The makeshape package*
and
A Method for Creating Custom Shapes in PGF

Adrian P. Robson†

25 January 2013

The makeshape package simplifies writing PGF shapes. Declaring a custom shape with a correct anchor border can be difficult. Complex shapes often need complicated calculations to find the touching point of a connecting line. This package only requires that a developer write a PGF path describing the anchor border. It also provides macros that help with the management of shape parameters and the definition of anchor points.

1 Introduction

The `\pgfdeclareshape` command can be used to create new shapes with the PGF package ([1], *Declaring New Shapes*, §75.5, pp 625-631). The following are typically needed:

- A shape name.
- Code for computing saved anchors and saved dimensions.
- Code for computing anchor positions in terms of the saved anchors.
- Code for computing border anchors (`\anchorborder`).
- Code for drawing a background path.

Writing these can be hard for completely new shapes, and in particular the declaration of a suitable `\anchorborder` command can be very difficult for complex shapes.

This paper presents a method that makes the process of writing new shapes from scratch a little easier. Its key features are: a mechanism for specifying the `\anchorborder` behaviour as a path; a package that provides a set of useful macros for writing and allocating anchor and background paths; and an example `\pgfdeclareshape` command that can be used as a template for new shapes.

The method involves writing some original code and modifying the template. In brief, the process involves:

1. Writing *background path* and *anchor path* macros.
2. Writing one or more *saved anchors*.
3. Writing *anchors* using the above saved anchors.

*This document corresponds to makeshape 2.1, dated 2013/01/25.

†adrian.robson@nepsweb.co.uk

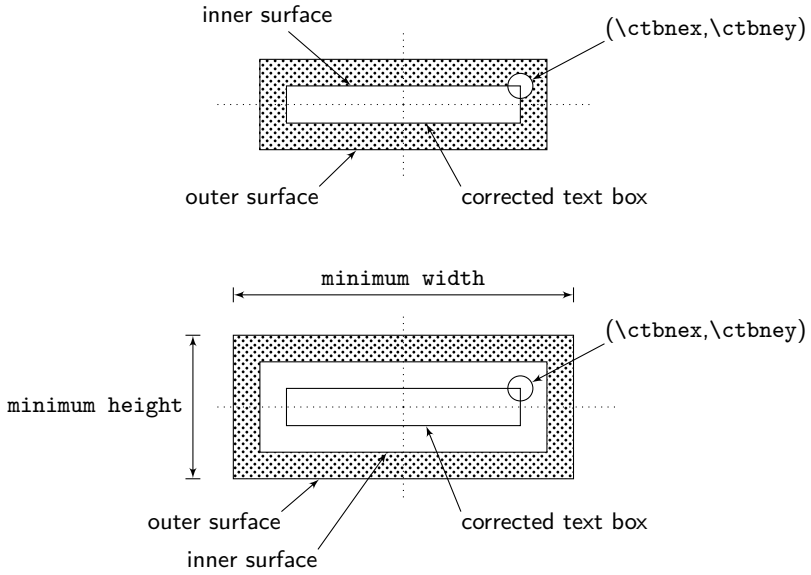


Figure 1: The Background Path

2 Preliminaries

2.1 Background Path Macro

The *background path*, as illustrated in figure 1, describes the curves that form the actual shape. It can be just a single line and does not have to be closed. However, there are no limits on is complexity. If it is more than a simple outline, then its *inner* and *outer* surfaces need to be considered separately.

The inner surface should contain the *corrected text box*, which is the shape's text box surrounded by the space specified by its `inner xsep` and `inner ysep` keys. The coordinates of this box are given by the package's `\ctbnex` and `\ctbnex` macros, which are described in §3.1.1.

The outer surface should have height and width dimensions that are the same as the shape's `minimum width` and `minimum height` keys if these are larger than the basic dimensions. These keys can be obtained with the `\pgfshapeminwidth` and `\pgfshapeminheight` macros, as described in §3.1.3. The `\mincorrect` macro described in §3.1.2 can be used make the necessary comparison and assignment.

2.1.1 Background Path Algorithm

The background path should be implemented as a macro with the following structure:

1. Use `\ctbnex` and `\ctbnex`, which give the north east corner of the corrected text box, to calculate significant reference coordinates.

2. Use `\mincorrect` to modify the reference coordinates to take account of the shape's minimum dimension keys, which are given by `\pgfshapeminwidth` and `\pgfshapeminheight`.

3. Use the reference coordinates from step 2 to draw the actual path using PGF path commands ([1], *Constructing Paths*, §71, pp 579-589). Typical commands are `\pgfpathmoveto`, `\pgfpathlineto` and `\pgfpathclose`. The `\pgfpoint` command will also be needed. Do this for

   (a) the outer surface curve, and

   (b) curves in the shape's inner path space

   Usually when a path is defined, it is displayed with `\pgfusepath{stroke}`. However, it is not needed here.

2.2 Anchor Path Macro

The *anchor path*, as illustrated in figure 2, describes the curve that forms the surface where connecting lines meet the shape. It it must be a single closed path, and it is normally strongly related to background path's outer surface (see §2.1).

Like the background path, it should use the fundamental reference point of the *corrected text box*, which automatically includes the shape's inner separation. It should also take regard of the shape's `minimum width` and `minimum height` keys if these are larger than the basic dimensions. Furthermore, it should compensate for the `outer xsep` and `outer ysep` keys as shown in figure 2.

The corrected text box macros are decribed in §3.1.1. The minimum dimension keys are given by `\pgfshapeminwidth` and `\pgfshapeminheight`, and the outer separation keys are given by `\pgfshapeouterxsep` and `\pgfshapeouterysep`, as described in §3.1.3. The minimum dimension comparision and assignment is provided by `\mincorect` macro, which is discussed in §3.1.2.

2.2.1 Anchor Path Algorithm

The anchor path should be implemented as a macro with the following structure:

1. Use `\ctbnex` and `\ctbnex`, which give the north east corner of the corrected text box, to calculate significant reference coordinates. (Normally the same as §2.1.1 step 1).

2. Use `\mincorrect` to modify the reference coordinates to take account of the shape's minimum dimension keys, which given by `\pgfshapeminwidth` and `\pgfshapeminheight`. (Normally the same as §2.1.1 step 2).

3. Modify the reference coordinates to take account of outer separation, which is given by `\pgfshapeouterxsep` and `\pgfshapeouterysep`.

4. Use the corrected reference coordinates from step 3 to specify the anchor path using PGF path commands ([1], *Constructing Paths*, §71, pp 579-589). The path should be closed so a `\pgfpathclose` command is required. The path will not actually be drawn, and the `\pgfusepath{stroke}` macro should not be used. (Normally the same as §2.1.1 step 3a).

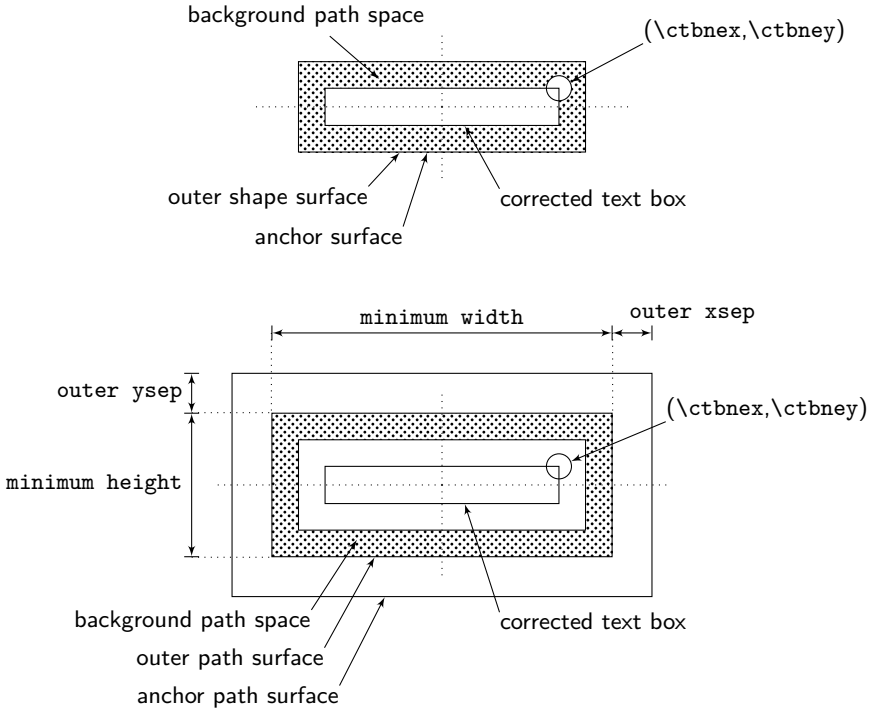


Figure 2: The Anchor Path

2.3 Anchors

A PGF shape requires the definition of a set of anchor point commands, and these need *saved anchors* to provide necessary coordinates. All anchor macros should return a point coordinate by assigning values to the registers `\pgf@x` and `\pgf@y`.

Like the anchor path macro, saved anchors should use the fundamental reference point of the *corrected text box*, which automatically includes the shape's inner separation. They should also take regard of the shape's `minimum width` and `minimum height` keys if these are larger than the basic dimensions. Furthermore, they should compensate for the `outer xsep` and `outer ysep` keys as shown in figure 2.

The corrected text box macros are decribed in §3.1.1. The minimum dimension keys are given by `\pgfshapeminwidth` and `\pgfshapeminheight`, and the outer separation keys are given by `\pgfshapeouterxsep` and `\pgfshapeouterysep`, as described in §3.1.3. The minimum dimension comparision and assignment is provided by `\mincorect` macro, which is discussed in §3.1.2.

2.3.1 Saved Anchor Algorithm

A saved anchor must set the PGF registers `\pgf@x` and `\pgf@y` with the required coordinates. It has a structure similar to the anchor path.

1. Use `\ctbnex` and `\ctbnex`, which give the north east corner of the corrected text box, to calculate significant reference coordinates. (Normally similar to

§2.2.1 step 1).

2. Use `\mincorrect` to modify the reference coordinates to take account of the shape's minimum dimension keys, which given by `\pgfshapeminwidth` and `\pgfshapeminheight`. (Normally the same as §2.2.1 step 2).

3. Modify the reference coordinates to take account of outer separation, which is given by `\pgfshapeouterxsep` and `\pgfshapeouterxsep`. (Normally the same as §2.2.1 step 3).

4. Use the reference coordinates from step 3 to calculate the location of the anchor point, and assign values to the `\pgf@x` and `\pgf@y` registers.

3 Making a Shape

3.1 Helper Macros

The following macros are available for use in the anchor path, background path and saved anchors.

3.1.1 Corrected text box

The *corrected text box* is a bounding box for the shape's text that includes the shape's `inner xsep` and `inner ysep` keys. The box is symmetric about the origin. Its x and y coordinates are given by the following commands:

`\ctbnex`: The x-coordinate of the north east corner of the shape's corrected text box.

`\ctbney`: The y-coordinate of the north east corner of the shape's corrected text box.

3.1.2 Correcting for minimum dimensions

A shape's size should be corrected for its `minimum width` and `minimum height` keys. The `\mincorrect` macro is provided to help with this.

`\mincorrect{`*dimreg*`}{`*minkey*`}`: Correct an x or y outer bound dimension for `minimum width` or `minimum height` keys.

dimreg – A dimension register. On entry, it should hold the normal dimension of the shape. On exit, *dimreg* is assigned the larger of its original value or *minkey*/2.

minkey – A minimum dimension key value. In practice, one of the minimum height or width commands given in §3.1.3 below.

3.1.3 Getting key values

The shape's key values can be accesses with the following commands:

key	command
`minimum width`	`\pgfshapeminwidth`
`minimum height`	`\pgfshapeminheight`
`outer xsep`	`\pgfshapeouterxsep`
`outer ysep`	`\pgfshapeouterysep`

The outer separation keys are not relevant to the background path, but should be used by the anchor path and saved anchors.

The shape's inner separation keys are automatically included in the *corrected text box* coordinates `\ctbnex` and `\ctbney`.

3.1.4 Testing support

There are a couple of macros can be used in the background path to delineate a shape's text box during development:

`\path@textbox` draws a line around the the shape's text that is the uncorrected text box.

`\path@ctextbox` shows the shape's corrected text box, which includes inner separation.

These macros should not be used in 'production' shapes.

3.2 Procedure

The procedure in outline for making a new shape is as follows:

1. Create the border path
2. Create the anchor path
3. Declare the shape
   (a) Use `\setpaths` to employ the paths
   (b) Create the anchors

The sample shape given in §4.4 can be used as a template to create a new shape. The code can be copied from there, or it can be acquired as the file `sampleshape.tex`, which is part of the package's distribution.

The procedure for adapting the template code is as follows:

1. Change the name of the shape in the `\pgfdeclareshape` command from '`sample`' to the name of the new shape.

2. Write *anchor path* and *background path* macros.

   These and any sub macros are placed at the start of the file and replace the sample `\sampleanchor` and `\sampleborder` macros.

   As discussed in §2.1 and §2.2, the background path should take account of the minimum dimension keys, and the anchor path should handle the minimum keys, and outer separation. The macros given in §3.1 are provided to help with this, and the sample code gives an indication of how they can be used.

3. Change the arguments of `\setpaths` in the template to the macros written in step 2. The first is the anchor path and the second is the background path.

4. Write the saved anchors that will needed by the anchors written in step 5.

   Saved anchors, like the anchor path written in step 2, should correct for outer separation, and minimum dimensions using the macros in §3.1. They should set the `\pgf@x` and `\pgf@y` registers.

5. Write anchors that set `\pgf@x` and `\pgf@y` registers using the saved anchors from step 4.

   (a) Write anchors for the cardinal and sub cardinal points:

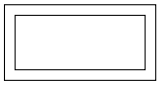
   ```
   \anchor{north}{ ... }
   \anchor{north east}{ ... }
   \anchor{east}{ ... }
   \anchor{south east}{ ... }
   \anchor{south}{ ... }
   \anchor{south west}{ ... }
   \anchor{west}{ ... }
   \anchor{north west}{ ... }
   ```

   (b) Write any additional anchors that the shape needs.

The PGF registers used throughout this method of declaring new shapes have the `@` character in their names. Unfortunately, this character has a special role in LaTeX. So either the `\makeatletter` and `\makeatother` commands must be used as in `sampleshape.tex`, or the shape must be defined in a `sty` file.

4 Path and Anchor Examples

Here an example of an actual shape that is implemented using the makeshape method:



Its background path, anchor path, and anchor points use a common macro that gives the space between the outer and inner boundaries of the shape.

```
\def\gap{4pt}
```

4.1 Background Path

In this example, the background path macro is called `sampleshape`.

```
1 \def\sampleshape{
```

First we get the corrected text box coordinates, and add space for the shape using the `\gap` macro.

```
2     \pgf@xa=\ctbnex
3     \pgf@ya=\ctbney
4     \advance\pgf@xa by \gap
5     \advance\pgf@ya by \gap
```

Then the standard correction for minimum size is applied using the `\mincorrect` macro.

```
    \mincorrect{\pgf@xa}{\pgfshapeminwidth}
    \mincorrect{\pgf@ya}{\pgfshapeminheight}
```

Using these coordinates the outer boundary of the shape can be drawn

```
    \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
    \pgfpathclose
```

Finally, the reference coordinates are moved to the inner boundary and the shape is completed

```
    \advance\pgf@xa by -\gap
    \advance\pgf@ya by -\gap
    \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
    \pgfpathclose
}
```

4.2 Anchor Path

The shape's anchor path macro called `sampleanchor`, and is similar to the path macro in §4.1, but it also handles outer separation.

```
\def\sampleanchor{
```

First we get the corrected text box coordinates, and add space for the shape using the `\gap` macro.

```
    \pgf@xa=\ctbnex
    \pgf@ya=\ctbney
    \advance\pgf@xa by \gap
    \advance\pgf@ya by \gap
```

Then corrections for minimum size and outer separation are applied:

```
    \mincorrect{\pgf@xa}{\pgfshapeminwidth}
    \advance\pgf@xa\pgfshapeouterxsep
    \mincorrect{\pgf@ya}{\pgfshapeminheight}
    \advance\pgf@ya\pgfshapeouterysep
```

Using these coordinates the anchor boundary path is finally drawn

```
    \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
    \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
    \pgfpathclose
}
```

4.3 Shape Declaration and Anchor Points

The `sample` shape declaration is started with

```
\pgfdeclareshape{sample}{
```

Then the shape and anchor paths, which were defined in §4.1 and §4.2, are employed with

```
   \setpaths{\sampleanchor}{\sampleshape}
```

4.3.1 Anchors

Next in the shape's declaration are its saved anchors. In this case, we establish the north east anchor location with code that is equivalent to the first part of the anchor path macro (§4.2). Note how the anchor point's coordinate is returned in `\pgf@x` and `\pgf@y`.

```
   \savedanchor{\northeast}{
      \pgf@x = \ctbnex
      \pgf@y = \ctbney
      \advance\pgf@xa by \gap
      \advance\pgf@ya by \gap
      \mincorrect{\pgf@x}{\pgfshapeminwidth}
      \mincorrect{\pgf@y}{\pgfshapeminheight}
      \advance\pgf@x\pgfshapeouterxsep
      \advance\pgf@y\pgfshapeouterysep
   }
```

The saved anchor is then used to construct the shape's anchors, and the declaration is ended as follows:

```
   \anchor{north}{\northeast \pgf@x=0pt}
   \anchor{north east}{\northeast}
   \anchor{east}{\northeast \pgf@y=0pt}
   \anchor{south east}{\northeast \pgf@y=-\pgf@y}
   \anchor{south}{\northeast \pgf@x=0pt \pgf@y=-\pgf@y}
   \anchor{south west}{\northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y}
   \anchor{west}{\northeast \pgf@x=-\pgf@x \pgf@y=0pt}
   \anchor{north west}{\northeast \pgf@x=-\pgf@x}
}
```

4.4 Sample Shape Code

The following is a full listing of the code needed to make the sample shape described above. It can be used as a templete for new shapes. A copy of this code can be found in the the file `sampleshape.tex`, which is included in the package's distribution. Note the use of the `\makeatletter` and `\makeatother` commands, which are required because the sample shape is in a `tex` file.

```
\usepackage{makeshape}
\makeatletter
```

```
%% Constant for sample shape
\def\gap{4pt}

%% Anchor path <- change and rename as required
\def\sampleanchor{
   % get corrected text box
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
   % make room for shape
   \advance\pgf@xa by \gap
   \advance\pgf@ya by \gap
   % correct for minheight and minwidth and
   % for outerxsep or outerysep
   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
   \advance\pgf@xa\pgfshapeouterxsep
   \mincorrect{\pgf@ya}{\pgfshapeminheight}
   \advance\pgf@ya\pgfshapeouterysep
   % draw the path
   \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
   \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
   \pgfpathclose
}

%% Background path <- change and rename as required
\def\sampleborder{
   % get corrected text box
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
   % make room for shape
   \advance\pgf@xa by \gap
   \advance\pgf@ya by \gap
   % correct for minheight and minwidth but
   %    not for outerxsep or outerysep
   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
   \mincorrect{\pgf@ya}{\pgfshapeminheight}
   % draw outer shape
   \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
   \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
   \pgfpathclose
   % draw inner shape
   \advance\pgf@xa by -\gap
   \advance\pgf@ya by -\gap
   \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
   \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
   \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
```

```
   \pgfpathclose
}

%%-------------------------------------------------
%% Shape declaration <- Change name as required
\pgfdeclareshape{sample}{

   % Set paths <- change path macros as required
   \setpaths{\sampleanchor}{\sampleborder}

   % Saved anchors <- change as required
   \savedanchor{\northeast}{
      \pgf@x = \ctbnex
      \pgf@y = \ctbney
      \advance\pgf@x by \gap
      \advance\pgf@y by \gap
      \mincorrect{\pgf@x}{\pgfshapeminwidth}
      \mincorrect{\pgf@y}{\pgfshapeminheight}
      \advance\pgf@x\pgfshapeouterxsep
      \advance\pgf@y\pgfshapeouterysep
   }

   % Anchors <- change as required
   \anchor{north}{\northeast \pgf@x=0pt}
   \anchor{north east}{\northeast}
   \anchor{east}{\northeast \pgf@y=0pt}
   \anchor{south east}{\northeast \pgf@y=-\pgf@y}
   \anchor{south}{\northeast \pgf@x=0pt \pgf@y=-\pgf@y}
   \anchor{south west}{\northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y}
   \anchor{west}{\northeast \pgf@x=-\pgf@x \pgf@y=0pt}
   \anchor{north west}{\northeast \pgf@x=-\pgf@x}

}

\makeatother
```

References

[1] Till Tantau, *The TikZ and PGF Packages, Manual for version 2.10*, 2010. Available as `pgfmanual.pdf` from the Comprehensive TeX Archive Network.