## NAME

**libpathplan** − finds and smooths shortest paths

## SYNOPSIS

```
#include <graphviz/pathplan.h>

typedef struct Pxy_t {
    double x, y;
} Pxy_t;

typedef struct Pxy_t Ppoint_t;
typedef struct Pxy_t Pvector_t;

typedef struct Ppoly_t {
    Ppoint_t *ps;
    int pn;
} Ppoly_t;

typedef Ppoly_t Ppolyline_t;

typedef struct Pedge_t {
    Ppoint_t a, b;
} Pedge_t;

typedef struct vconfig_s vconfig_t;

#define POLYID_NONE
#define POLYID_UNKNOWN
```

## FUNCTIONS

```
int Pshortestpath(Ppoly_t *boundary, Ppoint_t endpoints[2], Ppolyline_t *output_route);
```

Finds a shortest path between two points in a simple polygon. You pass endpoints interior to the polygon boundary. A shortest path connecting the points that remains in the polygon is returned in output_route. If either endpoint does not lie in the polygon, an error code is returned. (what code!!)

```
vconfig_t *Pobsopen(Ppoly_t **obstacles, int n_obstacles);
int Pobspath(vconfig_t *config, Ppoint_t p0, int poly0, Ppoint_t p1, int poly1, Ppolyline_t *output_route);
void Pobsclose(vconfig_t *config);
```

These functions find a shortest path between two points in a simple polygon that possibly contains polygonal obstacles (holes). Pobsopen creates a configuration (an opaque struct of type vconfig_t) on a set of obstacles. Pobspath finds a shortest path between the endpoints that remains outside the obstacles. If the endpoints are known to lie inside obstacles, poly0 or poly1 should be set to the index in the obstacles array. If an endpoint is definitely not inside an obstacle, then POLYID_NONE should be passed. If the caller has not checked, then POLYID_UNKNOWN should be passed, and the path library will perform the test.

(!! there is no boundary polygon in this model?!!!)

```
int Proutespline (Pedge_t *barriers, int n_barriers, Ppolyline_t input_route, Pvector_t endpoint_slopes[2],
                Ppolyline_t *output_route);
```

This function fits a Bezier curve to a polyline path. The curve must avoid a set of barrier segments. The polyline is usually the output_route of one of the shortest path finders, but it can be any simple path that doesn't cross any obstacles. The input also includes endpoint slopes and 0,0 means unconstrained slope.

Finally, this utility function converts an input list of polygons into an output list of barrier segments:

```
int Ppolybarriers(Ppoly_t **polys, int n_polys, Pedge_t **barriers, int *n_barriers);
```

## AUTHORS

David Dobkin (dpd@cs.princeton.edu), Eleftherios Koutsofios (ek@research.att.com), Emden Gansner (erg@research.att.com).