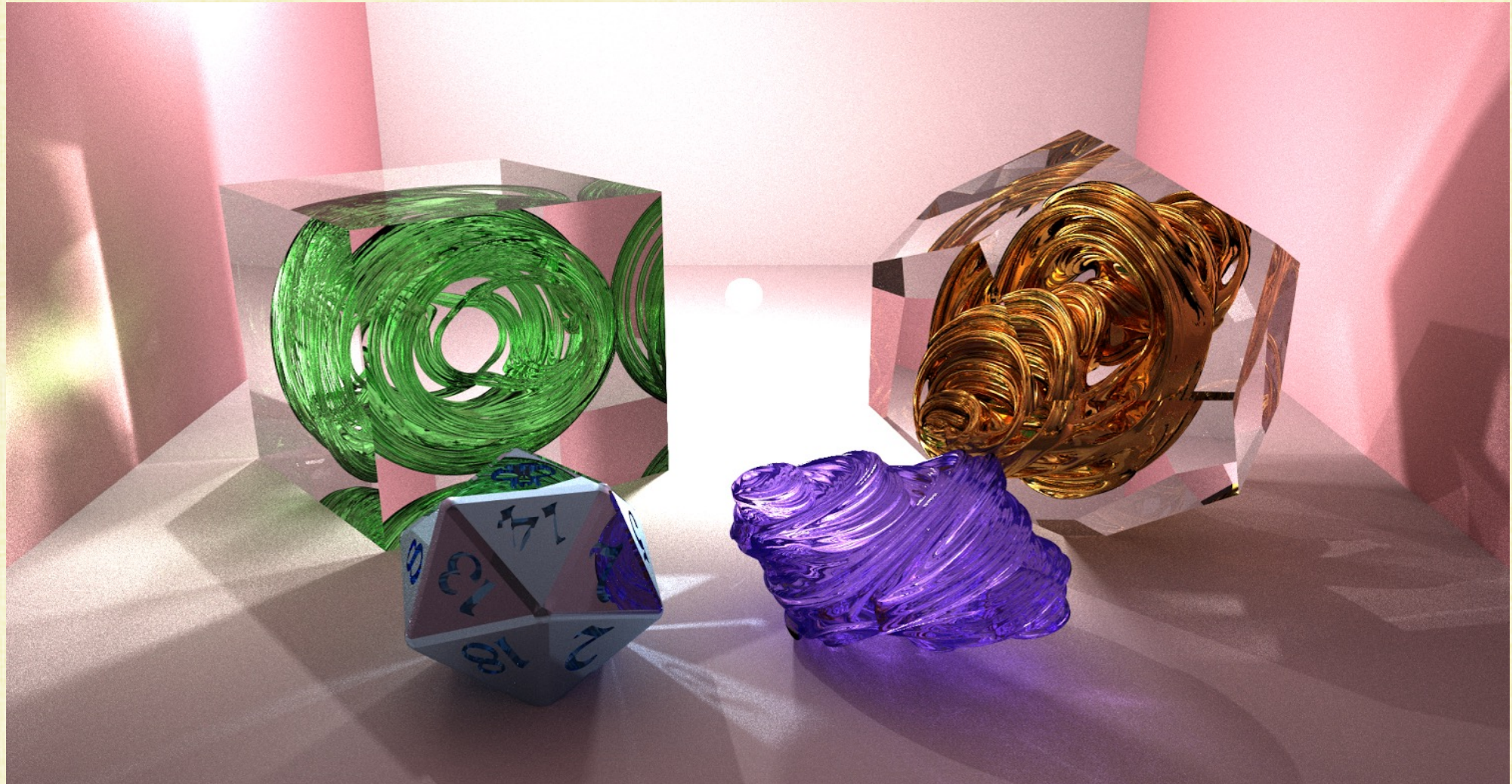
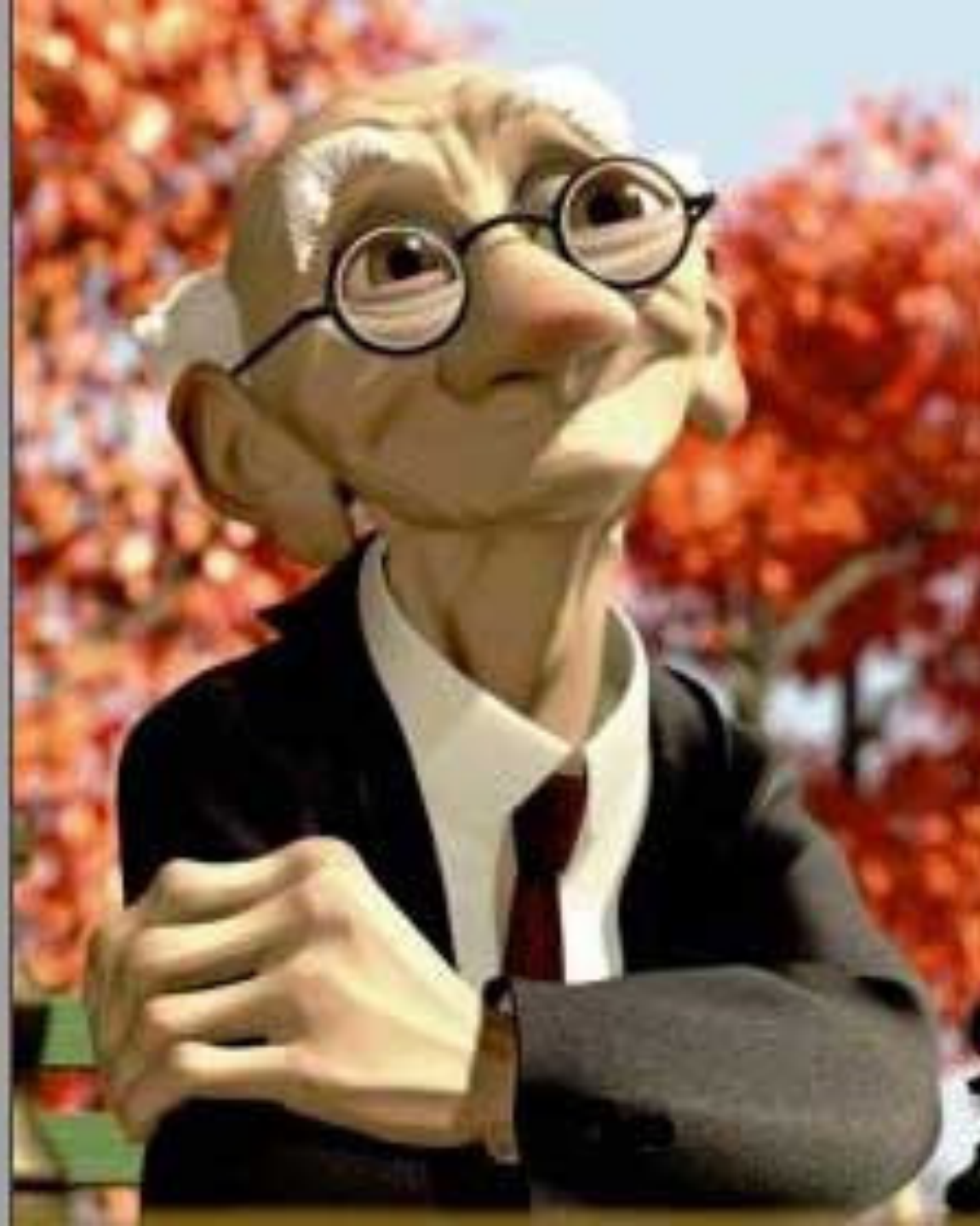
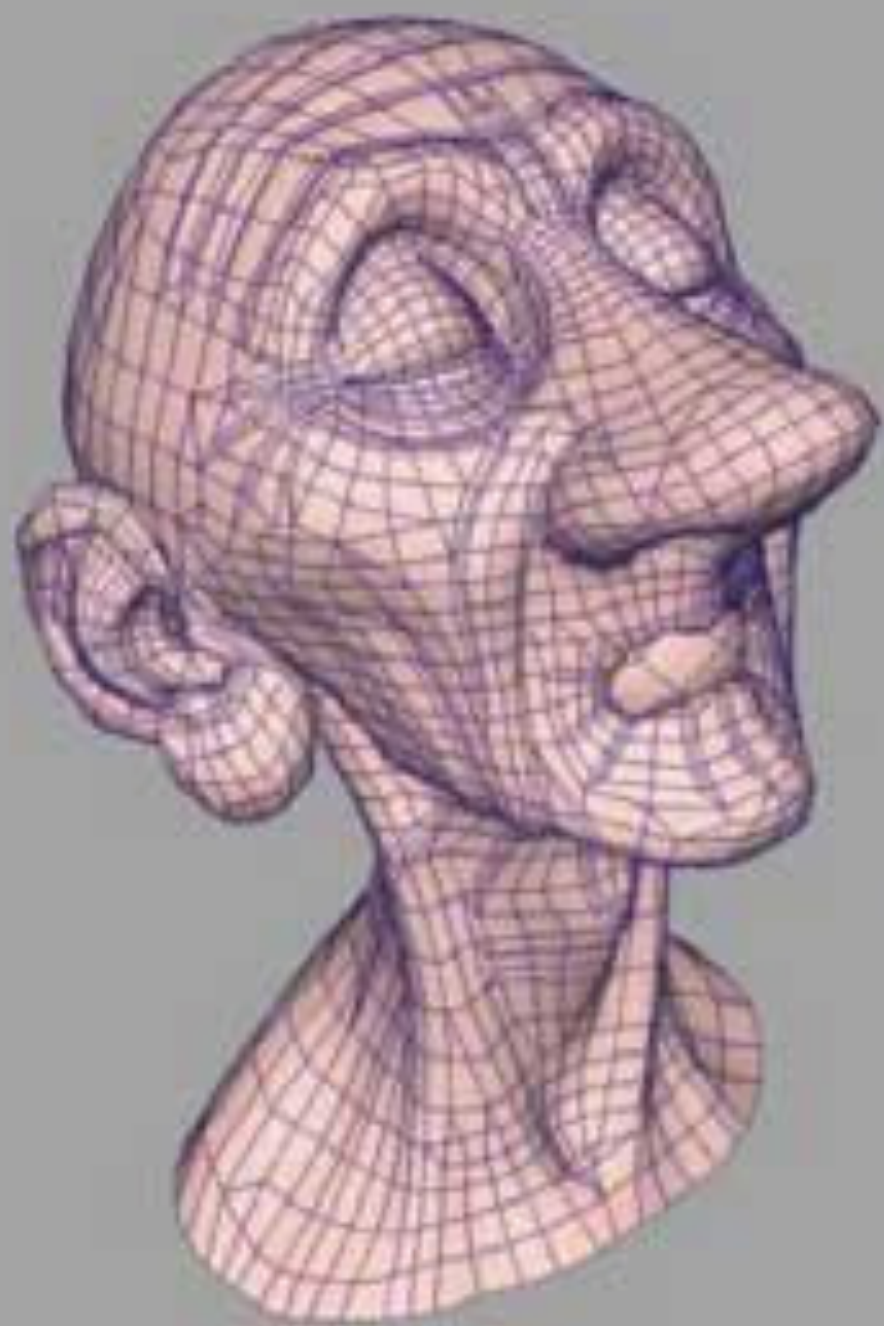


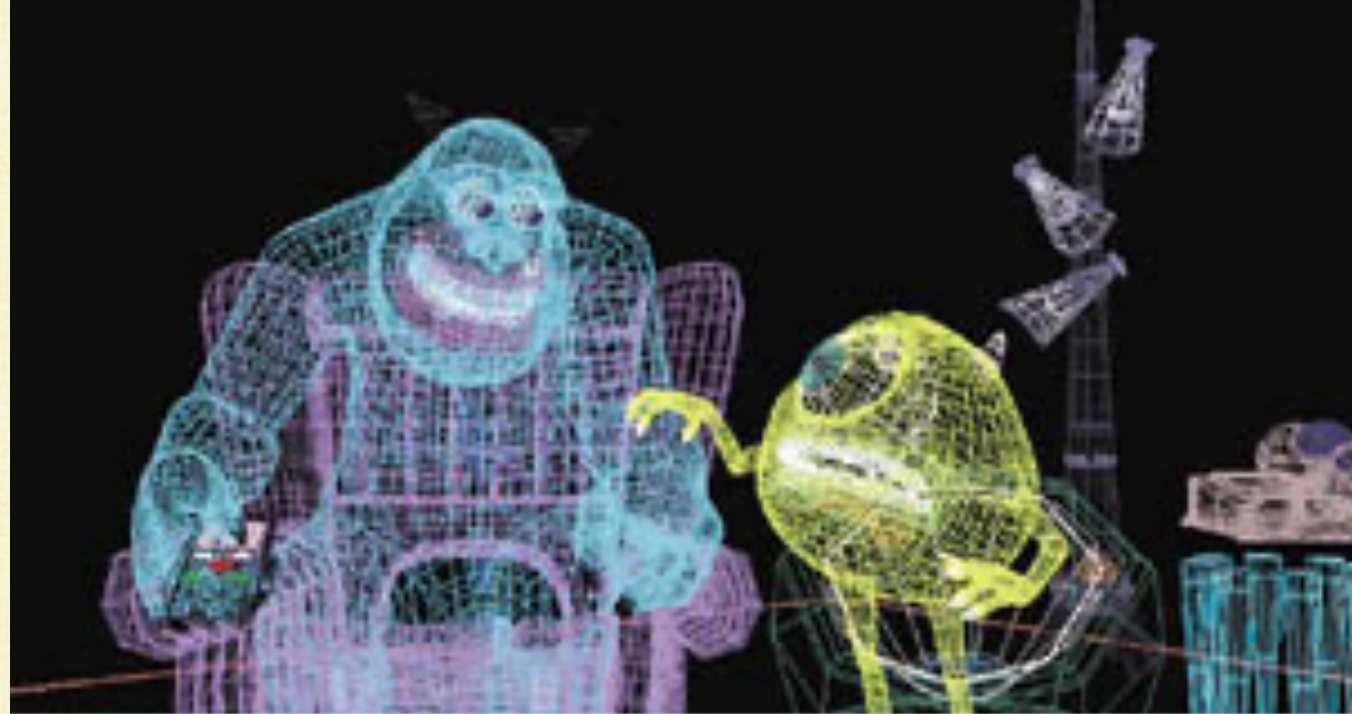
# Geometric Modeling

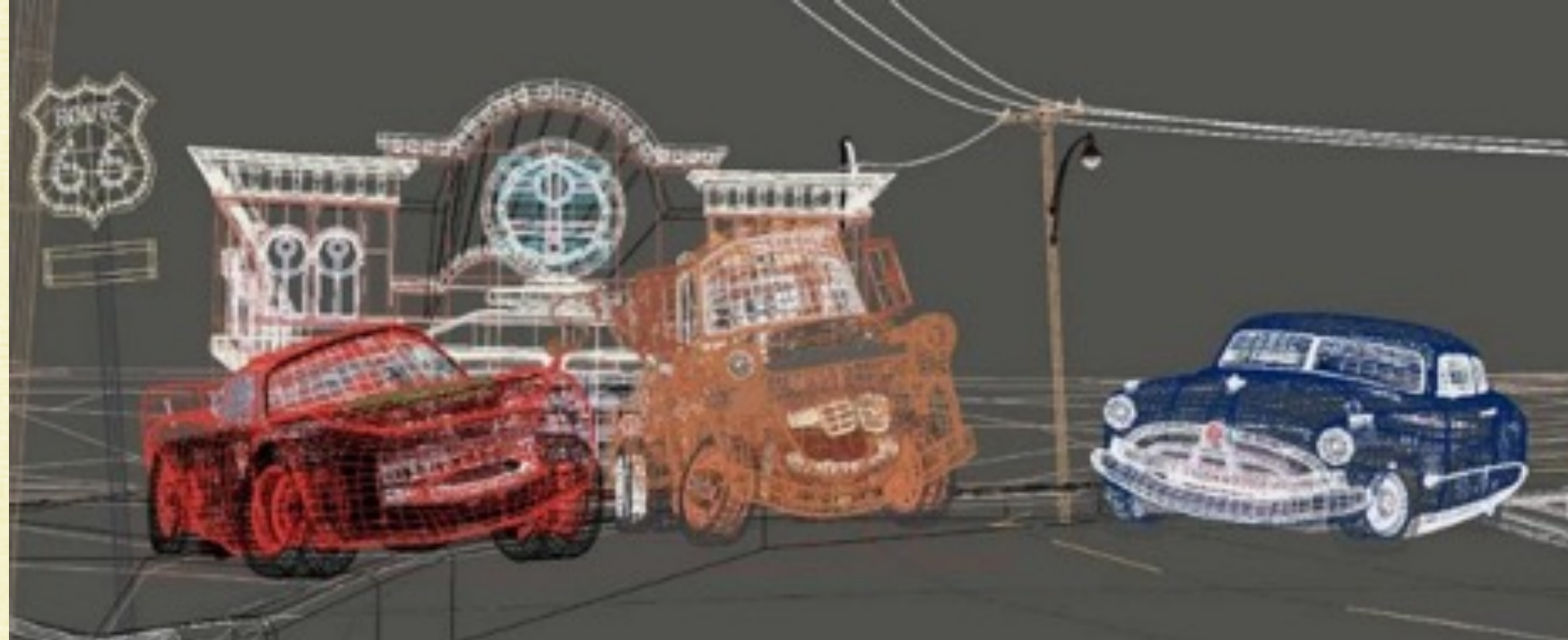


# Examples



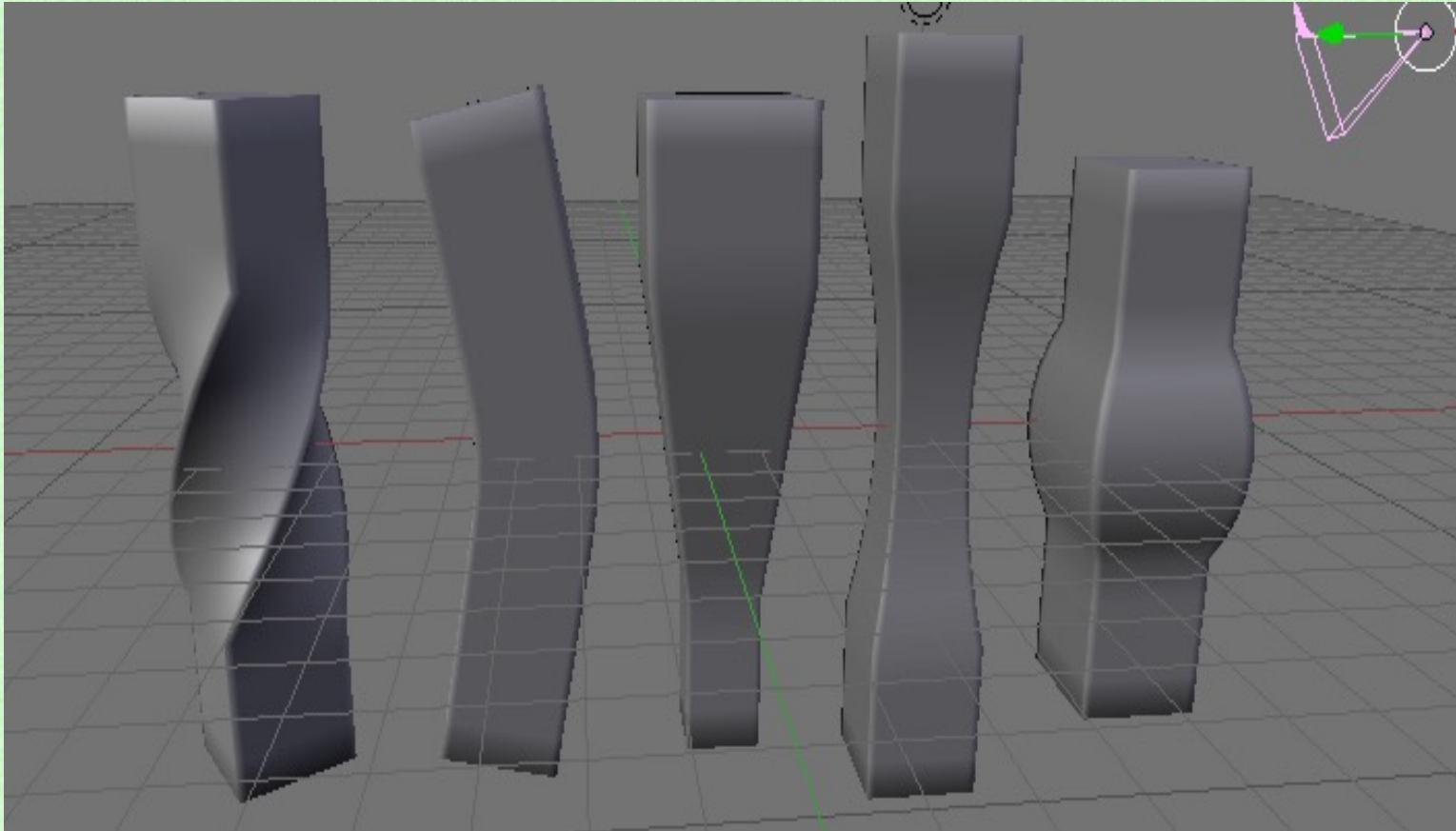






# Mesh Editing

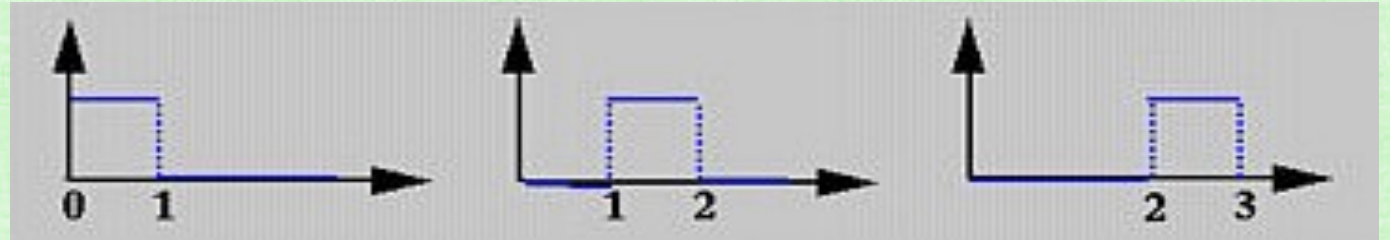
- Manipulate a few control/mesh points, while an algorithm procedurally deforms the mesh
  - e.g., twist, bend, stretch, etc.



# Basis Functions

- Knots: node locations  $\{u_0, u_1, \dots, u_m\}$  in parameter space
- Lowest level building blocks are piecewise constant (m of them):

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$



0-th order:

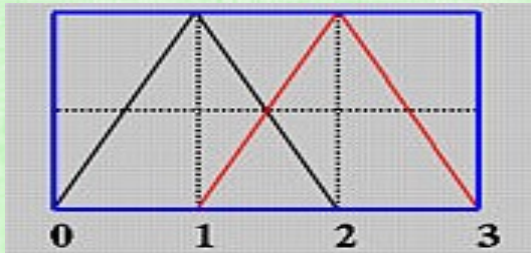
$N_{0,0}$

$N_{1,0}$

$N_{2,0}$

- Higher order building blocks are defined recursively: the i-th basis function of order j is

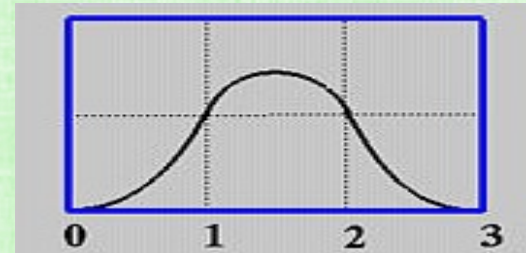
$$N_{i,j}(u) = \frac{u - u_i}{u_{i+j} - u_i} N_{i,j-1}(u) + \frac{u_{i+j+1} - u}{u_{i+j+1} - u_{i+1}} N_{i+1,j-1}(u)$$



1st order:

$N_{0,1}$

$N_{1,1}$



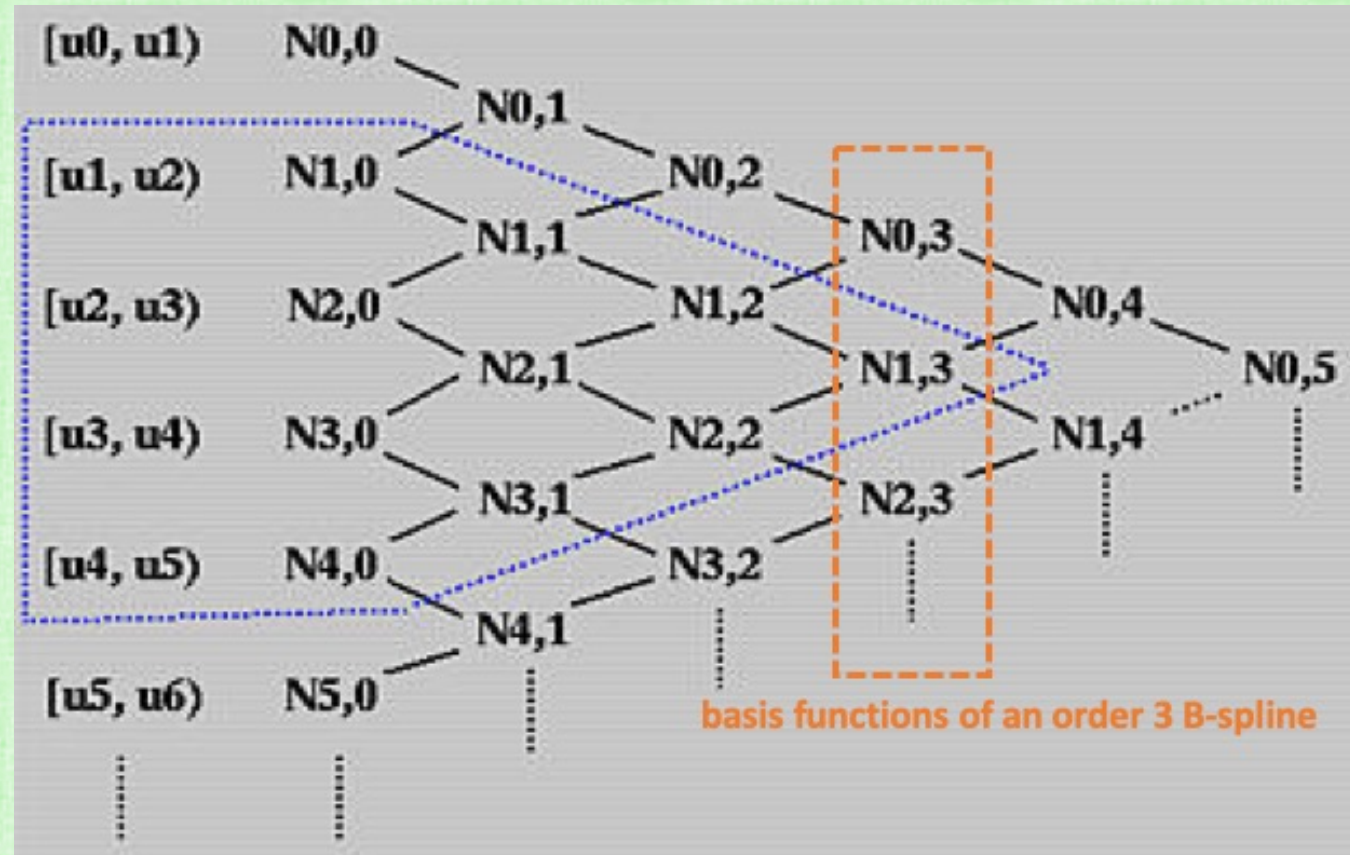
2nd order:

$N_{0,2}$

# B-Splines

- A B-spline of order  $p$  has basis functions:  $N_{i,p}(u)$
- Let  $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$  be control points (in 2D or 3D)
- The B-spline curve is:  $C(u) = \sum_{i=0}^n N_{i,p}(u) \mathbf{P}_i$

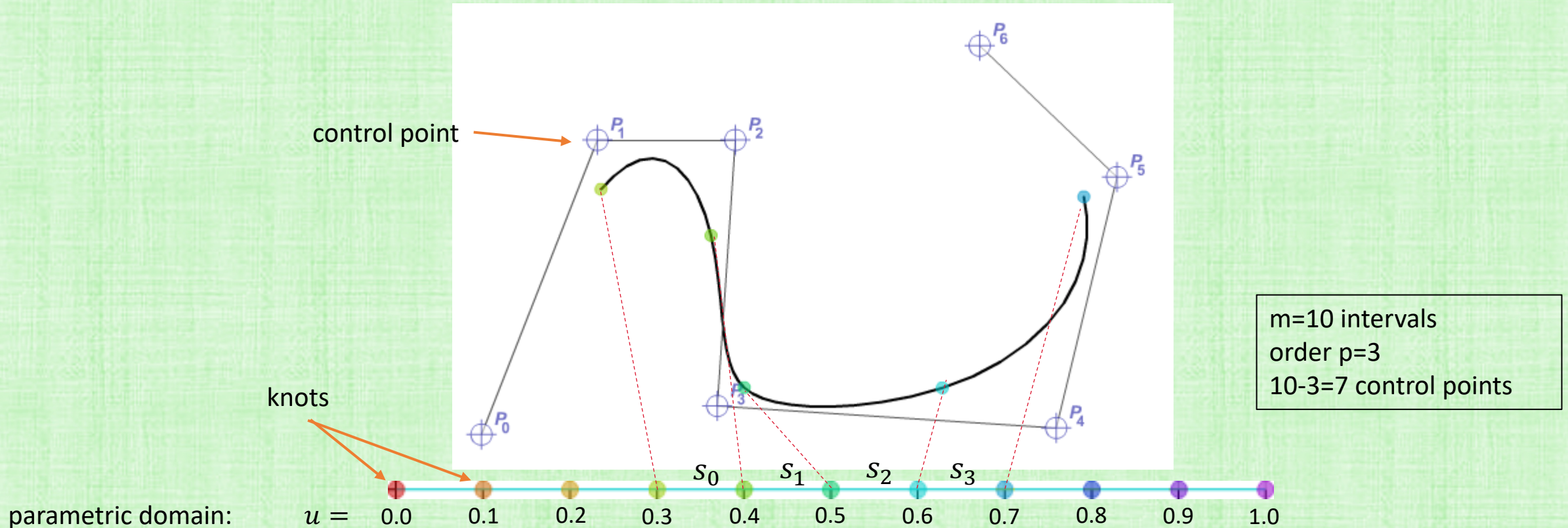
- Given  $m$  intervals, a B-spline of order  $p$  requires  $m-p$  control points  $\mathbf{P}_i$
- E.g.,  $m=6$  intervals with order  $p=3$  requires  $m-p=3$  (orange box) control points





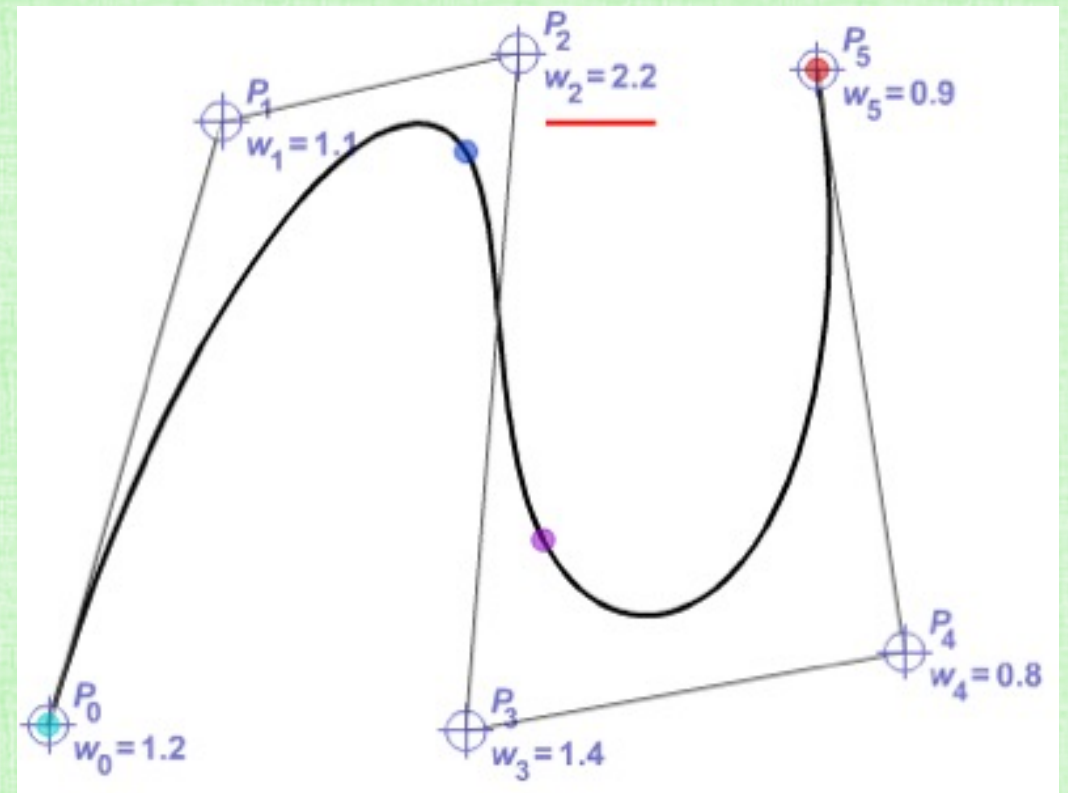
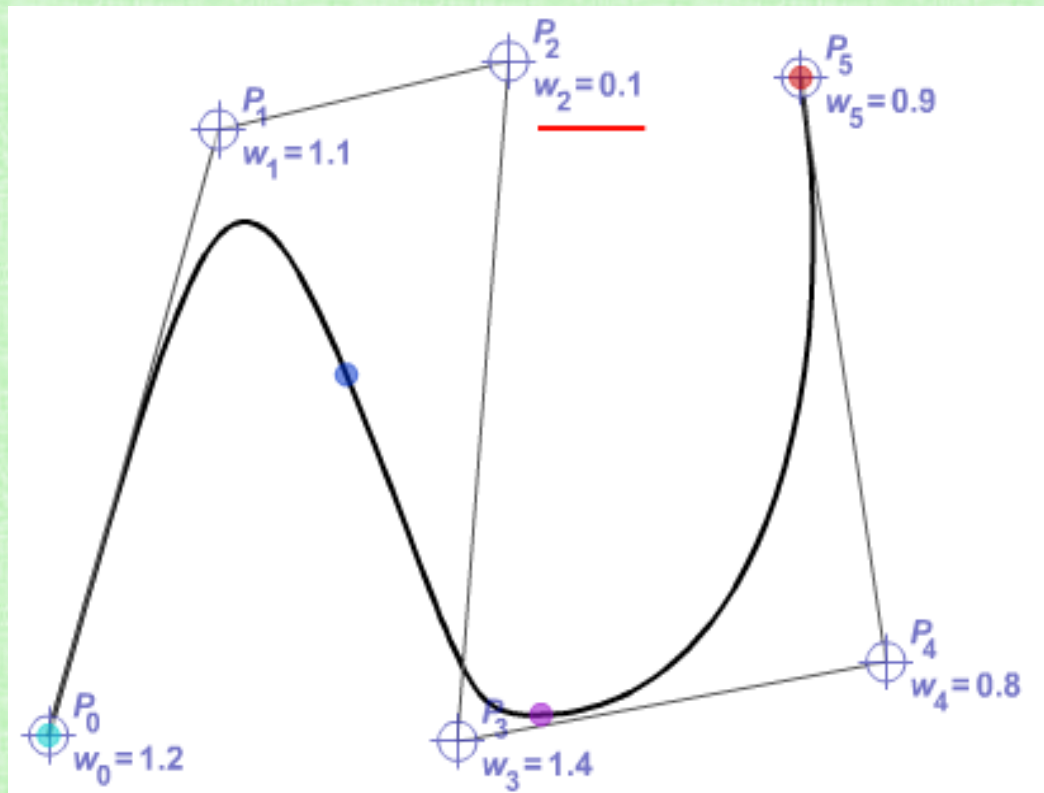
# Cubic B-Spline

- Ignore boundaries (where there are only 1, 2, 3 cubic basis functions  $N_{i,3}$ ):
  - 3 parametric intervals are lost off of each side
  - $[u_3, u_4)$  is the first interval to have a full representation (in the orange box of prior slide):  $N_{0,3}, N_{1,3}, N_{2,3}, N_{3,3}$
- 4 **control points** define the **shape** of each curve segment
  - e.g.,  $s_0 = (.3, .4)$  is controlled  $P_0, P_1, P_2, P_3$



# NURBS Curve

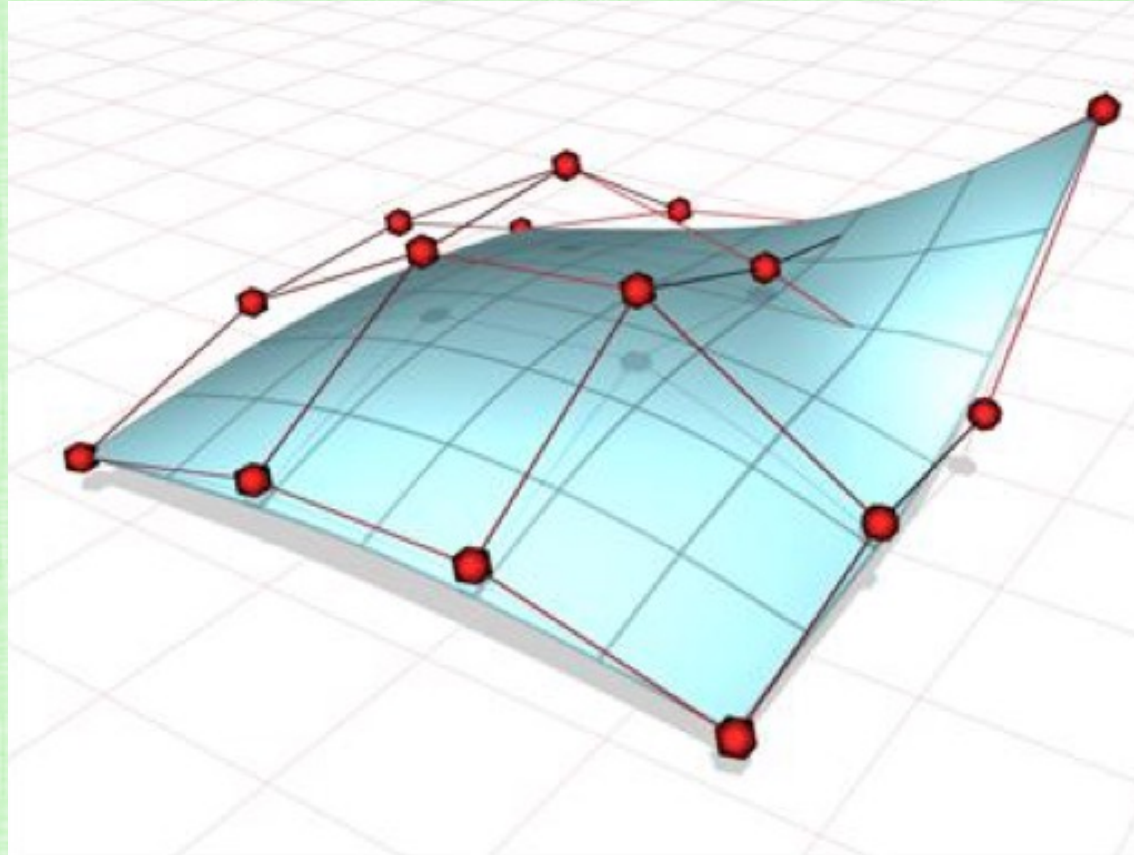
- Non-Uniform Rational B-Spline (NURBS):  $C(u) = \frac{\sum_{i=0}^n N_{i,p}(u)w_i P_i}{\sum_{i=0}^n N_{i,p}(u)w_i}$
- Increasing the weight  $w_i$  pulls the curve closer to  $P_i$
- Decreasing  $w_i$  releases the curve to move farther away from  $P_i$



# NURBS Surface

- Extending the ideas from curves to surfaces:

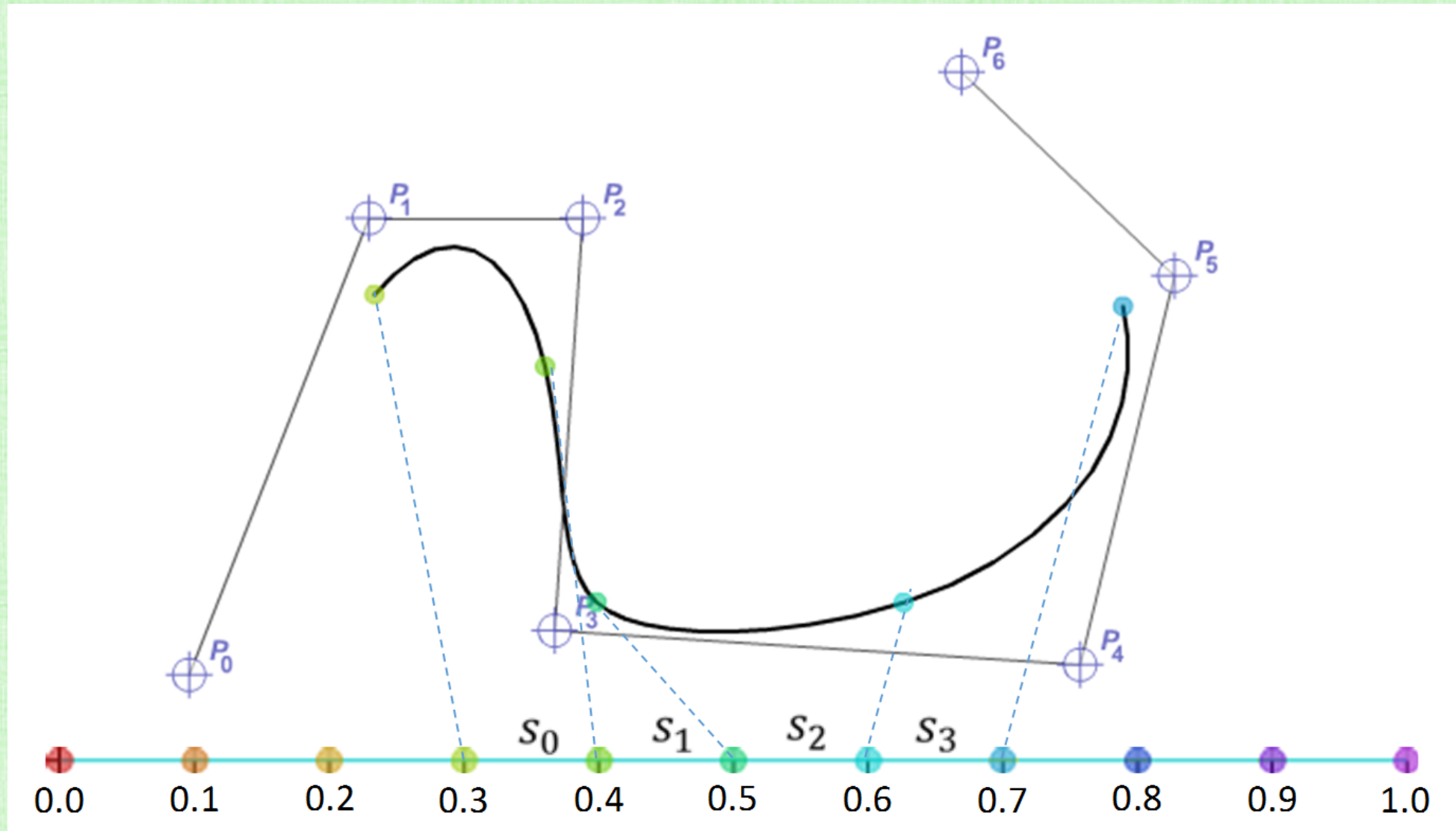
$$S(u, v) = \frac{\sum_{i=0}^m \sum_{j=0}^n N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=0}^m \sum_{j=0}^n N_{i,p}(u) N_{j,q}(v) w_{i,j}}$$



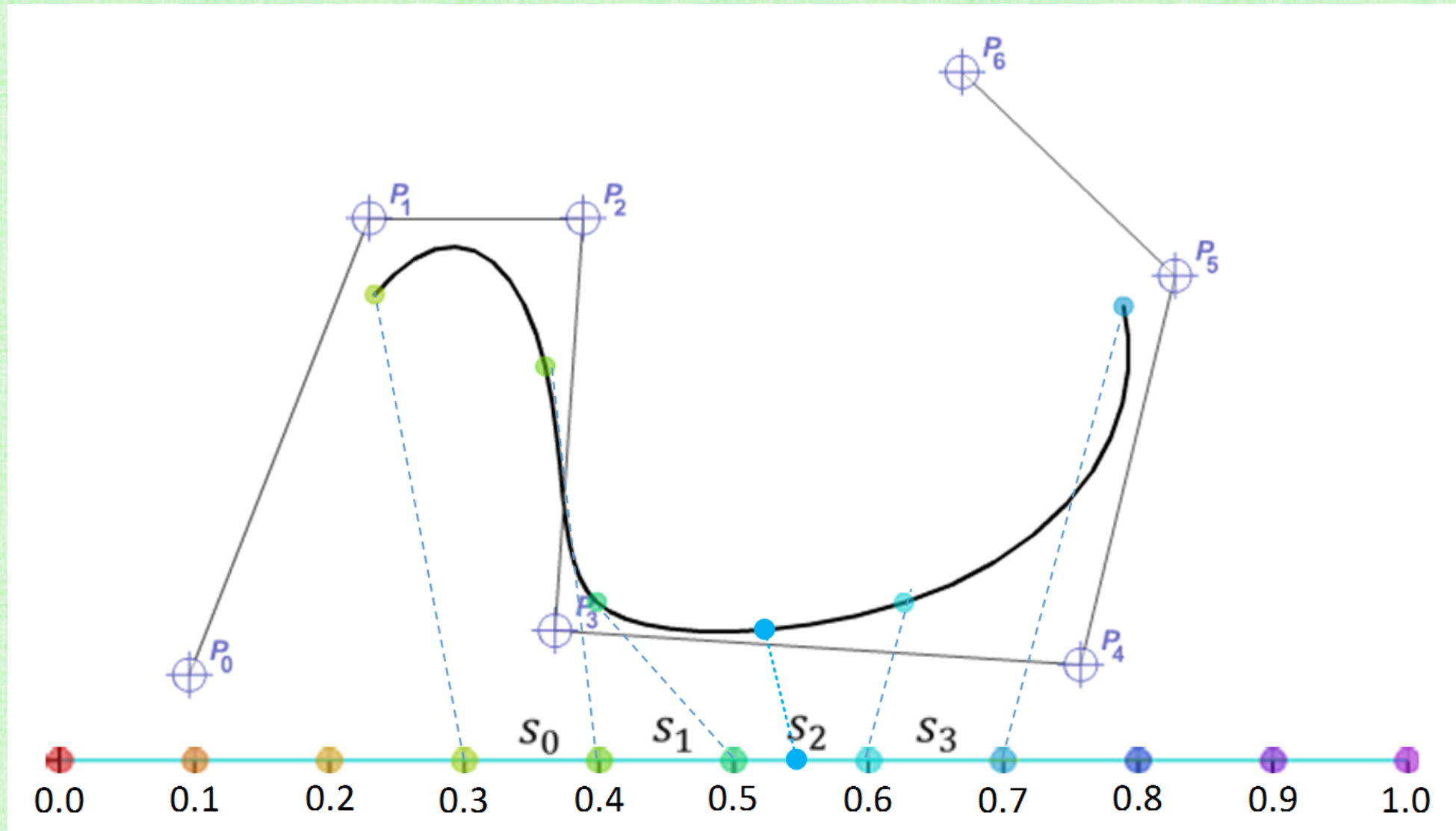
# Adding Finer Details

- First, insert new knots (and control points) without changing the shape of the curve
- Then, move around control points to modify the shape of the curve

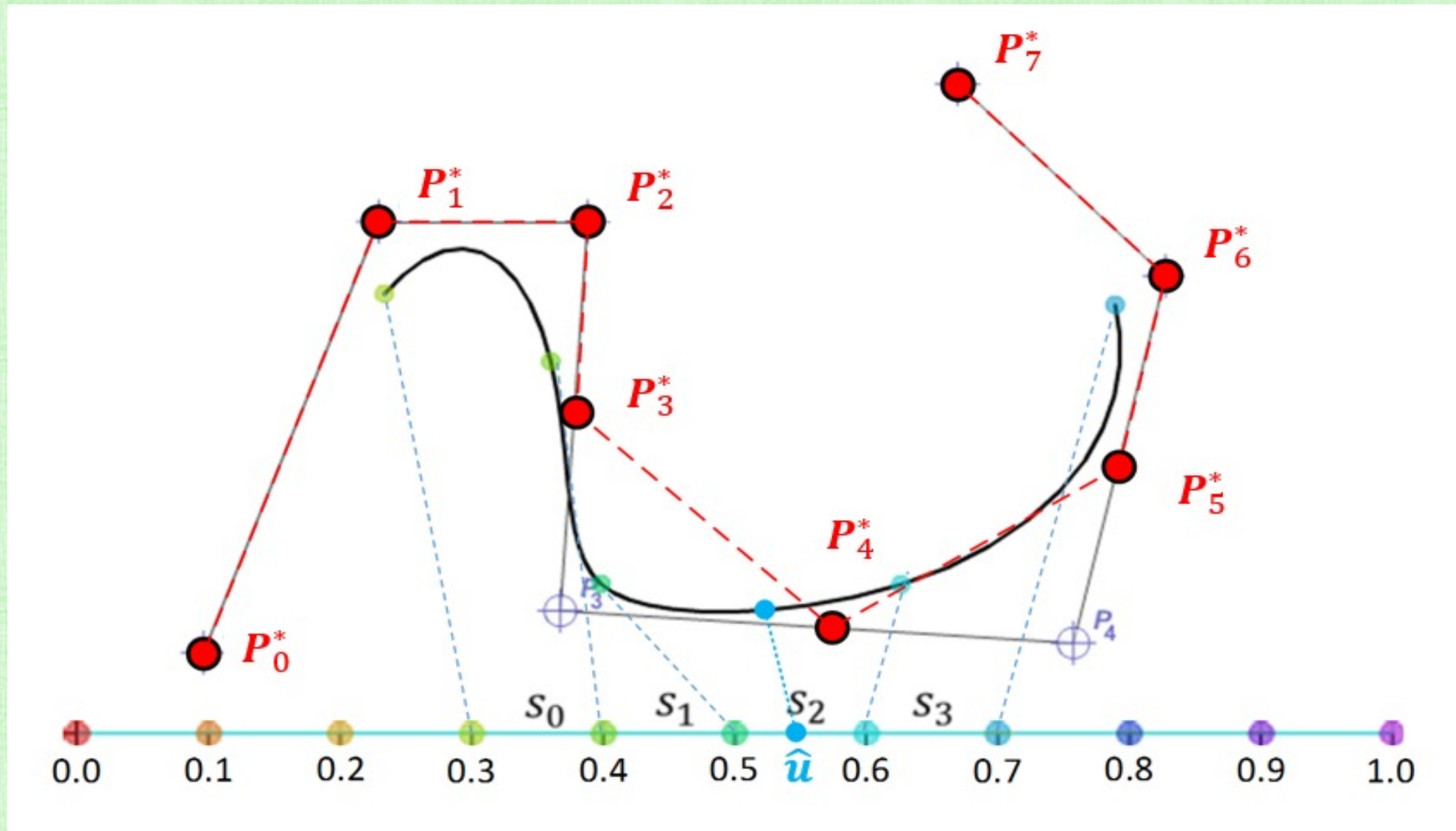
# Original Curve



# Insert a Knot

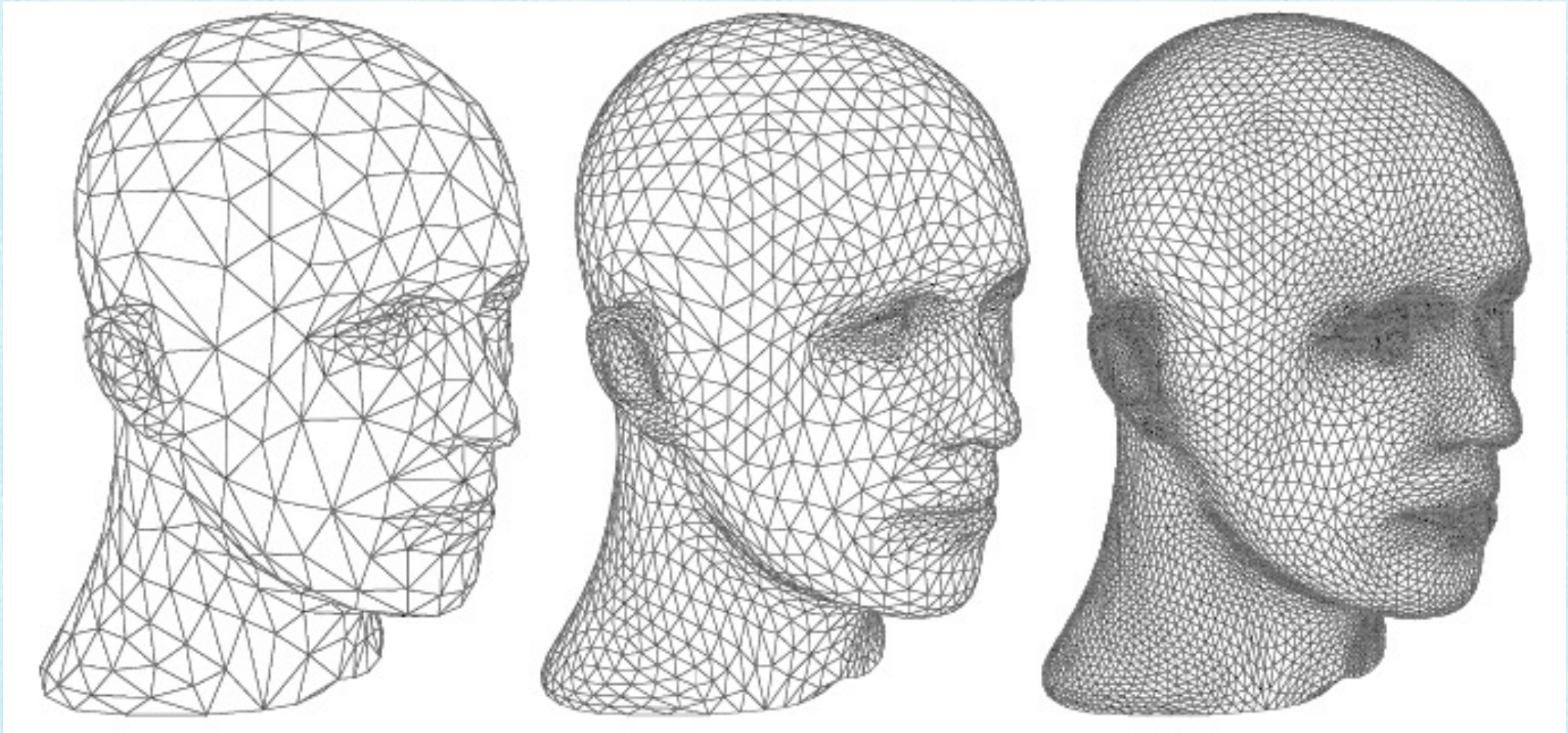


# Add/Adjust Control Points



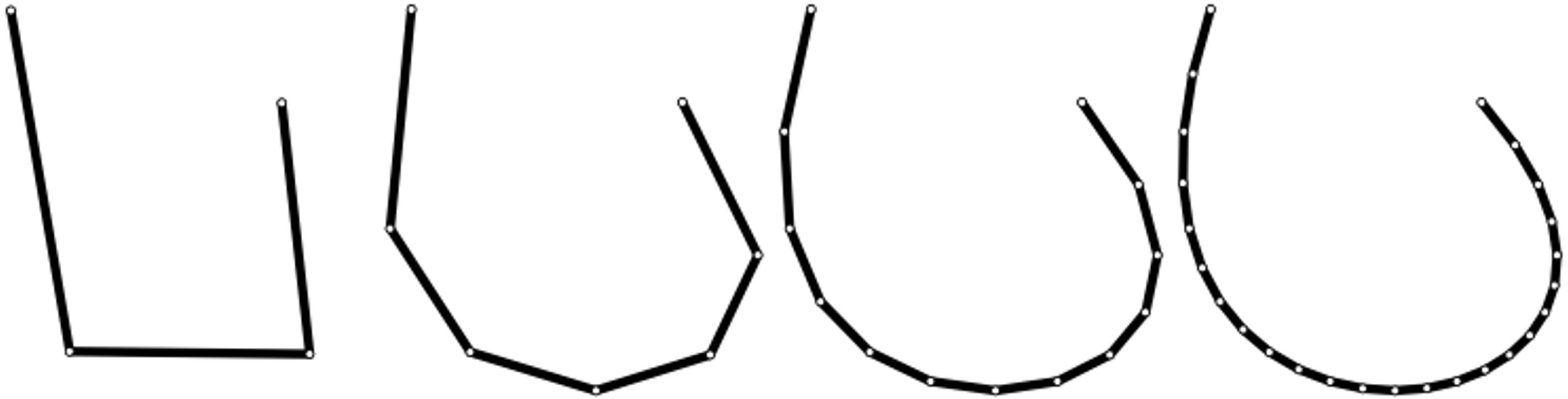
# Subdivision

- Automatically generate a finer (and smoother) mesh from coarser geometry
- Can be done repeatedly



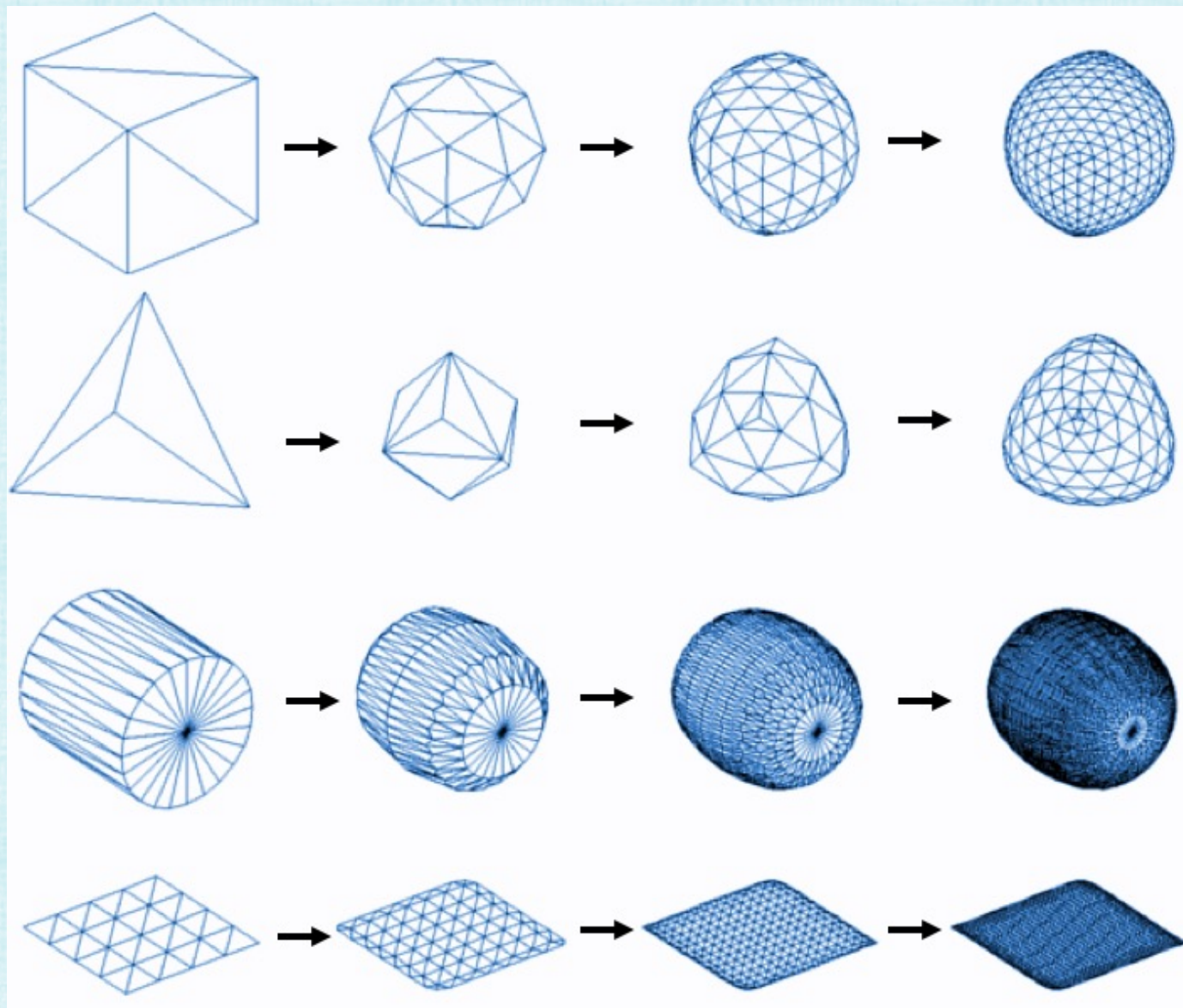


# Subdivision Curves



# Subdivision Surfaces

- Subdivide each triangle into 4 sub-triangles
- Move the old and new vertices
- Repeat (if desired)



# Charles Loop

**Smooth Subdivision Surfaces Based on Triangles**

by  
Charles Teorell Loop

A thesis submitted to the faculty of  
the University of Utah  
in partial fulfillment of the requirements for the degree of

**Master of Science**

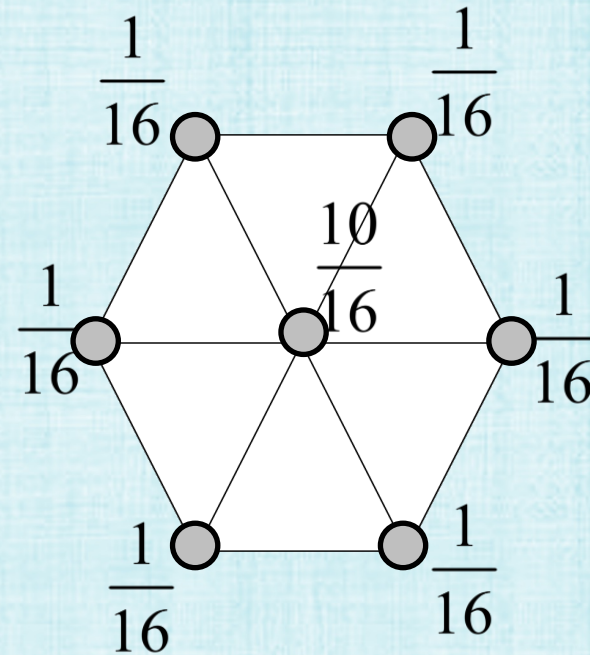
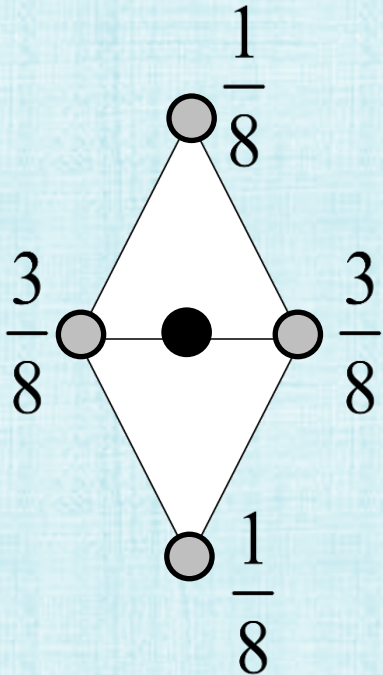
Department of Mathematics  
The University of Utah  
August 1987



2406 citations (for an MS thesis!)

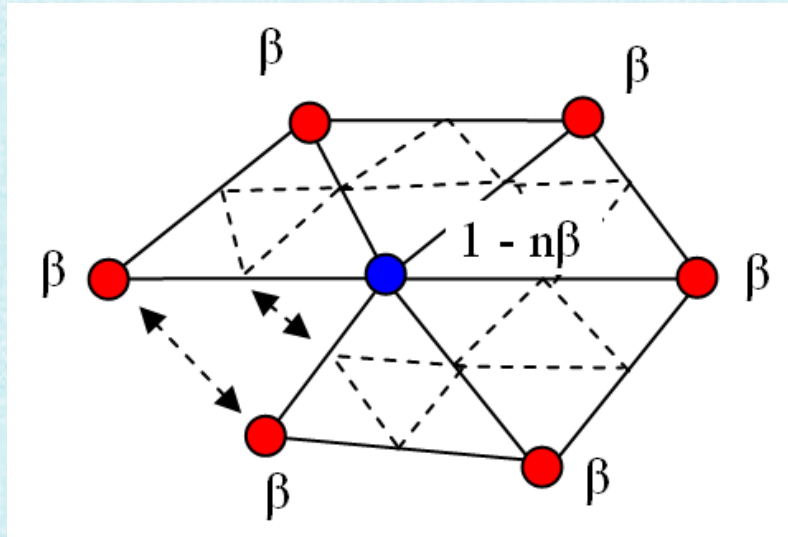
# Move Old/New Vertices

- Set the position of a new vertex (shown in black) using a weighted average of the four nearby original vertices (shown in grey)
- Change the position of each **regular** original vertex (shown in grey) using a weighted average that includes the **six** adjacent original vertices (also shown in grey)

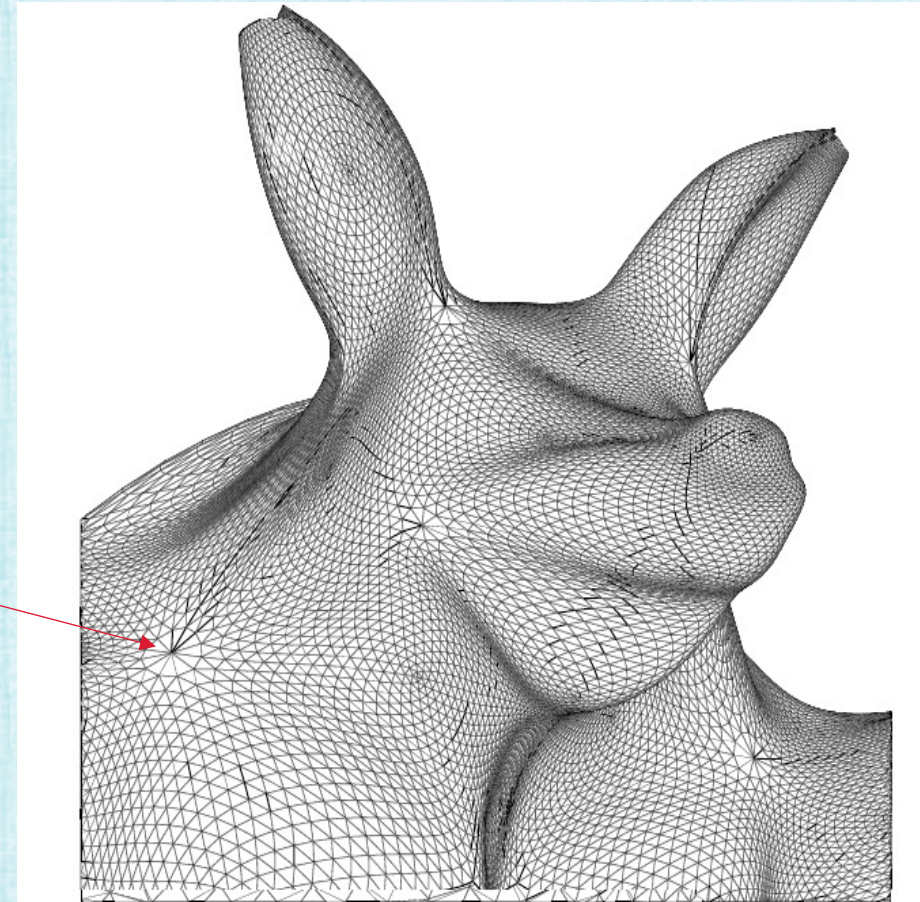


# Extraordinary Vertices

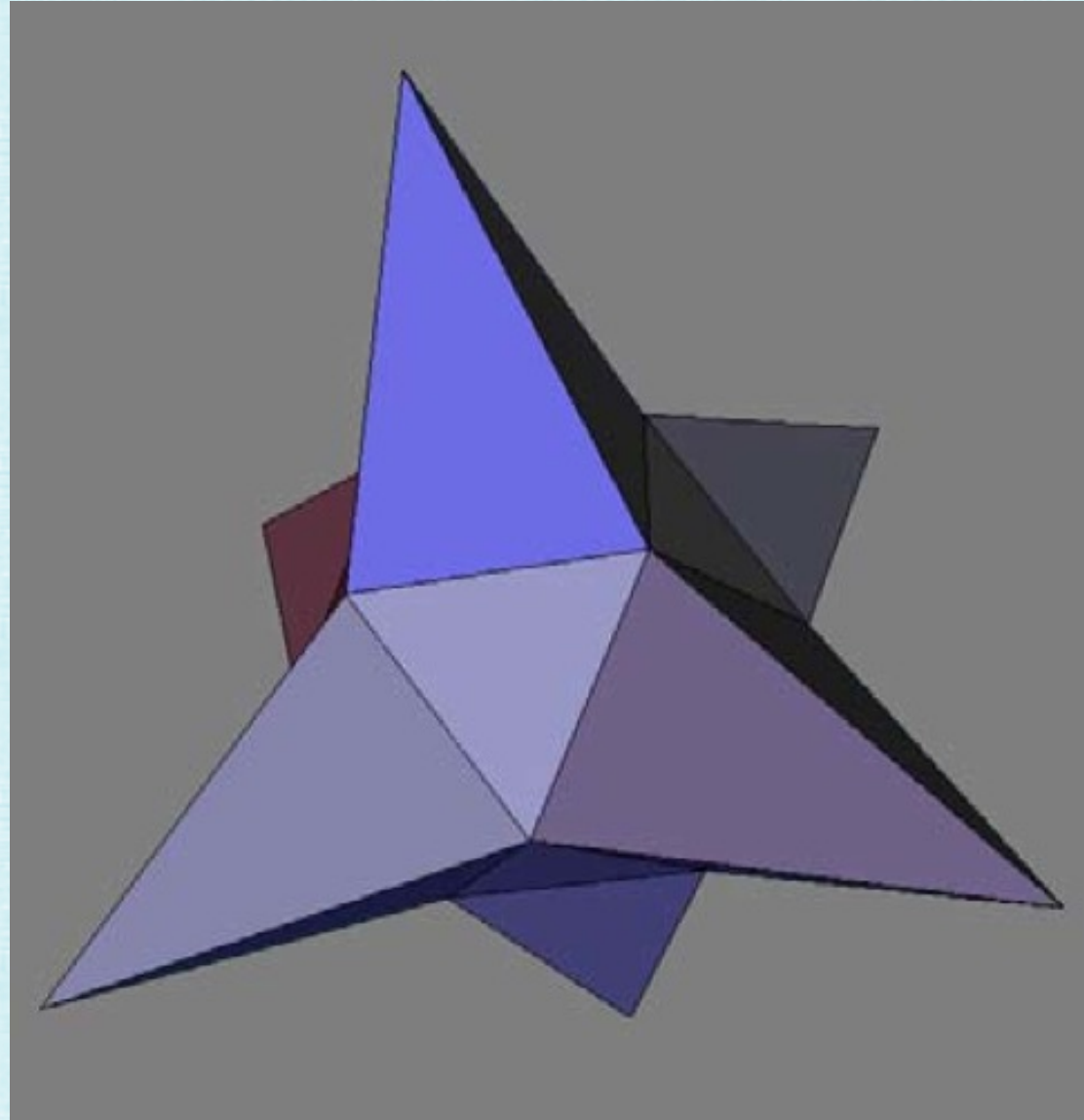
- Most vertices are regular (degree 6)
- If a mesh is topologically equivalent to a sphere, not all vertices are regular
- At extraordinary vertices, use special weights:
  - If number of neighbors is  $n = 3$ ,  $\beta = \frac{3}{16}$
  - else  $\beta = \frac{3}{8n}$



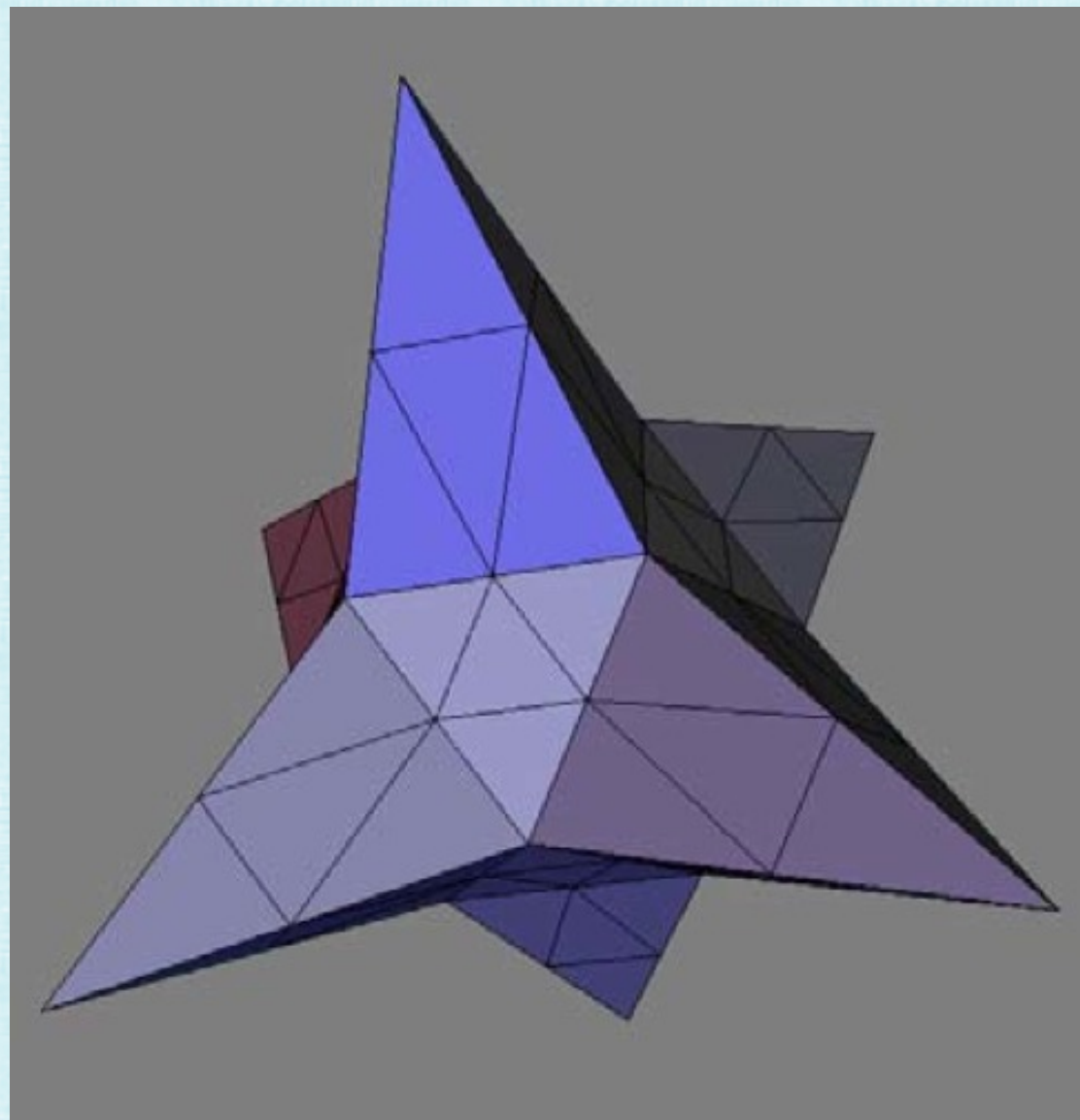
extraordinary  
vertex



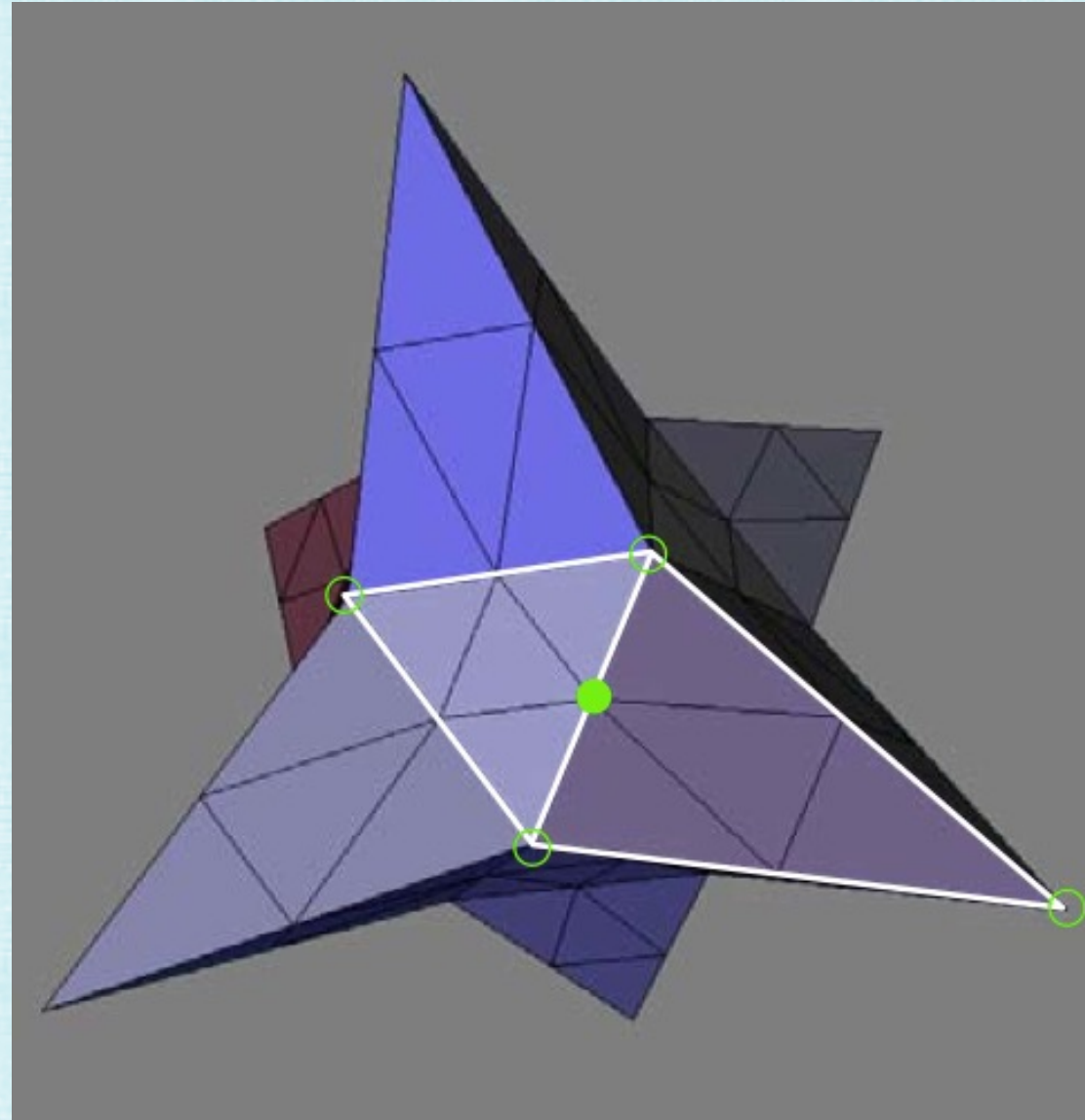
# Initial Mesh



# Add New Vertices

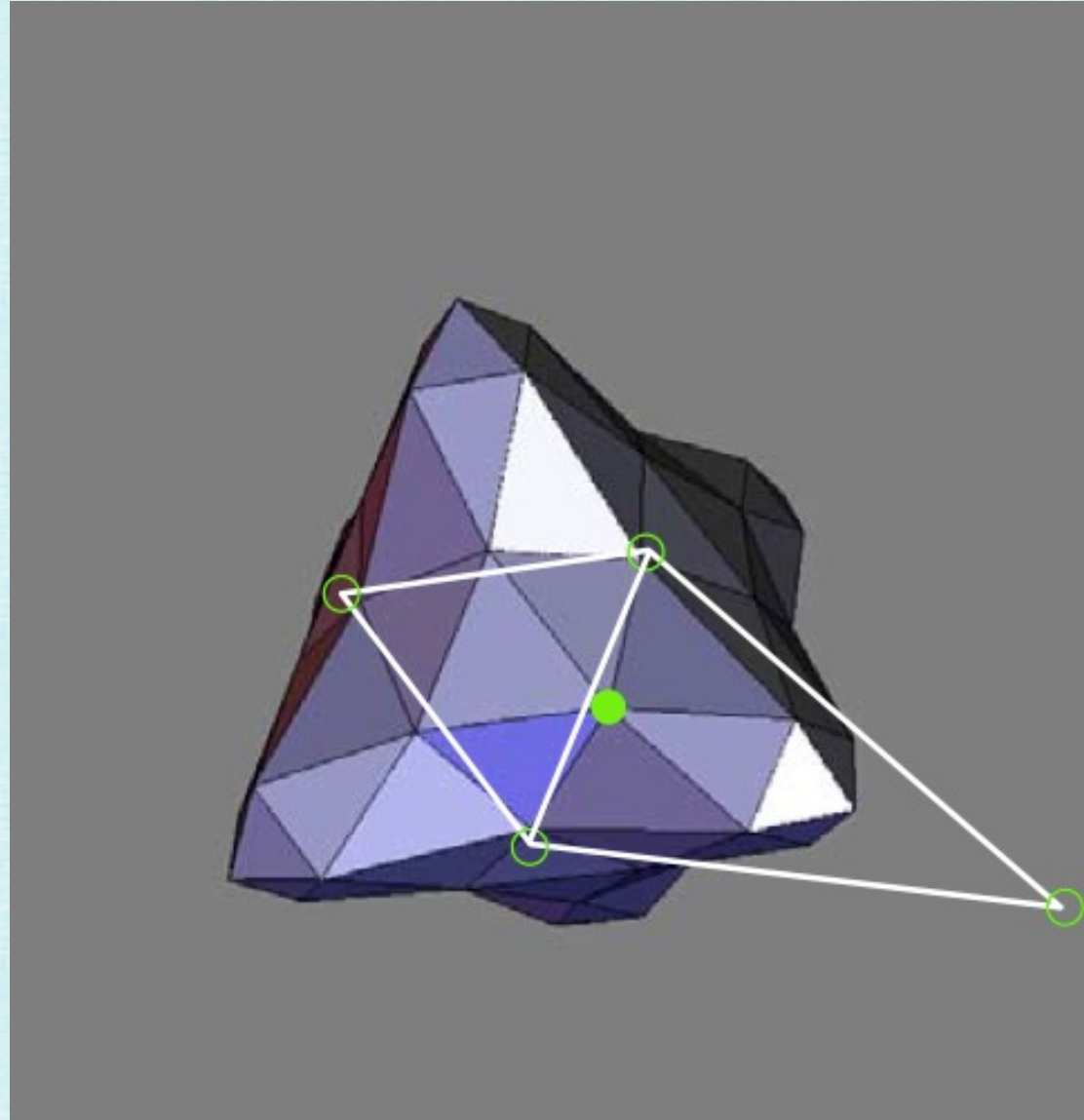


# New Vertex and Stencil

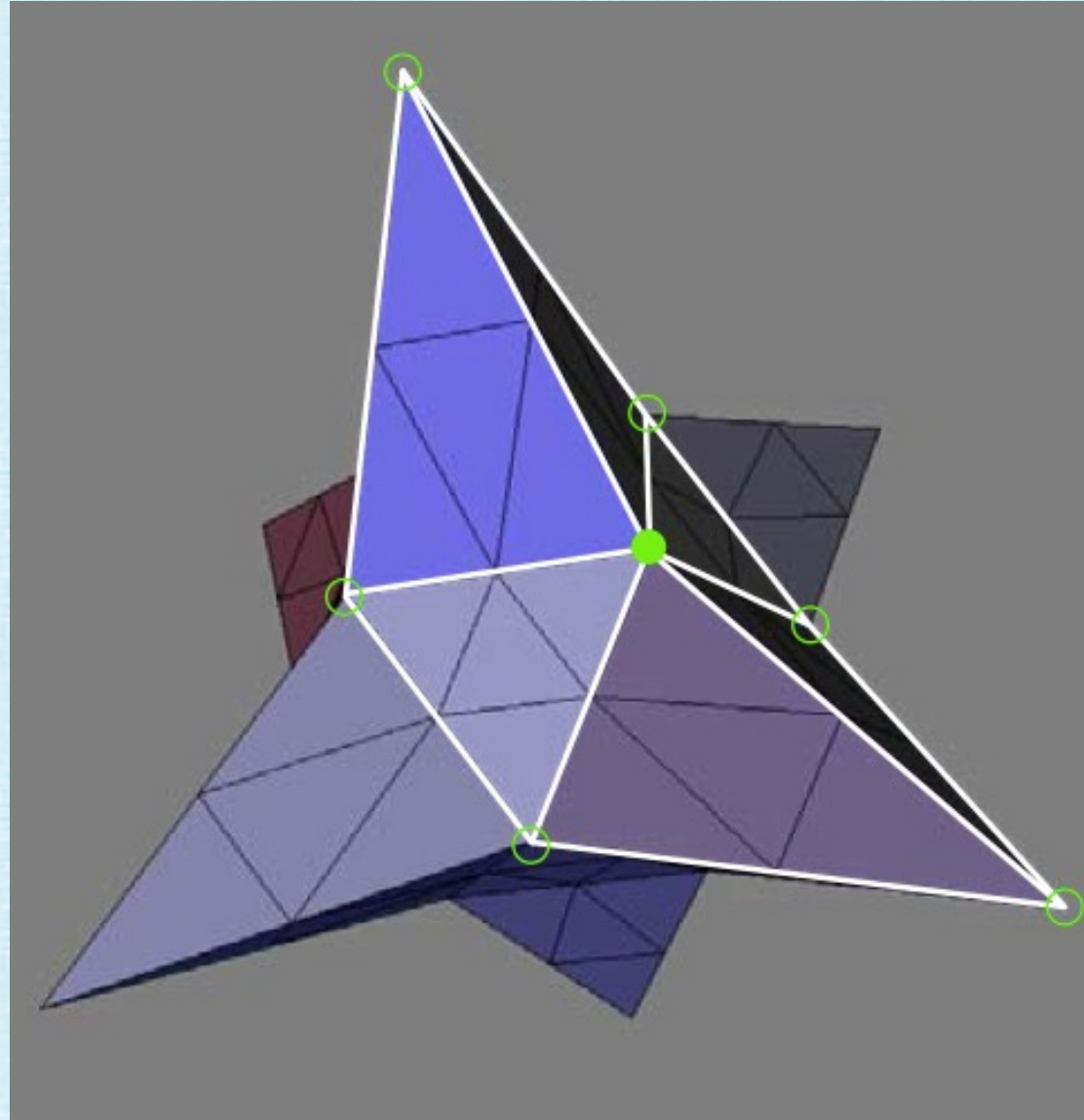




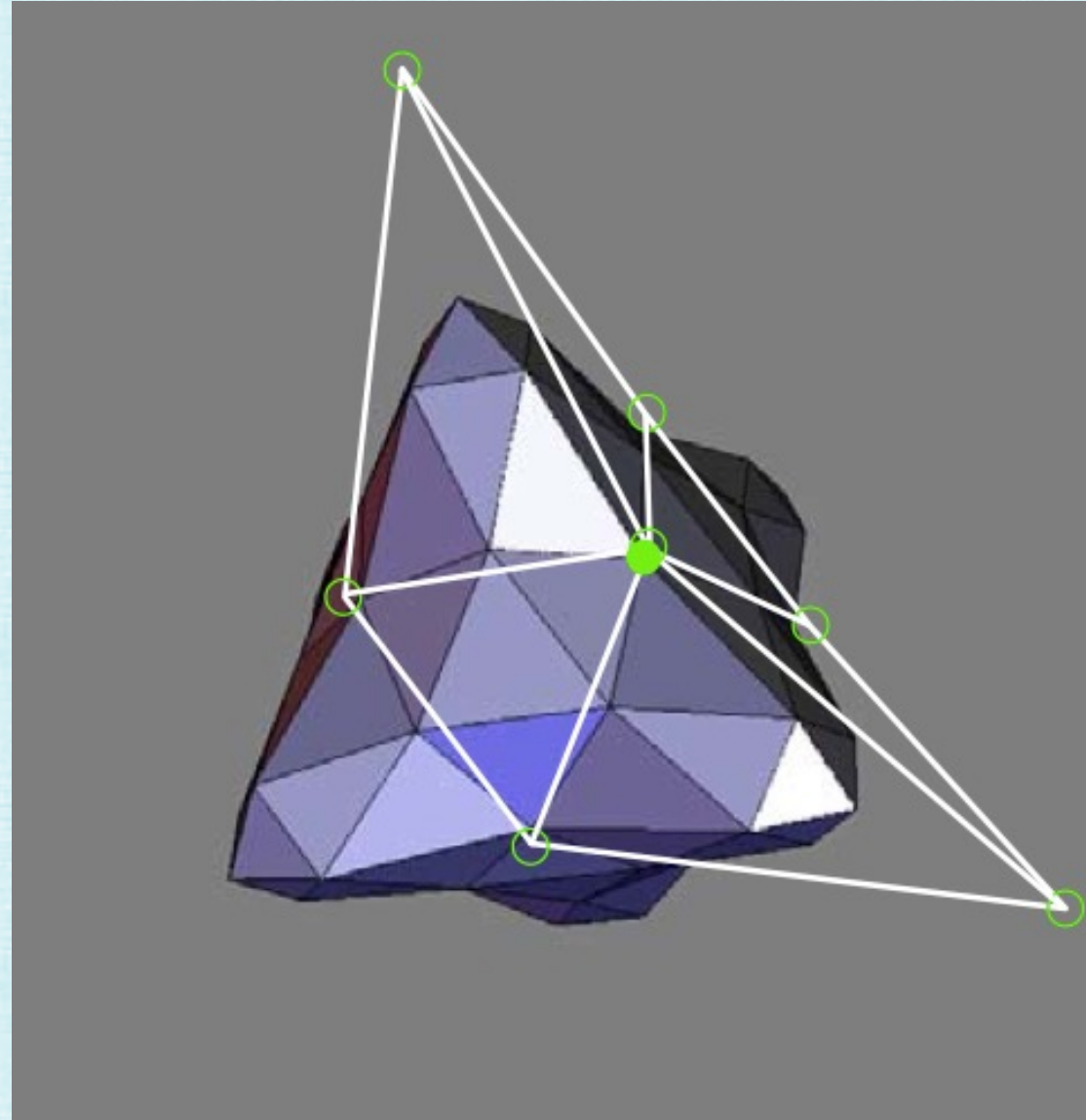
# Move New Vertex



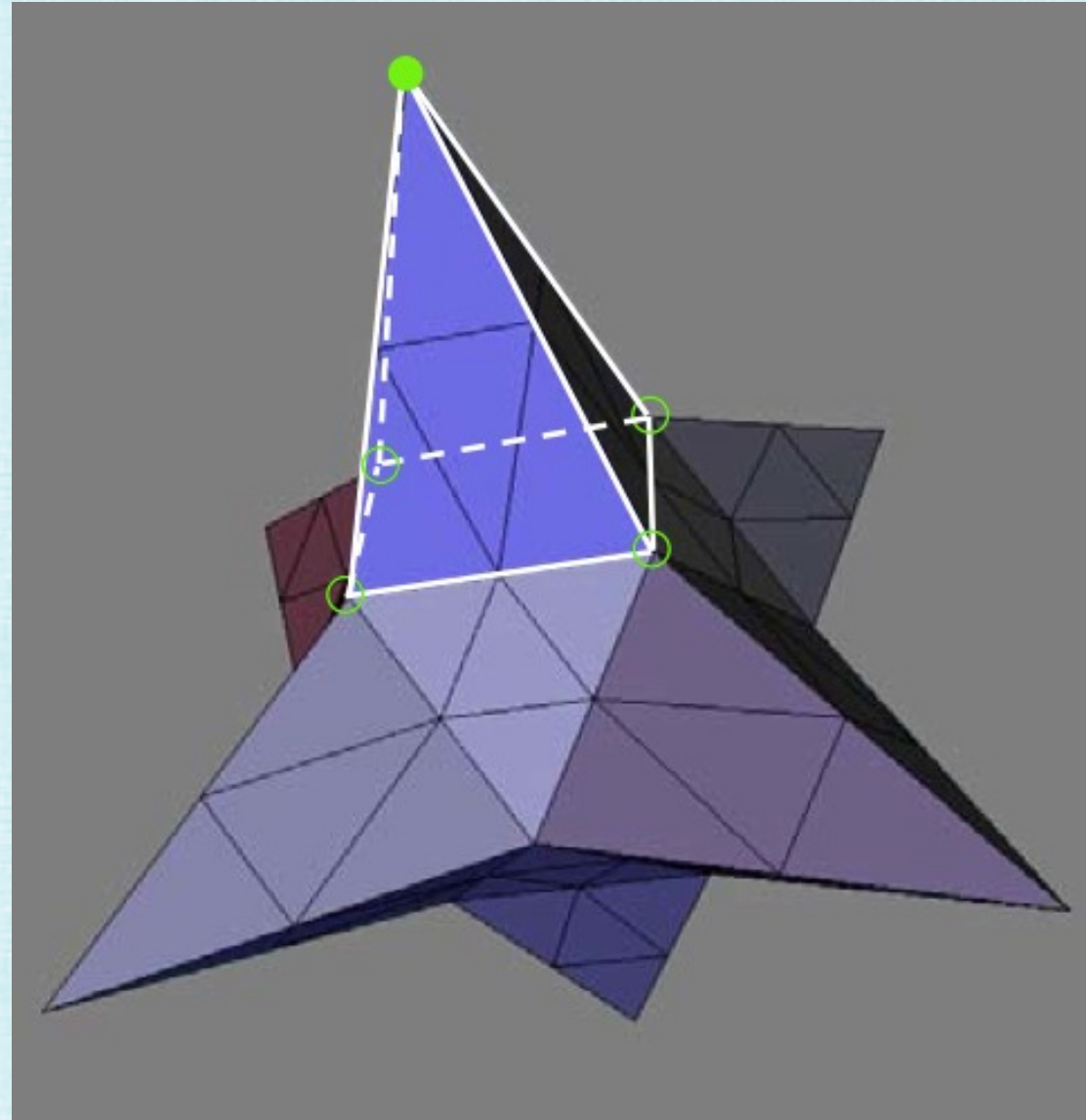
# Original (Regular) Vertex and Stencil



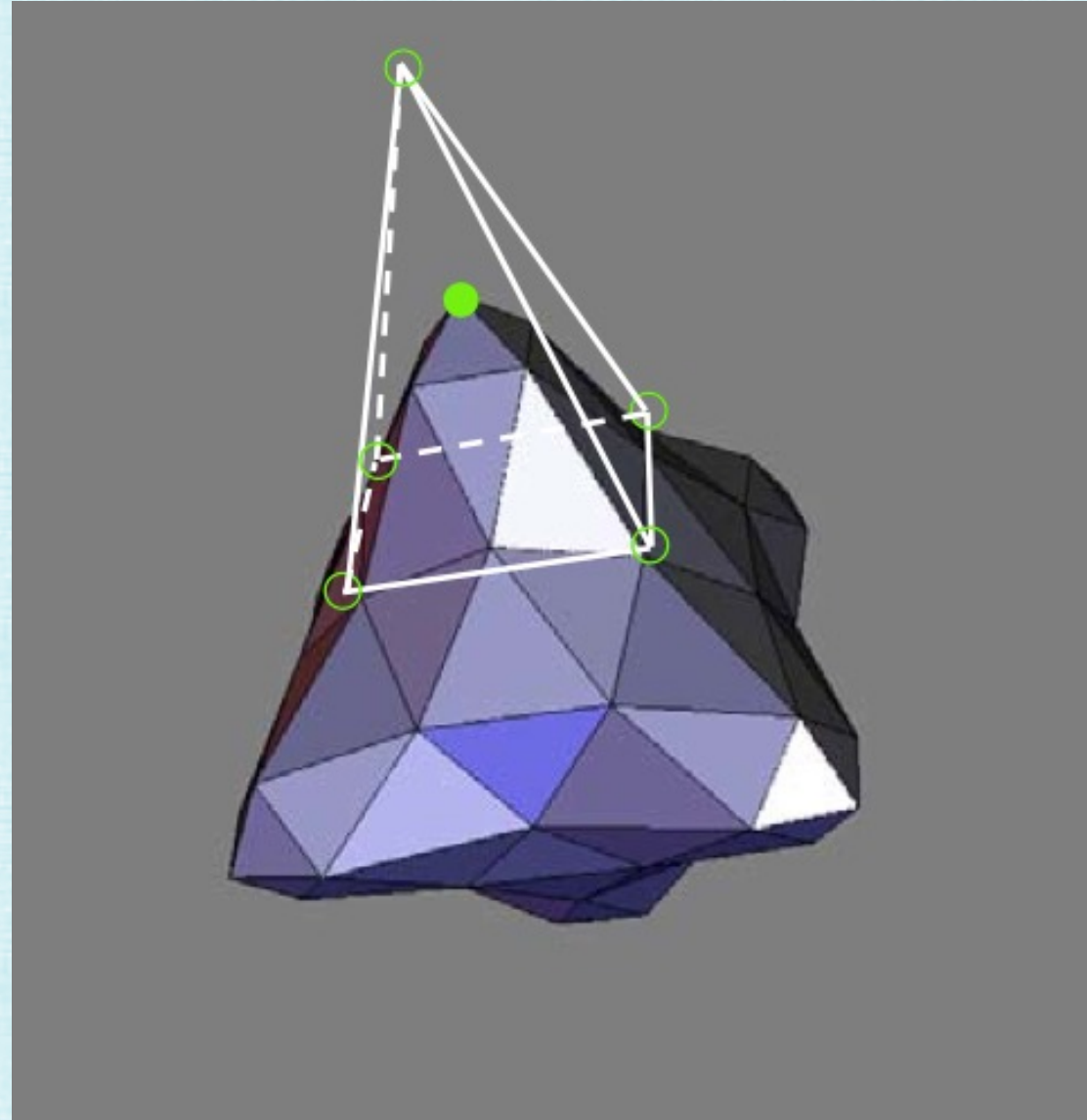
# Move Original (Regular) Vertex



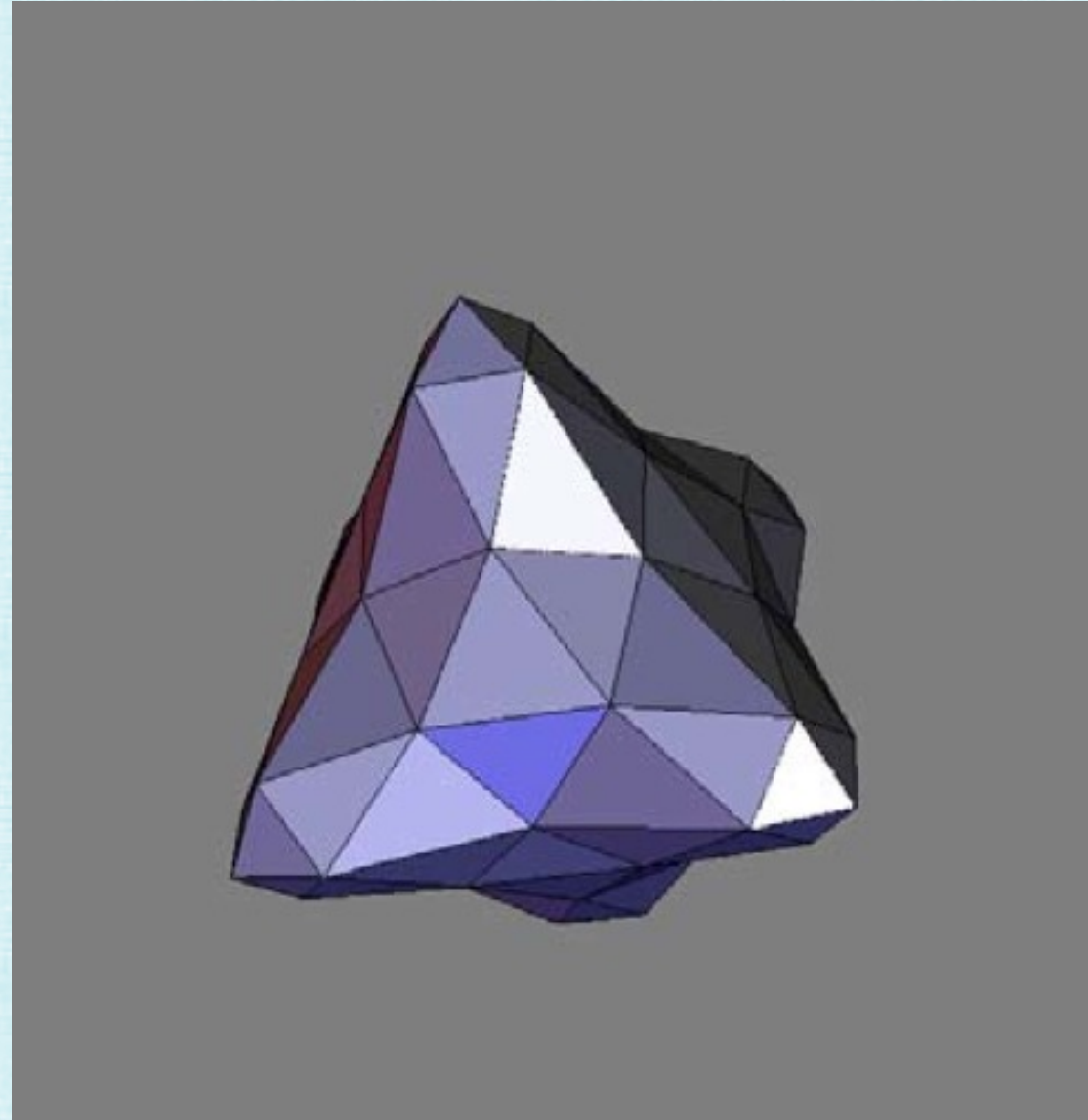
# Original (Extraordinary) Vertex and Stencil



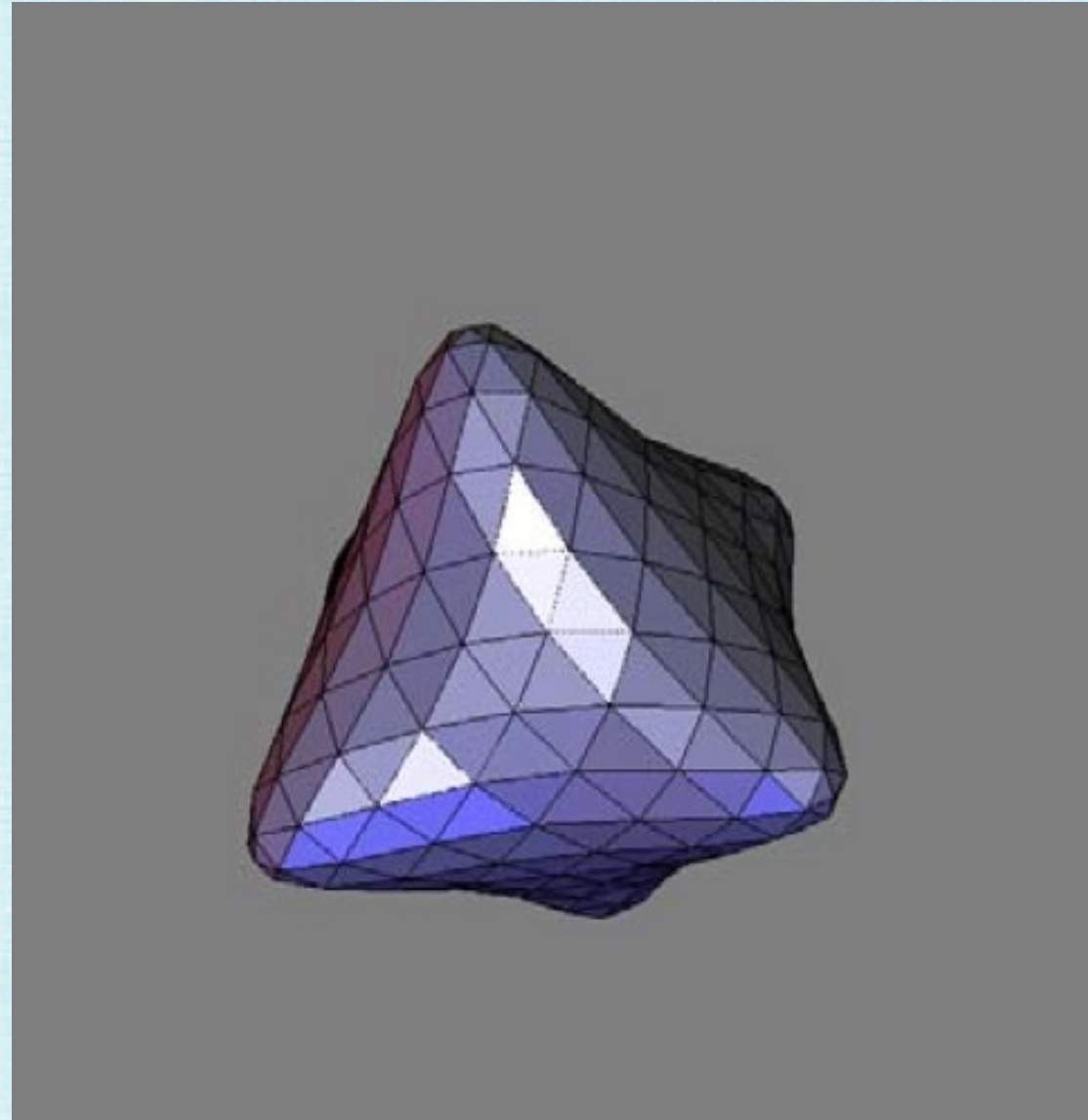
# Move Original (Extraordinary) Vertex



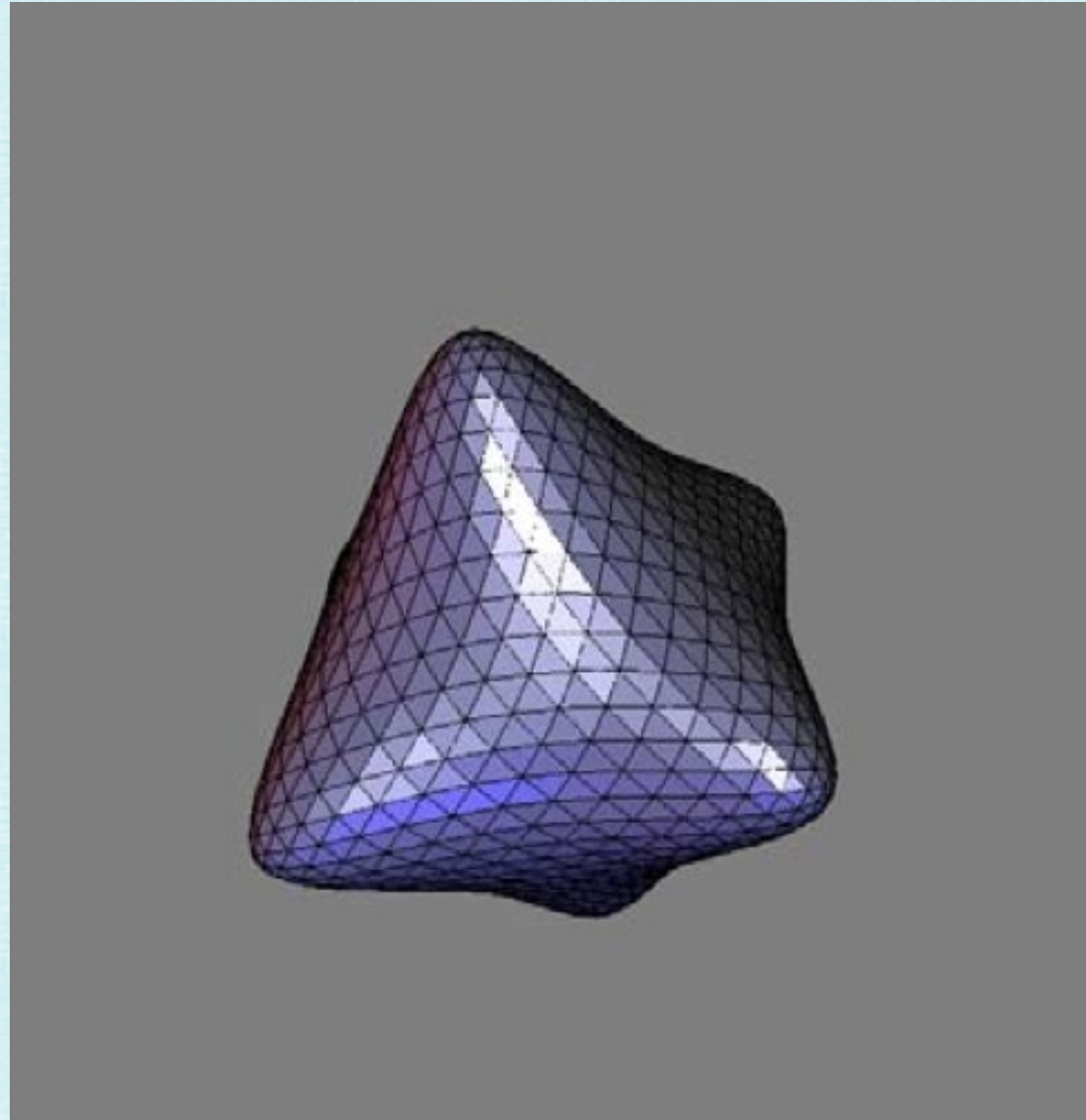
# Subdivided Surface



# Subdivide Again

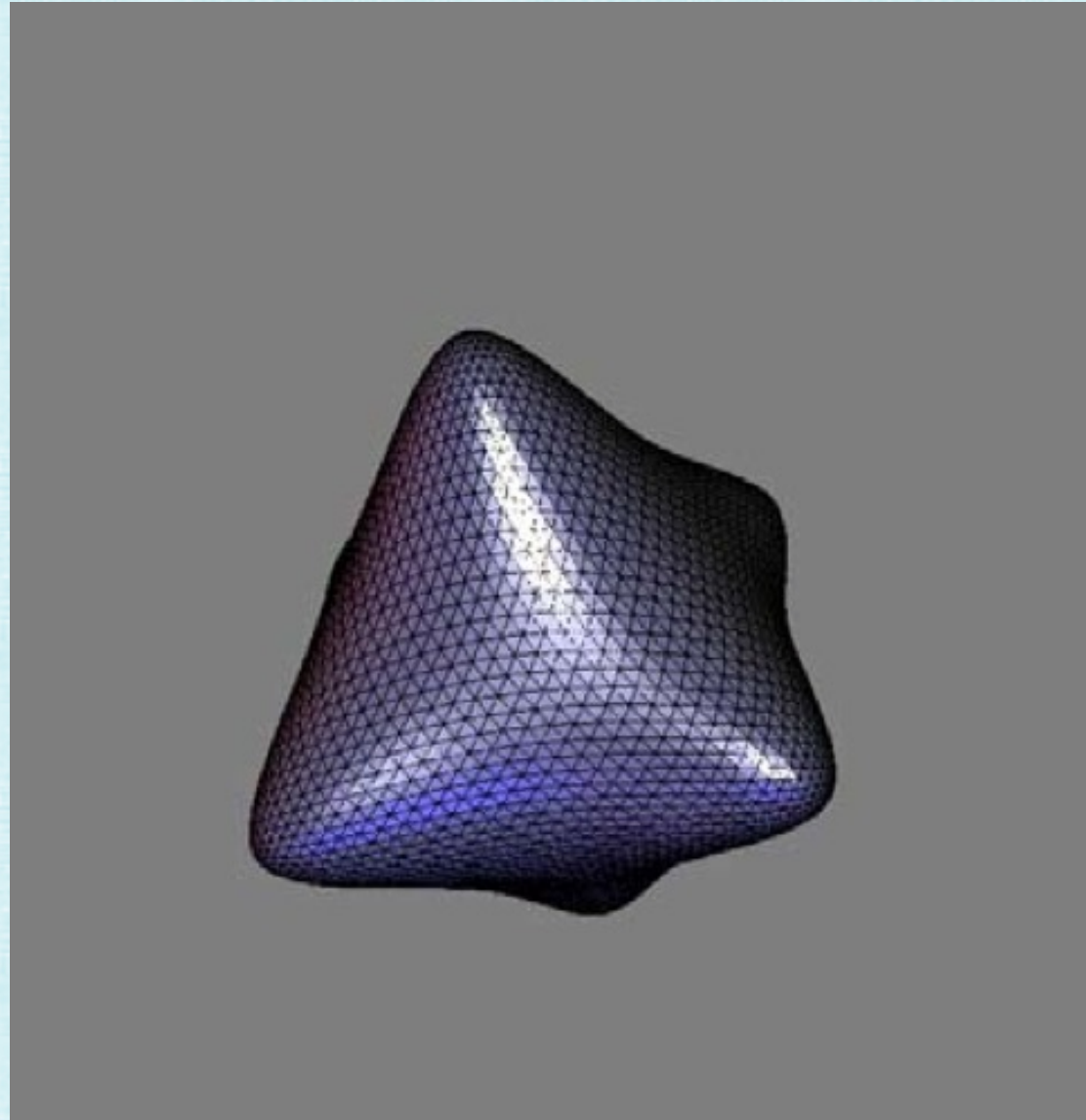


And Again

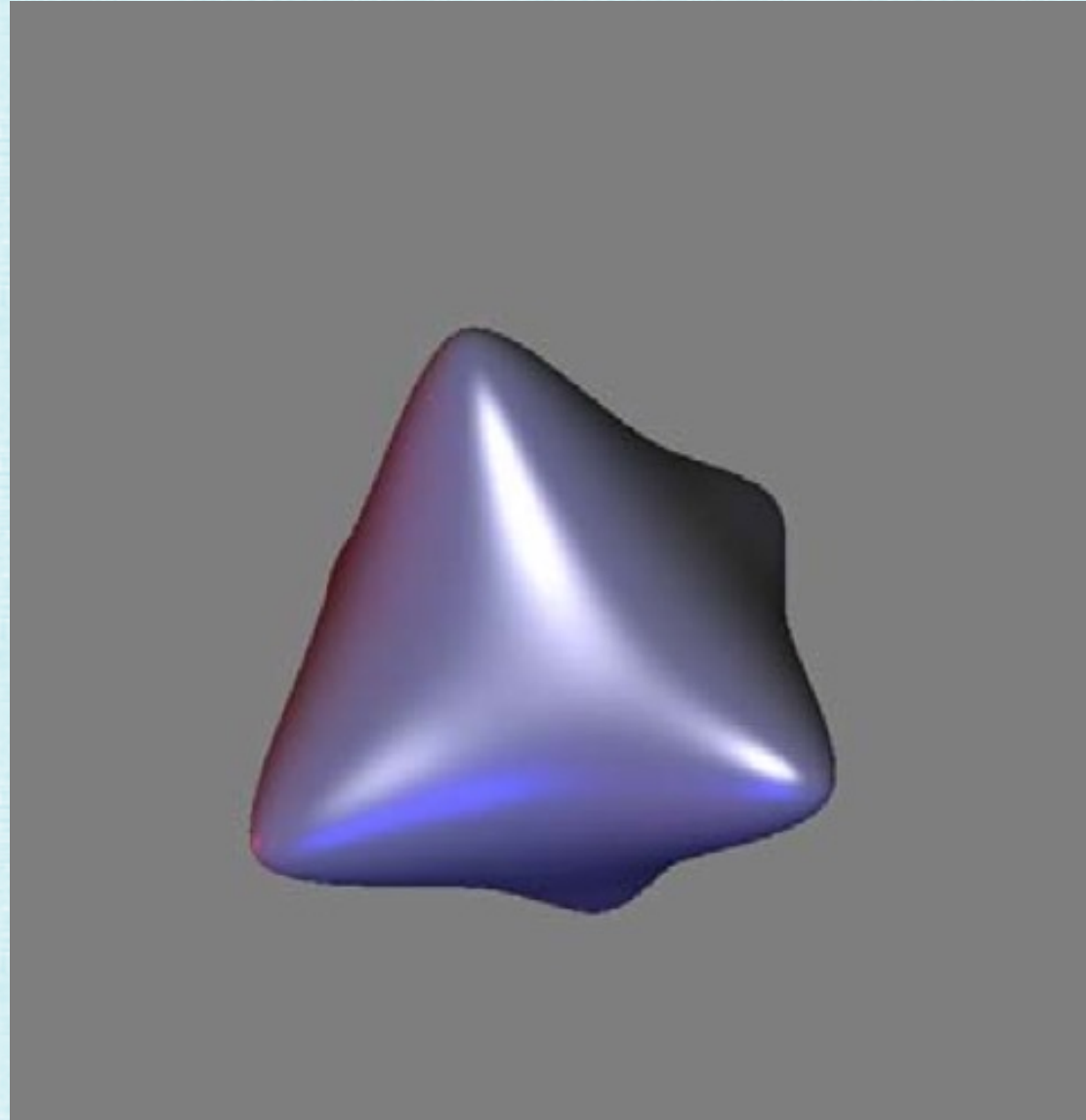




And Again

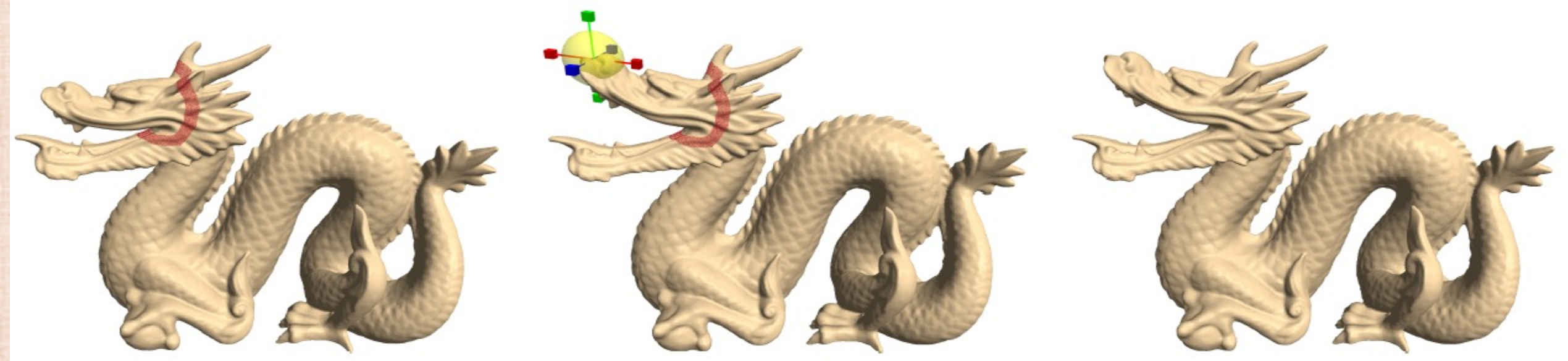


# Final (Smooth) Limit Surface



# Laplacian Mesh Editing

- Instead of editing via control points (as was done for NURBS), Laplacian mesh editing allows one to (more directly) move vertices
  - Select a subset of the mesh to move (and target positions)
  - Select a region of influence containing the subset (so that the deformation is smooth)



# Differential Coordinates

- The differential coordinate is the difference between a vertex's position  $x_i$  and the average position of its neighbors:

$$d_i = L(x_i) = x_i - \frac{1}{n_i} \sum_{j \in N_i} x_j$$

- Let  $X$  be the vector of mesh vertices (one entry for each vertex)
  - Let  $D$  be the vector of differential coordinates (one entry for each vertex)
  - Since each  $d_i$  is linear in  $x_i$  and all  $x_j$ , can write  $D = MX$  where  $M$  is a constant coefficient (sparse) matrix
- 
- Differential coordinates approximate the local shape:
    - The direction of  $d_i$  approximates the direction of the normal vector
    - The magnitude of  $d_i$  approximates the mean curvature

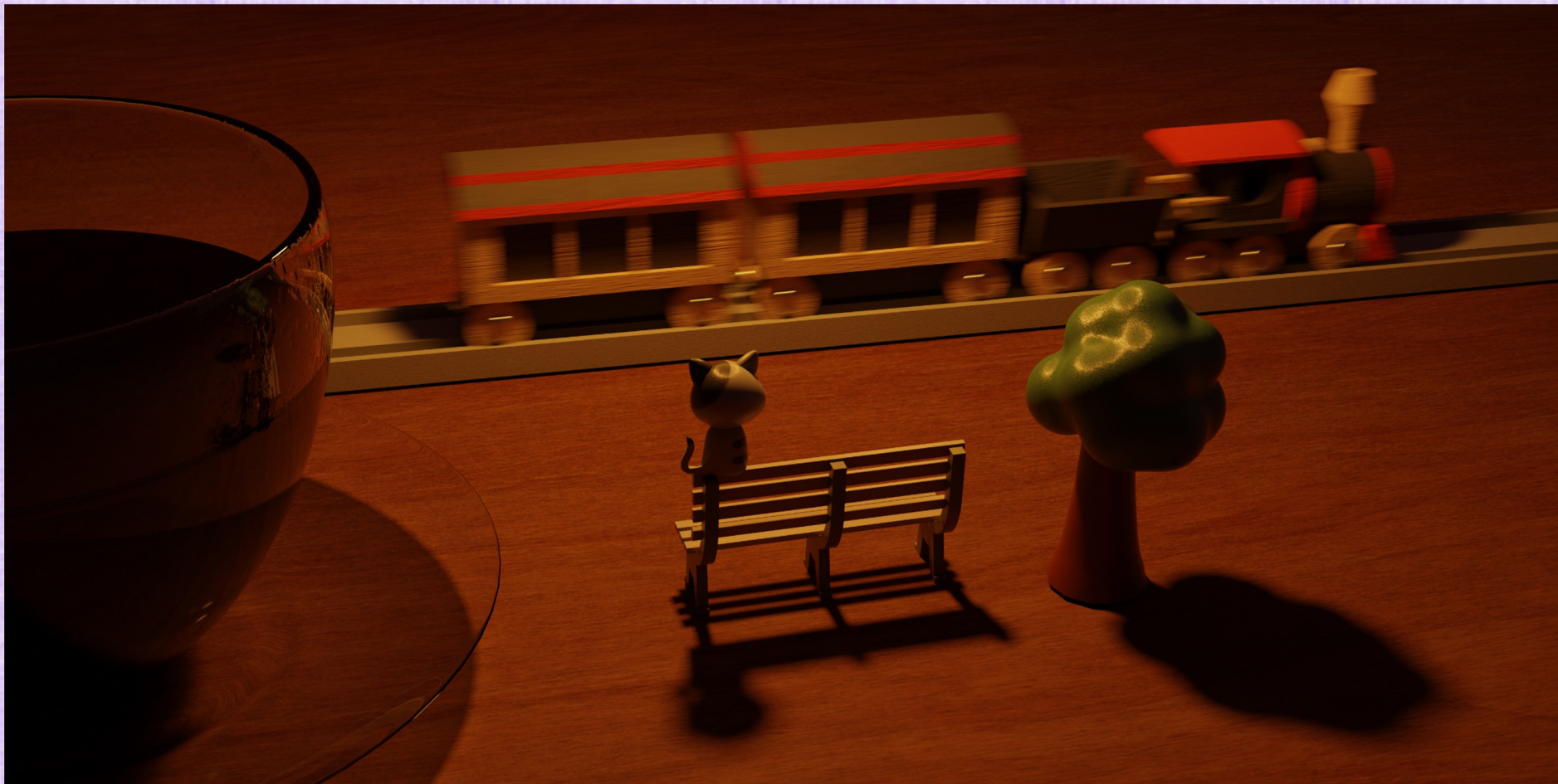
# Laplacian Mesh Editing

- Select mesh points  $x_c$  and target positions  $p_c$
- Select a region of influence  $\Omega$  containing (at least) all the  $x_c$
- Solve (the sparse linear system)  $M\hat{X} = D$  for all  $\hat{x}_i \in \Omega$ , with soft constraints  $\hat{x}_c = p_c$
- $M\hat{X} = D$  aims to preserve the shape, while  $\hat{x}_c = p_c$  targets the deformation
- Minimize  $E(\hat{X}) = \sum_{i \in \Omega} \|L(\hat{x}_i) - d_i\|_2^2 + \sum_c \|\hat{x}_c - p_c\|_2^2$
- Improved results can be obtained by approximating part of the deformation via a transform  $T_i(\hat{X})$ , computed for each vertex via:  $\min_{T_i} \left( \|T_i x_i - \hat{x}_i\|_2^2 + \sum_{j \in N_i} \|T_i x_j - \hat{x}_j\|_2^2 \right)$
- Then  $E(\hat{X}) = \sum_{i \in \Omega} \|L(\hat{x}_i) - T_i(\hat{X})d_i\|_2^2 + \sum_c \|\hat{x}_c - p_c\|_2^2$



Blender, Clip Studio Paint, TurboSquid

- Pear is downloaded from TurboSquid, all other geometry created from scratch
- Cat character is created through modeling from geometric primitives and sculpting
- Cup is broken through cell fracture and rigid body collisions
- Liquid simulations to model fluid inside cup
- Leaves/trees created through particle system
- Textures are created/modified through Clip Studio Paint or taken from online



- Set train at different positions on separate keyframes to create motion blur.
- Used loop cuts to make fine-grained adjustments to the geometry of the mesh.
- Applied noise textures, bump map, and color ramp to create plastic material.
- Marked edges as sharp and used Auto Smooth to ensure that intentional creases are not smoothed out by Shade Smooth or Subdivision Surface Modifier.
- Downloaded the "UV Squares" add-on that helps streamline UV unwrapping by converting a UV map to a square grid.

**Edwin Pua, CS148  
2021**



- Tree created from scratch using a skin modifier on a skeleton of vertices and edges forming the shape of a tree, and subdivided later to apply weight painting for the particle system.
- Bridge, stone, ground, and mountains created from scratch using extrusion and various low-poly modeling techniques.
- Lotus flower, lily pads, cherry blossoms downloaded from online sources.
- Generated cherry blossoms onto the tree randomly using weight painting and hair particle system onto the tree.

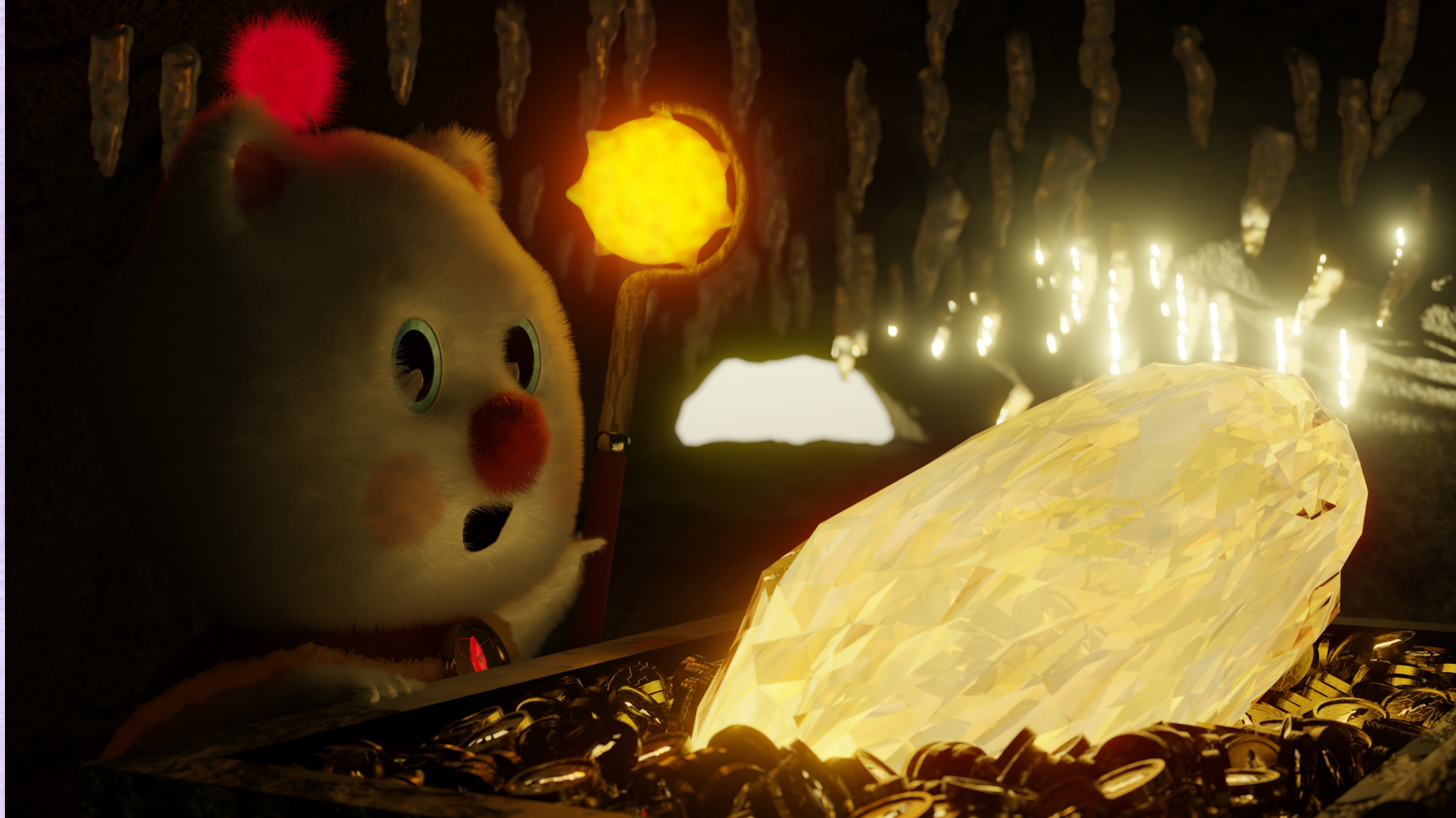
Amy Flo, Vivian Xiao





Group Members: Nicholas Negrete (individual project)

- Scaled a sphere to model the basic body shape, then applied decimate to get the “crystal” look
- Applied spin to a cylinder to model the rounded swan neck
- Used the bevel function to round the wood surface edges
- Used the “inset faces” method on the face of a cube and moved the face back to create the case
- Used boolean modifiers to make the different parts look like one piece of glass



## Crystal Explorer by Kent Vainio (2020)

### **METHODS:**

- Cloth and rigid body simulations for the cape and treasure chest items
- Blender's hair particle system to create the fluffy fur, cape fur and antenna
- Multiple procedural textures (mostly Voronoi and Musgrave) along with various shaders (glass, emission, etc.) to create a vibrant scene
- Moogle (the fluffy creature - a character from the Final Fantasy video game series) was modeled and sculpted from scratch using reference images



Used loop cut, subdivision, solidify, bezel tool, bisect tool and displacement to create the whiskey, moon rock award, photo frames, etc  
Used cloth simulation to model the newspaper. Took pictures of the book covers and created UV textures with a photo editing tool for books.  
Combined LightPath node, Transparent, Glossy and Emission BSDF to create a reflective window and its imperfect surface  
Downloaded the rocket ship, lunar lander and NASA medal in SolidWorks assembly format and converted with AutoDesk Inventor  
**Yanjia Li, Lingjie Kong, CS148 2020**



Open Shading Language (OSL) Node; Displacement map; Dynamic Sky; Particle System; Motion Blur

- Geometrical low-poly modeling from scratch for all the geometries except for the grass (downloaded from TurboSquid)
- OSL node for procedural stone road texture (idea from [Erindale](#) YouTube Channel)
- Random displacement mapping to generate water wrinkles (from a subdivided plane) and tree crown (from an icosphere)
- Particle system with wind force for drifting cherry blossom petals; motion blur applied while rendering
- Blender official add-on “Dynamic Sky” for the basic light source and background; additional lighting for the reflection on the water wrinkles

# Example Slides

- Extra Credit: 5 HW points (i.e. worth one full HW assignment) for contributing:
  - a slide with a nice image
  - a few bullet points below the image on how the **geometry** was modeled
  - names of both partners (if applicable)
- The CAs will assign anywhere from 0-5 HW points for a submission (guaranteed 5 points if we use it for future classes)
- Recall: there is no overflow past the 50 percentile HW cap
- Deadline: end of the quarter