1 3D Tools

TikZ Library 3dtools

```
\usetikzlibrary{3dtools} % LaTeX and plain TeX
\usetikzlibrary[3dtools] % ConTeXt
```

This library provides additional tools to create 3d–like pictures.

TikZ has the `3d` and `tpp` libraries which deal with the projections of three–dimensional drawings. This library provides some means to manipulate the coordinates. It supports linear combinations of vectors, vector and scalar products. **Note:** Hopefully this library is only temporary and its contents will be absorbed in slightly extended versions of the `3d` and `calc` libraries.

1.1 Coordinate computations

The 3dtools library has some options and styles for coordinate computations.

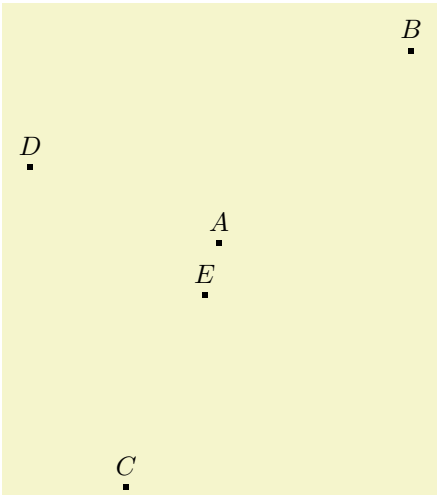
`/tikz/3d parse` (no value)

Parses and expression and inserts the result in form of a coordinate.

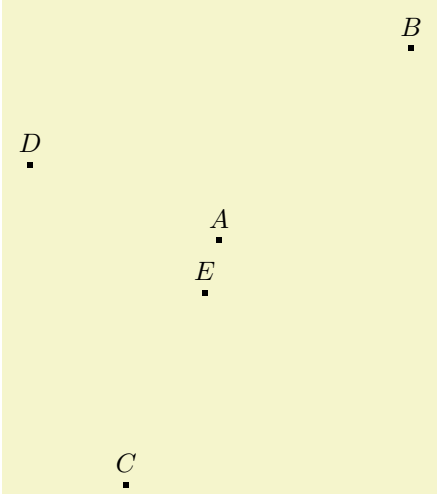
`/tikz/3d coordinate` (no value)

Allow one to define a 3d coordinate from other coordinates.

Both keys support both symbolic and explicit coordinates but for the explicit ones one needs additional braces.




```
\begin{tikzpicture}
 \path (1,2,3) coordinate (A)
  (2,3,-1) coordinate (B)
  (-1,-2,1) coordinate (C)
  [3d parse={0.25*(1,2,3)x(B)}]
  coordinate(D)
  [3d parse={0.25*(C)x(B)}]
  coordinate(E);
 \path foreach \X in {A,...,E}
 {(\X) node[fill,inner sep=1pt,
 label=above:£\X£]{}};
\end{tikzpicture}
```




```
\begin{tikzpicture}
 \path (1,2,3) coordinate (A)
  (2,3,-1) coordinate (B)
  (-1,-2,1) coordinate (C)
  [3d coordinate={(D)=0.25*(1,2,3)x(B)},
  3d coordinate={(E)=0.25*(C)x(B)}];
 \path foreach \X in {A,...,E}
 {(\X) node[fill,inner sep=1pt,
 label=above:£\X£]{}};
\end{tikzpicture}
```

The library comes also with a function `\pgfmathtdparse` that allows one to parse 3d expressions. The supported vector operations are `+` (addition $+$), `-` (subtraction $-$), `*` (multiplication of the vector by a scalar), `x` (vector product $\times$) and `o` (scalar product).

`\pgfmathtdparse{`$\langle x\rangle$`}`

Parses 3d expressions.

0.0,0.0,1.0

```
\pgfmathtdparse{(1,0,0)x(0,1,0)}\pgfmathresult
```

In order to pretty-print the result one may want to use `\pgfmathprintvector`, and use the math function `TD` for parsing.

`\pgfmathprintvector{`$\langle x\rangle$`}`

Pretty-prints vectors.

$0.2\,\vec{A} - 0.3\,\vec{B} + 0.6\,\vec{C} = (-1, -1.7, 1.5)$

```
\pgfmathparse{TD("0.2*(A)
-0.3*(B)+0.6*(C)")}%
$0.2\,\vec A-0.3\,\vec B+0.6\,\vec C
=(\pgfmathprintvector\pgfmathresult)$
```

The alert reader may wonder why this works, i.e. how would Ti*k*Z "know" what the coordinates A, B and C are. It works because the coordinates in Ti*k*Z are global, so they get remembered from the above example.

$(1,0,0)^T \times (0,1,0)^T = (0,0,1)^T$

```
\pgfmathparse{TD("(1,0,0)x(0,1,0)")}%
$(1,0,0)^T\times(0,1,0)^T=
(\pgfmathprintvector\pgfmathresult)^T$
```

$\vec{A} \cdot \vec{B} = 5$

```
\pgfmathparse{TD("(A)o(B)")}%
$\vec A\cdot \vec B=
\pgfmathprintnumber\pgfmathresult$
```