Three-dimensional graphics with PGF/Ti*k*Z

Keith Wolcott

Abstract

PGF and Ti*k*Z are languages for creating graphics. These packages are predominantly two-dimensional graphics packages, so three-dimensional graphing is more challenging, but still possible. A demonstration of how to draw surfaces of revolution, satellite orbits, and intersections of spheres is given. As is typical with three-dimensional graphics, the technique is to rotate in three-space and then project to the drawing surface. The mathematics involved is discussed and sample code is provided.

1 Introduction

PGF (Portable Graphics Format) is a lower-level language and Ti*k*Z is a set of higher-level macros that use PGF. Ti*k*Z is a recursive acronym for "Ti*k*Z ist kein Zeichenprogramm" or "Ti*k*Z is not a drawing program". These languages were created by Till Tantau [1]. PGF and Ti*k*Z commands are invoked as (LA)TEX macros.

Ti*k*Z is packed with features for two-dimensional drawings of lines, circles, ellipses, paths, graphs, etc. The PGF/Ti*k*Z manual [1] has many examples to facilitate learning these languages. Andrew Mertz and William Slough [2] have written a very nice sequence of examples which is an excellent way to get started using Ti*k*Z.

The graph in figure 1 is an example of a function graphed using PGF and Ti*k*Z.

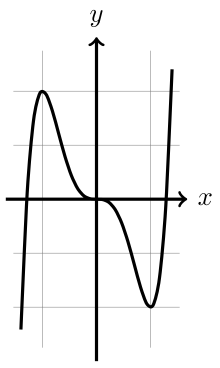


Figure 1: $f(x) = 3x^5 - 5x^3$.

The following code for this figure, and all of the examples in this paper, can be run by cutting-and-pasting into a LATEX document of this form (be sure to use at least a 2011 version since there are compatibility issues with earlier versions of PGF/Ti*k*Z):

```
\documentclass[12pt]{article}
\usepackage{tikz}
\usepackage{ifthen}\newboolean{color}
\begin{document}
% insert code here
\end{document}
```

Here is the code to generate figure 1:

```
\begin{figure}[H]
\centering
% Set the x = a and x = b values of the domain here
% where a <= x <= b.
\def\aDomain{-1.4}
\def\bDomain{1.4}
% Set min and max values of the function
% (c <= f(x) <= d). Used for the y-axis.
\def\cRange{-2.5}
\def\dRange{2.5}
\pgfmathsetmacro\scaleAttempt{2/(\bDomain-\aDomain)}

\begin{tikzpicture}[scale= \scaleAttempt,
   domain= \aDomain : \bDomain]
\draw[very thin,color=gray]
   (1.1*\aDomain,1.1*\cRange)
   grid (1.1*\bDomain, 1.1*\dRange);
\draw[very thick, ->] (1.2*\aDomain, 0) --
   (1.2*\bDomain, 0) node[right] {$x$};
\draw[very thick, ->] (0, 1.2*\cRange) --
   (0, 1.2*\dRange) node[above] {$y$};
\draw[smooth, very thick]
   plot (\x, 3*\x^5 - 5*\x^3);
\end{tikzpicture}

\caption{$f(x) = 3x^5 - 5x^3$.}
\end{figure}
```

2 Three-dimensional graphing

The graph in figure 2 is rotated about the y-axis. The code for the figure is given in appendix A.

First we give additional examples, and then an explanation of the methods used. To create additional examples, use the eleven parameters that are

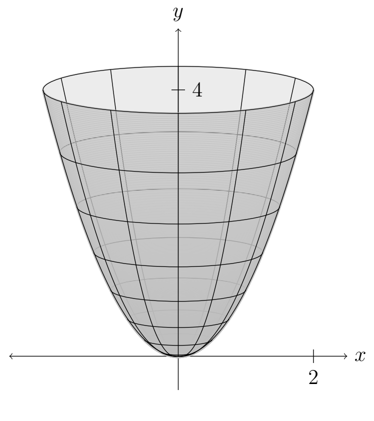


Figure 2: $f(x) = x^2$ rotated about the y-axis.

set at the beginning of the code. They are the domain (min and max), function, range (min and max), back color, front color, shading steps, `xGridSteps`, `rotationGridSteps`, and the viewing angle.

Changing the following four parameters and the figure title results in figure 3.

```
\def\bDomain{6.3}
\def\fcn{cos(\x r)} % The r means to use radians.
\def\cRange{-1}
\def\dRange{1}
```

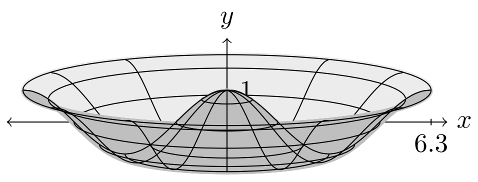


Figure 3: $f(x) = \cos x$ rotated about the y-axis.

We may not like the back and front colors so we change them both to `lightgray`. We also change the viewing angle from 10 to 15 degrees. To give some idea of how other changes can be made, we enlarge it and also comment out the x-axis and the axis number labels. The altered lines of code for figure 4 are:

```
\def\backColor{lightgray}
\def\frontColor{lightgray}

\def\phi{15} % Viewing angle of 15 degrees.

\pgfmathsetmacro\scaleAttempt{6/\bDomain}% 6 was 4.

%\draw[<->] (-\bDomain -.5, 0) -- (\bDomain + .5, 0)
%   node[right] {$x$};
%\draw (\bDomain, .1) -- (\bDomain, -.1)
%   node[below] {\bDomain};
%\pgfmathsetmacro\yLabel {cos(\phi)* \dRange}
%\draw (-.1, \yLabel) -- (.1, \yLabel)
%   node[right] {\dRange};
```

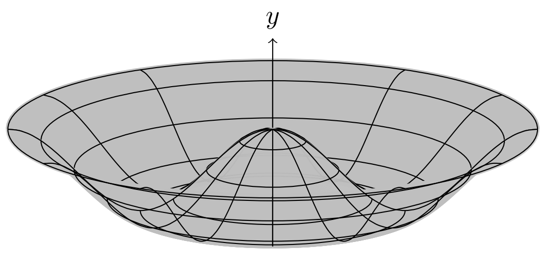


Figure 4: $f(x) = \cos x$ rotated about the y-axis.

Figure 5 is another example.

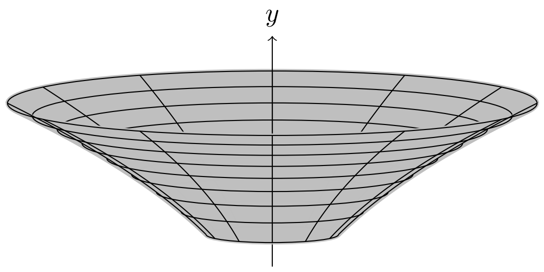


Figure 5: $f(x) = 2\sqrt{x}$, from 1 to 4, rotated about the y-axis.

Figure 6 is figure 5 rotated toward the viewer by changing the view angle from 7 to 30 degrees.

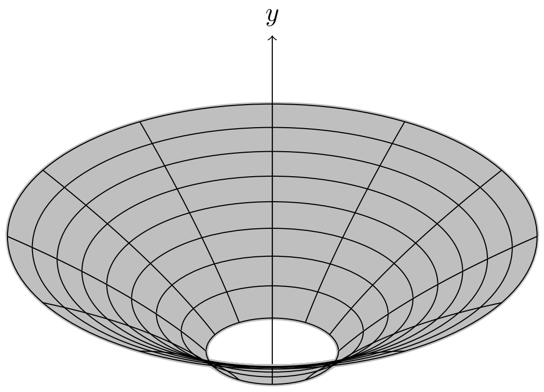


Figure 6: $f(x) = 2\sqrt{x}$, from 1 to 4, rotated about the y-axis, with a view angle of 30 degrees.

With similar code we rotate about the x-axis.

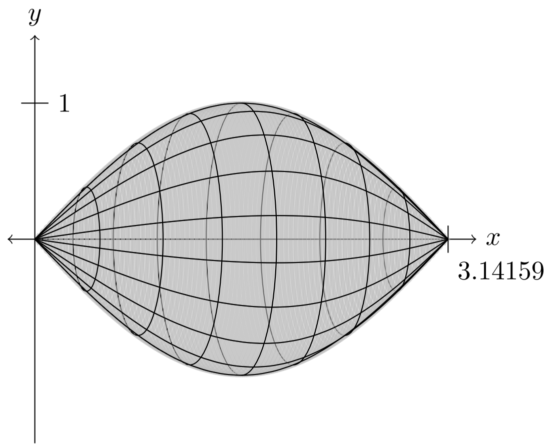


Figure 7: $f(x) = \sin x$ rotated about the x-axis.

The code for figure 7 is included in appendix B.

3 The mathematics behind the scenes

This project began with the goal of drawing two spheres and their circle of intersection. A Google search turned up Tomasz M. Trzeciak's [3] beautiful spheres (see figure 8). He very effectively draws spheres, drawing the latitude and longitude curves by creating a circle, rotating it in three-space, and then projecting to the xy-plane.

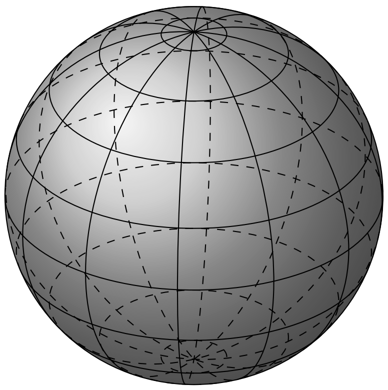


Figure 8: Beautiful sphere created by Tomasz M. Trzeciak.

We will show how to use these methods to draw the surface of revolution in figure 2.

The drawing surface is the xy-plane with the x-axis pointing to the right and the y-axis pointing up. The z-axis points toward the viewer. The rotation matrices about the x, y, and z-axes at an angle θ in the counterclockwise direction are:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix},$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix},$$

and

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

See [4] to learn about rotation matrices.

First we draw the latitude lines, which are circles with radius x drawn at height $f(x)$. To do this, draw a circle centered at the origin with radius x, but give Ti*k*Z the transformation of the plane that rotates it to give the correct viewing angle and shifts it to height $f(x)$. To find the affine transformation of the plane that does this, the following matrix rotates three-space counter-clockwise around the x-axis. Since we start with a circle in the xy-plane we need to rotate it $90 + \phi$ degrees.

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

Next, project to the xy-plane by removing the third row and third column. This gives the two-by-two matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & \cos\phi \end{pmatrix}$$

that is the transformation of the xy-plane that we tell Ti*k*Z to apply to our circle.

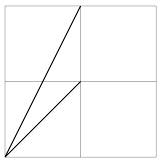
Remark: On page 253 of the PGF manual [1] version 2.10, it says that

```
\tikzset{xyplane/.estyle={cm={
   a,b,c,d,(e,f)}}}
```

uses the transformation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}.$$

For example:



```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (2,2);
\draw (0,0) -- (1,1);
\draw[cm={1,1,0,1,(0,0)}] (0,0) -- (1,1);
\end{tikzpicture}
```

The transformation should map the point (1, 1) to (2, 1) rather than (1, 2) as shown in the figure. Thus, Ti*k*Z (or the underlying PGF) is in fact using

$$(x \quad y) \begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}.$$

Since our rotation matrices operate on vectors on the left, this is the same as operating on the right with the transpose of the matrix. For a rotation matrix, the transpose of the matrix is the inverse of the matrix, which means that the direction of rotation is reversed. Thus, earlier we said that the rotation matrix $R_x(\psi)$ rotates counterclockwise, but for this application, it rotates clockwise.

In conclusion, the rotations are clockwise, and if we wish to do a sequence of rotations, the first rotation matrix is on the left.

Each circle with radius x needs to be shifted up to the correct height $f(x)$, but because of the viewing angle rotation ϕ, this gets foreshortened to

$f(x)\cos\phi$. The following code does this for 9 circles where x is each quarter unit from 0 to 2. (The ninth is an invisible point at 0.)

```
\def\fcn{\x^2}
\def\phi{-10}
\foreach \x in {0, .25, ..., 2} {
    \pgfmathsetmacro\yshift{(cos(\phi))*(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       1, 0, 0, cos(90 + \phi),(0, \yshift)}}}
    \draw[xyplane, color=black] (0, 0) circle (\x);
}
```

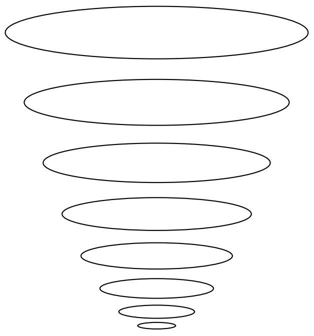


Figure 9: $f(x) = x^2$ rotated about the y-axis.

To draw the longitude lines we draw the given function $f(x)$ from 0 to 2, rotate every 30 degrees, and then change the viewing angle by ϕ degrees. Thus we use $R_y(\theta)R_x(\phi) =$

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} =$$

$$\begin{pmatrix} \cos\theta & \sin\theta\sin\phi & \sin\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}.$$

Next, project to the xy-plane by removing the third row and third column, resulting in the 2×2 matrix

$$\begin{pmatrix} \cos\theta & \sin\theta\sin\phi \\ 0 & \cos\phi \end{pmatrix}.$$

If θ and ϕ are both negated in the above matrix, it does not change. Thus, we will think of the rotations as being in the counterclockwise direction for positive values of θ and ϕ.

This is the transformation of the xy-plane that we tell TikZ to apply to the graph of $f(x)$.

```
\def\fcn{\x^2}
\def\phi{10}
\foreach \theta in {0, 30, ..., 360} {
 \tikzset{xyplane/.estyle={cm={
    cos(\theta),sin(\theta)*sin(\phi),
    0,cos(\phi),(0, 0)}}}
 \draw[xyplane, color=black, thin, smooth]
    plot (\x, \fcn) ;
}
```

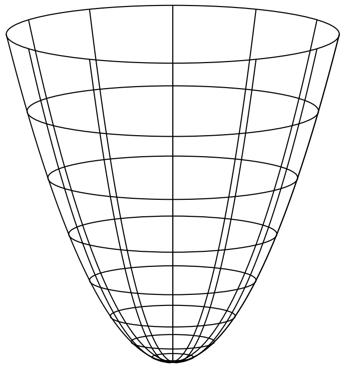


Figure 10: $f(x) = x^2$ rotated about the y-axis.

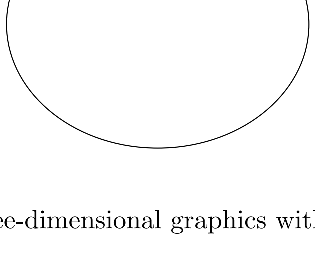
Figure 10 gives the wire frame of the surface of revolution. The remaining code for creating the final version in figure 2 is more of the same. The coloring of the front consists of the front half of many latitude curves (this slows the code down) that typically look good with opacity set to be less than 1 so that the wire frame on the back shows through a little.

4 Global Positioning System (GPS) orbits

As another example, let us draw the orbit of a GPS satellite around the earth. The orbit is inclined 55 degrees from the equator, which is the same as tilted 35 degrees from the north pole. Thus, we can start with a circle in the xy-plane and tilt it 35 degrees toward the viewer. Thus, the 35 degree tilt is a -35 degree clockwise rotation about the x-axis. The rotation matrix about the x-axis at an angle ψ in the clockwise direction is

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix}.$$

Projecting to the xy-plane (removing the third row and third column), we have $\begin{pmatrix} 1 & 0 \\ 0 & \cos\psi \end{pmatrix}$. This is the transformation of the xy-plane that we tell TikZ to apply to our circle as follows. The angle ψ should be -35 degrees since we are tilting the top of the circle toward the viewer.



```
\begin{tikzpicture}
 \tikzset{xyplane/.estyle={cm={
    1,0,0,cos(-35), (0, 0)}}}
 \draw[xyplane, color=black, thin]
    (0, 0) circle (2);
\end{tikzpicture}
```

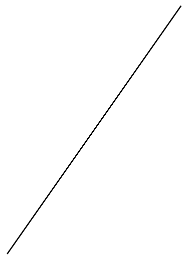
Now we wish to rotate our tilted orbit θ degrees around the y-axis. To rotate first ψ degrees about the x-axis and then θ degrees about the y-axis, we compute

$$R_x(\psi)R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ \sin\psi\sin\theta & \cos\psi & -\sin\psi\cos\theta \\ -\cos\psi\sin\theta & \sin\psi & \cos\psi\cos\theta \end{pmatrix}.$$

Next, project to the xy-plane, which is the transformation

$$\begin{pmatrix} \cos\theta & 0 \\ \sin\psi\sin\theta & \cos\psi \end{pmatrix}.$$

If θ, ψ, and ϕ are all negated in the above matrix, it does not change. Thus, we will think of the rotations as being in the counterclockwise direction for positive values of θ, ψ, and ϕ.



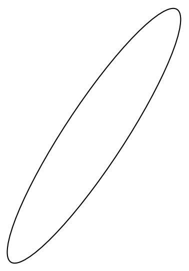
```
\begin{tikzpicture}
 \tikzset{xyplane/.estyle={cm={
    cos(90),0,sin(35)*sin(90),cos(35),(0, 0)}}}
 \draw[xyplane] (0, 0) circle (2);
\end{tikzpicture}
```

Now suppose that we wish to view this from above and thus rotate some angle ϕ around the x-axis to tilt the top of the orbit toward the viewer. This is the product $R_x(\psi)R_y(\theta)R_x(\phi)$ that has the projected matrix

$$\begin{pmatrix} \cos\theta & \sin\theta\sin\phi \\ \sin\psi\sin\theta & \cos\psi\cos\phi - \sin\psi\cos\theta\sin\phi \end{pmatrix}.$$

Again, if θ, ψ, and ϕ are all negated in the above matrix, it does not change. Thus, we will think of the rotations as being in the counterclockwise direction for positive values of θ, ψ, and ϕ.

With $\psi = 35$, $\theta = 90$, and $\phi = 20$, we rotate the top of the above circle (that looks like a line) toward the viewer 20 degrees so it looks like an ellipse again.



```
\begin{tikzpicture}
  \tikzset{xyplane/.estyle={cm={
    cos(90),sin(90)*sin(20),sin(35)*sin(90),
    cos(35)*cos(20)-sin(35)*cos(90)*sin(20),(0,0)}}}
  \draw[xyplane] (0, 0) circle (2);
\end{tikzpicture}
```

As an example of the above rotation and projection matrix, we draw the U.S. GPS (Global Positioning System) satellite orbits. There are six orbits, each tilted $\psi = 35$ degrees from the north pole and then rotated to be positioned every 60 degrees (θ) around the earth. These orbits can then be viewed from different angles ϕ. Figure 11 shows a view from the equator. The code is included in appendix C.

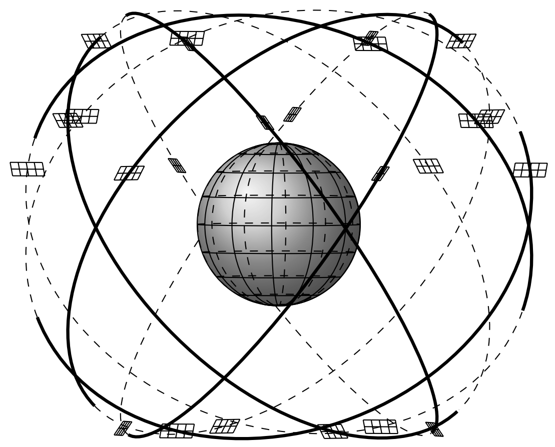


Figure 11: The U.S. GPS system. This view, looking directly at the equator, shows that the orbits never pass over the north or south poles. Each orbit has four satellites spaced 30 degrees, 105 degrees, 120 degrees, and 105 degrees apart.

By changing the viewing angle `\angEl` $= \psi$, we can get additional views. See figures 12 and 13 for two such views.

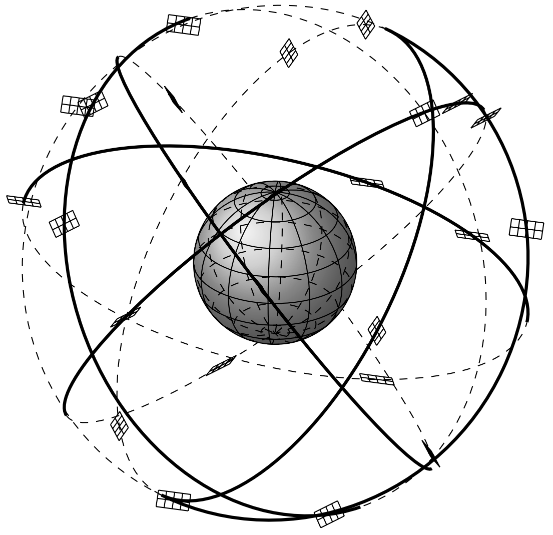


Figure 12: Looking down on the earth, 30 degrees above the equator. Dotted lines are on the back side of the orbit.

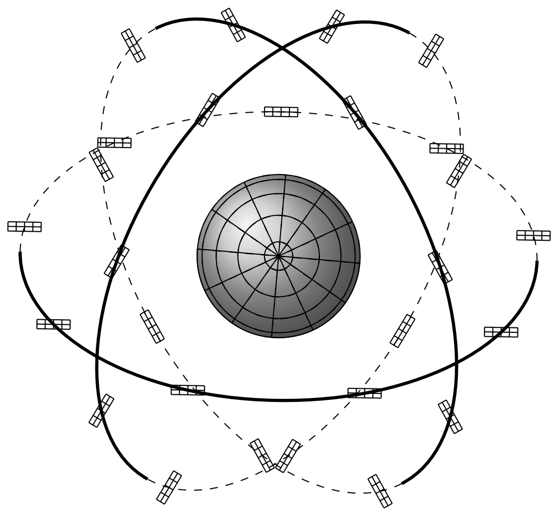


Figure 14: The European Union's planned GPS has three orbital planes, 120 degrees apart, inclined at 56 degrees, that divide the earth's surface into eight congruent spherical triangles. Each of the three orbits has nine satellites, equally spaced, 40 degrees apart.

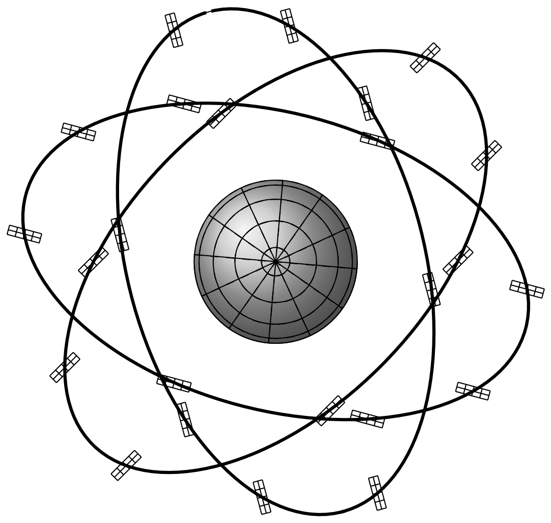


Figure 13: Looking directly down on the north pole. Pairs of orbits overlap in this view, but the front and back parts of these orbits are on opposite sides of the earth.

For comparison, figure 14 is the configuration of satellites that the European Union is using for the GPS system that they are currently implementing that has just three orbits with nine satellites in each.

5 Intersections of spheres

Rotations around an axis other than the coordinate axes can also be useful. Consider figure 15.

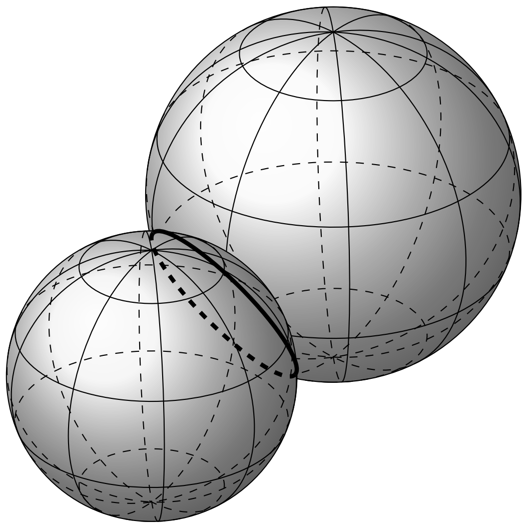


Figure 15: These two spheres intersect in the bold circle.

The goal was to draw the intersection of the two spheres. The plane of the circle of intersection is perpendicular to the vector between the two sphere centers. Thus to find the circle of intersection we rotate three-space so that the z-axis (which is perpendicular to the xy-plane) rotates to be parallel to the vector between the sphere centers. Thus the rotation axis is perpendicular to both of these vectors and we compute it by taking the cross product of them. Then we use the rotation matrix about this vector (see [4]), project to the xy-plane, and then shift to the correct position. This figure needs more work, since it does not show the overlapping parts of the spheres correctly, but it is still an example of how to find and draw the intersection using these techniques. The code for figure 15 is included in appendix E.

6 Conclusion

The PGF and Ti*k*Z languages are predominantly for two-dimensional graphics, but with an understanding of a few rotation matrices, some three-dimensional graphics can be drawn fairly easily.

Appendix A Rotation about the y-axis

Code for figure 2. The function $f(x) = x^2$ is rotated about the y-axis.

```
\begin{figure}[h]
\begin{center}
%%%%%%%%   Set function values  %%%%%%%
% Set the x = a and x = b values of the
% domain here where a <= x <= b.
\def\aDomain{0}
\def\bDomain{2}
% Set the function.
% The variable must be \x, e.g. \x^2.
\def\fcn{\x^2}
% Set min and max values of the function
% (c <= f(x) <= d). Used for the y-axis.
\def\cRange{0}
\def\dRange{4}
% Set the color of the back half.
% This can look good as a different color
% if it looks like the inside.
\def\backColor{brown}
% Set the color of the front half.  lightgray looks
% good for both back and front.
\def\frontColor{red}
% Set the number of shading circles to draw.
% More gives a more even color.
% Enter 1 for no shading.
\def\xShadingSteps{300}
% Set the number of x radius grid circles.
\def\xGridSteps{8}
% Set the number of radial grid lines.
\def\rotationGridSteps{12}
% Set the viewing elevation angle,
% which is the angle up from horizontal.
\def\phi{10}
%%%%%%%%%%%%%%%%%%%%%%%%%%
\pgfmathsetmacro\scaleAttempt{4/\bDomain}
\begin{tikzpicture}[scale= \scaleAttempt,
   domain= \aDomain: \bDomain]
\pgfmathsetmacro\intervalLength{\bDomain - \aDomain}
\pgfmathsetmacro\xGridStepsize{
   \intervalLength/\xGridSteps}
\pgfmathsetmacro\xShadingStepsize{
   \intervalLength/\xShadingSteps}
\pgfmathsetmacro\rotationGridStepsize{
   360/\rotationGridSteps}
% Draw the shading of the back half.
% Top half of a circle, rotated back (around x-axis)
% 90 - \phi degrees and shifted up or down
% to the correct height.
\pgfmathsetmacro\nextShadingStep{
   \aDomain + \xShadingStepsize}
\foreach \x in
   {\aDomain, \nextShadingStep, ..., \bDomain} {
    \pgfmathsetmacro\ysh {(cos(\phi))*(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       1,0,0,cos(90-\phi), (0, \ysh)}}}
    \draw[xyplane,\backColor,ultra thick,opacity=1]
       (\x, 0) arc (0:180:\x);
}
% Back longitude lines.
% Rotates graph around y-axis, then
% projects to xy-plane.
\foreach \theta in
   {0, \rotationGridStepsize, ..., 180} {
 \tikzset{xyplane/.estyle={cm={
    cos(-\theta), sin(-\theta)*sin(-\phi),
    0, cos(-\phi),  (0, 0)}}}
 \draw[xyplane, smooth] plot (\x, \fcn);
}
% Back latitude lines.
% Top half of a circle, rotated back
% (around x-axis) 90 - \phi degrees and
% shifted up or down to the correct height.
\pgfmathsetmacro\nextStep{\aDomain + \xGridStepsize}
\foreach \x in {\aDomain,\nextStep, ...,\bDomain} {
    \pgfmathsetmacro\ysh {(cos(\phi))*(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       1,0,0,cos(90-\phi), (0, \ysh)}}}
    \draw[xyplane] (\x, 0) arc (0: 180:\x);
}
% Draw the axis.
\pgfmathsetmacro\yHeight{
   \dRange + \bDomain*sin(\phi) + .5}
\draw[->] (0, \cRange - .5) -- (0, \yHeight)
   node[above] {$y$};
% Comment out the next four commands
% if you don't want an x-axis, and labels.
\draw[<->] (-\bDomain -.5, 0) -- (\bDomain + .5, 0)
   node[right] {$x$};
\draw (\bDomain, .1) -- (\bDomain, -.1)
   node[below] {\bDomain};
\pgfmathsetmacro\yLabel {cos(\phi)* \dRange}
\draw (-.1, \yLabel) -- (.1, \yLabel)
   node[right] {\dRange};

% Draw the shading of the front half.
% Top half of a circle, rotated back (around x-axis)
% 90 - \phi degrees and shifted up or down
% to the correct height.
\foreach \x in
   {\aDomain, \nextShadingStep, ..., \bDomain} {
    \pgfmathsetmacro\ysh {(cos(\phi))*(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       1,0,0,cos(90-\phi), (0, \ysh)}}}
    \draw[xyplane,\frontColor,ultra thick,
       opacity=.6]
       (-\x, 0) arc (-180:0:\x);
}
% Front longitude lines.
\foreach \theta in
   {0, \rotationGridStepsize, ..., 180} {
 \tikzset{xyplane/.estyle={cm={
    cos(\theta), sin(\theta)*sin(-\phi),
    0, cos(-\phi),  (0, 0)}}}
 \draw[xyplane,smooth] plot (\x, \fcn);
}
% Front latitude lines.
% Bottom half of a circle, rotated back
% (around x-axis) 90 - \phi degrees and
% shifted up or down to the correct height.
\foreach \x in {\aDomain, \nextStep, ..., \bDomain}{
    \pgfmathsetmacro\ysh {(cos(\phi))*(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       1,0,0,cos(90-\phi),(0, \ysh)}}}
    \draw[xyplane] (-\x, 0) arc (-180: 0:\x);
}
\end{tikzpicture}
\caption{Rotate $f(x) = x^2$ about the $y$-axis.}
\end{center}
\end{figure}
```

Remark: Since the back and front shading is drawn by drawing circles that are rotated in three-space, and the above code uses 300 such circles, it is slow (about 3 seconds). Thus, when working with the document, it is useful to set `\xShadingSteps` to 1 and then change it to 300 for the final version.

Appendix B Rotation about the x-axis

Code for figure 7. The function $\sin x$ is rotated about the x-axis.

```
\begin{figure}[h]
\begin{center}
%%%%%%%%   Set function values  %%%%%%%
% Set the x = a and x = b values of the
% domain here where a <= x <= b.
\def\aDomain{0}
\def\bDomain{3.14159}
% Set the function.
% The variable must be \x, e.g. \x^2.
\def\fcn{sin(\x r)}
%\def\fcn{sqrt(\x)}
% Set min and max values of the function
% (c <= f(x) <= d). Used for the y-axis.
\def\cRange{0}
\def\dRange{1}
% Set the color of the back half.
% This can look good as a different color
% if it looks like the inside.
\def\backColor{red!70!black}
% Set the color of the front half.  lightgray looks
% good for both back and front.
\def\frontColor{red!70!black}
% Set the number of shading circles to draw.
% More gives a more even color.  Enter 1 for
% no shading; a large number makes it slow.
% Use the following two lines while editing and then
% change the speed to 100 for the final version.
%\def\speed{1}
%\pgfmathsetmacro\xShadingSteps{3* \speed}
\pgfmathsetmacro\xShadingSteps{300}
% Set the number of x radius grid circles.
\def\xGridSteps{8}
% Set the number of radial grid lines.
\def\rotationGridSteps{18}
% Set the viewing elevation angle,
% which is the angle up from horizontal.
\def\phi{15}
%%%%%%%%%%%%%%%%%%%%%%%%%
\pgfmathsetmacro\scaleAttempt{3.4/\dRange}
\begin{tikzpicture}[scale= \scaleAttempt,
   domain= \aDomain: \bDomain]
\pgfmathsetmacro\intervalLength{\bDomain - \aDomain}
\pgfmathsetmacro\xGridStepsize{
   \intervalLength/\xGridSteps}
\pgfmathsetmacro\xShadingStepsize{
   \intervalLength/\xShadingSteps}
\pgfmathsetmacro\rotationGridStepsize{
   360/\rotationGridSteps}
% Draw the shading of the back half.
% Left half of a circle, rotated right
% (around y-axis) 90 - \phi degrees and
% shifted right or left to the correct height.
\pgfmathsetmacro\nextShadingStep{
   \aDomain + \xShadingStepsize}
\foreach \x in
  {\aDomain, \nextShadingStep, ..., \bDomain} {
    \pgfmathsetmacro\xsh{(cos(\phi))*({\x})}
    \pgfmathsetmacro\rad{(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       cos(\phi - 90), 0,0,1, (\xsh, 0)}}}
    \draw[xyplane,\backColor,ultra thick,opacity=.6]
       (0, \rad) arc (90 : 270 : \rad);
}
% Back longitude lines.
% Rotates graph around y-axis,
% then projects to xy-plane.
\foreach \theta in
   {0, \rotationGridStepsize, ..., 180} {
 \tikzset{xyplane/.estyle={cm={
    cos(\phi), 0, sin(\theta)*sin(\phi),
    cos(\theta),  (0, 0)}}}
 \draw[xyplane,smooth] plot (\x, \fcn) ;
}
% Back latitude lines.
% Left half of a circle, rotated right
% (around y-axis) 90 - \phi degrees and
% shifted right or left to the correct height.
\pgfmathsetmacro\nextStep{\aDomain + \xGridStepsize}
\foreach \x in {\aDomain,\nextStep, ...,\bDomain} {
    \pgfmathsetmacro\xsh{(cos(\phi))*({\x})}
    \pgfmathsetmacro\rad{(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       cos(\phi - 90), 0,0,1,(\xsh, 0)}}}
    \draw[xyplane,black,thin,opacity=1]
       (0, \rad) arc (90 : 270 : \rad);
}
% Draw the axis.
\pgfmathsetmacro\xdim{
   \bDomain + \dRange*sin(\phi) + .5}
\draw[->] (0, -\dRange - .5) -- (0, \dRange + .5)
   node[above] {$y$};
% Comment out the next four commands
% if you don't want an x-axis, and labels.
\draw[<->] (\aDomain -.5, 0) -- (\xdim, 0)
   node[right] {$x$};
\pgfmathsetmacro\xLabel{cos(\phi)*\bDomain}
\draw (\xLabel, .1) -- (\xLabel, -.1)
   node[below right] {\bDomain};
\draw (-.1, \dRange) -- (.1, \dRange)
   node[right] {\dRange};

% Draw the shading of the front half.
% Right half of a circle, rotated right
% (around y-axis) 90 - \phi degrees and
% shifted right or left to the correct height.
\foreach \x in
  {\aDomain, \nextShadingStep, ..., \bDomain} {
    \pgfmathsetmacro\xsh{(cos(\phi))*({\x})}
    \pgfmathsetmacro\rad{(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       cos(\phi - 90),0,0,1,(\xsh, 0)}}}
   \draw[xyplane,\frontColor,ultra thick,opacity=.6]
       (0, -\rad) arc (-90 : 90 : \rad);
}
% Front longitude lines.
\foreach \theta in
   {0, \rotationGridStepsize, ..., 180} {
 \tikzset{xyplane/.estyle={cm={
    cos(\phi), 0,
    sin(\theta)*sin(\phi),cos(\theta),(0, 0)}}}
 \draw[xyplane,smooth] plot (\x, \fcn) ;
}
% Front latitude lines.
% Right half of a circle, rotated right
% (around y-axis) 90 - \phi degrees and
% shifted right or left to the correct height.
\foreach \x in {\aDomain, \nextStep, ..., \bDomain}{
    \pgfmathsetmacro\xsh{(cos(\phi))*({\x})}
    \pgfmathsetmacro\rad{(\fcn)}
    \tikzset{xyplane/.estyle={cm={
       cos(\phi-90),0,0,1, (\xsh, 0)}}}
    \draw[xyplane] (0, -\rad) arc (-90 : 90 : \rad);
}
\end{tikzpicture}
\caption{$f(x)=\sin{x}$ rotated about the $x$-axis.}
\label{rot1x}
\end{center}
\end{figure}
```

Appendix C GPS satellites

Code for figure 11, the U.S. GPS satellite orbits.

```
% GPS satellite orbits.
\begin{tikzpicture}[scale=.77]
\def\R{1.4} % sphere radius
\def\orbitRadius{3.172*\R}
\def\angEl{1} % elevation angle
\def\x{0} % x coordinate of center
\def\y{0} % y coordinate of center
\def\z{0} % z coordinate of center
% First tilt the orbit from the north
% pole (rotate about the x-axis).
\pgfmathsetmacro\psi{35}
% Second, rotate around the y-axis.
\pgfmathsetmacro\firstTheta{-135}
% Third, rotate about the x-axis.
\pgfmathsetmacro\phi{\angEl}
\draw[color=red, fill=blue, opacity=.15]
   (0, 0) circle (\orbitRadius);
% Set the variables, theta, c = color, angVis,
% and \thetaSatShift for each of 6 orbits.
\foreach \theta/\c in{\firstTheta/red,
   \firstTheta+60/blue,
   \firstTheta+2*60/green,
   \firstTheta+3*60/black,
   \firstTheta+4*60/cyan,
   \firstTheta+5*60/brown}{
% Set the drawing plane affine transformation.
\tikzset{xyplane/.estyle={cm={
   cos(\theta),sin(\theta)*sin(\phi),
   sin(\theta)*sin(\psi),cos(\psi)*cos(\phi)-
   sin(\psi)*cos(\theta)*sin(\phi),(0, 0)}}}
    % Draw the back half of the orbit.
    \getFrontArcStartPosition\angle\anglex{
       \psi}{\theta}{\phi}
   \pgfmathtruncatemacro\angleInt{\angle}
    \ifthenelse{\angleInt < -180}
    {\pgfmathsetmacro\angleInt{\angleInt + 360}}
    {}
    \draw[xyplane, dashed, color=\c]
       (\angleInt -180: \orbitRadius)
       arc (\angleInt -180: \angle: \orbitRadius);
}
% Draw the earth.
\tikzset{current plane/.estyle={cm={1,0,0,1,(0,0)}}}
\filldraw[current plane][shift={(\x, \y)}]
   [ball color=blue,opacity=.7] (0,0,0) circle (\R);
\foreach \t in {-80,-60,...,80} {
   \DrawLatitudeCircle[\R]{\t}{\x}{\y}}
\foreach \t in {-5,-35,...,-175} {
   \DrawLongitudeCircle[\R]{\t}{\x}{\y}}
\pgfmathsetmacro\orbitBaseAngle{30}
% Draw the front half of the orbit.
\foreach \theta/\c in {\firstTheta/red,
   \firstTheta+60/blue,
   \firstTheta+2*60/green,
   \firstTheta+3*60/black,
   \firstTheta+4*60/cyan,
   \firstTheta+5*60/brown}{
% Set the drawing plane affine transformation again.
   \tikzset{xyplane/.estyle={cm={cos(\theta),
      sin(\theta)*sin(\phi),sin(\theta)*sin(\psi),
      cos(\psi)*cos(\phi)-sin(\psi)*cos(\theta)*
      sin(\phi),(0,0)}}}
% Draw the front half of the orbit.
\getFrontArcStartPosition\angle\anglex{
   \psi}{\theta}{\phi}
\pgfmathtruncatemacro\angleInt{\angle}
\ifthenelse{\angleInt > 180}
     {\pgfmathsetmacro\angleInt{\angleInt - 360}}
     {}
\draw[xyplane,very thick,color=\c]
   (\angleInt:\orbitRadius) arc
   (\angleInt:\angleInt+180:\orbitRadius);
     % Draw the satellites.
     \foreach \thetaSat in {\orbitBaseAngle,
   \orbitBaseAngle + 30, \orbitBaseAngle + 135,
   \orbitBaseAngle + 255} {
           \pgfmathsetmacro\xsh{
            (7/1)*\orbitRadius*cos(\thetaSat)}
            \pgfmathsetmacro\ysh{
    (7/1)*\orbitRadius*sin(\thetaSat)}
           \draw[xyplane,color=\c,scale=1/7][shift=
              {(\xsh,\ysh)}](-2,-1) grid (2,1);
    }
}
\end{tikzpicture}
```

This code requires some helper functions written by Tomasz M. Trzeciak [3] for drawing spheres. For completeness, these functions are listed below. Place them just before `\begin{document}`.

```
\newcommand\pgfmathsinandcos[3]{%
  \pgfmathsetmacro#1{sin(#3)}%
  \pgfmathsetmacro#2{cos(#3)}%
}
\newcommand\LongitudePlane[3][current plane]{%
  \pgfmathsinandcos\sinEl\cosEl{#2} % elevation
  \pgfmathsinandcos\sint\cost{#3} % azimuth
  \tikzset{#1/.estyle={cm={
     \cost,\sint*\sinEl,0,\cosEl,(0,0)}}}
}
\newcommand\LatitudePlane[3][current plane]{%
  \pgfmathsinandcos\sinEl\cosEl{#2} % elevation
  \pgfmathsinandcos\sint\cost{#3} % latitude
  \pgfmathsetmacro\yshift{\cosEl*\sint}
  \tikzset{#1/.estyle={cm={
     \cost,0,0,\cost*\sinEl,(0,\yshift)}}} %
}
\newcommand\DrawLongitudeCircle[4][1]{
 \LongitudePlane{\angEl}{#2}
 \tikzset{current plane/.prefix style={scale=#1}}
  % angle of "visibility"
 \pgfmathsetmacro\angVis{
    atan(sin(#2)*cos(\angEl)/sin(\angEl))} %
 \draw[shift={(#3, #4)}][current plane]
    (\angVis:1) arc (\angVis:\angVis+180:1);
 \draw[shift={(#3, #4)}][current plane,dashed]
    (\angVis-180:1)arc(\angVis-180:\angVis:1);
}
\newcommand\DrawLatitudeCircle[4][1]{
 \LatitudePlane{\angEl}{#2}
 \tikzset{current plane/.prefix style={scale=#1}}
 \pgfmathsetmacro\sinVis{
    sin(#2)/cos(#2)*sin(\angEl)/cos(\angEl)}
 % angle of "visibility"
 \pgfmathsetmacro\angVis{
    asin(min(1,max(\sinVis,-1)))}
 \draw[shift={(#3, #4)}][current plane]
    (\angVis:1) arc (\angVis:-\angVis-180:1);
 \draw[shift={(#3, #4)}][current plane,dashed]
    (180-\angVis:1)arc(180-\angVis:\angVis:1);
}
```

This uses a macro `\getFrontArcStartPosition` to compute which parts of the orbit arcs are on the front side of the orbit so they can be drawn last and the back side can be drawn first with dotted lines. There is likely an easier way to do this, but this solution involves using the spherical law of sines on various triangles on the sphere. For completeness, an explanation of the formulas follows the code.

```
\newcommand\getFrontArcStartPosition[5]{
% Theta must be between -180 and 180.
\pgfmathtruncatemacro\psiInt{#3}
\pgfmathtruncatemacro\thetaInt{#4}
\pgfmathtruncatemacro\phiInt{#5}
\pgfmathtruncatemacro\psiTemp{\psiInt}
\ifthenelse{\thetaInt < 0}
% Negate theta and negate the results at the end.
    {\pgfmathtruncatemacro\thetaTemp{-\thetaInt}}
    {\pgfmathtruncatemacro\thetaTemp{\thetaInt}}
\pgfmathtruncatemacro\phiTemp{\phiInt}
\pgfmathsetmacro\anglexTemp{atan(sin(\thetaTemp)/
   (cos(\thetaTemp)*sin(\psiTemp)))}

\ifthenelse{\thetaTemp > 90}
    {\pgfmathsetmacro\anglexTemp{\anglexTemp + 180}}
    {}
\pgfmathsetmacro\result{atan(sin(\thetaTemp)*
   cos(\phiTemp)*sin(\anglexTemp)/
(sin(\anglexTemp)*cos(\psiTemp)*sin(\phiTemp)
+cos(\anglexTemp)*cos(\phiTemp)*sin(\thetaTemp)))}
\pgfmathsetmacro\specialAngle{(cos(\phiTemp)*
   cos(\phiTemp)-cos(\psiTemp)*cos(\psiTemp))/
   (cos(\phiTemp)*cos(\phiTemp)*
   sin(\psiTemp)*sin(\psiTemp))}
\ifthenelse{\phiInt < \psiInt}{
    \pgfmathparse{sqrt(\specialAngle)}
    \pgfmathsetmacro\specialAngle{
       asin(-\pgfmathresult)+180}
    \ifthenelse{\thetaTemp > \specialAngle}{
        \pgfmathsetmacro\result{\result + 180}
        \pgfmathsetmacro#1{\result}}
    {
        \pgfmathsetmacro#1{\result}}}
{}
% Negate the results if theta is negative.
\ifthenelse{\thetaInt < 0}{
    \pgfmathsetmacro#1{-\result}
    \pgfmathsetmacro#2{-\anglexTemp}}
{
    \pgfmathsetmacro#1{\result}
    \pgfmathsetmacro#2{\anglexTemp}}
}% End of \getFrontArcStartPosition function.
```

Appendix D Explanation of formulas in `\getFrontArcStartPosition`

The function `\getFrontArcStartPosition` in appendix C is used to find which parts of an orbit are on the front and which on the back, so that we can draw the back as a dotted line.

Figure 16 shows a red orbit that is tilted ψ degrees from the north pole and then rotated θ degrees with the viewing angle tilted ϕ degrees as the orbits are for figure 11. Our goal is to find the length of arc $DC = x'$ since that is where the orbit comes around to the front of the sphere.

The spherical law of sines says that for a triangle on a sphere

$$\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$$

where a, b, and c are the three angles of the triangle and A, B, and C are the three corresponding opposite side lengths (which are measured as an angle from the center of the sphere).

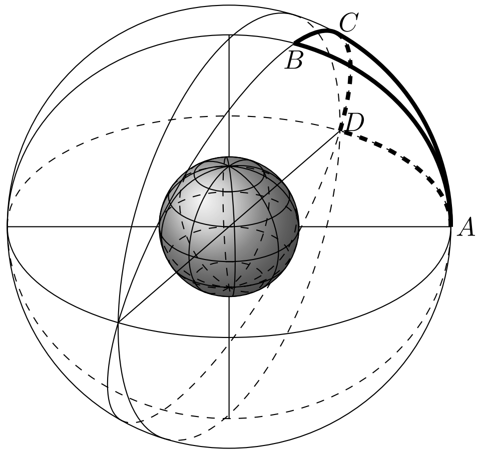


Figure 16: Given the orbit containing points B, C, and D, find the angular distance from D to C. If this distance is found, then the orbit is drawn with a solid arc from that point at C for 180 degrees and then for another 180 degrees as a dashed line.

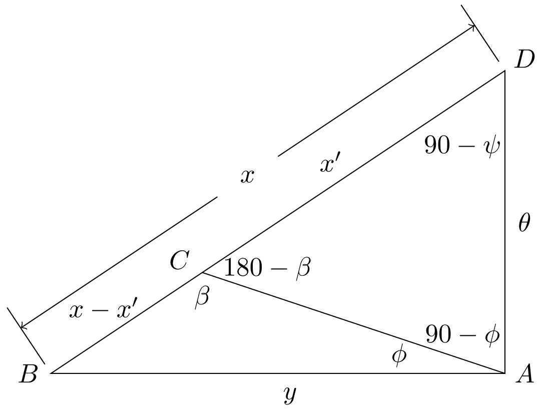


Figure 17: Triangles of figure 16 drawn in the plane.

Using the law of sines on the large triangle ABD, we have that

$$\frac{\sin y}{\sin(90-\psi)} = \frac{\sin x}{\sin 90}. \tag{1}$$

Using the law of sines on the lower small triangle ABC, we have that

$$\frac{\sin\beta}{\sin y} = \frac{\sin\phi}{\sin(x-x')}. \tag{2}$$

Simplifying and solving (1) for $\sin y$ and substituting into (2) and solving for $\sin\beta$, results in

$$\sin\beta = \frac{\cos\psi\sin x\sin\phi}{\sin(x-x')}. \tag{3}$$

Using the law of sines on the upper small triangle ACD, we have that

$$\frac{\sin(180-\beta)}{\sin\theta} = \frac{\sin(90-\phi)}{\sin x'}. \tag{4}$$

Solving (4) for $\sin\beta$ and setting equal to the right side of (3) results in

$$\frac{\sin\theta\cos\phi}{\sin x'} = \frac{\cos\psi\sin x\sin\phi}{\sin(x-x')}$$

or

$$\frac{\sin\theta\cos\phi}{\sin x'} = \frac{\cos\psi\sin x\sin\phi}{\sin x\cos x' - \cos x\sin x'}.$$

Cross multiplying, dividing by $\cos x'$ and solving for $\tan x'$ gives

$$\tan x' = \frac{\sin\theta\cos\phi\sin x}{\sin\theta\cos\phi\cos x + \cos\psi\sin\phi\sin x}.$$

Thus we can compute x' in terms of x. In order to compute x, see figure 18 where we have added points N and E.

Point E is chosen such that angle NEB is a right angle. Triangle NED is drawn in figure 19.

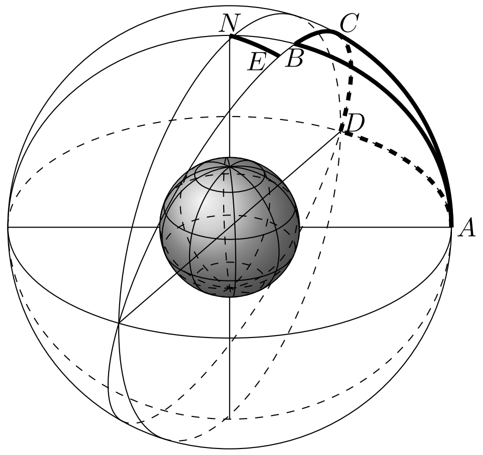


Figure 18: This is the same as figure 16 with added points N and E where segment NE is perpendicular to both red orbits. Triangle NED is used to find the value of x which is the arc BD.

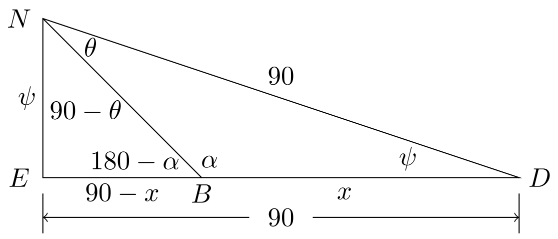


Figure 19: Triangle NED of figure 18 drawn in the plane.

The law of sines applied to triangle NBD on the right gives

$$\sin\alpha = \frac{\sin\theta}{\sin x}. \tag{5}$$

The law of sines applied to triangle NEB on the left gives $\sin(180-\alpha) =$

$$\sin\alpha = \frac{\sin\psi\sin(90-\theta)}{\sin(90-x)} = \frac{\sin\psi\cos\theta}{\cos x}. \tag{6}$$

Setting the values of $\sin\alpha$ equal from (5) and (6) and solving for $\tan x$ gives

$$\tan x = \frac{\tan\theta}{\sin\psi}.$$

Thus we now have x in terms of θ and ψ. This explains the formulas used for $0 \le \theta \le 90$ in the macro `\getFrontArcStartPosition`. In the case that $\psi \le \phi$, the same formulas work for when $90 \le \theta \le 180$. There are some complications when $\psi > \phi$ and $90 \le \theta \le 180$. In this case, when x' exceeds 90 degrees there is a sign change in the computation. Some more spherical trigonometry reveals that this happens when

$$\sin^2\theta = \frac{\cos^2\phi - \cos^2\psi}{\cos^2\phi\sin^2\psi}$$

which we use in `\getFrontArcStartPosition` to compute the value of θ where this change occurs. This allows us to compute x' for all $0 \le \theta \le 180$. For $-180 \le \theta \le 0$, we use symmetry and return the negative of the x' computed for $|\theta|$.

Appendix E Intersection of two spheres

Code for figure 15.

```
% The intersection of two spheres.
\begin{tikzpicture}[scale=.8]
% Draw the first sphere.
\def\Rb{3.1} % sphere radius
\def\angEl{30} % elevation angle
\def\xb{3} % x coordinate of center
\def\yb{3} % y coordinate of center
\def\zb{-1} % z coordinate of center
\filldraw[shift={(\xb, \yb)}][ball color= blue]
   (0, 0, 0) circle (\Rb);
\foreach \t in {-60,-20,...,80} {
   \DrawLatitudeCircle[\Rb]{\t}{\xb}{\yb}}
\foreach \t in {-5,-45,...,-175} {
   \DrawLongitudeCircle[\Rb]{\t}{\xb}{\yb}}
% Draw the second sphere.
\def\Rc{2.4} % sphere radius
\def\angEl{30} % elevation angle
\def\xc{0} % x coordinate of center
\def\yc{0} % y coordinate of center
\def\zc{0} % z coordinate of center
\filldraw[shift={(\xc, \yc)}][ball color= red]
   (0,0,0) circle (\Rc);
\foreach \t in {-60,-20,...,80} {
   \DrawLatitudeCircle[\Rc]{\t}{\xc}{\yc}}
\foreach \t in {-5,-45,...,-175} {
   \DrawLongitudeCircle[\Rc]{\t}{\xc}{\yc}}
\drawIntersectionOfSpheres{\xc}{\yc}{\zc}{\xb}
   {\yb}{\zb}{\Rc}{\Rb}{yellow}
\end{tikzpicture}
```

This code also requires the helper functions for drawing spheres, given in appendix C. Place them just before `\begin{document}`.

The following are additional helper functions to the main function that draws the intersection of the two spheres, `\drawIntersectionOfSpheres`.

```
\newcommand\calculateCenterSpan[4]{
  \pgfmathsetmacro#1{((#2)^2+(#3)^2+(#4)^2)^(1/2)}}
\newcommand\calculateAngtheta[3]{
  \pgfmathsetmacro#1{acos(#2/#3)}}
\newcommand\calculateShiftDistance[4]{
 \pgfmathsetmacro#1{((((#2)^2-(#3)^2)/(2*#4)+#4/2))}}
\newcommand\calculateShiftDistancePercent[3]{
  \pgfmathsetmacro#1{#2/#3}}
\newcommand\calculateCircleRadius[3]{
  \pgfmathsetmacro#1{((#2)^2 - (#3)^2)^(1/2)}
}
% The function below does not use the
% rotation matrices about coordinate axes,
% but instead computes the vector that we
% want to rotate around, then the
% corresponding rotation matrix, and then
% (as before) projects to the xy-plane.
% It also shifts to the correct location.
\newcommand\drawIntersectionOfSpheres[9]{
% Parameters are:  CenterSphere1x, CenterSphere1y,
% CenterSphere1z, CenterSphere2x,
% CenterSphere2y, CenterSphere2z,
% RadiusSphere1, RadiusSphere2, DrawColor.
\pgfmathsetmacro\xchange{#4 - #1}
\pgfmathsetmacro\ychange{#5 - #2}
\pgfmathsetmacro\zchange{#6 - #3}
\pgfmathsetmacro\firstSphereCenterx{#1}
\pgfmathsetmacro\firstSphereCentery{#2}
\calculateCenterSpan\centerSpan{
   \xchange}{\ychange}{\zchange}
\calculateAngtheta\angtheta{\zchange}{\centerSpan}
\calculateShiftDistance\shiftDistance{
   #7}{#8}{\centerSpan}
\calculateShiftDistancePercent\shiftDistancePercent{
   \shiftDistance}{\centerSpan}
\calculateCircleRadius\circleRadius{
   #7}{\shiftDistance}
\pgfmathsetmacro\ux{\ychange}
\pgfmathsetmacro\uy{-\xchange}
\pgfmathsetmacro\C{1-cos(\angtheta)}
\pgfmathsetmacro\L{((\ux)^2 + (\uy)^2)^(1/2)}
\pgfmathsetmacro\first{
  cos(\angtheta) + (\ux)^2*\C/(\L^2)}
\pgfmathsetmacro\second{\ux*\uy*\C/(\L^2)}
\pgfmathsetmacro\third{\ux*\uy*\C/(\L^2)}
\pgfmathsetmacro\fourth{
  cos(\angtheta)+(\uy)^2*\C/(\L^2)}
\tikzset{xyplane/.estyle={cm={\first,\second,\third,
  \fourth,(\firstSphereCenterx+\shiftDistancePercent
  *\xchange,\firstSphereCentery
           +\shiftDistancePercent*\ychange)}}}
\pgfmathsetmacro\dAng{atan(\xchange/\ychange)}
\draw[xyplane, color=#9, ultra thick]
  (-\dAng: \circleRadius) arc
  (-\dAng:-\dAng-180:\circleRadius);
\draw[xyplane, color=#9, ultra thick, dashed]
  (-\dAng+180: \circleRadius) arc
  (-\dAng+180:-\dAng:\circleRadius);
}
```

References

[1] Till Tantau. PGF/TikZ manual, version 2.10. http://mirror.ctan.org/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf, 2010.

[2] Andrew Mertz and William Slough. Graphics with PGF and TikZ. *TUGboat* 28:1, Proceedings of the Practical TeX 2006 Conference, http://tug.org/TUGboat/tb28-1/tb88mertz.pdf, 2007.

[3] Tomasz M. Trzeciak. http://www.texample.net/tikz/examples/map-projections/, 2008.

[4] Wikipedia, http://en.wikipedia.org/wiki/Rotation_matrix.

◇ Keith Wolcott
Dept. of Mathematics and Computer Science
Eastern Illinois University
Charleston, IL 61920-3099 USA
kwolcott (at) eiu dot edu