

lecture 11

meshes

- basic definitions and data structures, parameterization, acquisition
- level of detail, simplification, subdivision

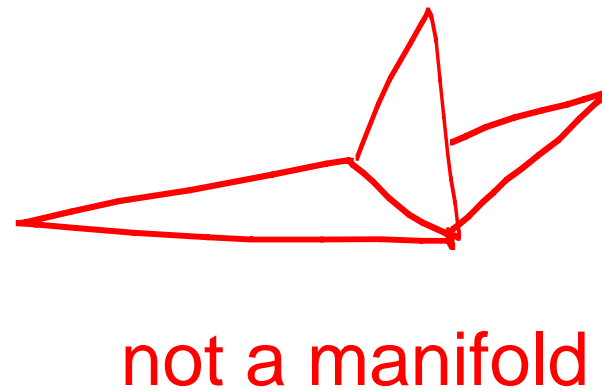
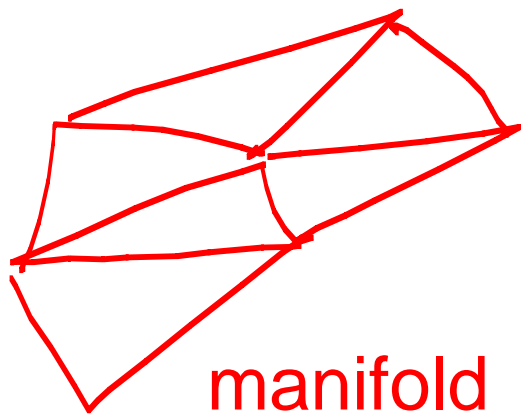
What is a mesh ?

- a mesh is an undirected graph $G = (V, E)$
 - V: vertices are in \mathbb{R}^3
 - E: edges are line segments
 - F: faces are minimal cycles (polygons)






Typically we have strong restriction on edges and faces.

e.g. each edge belongs to either one face ("boundary edge") or two faces ("regular edge")

In this case, the mesh gives us a 2D surface (technically, a "manifold").



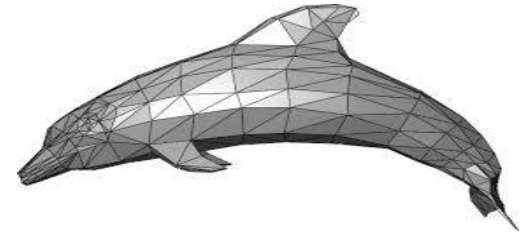
A "polyhedron" is a closed mesh (bounding a solid). e.g. Platonic Solids

| Platonic solids | Name | Faces | Edges | Vertices |
|---|--------------|-------|-------|----------|
|  | tetrahedron | 4 | 6 | 4 |
|  | octahedron | 8 | 12 | 6 |
|  | icosahedron | 20 | 30 | 12 |
|  | cube | 6 | 12 | 8 |
|  | dodecahedron | 12 | 30 | 20 |

Topology of Polyhedra

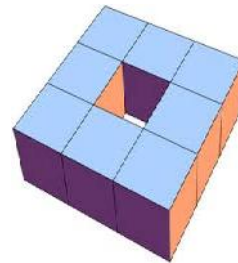
For a polyhedron with no "holes":

$$V - E + F = 2 \quad (\text{Euler}).$$



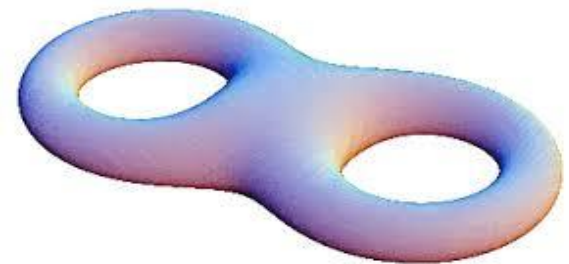
For a polyhedron with one hole:

$$V - E + F = 0$$



For a polyhedron with n holes:

$$V - E + F = 2 - 2n$$



Data Structures for Meshes

Vertex Table

| | |
|----|--------------|
| v1 | (x1, y1, z1) |
| v2 | (x2, y2, z2) |
| v3 | (x3, y3, z3) |
| v4 | (x4, y4, z4) |
| : | : |
| vn | (xn, yn, zn) |

Edge Table

| | |
|----|------------|
| e1 | (v1, v2) |
| e2 | (v1, v5) |
| e3 | (v2, v9) |
| e4 | (v3, v243) |
| : | : |
| ek | (v92, v64) |

Face Table*

| | |
|----|-----------------|
| f1 | (e1, e2, e7) |
| f2 | (e1, e3, e10) |
| f3 | (e7, e8, e5) |
| f4 | (e11, e13, e98) |
| : | : |
| fm | (e34, e21, e16) |

*Lots of flexibility here. e.g. Faces can be represented by a sequence of vertices instead of a sequence of edges.

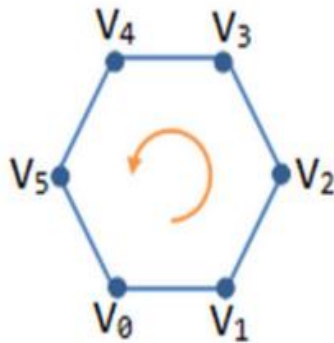
The above representation is called a **polygon soup**.

It does *not* allow you to answer queries like ...

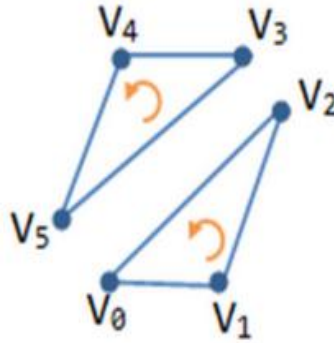
- Given a vertex, which edges (or faces) does it belong to ?
- Given an edge, which faces does it belong to ?
- Given a face, what are the adjacent faces ?

You can augment the tables to include this connectivity information. But you have to pay for it...

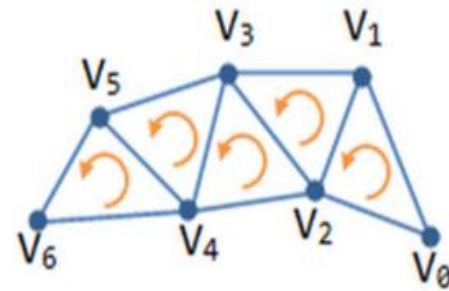
ASIDE: OpenGL 1.0



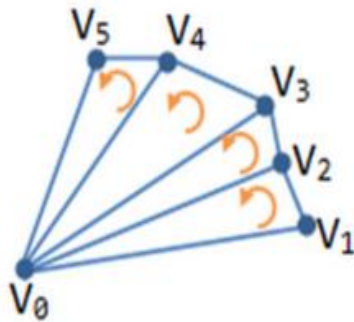
GL_POLYGON



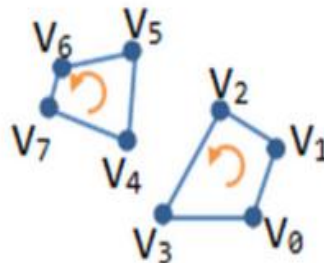
GL_TRIANGLES



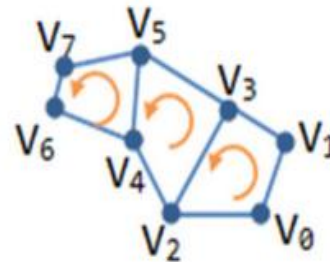
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



GL_QUADS

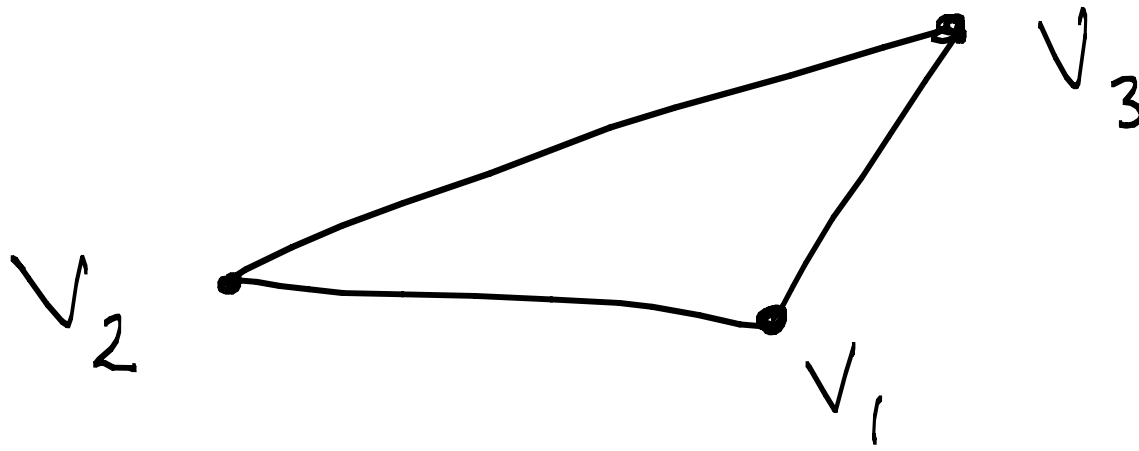


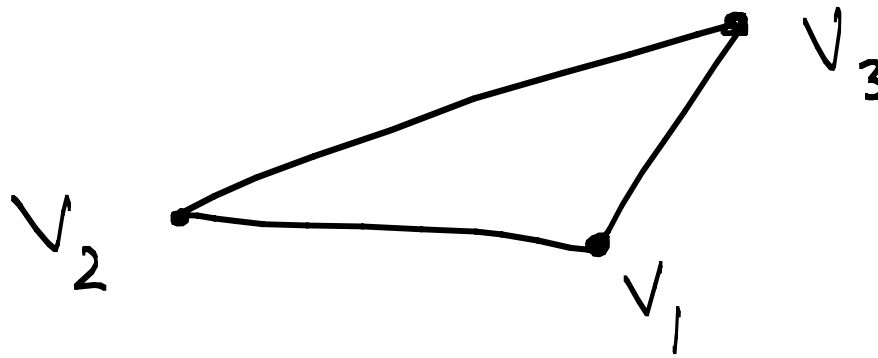
GL_QUAD_STRIP

Strips and fans provide some space efficiency. But they still have limited expressive power (for connectivity).

How to parameterize *all* the points on a mesh surface ?

A more basic question: How to parameterize points in a triangle ?





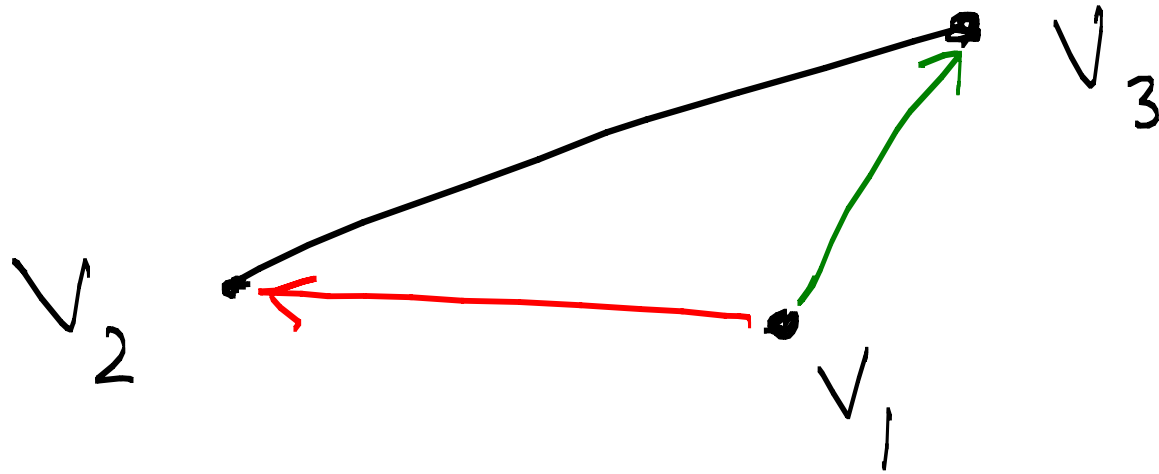
Consider

$$\{ a v_1 + b v_2 + c v_3 \}$$

This set spans all \mathbb{R}^3 .

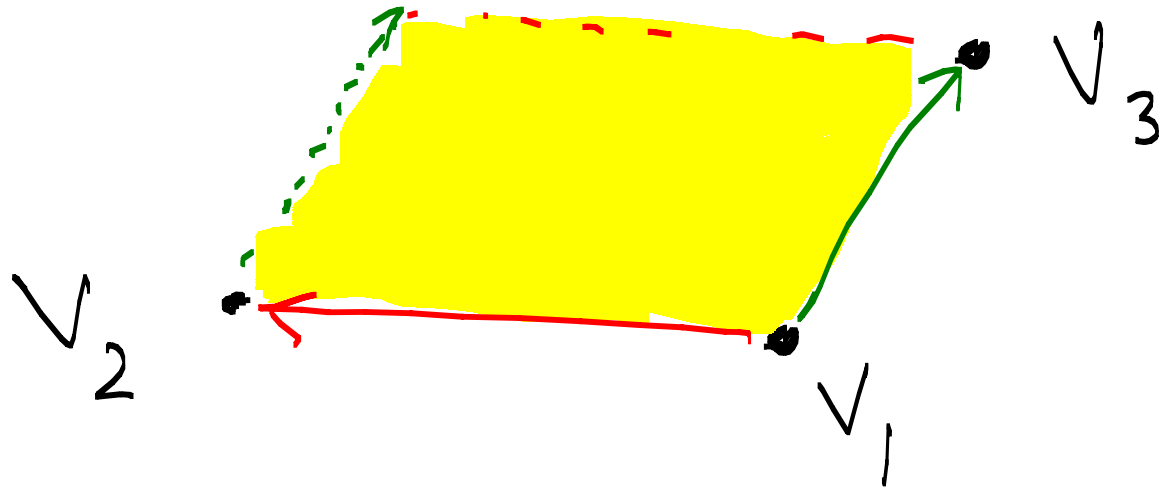
Claim: Restricting to $a + b + c = 1$

gives a plane in \mathbb{R}^3 , which contains the triangle. Why?



$$v_1 + b(v_2 - v_1) + c(v_3 - v_1)$$

This spans a (2D) plane in \mathbb{R}^3 , namely the plane containing the triangle.



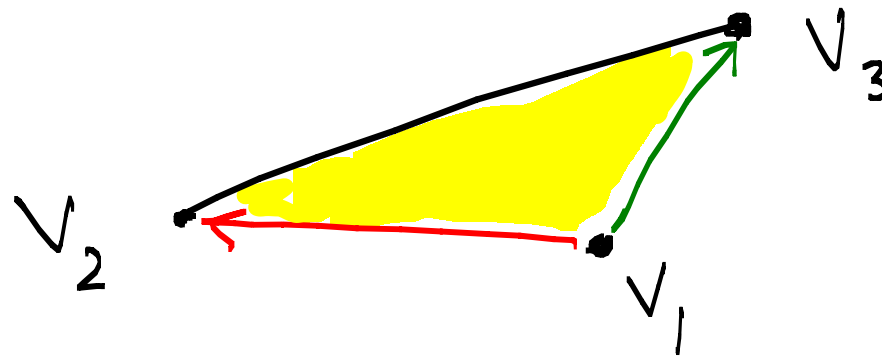
$$v_1 + b(v_2 - v_1) + c(v_3 - v_1)$$

If

$$0 \leq b \leq 1$$

$$0 \leq c \leq 1$$

then we get a parallelogram (quad).



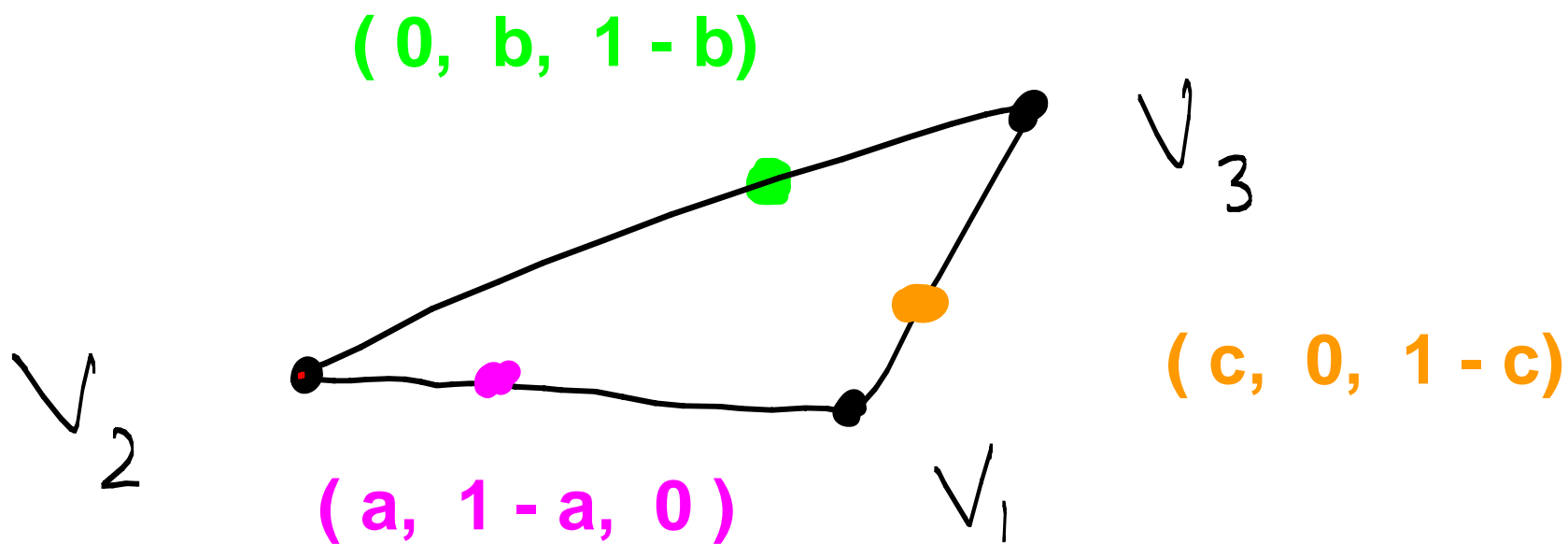
$$\begin{aligned} & v_1 + b(v_2 - v_1) + c(v_3 - v_1) \\ &= (1 - b - c)v_1 + bv_2 + cv_3 \\ &= a v_1 + b v_2 + c v_3 \end{aligned}$$

where $a = 1 - b - c$.

If

$$\begin{aligned} 0 &\leq a \leq 1 \\ 0 &\leq b \leq 1 \\ 0 &\leq c \leq 1 \end{aligned}$$

then we get "convex combinations" of v_1 , v_2 , v_3 .

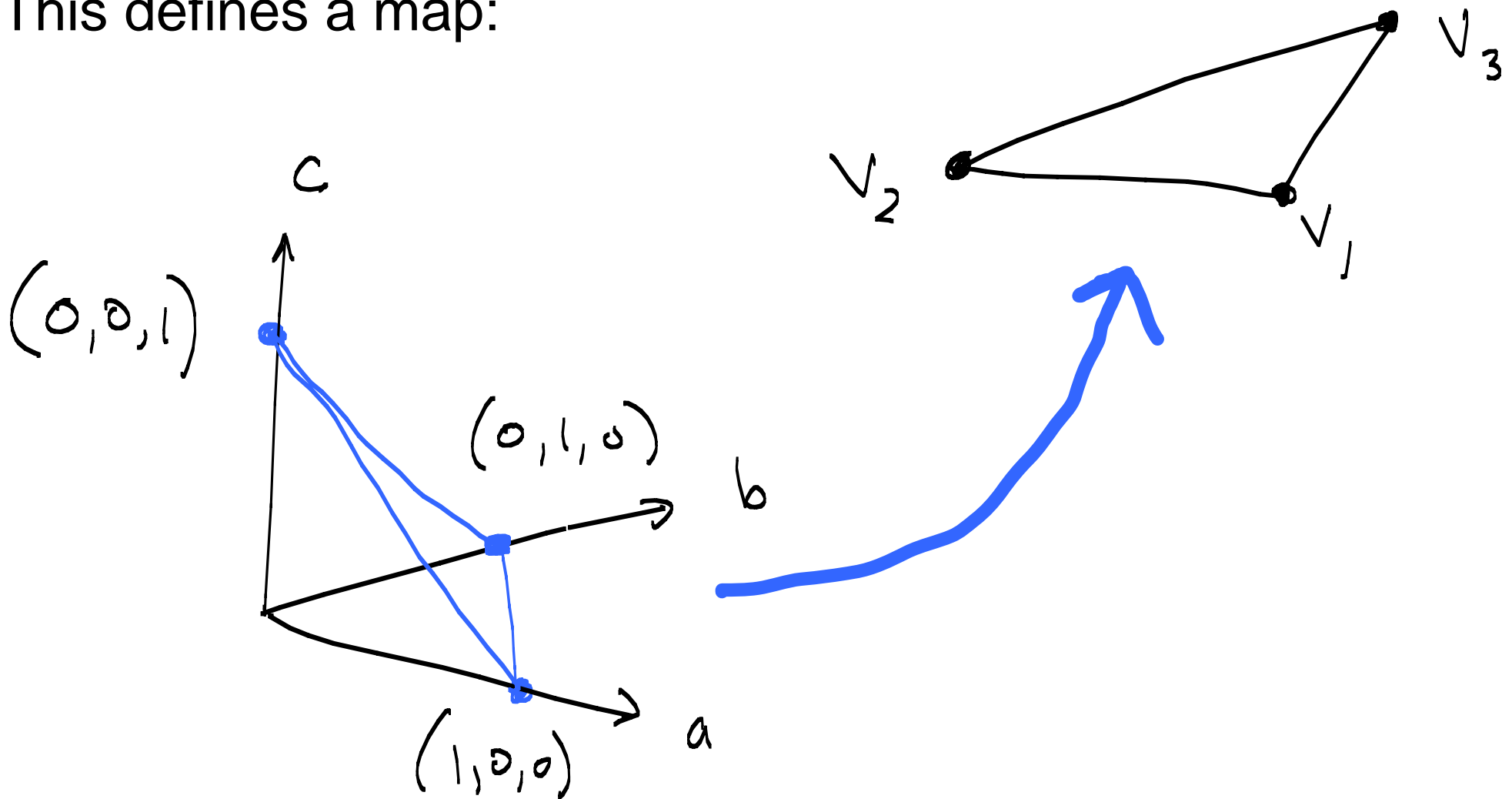


$$a v_1 + b v_2 + c v_3$$

ASIDE: A triangle in 3D mesh is called a "2-simplex".

The coefficients a , b , c are called "barycentric coordinates" of points in the triangle.

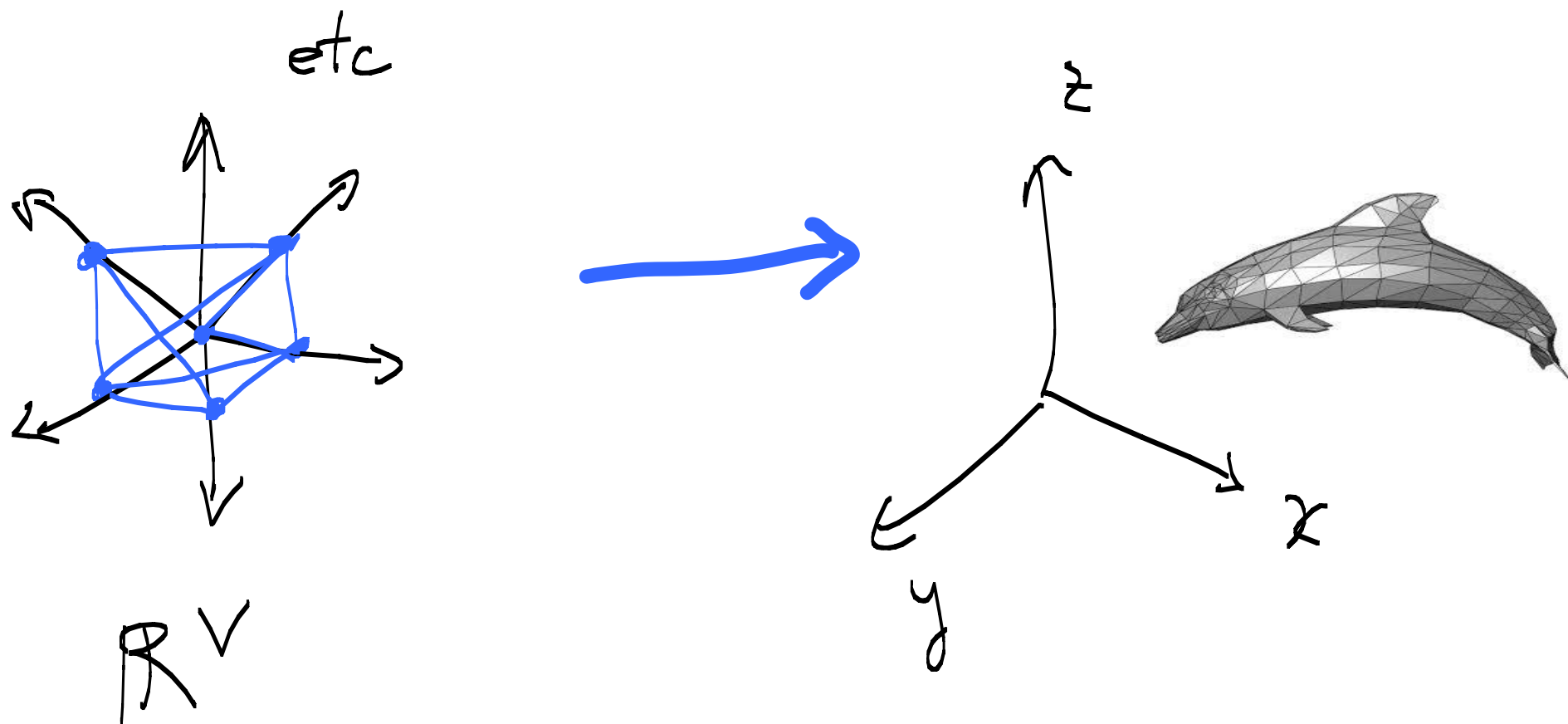
This defines a map:



Representing the mesh in \mathbb{R}^V using 2-simplexes.

For each triangle in the mesh, we have a 2-simplex in \mathbb{R}^V .

There are F of these 2-simplices, one for each triangle.
Their points are in 1-1 correspondence with the points on the mesh in \mathbb{R}^3 .



Example

Consider a point that lies near middle of face (v1, v2, v7).
This point can be parameterized by a point in \mathbb{R}^V :

| | |
|-----|-----|
| v1 | .3 |
| v2 | .35 |
| v3 | 0 |
| v4 | 0 |
| v5 | 0 |
| v6 | 0 |
| v7 | .35 |
| v8 | 0 |
| : | |
| v_V | 0 |

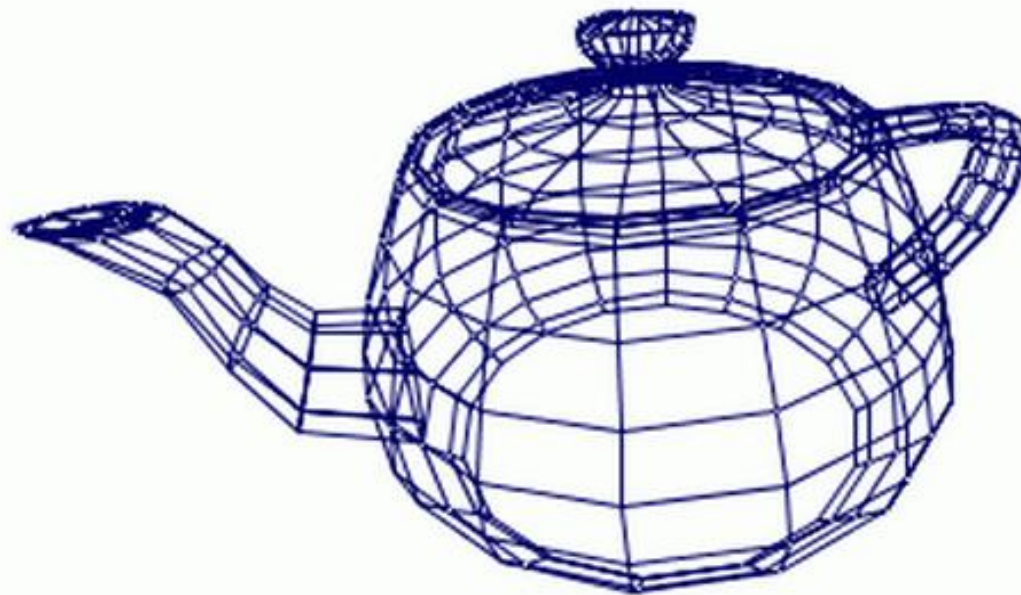
i.e. number of vertices in mesh is V

Where do polygon meshes come from ?

- construct "by hand"

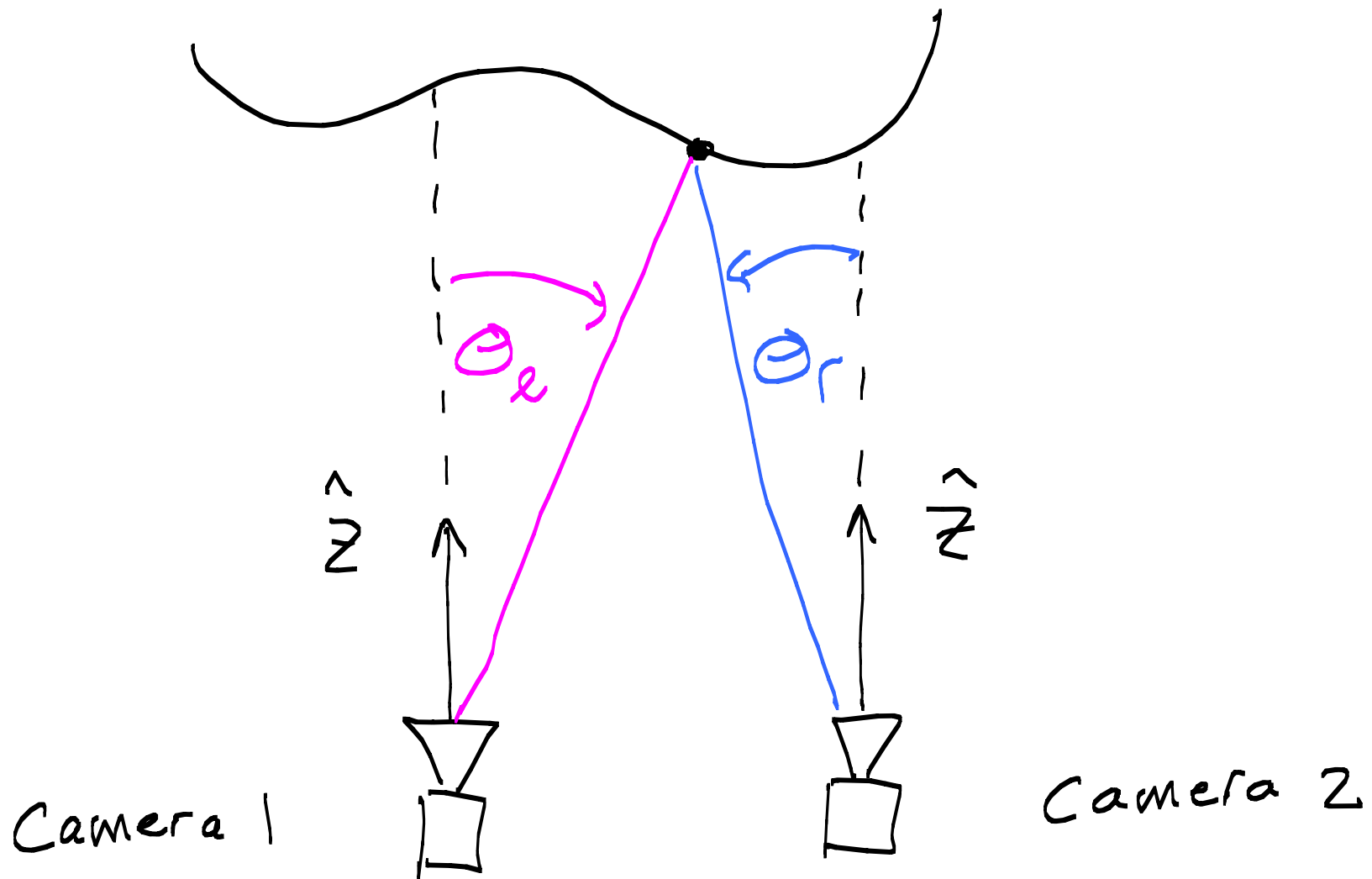
e.g. Utah Teapot (1974) 28 Bézier (bicubic) patches

http://www.sjbaker.org/wiki/index.php?title=The_History_of_The_Teapot

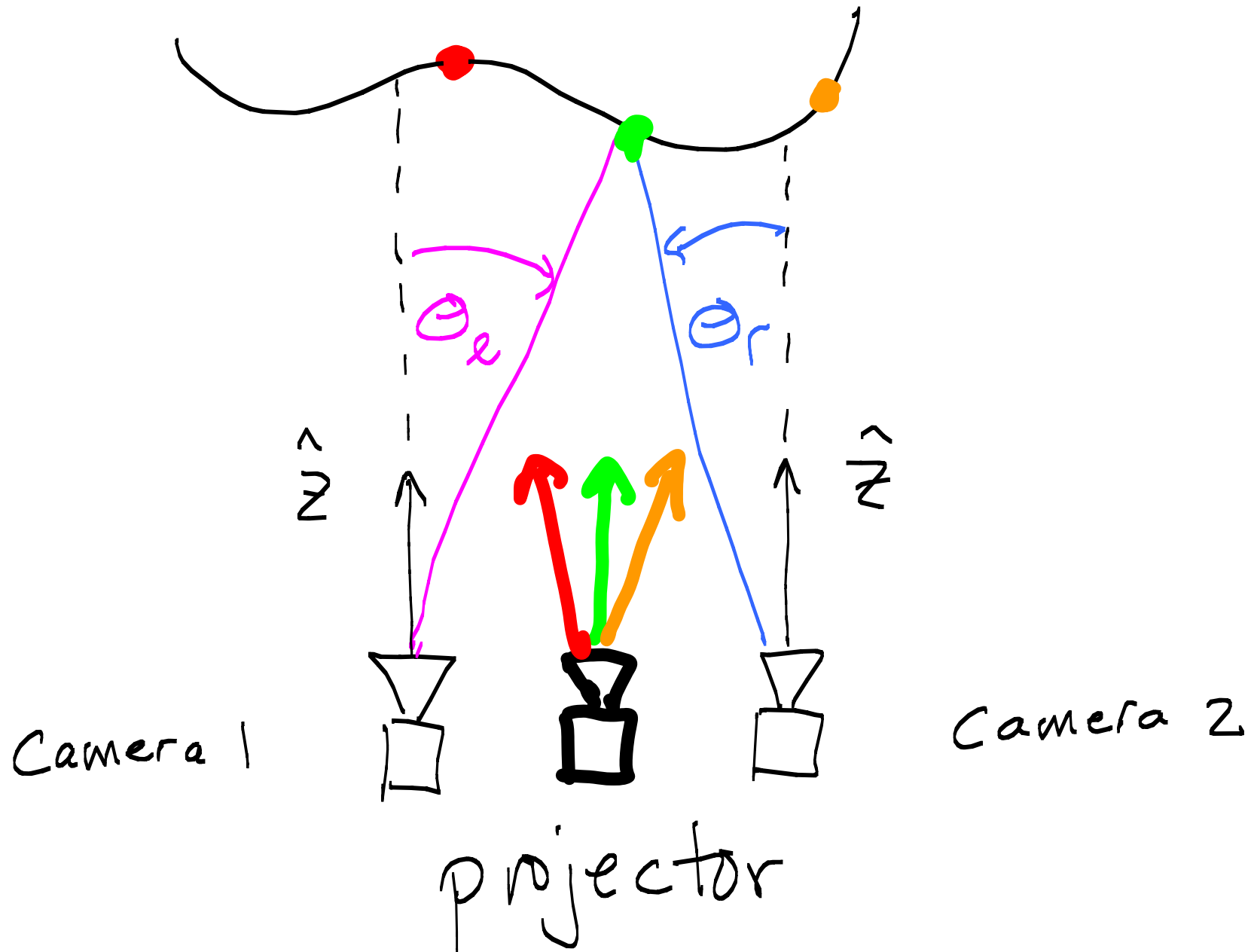


Where do polygon meshes come from ?

- automatically acquire using computer vision
e.g. two cameras + principle of triangulation.



Computer Vision (two cameras + structured light)



Examples (Technologies)

- Cyberware scanner (1990's)
- Light stage: <http://www.pauldebevec.com/>
(SIGGRAPH in 2000.)
- Recently, consumer level (but quality is poor)

Microsoft Kinect captures RGB + Z

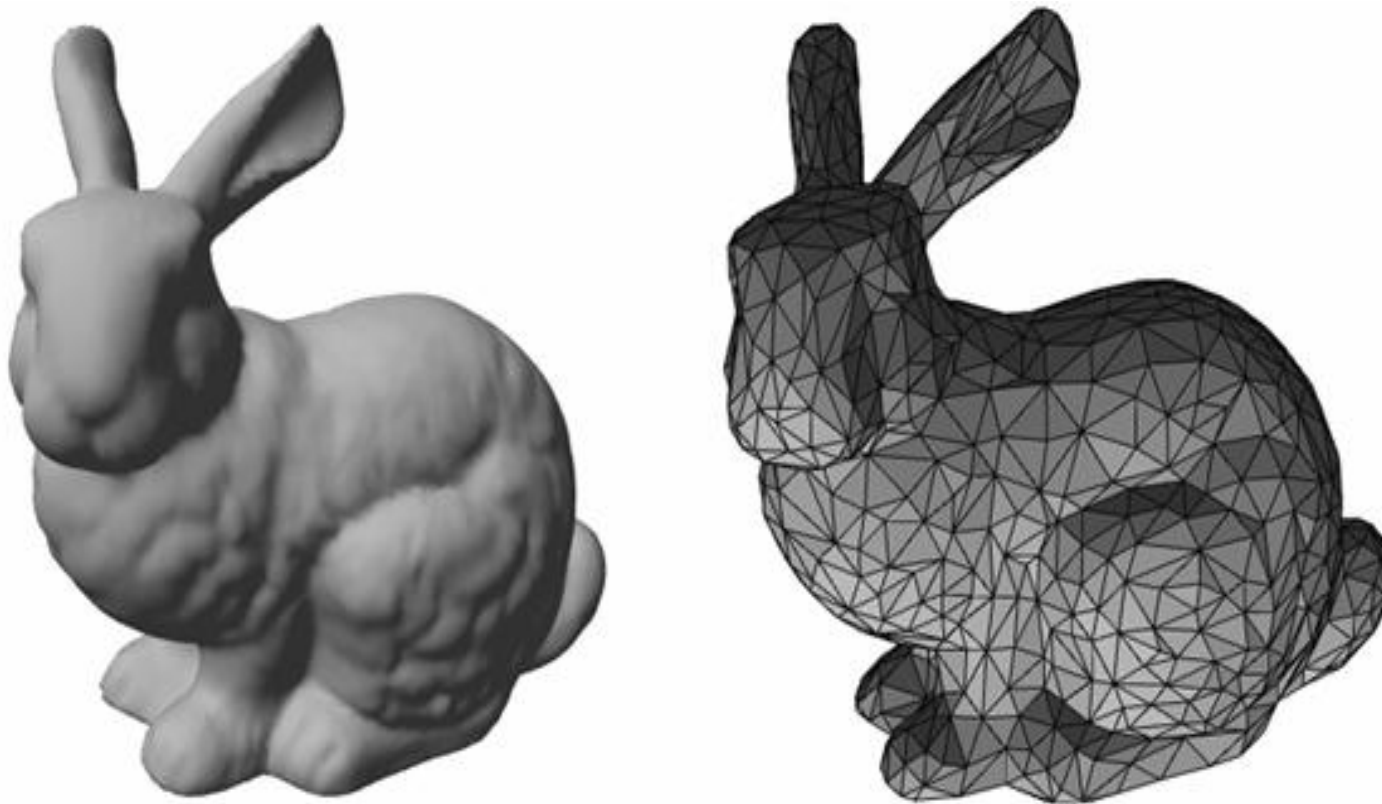
https://www.youtube.com/watch?v=uq9SEJxZiUg&feature=em-subscriptions_digest-vrecs

You can capture an image (RGB) too, i.e. RGB+Z.

Stanford bunny (1994)

<http://graphics.stanford.edu/data/3Dscanrep/>

Used a Cyberware scanner. 70,000 polygons.



<http://www.cc.gatech.edu/~turk/bunny/bunny.html>

See many amusing examples of how bunny has been used.

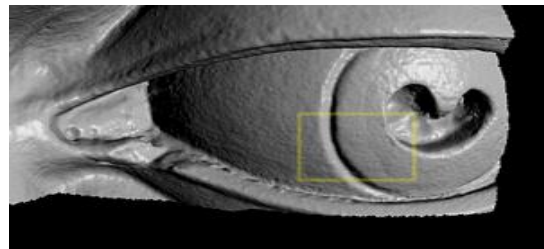
Digital Michelangelo (2000)

<http://graphics.stanford.edu/projects/mich/>

Used a custom built Cyberware scanner.
~2 billion polygons. Resolution was 0.2 mm.



eye (photo)



eye (rendered from model)



Need to register, clip, and stitch together multiple scans.

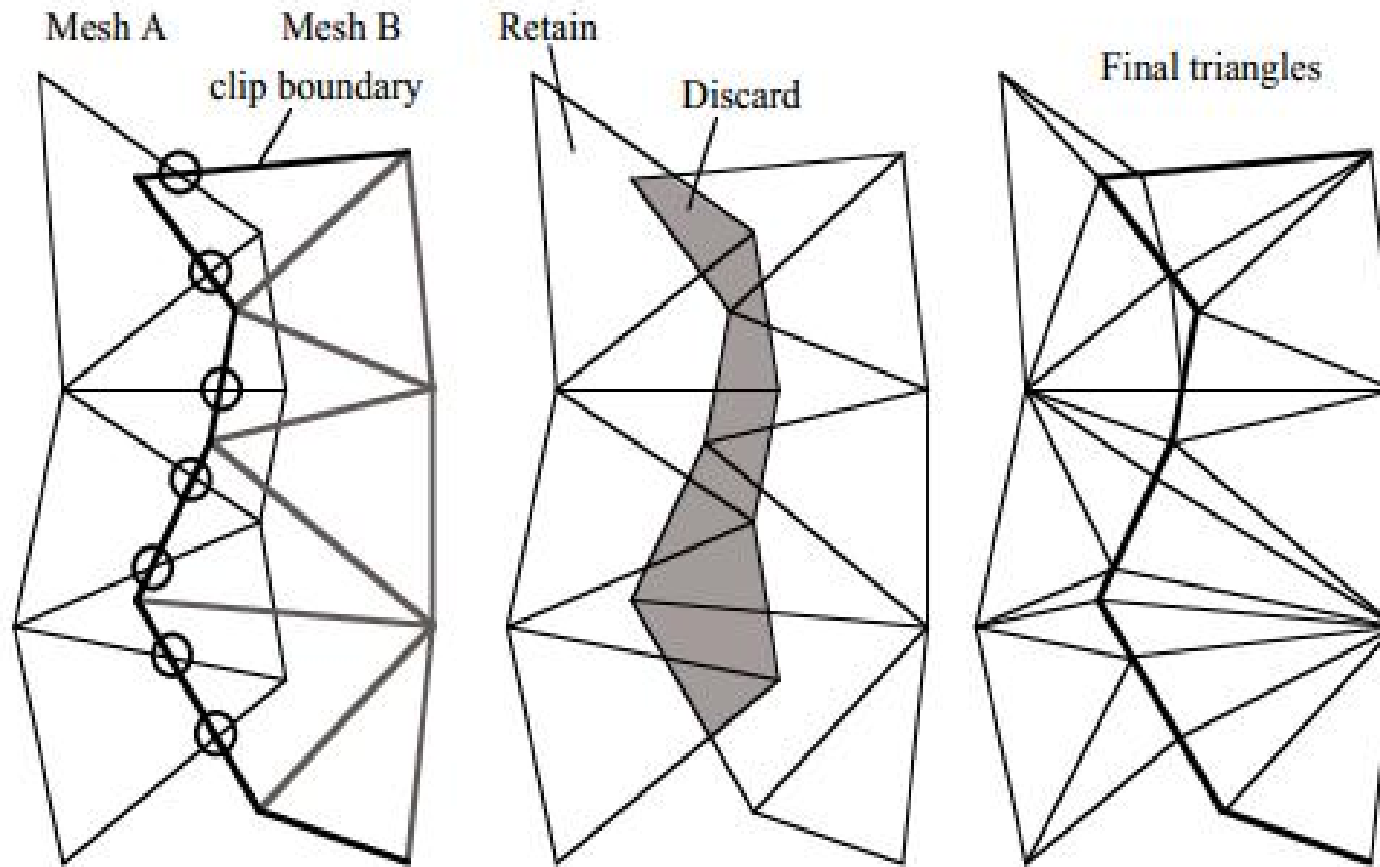
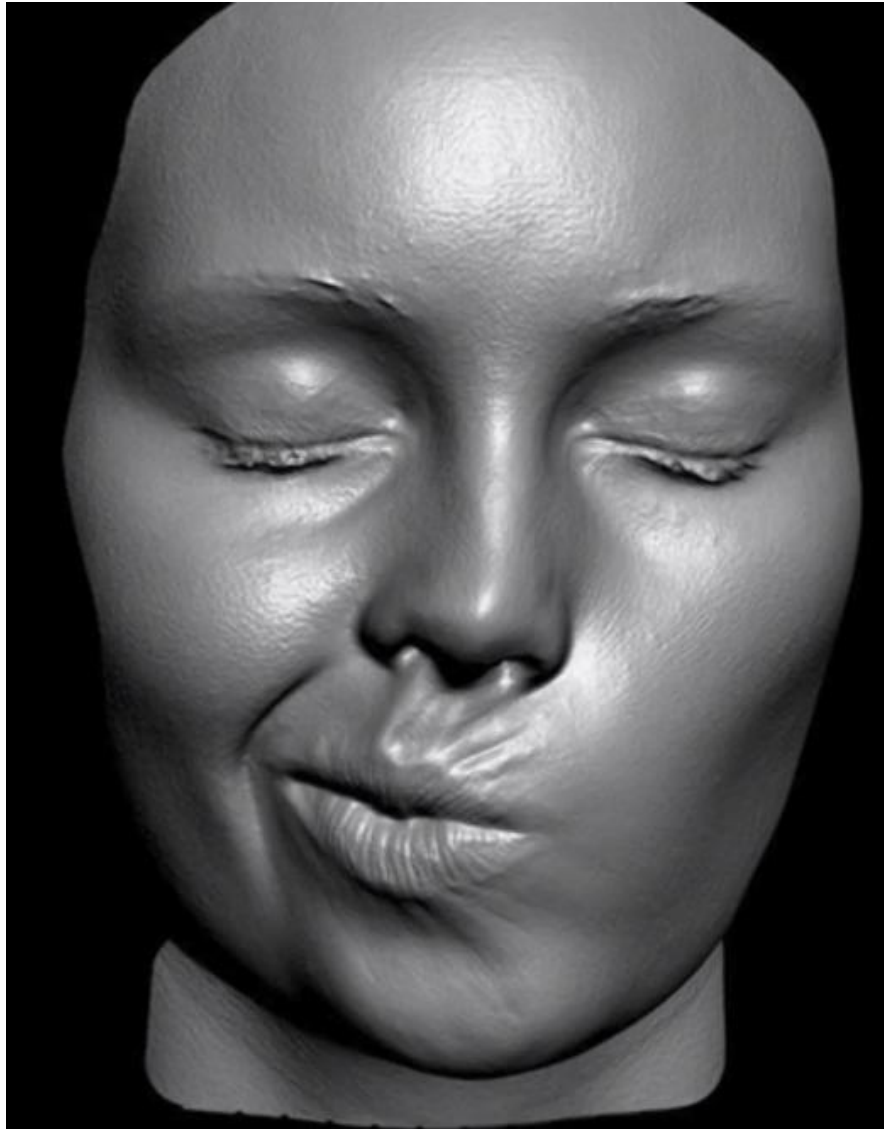
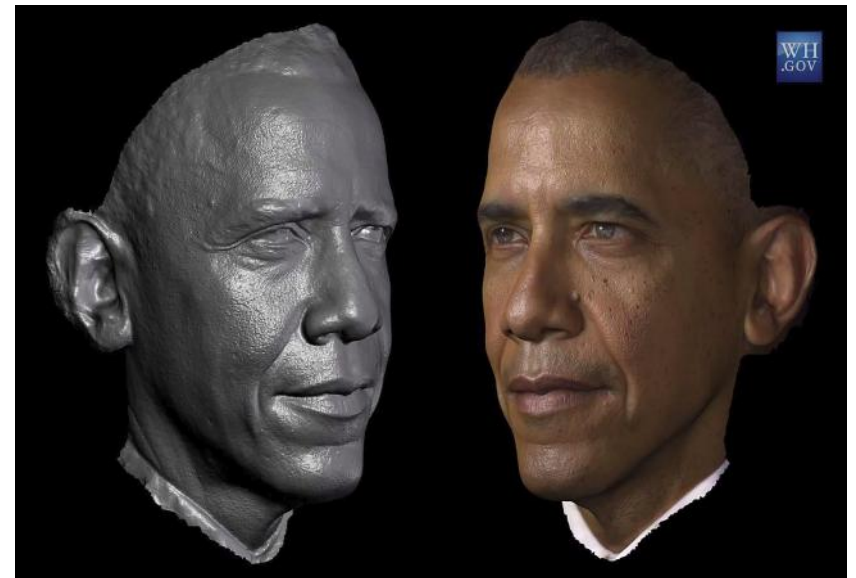


Figure 5: Mesh *A* is clipped against the boundary of mesh *B*. Circles (left) show intersection between edges of *A* and *B*'s boundary. Portions of triangles from *A* are discarded (middle) and then both meshes incorporate the points of intersection (right).

Light Stage scans not just geometry and color but also surface material (reflectance/shinyness).

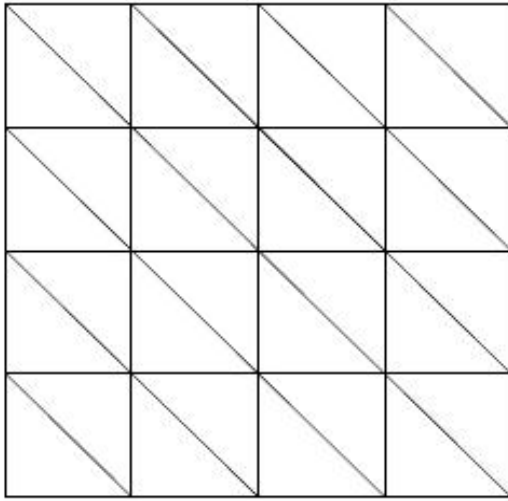


<http://gl.ict.usc.edu/Research/PresidentialPortrait/>

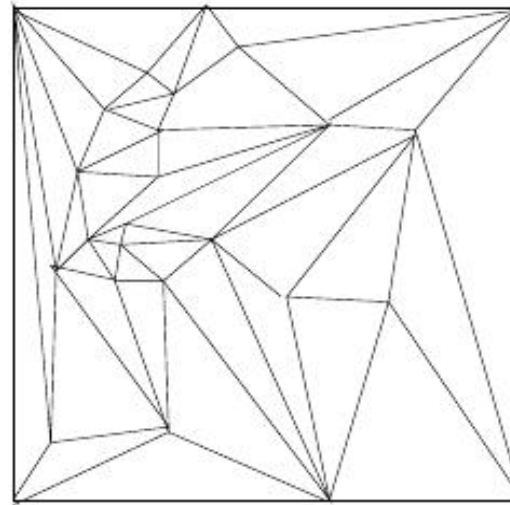


http://www.ted.com/talks/paul_debevec_animates_a_photo_real_digital_face?language=en

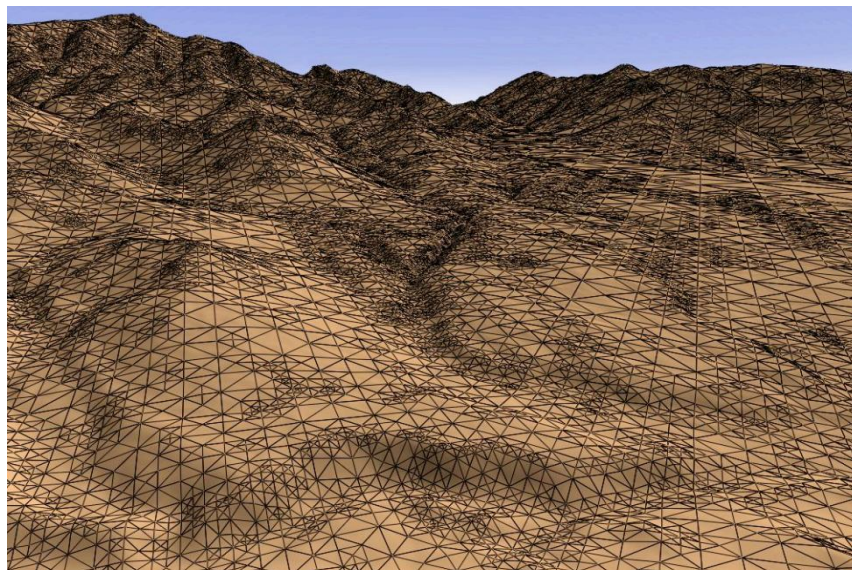
Terrain Mesh $(x, y, z(x,y))$



regular (x,y)



non-regular (x,y)



lecture 11

meshes

- basic definitions and data structures, parameterization, acquisition
- level of detail, simplification, subdivision

Level of Detail



60
triangles



600
triangles



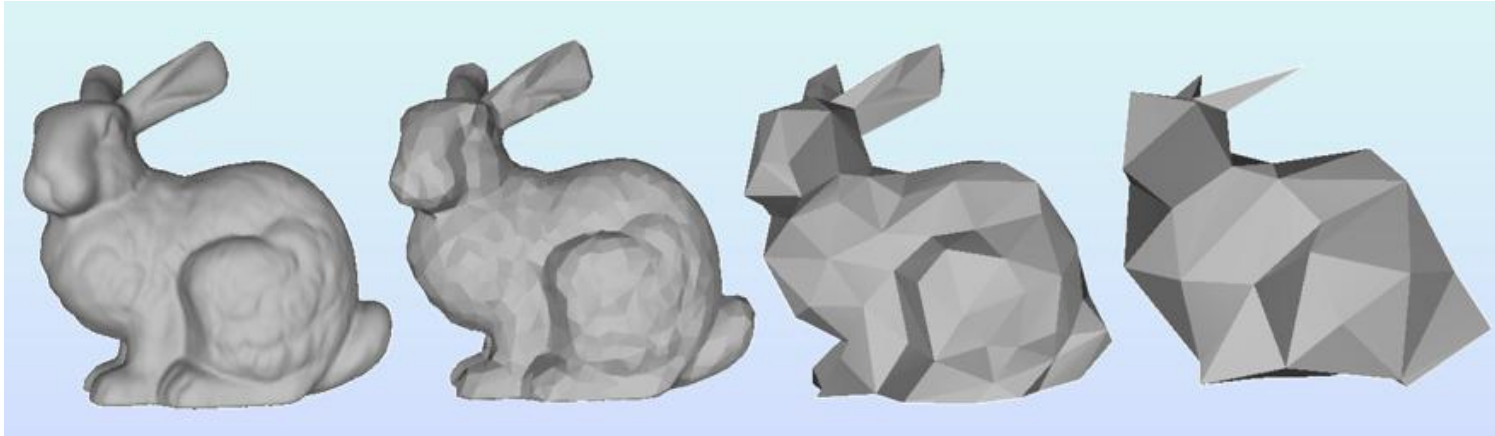
6000
triangles



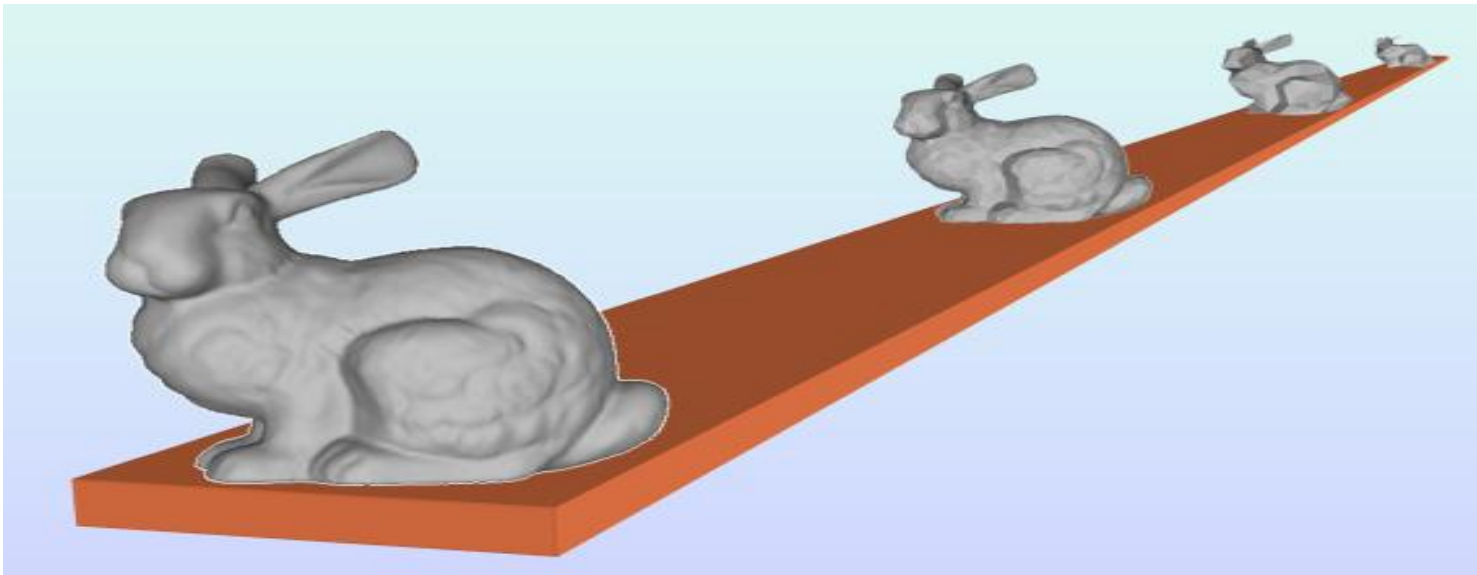
60000
triangles

More triangles --> more accurate (good)
but takes longer to draw (bad)

Precompute different LOD's.

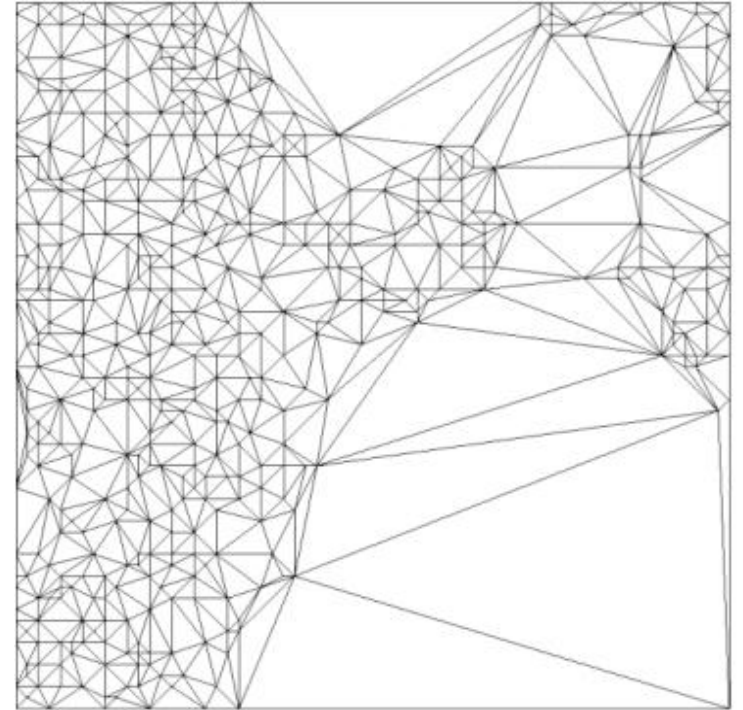
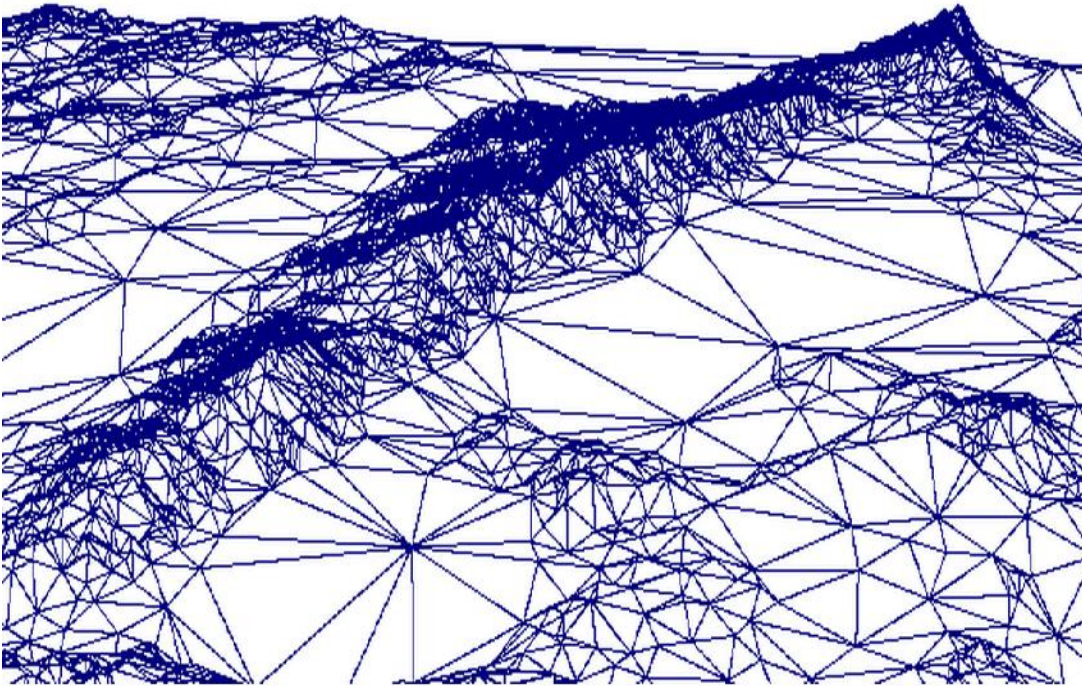


Use coarse LOD for distant objects. (Easy to program.)



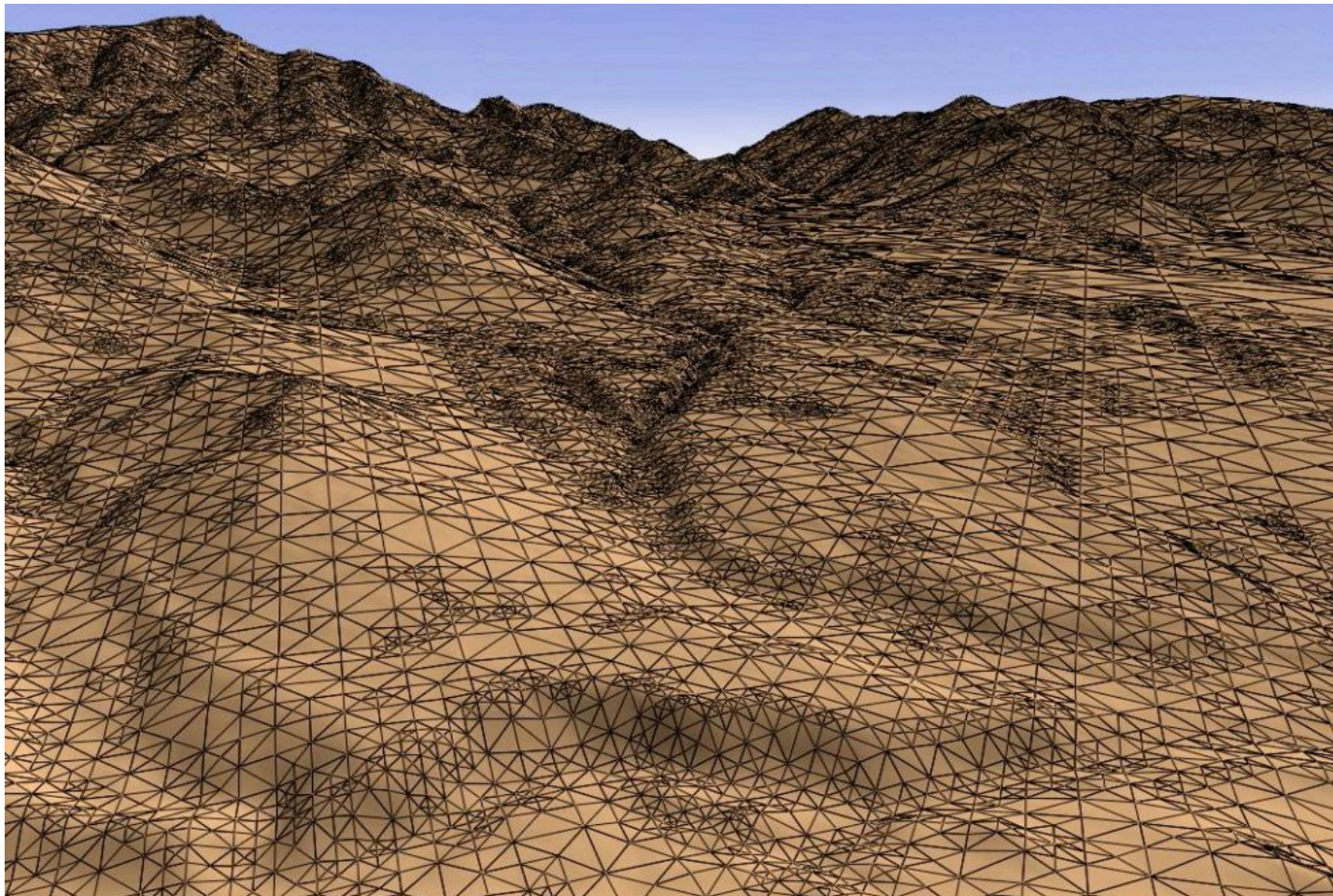
from David Luebke's slides at U Virginia

Terrains and LOD (level of detail)



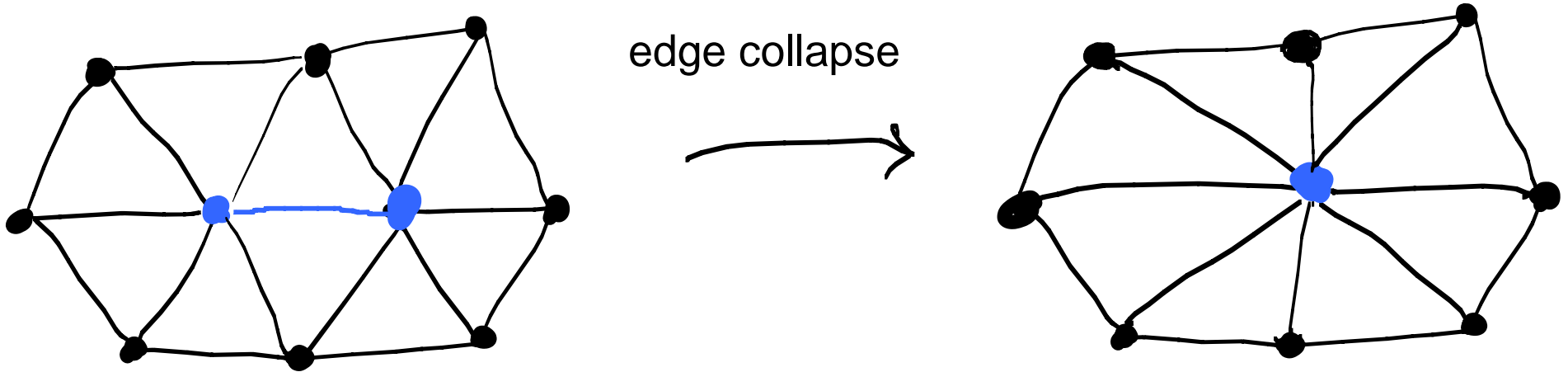
Use higher sampling in complicated regions of surface, lower sampling in smooth regions.

Combining the above two approaches ? You could try to use coarse LOD for distant parts of the terrain.
(This was NOT done in image below.)



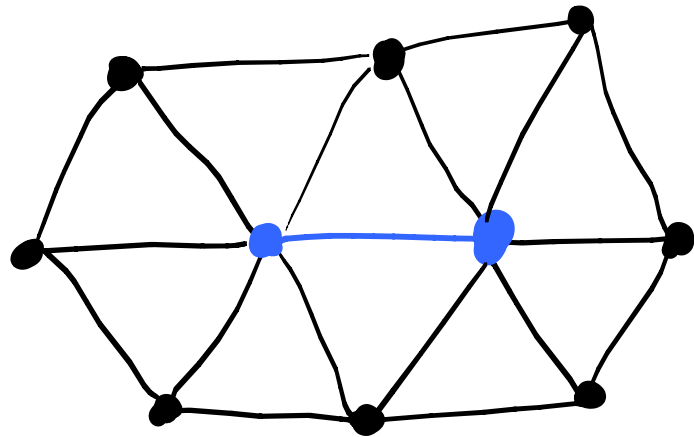
Lindstrom 1996

Mesh Simplification

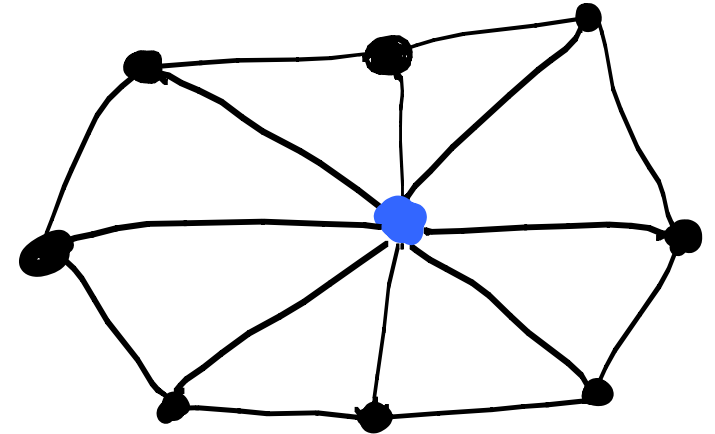


Edge collapse

- number of edges decreases by 3
- number of faces decreases by 2
- number of vertices decreases by 1

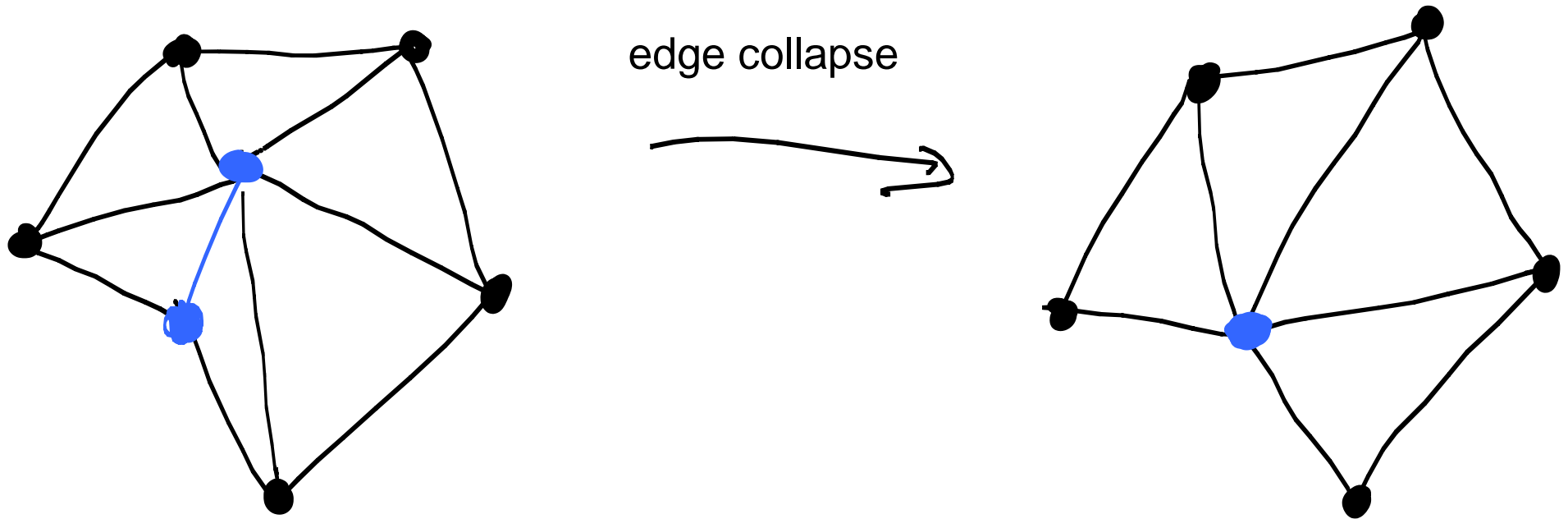


vertex split

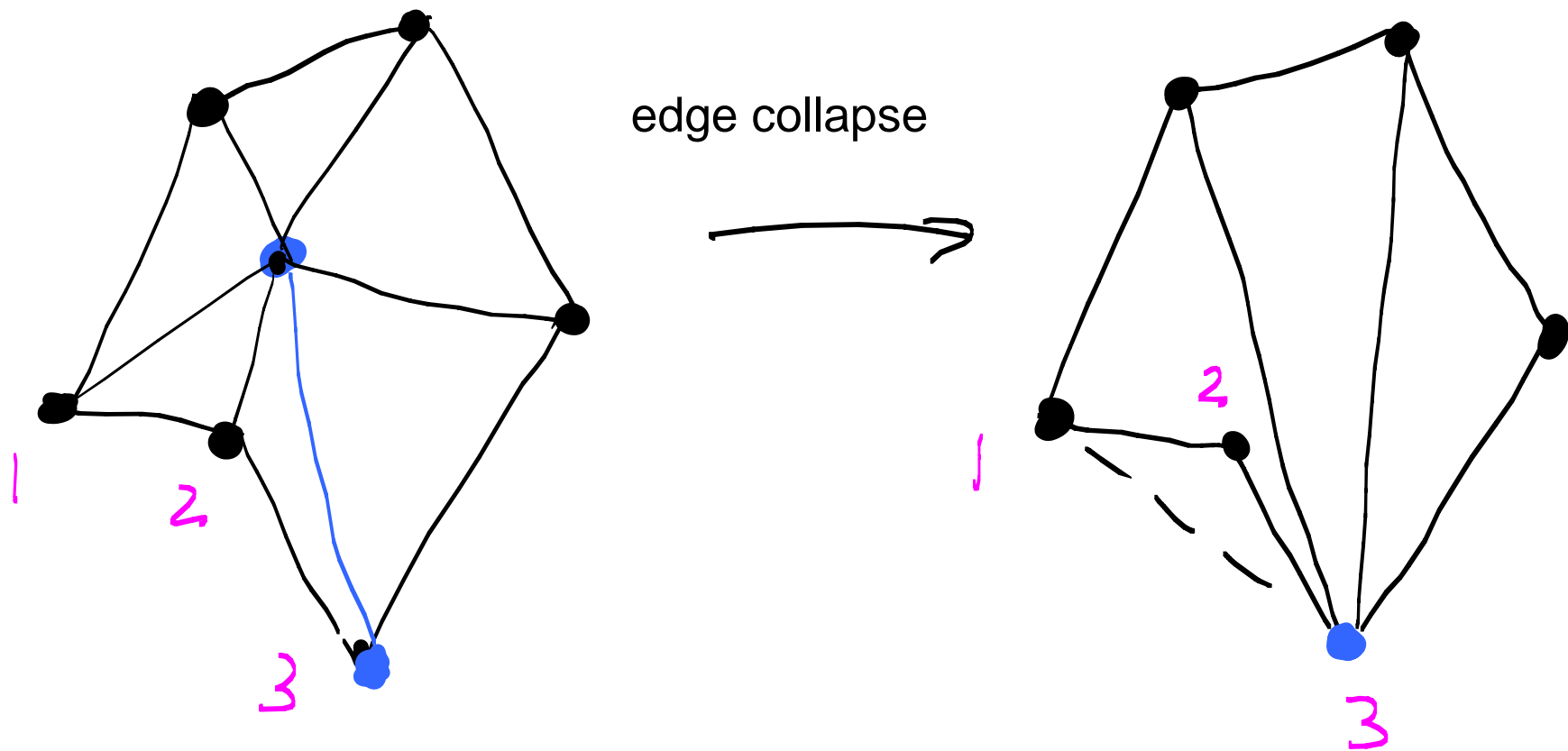


Vertex split is just the inverse of an edge collapse,....

- number of edges increases by 3
- number of faces increases by 2
- number of vertices increases by 1

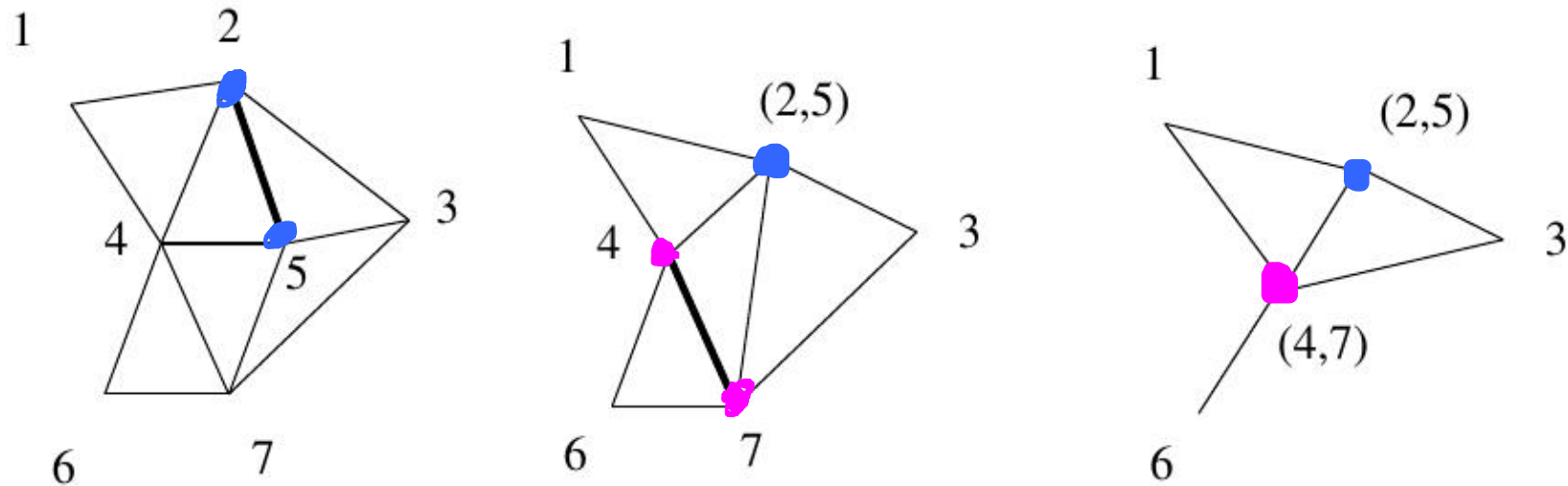


Once we choose the edge to collapse, we still need to decide which vertex to keep (or we can move the vertex to a new position).

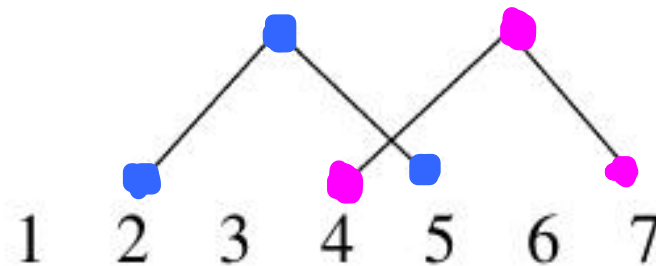


In this example, we make a bad choice.
The new triangle 123 is flipped.

Forest data structure for simplified meshes



vertices

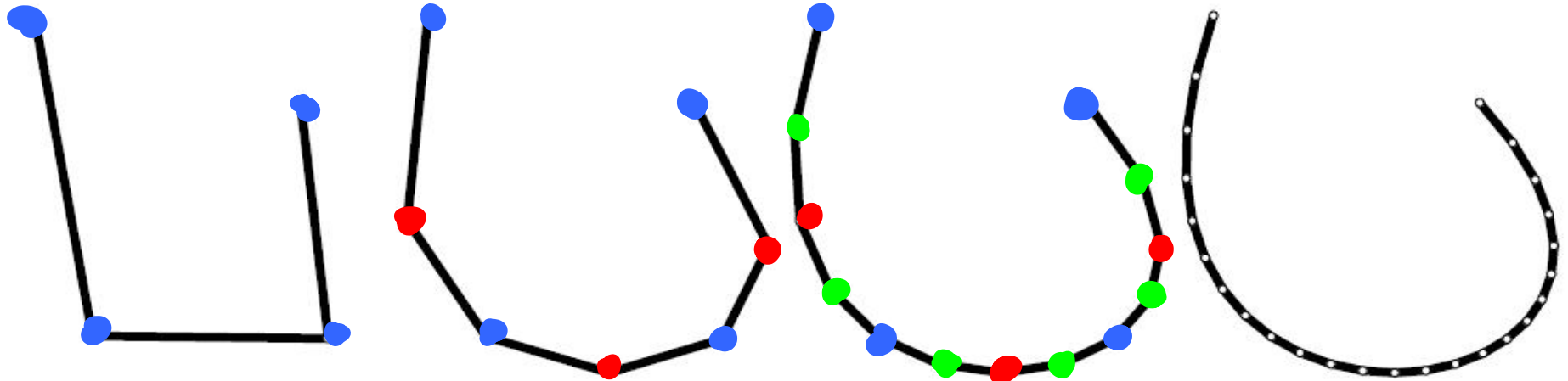


Mesh Modelling

- using bicubic patches and splines
- raw capture + simplification and LOD
(assume we start with high resolution)
- subdivision surfaces
(assume we start with low resolution)

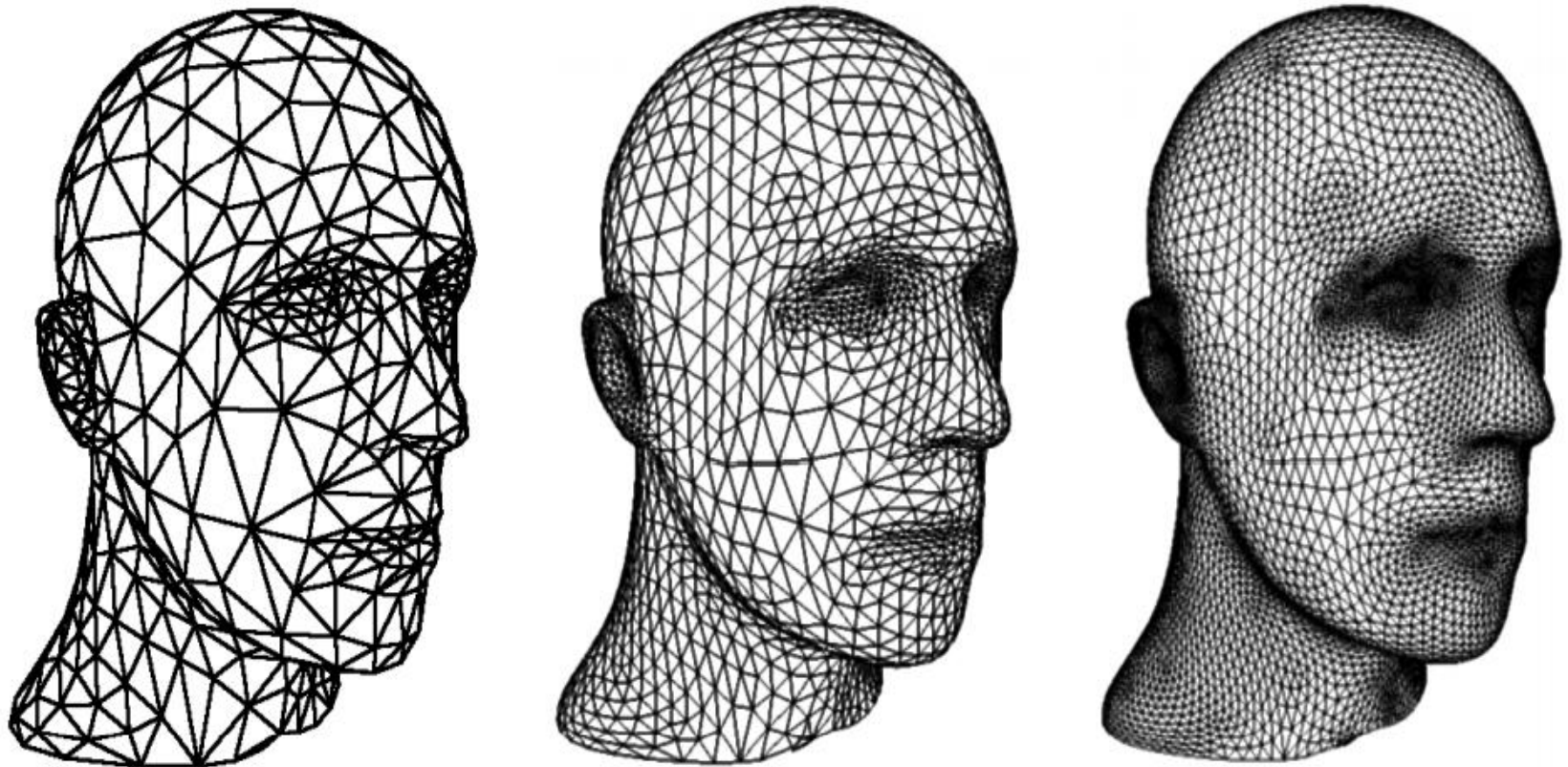
Subdivision Curves

Given a coarsely sampled *curve*, iteratively find a smooth approximation.



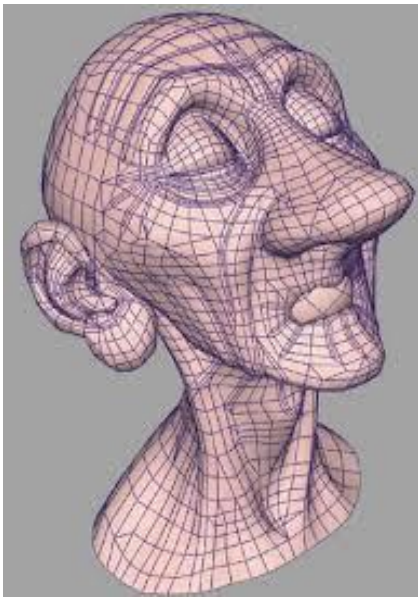
Subdivision Surfaces

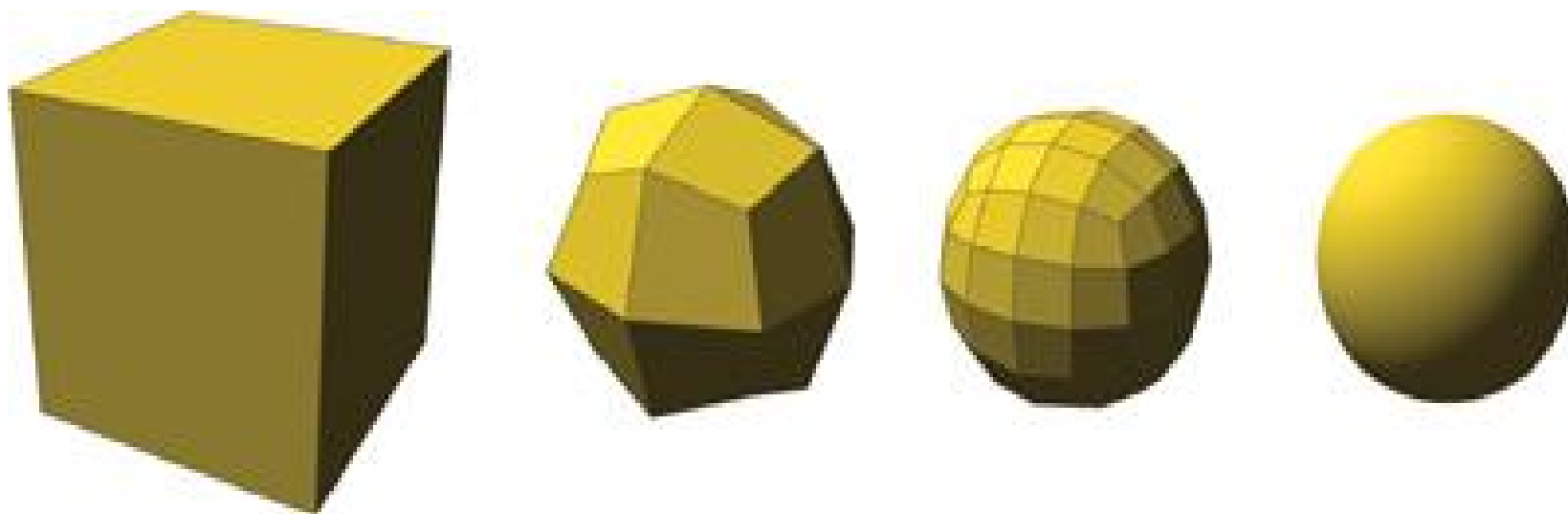
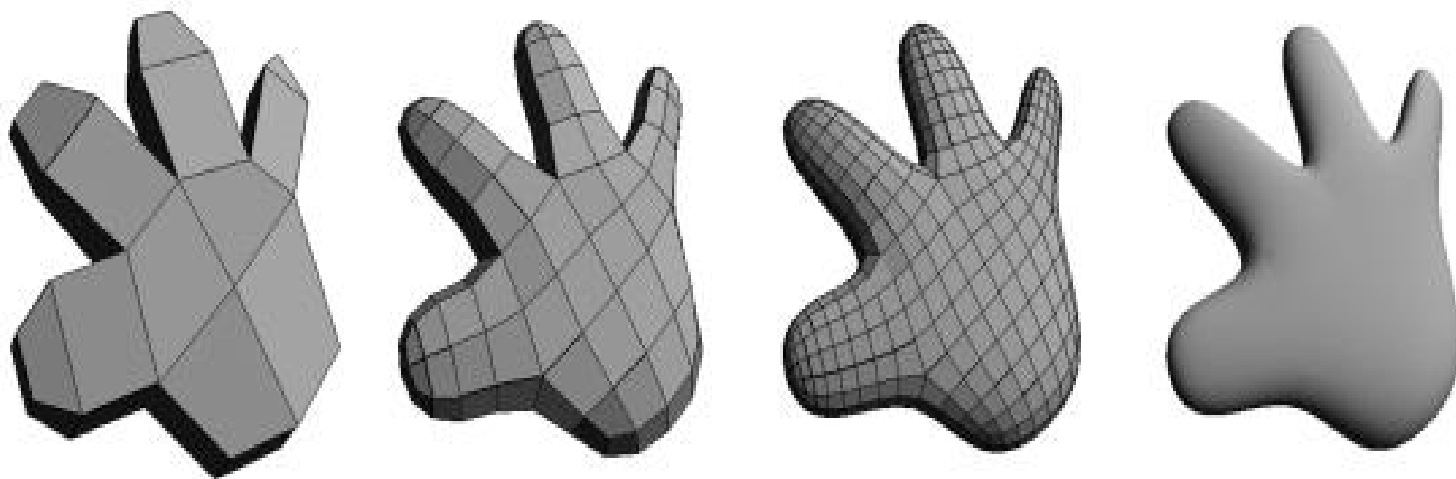
Given a coarsely sampled *surface*, iteratively find a smooth approximation.



Used by Pixar in a production first time in 1997 (Geri's game)

<https://www.youtube.com/watch?v=9IYRC7g2ICg>

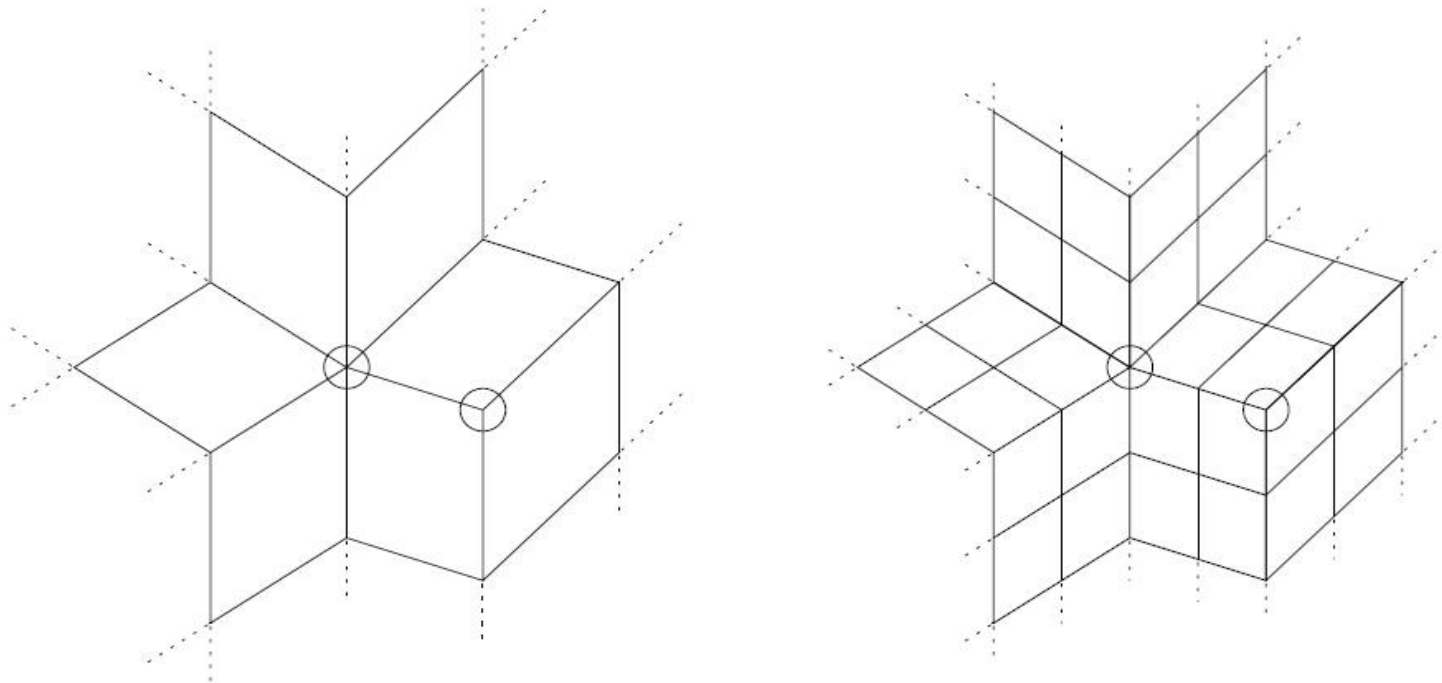




Q: Why not use splines ?

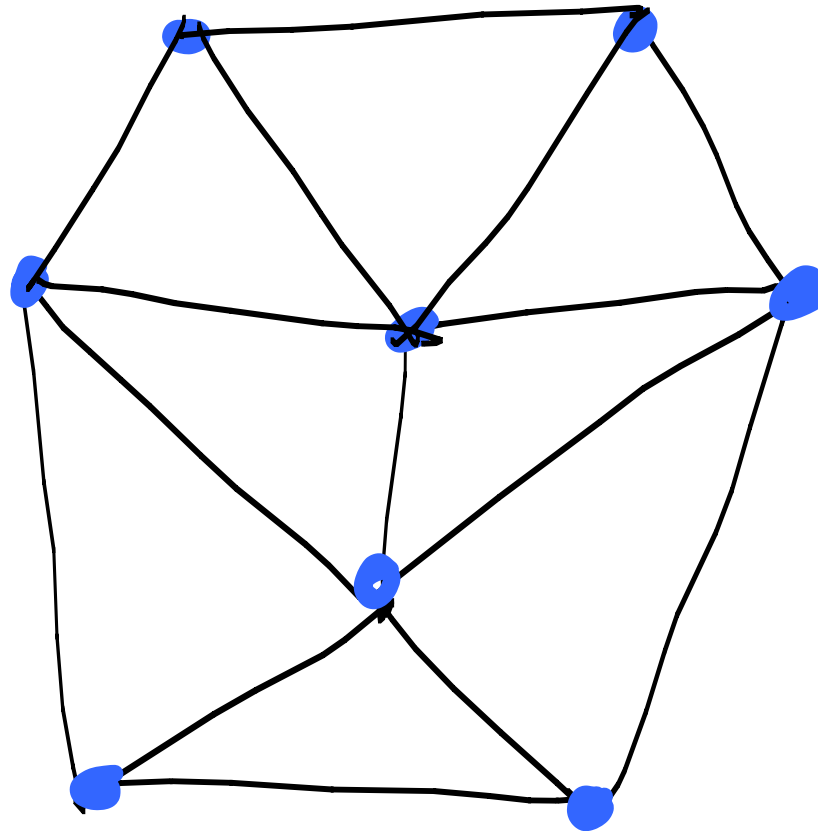
A: Subdivision methods handle general mesh connectivity.
(Splines tend to be less flexible e.g. bicubics require a 4×4 grid)

In the example below, two vertices are marked. one with degree 6 and one with degree 3. (Quad patches are used.)



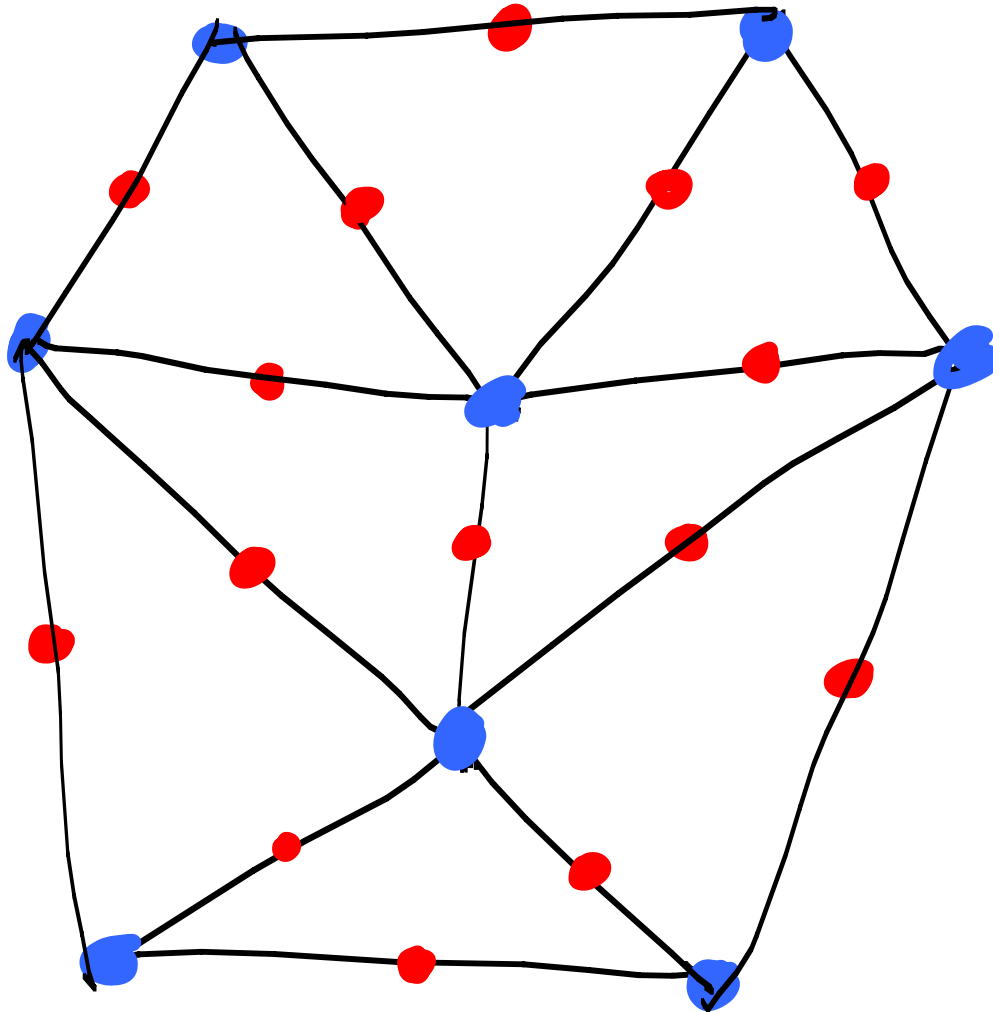
Loop Subdivision [Charles Loop 1987]

- defined for triangulated meshes only



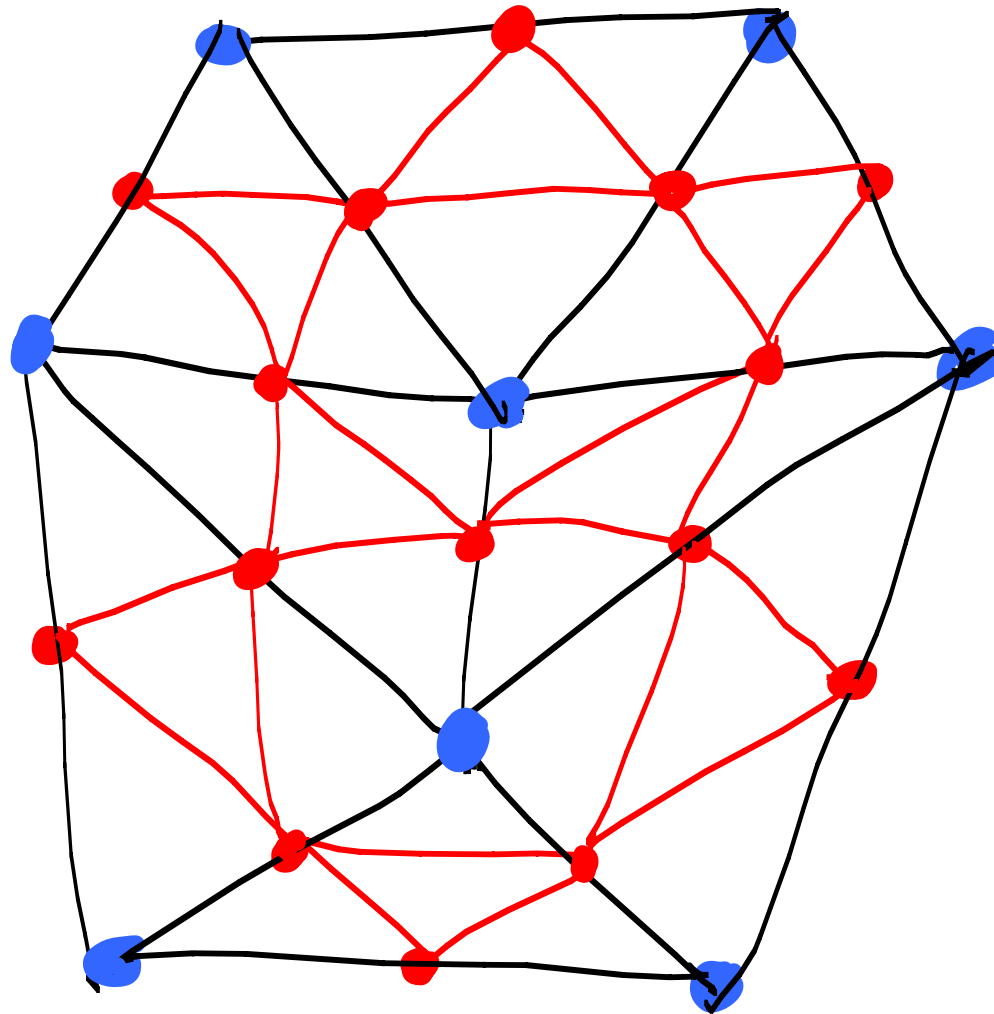
Step 1a (Refinement):

Add **new vertex** to midpoint of each edge.



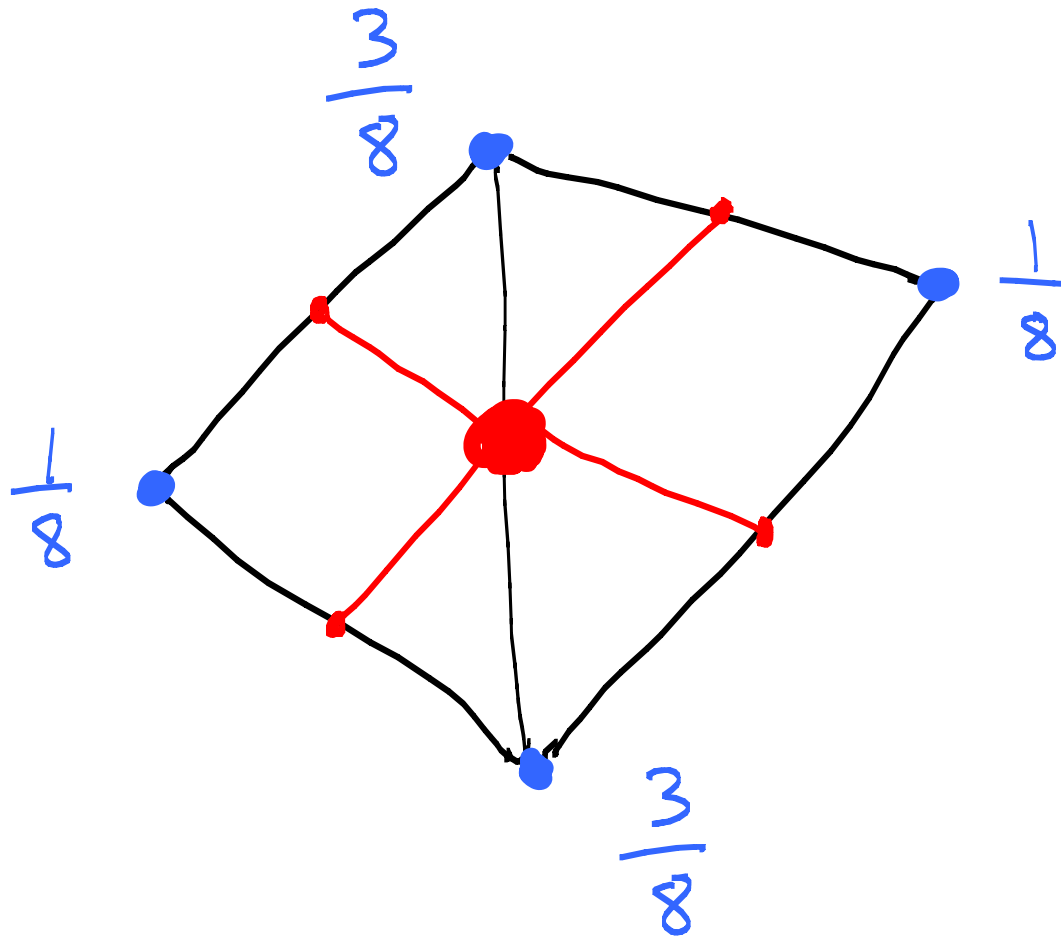
Step 1b (Refinement):

Add **new edges** between **new vertices**.



Step 2a (Smoothing):

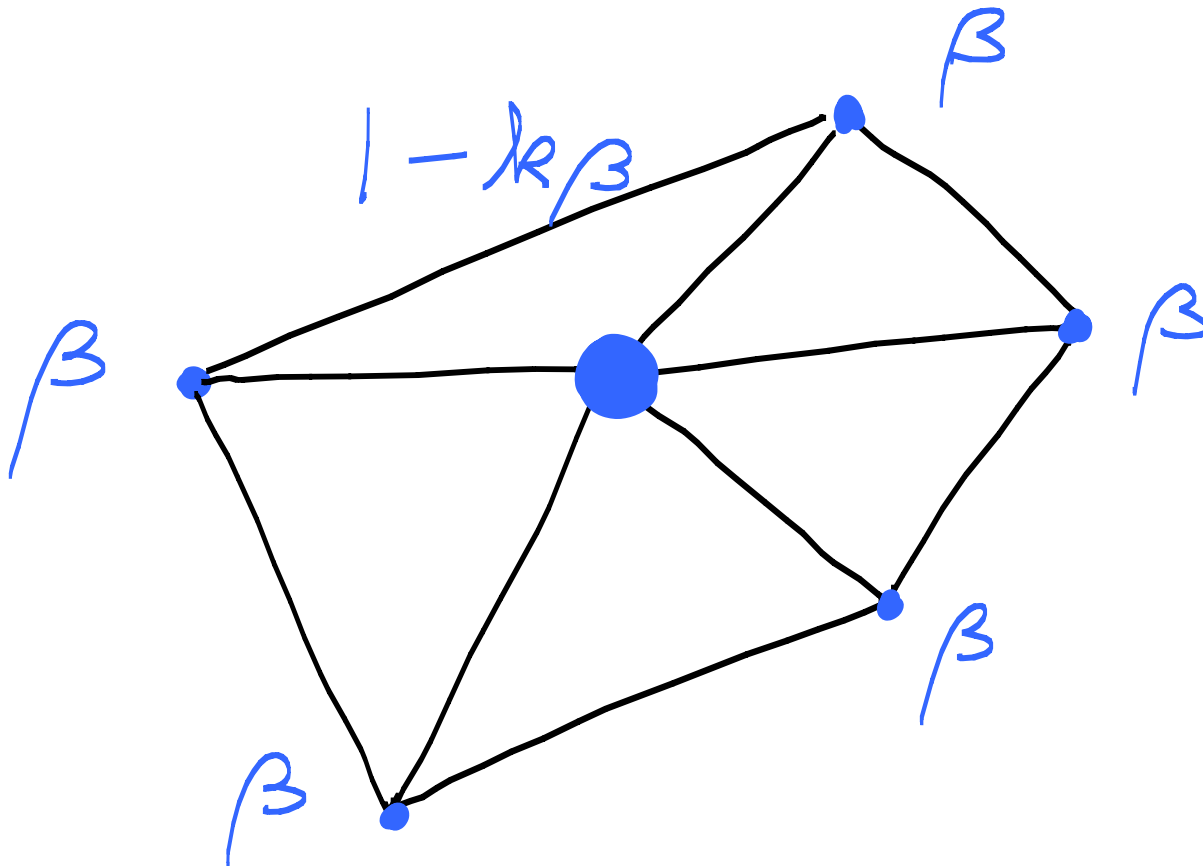
Position of **new vertex** is a weighted sum of the positions of **four neighboring old vertices** (uniquely defined, as in configuration below).



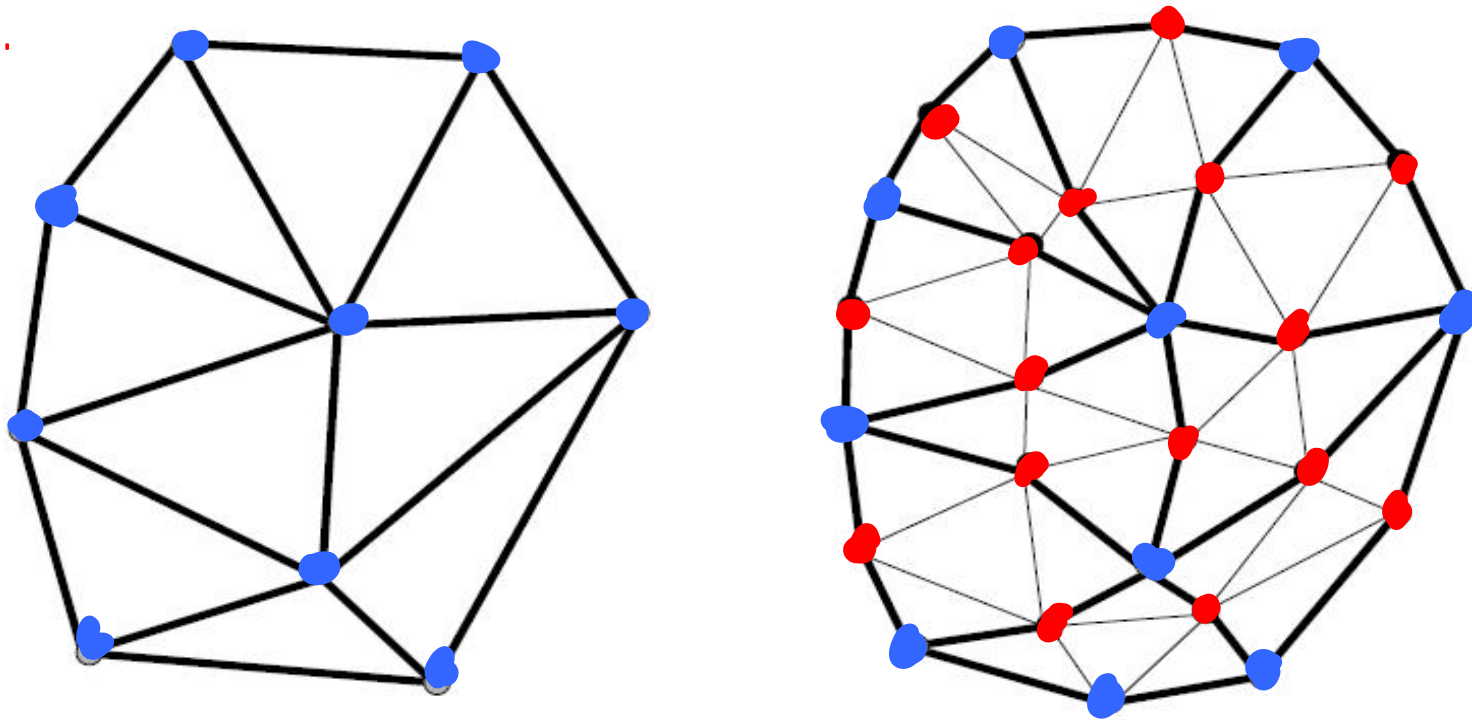
Step 2b (Smoothing):

New position of each **old vertex** is a weighted sum of the positions of **all neighboring old vertices** (uniquely defined, as in configuration below).

The constant β can be chosen as you wish.



Example



I have been discussing internal vertices and edges only. Other linear combination rules are needed for new edges and vertices on boundary. (Details omitted here).

"Interpolating"

- given a fixed set of vertices, fill in curve or surface that contains these vertices

(previous lectures: midpoint displacement, Hermite splines)

"Approximating"

- given an initial set of vertices, fit a curve or surface that comes close to (but might not contain) these vertices

(previous lecture: Bezier splines
this lecture: Loop subdivision)

(Mesh simplification can be interpolating or approximating.)

Announcements

- A1 grades by tonight
- midterm exam Thurs Feb 19.
Last name A-P (here)
Last name Q-Z (RPHYS 114)

(Material is up to today only)
- A2 (under construction)