Lecture 3 : SUBDIVISION

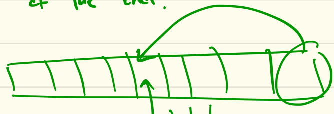
```
readFile
{
    if ( t == 'v')
    {
        . read in x,y,z ;
        . myVertex * v = new myVertex();
        . v->point = new myPoint3D(x,y,z);
        . vertices.push_back (v) ;
    }

if ( v == 'f')
{
    vector < int > face_indice s;
    vector < myHalfedge *> face_edges;
    myFace * f = new myFace () ;
    while (myline >> u)
    {
        face_indices.push_back( face index -1 );
        face_edges .push_back(new myHalfedge());
    }
    for ( int i=0 ; i < face_indices.size(); i++)
    {
        int ipo = (i+1) % face_indices.size();
        int imo = (i-1 + face_indices.size()) % face_i.size;
        face_edges [i] . next = face_edges[ipo]
                        prev =    "  "  [imo]
                        face =  f ;
                        Source =   vertice[face_indices[i]] ;
        vertices[face_indices[i]] -> originof = face_edges[i];
```

myMesh

- vector<myVertex *> vertices
- vector<myFace *> faces
- $\boxed{\text{vector}}$<myHalfedges *>halfedges

array wrapper

1. easy memory mangemt
2. add/del at the end.



myFace

- myHalfedge * adj;

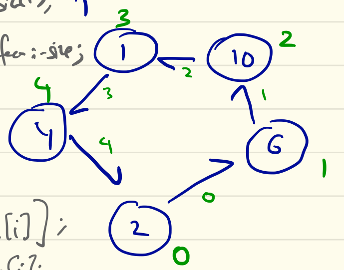
myVertex

- myPoint3D * point;
- myHelfedge * originof;

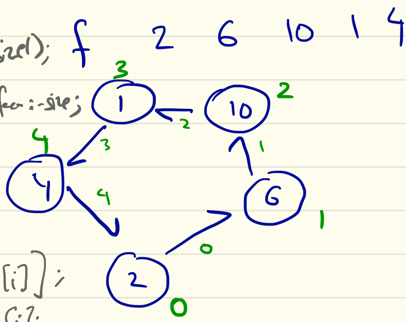
myHalfedge

- myVertex * source;
- myHalfedge * twin;
  next;
  prev;
- myFace * adj_face;

f 2 6 10 1 4



```
for ( int i=0; i< face_indices.size(); i++)
{
    int ipo = (i+1) % face_indices.size();
    int imo = (i-1 + face_indices.size()) % face_indices.size;
    face_edges[i].next = face_edges[ipo]
                 prev =    "  "   [imo]
                 face = f;
                 source = vertice[face_indices[i]];
    vertices[face_indices[i]] -> origin of = face_edge[i];
    halfedges.push_back( face_edge[i] );
}
    f -> adj_halfedges = face_edges[0];
    faces.push_back(f);
}
```



Question: given : myMesh *in

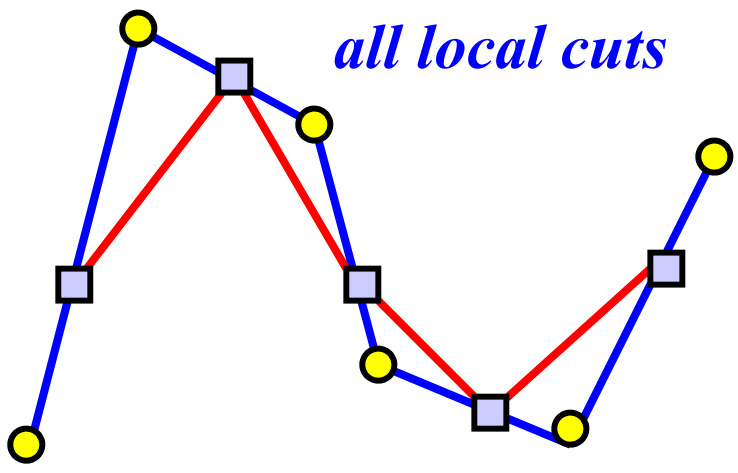
output : myMesh *out, where out is a copy of in.

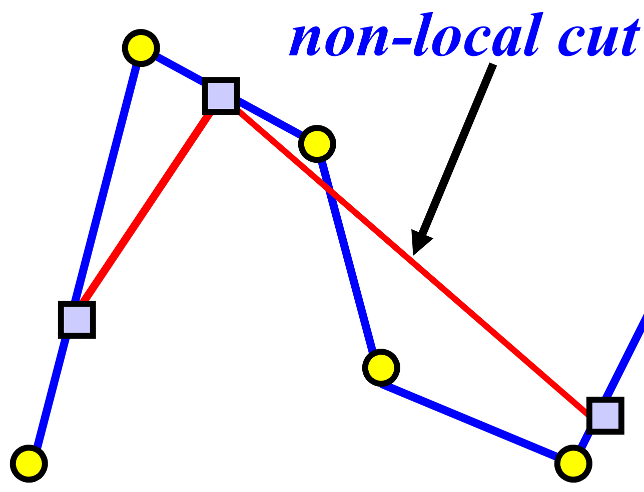
```
copyMesh ( myMesh *in )
{
    for ( int i=0 ; i< in->vertices.size(); i++ )
        vertices.push_back( in->vertices [i] );

    for ( int i=0 ; i< in->faces.size(); i++ )
        faces.push_back ( in->faces [i] );

    for ( int i=0 ; i< in->halfedges.size(); i++ )
        halfedges.push_back ( in->halfedges [i] );

}
```

Solution to deep copy : use index in each class !!

Curve Corner Cutting

- Take two points on different edges of a polygon and join them with a line segment. Then, use this line segment to replace all vertices and edges in between. This is corner cutting!
- Corner cutting can be local or non-local.
- A cut is *local* if it removes exactly one vertex and adds two new ones. Otherwise, it is *non-local*.





FYI

- Subdivision and refinement has its first significant use in Pixar's *Geri's Game*.
- Geri's Game received the Academy Award for Best Animated Short Film in 1997.

- http://www.pixar.com/shorts/gg/

Facts about Subdivision Surfaces

- **Subdivision surfaces are *limit surfaces*:**
  - **It starts with a mesh**
  - **It is then refined by repeated subdivision**
- **Since the subdivision process can be carried out infinite number of times, the intermediate meshes are *approximations* of the actual subdivision surface.**
- **Subdivision surfaces is a simple technique for describing complex surfaces of arbitrary topology with guaranteed continuity.**
- **Also supports Multiresolution.**

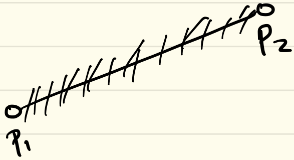
What Can You Expect from ...?

- **It is easy to model a large number of surfaces of various types.**
- **Usually, it generates smooth surfaces.**
- **It has simple and intuitive interaction with models.**
- **It can model sharp and semi-sharp features of surfaces.**
- **Its representation is simple and compact (*e.g.*, winged-edge and half-edge data structures, etc).**
- **We only discuss 2-manifolds without boundary.**

Catmull-Clark Algorithm: 1/10

- Catmull and Clark proposed another algorithm in the same year as Doo and Sabin did (1978).
- In fact, both papers appeared in the journal *Computer-Aided Design* back to back!
- Catmull-Clark's algorithm is rather complex. It computes a face point for each face, followed by an edge point for each edge, and then a vertex point for each vertex.
- Once these new points are available, a new mesh is constructed.

Convex combination.

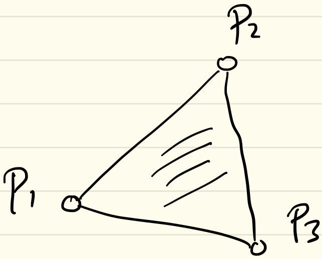


$$\alpha \cdot P_1 + \beta \cdot P_2$$

where

$$0 \le \alpha, \beta \le 1.$$

$$\alpha + \beta = 1$$

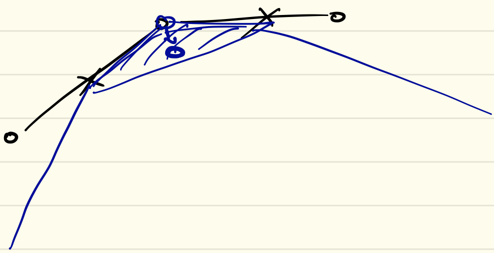


$$\alpha \cdot P_1 + \beta \cdot P_2 + \gamma \cdot P_3$$

where

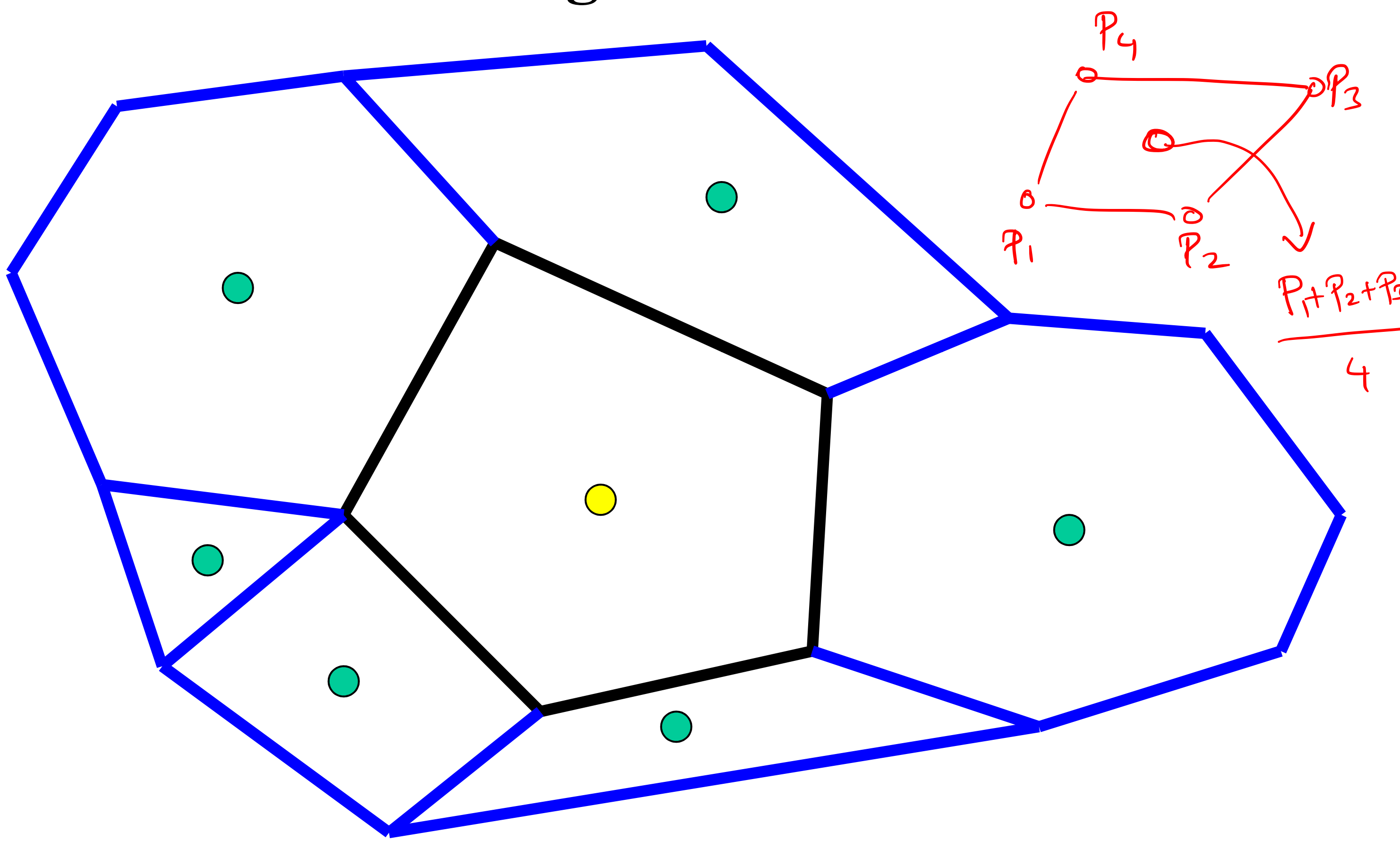
$$0 \le \alpha, \beta, \gamma \le 1$$

$$\alpha + \beta + \gamma = 1$$



Catmull-Clark Algorithm: 2/10

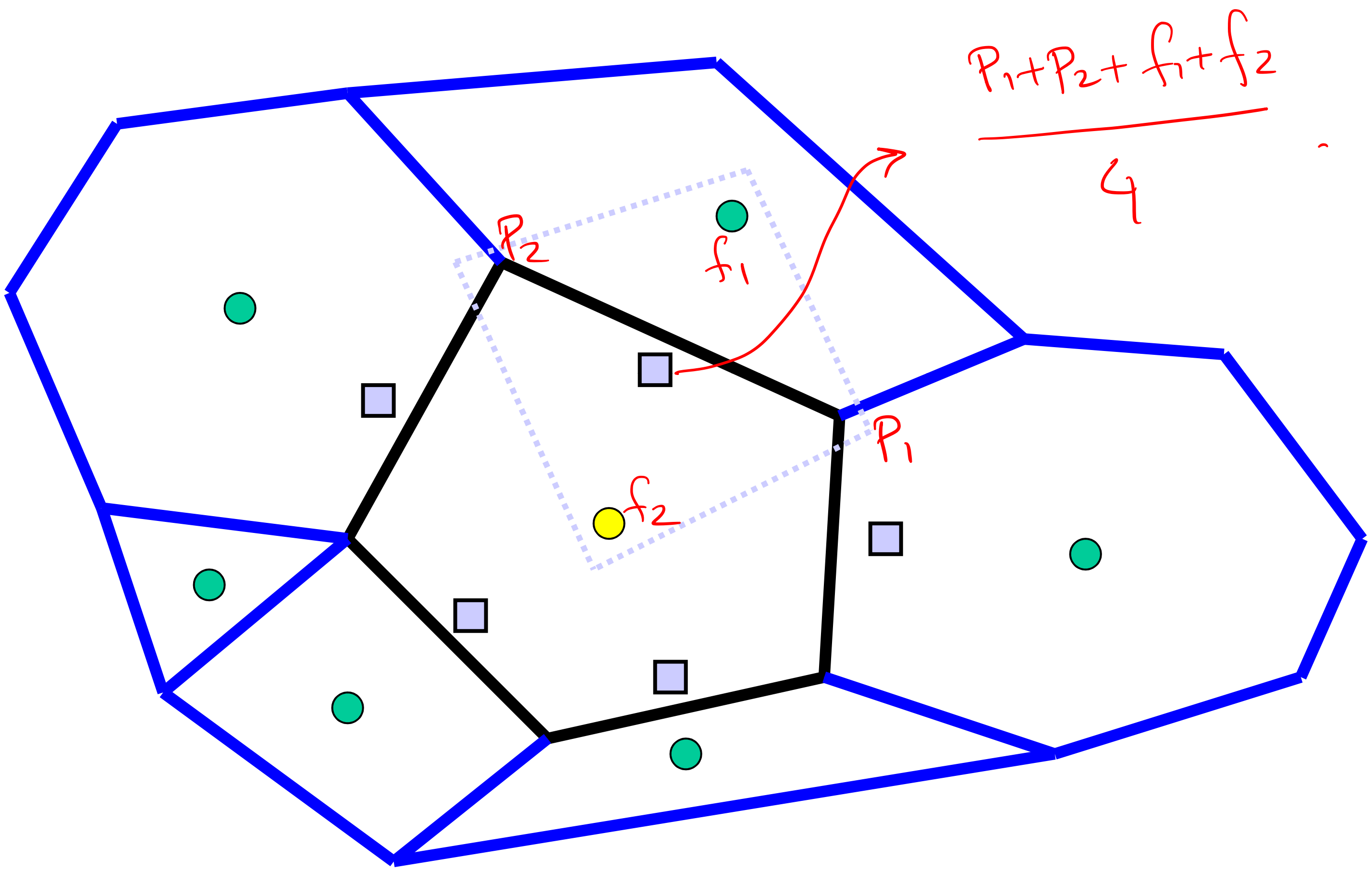
- Compute a face point for each face. This face point is the gravity center or centroid of the face, which is the average of all vertices of that face:



$$\frac{P_1+P_2+P_3+P_4}{4}$$

Catmull-Clark Algorithm: 3/10

- Compute an edge point for each edge. An edge point is the average of the two endpoints of that edge and the two face points of that edge's adjacent faces.



Catmull-Clark Algorithm: 4/10

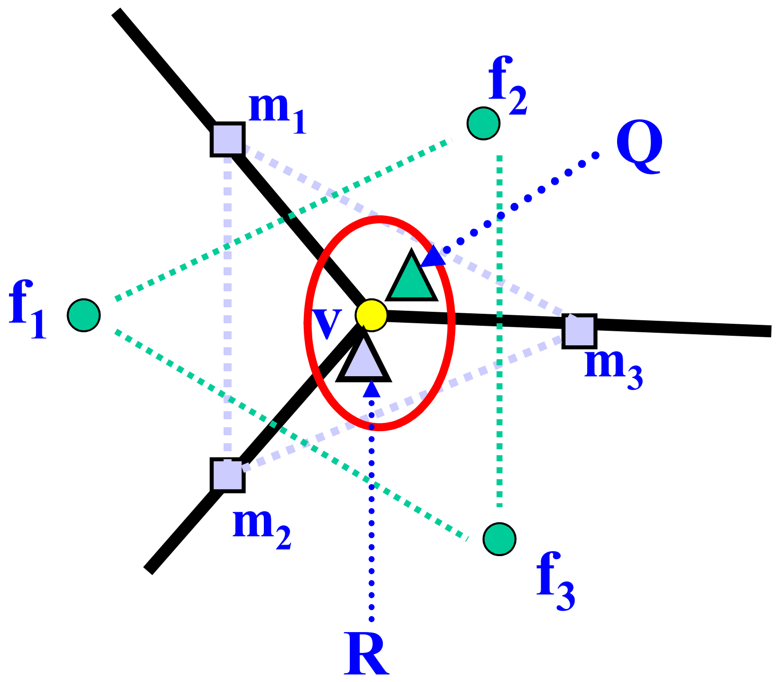
- Compute a vertex point for each vertex v as follows:

$$\mathbf{v}' = \frac{1}{n}\mathbf{Q} + \frac{2}{n}\mathbf{R} + \frac{n-3}{n}\mathbf{v}$$

$\frac{f_1+f_2+f_3}{3}$ $\frac{m_1+m_2+m_3}{3}$

weights

$\frac{1}{n}+\frac{2}{n}+\frac{n-3}{n}=1$.



Q – the average of all new face points of **v**

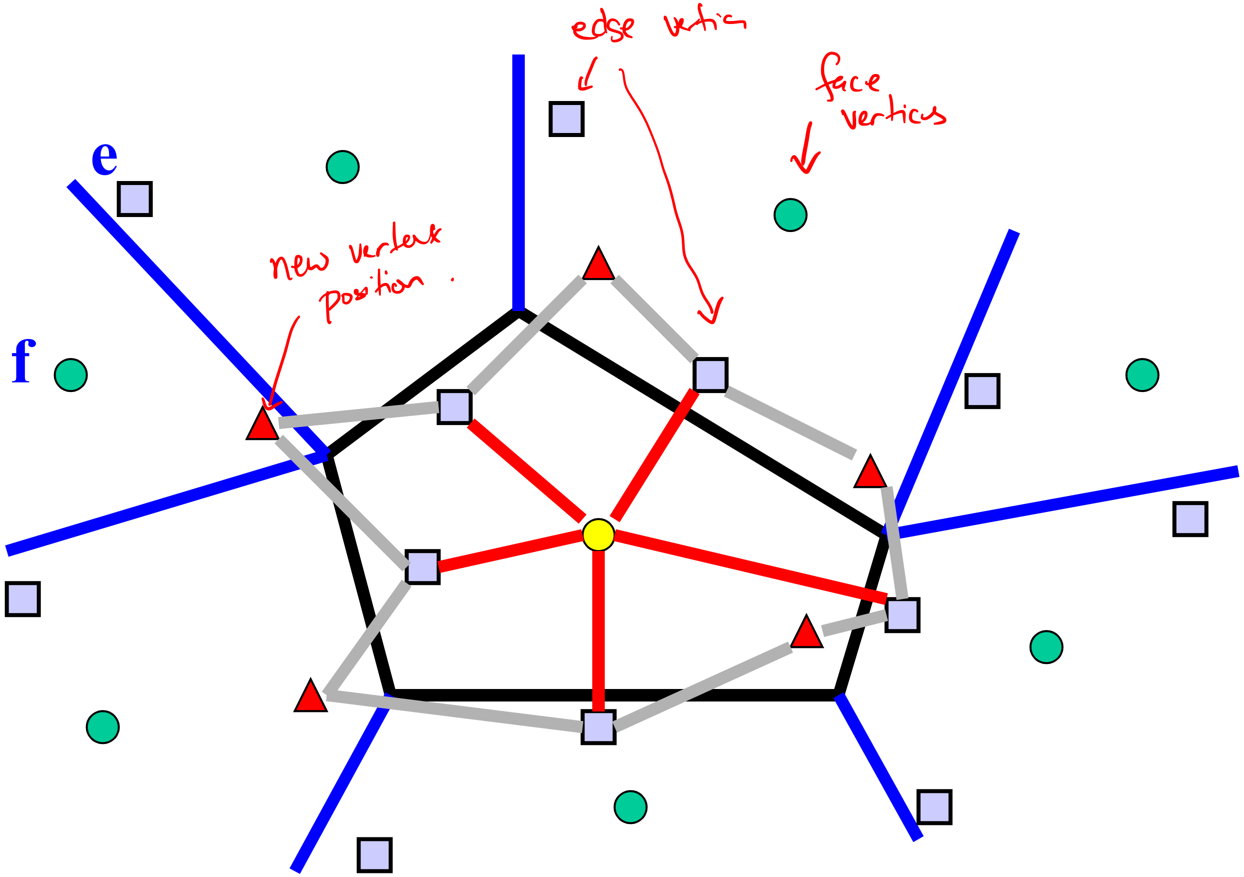
R – the average of all mid-points (*i.e.*, $\mathbf{m}_i$'s) of vertex **v**

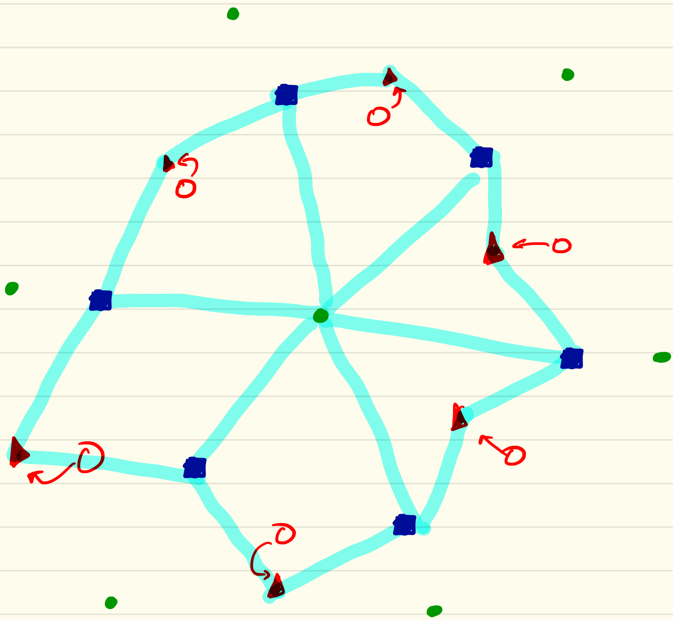
v - the original vertex

n - # of incident edges of **v**

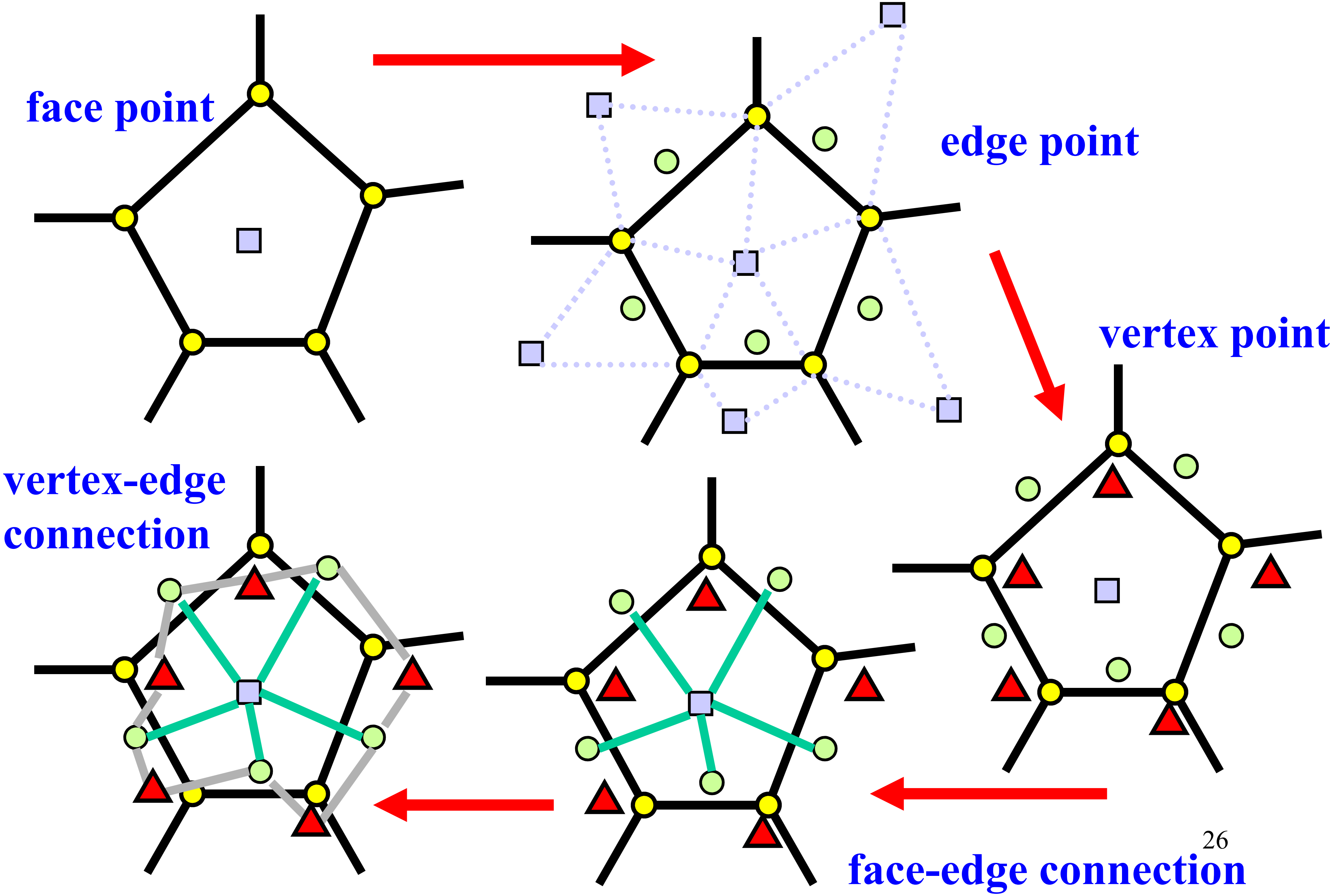
Catmull-Clark Algorithm: 5/10

- For each face, connect its face point **f** to each edge point, and connect each new vertex **v'** to the two edge points of the edges incident to **v**.




0
0
0
0
0
0

Catmull-Clark Algorithm: 6/10



Q: Given a mesh with

n vertices

m edges

f faces.

New mesh after one round of Catmull-Clark algorithm for subdivision:

$\boxed{n+m+f}$ Vertices

$\boxed{4m}$ edges

$\boxed{2m}$ faces

Catmull-Clark Algorithm: 7/10

- After the first run, all faces are four sided.
- If all faces are four-sided, each has four edge points $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ and $\mathbf{e}_4$, four vertices $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ and $\mathbf{v}_4$, and one new vertex $\mathbf{v}$. Their relation can be represented as follows:

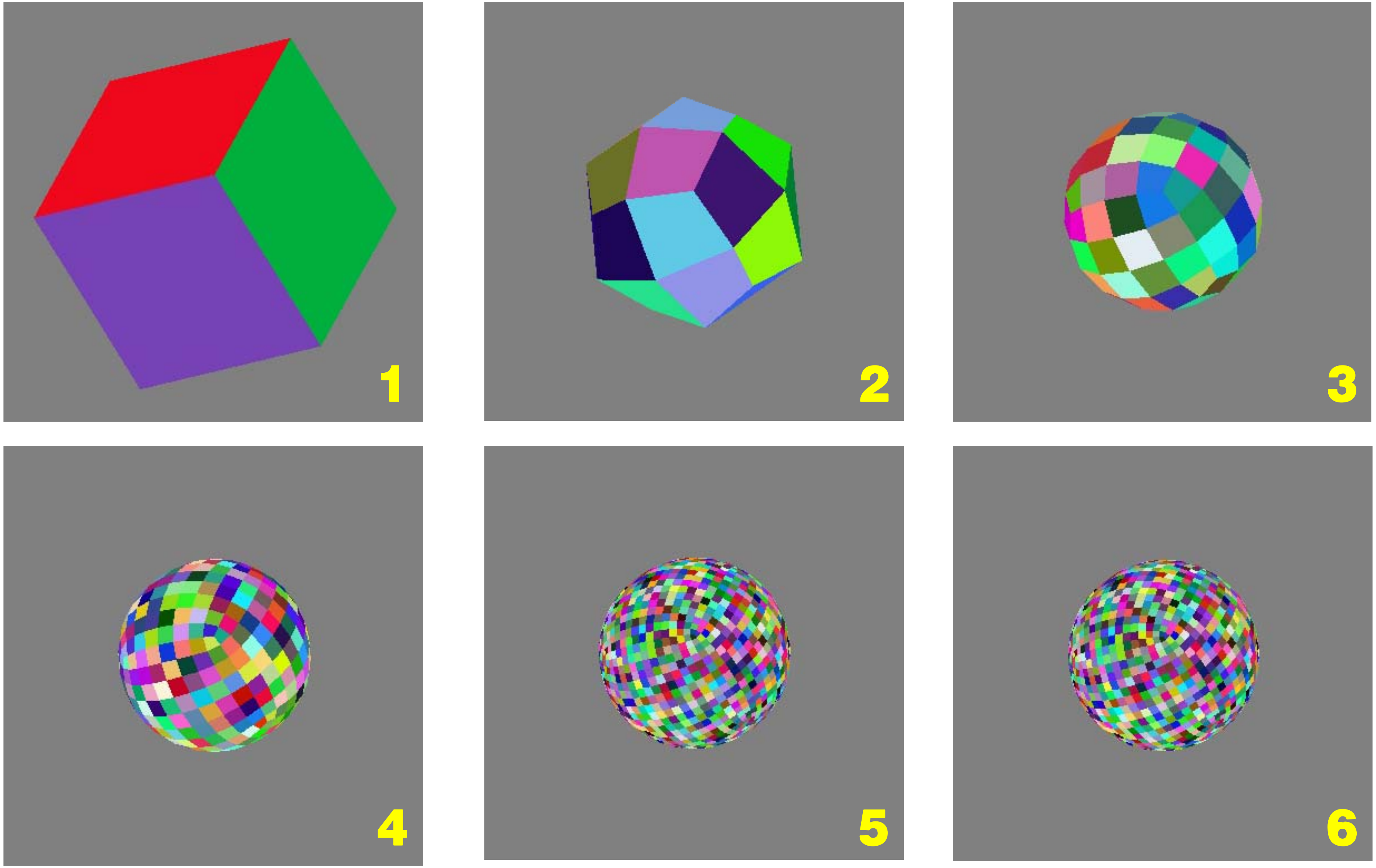
$$\begin{bmatrix} \mathbf{v}' \\ \mathbf{e}'_1 \\ \mathbf{e}'_2 \\ \mathbf{e}'_3 \\ \mathbf{e}'_4 \\ \mathbf{v}'_1 \\ \mathbf{v}'_2 \\ \mathbf{v}'_3 \\ \mathbf{v}'_4 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 9 & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 6 & 6 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 6 & 1 & 6 & 1 & 0 & 1 & 1 & 0 & 0 \\ 6 & 0 & 1 & 6 & 1 & 0 & 1 & 1 & 0 \\ 6 & 1 & 0 & 1 & 6 & 0 & 0 & 1 & 1 \\ 4 & 4 & 4 & 0 & 0 & 4 & 0 & 0 & 0 \\ 4 & 0 & 4 & 4 & 0 & 0 & 4 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 & 0 & 0 & 4 & 0 \\ 4 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v} \\ \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \mathbf{e}_4 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix}$$

- A vertex at any level converges to the following:

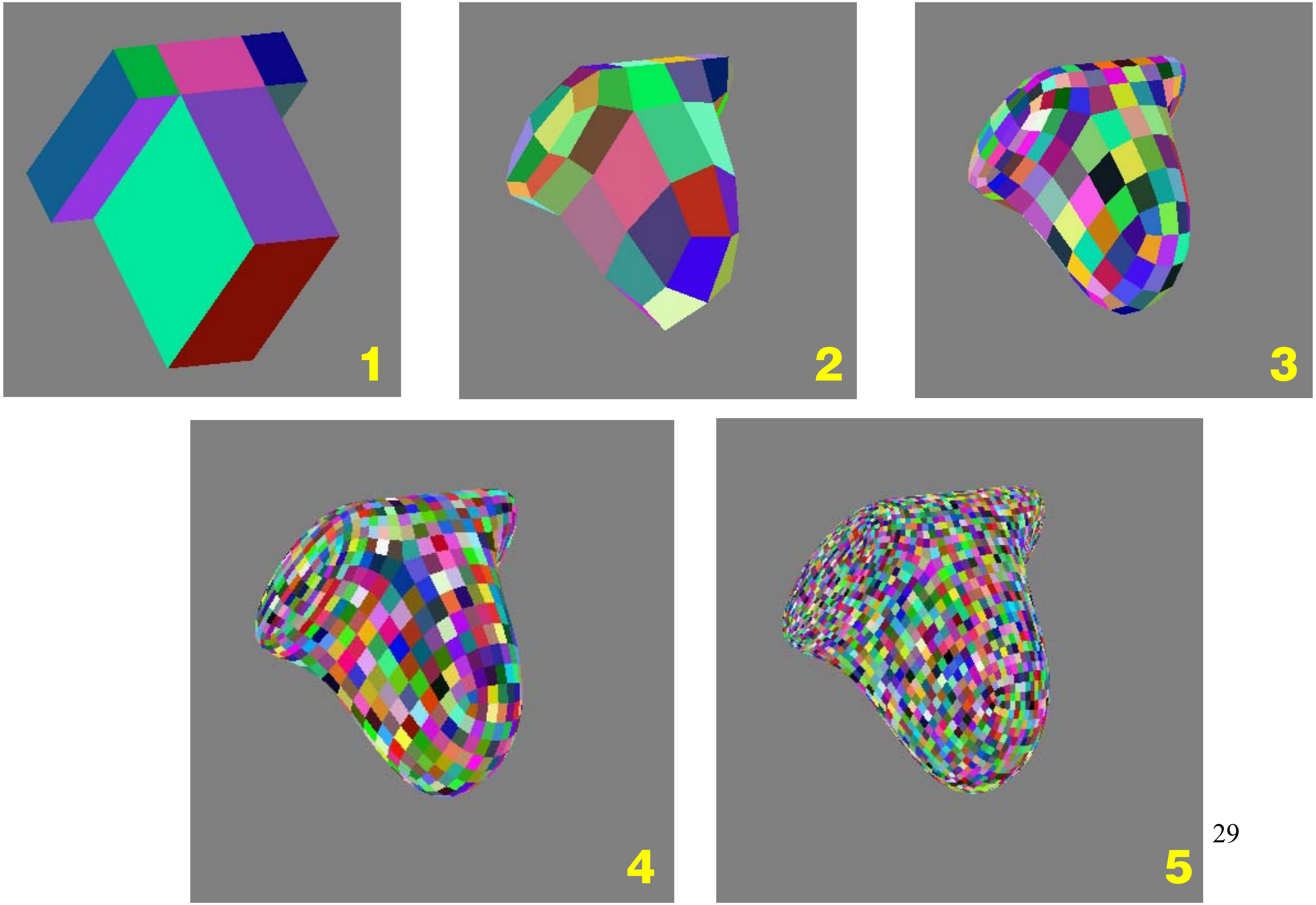
$$\mathbf{v}_\infty = \frac{n^2\mathbf{v} + 4\sum_{j=1}^{4}\mathbf{e}_j + \sum_{j=1}^{4}\mathbf{f}_j}{n(n+5)}$$

- The limit surface is a B-spline surface of degree (3,3).

Catmull-Clark Algorithm: 8/10



Catmull-Clark Algorithm: 9/10



Catmull-Clark Algorithm: 10/10