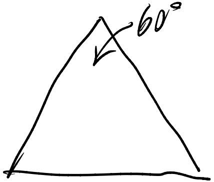
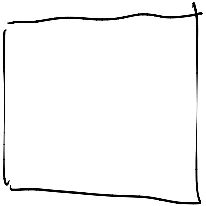
21 – Polygonal Modeling (Data structures, Triangle Meshes, Subdivision Surfaces)

Regular Polygons ~ convex, congruent sides & angles

60°

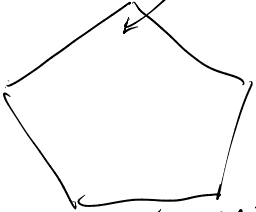


equilateral triangle



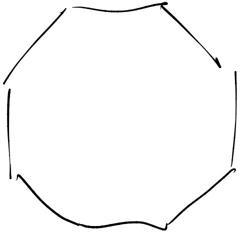
squares

108°



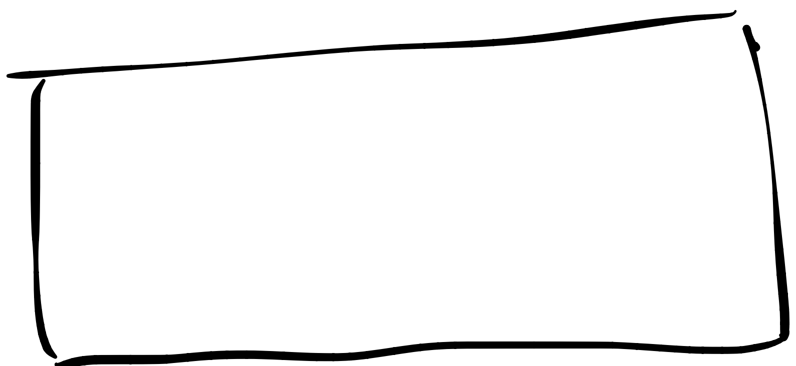
pentagons

hexagons

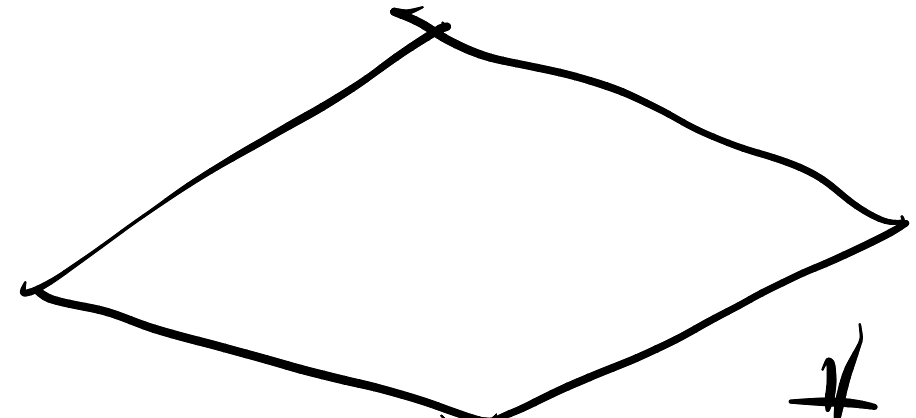


heptagons

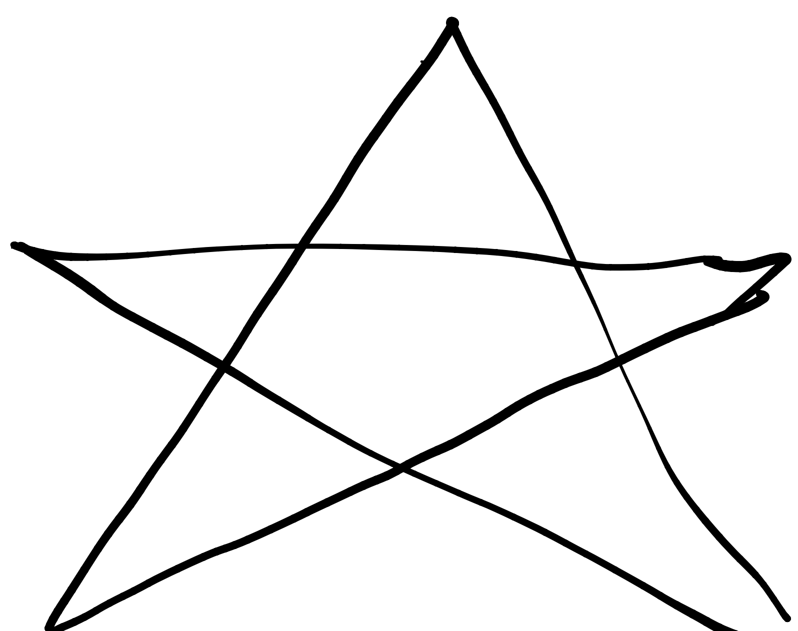
not regular



sides aren't congruent



angles aren't congruent

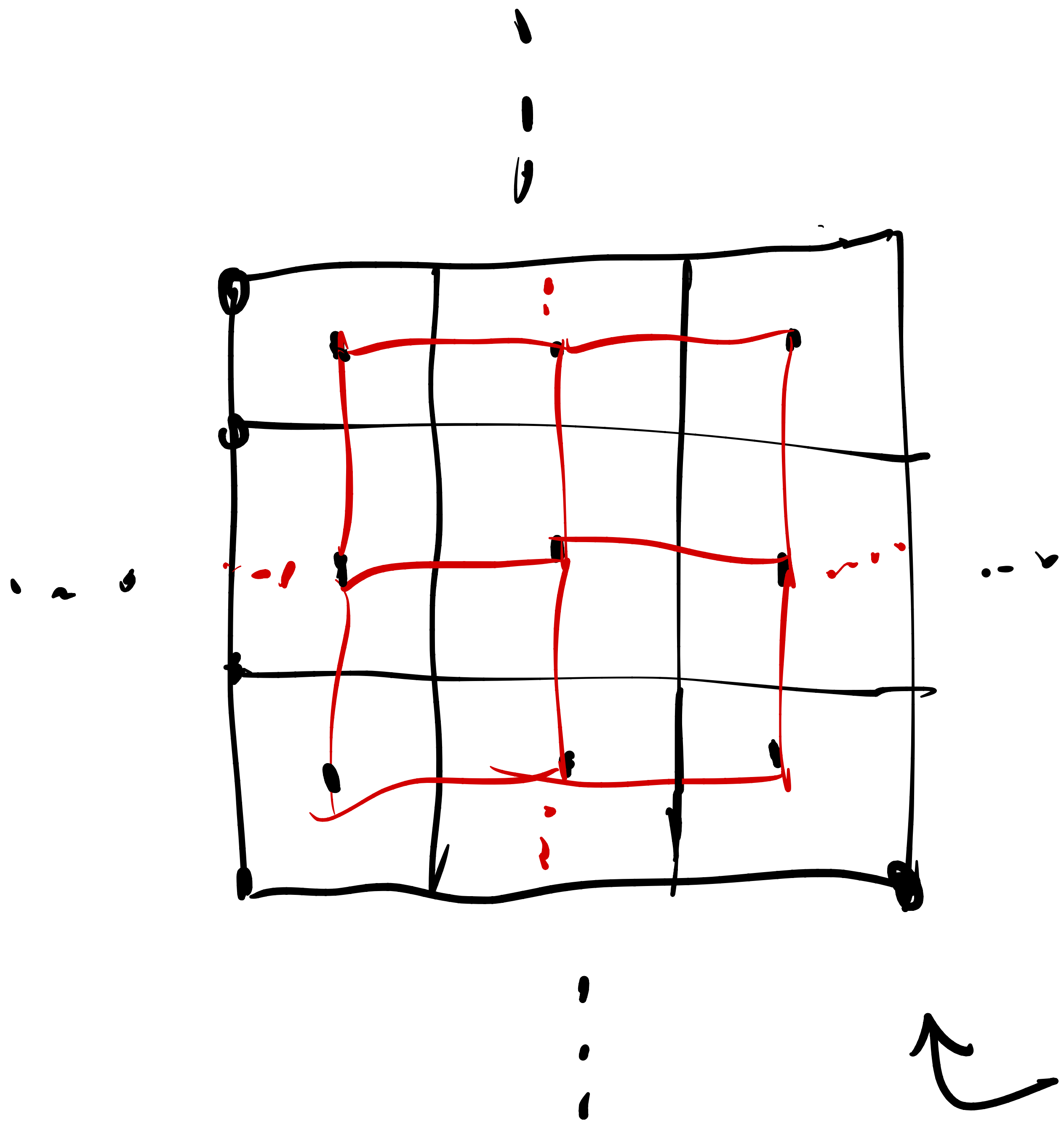


not convex

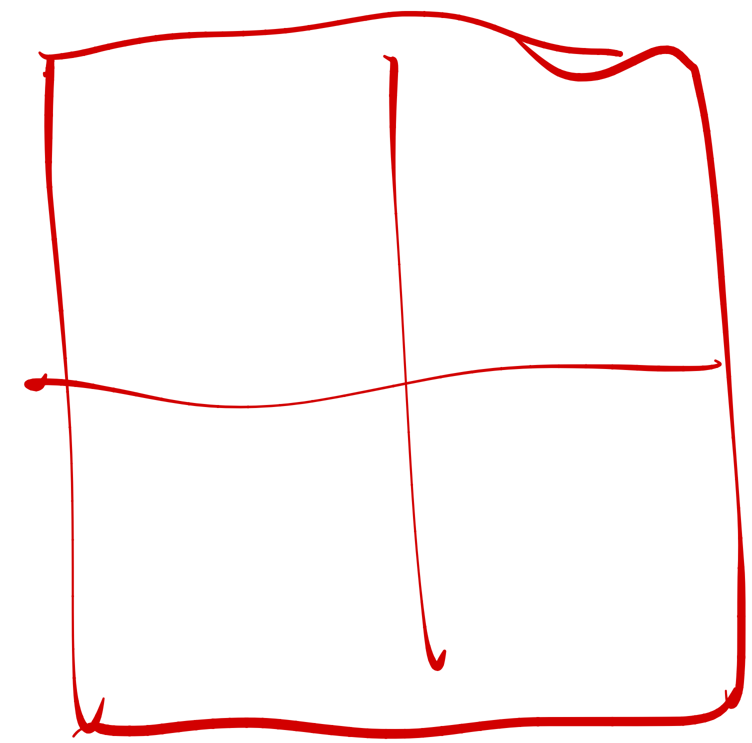
Regular Planar Tilings

→ fill plane with reg poly's
↳ no gaps, no overlaps

valence = number of faces surrounding a vertex



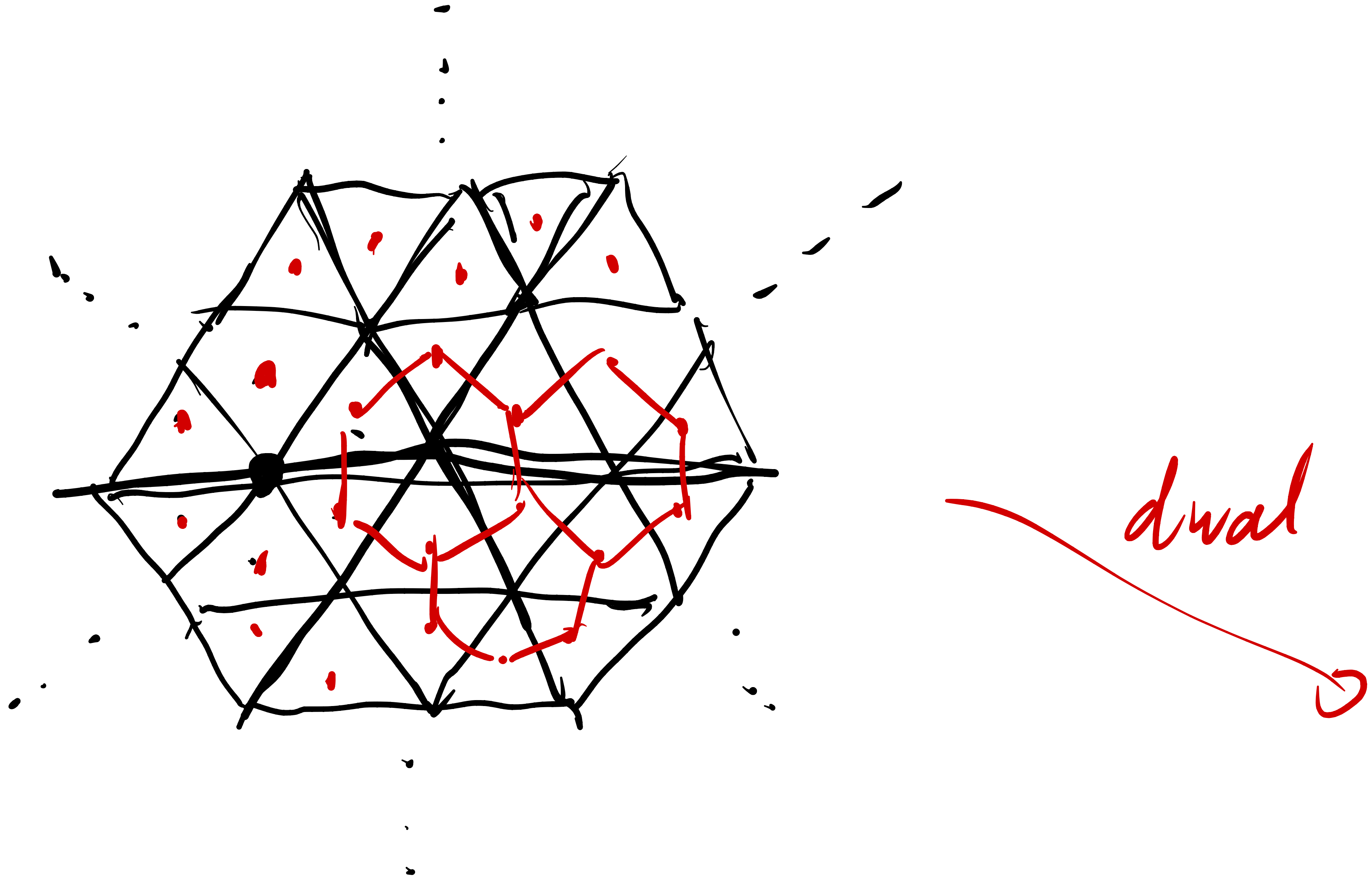
dual →



valence of 4

of sides is 4

swap # of sides & valence of vertices



Represents any regular polygon?

1 number to specify sides

1 number for scale

Hexagons

6 sides per face

valence of 3

Triangles.

3 sides per triangle

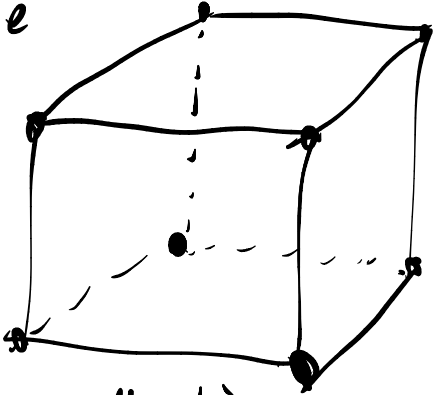
valence is 6

Regular Planar Tilings

Platonic Solids → 3D

regular & convex
& congruent sides
& angles

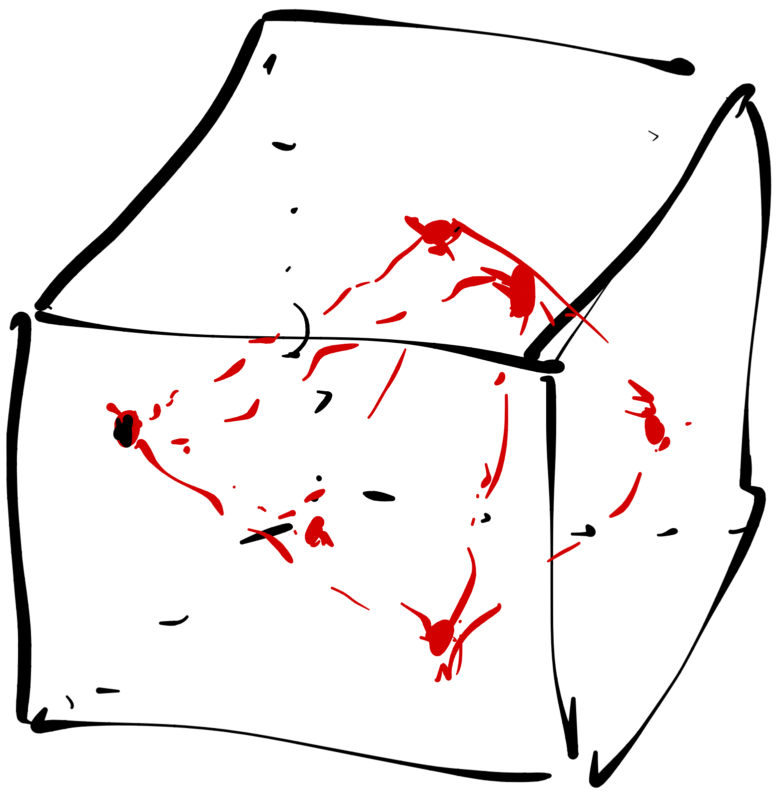
Cube



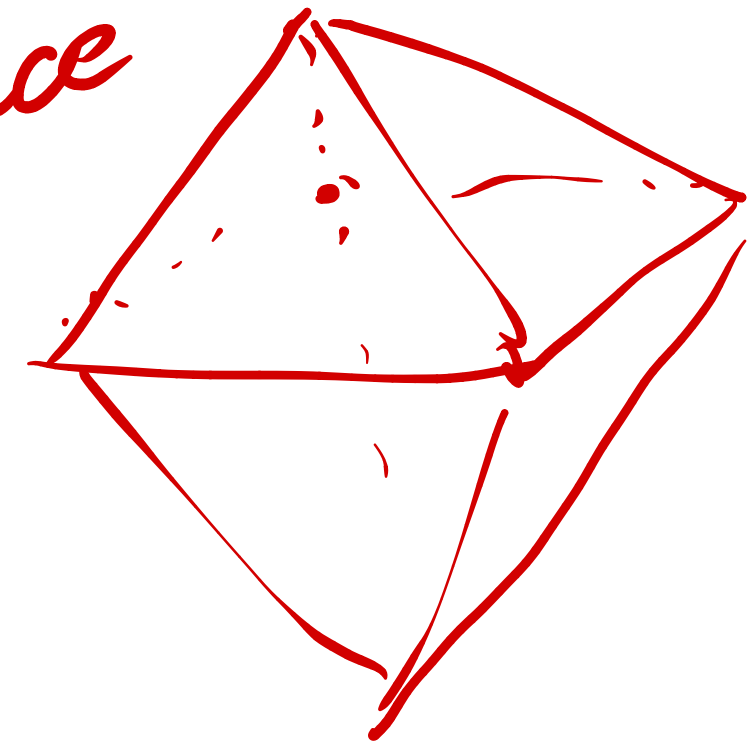
fill space
⇕
tiling in 3D

$(1, 1, 1)$
$(-1, -1, -1)$
$(1, -1, -1)$
$\vdots$

	Vertices	Faces	Edges
cube	8	6	12



Not fill space



Octahedron

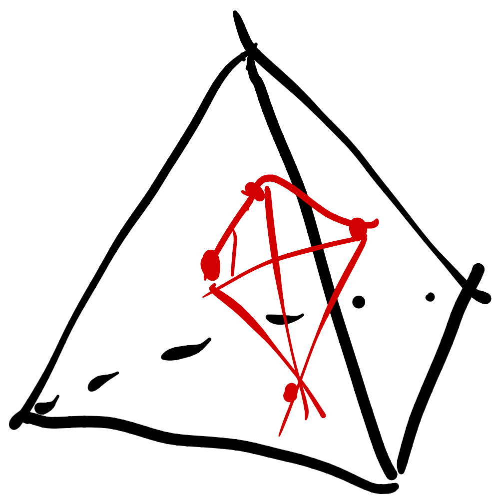
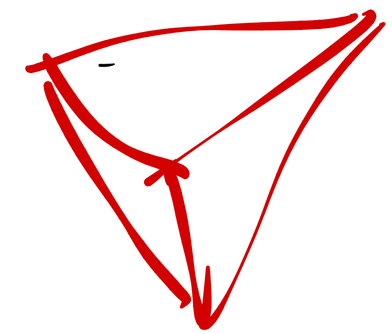
$(1,0,0)$
$(-1,0,0)$

V	F	Edges
8	6	12
6	8	12

dual

cube
octahedron

Tetrahedron

 $\xrightarrow{\text{dual}}$ 

can't fill space

tetrahedrons
+
octahedrons
fill space

Name	V	F	E
cubes	8	6	12
octahedrons	6	8	12
tetrahedrons	4	4	6
icosahedrons	12	20	30
dodecahedrons	20	12	30

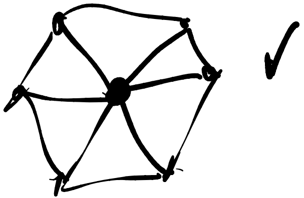
Platonic Solids

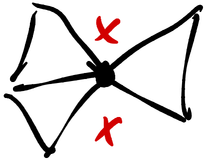
Polyhedrons : - surfaces composed of polygons
- polygons made of vertices & faces

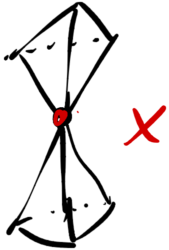
- solids
- approximate smooth surfaces
↳ rendering lets us smooth apperance

Manifold → vertices
→ objects

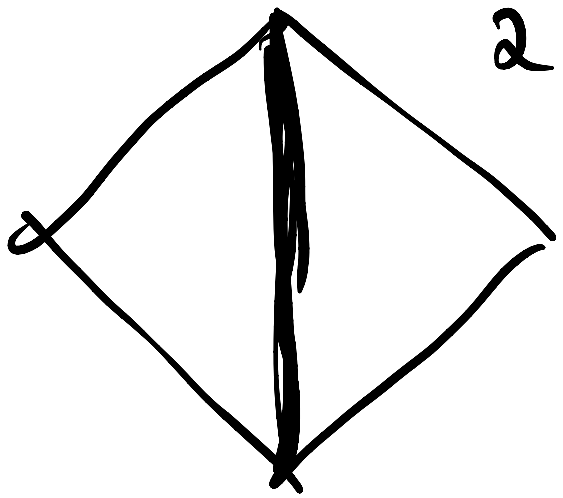
Manifold vertex: surface arounds it looks like a plane (possibly bent)



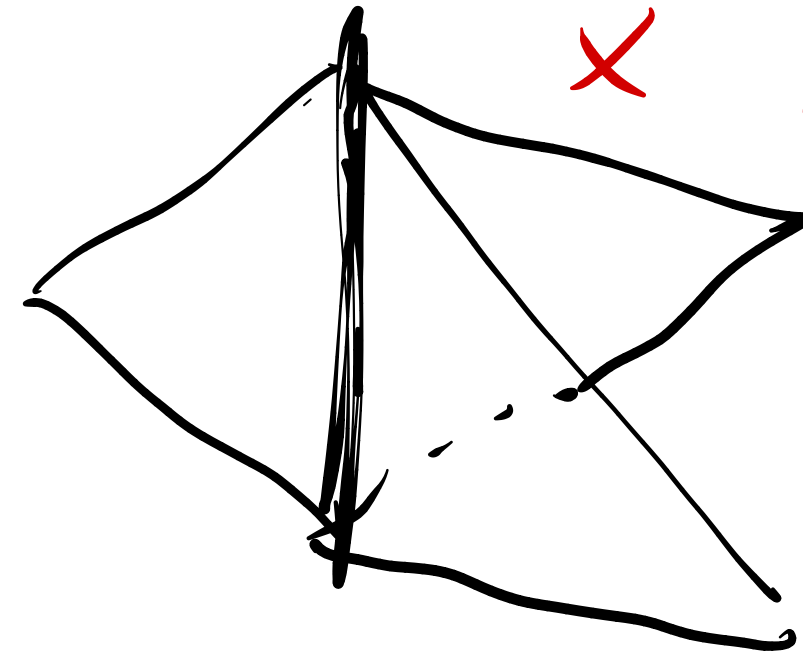




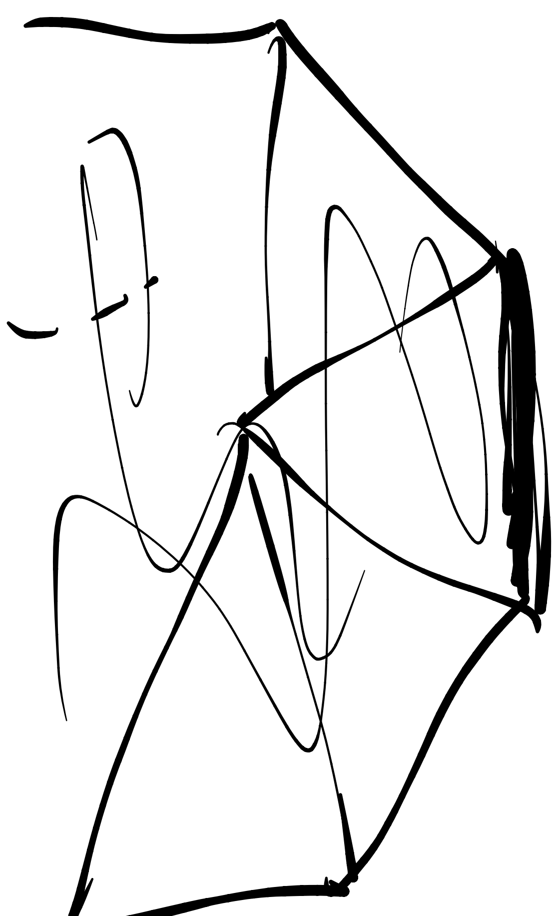
Manifold Edges



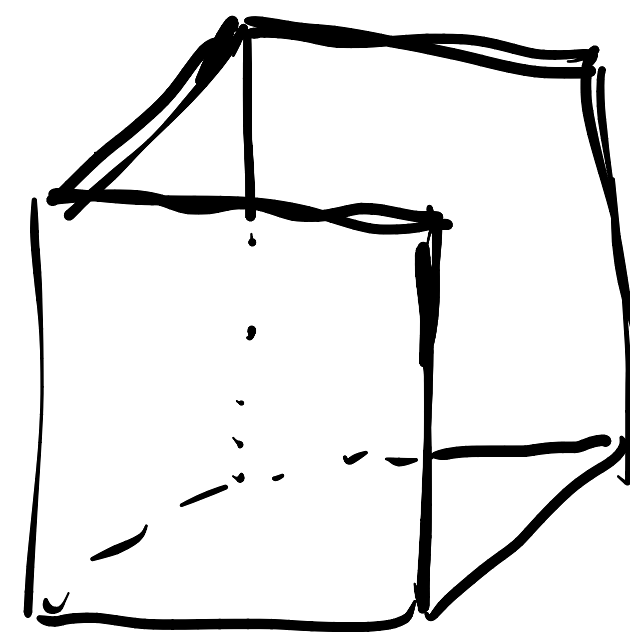
2 polgons connected by the edge



× 3 polygons



boundary edge

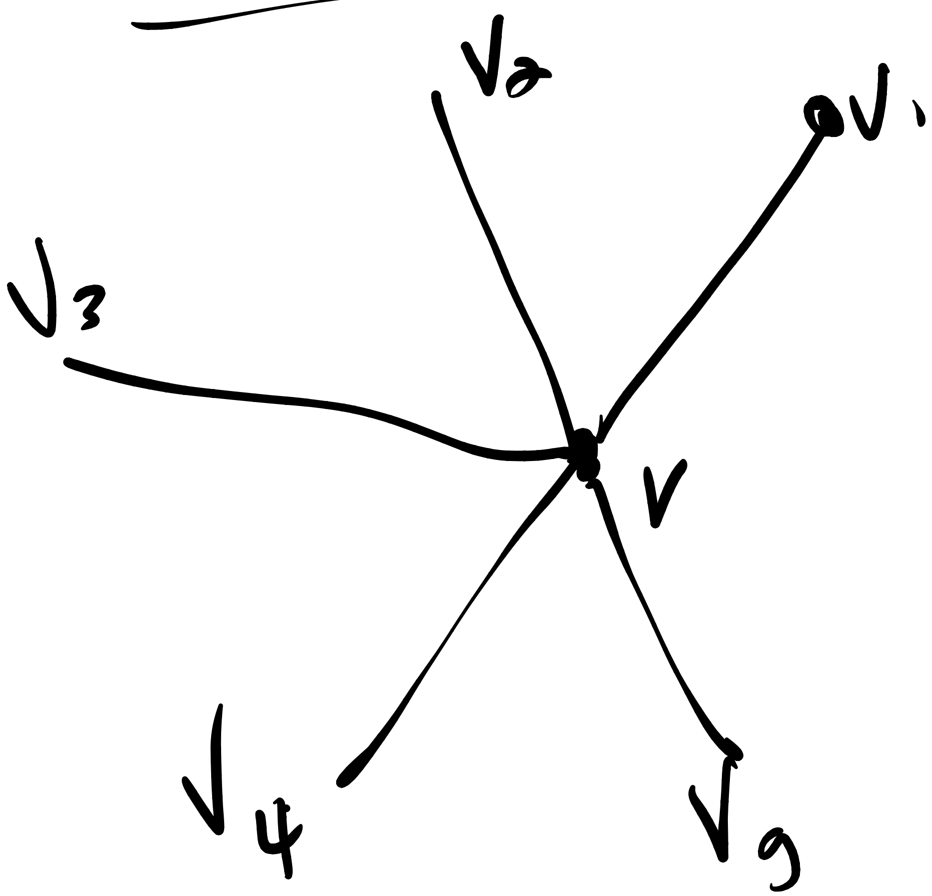


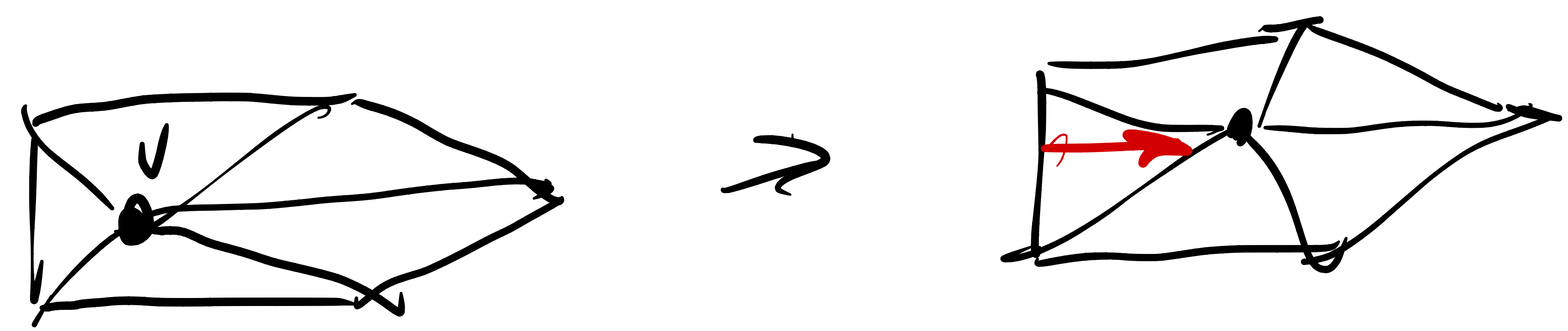
Some operations on polyhedra:

- Laplacian smoothing
- Face subdivision
- Triangulation

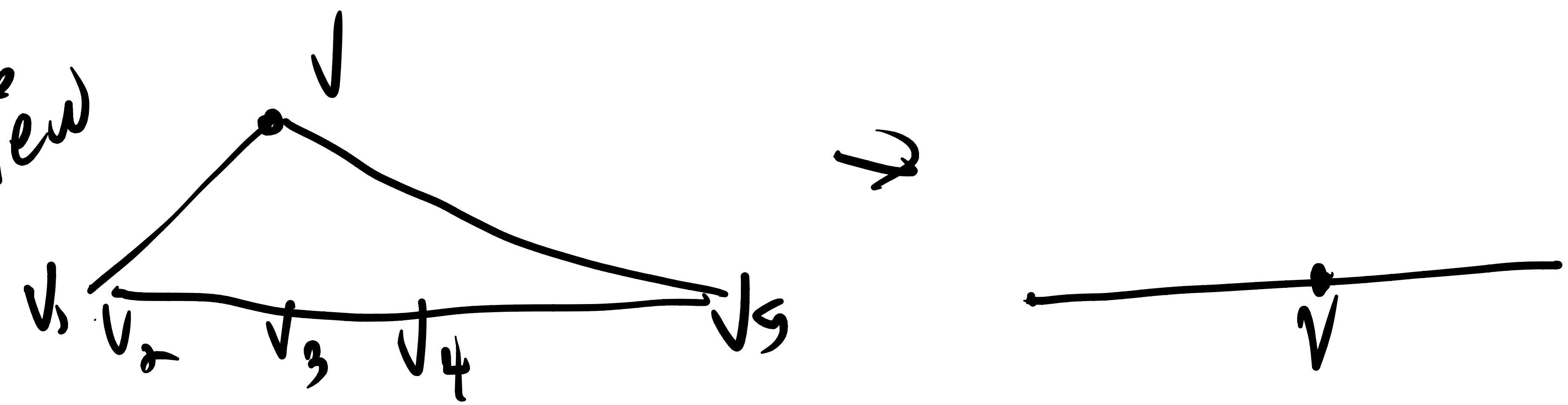
Laplacian Smoothing

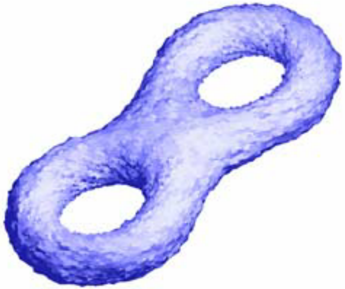
$$\text{new } v = \frac{1}{5}(v_1 + v_2 + v_3 + v_4 + v_5)$$

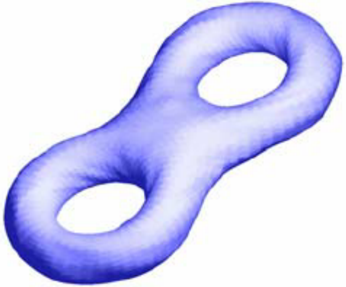


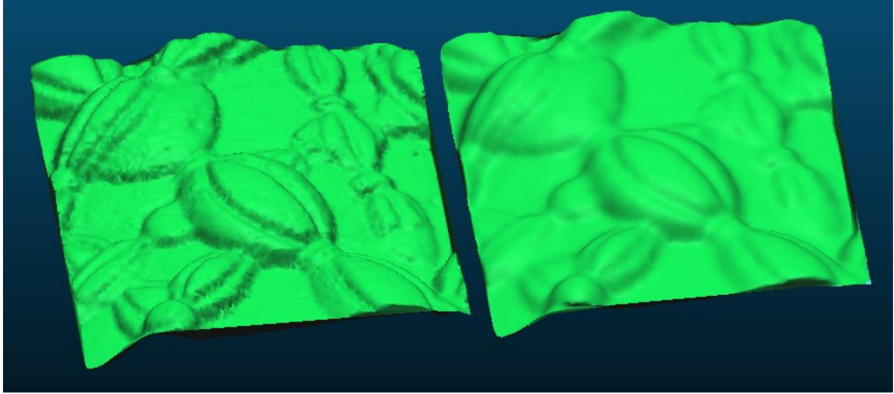


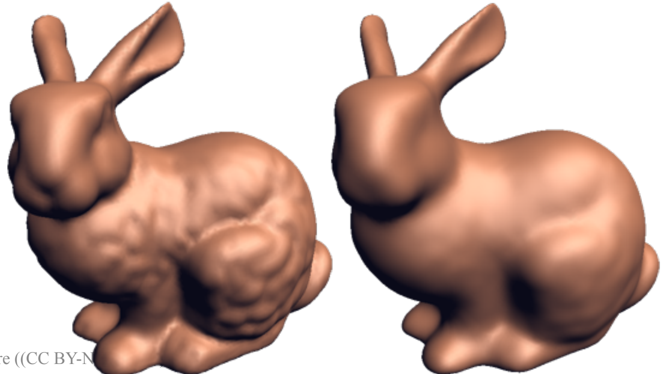
side view



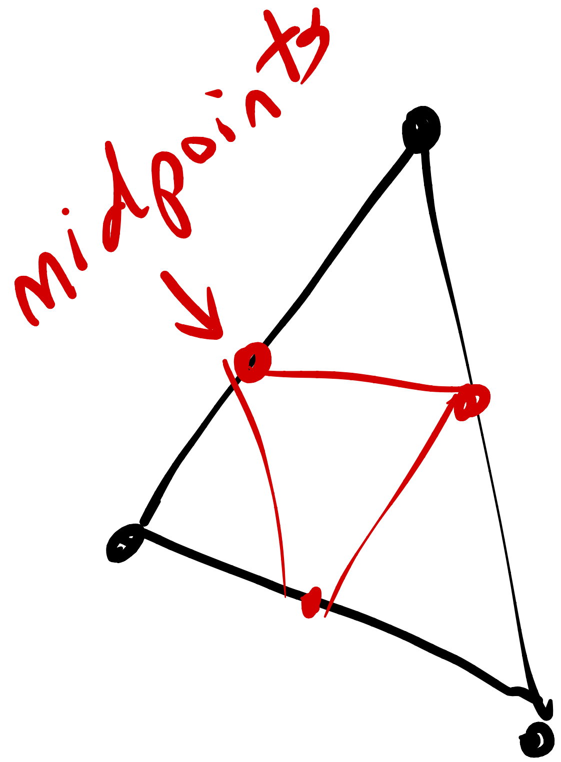


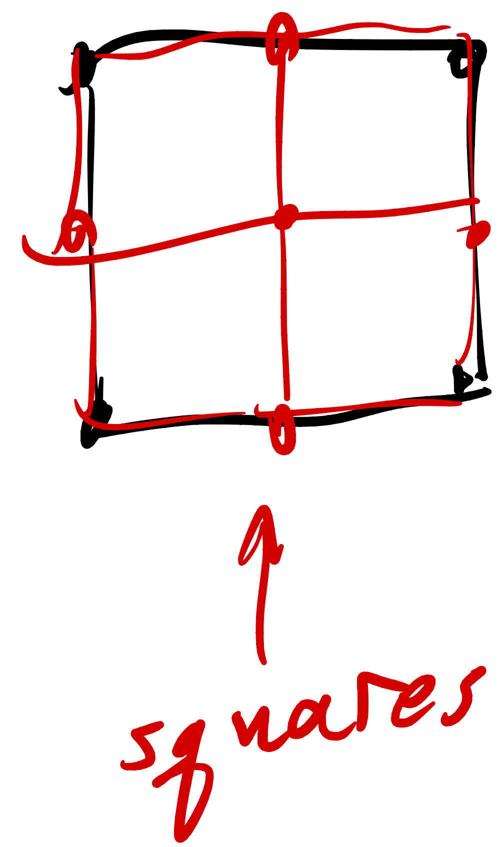


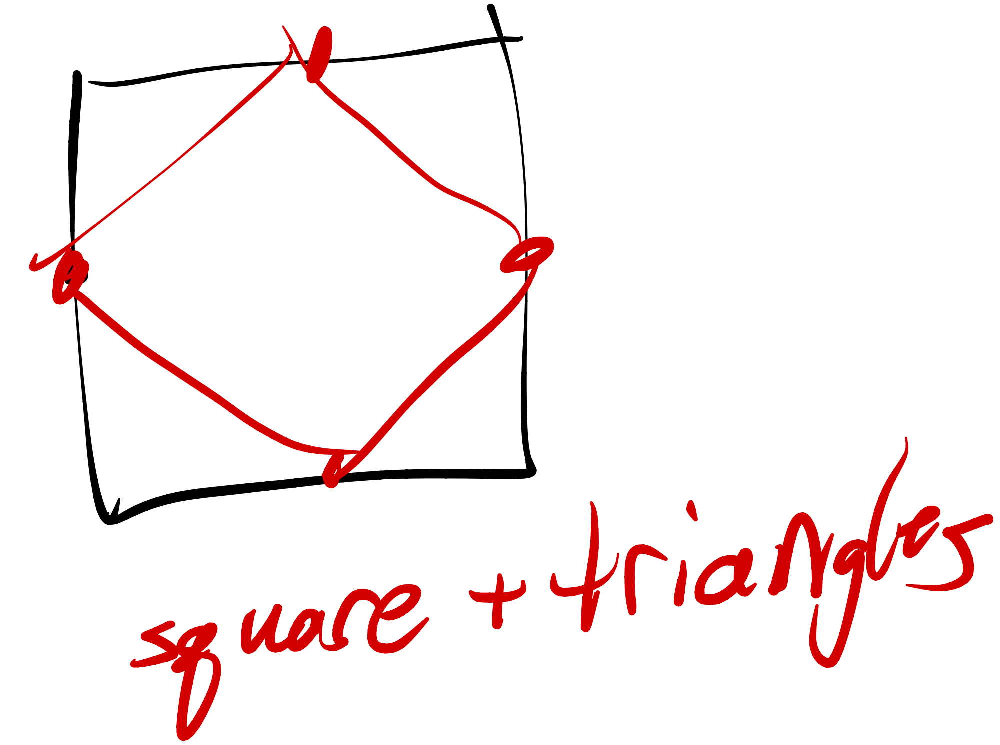




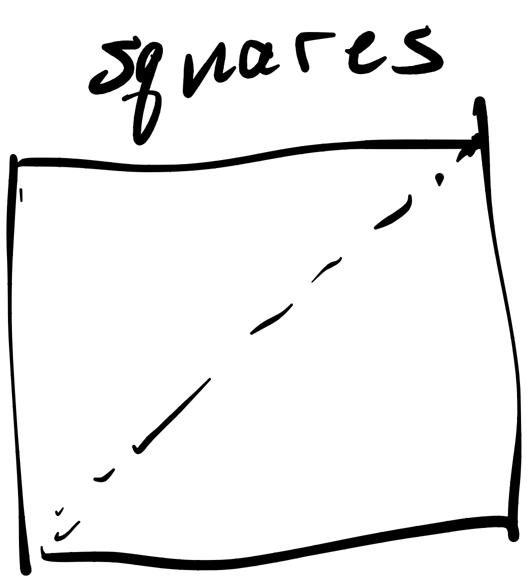
Face subdivision

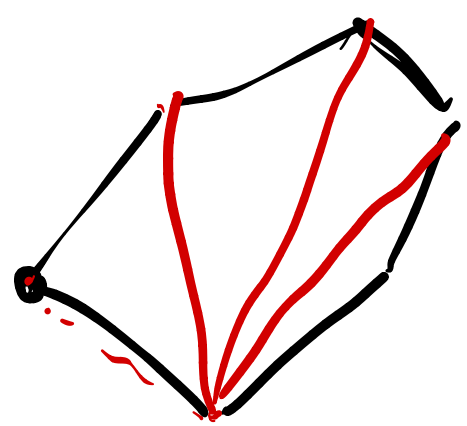


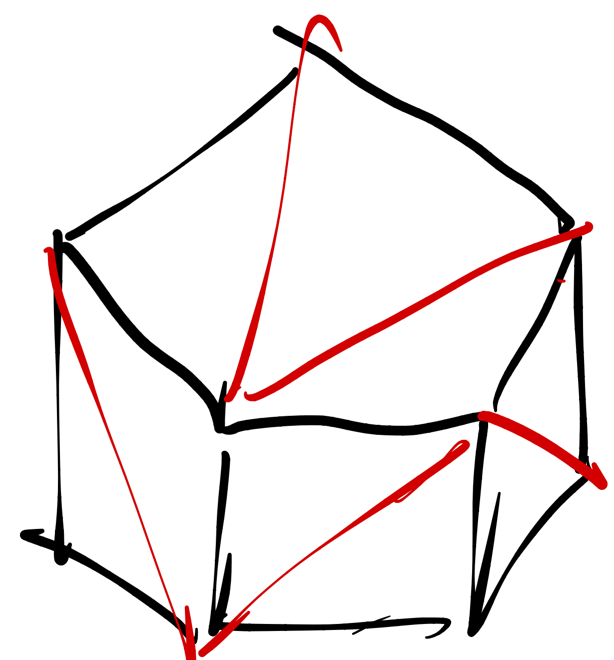




Triangulation

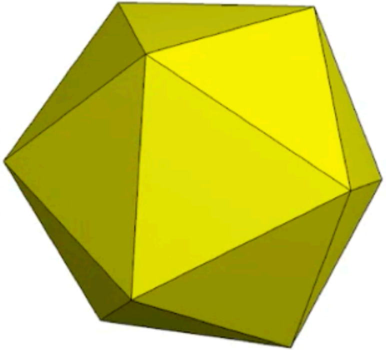




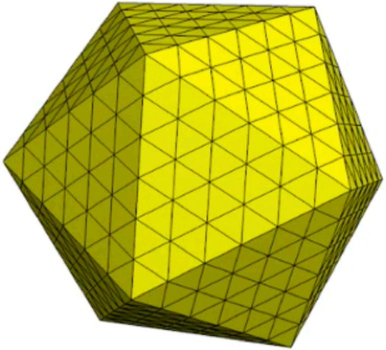


pentagonal prism

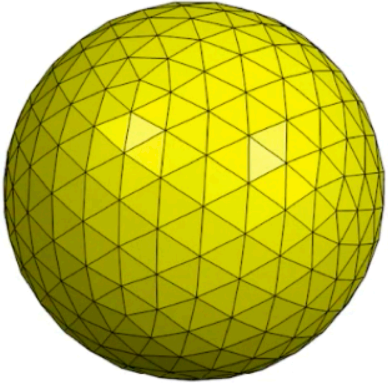
normalize new vertices (assuming unit sphere)



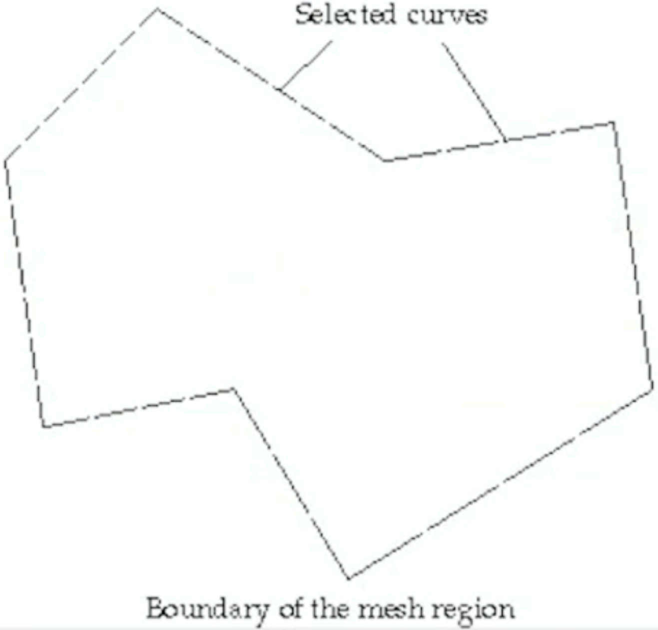
Icosahedron

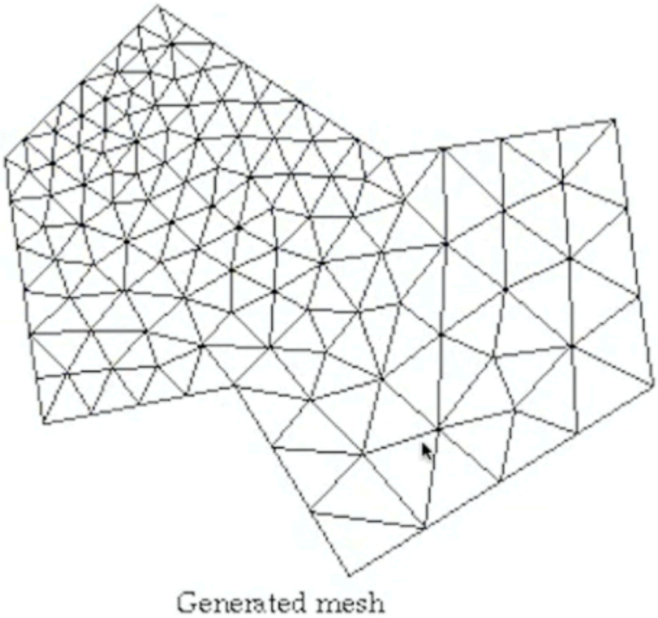


6-frequency subdivision



Vertices projected onto sphere


Selected curves
Boundary of the mesh region


Generated mesh

Representing Polyhedra (Meshes)

Polygon Soup each polygon is represented by itself

```
class Face {
  float x1, y1, z1;
  float x2, y2, z2;
  float x3, y3, z3;
}
```

polyhedra is a list of faces

```
class polyhedra {
  face[] : faces
}
```

files

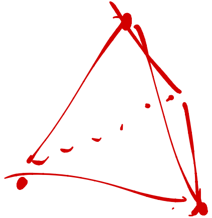
.stl

Shared Vertices / Indexed face set

→ hardware supports soup or indexed set

.obj ← implicit indices

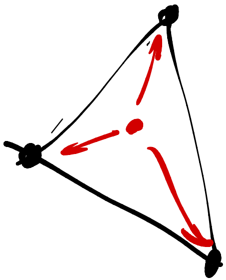
```
v  -1 -1 -1
v   1  1 -1
v   1 -1  1
v  -1  1  1
f   0  1  2
f   0  2  3
f   0  1  3
f   1  2  3
```

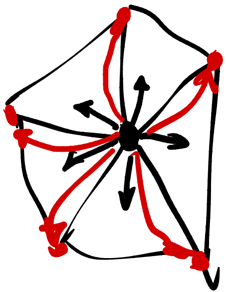


- More compact
- know about shared vertices
  ↳ structure

Vertices, faces, References

. face





```
class vertex {
  float x, y, z;
  float nx, ny, nz;
  vertex[]  vertices;
  face[]  faces;
}
```

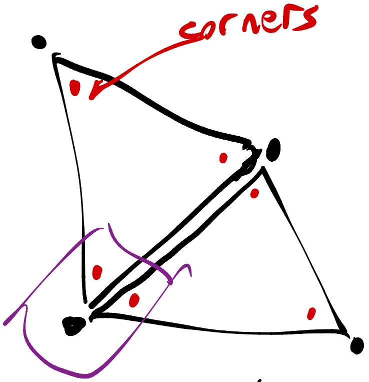
lots of repetition of data

corner representation — Jarek Rossignac

- only works for triangles
- assumes manifold surface → but it can handle holes & boundaries

Geometry Table.

	x	y	z
V0			
V1			
V2			
⋮			



1 vertex
2 corners

(c) Vertex Table

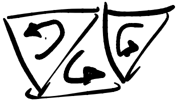
corners are implied

indices into geometry

0	0
1	1
2	2
3	1
4	0
5	3
6	⋮
⋮	

triangle 0 (corners 0–2)

triangle 1 (corners 3–5)



ccw order for every triangle

- triangle num

$$c.t = \underbrace{c \text{ div } 3}_{\text{int}}$$

- next corner

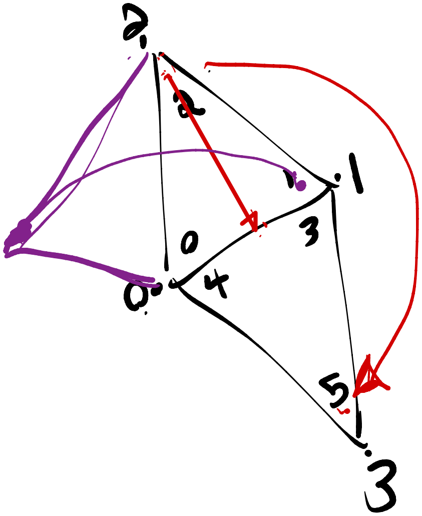
$$c.n = 3 * c.t + (c+1)\%3$$

- previous corner

$$c.p = c.n.n$$

Adjacency using Opposite Table

	V	O
$\Delta 0$	0	7
	1	8
	2	5
$\Delta 1$	1	9
	0	6
	3	2
	⋮	



c.o looks up opposite

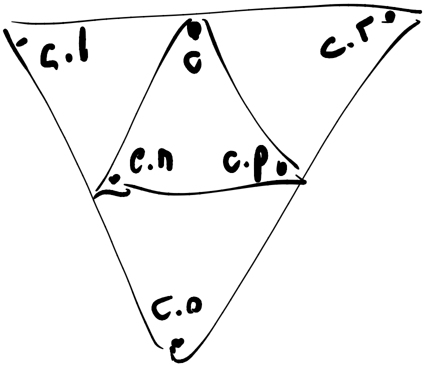
c.o → opposite

right neighbor c.r = c.n.o

left neighbor c.l = c.p.o.

c.s = c.n.o.n

swing



Subdivision Surfaces

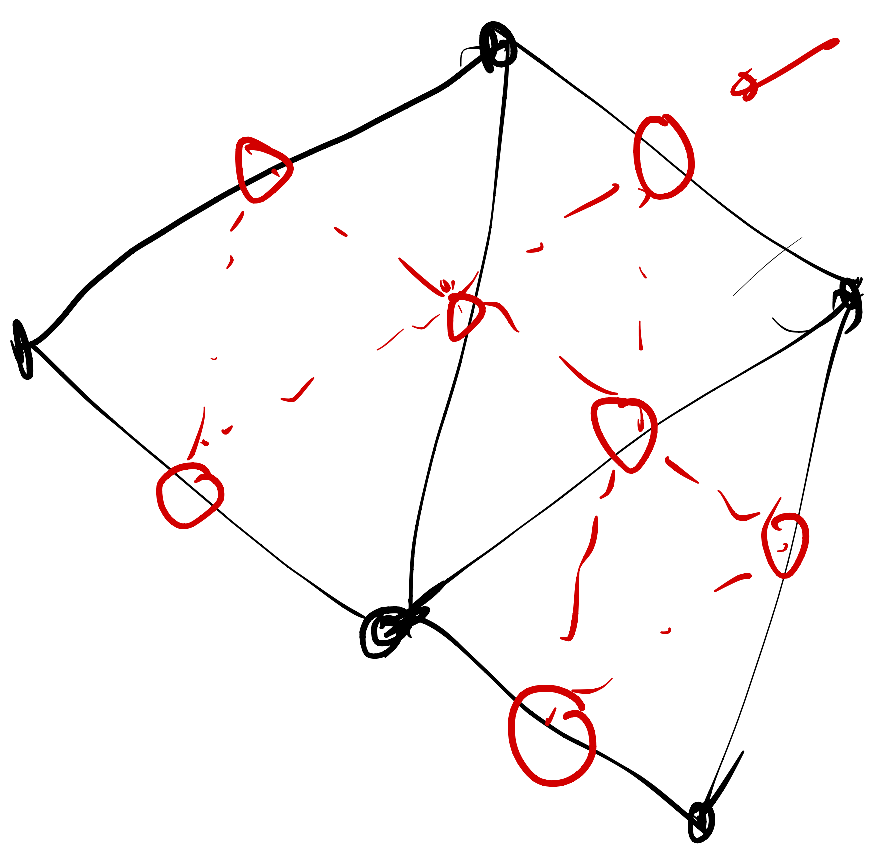
spline/etc (smooth) surfaces have issues

- must subdivide into patches
- continuity between patches

Alternative: Subdivision surfaces

- take triangle. (polygon) mesh convert into a smooth
- simple to program

Loop scheme (Triangles) → well with corner representation



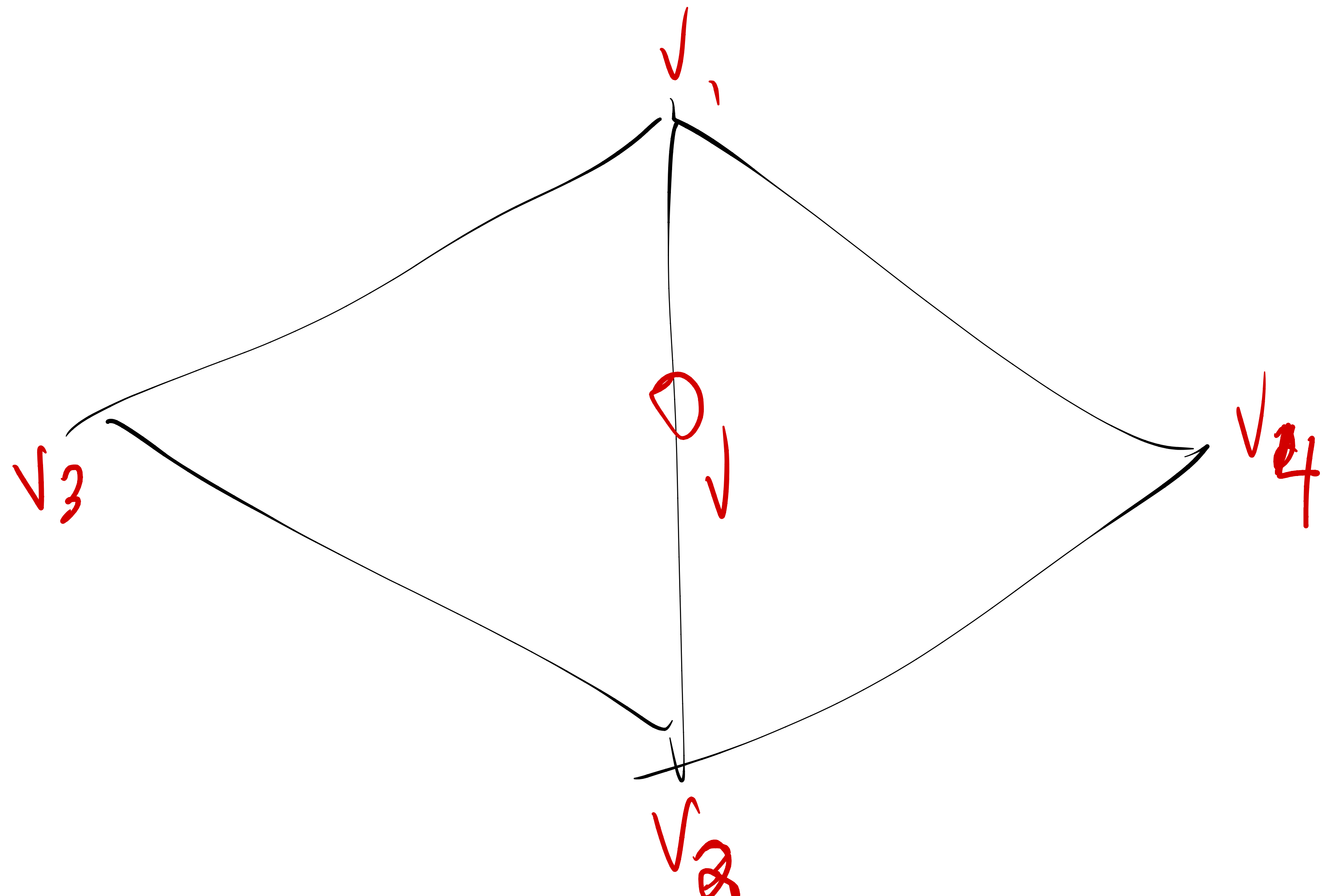
subdivision:

1) compute location new vertices

2) Move old vetices.

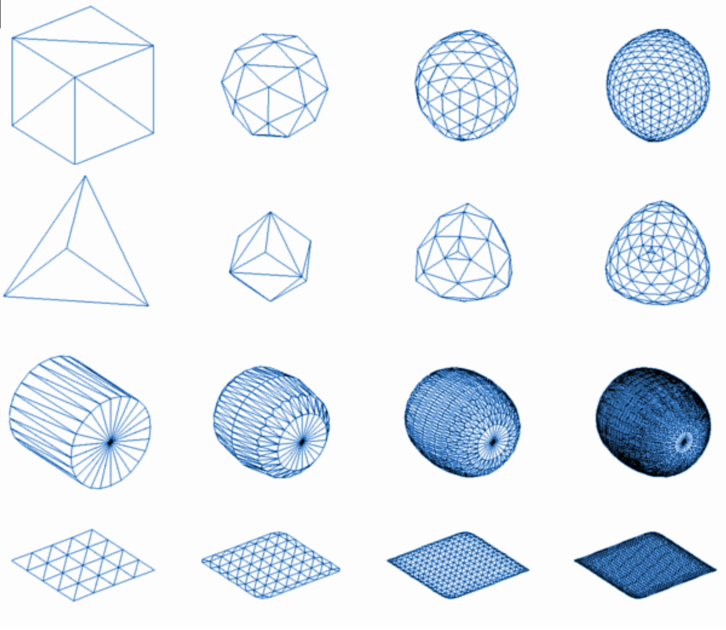
3) Make new triangles

smoothing

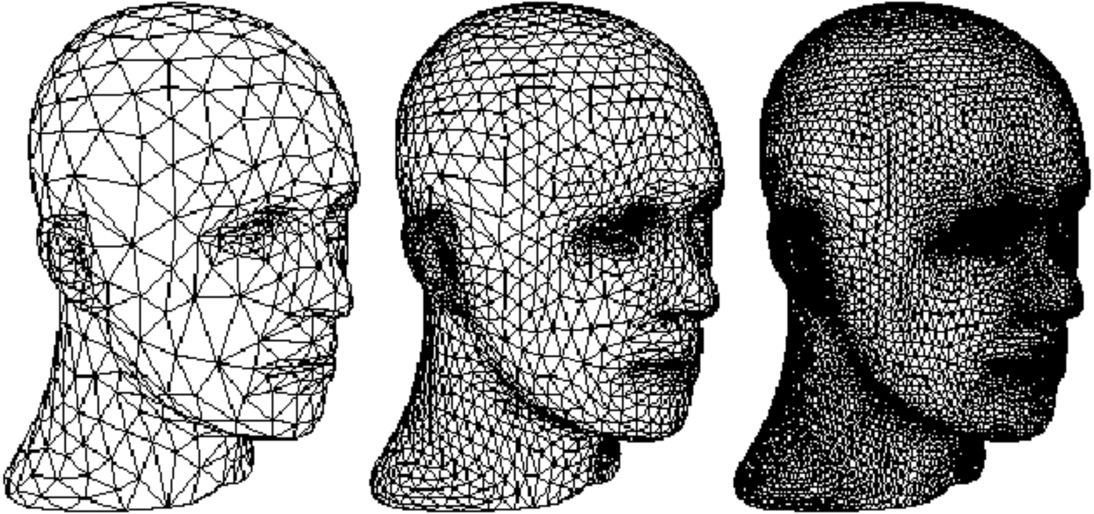


$$v = \frac{3}{8}(v_1 + v_2) + \frac{1}{8}(v_3 + v_4)$$

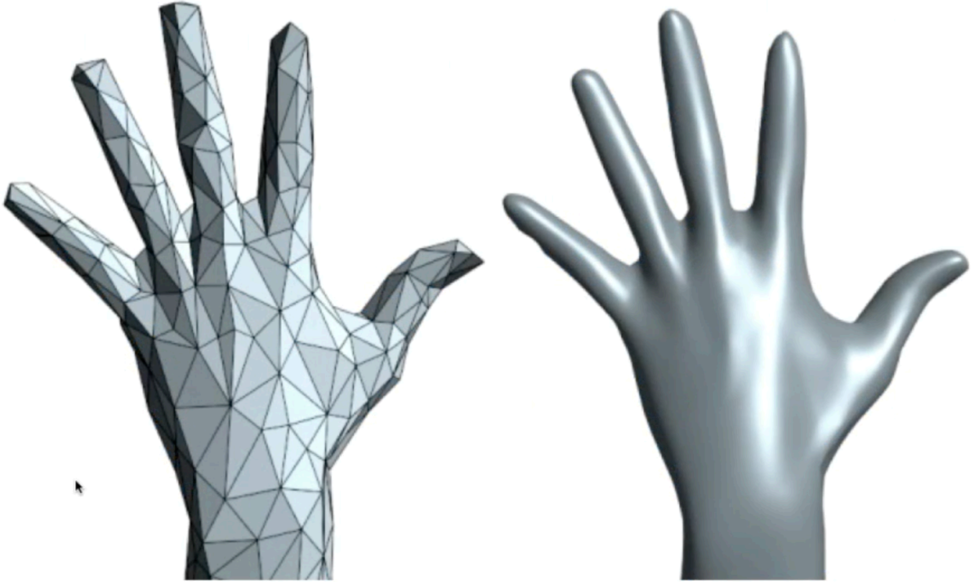
Loop Sub



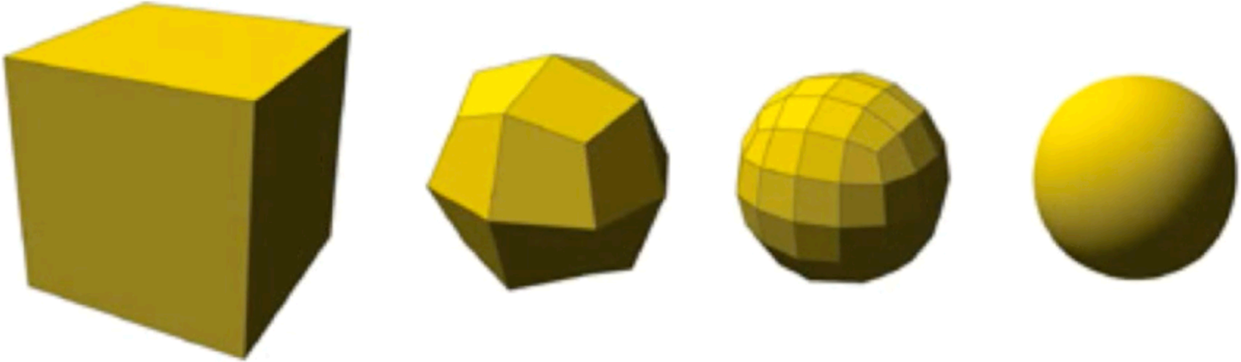
Loop Subdivision



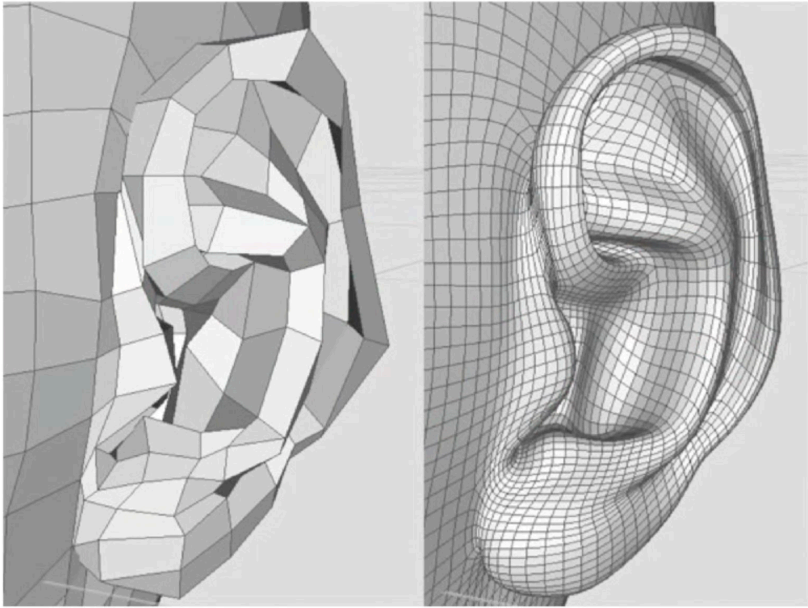
Loop Subdivision



Catmull-Clark Subdivision



Catmull-Clark Subdivision



Catmull-Clark Subdivision: Sharp Edges