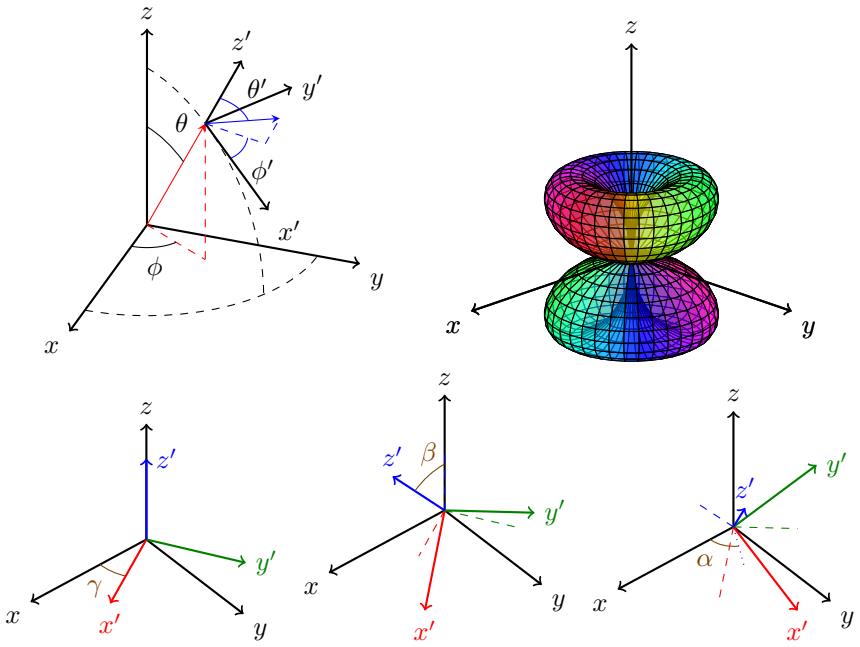
# The `tikz-3dplot` Package

Jeff Hein
http://tikz3dplot.wordpress.com

September 24, 2017



# 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 $\theta$ and $\phi$. With this function, the surface hue can be given explicitly, or expressed as a user-defined function of $r$, $\theta$, and $\phi$. This command is discussed in Section **??**.

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

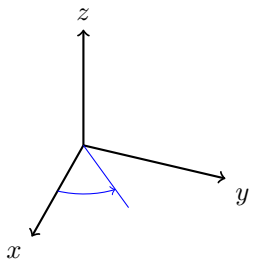



Figure 1.1: `tikz-3dplot` coordinate and positive angle convention.

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 **??**.

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:

- Ti*k*Z
- `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 $\theta_d$ and $\phi_d$. 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 $\theta_d$, and
- Rotate the coordinate system about the (rotated) body $z$ axis by the amount $\phi_d$.

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

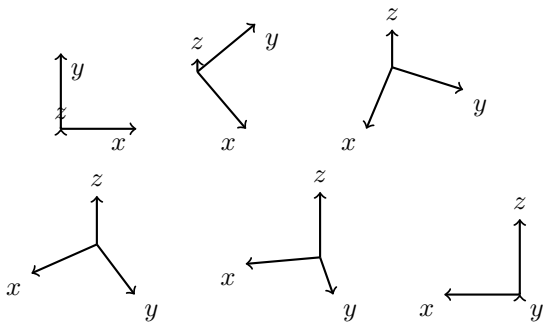



Figure 2.1: Examples of coordinate systems for various choices of $\theta_d$ and $\phi_d$.

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 $\gamma$ about the world $z$ axis,
- Rotate by angle $\beta$ about the (unrotated) world $y$ axis, and
- Rotate by angle $\alpha$ about the (unrotated) world $z$ axis.

These rotations are shown in Figure 2.2.

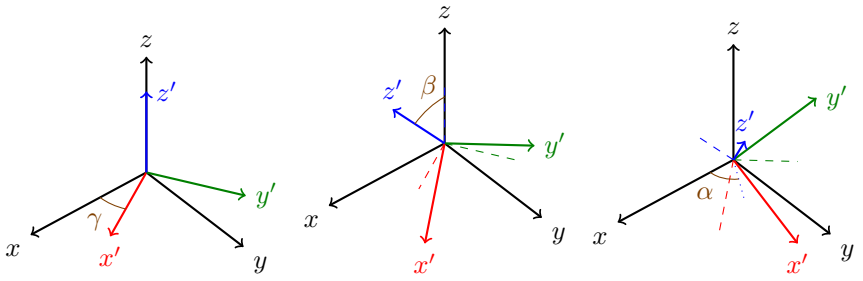



Figure 2.2: Positioning the rotated coordinate frame $(x', y', z')$ using Euler angles $(\alpha, \beta, \gamma)$.

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.

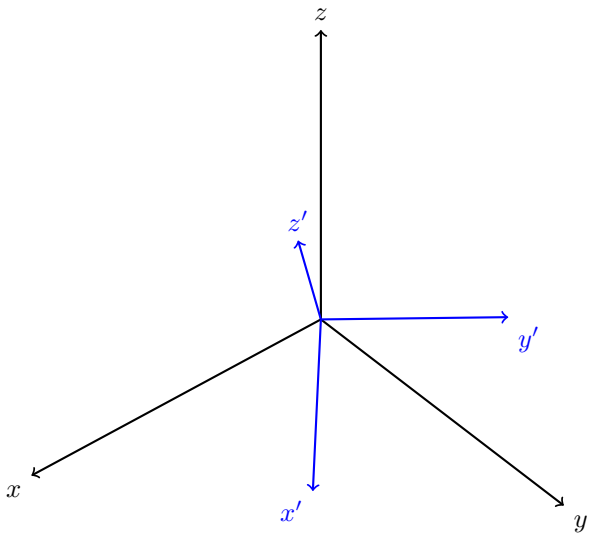



Figure 2.3: The rotated coordinate frame $(x', y', z')$ displayed within the main coordinate frame $(x, y, z)$. Both are completely specified by user-defined angles: $(\theta_d, \phi_d)$ for the main coordinate frame, and $(\alpha, \beta, \gamma)$ for the rotated coordinate frame.

## 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 $\theta$ of some vector. This $\theta$ arc exists in a plane which contains the $z$ axis, and is rotated about the $z$ axis by the angle $\phi$ from the $xz$ plane. For lack of a better name, this plane is referred to as the "theta plane" within a given coordinate system.

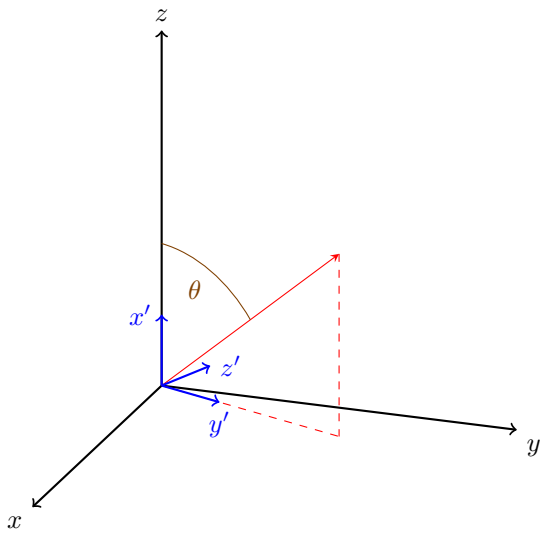



Figure 2.4: Drawing arcs outside the $xy$ plane by using a rotated coordinate frame in the "theta plane" of the main coordinate frame.

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 $(\theta_d, \phi_d)$. $\theta_d$ denotes the rotation around the $x$ axis, while $\phi_d$ 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{ $\theta_d$ }{ $\phi_d$ }

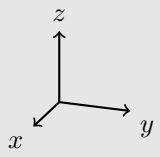
**Parameters:**

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

$\phi_d$ 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 $(\alpha, \beta, \gamma)$. Rotations use the $z(\alpha)y(\beta)z(\gamma)$ convention of Euler rotations, where the

system is rotated by $\gamma$ about the $z$ axis, then $\beta$ about the (world) $y$ axis, and then $\alpha$ about the (world) $z$ axis.

**Syntax:** `\tdplotsetrotatedcoords{`$\alpha$`}{`$\beta$`}{`$\gamma$`}`

**Parameters:**

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

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

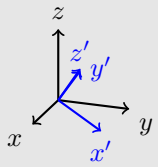
$\gamma$ The angle (in degrees) through which the rotated frame is rotated about the world $z$ axis.

**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.

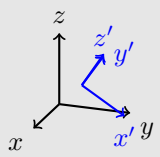
**Example:**

```
\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 $\theta$ 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{`$\phi$`}`

**Parameters:**

$\phi$ 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}
```