class Utility

It is just the collection of common helper functions. All functions are static, so no instance of Utility is required when using them.

There are some categories of these functions:

1. position test/comparison
2. plane-polygon splitting
3. floating-point comparison
4. list permutation

- Position test/comparison

Test/compare where an object is relative to a plane, like, if a point is in front of a plane.

1) coplanar_det

```
float coplanar_det(const Point &a, const Point &b, const Point &c,
                   const Point &p);
float coplanar_det(const Polygon &f, const Point &p);
```

'coplanar_det' means coplanar test determinant and this function computes the value of it. A coplanar test decide if 4 points are on the same plane.

In the first form, points a, b, c defines a polygon that represents a plane. Point p is the point being tested. The second form just uses the polygon to replace the points a, b, c in the first one.

As far as I know, the determinant is the one for a 3D version of orientation test. The one I use looks like this:

$$\begin{vmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_p & y_p & z_p & 1 \end{vmatrix}$$

where x, y, z are the coordinates of the 4 points. Its value is 0 if they are coplanar. Positive value means point p is in front of the plane defined by a, b, c while negative value gives opposite result.

2) before

```
bool before(const Polygon &a, const Polygon &b);
bool before(const Point &a, const Polygon &plane);
```

Test if a polygon/point is in front of a plane.

In the first form, polygon **a** represents **the polygon being tested**, while **b** represents **the plane**. The second form is used for points.

It returns true if the test subject is in front of the plane, false otherwise.

3) behind

```
bool behind(const Polygon &a, const Polygon &b);
bool behind(const Poing &a, const Polygon &plane);
```

Test if a polygon/point is behind a plane.

In the first form, polygon **a** represents **the polygon being tested**, while **b** represents **the plane**. The second form is used for points.

It returns true if the test subject is behind the plane, false otherwise.

4) coplanar

```
bool coplanar(const Polygon &a, const Polygon &b);
```

Test if a polygon is contained in a plane.

Polygon **a** represents **the polygon being tested**, while **b** represents **the plane**.

It returns true if the polygon is on the plane, false otherwise.

- Plane-Polygon splitting

Includes functions related to splitting polygons by a plane.

Note: functions in this category are mainly used privately by the implementation of the BSPTree class.

1) intersection

```
Point intersection(const Polygon &f,
       const Point &a, const Point &b);
```

Compute the intersection between a line and a plane. Points a and b define the line, and polygon f represents the plane.

Returns the intersection as a Point object.

Note: I never test the results when the line and the plane do not intersect.

2) split_polygon

```
void split_polygon(const Polygon &polygon, const Polygon &plane,
       vector<Polygon> &front_list, vector<Polygon> &back_list);
```

Split a polygon into two sets according to its position relative to the plane.

polygon - the polygon to split.
plane - the plane.
front_list - set of fragments that are in front of the plane
back_list - set of fragments that are behind the plane.

After execution the fragments will be stored in the two vectors.

- Floating-point comparison

Just as the name suggests...

```
bool is_zero(float v);
bool is_nonpositive(float v);
bool is_nonnegative(float v);
```

Note: as their names suggest, is_nonpositive()(and is_nonnegative()) returns true when v is less than or equal to (larger than or equal to) 0. Kinda bad design here:P

- List permutation

```
   void permute_list(vector<Polygon> &list);
```

Used solely by class BSPTree to generate a random permutation of input polygons.