# An Implementation of Halfedge Data Structure in Catmull-Clark Subdivision for 2-Manifold Single-sided Surface

Yu Wang

August 2015

# 1 Introduction

Catmull-Clark subdivision is widely applied to construct a smooth surface from an initial mesh of polygons. It is independent of the topology of initial mesh.

# 2 Halfedge Data Structure

An object in the 3D Euclid space can be modeled as several meshes of polygons. For a single mesh, it comprises three types of geometry elements: vertex, edge, and face. Adjacency data structure is need to store the topological information (adjacency and connectivity) between these elements.
Several adjacency structures have been fully developed, including simple data structure, winged edge data structure (Baumgart, 1975), halfedge data structure (Eastman, 1982), QuadEdge Data structure (Guibas and Stolfi), and FaceEdge Data Structure (Dobkin and Laszlo, 1987).
Among all these data structures, the author chooses halfedge data structure in this project to realize Catmull-Clark subdivision, because 1) the storage size is independent of the mesh topology, and 2) it has a simple implementation. The author also extends its definition to add the ability in dealing with single-sided surfaces (or non-orientable object).

## 2.1 Vertex, Halfedge, and Face

The definitions and assumptions of vertex, halfedge and face follow the assumption of 2-manifold, as shown in Table 1. Element IDs are unique. When two elements fall into the same group of element, they can not have the same ID. (Mobius sibling halfedges are exceptions as we discuss later). A quadrilateral face made with four halfedges and four vertices is shown in Figure 1.

Every element stores two types of information: self-information and adjacency information. As shown in Table 2. The adjacency information include adjacency in a face and adjacency between faces. The adjacency between faces include sibling links and boundary links. (And we will discussion more in the mesh section.)

| | Definition | Assumption |
|---|---|---|
| Vertex | A 3-dimensional point. | No overlapping vertices exits in a mesh. But overlapping vertices can exist in different meshes. |
| Halfedge | An edge that starts from one vertex and ends at another vertex. | A halfedge connects exactly two non-overlapping vertices and it has a direction. Less than two halfedges start from the same vertex and end at the same vertex in a single mesh. |
| Face | A polygon that contains a loop of vertices and halfedges. | A face has at least three non-overlapping vertices so it makes a polygon. The face has to be constructed with a complete loop of halfedges with on openings. |

Table 1: Definitions and assumptions of vertex, halfedge, and face

## 2.2 Mesh

We define a mesh as a collection of basic elements (i.e. vertex, halfedge, and face). Hashtable is implemented to represent the collection in this project, because of its constant serach time for element. We construct three hashtables for all vertices, halfedges, and faces in the mesh respectively. The keys for these hashtable are element IDs and the contents are the element pointers.

The ID of a halfedge is related with the IDs of its start vertex and end vertex. In this project, we define the ID of a halfedge as start vertex ID * maximum number of vertices in a mesh + end vertex ID. This definition will guarantee a unique ID for every halfedge when they have a different start or different end vertex from other halfedges.

A mesh also includes the adjacency and connectivity of elements in it. They can be classifed into three types: 1) halfedge flow in an individual face, 2) face connections, and 3) boundary connections.

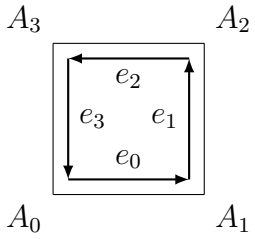


Figure 1: A quadrilateral face made with four halfedges

| Element | Self-Information | Adjacency Information |
|---|---|---|
| Vertex | 1. vertex ID<br>2. vertex position<br>3. vertex normal | 1. one outgoing halfedge<br>2. on mobius connection? |
| Halfedge | 1. edge ID | 1. start and end vertex<br>2. link to parent face<br>3. predecessor and successor in the parent face<br>4. sibling links to adjacent face<br>5. boundary links to adjacent face |
| Face | 1. face ID<br>2. face normal | 1. one side halfedge |

Table 2: Definitions and assumptions of vertex, halfedge, and face

### 2.2.1 Face Connections and Sibling Links

There are two types of face connections, the normal connection and the mobius connection, as shown in Figure 2. In a typical halfedge data structure, with the assumption of double-sided surface, a pair of halfedges between two faces are defined with opposite direction. We extends this idea to represent single-sided surface by adding another type of connection, named as mobius connection. In a mobius connection, a pair of halfedges are in same direction. In the example of Figure 2, on the top, $e_1$ and $e_1'$ are siblings to each other. On the bottom, $e_1$ and $e_1'$ are mobius siblings to each other.

One thing to point out here is that mobius sibling halfedges have the same element ID. Because they have the start vertex and end vertex. However, we could check for the mobius sibling pointers in order to find all halfedges in

the edge traveral of a mesh.

With this extension mobius connection, there can be three different situations for a halfedge in a mesh: 1) it has a sibling pointer and on a normal connection, 2) it has a mobius pointer and on a mobius connection, and 3) it does not have a sibling pointer or mobius poniter, and it lies on the boundary of the surface.

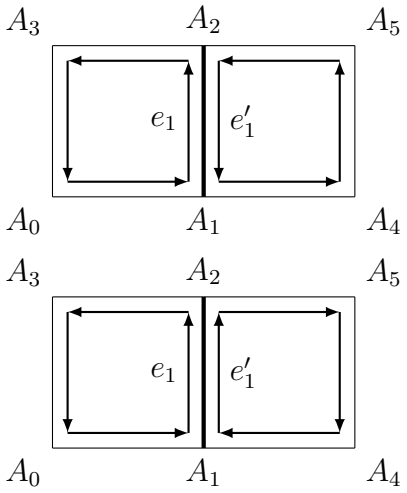


Figure 2: Normal connection (up) and mobius connection (bottom) between two faces

## 2.3 Boundary Links

For halfedges on the boundary of a mesh, we connect them with boundary links. In a surface with no mobius connetion, the boundary halfedges follows a continous flow. We can traverse the boundary by starting and ending at one vertex following the natural flow on the boundary. We create previous and next boundary pointers to these halfedges. However, when mobius connection occurs in a mesh, some adjacent boundary halfedges start or end at same vertex. We build mobius boundary pointers to these halfedges. Figure X shows an example of boundary halfedges with and without mobius connections.

¡Add normal boundary links figure and mobius boundary link figures here!¿

## 2.4 Build Mesh from Elements

In order to construct a mesh from basic elements, we need four steps.

Step1: create individual vertices. We create instances of vertices, with their position and ID defined. The position of a vetex can be the same but the ID should always be unique. This step taks O(V) time, where V is the number of vertices in the mesh.

Step2: construct indivdual face. To build an indivdual face, we create consecutive instances of halfedges, with their start and end vertices given. We also add the

Step3: build sibling links. if direct connection exist between a pair of faces, and

Step4: build boundary links. for boundary halfedges the whole mesh.

In step 1, It includes create n

## 2.5 Traversals

Two basic traversals are necessary to implement Catmull-Clark subdivision: 1) traversal around a face, and 2) traversal around a vertex.

### 2.5.1 Traversal Around Face

The traversal around a face lead to all the edges and vertices belong to this face. It starts from one side halfedge, follows the halfedge flow, and ends at the halfedge that the face traverse starts. Traversal of all faces in a mesh takes O(F) time, where F is the total number of faces in the mesh.

### 2.5.2 Traversal Around Vertex

The traversal around a vertex lead to all the edges and faces that contains this vertex. The traversal of a vertex need to consider two issues: 1) is this vertex on a boundary, and 2) is this vertex on a mobius connection. This makes four different types of vertex traversals. See Figure 3 - 6 for examples.

In the vertex traversal without boundary and mobius issue, we start from one outgoing halfedge of this vertex. We continue to go to the next outgoing halfedge by going to the successor of its sibling until we hit the first outgoing halfedge. In the example of Figure 3, if start and end at halfedge $e_1$, the sequence of traversal is: $e_1$ (sibling link to) $e_1'$ (successor link to) $e_2$ (sibling

link to) $e_2'$ (successor link to) $e_3$ (sibling link to) $e_3'$ (successor link to) $e_4$ (sibling link to ) $e_4'$ (successor link to) $e_1$.

In order to address the issue of a vertex on boundary, instead of using sibling links, we use boundary links. In the example of Figure 5, we have a boundary $e_4'$ to $e_2$. If start and end at halfedge $e_1$, the sequence of traversal is: $e_1$ (sibling link to) $e_1'$ (successor link to) $e_2$ (boundary link to) $e_4'$ (successor link to) $e_1$.

To address the issue of vertex on a mobius connection, instead of using normal links, we use mobius links. At the same time, we switch between the successor and predecessor to the sibling every time we hit a mobius connection. In the example of Figure 3, $e_1$ to $e_1'$ and $e_3$ to $e_3'$ have mobius siblings rather than normal sibling. If start and end at halfedge $e_1$, the sequence of traversal is: $e_1$ (mobius sibling link to) $e_1'$ (predecessor link to) $e_2$ (sibling link to) $e_2'$ (predecessor link to) $e_3$ (mobius sibling link to) $e_3'$ (successor link to) $e_4$ (sibling link to ) $e_4'$ (successor link to) $e_1$.

If boundary and mobius connection both occur, we use a combination of the two methods above. In the example of Figure 3, $e_1$ to $e_1'$ and $e_3$ to $e_3'$ have mobius siblings rather than normal sibling and we have a mobius boundary connection between $e_4'$ and $e_2$. If start end end at halfedge $e_1$, the sequence of traversal is: $e_1$ (mobius sibling link to) $e_1'$ (predecessor link to) $e_2$ (mobius boundary link to ) $e_4'$ (successor link to) $e_1$.

As a summary of the four different situations above, in a vertex traversal, we 1) start the traversal from with one outgoing halfedge of the vertex, 2) go to sibling or boundary link halfedge, 3) go to the next or previous halfedge that contains the vertex as one end, and 4) repeat 2) and 3) until we reach to the starting outgoing halfedge. This vertex traversal runs in O(E) time, where E is the total number of halfedges in the mesh.

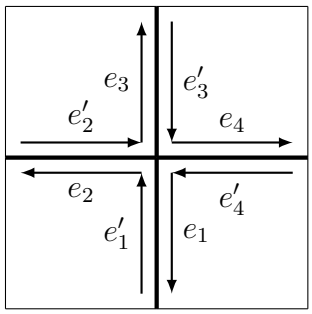


Figure 3: Vertex traversal without boundary and without moibus connection

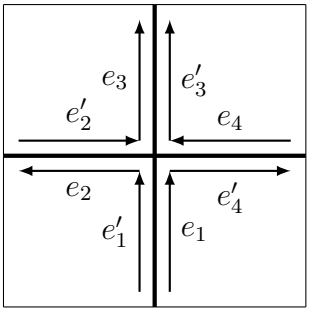


Figure 4: Vertex traversal without boundary and with moibus connection

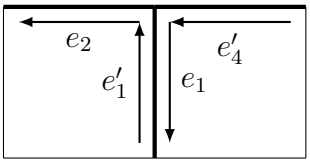


Figure 5: Vertex traversal withboundary and without moibus connection

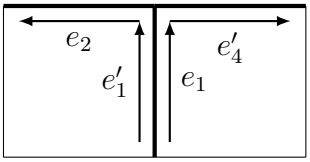


Figure 6: Vertex traversal with boundary and with moibus connection

## 2.6 Mesh Construction

### 2.6.1 Build from Elements

In order to build a mesh from elements, we need to do three steps: 1) contruct every indivdual face, 2) build the sibling links if normal connection or mobius connection exist between a pair of faces, and 3) build the boundary links for the whole mesh.