The three-dimensional mkpath package

Here are the routines defined in the file `mkpath3d.inc`. Roughly speaking, the entire package is concerned with building paths in $3D$.

ROUTINE:	**mkpath3d**
ARGUMENTS:	$\ell\ a\ f\ t_0\ t_f\ N$
RETURNS:	Adds a sequence of Bezier curves to the current path

Here ℓ is the **location matrix**, of the form $[M\ v]$ where M is a 3×3 matrix and v a 3-vector. This is to be interpreted as a rigid affine transformation, describing the shifted position of the path to be drawn. The procedure f is called by name (with `/`). It has two arguments, an array of parameters and a value of t. Output from f is a three-dimensional parametrization in the format

$$[[x(t)\ y(t)\ z(t)][x'(t)\ y'(t)\ z(t)]]$$

ROUTINE:	**mkpolypath3d**
ARGUMENTS:	$\ell\ a\ f\ t_0\ t_f\ N$
RETURNS:	Adds a sequence of line segments to the current path

Is to `mkpath3d` as `mkpolypath` is to `mkpath`.

ROUTINE:	**mkpolygon3d**
ARGUMENTS:	A location matrix ℓ, an array of $3D$ points p
RETURNS:	Adds a sequence of line segments to the current path

The array looks like

$$[[x_0\ y_0\ z_0][x_1\ y_1\ z_1]\ldots[x_n\ y_n\ z_n]]$$

This can be used to draw a cube or faces of some other polyhedron, for example.

ROUTINE:	**use-perspective**
ARGUMENTS:	None
RETURNS:	Sets up use of perspective for rendering $3D$ to $2D$

ROUTINE:	**use-projection**
ARGUMENTS:	None
RETURNS:	Sets up use of projection

Exactly one of these must be used before any three-dimensional drawing is done. If you don't do this, you will get an error message about `convert` being undefined.