advantage of the powerful interactive 3D functionality described in this article.

10 Conclusions

We believe that Asymptote is the first software package to lift TEX into 3D. It also provides a self-contained open source tool for producing portable 3D PDF files that support Bézier surfaces. As illustrated in the examples we have provided, these are important features for publication-quality scientific graphing. Interactivity is critical for visualization and mental reconstruction of 3D data, as it helps the human brain resolve the degeneracy inherent in 2D projection.

11 Credits

We thank Philippe Ivaldi, Radoslav Marinov, Malcolm Roberts, Jens Schwaiger, and Olivier Guibé for discussions related to this work. Special thanks goes to Andy Hammerlindl, who designed much of the underlying Asymptote language. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada.

A Extracting Bézier curves from TEX

We now describe the PostScript code used to extract smooth font descriptions from Dvips output. First, a PostScript procedure is defined to output a coordinate:

```
/ASYo {( ) print 12 string cvs print} bind def
```

The PostScript `/show` operator can then be overloaded, using the `pathforall` operator to obtain the coordinates of the Bézier control points:

```
/show {currentpoint newpath moveto false charpath
{( moveto) print ASYo ASYo}
{( lineto) print ASYo ASYo}
{( curveto) print ASYo ASYo ASYo ASYo ASYo ASYo}
{( closepath) print}
pathforall} bind def
```

The filled rectangles that TEX and Dvips use to draw square root symbols and fraction bars are extracted by overloading the `/v` procedure:

```
/v {neg exch 4 copy 4 2 roll 2 copy 6 2 roll
    2 copy
( moveto) print ASYo ASYo
( lineto) print ASYo add ASYo
( lineto) print add ASYo add ASYo
( lineto) print add ASYo ASYo
( closepath) print} bind def
```

This technique was used to form the TEX characters in the 3D Asymptote logo in Fig. 15.

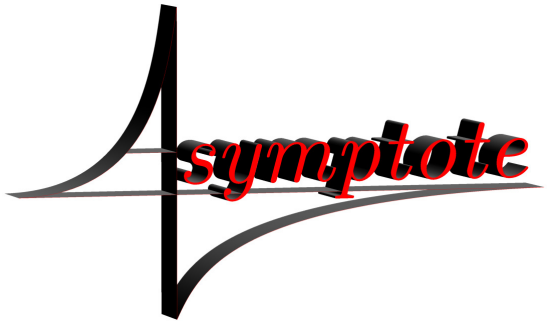


Figure 15: The Asymptote logo in three dimensions.

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] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[3] Orest Shardt and John C. Bowman. Three-dimensional vector representations of nonsimply connected planar surfaces. *Submitted to ACM Trans. Graph.*, 2009.

[4] 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/

◇ Orest Shardt
Dept. of Chemical and Materials Engineering
University of Alberta
Edmonton, Alberta
Canada T6G 2V4
shardt (at) ualberta dot ca