Notes on polygon meshes

1 Basic definitions

Definition 1 *A* polygon mesh *(or* polymesh*) is a triple* (V, E, F)*, where*

V			*a set of vertices (points in space)*
E	$\subset$	$(V \times V)$	*a set of edges (line segments)*
F	$\subset$	E^*	*a set of faces (convex polygons)*

with the following properties:

1. *for any* $v \in V$*, there exists* $(v_1, v_2) \in E$ *such that* $v = v_1$ *or* $v = v_2$*.*
2. *for and* $e \in E$*, there exists a face* $f \in F$ *such that* e *is in* f*.*
3. *if two faces intersect in space, then the vertex or edge of intersection is in the mesh.*

If all of the faces of a polygon mesh are triangles, then we call it a *triangle mesh* (*trimesh*). Polygons can be *tessellated* to form triangle meshes.

Definition 2 *We classify edges in a mesh based on the number of faces they are part of:*

- *A* boundry edge *is part of exactly one face.*
- *An* interior edge *is part of two or more faces.*
- *A* manifold edge *is part of exactly two faces.*
- *A* junction edge *is part of three or more faces.*

Junction edges are to be avoided; they can cause cracks when rendering the mesh.

Definition 3 *A polymesh is* connected *if the undirected graph* $G = (V_F, E_E)$*, called the* dual graph*, is connected, where*

- V_F *is a set of graph vertices corresponding to the faces of the mesh and*
- E_E *is a set of graph edges connecting adjacent faces.*

Definition 4 *A* polyhedron *is a polymesh that is*

1. connected and

2. each edge is manifold.

Definition 5 *A* polytope *is a polyhedron that encloses a convex resion R of $\Re^3$ (*i.e.*, any two points in R are connected by a line segment that is wholly contained in R).*

Definition 6 *A connected mesh is* manifold *if every edge in the mesh is either a boundry edge or a manifold edge.*

For most computer graphic applications, we use manifold meshes.

Definition 7 *A manifold mesh is* closed *if every edge is manifold and it is non-intersecting.*

2 Orientation

The *orientation* of a face determines which side is the front and which side is the back. The orientation can either be Counter Clockwise (CCW, which is the OpenGL default) or Clockwise (CW).

Definition 8 *Two faces, f_1 and f_2, that share a common edge e are* consistently *oriented if the head of e in f_1 is the tail of e in f_2 (and vice versa).*

Definition 9 *A manifold mesh is* orientable *if the vertex orderings of its faces can be chosen so that adjacent faces have consistent orderings (*i.e.*, all faces are either CW or CCW).*

3 Data structures

The data structures used to represent meshes vary by application. One popular representation is the *winged-edge* data structure. In this representation, the edge is the central part of the representation. In C, we might use the following pointer-based representation:

```
struct edge {
    struct vertex *src;    /* edge source */
    struct vertex *dst;    /* edge destination */
    struct face   *left;   /* face on left-hand-side of edge */
    struct face   *right;  /* face on right-hand-side of edge */
    struct edge   *lPred;  /* left-most predecessor edge */
    struct edge   *rPred;  /* right-most predecessor edge */
    struct edge   *lSucc;  /* left-most successor edge */
    struct edge   *rSucc;  /* right-most successor edge */
};
struct vert {
    struct edge *e;
    ...
};
struct face {
    struct edge *e;
    ...
};
```

In a graphical application, we will store other information with vertices and faces (colors, normals, texture coordinates, ...), hence the "`...`" in the code. For models where the mesh is static, we can use a table-based representation that is more compact (assuming that we can use the `char` or `short` type as table indices).