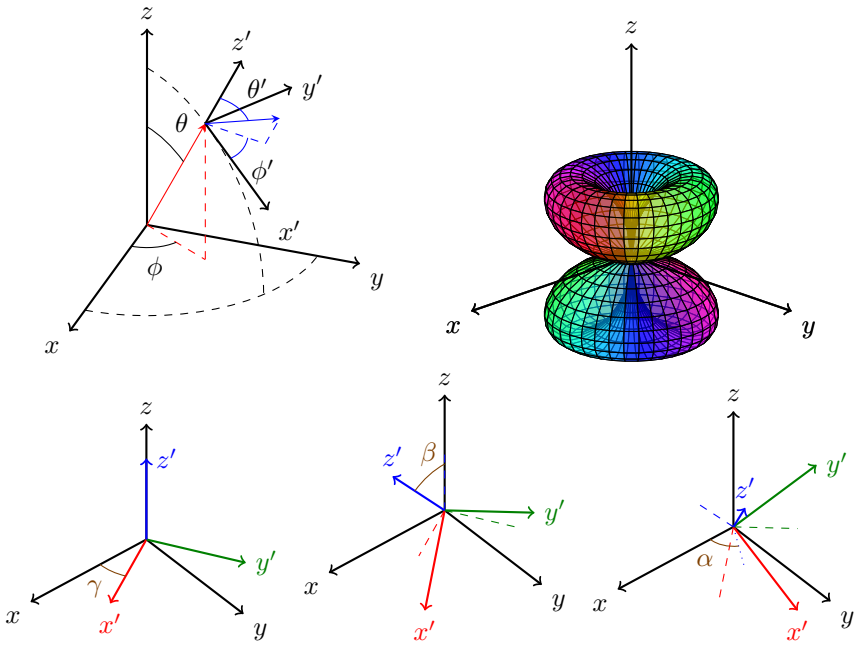
The `tikz-3dplot` Package

Jeff Hein
http://tikz3dplot.wordpress.com

January 23, 2021



Contents

Chapter 1

Introduction

1.1 Overview of the `tikz-3dplot` Package

The `tikz-3dplot` package offers commands and coordinate transformation styles for Ti*k*Z, providing relatively straightforward tools to draw three-dimensional coordinate systems and simple three-dimensional diagrams. The package is currently in its infancy, and is subject to change. Comments or suggestions are encouraged.

This document describes the basics of the `tikz-3dplot` package and provides information about the various available commands. Examples are given where possible.

1.1.1 What `tikz-3dplot` is

`tikz-3dplot` provides commands to easily specify coordinate transformations for Ti*k*Z, allowing for relatively easy plotting. I needed to draw accurate 3d vector images for a physics thesis, and this package was developed to meet this need.

Various plotting commands are used to identify coordinate locations using spherical polar or Cartesian coordinates. Coordinate transformation commands allow for the calculation of a coordinate in one frame based on its values in another frame. Some drawing commands have been developed to assist in the rendering of arcs. These commands do the number crunching required to position and render the arcs. These commands are discussed in Section 3.2.

In addition, the `\tdplotsphericalsurfaceplot` was developed to render three-dimensional surfaces in spherical polar coordinates, where the radius is expressed in terms of a user-defined function of θ and ϕ. With this function, the surface hue can be given explicitly, or expressed as a user-defined function of r, θ, and ϕ. This command is discussed in Section 3.6.

In `tikz-3dplot`, a right-handed coordinate system convention is used. In addition, all positive angles constitute a right-hand screw sense of rotation (see Figure 1.1). This means that a positive rotation about a given axis refers to a

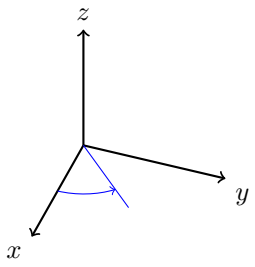


Figure 1.1: `tikz-3dplot` coordinate and positive angle convention.

clockwise rotation when viewing along the direction the axis, or counterclockwise when viewing against the direction of the axis.

1.1.2 What `tikz-3dplot` is not

`tikz-3dplot` does not, in general, consider polygons, surfaces, or object opacity. The one exception is the `\tdplotsphericalsurfaceplot` command, specifically designed to render spherical polar surfaces. The `\tdplotsphericalsurfaceplot` command is discussed in Section 3.6.

Tools like Sketch by Gene Ressler are better suited for more rigorous surface rendering. These can be found at http://www.frontiernet.net/~eugene.ressler/

1.1.3 Similar Work

To my knowledge, there is no other package available which allows straightforward rendering of 3d coordinates in Ti*k*Z, directly in a LaTeX document. Since this project is in its infancy, this may be subject to change based on feedback.

Sketch

The Sketch project can provide three-dimensional rendering of axes, points, and lines, but (as far as I understand the program) cannot draw arcs without using a series of line segments. Further, Sketch requires an external program to render the image, while `tikz-3dplot` can be developed and maintained right in a LaTeX document.

TeXample.net

There are a variety of Ti*k*Z examples listed at http://www.texample.net/tikz/examples. Some of these examples gave me inspiration to make this package. Some examples of note include the following:

- 3D cone

Author: Eugene Ressler

url: http://www.texample.net/tikz/examples/3d-cone/

Notes: This demonstrates the use of Sketch in TikZ figures.

- Annotated 3D box

  Author Alain Matthes

  url http://www.texample.net/tikz/examples/annotated-3d-box/

  Notes This example demonstrates the direct use of coordinate transformations, as well as performing math directly within coordinates.

- Cluster of atoms

  Author Agustin E. Bolzan

  url http://www.texample.net/tikz/examples/clusters-of-atoms/

  Notes This uses shifts and slants rather than rotations to render an isometric look.

- Plane partition

  Author Jang Soo Kim

  url http://www.texample.net/tikz/examples/plane-partition/

  Notes This example draws solid surfaces with coordinate axes defined by rotations around the Ti*k*Z standard coordinate frame.

- Spherical and cartesian grids

  Author Marco Miani

  url http://www.texample.net/tikz/examples/spherical-and-cartesian-grids/

  Notes This example renders arcs and lines in three dimensions using explicit calculations. It takes into account the opacity of the spherical example, by showing hidden lines behind the sphere as dashed lines.

- Stereographic and cylindrical map projections

  Author Thomas M. Trzeciak

  url http://www.texample.net/tikz/examples/map-projections/

  Notes This example illustrates the use of coordinate transformations to draw planes and arcs for spherical coordinates.

1.2 Installing the `tikz-3dplot` Package

Get a copy of `tikz-3dplot` from http://www.ctan.org. Place the style file in the same directory as your LaTeX project. In your preamble, add the following line:

```
\usepackage{tikz-3dplot}
```

Make sure this line is written after all other required packages.

1.2.1 `tikz-3dplot` Requirements

To use this package, the following other packages must be loaded in the preamble first:

- TikZ
- `ifthen` (for the `tdplotsphericalsurfaceplot` command)

1.2.2 `tikz-3dplot` Package Options

Currently there are no options available for the `tikz-3dplot` package.

1.3 Using the `tikz-3dplot` Package

`tikz-3dplot` provides styles and commands which are useful in a `tikzpicture` environment. These commands and styles are described in Chapter 3.

Chapter 2

Overview of 3d in `tikz-3dplot`

2.1 Ti*k*Z 3d Plotting

When setting up a tikzpicture or a drawing style, the x, y, and z axes can be specified directly in terms of the original coordinate system. The following example shows how a `tikzpicture` environment can be configured to use customized axes.

```
\begin{tikzpicture}[%
    x={(\raarot cm,\rbarot cm)},%
    y={(\rabrot cm, \rbbrot cm)},%
    z={(\racrot, \rbcrot cm)}]
```

In this example, the terms `\raarot` and so on specify how the coordinates are represented in the original Ti*k*Z coordinate system, and are calculated by the `tikz-3dplot` package. Note that units are explicitly required so Ti*k*Z understands that these are absolute coordinates, not scales on the existing axis. See the PGF manual Version 2.00, section 21.2 on pages 217-218 for details on Ti*k*Z coordinate transformations.

2.2 The `tikz-3dplot` Main Coordinate System

`tikz-3dplot` offers two coordinate systems, namely the *main* coordinate system (x, y, z), and the *rotated* coordinate system (x', y', z'). The latter system is described in Section 2.3.

As the name suggests, the main coordinate system provides a user-specified transformation to render 3d points in a `tikzpicture` environment. The orientation of the main coordinate system is defined by the angles θ_d and ϕ_d. In the unrotated ($\theta_d = \phi_d = 0$) position, the xy plane of the main coordinate system coincides with the default orientation for a `tikzpicture` environment, while z

points "out of the page". The coordinate system is positioned by the following operations:

- Rotate the coordinate system about the body x axis by the amount θ_d, and
- Rotate the coordinate system about the (rotated) body z axis by the amount ϕ_d.

In this rotation sense, the z axis will always point in the vertical page direction. This transformation is given by the rotation matrix $R_d(\theta_d, \phi_d)$, as

$$\begin{aligned} R^d(\theta_d, \phi_d) &= R^{z'}(\phi_d) R^x(\theta_d) \\ &= \begin{pmatrix} \cos\phi_d & -\sin\phi_d & 0 \\ \sin\phi_d & \cos\phi_d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_d & -\sin\theta_d \\ 0 & \sin\theta_d & \cos\theta_d \end{pmatrix} \\ &= \begin{pmatrix} \cos\phi_d & \sin\phi_d & 0 \\ -\cos\theta_d \sin\phi_d & \cos\theta_d \cos\phi_d & -\sin\theta_d \\ \sin\theta_d \sin\phi_d & -\sin\theta_d \cos\phi_d & \cos\theta_d \end{pmatrix} \end{aligned} \tag{2.1}$$

Using this matrix, the Ti*k*Z coordinate transformation can be applied as described in Section 2.1 by the various matrix elements, as

$$\begin{aligned} x &= (R^d_{1,1}, R^d_{2,1}) \\ y &= (R^d_{1,2}, R^d_{2,2}) \\ z &= (R^d_{1,3}, R^d_{2,3}) \end{aligned} \tag{2.2}$$

Note that the third row of the rotation matrix is not needed for this transformation, since a screen coordinate is a 2d value. Once the transformed axes have been established, any 3d coordinate specified in Ti*k*Z will adhere to the

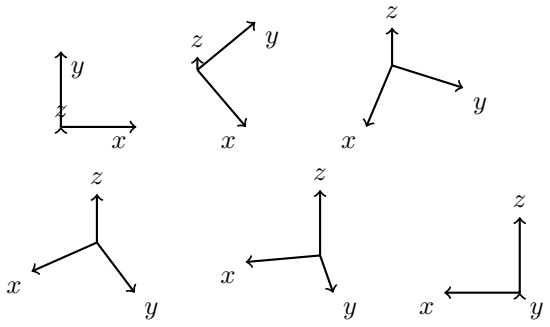


Figure 2.1: Examples of coordinate systems for various choices of θ_d and ϕ_d.

transformation, yielding a 3D representation. Lines and nodes can readily be drawn by using these 3d coordinates.

This coordinate transformation is accessible through `tikz-3dplot` using the command `tdplotsetmaincoords`, as described in Chapter 3.

2.3 The `tikz-3dplot` Rotated Coordinate System

Along with the main coordinate system, described in Section 2.2, `tikz-3dplot` offers a *rotated* coordinate system that is defined with respect to the main coordinate system. This system can be rotated to any position using Euler rotations, and can be translated so the origin of the rotated coordinate system sits on an arbitrary point in the main coordinate system.

Three rotations can be performed to give any arbitrary orientation of a rotated coordinate system. By convention, the following rotations are chosen:

- Rotate by angle γ about the world z axis,
- Rotate by angle β about the (unrotated) world y axis, and
- Rotate by angle α about the (unrotated) world z axis.

These rotations are shown in Figure 2.2.

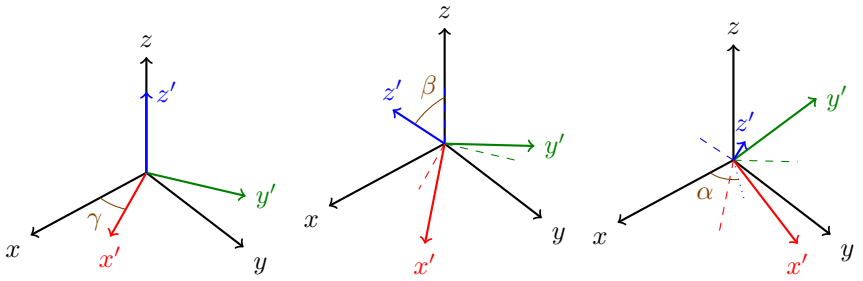


Figure 2.2: Positioning the rotated coordinate frame (x', y', z') using Euler angles (α, β, γ).

This rotation matrix $D(\alpha, \beta, \gamma)$ is given by

$$
\begin{aligned}
D(\alpha,\beta,\gamma) &= R^z(\alpha)R^y(\beta)R^z(\gamma) \\
&= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma & -\cos\alpha\cos\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta \\ \sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta \\ -\sin\beta\cos\gamma & \sin\beta\sin\gamma & \cos\beta \end{pmatrix}
\end{aligned}
\tag{2.3}
$$

To define the rotated coordinate frame, this rotation matrix is applied after rotation matrix $R^d(\theta_d, \phi_d)$ used to define the main coordinate frame. The full transformation for the rotated coordinate frame is then given by

$$R'^d(\theta_d, \phi_d, \alpha, \beta, \gamma) = D(\alpha, \beta, \gamma) R^d(\theta_d, \phi_d) \tag{2.4}$$

Using this matrix, the Ti*k*Z coordinate transformation can be applied as described in Section 2.1 by the various matrix elements, as

$$\begin{aligned} x' &= (R'^d_{1,1}, R'^d_{2,1}) \\ y' &= (R'^d_{1,2}, R'^d_{2,2}) \\ z' &= (R'^d_{1,3}, R'^d_{2,3}) \end{aligned} \tag{2.5}$$

This coordinate transformation is accessible through `tikz-3dplot` using the command `tdplotsetrotatedcoords`, as described in Chapter 3.

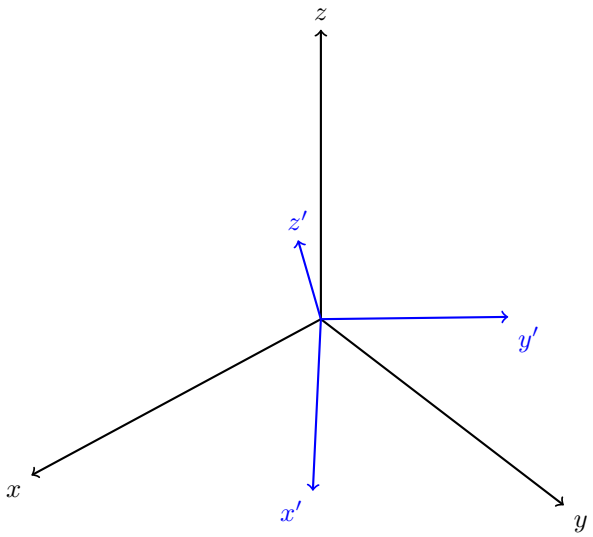


Figure 2.3: The rotated coordinate frame (x', y', z') displayed within the main coordinate frame (x, y, z). Both are completely specified by user-defined angles: (θ_d, ϕ_d) for the main coordinate frame, and (α, β, γ) for the rotated coordinate frame.

2.4 Arcs in 3d, and the "Theta Plane"

Arcs can be drawn in Ti*k*Z using commands described in the PGF manual Version 2.00, section 2.10 on pages 25-26. However, the arc commands accept 2d coordinates, and thus can only be drawn in the xy plane.

To draw an arc in any position other than within the xy plane of the main coordinate frame, the rotated coordinate frame must be used, where the $x'y'$ plane lies in the desired orientation within the main coordinate frame. Such an arc is needed, for example, when illustrating the polar angle θ of some vector. This θ arc exists in a plane which contains the z axis, and is rotated about the z axis by the angle ϕ from the xz plane. For lack of a better name, this plane is referred to as the "theta plane" within a given coordinate system.

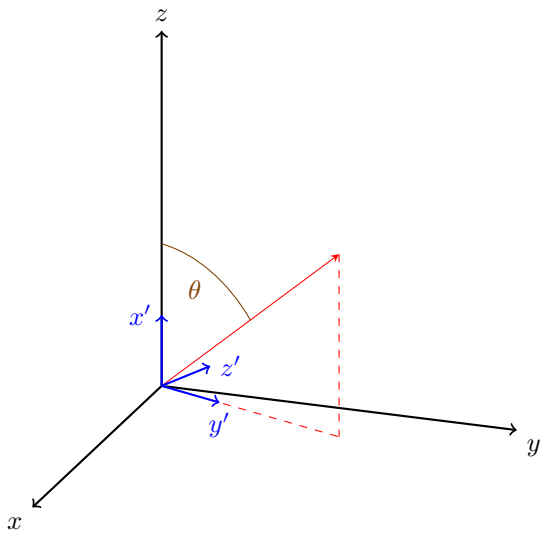


Figure 2.4: Drawing arcs outside the xy plane by using a rotated coordinate frame in the "theta plane" of the main coordinate frame.

As described in Chapter 3, `tikz-3dplot` offers the commands `tdplotsetthetaplanecoords` and `tdplotsetrotatedthetaplanecoords` to easily configure the rotated coordinate frame to lie within the desired theta plane.

Chapter 3

Using the tikz-3dplot Package

The tikz-3dplot package was developed to handle the number crunching described in Chapter 2, and provide a relatively simple and straightforward front-end for users.

The main and rotated coordinate frames are configured by using commands described in Section 3.2. These commands generate Ti*k*Z styles which can be used either in defining the tikzpicture environment, or directly in any Ti*k*Z command. The styles are described further in Section 3.1.

3.1 The tikz-3dplot Ti*k*Z Styles

3.1.1 tdplot_main_coords

The tdplot_main_coords style stores the coordinate transformation required to generate the main coordinate system. This style can either be used when the tikzpicture environment is started, or when an individual Ti*k*Z plotting command is used.

3.1.2 tdplot_rotated_coords

The tdplot_rotated_coords style stores the coordinate transformation (translation and rotation) required to generate the rotated coordinate system within the main coordinate system. This style can either be used when the tikzpicture environment is started, or when an individual Ti*k*Z plotting command is used.

3.1.3 tdplot_screen_coords

The tdplot_screen_coords style provides the standard, unrotated Ti*k*Z coordinate frame. This is useful to escape out of the user-defined 3d coordinates used at the beginning of the tikzpicture environment, and place something on an absolute scale in the figure. Tables, legends, and captions contained within the same figure as a 3d plot can make use of this style.

3.2 The tikz-3dplot Commands

This section lists the various commands provided by the tikz-3dplot package. Examples are provided where it is useful.

3.3 Coordinate Configuration Commands

3.3.1 tdplotsetmaincoords

Description: Generates the style tdplot_main_coords which provides the coordinate transformation for the main coordinate frame, based on a user-specified orientation (θ_d, ϕ_d). θ_d denotes the rotation around the x axis, while ϕ_d denotes the rotation around the z axis. Note that $(0, 0)$ is the default orientation, where x points right, y points up, and z points "out of the page".

Syntax: \tdplotsetmaincoords{ θ_d }{ ϕ_d }

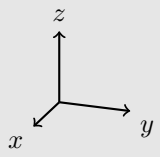
Parameters:

θ_d The angle (in degrees) through which the coordinate frame is rotated about the x axis.

ϕ_d The angle (in degrees) through which the coordinate frame is rotated about the z axis.

Example:

```
\tdplotsetmaincoords{70}{110}
\begin{tikzpicture}[tdplot_main_coords]
    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};
\end{tikzpicture}
```



3.3.2 tdplotsetrotatedcoords

Description: Generates the style tdplot_rotated_coords which provides the coordinate transformation for rotated coordinate frame within the current main coordinate frame, based on user-specified Euler angles (α, β, γ). Rotations use the $z(\alpha)y(\beta)z(\gamma)$ convention of Euler rotations, where the

system is rotated by γ about the z axis, then β about the (world) y axis, and then α about the (world) z axis.

Syntax: `\tdplotsetrotatedcoords{`α`}{`β`}{`γ`}`

Parameters:

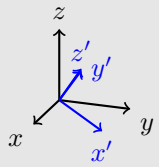
α The angle (in degrees) through which the rotated frame is rotated about the world z axis.

β The angle (in degrees) through which the rotated frame is rotated about the world y axis.

γ The angle (in degrees) through which the rotated frame is rotated about the world z axis.

Example:

```
\tdplotsetmaincoords{70}{110}
\begin{tikzpicture}[tdplot_main_coords]
    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \tdplotsetrotatedcoords{60}{40}{30}

    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (.7,0,0) node[anchor=north]{$x'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (0,.7,0) node[anchor=west]{$y'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (0,0,.7) node[anchor=south]{$z'$};
\end{tikzpicture}
```



3.3.3 `tdplotsetrotatedcoordsorigin`

Description: Sets the origin of the rotated coordinate system specified by `tdplot_rotated_coords` using a user-defined point. This point can be either a literal or predefined point.

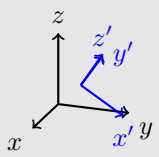
Syntax: `\tdplotsetrotatedcoordsorigin{point}`

Parameters:

`point` A point predefined using the Ti*k*Z `\coordinate` command.

Example:

```
\tdplotsetmaincoords{70}{110}
\begin{tikzpicture}[tdplot_main_coords]
    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \tdplotsetrotatedcoords{60}{40}{30}

    \coordinate (Shift) at (0.5,0.5,0.5);
    \tdplotsetrotatedcoordsorigin{(Shift)}

    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (.7,0,0) node[anchor=north]{$x'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (0,.7,0) node[anchor=west]{$y'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0) --
        (0,0,.7) node[anchor=south]{$z'$};
\end{tikzpicture}
```



3.3.4 `tdplotresetrotatedcoordsorigin`

Description: Resets the origin of the rotated coordinate system back to the origin of the main coordinate system.

Syntax: `\tdplotresetrotatedcoordsorigin`

Parameters: None

3.3.5 `tdplotsetthetaplanecoords`

Description: Generates a rotated coordinate system such that the $x'y'$ plane is coplanar to a plane containing the polar angle θ projecting from the main coordinate system z axis. This coordinate system is particularly useful for drawing within this "theta plane", as Ti*k*Z draws arcs in the xy plane. As with `tdplotsetrotatedcoords`, this coordinate system is accessible through the `tdplot_rotated_coords` style. Note that any rotated coordinate frame offset previously set by `tdplotsetrotatedcoordsorigin` is automatically reset when this command is used.

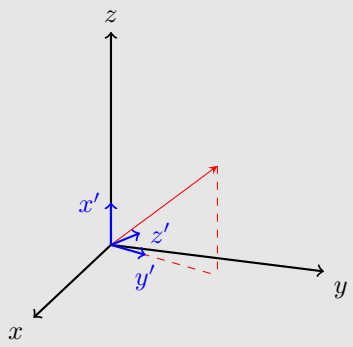
Syntax: `\tdplotsetthetaplanecoords{`ϕ`}`

Parameters:

ϕ The angle (in degrees) through which the "theta plane" makes with the xz plane of the main coordinate system.

Example:

```
\tdplotsetmaincoords{70}{110}

\begin{tikzpicture}[scale=3,tdplot_main_coords]
    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \tdplotsetcoord{P}{.8}{50}{70}

    %draw a vector from origin to point (P)
    \draw[-stealth,color=red] (O) -- (P);

    %draw projection on xy plane, and a connecting line
    \draw[dashed, color=red] (O) -- (Pxy);
    \draw[dashed, color=red] (P) -- (Pxy);


    \tdplotsetthetaplanecoords{70}

    \draw[tdplot_rotated_coords,color=blue,thick,->] (0,0,0)
        -- (.2,0,0) node[anchor=east]{$x'$};
    \draw[tdplot_rotated_coords,color=blue,thick,->] (0,0,0)
        -- (0,.2,0) node[anchor=north]{$y'$};
    \draw[tdplot_rotated_coords,color=blue,thick,->] (0,0,0)
        -- (0,0,.2) node[anchor=west]{$z'$};

\end{tikzpicture}
```



3.3.6 `tdplotsetrotatedthetaplanecoords`

Description: Just like `tdplotsetthetaplanecoords`, except this works for the rotated coordinate system. Generates a rotated coordinate system such that the $x' - y'$ plane is coplanar to a plane containing the polar

angle θ' projecting from the current rotated coordinate system z' axis. Note that the current rotated coordinate system is overwritten by this theta plane coordinate system after the command is completed.

Syntax: `\tdplotsetrotatedthetaplanecoords{`ϕ'`}`

Parameters:

ϕ' The angle (in degrees) through which the "theta plane" makes with the $x' - z'$ plane of the current rotated coordinate system.

Example:

```
\tdplotsetmaincoords{60}{110}
\begin{tikzpicture}[scale=3,tdplot_main_coords]
    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \coordinate (Shift) at (2,2,2);

    \tdplotsetrotatedcoords{-20}{10}{0}
    \tdplotsetrotatedcoordsorigin{(Shift)}

    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (1,0,0) node[anchor=south east]{$x'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,1,0) node[anchor=west]{$y'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,1) node[anchor=south]{$z'$};

    \tdplotsetrotatedthetaplanecoords{30}

    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (.5,0,0) node[anchor=south east]{$x''$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.5,0) node[anchor=west]{$y''$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.5) node[anchor=south]{$z''$};
\end{tikzpicture}
```

3.3.7 `tdplotcalctransformmainrot`

Description: Calculates the rotation matrix used to transform a coordinate from the main coordinate frame to the rotated coordinate frame. The matrix elements are stored in the macros `\raaeul` through `\rcceul`. This transformation is accessed using `\tdplottransformmainrot`.

3.3.8 `tdplotcalctransformrotmain`

Description: Calculates the rotation matrix used to define the rotated coordinate frame, as well as transform a coordinate from the rotated coordinate frame to the main coordinate frame. The matrix elements are stored in the macros `\raaeul` through `\rcceul`. This transformation is used in the `\tdplotsetrotatedcoords` command, and is accessed using `\tdplottransformrotmain`.

3.3.9 `tdplotcalctransformmainscreen`

Description: Calculates the rotation matrix used to define the main coordinate frame, as well as transform a coordinate from the main coordinate frame to the screen coordinate frame. The matrix elements are stored in the macros `\raarot` through `\rccrot`. This transformation is used in the `\tdplotsetmaincoords` command, and is accessed using `\tdplottransformmainscreen`.

3.4 Point Calculation Commands

3.4.1 `tdplotsetcoord`

Description: Generates a TikZ coordinate of specified name, along with coordinates for the $x-$, $y-$, $z-$, $xy-$, $xz-$, and $yz-$ projections of the coordinate, based on user-specified spherical coordinates. Note that this coordinate only works in the main coordinate system. All points in the rotated coordinate system must be specified as literal points.

Syntax: `\tdplotsetcoord{point}{`r`}{`θ`}{`ϕ`}`

Parameters:

`point` The name of the TikZ coordinate to be assigned. Note that the () parentheses must be excluded.

r Point radius.

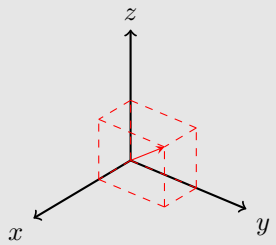
θ Point polar angle.

ϕ Point azimuthal angle.

Example:

```
\tdplotsetmaincoords{60}{130}
\begin{tikzpicture}[scale=2,tdplot_main_coords]

    \coordinate (O) at (0,0,0);

    \tdplotsetcoord{P}{.8}{55}{60}

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
```

```
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \draw[-stealth,color=red] (O) -- (P);

    \draw[dashed, color=red] (O) -- (Px);
    \draw[dashed, color=red] (O) -- (Py);
    \draw[dashed, color=red] (O) -- (Pz);
    \draw[dashed, color=red] (Px) -- (Pxy);
    \draw[dashed, color=red] (Py) -- (Pxy);
    \draw[dashed, color=red] (Px) -- (Pxz);
    \draw[dashed, color=red] (Pz) -- (Pxz);
    \draw[dashed, color=red] (Py) -- (Pyz);
    \draw[dashed, color=red] (Pz) -- (Pyz);
    \draw[dashed, color=red] (Pxy) -- (P);
    \draw[dashed, color=red] (Pxz) -- (P);
    \draw[dashed, color=red] (Pyz) -- (P);
\end{tikzpicture}
```



3.4.2 `tdplottransformmainrot`

Description: Transforms a coordinate from the main coordinate frame to the rotated coordinate frame. This command cannot use a Ti*k*Z coordinate, and does not account for a shifted rotated coordinate frame. The results are stored in the `\tdplotresx`, `\tdplotresy`, and `\tdplotresz` macros.

Syntax: `\tdplottransformmainrot{x}{y}{z}`

Parameters:

x The x-component of the coordinate in the main coordinate frame.

y The y-component of the coordinate in the main coordinate frame.

z The z-component of the coordinate in the main coordinate frame.

Output: The following macros are assigned:

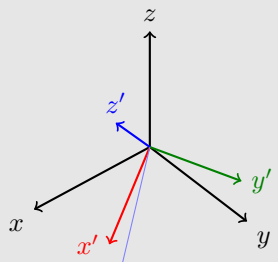
`tdplotresx` The transformed coordinate x component in the rotated coordinate frame.

`tdplotresy` The transformed coordinate y component in the rotated coordinate frame.

tdplotresz The transformed coordinate z component in the rotated coordinate frame.

Example:

```
\tdplotsetmaincoords{50}{140}
\begin{tikzpicture}[scale=2,tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \pgfmathsetmacro{\ax}{2}
    \pgfmathsetmacro{\ay}{2}
    \pgfmathsetmacro{\az}{1}

    \tdplotsetrotatedcoords{20}{40}{00}

    \draw[thick,color=red,tdplot_rotated_coords,->] (0,0,0)
        -- (.7,0,0) node[anchor=east]{$x'$};
    \draw[thick,color=green!50!black,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.7,0) node[anchor=west]{$y'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.7) node[anchor=south]{$z'$};


    \tdplottransformmainrot{\ax}{\ay}{\az}

    \draw[tdplot_rotated_coords,->,blue!50] (0,0,0)
        -- (\tdplotresx,\tdplotresy,\tdplotresz);

    \node[tdplot_main_coords,anchor=south]
        at (\ax,\ay,\az){Main coords: (\ax, \ay, \az)};
    \node[tdplot_rotated_coords,anchor=north]
        at (\tdplotresx,\tdplotresy,\tdplotresz)
        {Rotated coords: (\tdplotresx, \tdplotresy, \tdplotresz)};

\end{tikzpicture}
```



Main coords: (2, 2, 1)

Rotated coords: (1.32086, 1.19534, 2.41377)

3.4.3 `tdplottransformrotmain`

Description: Transforms a coordinate from the rotated coordinate frame to the main coordinate frame. This command cannot use a Ti*k*Z coordinate, and does not account for a shifted rotated coordinate frame. The results are stored in the `\tdplotresx`, `\tdplotresy`, and `\tdplotresz` macros.

Syntax: `\tdplottransformrotmain{x}{y}{z}`

Parameters:

x The x-component of the coordinate in the rotated coordinate frame.

y The y-component of the coordinate in the rotated coordinate frame.

z The z-component of the coordinate in the rotated coordinate frame.

Output: The following macros are assigned:

`tdplotresx` The transformed coordinate x component in the main coordinate frame.

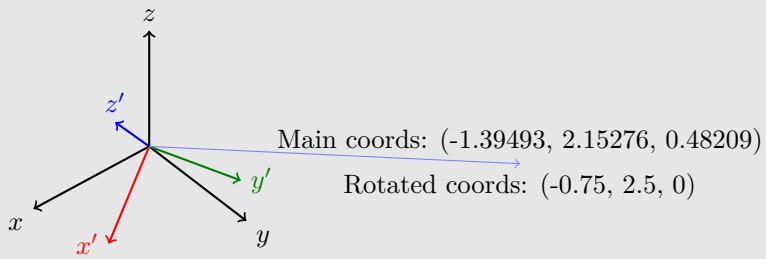
`tdplotresy` The transformed coordinate y component in the main coordinate frame.

`tdplotresz` The transformed coordinate z component in the main coordinate frame.

Example:

```
\tdplotsetmaincoords{50}{140}
\begin{tikzpicture}[scale=2,tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \pgfmathsetmacro{\ax}{-.75}
    \pgfmathsetmacro{\ay}{2.5}
    \pgfmathsetmacro{\az}{0}

    \tdplotsetrotatedcoords{20}{40}{00}

    \draw[thick,color=red,tdplot_rotated_coords,->] (0,0,0)
        -- (.7,0,0) node[anchor=east]{$x'$};
    \draw[thick,color=green!50!black,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.7,0) node[anchor=west]{$y'$};
    \draw[thick,color=blue,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.7) node[anchor=south]{$z'$};


    \tdplottransformrotmain{\ax}{\ay}{\az}

    \draw[tdplot_main_coords,->,blue!50] (0,0,0)
        -- (\tdplotresx,\tdplotresy,\tdplotresz);
```

```
    \node[tdplot_rotated_coords,anchor=north]
        at (\ax,\ay,\az){Rotated coords: (\ax, \ay, \az)};
    \node[tdplot_main_coords,anchor=south]
        at (\tdplotresx,\tdplotresy,\tdplotresz)
        {Main coords: (\tdplotresx, \tdplotresy, \tdplotresz)};


\end{tikzpicture}
```



3.4.4 `tdplottransformmainscreen`

Description: Transforms a coordinate from the main coordinate frame to the screen coordinate frame. This command cannot use a Ti*k*Z coordinate. The results are stored in the `\tdplotresx` and `\tdplotresy` macros.

Syntax: `\tdplottransformmainscreen{x}{y}{z}`

Parameters:

x The x-component of the coordinate in the main coordinate frame.

y The y-component of the coordinate in the main coordinate frame.

z The z-component of the coordinate in the main coordinate frame.

Output: The following macros are assigned:

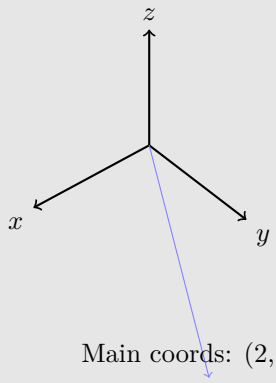
`tdplotresx` The transformed coordinate x component in the screen coordinate frame.

`tdplotresy` The transformed coordinate y component in the screen coordinate frame.

Example:

```
\tdplotsetmaincoords{50}{140}
\begin{tikzpicture}[scale=2,tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
```

```
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \pgfmathsetmacro{\ax}{2}
    \pgfmathsetmacro{\ay}{3}
    \pgfmathsetmacro{\az}{1}

    \tdplottransformmainscreen{\ax}{\ay}{\az}

    \draw[tdplot_screen_coords,->,blue!50] (0,0)
        -- (\tdplotresx,\tdplotresy);

    \node[tdplot_main_coords,anchor=south]
        at (\ax,\ay,\az){Main coords: (\ax, \ay, \az)};
    \node[tdplot_screen_coords,anchor=north]
        at (\tdplotresx,\tdplotresy)
        {Screen coords: (\tdplotresx, \tdplotresy)};

\end{tikzpicture}
```



Main coords: (2, 3, 1)

Screen coords: (0.3963, -1.53752)

3.4.5 `tdplotgetpolarcoords`

Description: Calculates the θ polar coordinate for the specified point. The result is specified in the `\tdplotrestheta` macro.

Syntax: `\tdplotgetpolarcoords{x}{y}{z}`

Parameters:

x The x-component of the coordinate.

y The y-component of the coordinate.

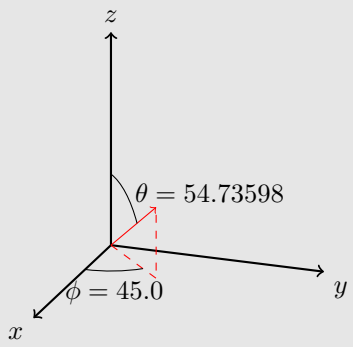
z The z-component of the coordinate.

Output: **`tdplotrestheta`** The θ polar coordinate.

`tdplotresphi` The ϕ polar coordinate.

Example:

```
\tdplotsetmaincoords{70}{110}
\begin{tikzpicture}[tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (3,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,3,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,3) node[anchor=south]{$z$};

    \pgfmathsetmacro{\ax}{1}
    \pgfmathsetmacro{\ay}{1}
    \pgfmathsetmacro{\az}{1}

    \draw[->,red] (0,0,0) -- (\ax,\ay,\az);

    \draw[dashed,red] (0,0,0) -- (\ax,\ay,0) -- (\ax,\ay,\az);

    \tdplotgetpolarcoords{\ax}{\ay}{\az}

    \tdplotsetthetaplanecoords{\tdplotresphi}

    \tdplotdrawarc[tdplot_rotated_coords]{(0,0,0)}{1}{0}%
        {\tdplotrestheta}{anchor=west}{$\theta = \tdplotrestheta$}
\end{tikzpicture}
```



3.4.6 `tdplotcrossprod`

Description: Calculates the cross product of two vectors specified by two coordinates with respect to the origin. The result vector is specified by the coordinates `\tdplotresx`, `\tdplotresy`, and `\tdplotresz` with respect to the origin.

Syntax: `\tdplotcrossprod(`a_x`,`a_y`,`a_z`)(`b_x`,`b_y`,`b_z`)`

Parameters:

a_x The x-component of the first vector with respect to the origin.

a_y The y-component of the first vector with respect to the origin.

a_z The z-component of the first vector with respect to the origin.

b_x The x-component of the second vector with respect to the origin.

b_y The y-component of the second vector with respect to the origin.

b_z The z-component of the second vector with respect to the origin.

Output: The following macros are assigned.

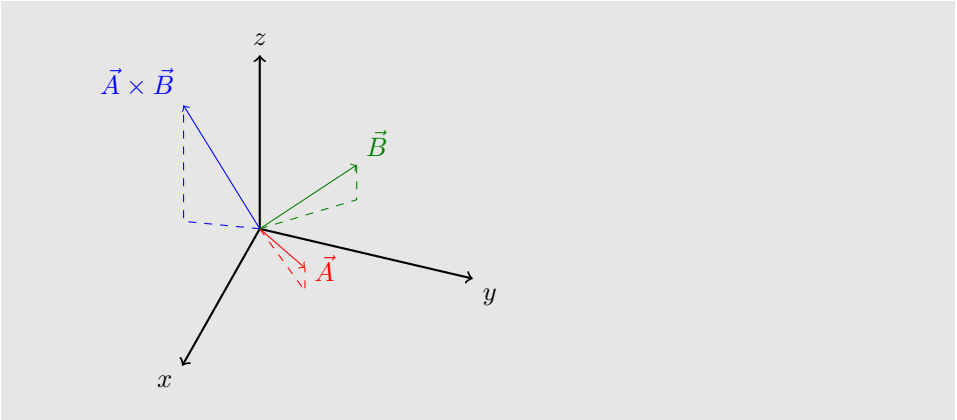
tdplotresx The x-component of the cross product with respect to the origin.

tdplotresy The y-component of the cross product with respect to the origin.

tdplotresz The z-component of the cross product with respect to the origin.

Example:

```
\tdplotsetmaincoords{50}{110}
\begin{tikzpicture}[tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (3,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,3,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,3) node[anchor=south]{$z$};

    \pgfmathsetmacro{\ax}{1}
    \pgfmathsetmacro{\ay}{1}
    \pgfmathsetmacro{\az}{.4}
    \pgfmathsetmacro{\bx}{-1}
    \pgfmathsetmacro{\by}{1}
    \pgfmathsetmacro{\bz}{.6}

    \tdplotcrossprod(\ax,\ay,\az)(\bx,\by,\bz)

    \draw[->,red] (0,0,0) -- (\ax,\ay,\az) node[anchor=west]{$\vec{A}$};
    \draw[dashed,red] (0,0,0) -- (\ax,\ay,0) -- (\ax,\ay,\az);
    \draw[->,green!50!black] (0,0,0) --
        (\bx,\by,\bz) node[anchor=south west]{$\vec{B}$};
    \draw[dashed,green!50!black] (0,0,0) -- (\bx,\by,0) -- (\bx,\by,\bz);

    \draw[->,blue] (0,0,0) -- (\tdplotresx,\tdplotresy,\tdplotresz)
        node[anchor=south east]{$\vec{A}\times\vec{B}$};
    \draw[dashed,blue] (0,0,0) -- (\tdplotresx,\tdplotresy,0)
        -- (\tdplotresx,\tdplotresy,\tdplotresz);
\end{tikzpicture}
```



3.4.7 `tdplotdefinepoints`

Description: Assigns the values of three coordinates, to be used in the `\tdplotdrawpolytopearc`

Syntax: `\tdplotdefinepoints`$(v_x,v_y,v_z)(a_x,a_y,a_z)(b_x,b_y,b_z)$

Parameters:

- v_x The x-component of the vertex.
- v_y The y-component of the vertex.
- v_z The z-component of the vertex.
- a_x The x-component of the first point.
- a_y The y-component of the first point.
- a_z The z-component of the first point.
- b_x The x-component of the second point.
- b_y The y-component of the second point.
- b_z The z-component of the second point.

Output: The following macros are assigned:

- **`tdplotvertexx`** The x-component of the vertex.
- **`tdplotvertexy`** The y-component of the vertex.
- **`tdplotvertexz`** The z-component of the vertex.
- **`tdplotax`** The x-component of the first point.
- **`tdplotay`** The y-component of the first point.
- **`tdplotaz`** The z-component of the first point.
- **`tdplotbx`** The x-component of the second point.
- **`tdplotby`** The y-component of the second point.
- **`tdplotbz`** The z-component of the second point.

3.5 Drawing Commands

Along with all the conventional Ti*k*Z drawing commands, the following `tikz-3dplot` commands can be used.

3.5.1 `tdplotdrawarc`

Description: Draws an arc in the xy (or optionally $x'y'$) plane starting from the specified polar angle ϕ, of specified radius and angular length, at specified center point, and labels the arc with specified node text and options. By default, draws in the main coordinate frame, but can draw in the rotated coordinate frame by specifying `tdplot_rotated_coords` in the option field.

Syntax: `\tdplotdrawarc[coordinate system, draw styles]{center}{r}{angle start}{angle end}{label options}{label}`

Parameters:

(*Optional*) coordinate system,draw styles Optional argument containing the name of the coordinate system to use (default is main coordinate system), and any optional draw styles.

center Center point through which to draw the arc. If using the rotated coordinate system, this must be a literal value.

r The arc radius of curvature.

angle start the initial angle (in degrees) through which to draw. 0 points along the x (or x') axis.

angle end the final angle (in degrees) through which to draw.

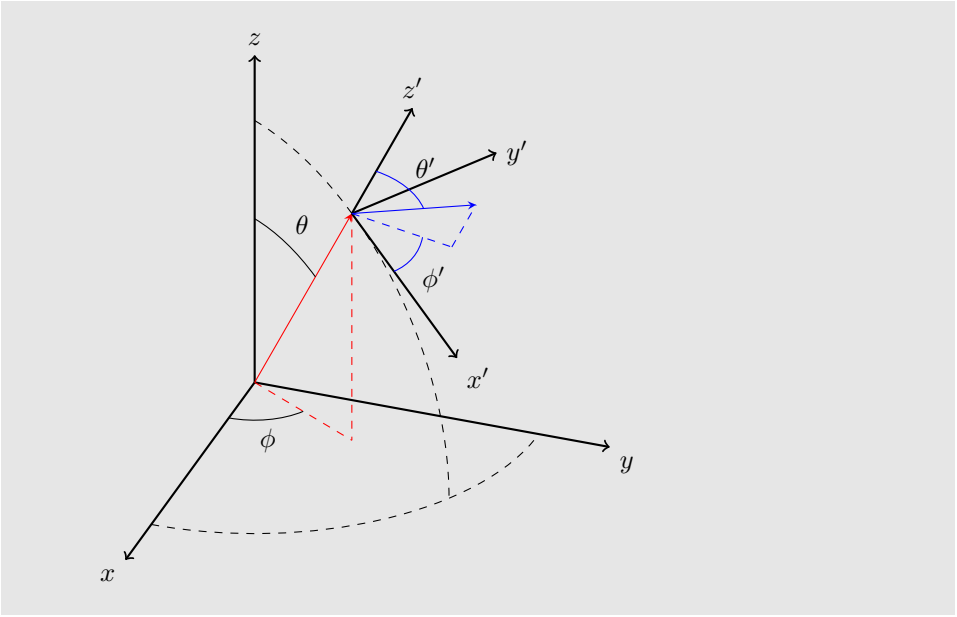
label options any style options for a Ti*k*Z`\node` object. If none, make sure to leave a blank delimiter `{}` in its place.

label any text for the Ti*k*Z `\node` which appears at the center of the arc. If none, make sure to leave a blank delimiter `{}` in its place.

Example:

```
\tdplotsetmaincoords{60}{110}
%
\pgfmathsetmacro{\rvec}{.8}
\pgfmathsetmacro{\thetavec}{30}
\pgfmathsetmacro{\phivec}{60}
%
\begin{tikzpicture}[scale=5,tdplot_main_coords]

    \coordinate (O) at (0,0,0);

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};
```

```
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \tdplotsetcoord{P}{\rvec}{\thetavec}{\phivec}
    \draw[-stealth,color=red] (O) -- (P);
    \draw[dashed, color=red] (O) -- (Pxy);
    \draw[dashed, color=red] (P) -- (Pxy);

    \tdplotdrawarc{(O)}{0.2}{0}{\phivec}{anchor=north}{$\phi$}

    \tdplotsetthetaplanecoords{\phivec}

    \tdplotdrawarc[tdplot_rotated_coords]{(0,0,0)}{0.5}{0}%
        {\thetavec}{anchor=south west}{$\theta$}

    \draw[dashed,tdplot_rotated_coords] (\rvec,0,0) arc (0:90:\rvec);
    \draw[dashed] (\rvec,0,0) arc (0:90:\rvec);

    \tdplotsetrotatedcoords{\phivec}{\thetavec}{0}
    \tdplotsetrotatedcoordsorigin{(P)}

    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (.5,0,0) node[anchor=north west]{$x'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.5,0) node[anchor=west]{$y'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.5) node[anchor=south]{$z'$};

    \draw[-stealth,color=blue,tdplot_rotated_coords] (0,0,0) -- (.2,.2,.2);
    \draw[dashed,color=blue,tdplot_rotated_coords] (0,0,0) -- (.2,.2,0);
    \draw[dashed,color=blue,tdplot_rotated_coords] (.2,.2,0) -- (.2,.2,.2);

    \tdplotdrawarc[tdplot_rotated_coords,color=blue]{(0,0,0)}{0.2}{0}%
        {45}{anchor=north west,color=black}{$\phi'$}

    \tdplotsetrotatedthetaplanecoords{45}

    \tdplotdrawarc[tdplot_rotated_coords,color=blue]{(0,0,0)}{0.2}{0}%
        {55}{anchor=south west,color=black}{$\theta'$}
\end{tikzpicture}
```



3.5.2 `tdplotdrawpolytopearc`

Description: Draws an arc using three user-specified points and a radius. A vertex determines the center of curvature, while two points define the angular extent and the plane of the arc. The three points must be specified in the corresponding macros before this command is issued.

Prerequisites: The three points must be specified by using the `\tdplotdefinepoints` command.

Syntax: `\tdplotdrawpolytopearc[draw style]{r}{label options}{label}`

Parameters:

(*Optional*) draw styles Optional argument containing draw styles for rendering the arc.

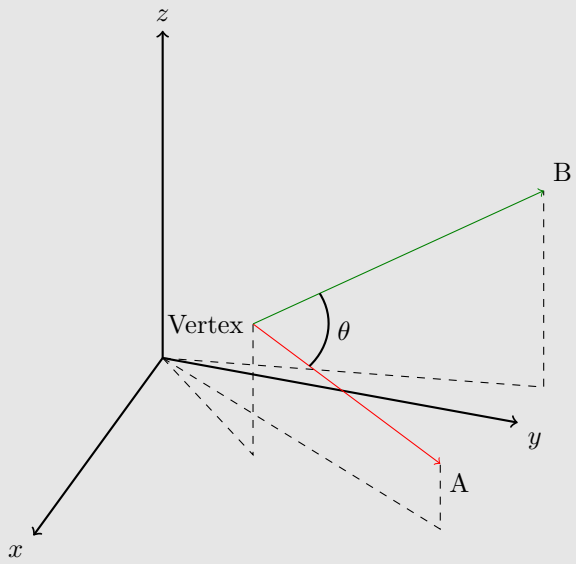
r The arc radius of curvature.

label options any style options for a Ti*k*Z`\node` object. If none, make sure to leave a blank delimiter `{}` in its place.

label any text for the Ti*k*Z `\node` which appears at the center of the arc. If none, make sure to leave a blank delimiter `{}` in its place.

Example:

```
\tdplotsetmaincoords{60}{110}
\begin{tikzpicture}[tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (5,0,0) node[anchor=north east]{$x$};
    \draw[thick,->] (0,0,0) -- (0,5,0) node[anchor=north west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,5) node[anchor=south]{$z$};

    \tdplotdefinepoints(2,2,2)(3,5,1)(-1,5,3)

    \draw[dashed] (0,0,0) -- (\tdplotvertexx,\tdplotvertexy,0) --
        (\tdplotvertexx,\tdplotvertexy,\tdplotvertexz);
    \draw[dashed] (0,0,0) -- (\tdplotax,\tdplotay,0)
        -- (\tdplotax,\tdplotay,\tdplotaz);
    \draw[dashed] (0,0,0) -- (\tdplotbx,\tdplotby,0)
        -- (\tdplotbx,\tdplotby,\tdplotbz);

    \draw[->,red] (\tdplotvertexx,\tdplotvertexy,\tdplotvertexz)
        -- (\tdplotax,\tdplotay,\tdplotaz);
    \draw[->,green!50!black] (\tdplotvertexx,\tdplotvertexy,\tdplotvertexz)
        -- (\tdplotbx,\tdplotby,\tdplotbz);

    \node[anchor=east] at (\tdplotvertexx,\tdplotvertexy,\tdplotvertexz){Vertex};
    \node[anchor=north west] at (\tdplotax,\tdplotay,\tdplotaz){A};
    \node[anchor=south west] at (\tdplotbx,\tdplotby,\tdplotbz){B};

    \tdplotdrawpolytopearc[thick]{1}{anchor=west}{$\theta$}
\end{tikzpicture}
```



3.6 The `tdplotsphericalsurfaceplot` Command

The `\tdplotsphericalsurfaceplot` command is quite complicated, and it seemed appropriate to occupy its own section. This command was initially developed to provide a method of rendering complex polar functions, $z = z(\theta, \phi)$, where the magnitude of the function is expressed by the radius, and the phase of the function is expressed by the hue, as

$$\begin{aligned} r &= |z(\theta, \phi)| \\ hue &= \text{Arg}\left[z(\theta, \phi)\right] \end{aligned} \tag{3.1}$$

The command has been generalized so that the hue can be specified in terms of the three polar coordinates, as

$$\begin{aligned} r &= f(\theta, \phi) \\ hue &= g(r, \theta, \phi) \end{aligned} \tag{3.2}$$

3.6.1 How `tdplotsphericalsurfaceplot` Works

To achieve the illusion of a 3d surface with proper persistence of vision, the `tikz-3dplot` package divides the drawing task into smaller sections. This division ensures the surface on the far side of the viewing perspective is properly occluded from view.

For a given perspective assigned by the main coordinate frame, a "view orientation" can be defined, giving the angles $(\theta_{view}, \phi_{view})$ that describe the orientation of the view. These angles determine how to dubdivide the surface rendering process. The following divisions are made:

- Divide the surface into "front" and "back", where the back is drawn before the front.
- Subdivide into "left" and "right".
- Subdivide further into "top" and "bottom".

The entire back half is drawn before the front half. For each half, the entire left or right side is drawn. For each side, all θ angles are drawn in wedges for each ϕ angle. When the back half is rendered, the θ angle is swept from θ_{view} toward the poles. When the front half is rendered, the θ angle is swept from the poles toward θ_{view}.

During this process, the x, y, and z axes are drawn at the appropriate time, ensuring the axes are occluded properly by the shape. The draw instructions for these axes are specified as user-defined parameters for this command.

3.6.2 Using `tdplotsphericalsurfaceplot`

Description: Draws a user-specified spherical polar function, with user-specified fill hues. Angular range to be displayed is specified with the `\tdplotsetpolarplotrange`

command. The line thickness can be specified by issuing the `\pgfsetlinewidth` PGF macro.

Syntax: `\tdplotsphericalsurfaceplot[fill color style]{theta steps}{phi steps}{function}{line color}{fill color}{x axis}{y axis}{z axis}`

Parameters:

(*Optional*) fill color style Specifies whether `fill color` is a function of (`\tdplotr`, `\tdplottheta`, `\tdplotphi`), or a direct Ti*k*Z color. Set to `parametricfill` to enable functional coloring.

theta steps The number of steps used to render the surface along the θ direction. For best results, this number should not be smaller than 12, and should be a factor of 360.

phi steps The number of steps used to render the surface along the ϕ direction. For best results, this number should not be smaller than 12, and should be a factor of 360.

function A mathematical expression, containing the variables `\tdplottheta` and `\tdplotphi`, used to define the radius of the surface for given angles. Note that the absolute value of the function is plotted.

line color Ti*k*Z color expression for surface lines.

fill color When the option *parametricfill* is used then this can be some mathematical expression containing `\tdplottheta` and `\tdplotphi`. If not, then this can be any Ti*k*Z expression for color. Note that if the function specified by `function` is negative, a shift of 180 is applied to the color. To avoid this, make sure `function` is always positive.

x axis Any draw commands used to render the x axis.

y axis Any draw commands used to render the y axis.

z axis Any draw commands used to render the z axis.

Example:

```
\tdplotsetmaincoords{70}{135}
\begin{tikzpicture}[scale=2,line join=bevel,tdplot_main_coords, fill opacity=.5]
\pgfsetlinewidth{.2pt}
\tdplotsphericalsurfaceplot[parametricfill]{72}{36}%
{sin(\tdplottheta)*cos(\tdplottheta)}{black}{\tdplotphi}%
    {\draw[color=black,thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};}%
    {\draw[color=black,thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};}%
    {\draw[color=black,thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};}%
    \node[tdplot_screen_coords,fill opacity=1] at (0,-1) {Parametric Fill in $\phi$};
\end{tikzpicture}
\begin{tikzpicture}[scale=2,tdplot_main_coords,line join=bevel,fill opacity=.8]
\pgfsetlinewidth{.1pt}
\tdplotsphericalsurfaceplot[parametricfill]{72}{36}%
{0.5*abs(cos(\tdplottheta))}{black}{2*abs(\tdplotr)}%
```

```
    {\draw[color=black,thick,->] (0,0,0)
                -- (1,0,0) node[anchor=north east]{$x$};}%
    {\draw[color=black,thick,->] (0,0,0)
                -- (0,1,0) node[anchor=north west]{$y$};}%
    {\draw[color=black,thick,->] (0,0,0)
                -- (0,0,1) node[anchor=south]{$z$};}%
    \node[tdplot_screen_coords,fill opacity=1] at (0,-1) {Parametric Fill in $r$};
\end{tikzpicture}
\begin{tikzpicture}[scale=2,line join=bevel,tdplot_main_coords, fill opacity=.7]
\pgfsetlinewidth{.4pt}
\tdplotsphericalsurfaceplot{72}{24}%
{0.5*cos(\tdplottheta)^2}{black}{red!80!black}%
    {\draw[color=black,thick,->] (0,0,0) -- (1,0,0) node[anchor=north east]{$x$};}%
    {\draw[color=black,thick,->] (0,0,0) -- (0,1,0) node[anchor=north west]{$y$};}%
    {\draw[color=black,thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};}%

    \node[tdplot_screen_coords,fill opacity=1] at (0,-1) {Solid Fill};
\end{tikzpicture}
```

3.6.3 The `tdplotsetpolarplotrange` Command

Description: Defines the range of angles to be displayed when using `\tdplotsphericalsurfaceplot`

Syntax: `\tdplotsetpolarplotrange{lowertheta}{uppertheta}{lowerphi}{upperphi}`

Parameters:

lowertheta The lower limit for `\tdplottheta`, in degrees.

uppertheta The upper limit for `\tdplottheta`, in degrees.

lowerphi The lower limit for `\tdplotphi`, in degrees.

upperphi The upper limit for `\tdplotphi`, in degrees.

Example:

```
\tdplotsetmaincoords{60}{110}
\begin{tikzpicture}[scale=2,line join=bevel,tdplot_main_coords,%
    fill opacity=.5]

\tdplotsetpolarplotrange{90}{180}{180}{360}
\tdplotsphericalsurfaceplot[parametricfill]{72}{36}%
{.5}{black}{\tdplotphi + 3*\tdplottheta}%
    {\draw[color=black,thick,->] (0,0,0)
            -- (1,0,0) node[anchor=north east]{$x$};}%
    {\draw[color=black,thick,->] (0,0,0)
            -- (0,1,0) node[anchor=north west]{$y$};}%
    {\draw[color=black,thick,->] (0,0,0)
            -- (0,0,1) node[anchor=south]{$z$};}%
\end{tikzpicture}
```

3.6.4 The `tdplotresetpolarplotrange` Command

Description: Resets the range of angles to the default full range when using `\tdplotsphericalsurfaceplot`

Syntax: `\tdplotresetpolarplotrange`

3.6.5 The `tdplotshowargcolorguide` Command

Description: Draws a "color guide" table which associates the hue of a parametric polar plot with an angle. Guide is drawn at user-specified screen coordinates with user-specified size. This guide is intended to illustrate the complex phase representation of the surface for a given θ and ϕ coordinate.

Syntax: `\tdplotshowargcolorguide{x position}{y position}{x size}{y size}`

Parameters:

x position The x screen coordinate to place the lower-left corner of the guide.

y position The y screen coordinate to place the lower-left corner of the guide.

x size The width of the color guide.

y size The height of the color guide.

Example:

```
\tdplotsetmaincoords{40}{0}
\begin{tikzpicture}[scale=2,line join=bevel,tdplot_main_coords,%
    fill opacity=1]
\tdplotsphericalsurfaceplot[parametricfill]{72}{36}%
    {sqrt(15/2)/2*sin(\tdplottheta)^2}{black}%
    {2*\tdplotphi - 6 * \tdplottheta}{}{}{}%
\tdplotshowargcolorguide{3}{-.2}{.1}{1}
\end{tikzpicture}
```

3.7 Miscellaneous Math Commands

The following commands are used to streamline the `tikz-3dplot` calculations in the background. There is generally no need to use these directly, but may be useful on their own for any desired calculations.

3.7.1 tdplotsinandcos

Description: Determines the sine and cosine of the specified angle, and stores in specified macros.

Syntax: `\tdplotsinandcos{sintheta}{costeta}{theta}`

Parameters:

sintheta A macro (eg. `\sintheta`) to store the sine of `theta`.

costheta A macro (eg. `\costheta`) to store the cosine of `theta`.

theta An angle (in degrees) to calculate. Can be a macro or literal value.

3.7.2 tdplotmult

Description: Determines the product of two specified values, and stores the result in the specified macro.

Syntax: `\tdplotmult{result}{multiplicand}{multiplicator}`

Parameters:

result A macro (eg. `\result`) to store the product of `multiplicand * multiplicator`.

multiplicand The multiplicand of the product. Can be a macro or literal value.

multiplicator The multiplicator of the product. Can be a macro or literal value.

3.7.3 tdplotdiv

Description: Determines the quotient of two specified values, and stores the result in the specified macro.

Syntax: `\tdplotdiv{result}{dividend}{divisor}`

Parameters:

result A macro (eg. `\result`) to store the quotient of `dividend / divisor`.

dividend The dividend of the quotient. Can be a macro or literal value.

divisor The divisor of the quotient. Can be a macro or literal value.

Chapter 4

Known Issues

There are various issues that have been found while developing the `tikz-3dplot` package. Some of these are currently open problems which will hopefully be resolved. Feedback and suggestions are welcome.

4.1 Predefined Points Don't Work in Rotated Frame

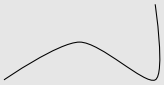
When a coordinate is defined using the Ti*k*Z command `\coordinate`, it will be transformed by the transformation specified at the beginning of the `tikzpicture` environment. These coordinates will not work for transformations applied at the actual `\draw` command.

This problem seems to be inherant with Ti*k*Z itself. By way of example, the following code is taken right from the PGF manual Version 2.00, section 21.2 on page 218:

```
Case A:
%this one works fine using literal coordinates
\begin{tikzpicture}[smooth]
    \draw plot coordinates{(1,0)(2,0.5)(3,0)(3,1)};
    \draw[x={(0cm,1cm)},y={(1cm,0cm)},color=red]
        plot coordinates{(1,0)(2,0.5)(3,0)(3,1)};
\end{tikzpicture}

Two distinct paths shown.  All is good.

Case B:
%this one does not work using predefined coordinates
\begin{tikzpicture}[smooth]
    \coordinate (A) at (1,0);
    \coordinate (B) at (2,0.5);
    \coordinate (C) at (3,0);
    \coordinate (D) at (3,1);
    \draw plot coordinates{(A)(B)(C)(D)};
    \draw[x={(0cm,1cm)},y={(1cm,0cm)},color=red] plot coordinates{(A)(B)(C)(D)};
```

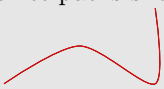
```
\end{tikzpicture}

Both paths draw overtop each other, and the coordinates are not transformed!
```

Case A:



Two distinct paths shown. All is good.

Case B:



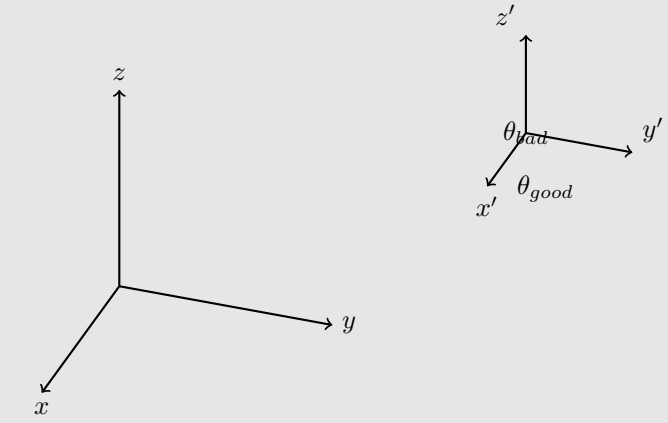
Both paths draw overtop each other, and the coordinates are not transformed!

4.2 `node` Command and `shift=(P)` Issues

When placing a node in a shifted coordinate frame, the `\node` command will not position properly. As a workaround, the `\draw` command must be used to position the node. By way of example.

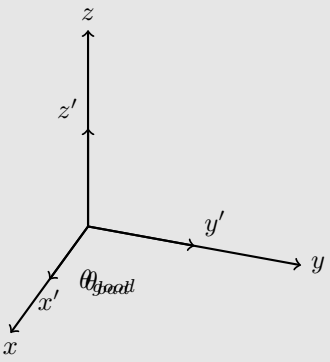
```
Case A:
\tdplotsetmaincoords{60}{110}
\begin{tikzpicture}[scale=3,tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

    \coordinate (P) at (3,3,3);
    \tdplotsetrotatedcoords{0}{0}{0}
    \tdplotsetrotatedcoordsorigin{(P)}

    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (.5,0,0) node[anchor=north]{$x'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.5,0) node[anchor=south west]{$y'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.5) node[anchor=south east]{$z'$};

    \node[tdplot_rotated_coords] at (30:.5){$\theta_{bad}$};
    \draw[tdplot_rotated_coords] (0,0,0) + (30:.5) node{$\theta_{good}$};
\end{tikzpicture}
```

```
Here, the rotated coordinate frame is shifted by amount (P) within the main%
coordinate frame.  The node labelled $\theta_{bad}$ does not accept any%
positioning coordinates.

Case B:
\tdplotsetmaincoords{60}{110}
\begin{tikzpicture}[scale=3,tdplot_main_coords]

    \draw[thick,->] (0,0,0) -- (1,0,0) node[anchor=north]{$x$};
    \draw[thick,->] (0,0,0) -- (0,1,0) node[anchor=west]{$y$};
    \draw[thick,->] (0,0,0) -- (0,0,1) node[anchor=south]{$z$};

%   \coordinate (P) at (3,3,3);
    \tdplotsetrotatedcoords{0}{0}{0}
%   \tdplotsetrotatedcoordsorigin{(P)}

    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (.5,0,0) node[anchor=north]{$x'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,.5,0) node[anchor=south west]{$y'$};
    \draw[thick,tdplot_rotated_coords,->] (0,0,0)
        -- (0,0,.5) node[anchor=south east]{$z'$};

    \node[tdplot_rotated_coords] at (30:.5){$\theta_{bad}$};
    \draw[tdplot_rotated_coords] (0,0,0) + (30:.5) node{$\theta_{good}$};
\end{tikzpicture}

Here, the shift is removed from the rotated coordinate frame.  The %
previously failing \verb|\node| command works properly.
```



Case A:

Here, the rotated coordinate frame is shifted by amount (P) within the main coordinate frame. The node labelled θ_{bad} does not accept any positioning coordinates.



Case B:

Here, the shift is removed from the rotated coordinate frame. The previously failing `\node` command works properly.

4.3 PGF `xyz spherical` Coordinate System

I have recently heard about the `xyz spherical` coordinate system offered by PGF. Unfortunately, when I try to use it, I get compile errors. I haven't spent much time looking into it though, so I'm probably just doing something silly.

```
\draw[-stealth,color=orange] (0,0,0)
    -- (xyz spherical cs:radius=.5,longitude=60,latitude=120);
%this gives the following compile error using MikTeX 2.8:
%   Undefined control sequence. <argument> \tikz@cs@radius.
```

Chapter 5

TODO list

This chapter contains notes and jots of ideas of things to do which can expand or improve the `tikz-3dplot`package.

- Figure out how to work in a variable scope that doesn't interfere with other packages.
- Find a way to check if TikZ is loaded, and give a compile error if necessary.
- Find a way to use predefined coordinates in rotated or translated coordinate frames, instead of just literal coordinates.
- Generalize matrix math if such a package exists.
- Look into using TikZ spherical polar coordinates explicitly to streamline coordinate definitions.
- Find a way to extract coordinate components defined by the `\coordinate` command and use them in macros defined by the `\pgfmathsetmacro` and `\pgfmathparse` commands.