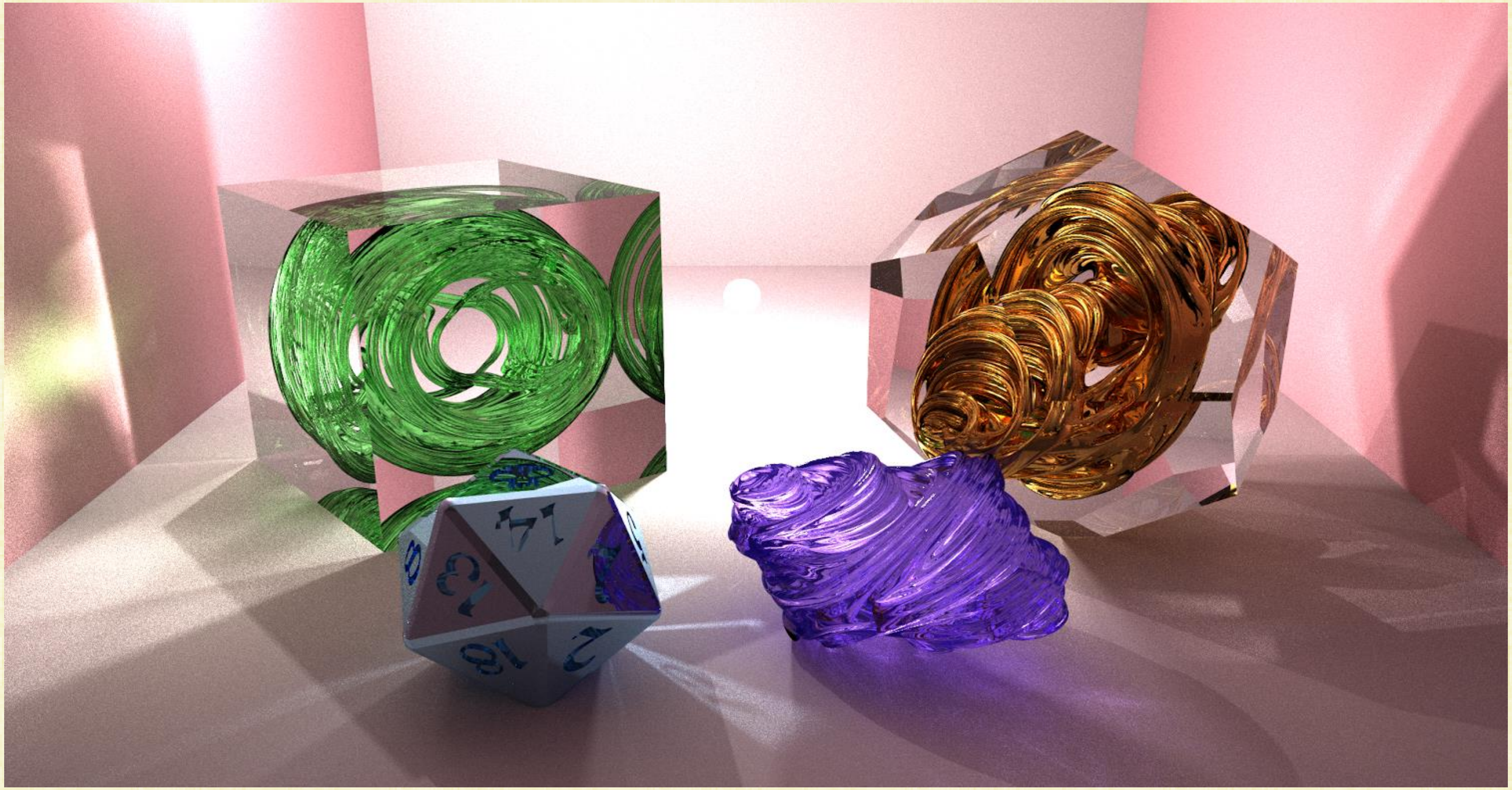
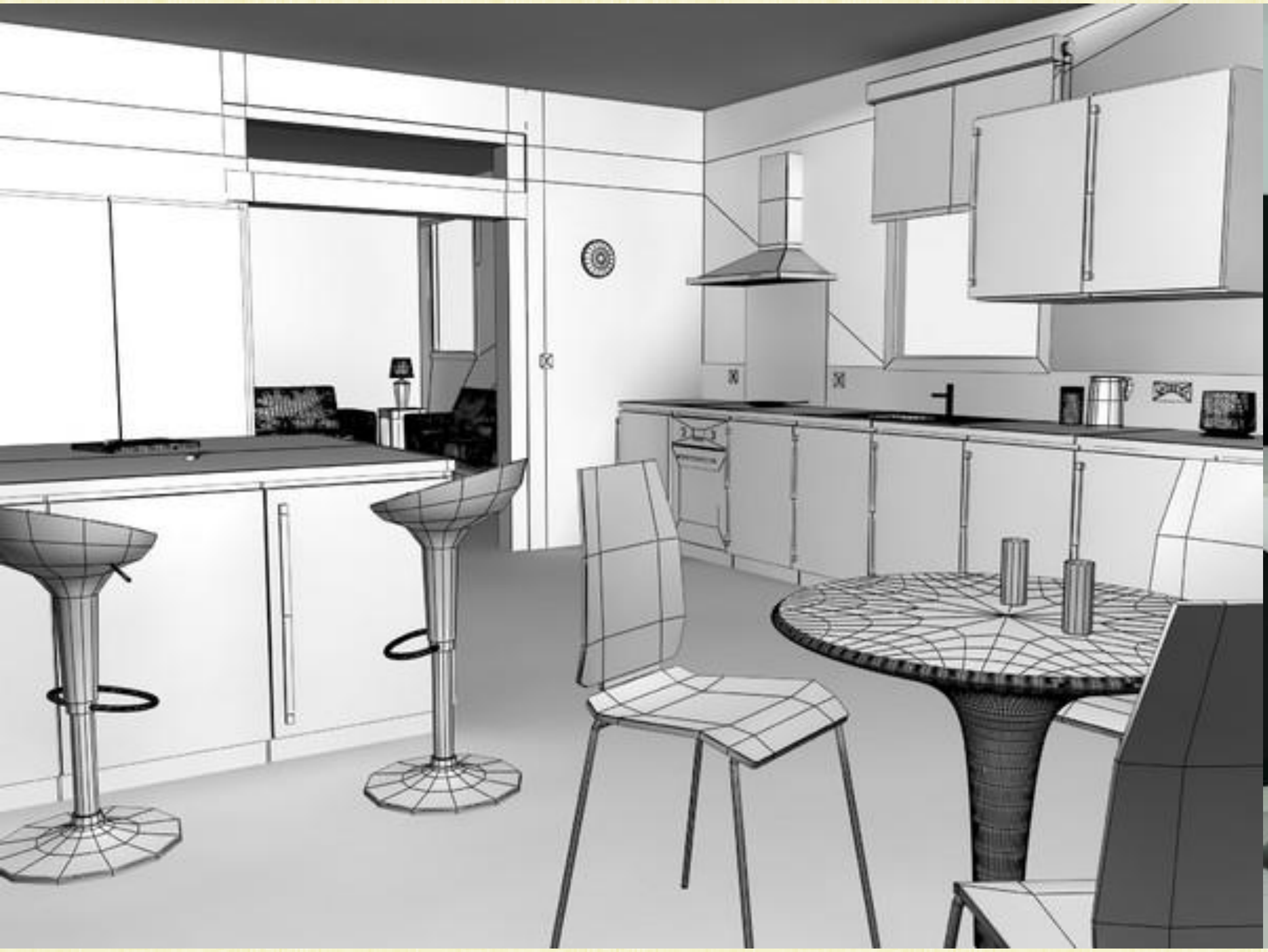
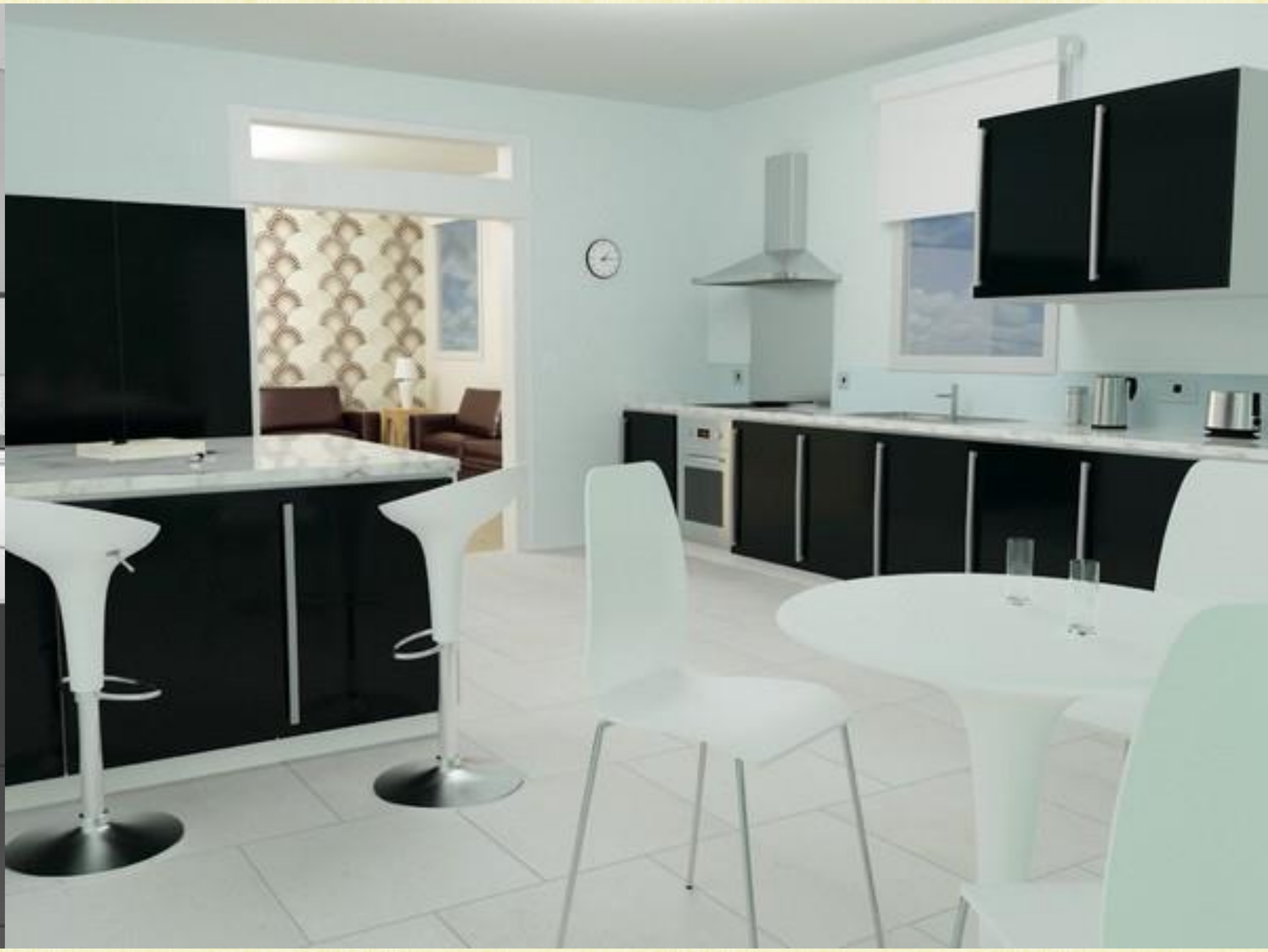
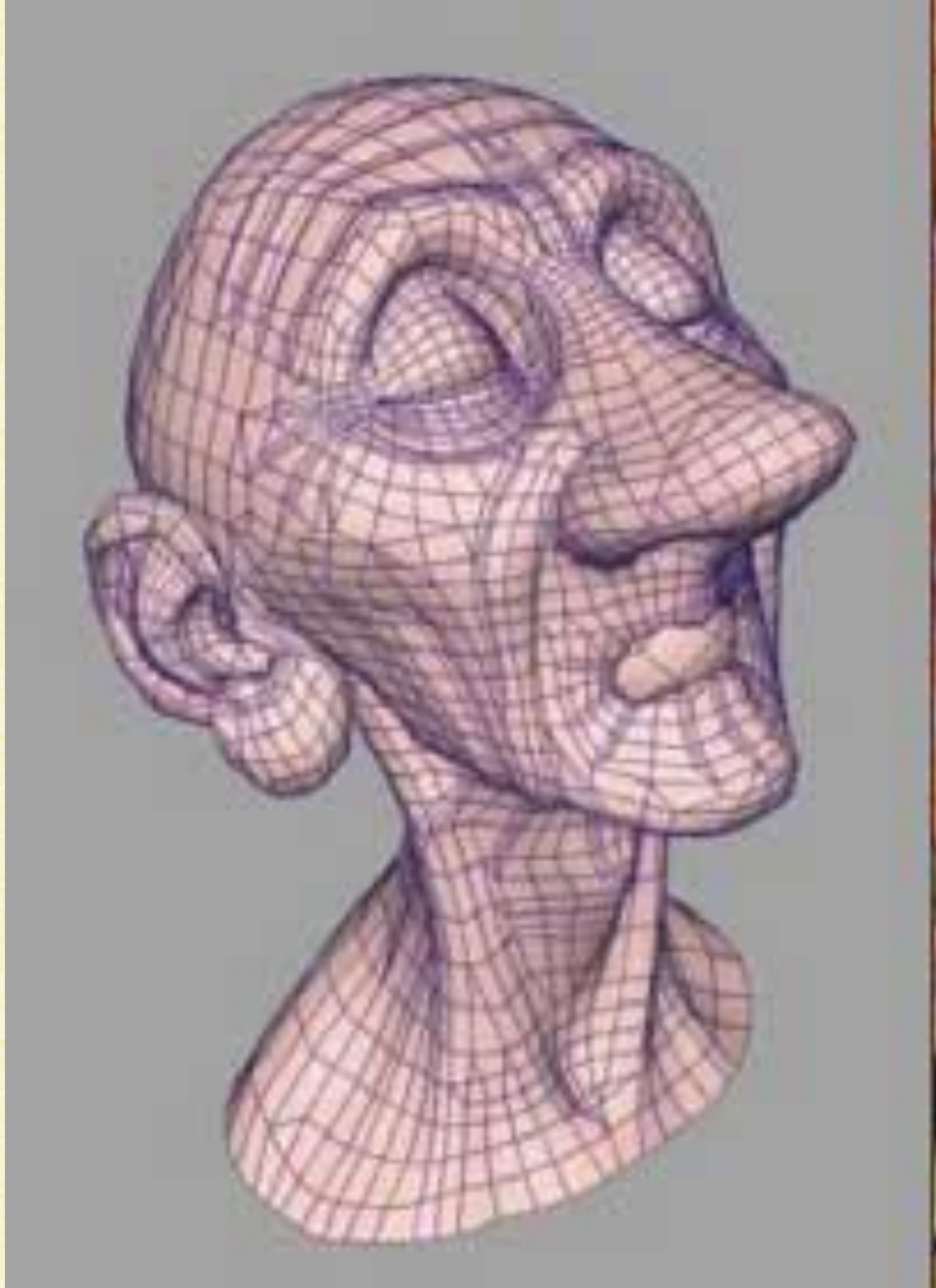
Geometric Modeling

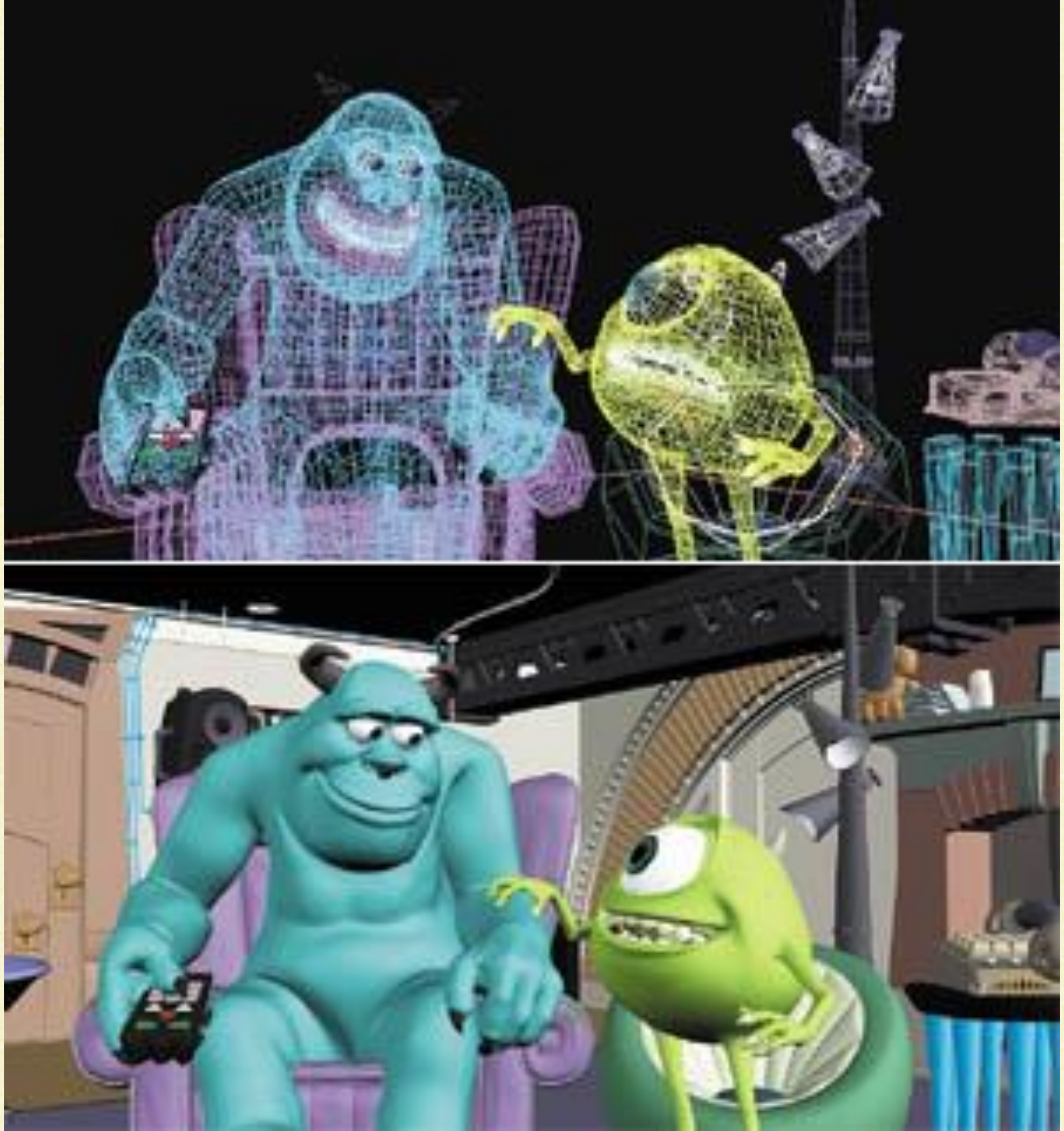


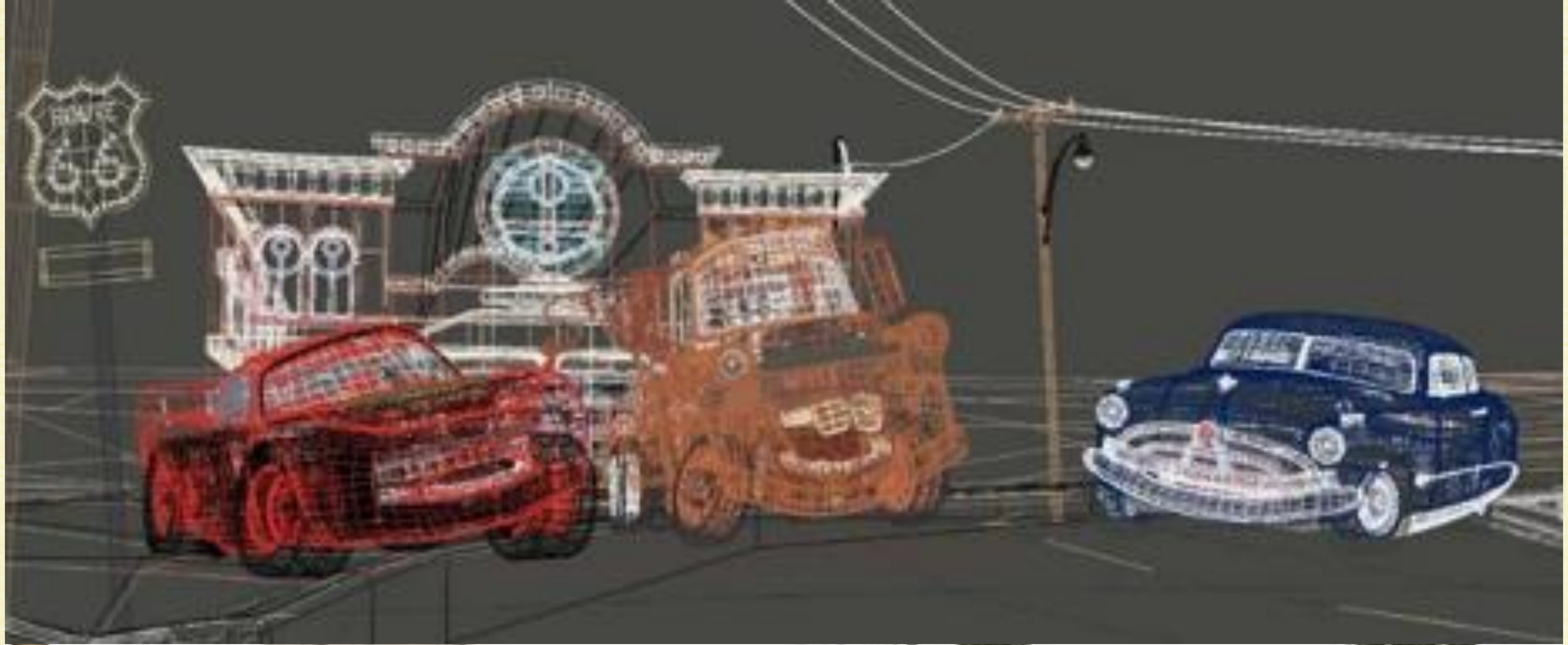
Examples







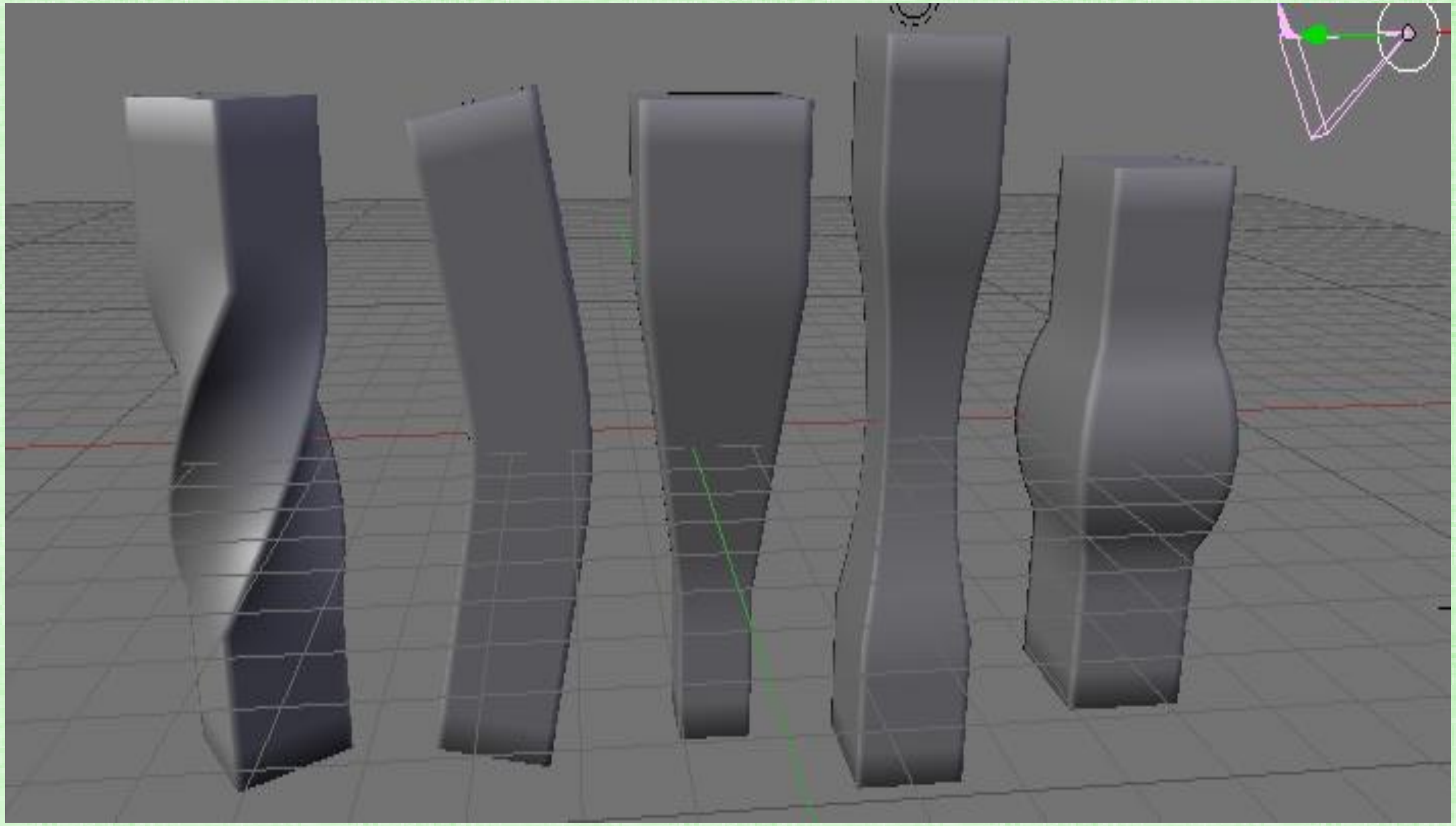



ROUTE
66

66

Mesh Editing

- Goal: manipulate only a few control points, while an algorithm procedurally deforms the mesh
  - e.g., twist, bend, stretch, etc.

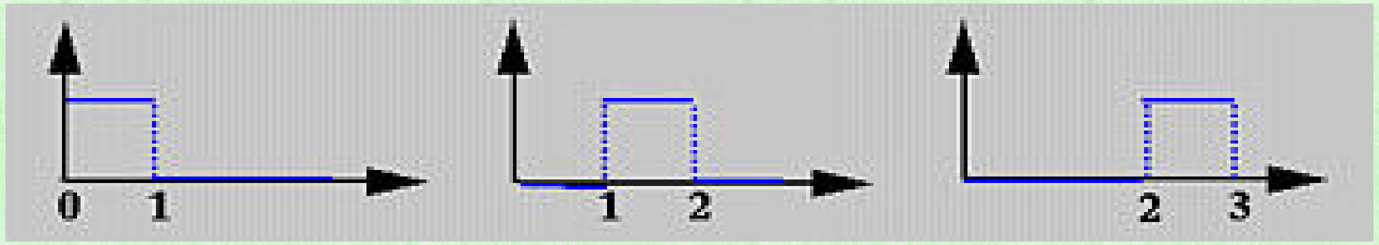


1D Hierarchical Basis Functions

- Building blocks are piecewise constant (i.e., 0th order):

$u_0 = 0, u_1 = 1, u_2 = 2$, etc. are knot locations in parameter space

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

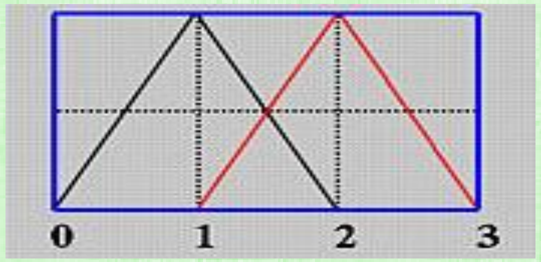


$\boldsymbol{N_{0,0}}$ $\boldsymbol{N_{1,0}}$ $\boldsymbol{N_{2,0}}$

- Higher order basis functions are defined recursively
- The i-th basis function of order p is:

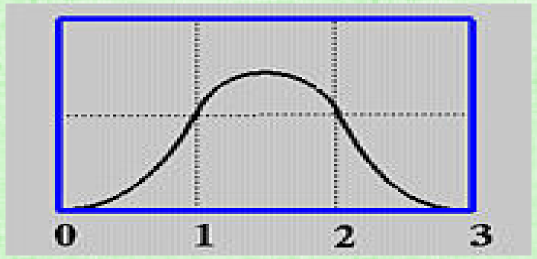
$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

1st order:



$\boldsymbol{N_{0,1}}$ $\boldsymbol{N_{1,1}}$

2nd order:



$\boldsymbol{N_{0,2}}$

Details for 1st Order

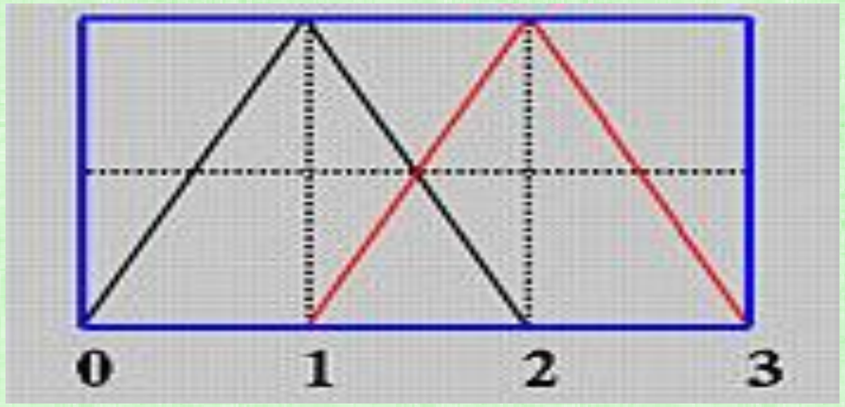
- Recursive Formula: $N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u)$
- Substitute $p = 1$: $N_{i,1}(u) = \frac{u-u_i}{u_{i+1}-u_i} N_{i,0}(u) + \frac{u_{i+2}-u}{u_{i+2}-u_{i+1}} N_{i+1,0}(u)$
- Denominators = 1: $N_{i,1}(u) = (u-u_i)N_{i,0}(u) + (u_{i+2}-u)N_{i+1,0}(u)$

$i = 0$

$$\begin{aligned} N_{0,1}(u) &= (u-0)N_{0,0}(u) + (2-u)N_{1,0}(u) \\ &= u \quad \text{in } [0,1] \\ &= 2-u \quad \text{in } [1,2] \end{aligned}$$

$i = 1$

$$\begin{aligned} N_{1,1}(u) &= (u-1)N_{1,0}(u) + (3-u)N_{2,0}(u) \\ &= u-1 \quad \text{in } [1,2] \\ &= 3-u \quad \text{in } [2,3] \end{aligned}$$



Details for 2nd Order

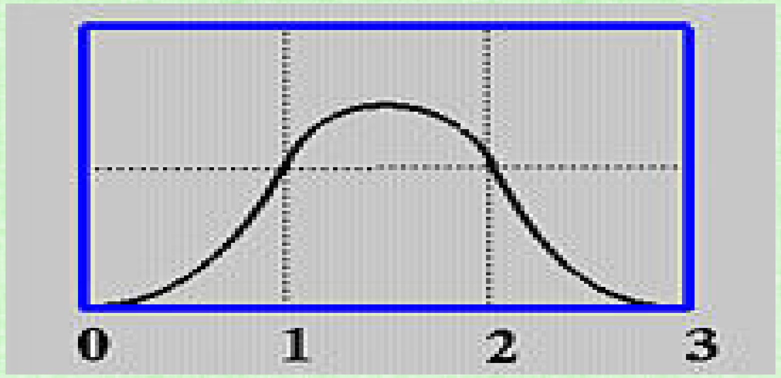
- Recursive Formula: $N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u)$
- Substitute $p = 2$: $N_{i,2}(u) = \frac{u-u_i}{u_{i+2}-u_i} N_{i,1}(u) + \frac{u_{i+3}-u}{u_{i+3}-u_{i+1}} N_{i+1,1}(u)$
- Denominators = 2: $N_{i,2}(u) = \frac{1}{2}(u-u_i)N_{i,1}(u) + \frac{1}{2}(u_{i+3}-u)N_{i+1,1}(u)$

$i = 0$

$$\begin{aligned} N_{0,2}(u) &= \frac{1}{2}(u-0)N_{0,1}(u) + \frac{1}{2}(3-u)N_{1,1}(u) \\ &= \frac{1}{2}uu && \text{in } [0,1] \\ &= \frac{1}{2}u(2-u) + \frac{1}{2}(3-u)(u-1) && \text{in } [1,2] \\ &= \frac{1}{2}(3-u)(3-u) && \text{in } [2,3] \end{aligned}$$

$$\begin{aligned} N_{0,1}(u) &= (u-0)N_{0,0}(u) + (2-u)N_{1,0}(u) \\ &= u && \text{in } [0,1] \\ &= 2-u && \text{in } [1,2] \end{aligned}$$

$$\begin{aligned} N_{1,1}(u) &= (u-1)N_{1,0}(u) + (3-u)N_{2,0}(u) \\ &= u-1 && \text{in } [1,2] \\ &= 3-u && \text{in } [2,3] \end{aligned}$$



$\frac{1}{2}u^2$ in [0,1]

$-\left(u-\frac{3}{2}\right)^2 + \frac{3}{4}$ in [1,2]

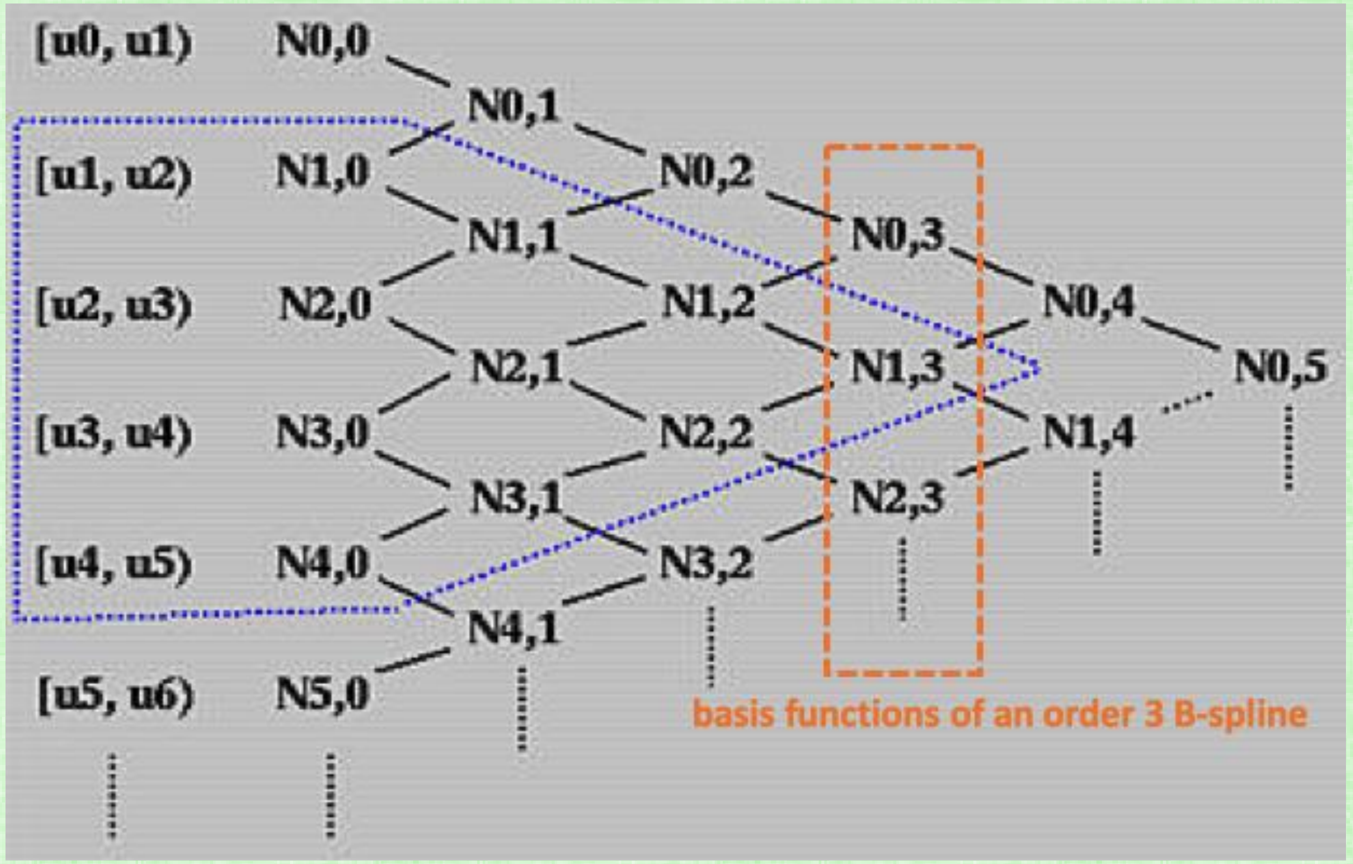
$\frac{1}{2}(3-u)^2$ in [2,3]

B-Splines

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

Counting:

- Given *k* intervals in parameter space, there are *k-p* order *p* basis functions
    - e.g. *k=6* intervals and order *p=3* gives *k-p=3* basis functions (orange box)
- Thus, an order *p* B-spline requires *k-p* control points $\boldsymbol{P}_i$



Domain Trimming

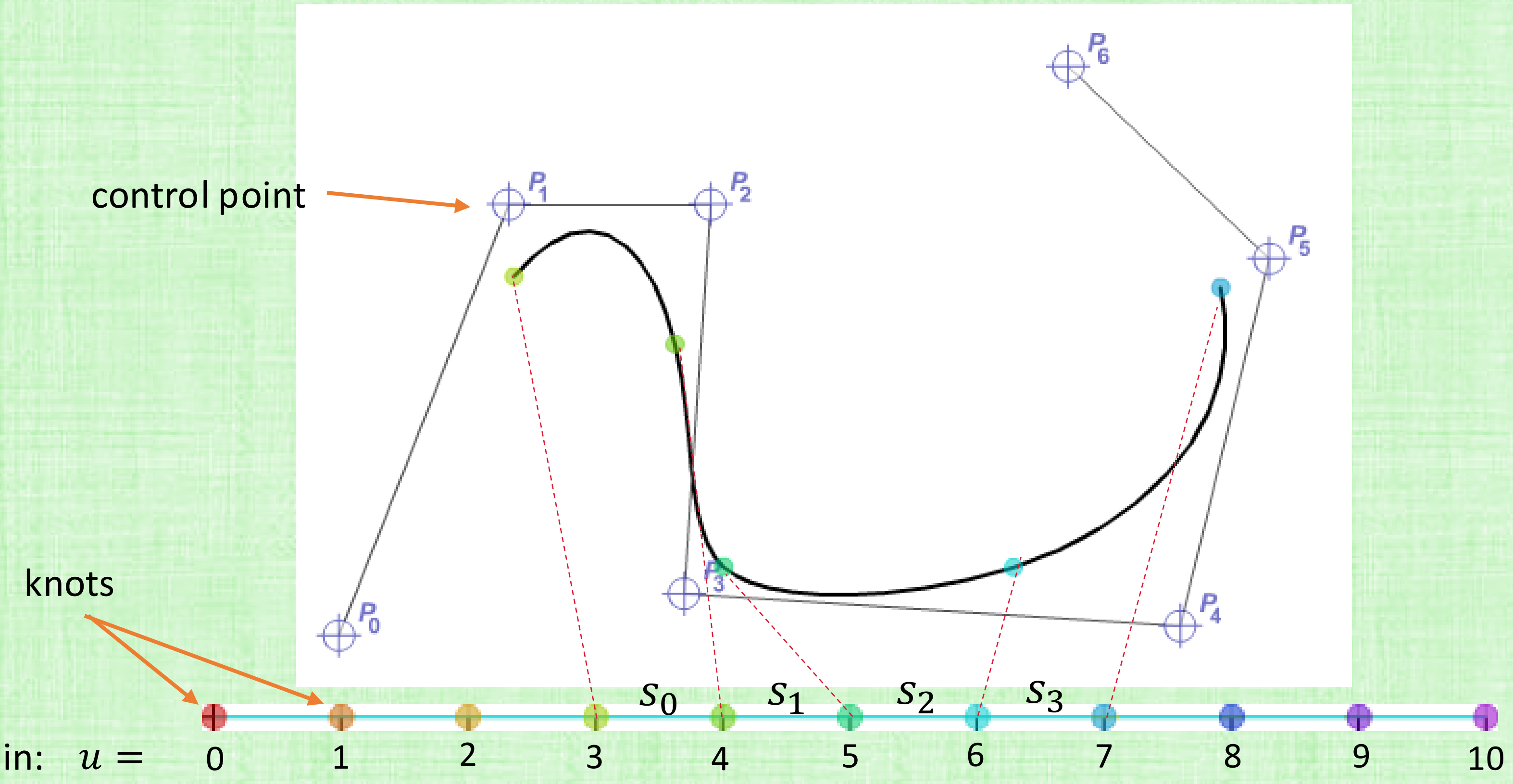
- The $N_{i,0}$ have no overlap in the parametric domain
- The $N_{i,1}$ have no overlap in the first and last intervals, but 2 of the $N_{i,1}$ overlap in each interior interval
- The $N_{i,2}$ have no overlap in the first and last intervals, 2 of the $N_{i,2}$ overlap in second and second from last intervals, and 3 overlap in all the other interior intervals
- The $N_{i,3}$ have no overlap in the first and last intervals, 2 of the $N_{i,3}$ overlap in second and second from last intervals, 3 overlap in third and third from last intervals, and 4 overlap in all the other interior intervals
- Etc…

- For consistency of behavior, these special intervals (which behave a bit differently) are ignored in the B-Spline curve $C(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P}_i$
- That is, $C(u)$ is only defined on the interior of the parametric domain

Cubic B-Splines

Counting:

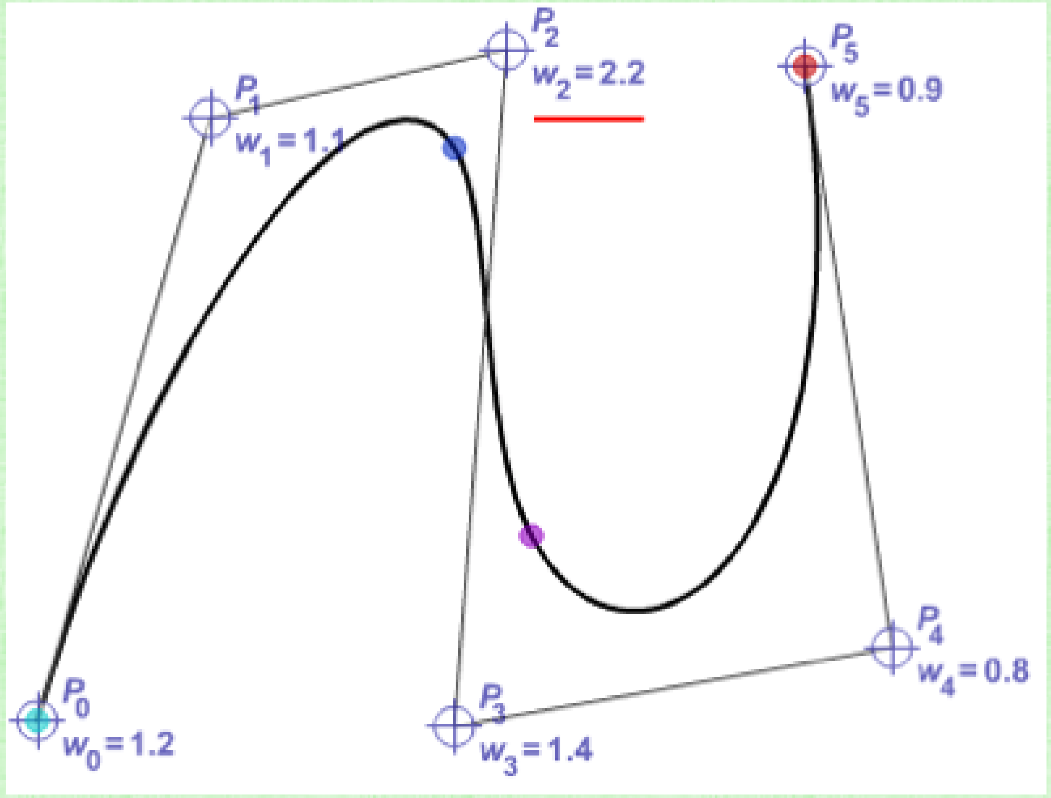
10 intervals
order 3 (cubic)
10-3=7 control points

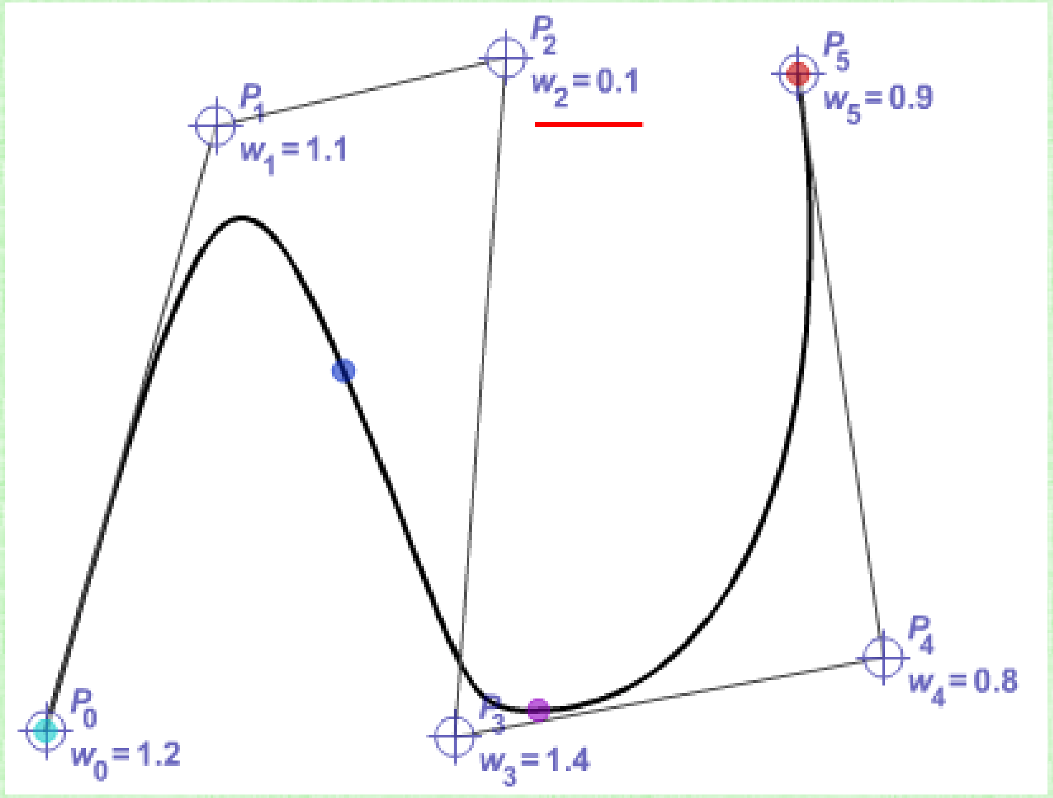


- Domain Trimming: the first and last 3 parametric intervals are ignored
- 4 control points define the shape of each curve segment
  - e.g., interval s_0 is given by $u \in [3,4]$ and controlled by $\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$

NURBS Curves

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

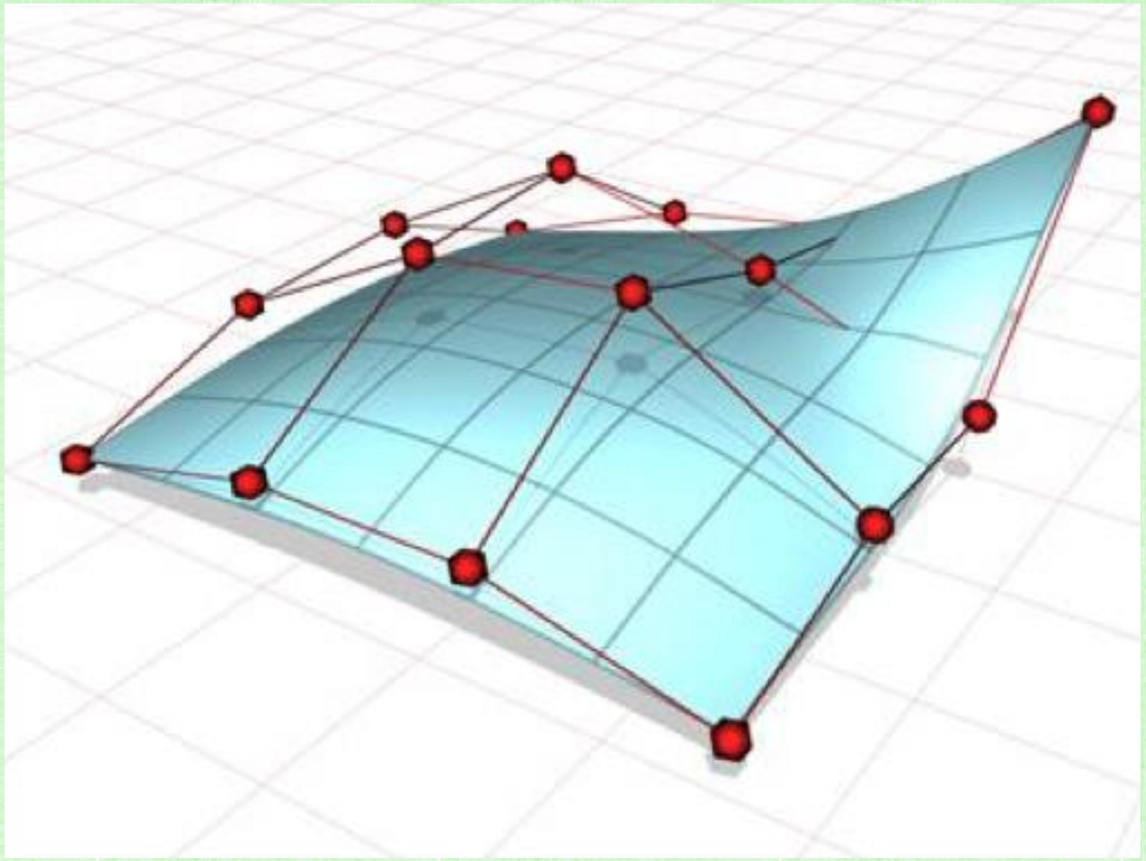




NURBS Surfaces

- Extending the ideas from curves to surfaces:

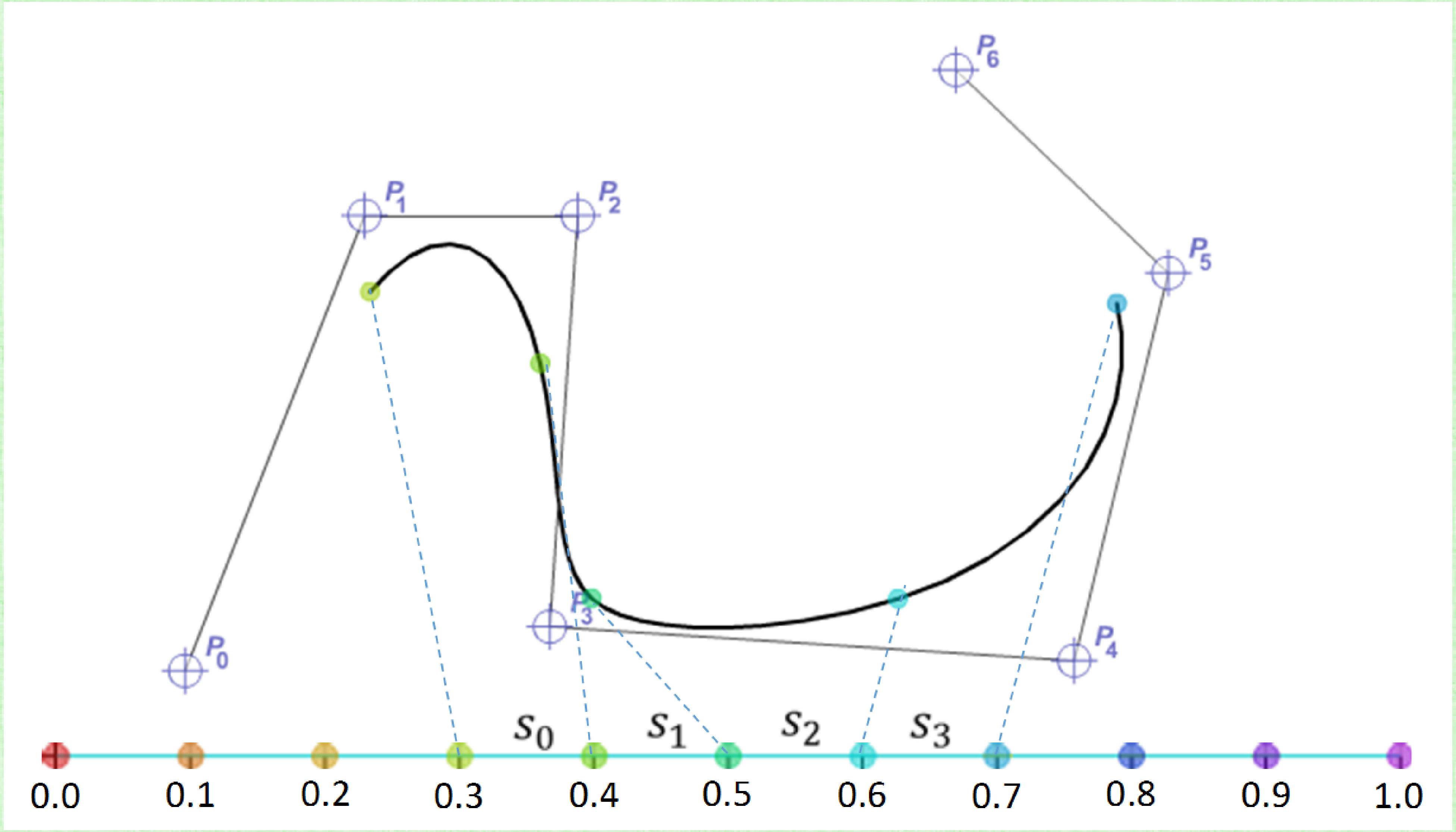
$$S(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n} \left(\frac{w_{i,j}}{\sum_{k_j=0}^{m} \sum_{k_i=0}^{n} w_{k_i,k_j} N_{k_i,p}(u) N_{k_j,q}(v)} \right) N_{i,p}(u) N_{j,q}(v) \boldsymbol{P}_{i,j}$$



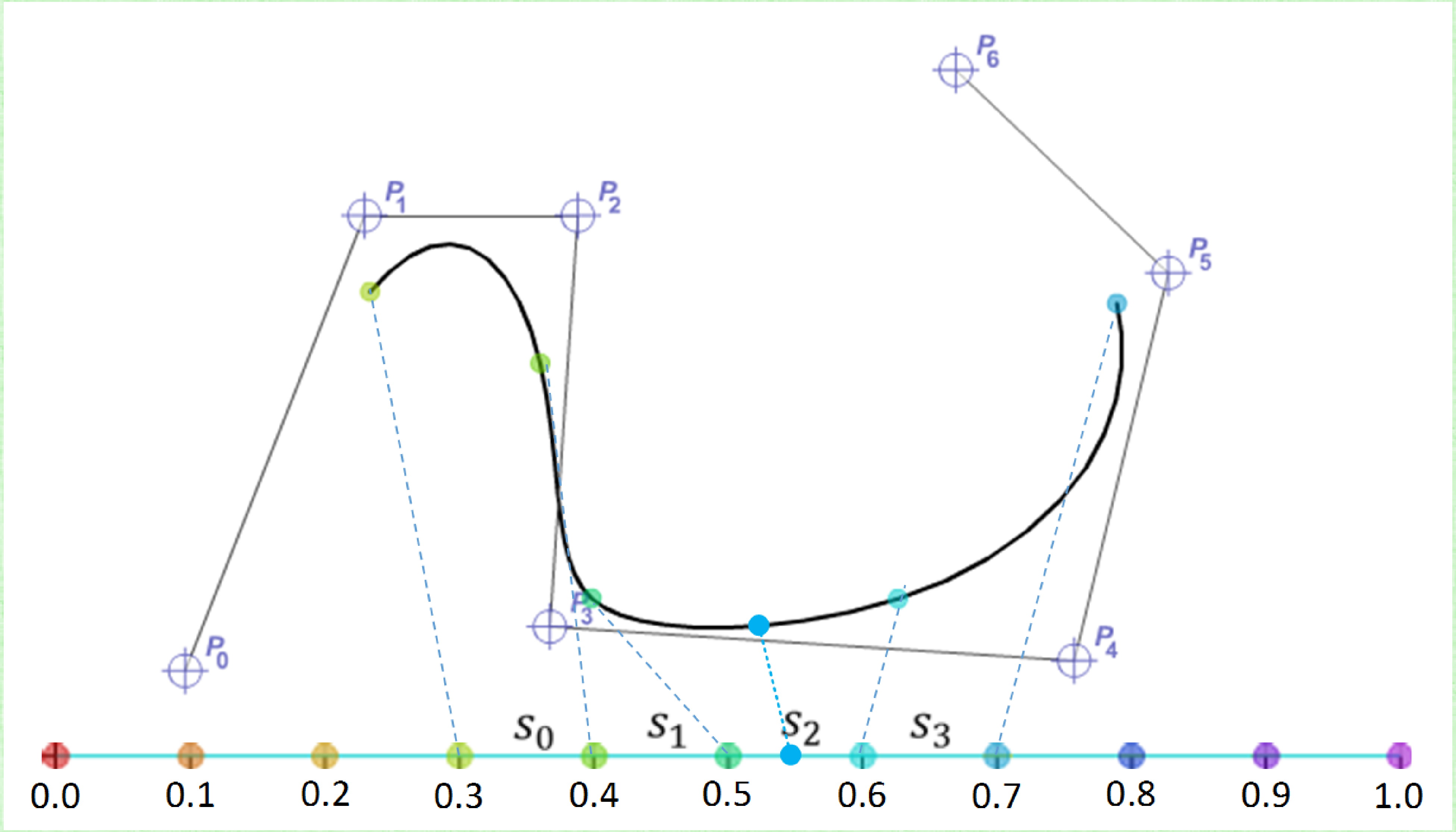
Adding Finer Details

- 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

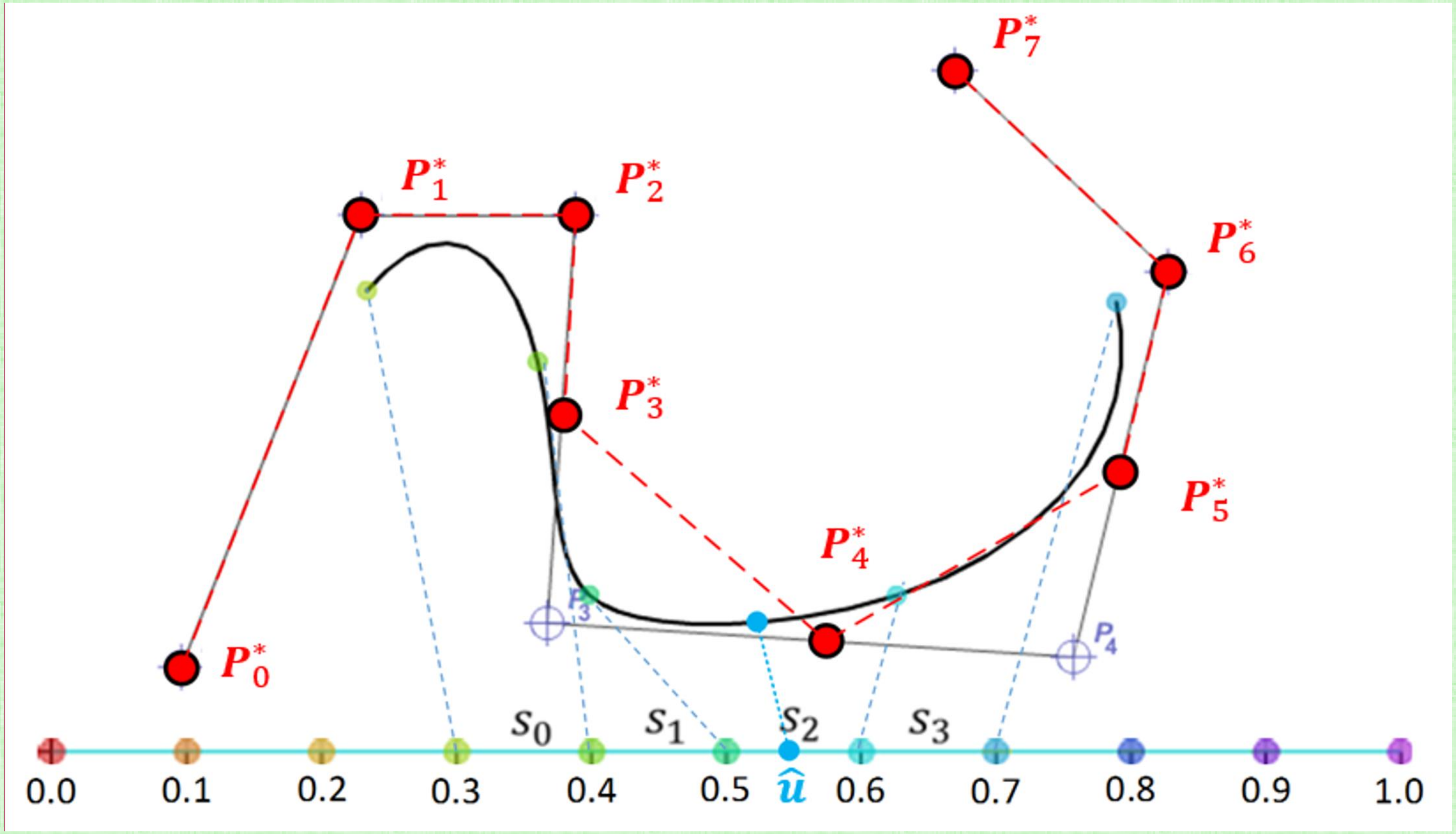
Original Curve



Insert a Knot

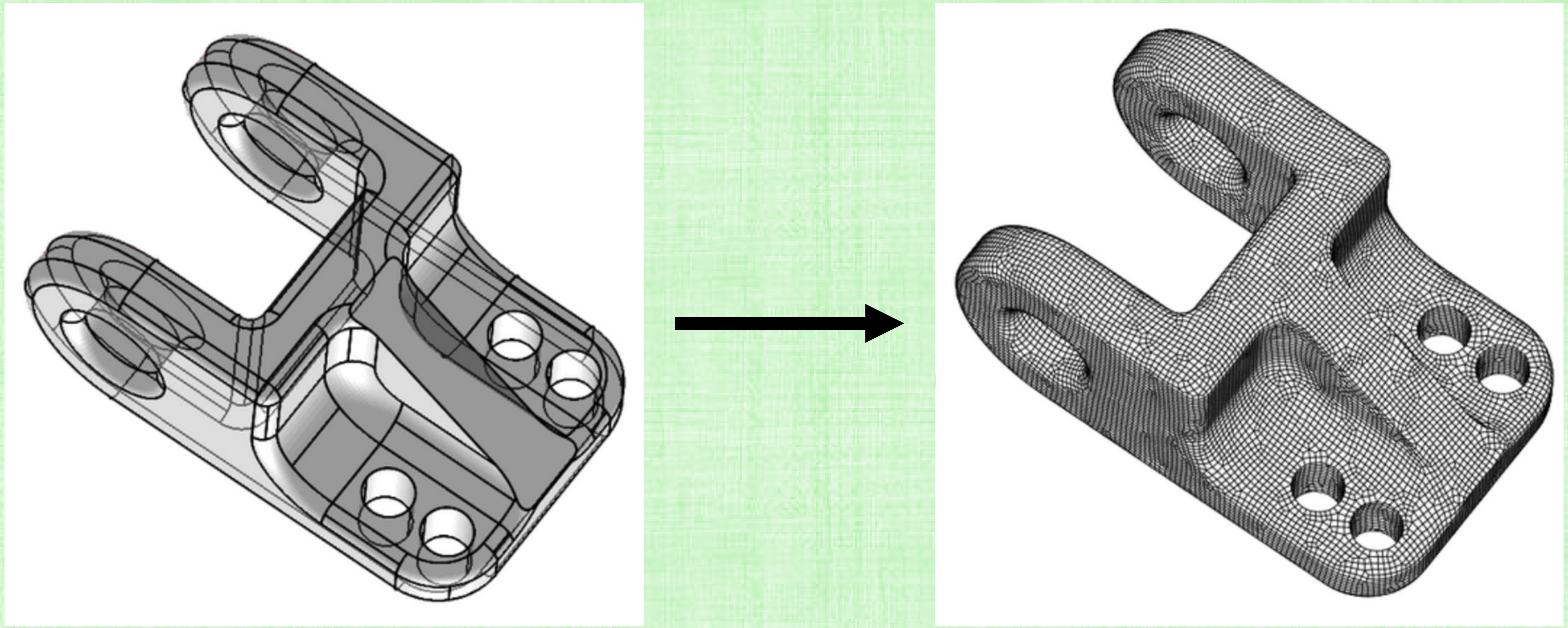


Add/Adjust Control Points



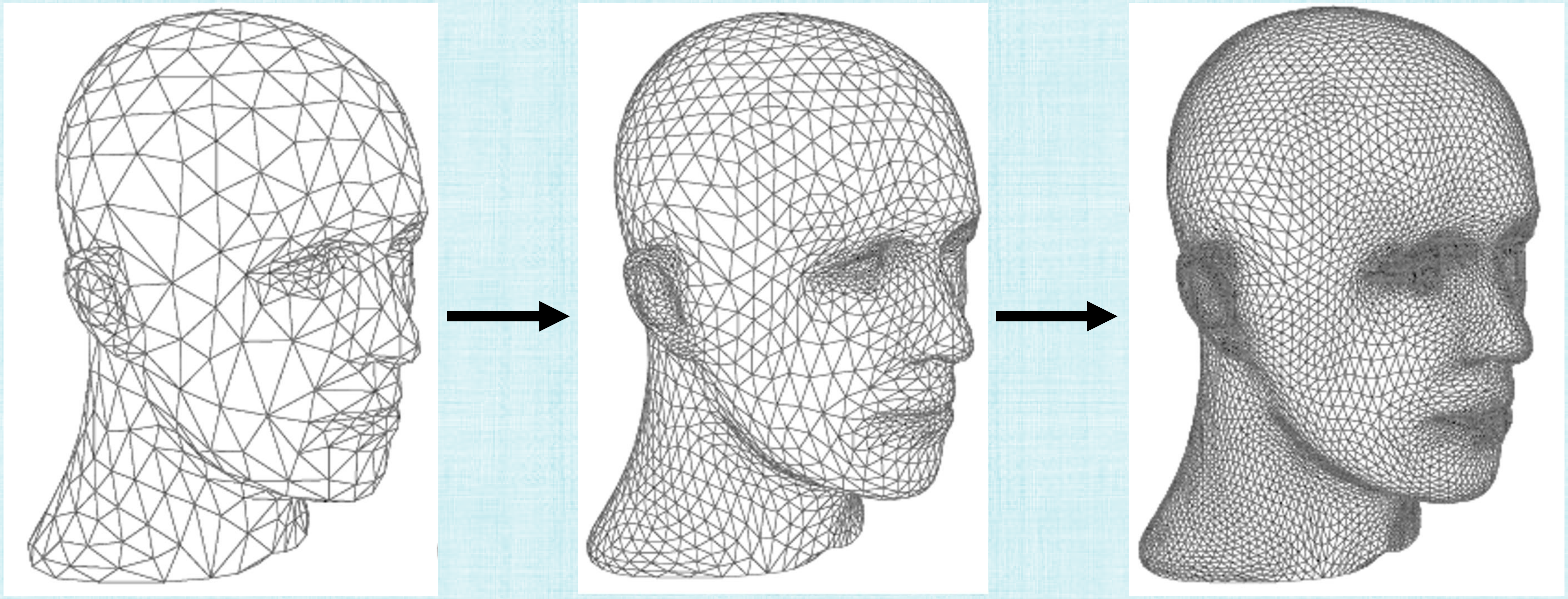
Converting NURBS to Triangles

- Evaluate $S(u_i, v_j)$ for all the points on a 2D grid of u and v values in parameter space
- Maintain the grid connectivity in the 3D space to obtain quadrilaterals
- Divide each quadrilateral into two triangles



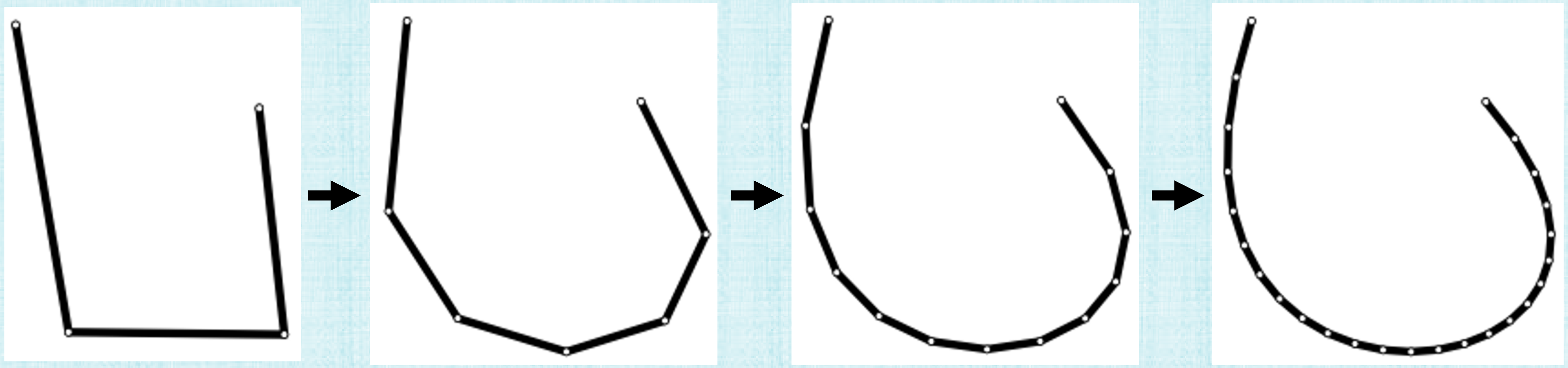
Subdivision

- An algorithm for automatically generating finer (and smoother) mesh geometry
- Can be done repeatedly



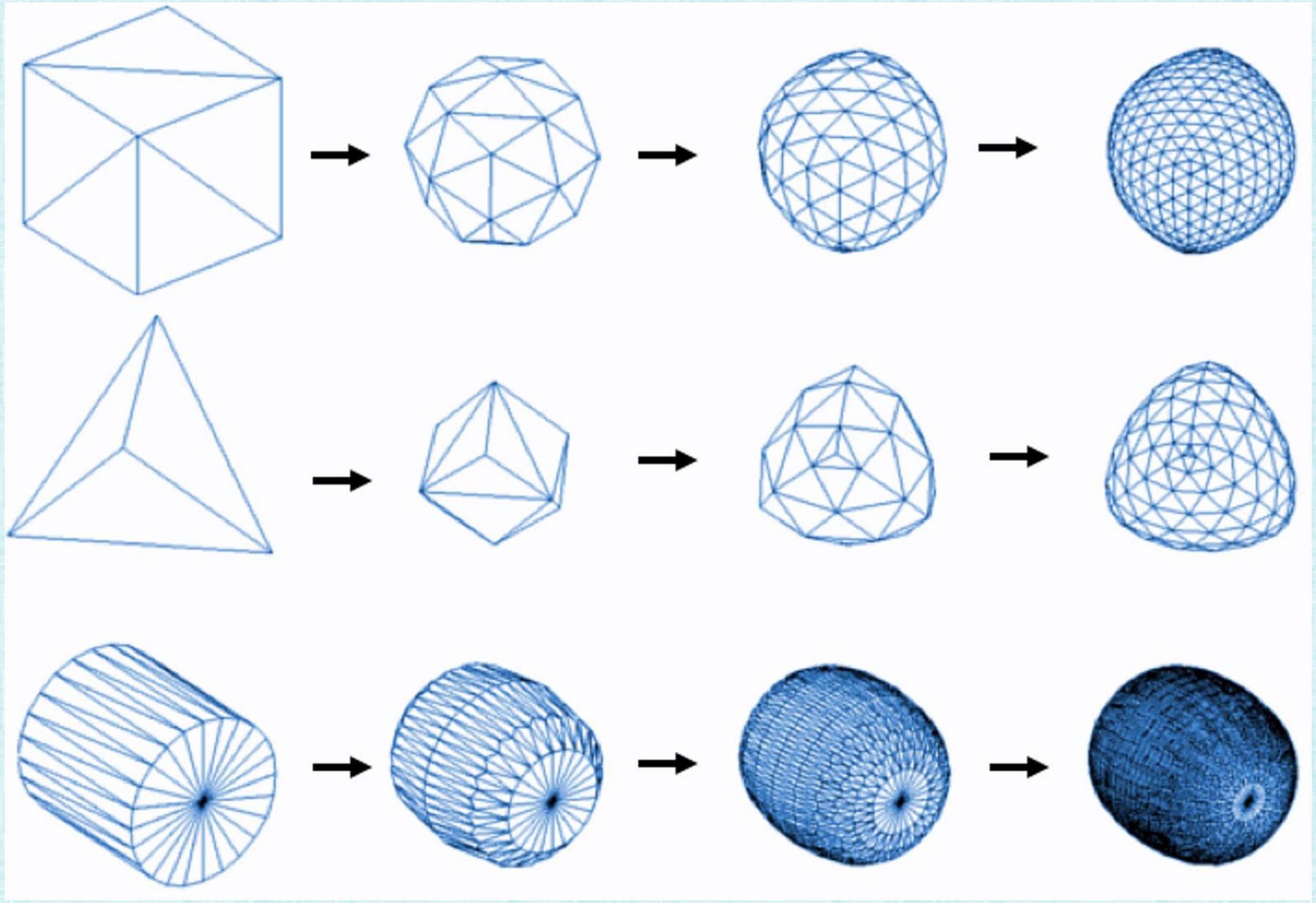
Subdivision Curves

- Subdivide each line segment into 2 sub-segments
- Move both the old and the new vertices
- Repeat…

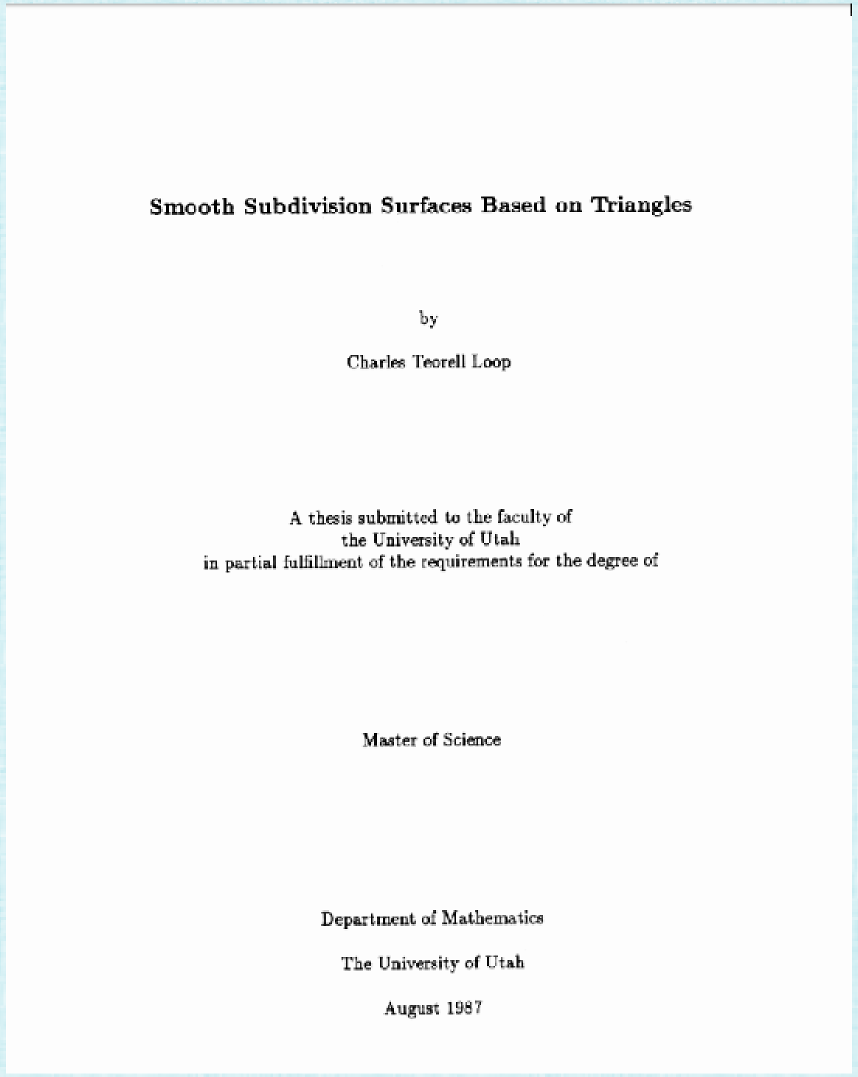


Subdivision Surfaces

- Subdivide each triangle into 4 sub-triangles
- Move both the old and the new vertices
- Repeat…



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

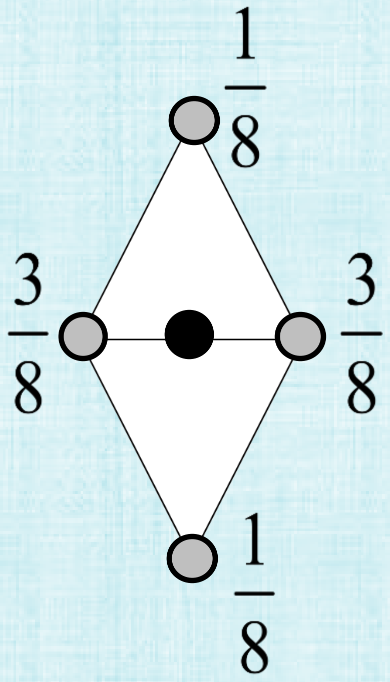
The University of Utah

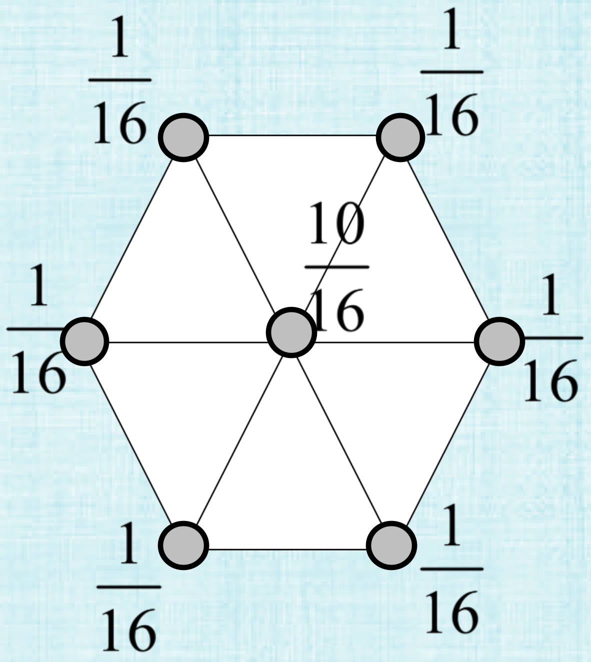
August 1987

over 2500 citations for an MS thesis!

Move Old and New Vertices

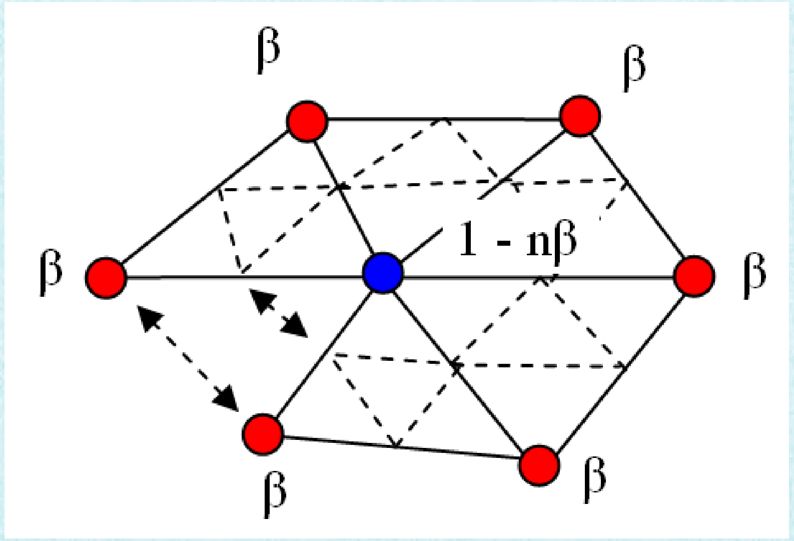
- The position of a new vertex (shown in black) is computed as a weighted average of the positions of the four nearby original old vertices (shown in grey)
- The position of a regular original vertex (shown in grey) is computed as a weighted average of the positions of the the six adjacent original vertices (also shown in grey)

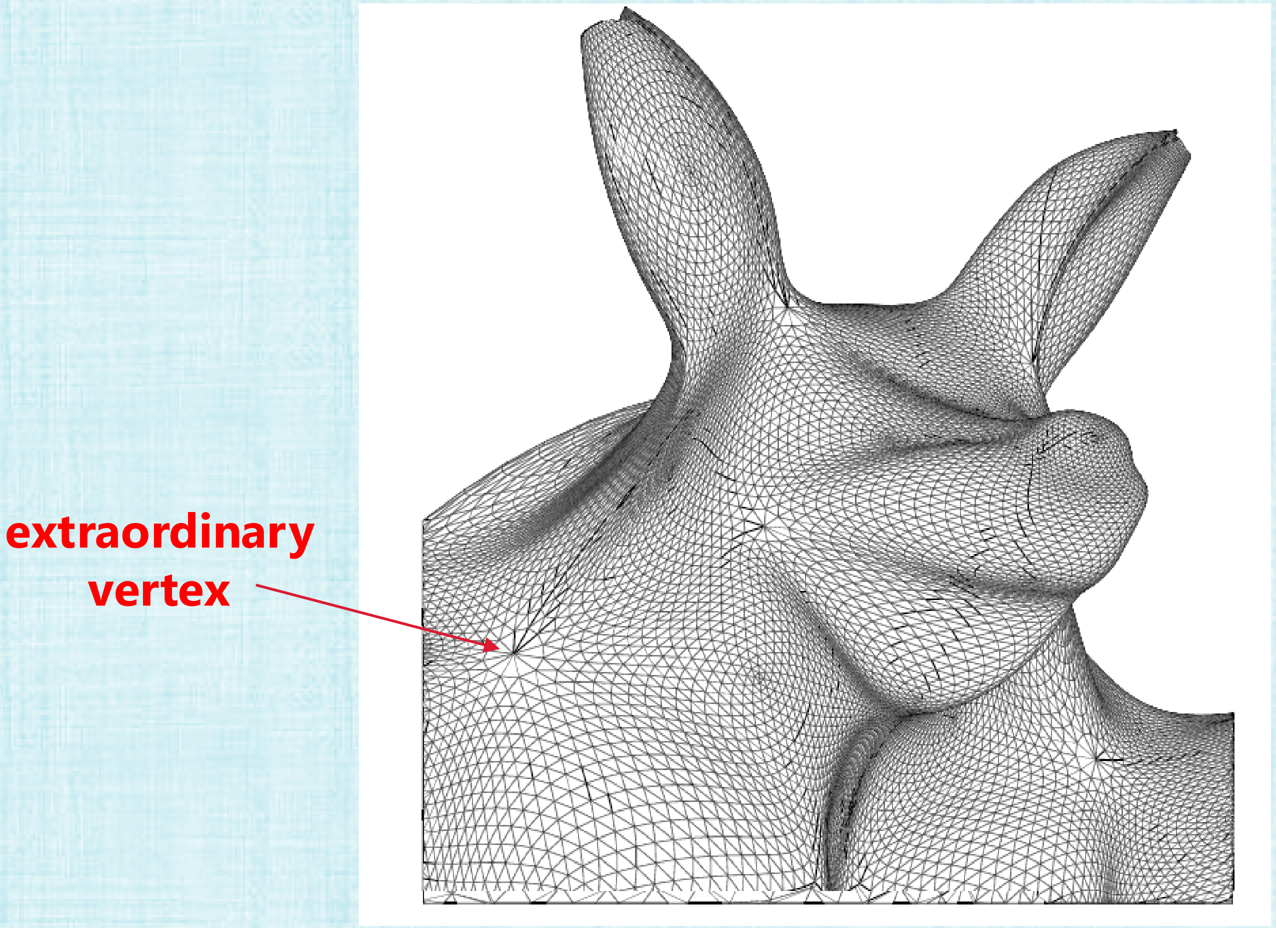




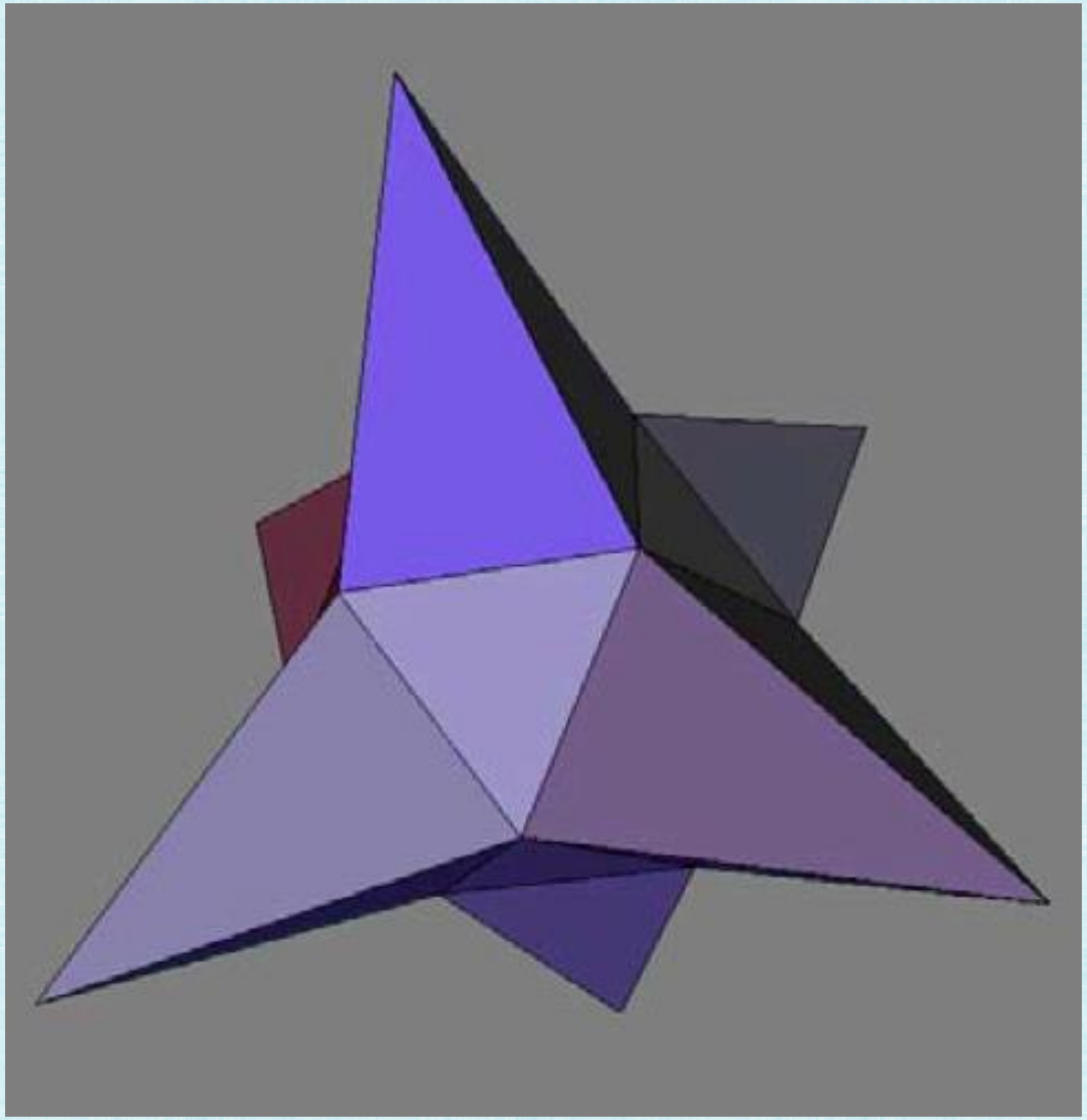
Extraordinary Vertices

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

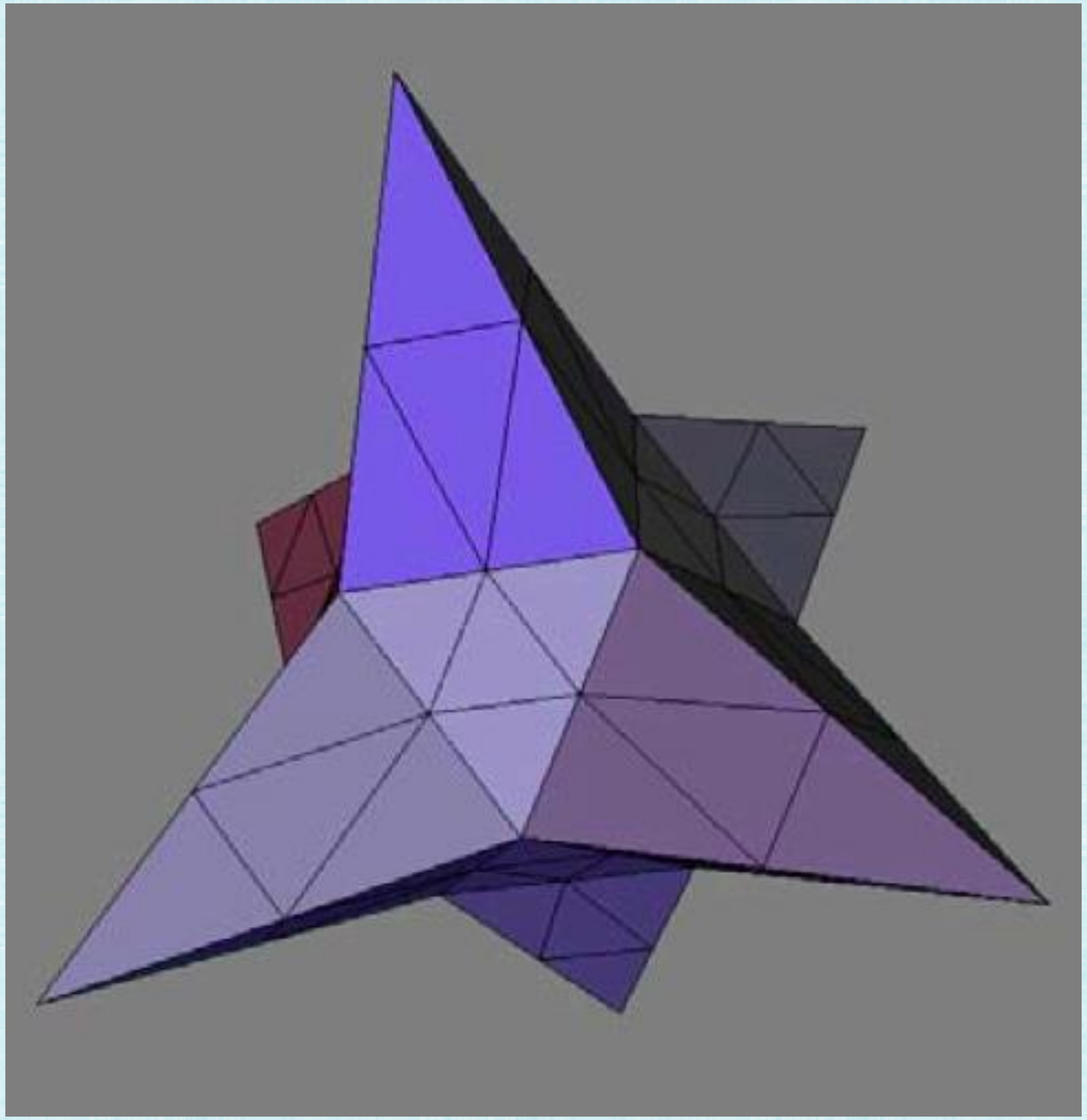




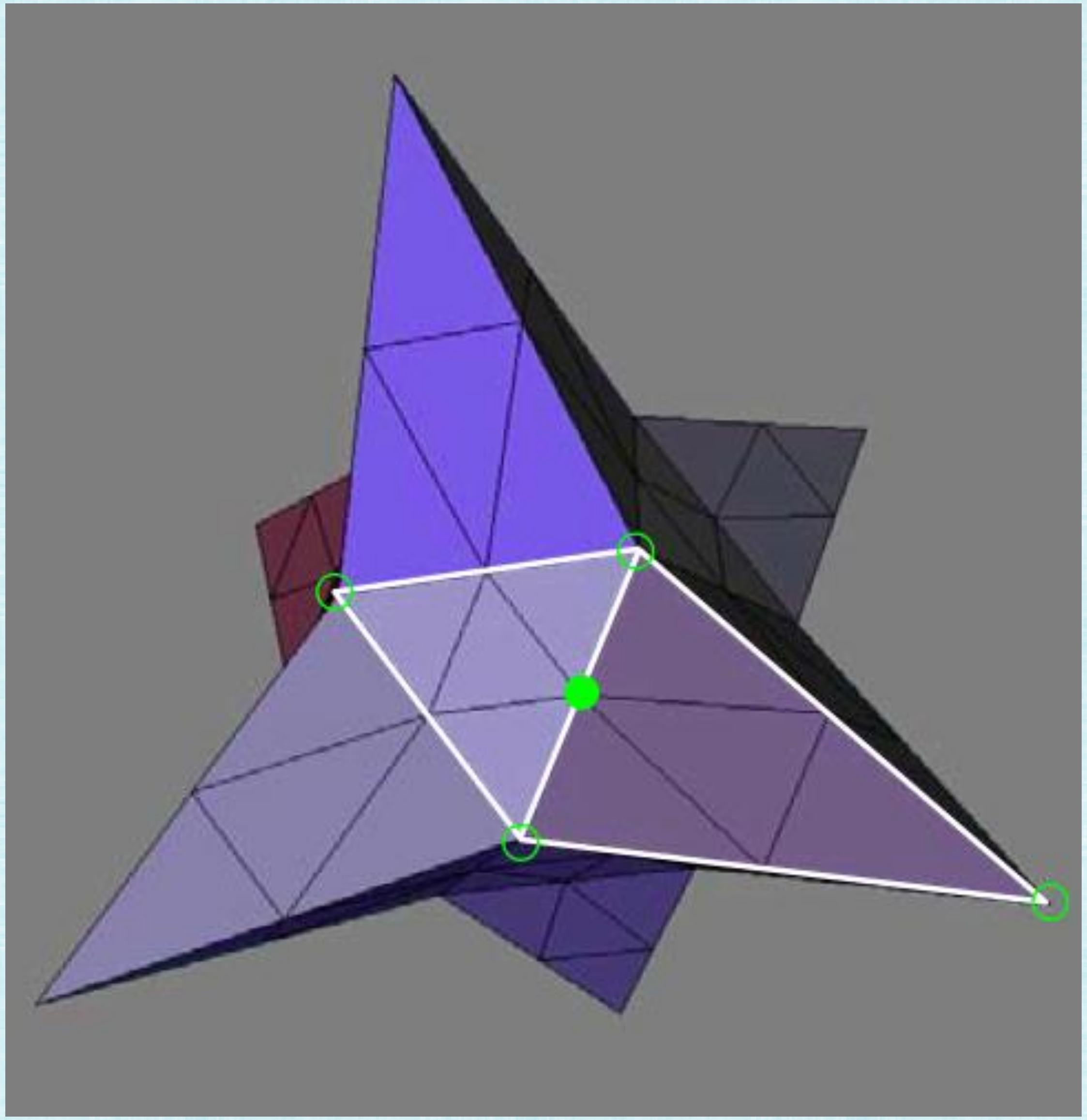
Initial Mesh



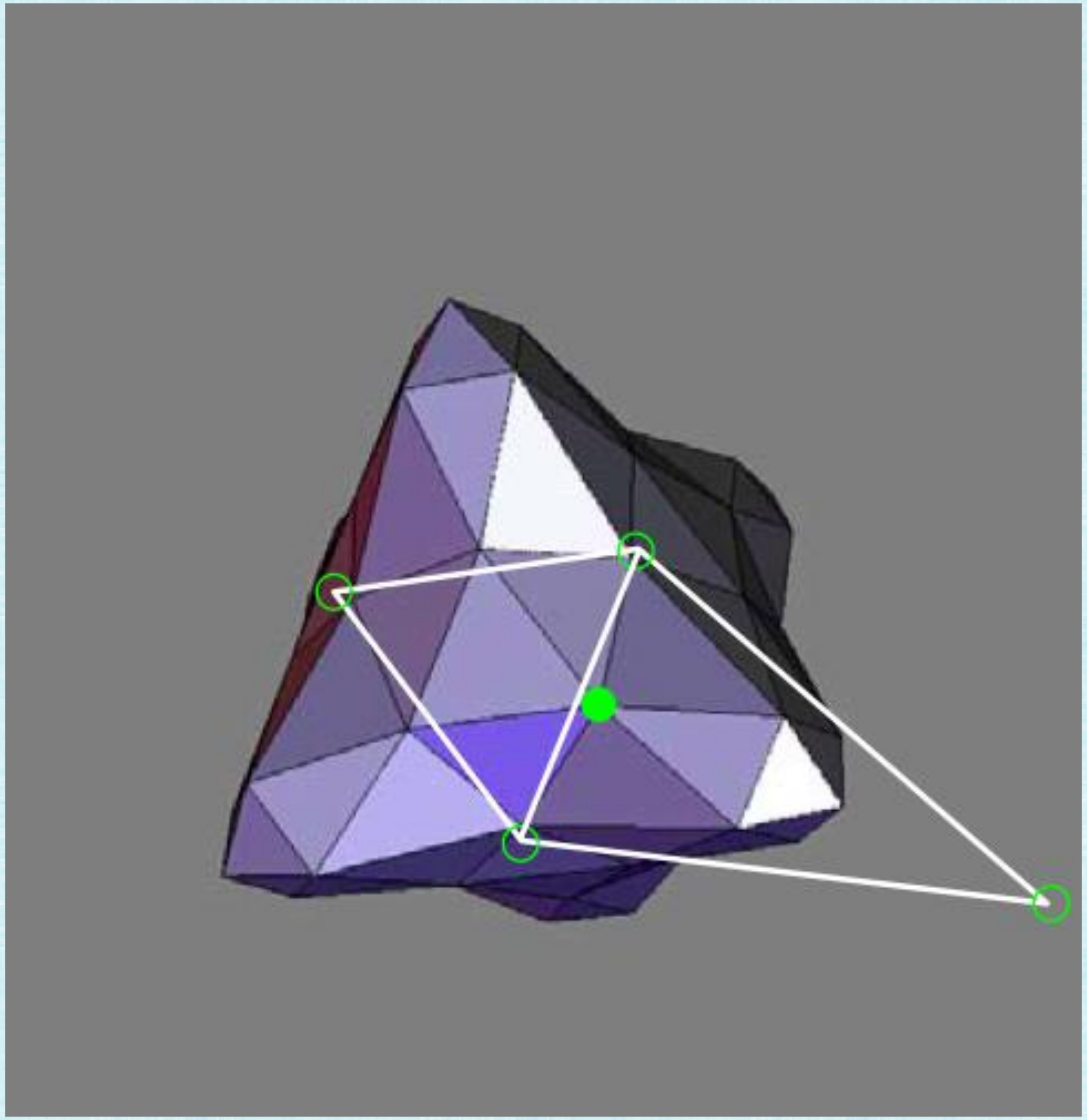
Add New Vertices



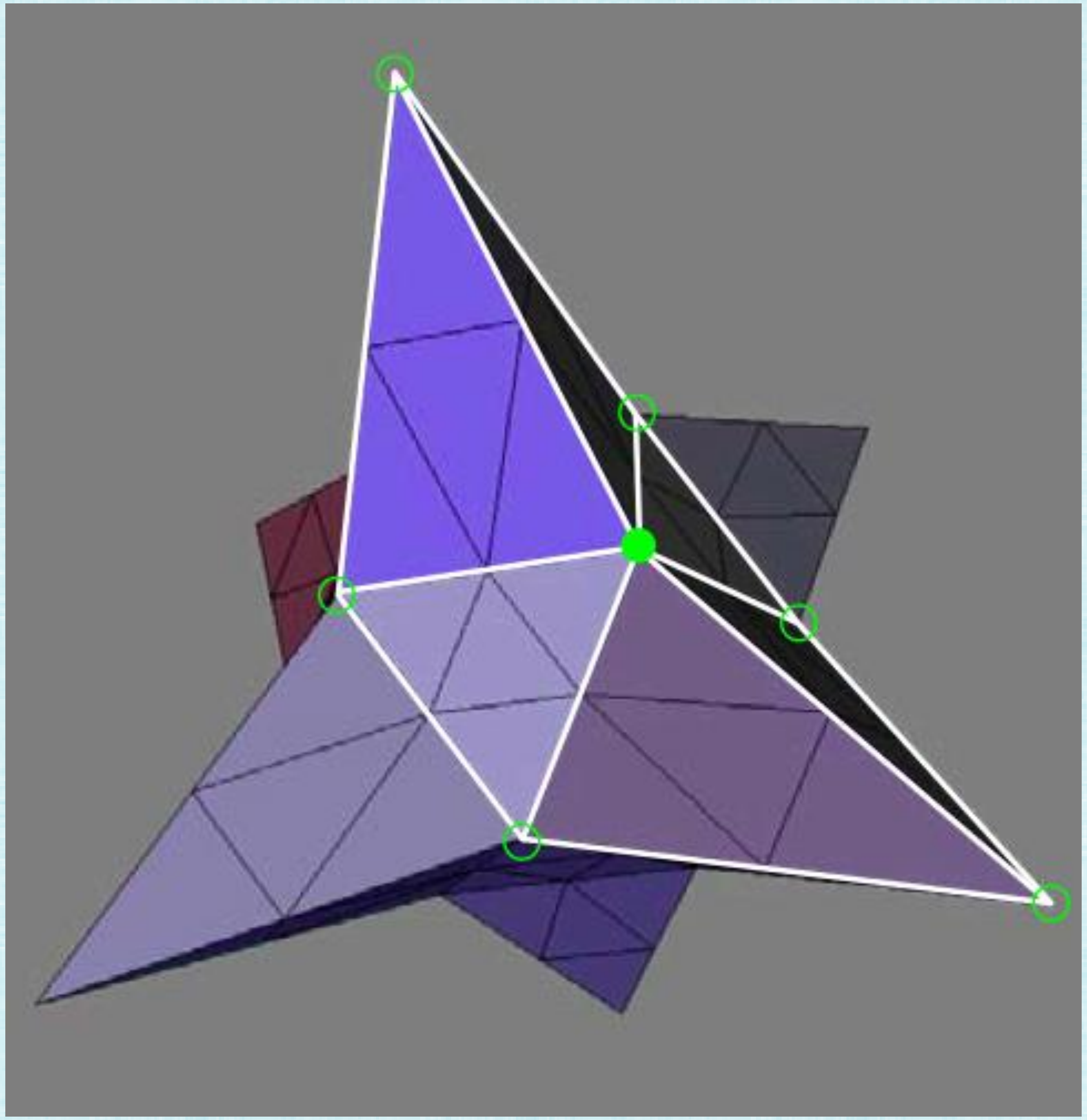
A New Vertex and Its Stencil



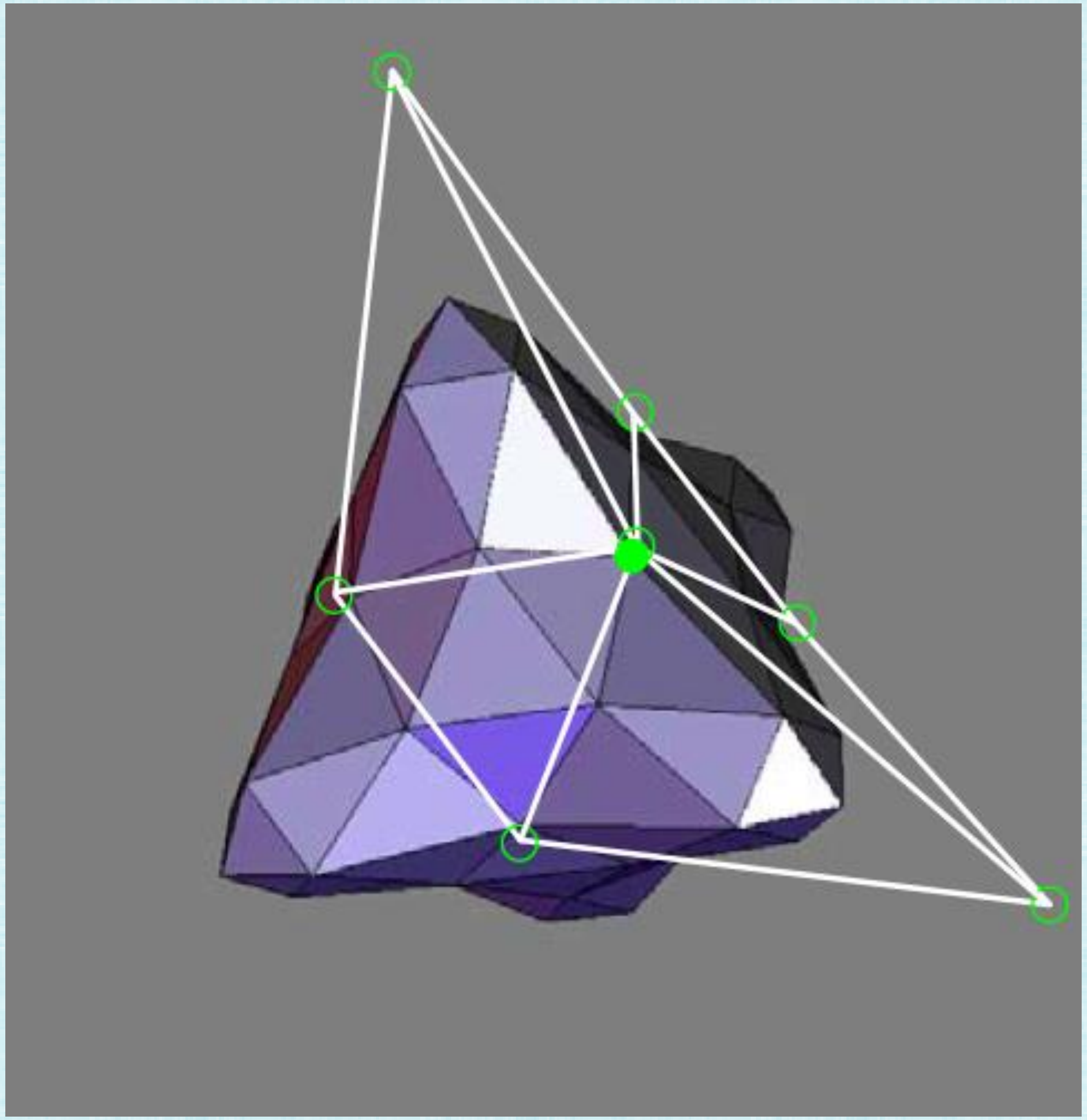
Move the New Vertex



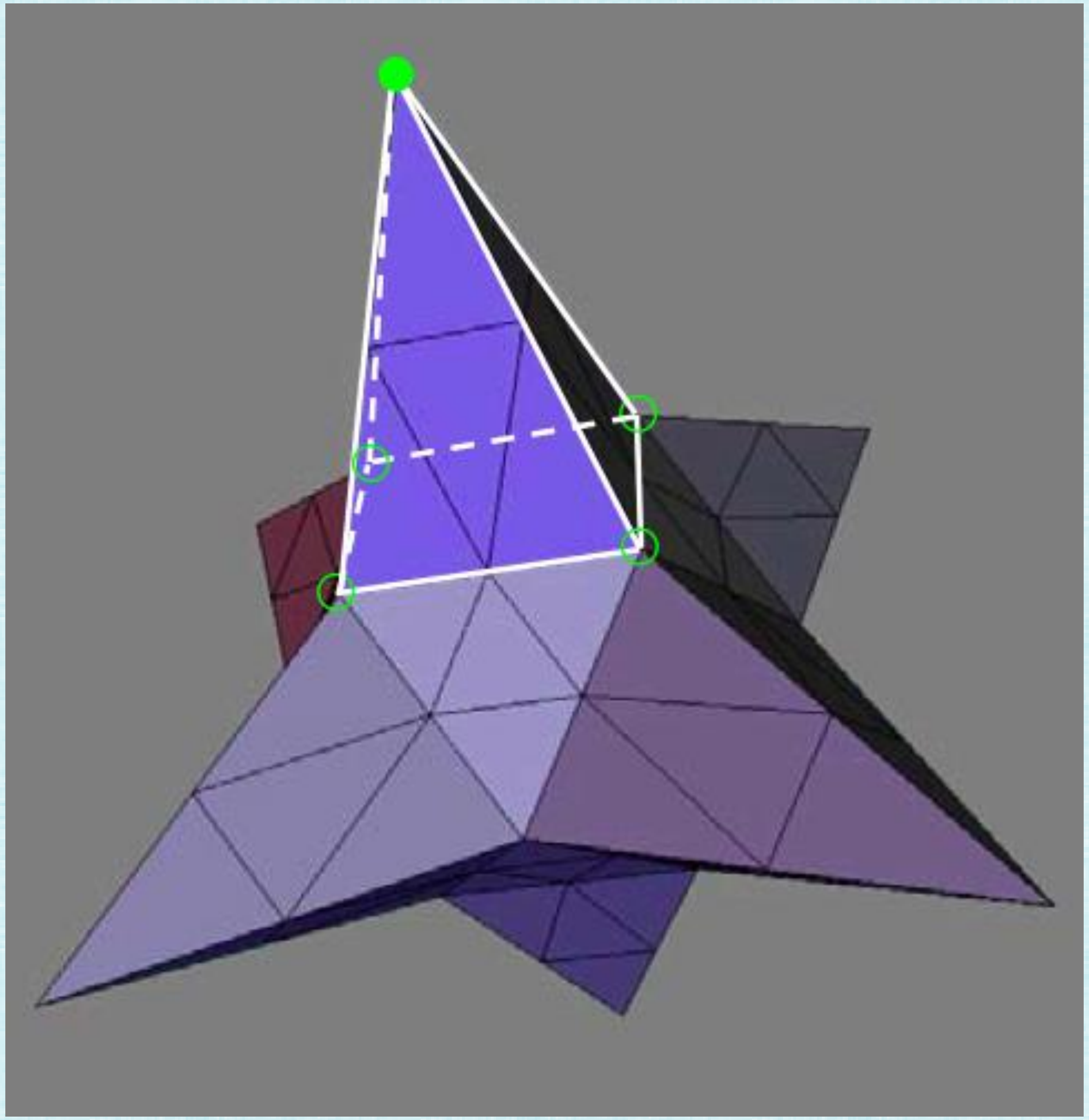
An Original (Regular) Vertex and Its Stencil



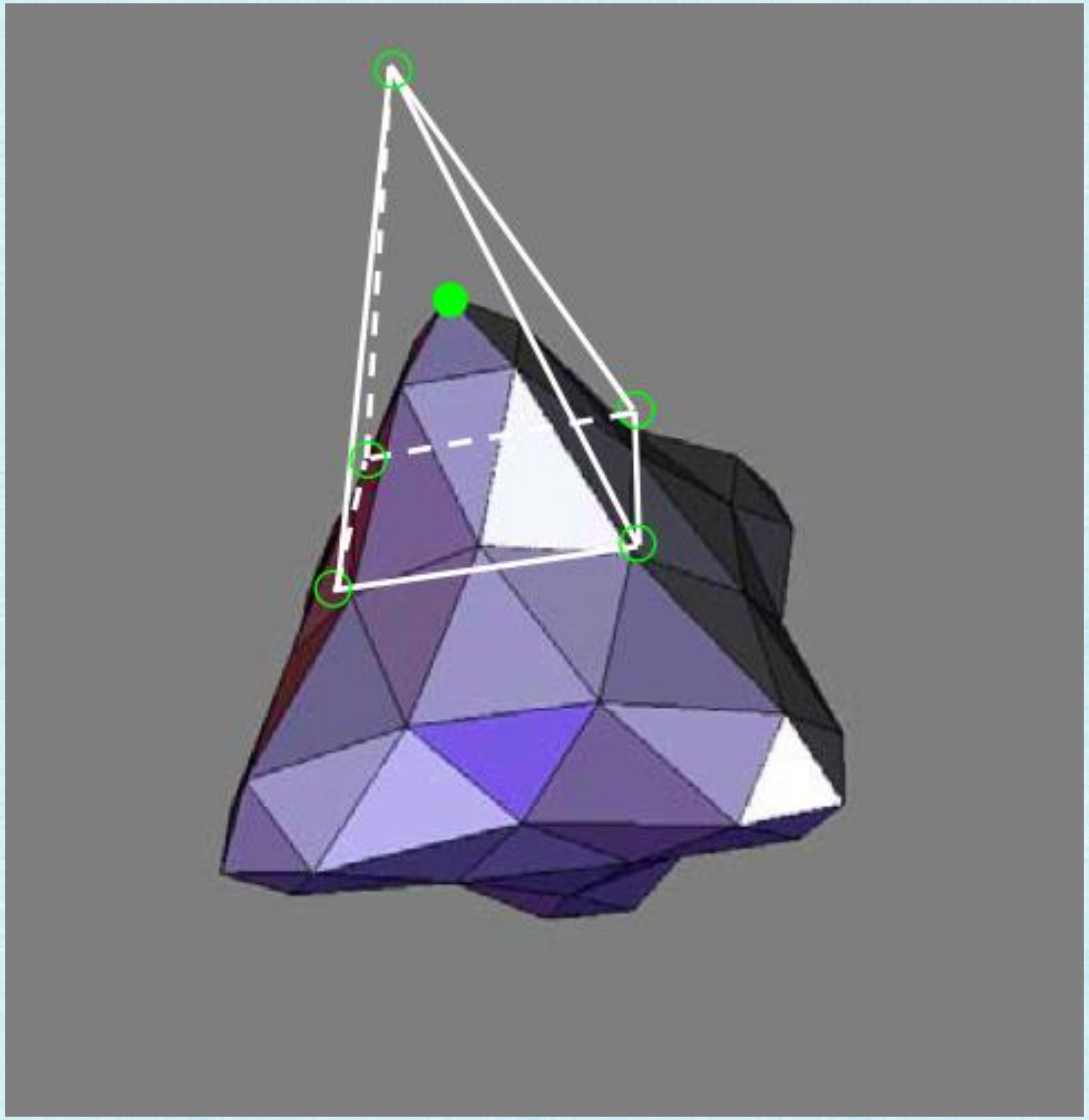
Move the Original (Regular) Vertex



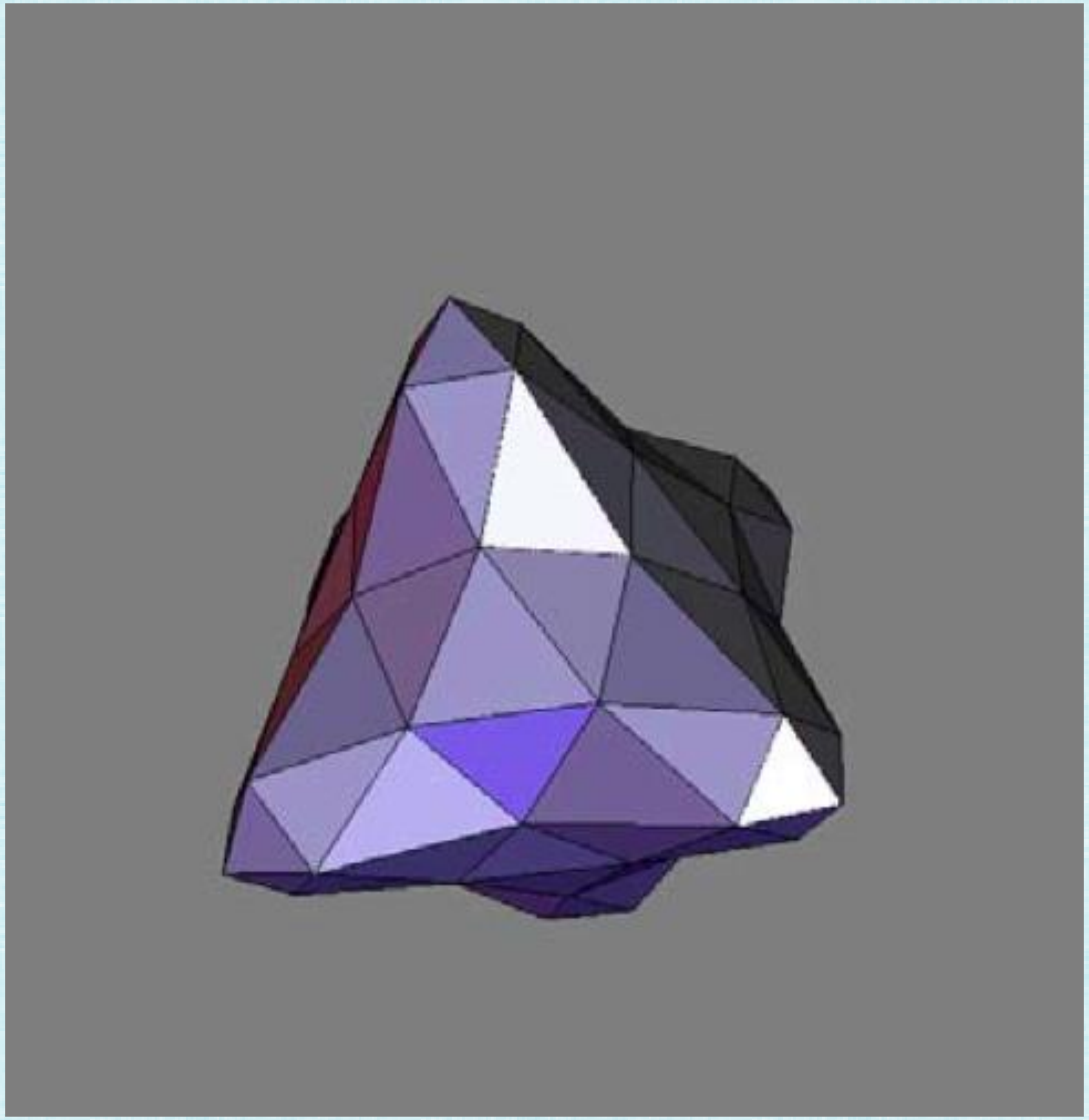
An Original (Extraordinary) Vertex and Its Stencil



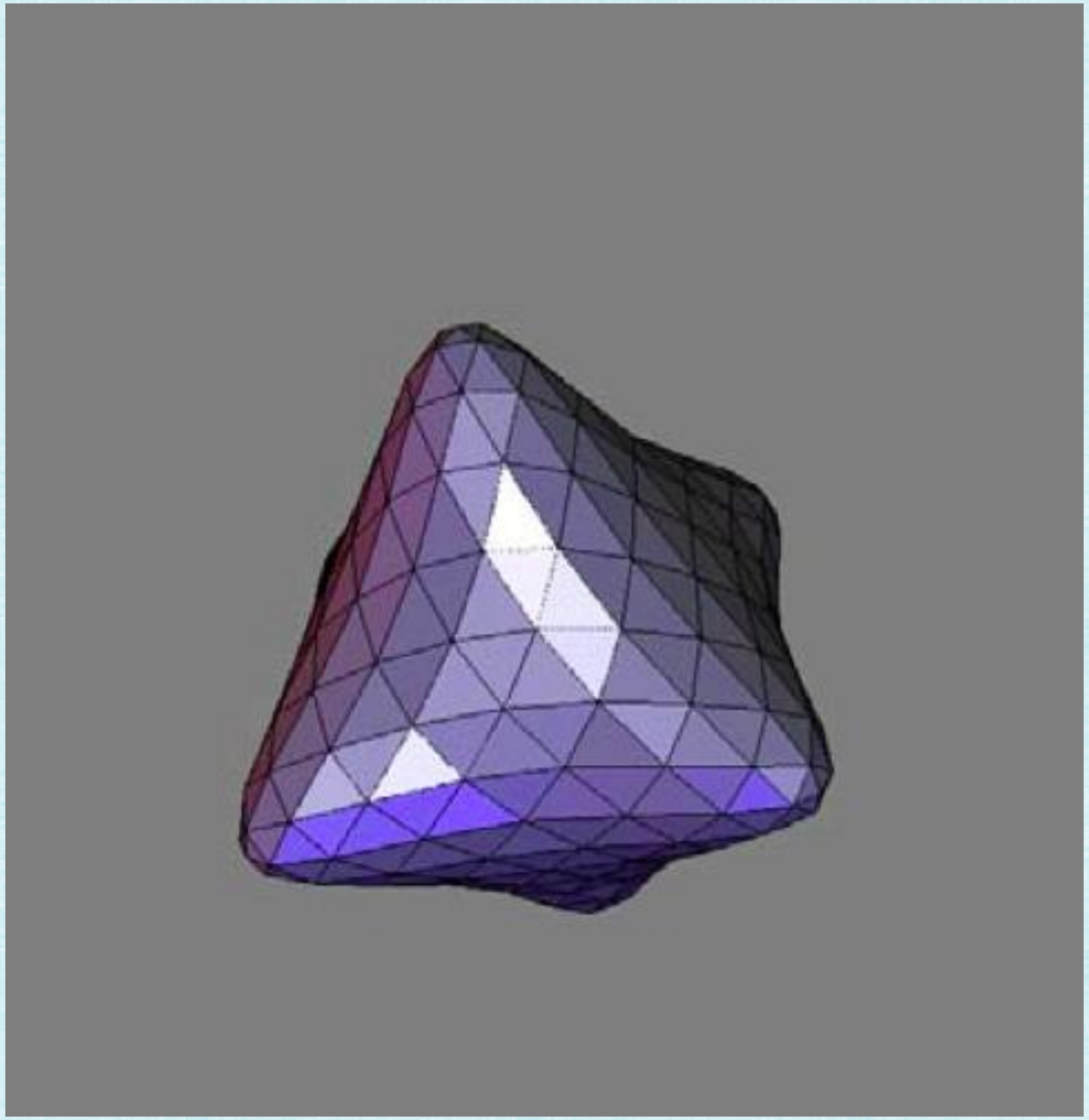
Move the Original (Extraordinary) Vertex



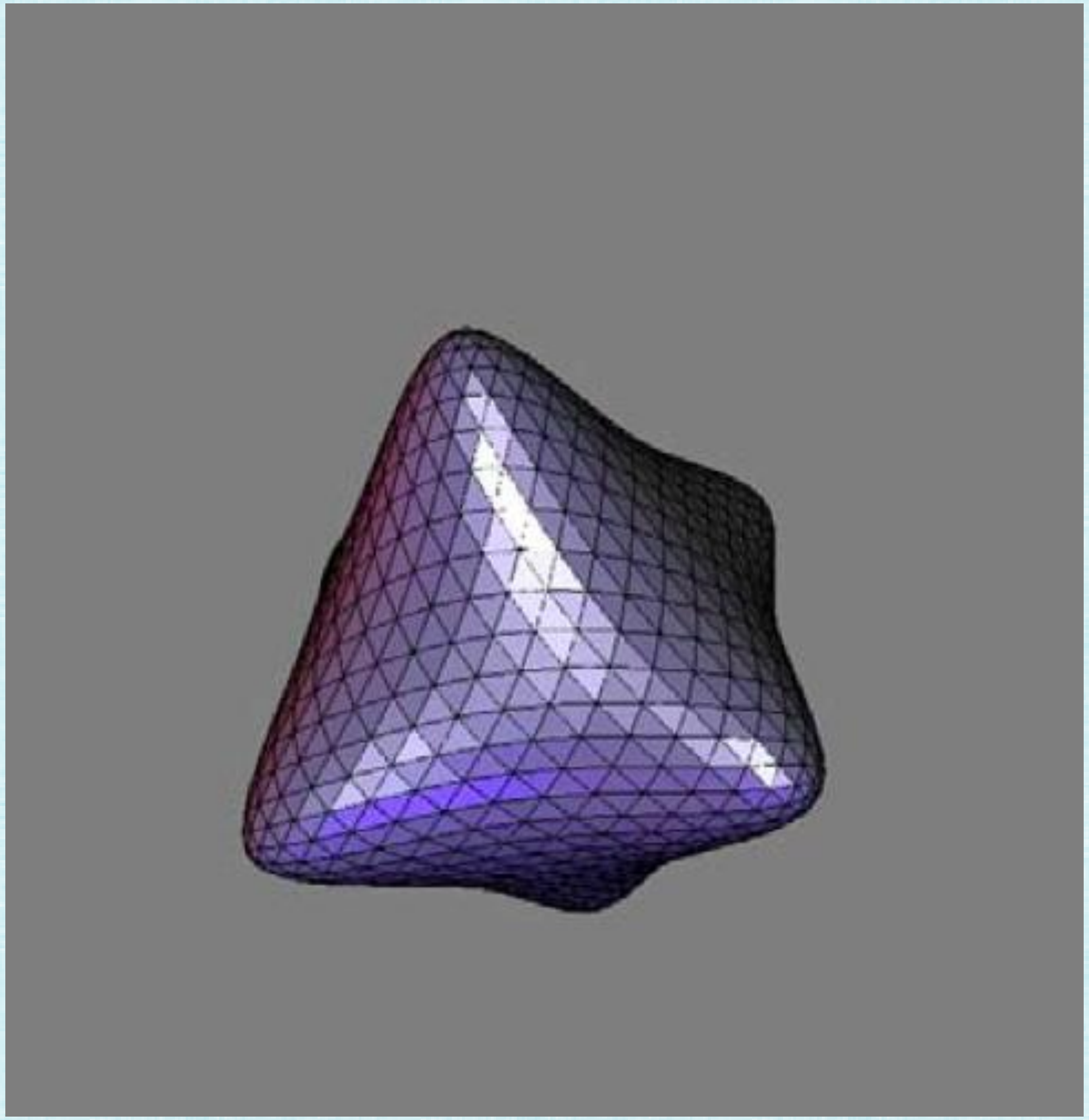
Subdivided Surface



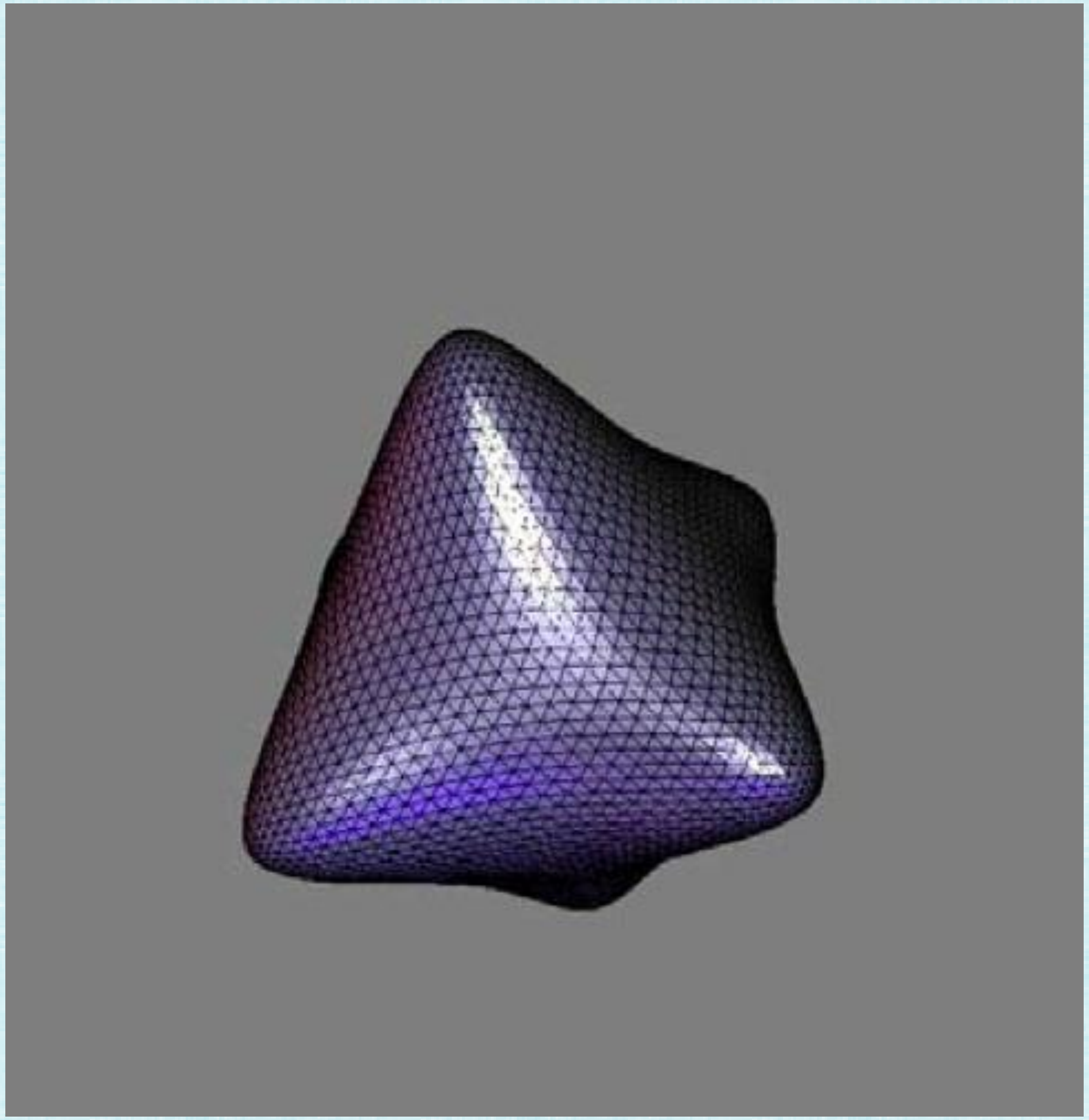
Subdivide Again



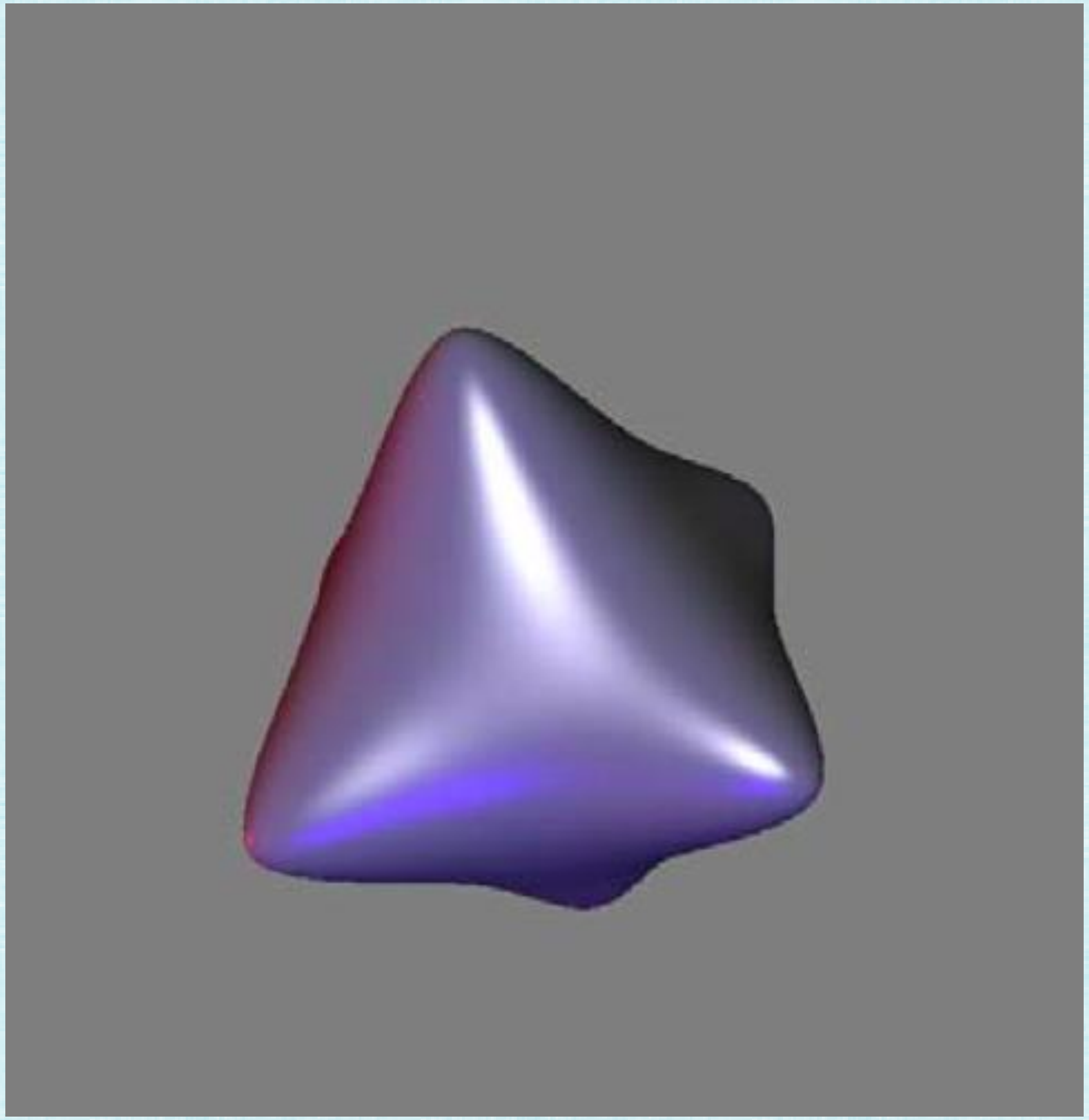
And Again



And Again

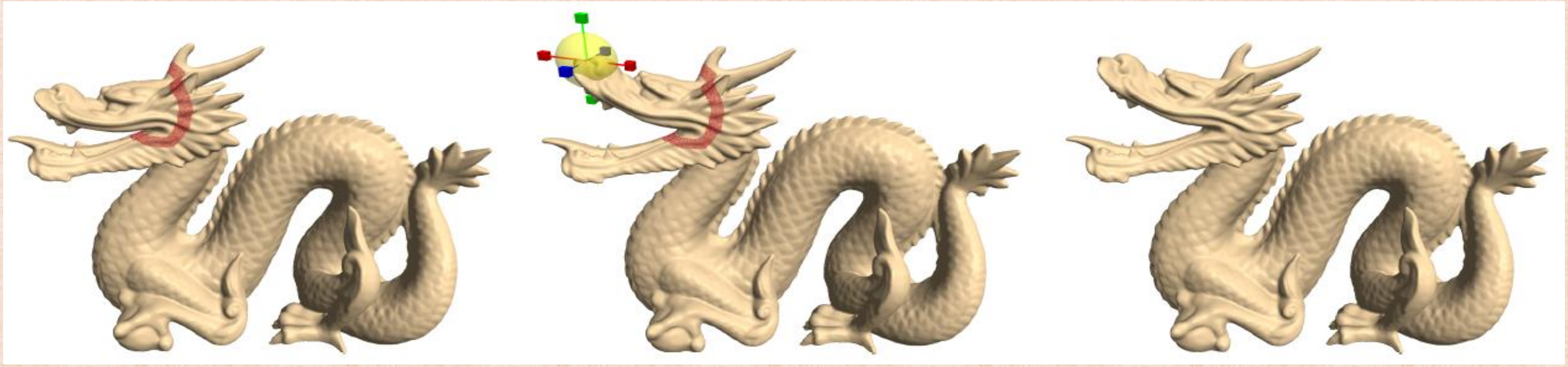


Final (Smooth) Limit Surface



Directly Editing a Triangulated Surface

- Converting NURBS to quads and subsequently performing triangle subdivision disconnects the editable NURBS representation from the triangulated surface
- In order to make further surface edits, need to edit triangle vertices directly:
  - Move a subset of the mesh (e.g. inside the the yellow sphere)
  - To keep the surface smooth, deform a larger region that contains the subset (e.g. everything in the top jaw to the left of the red boundary)



Describing Geometry via Differential Coordinates

- Each <u>differential coordinate</u> is computed as the difference between vertex position x_i and the average position of its neighbors:

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

- Differential coordinates approximate the local shape:
  - The direction of d_i crudely approximates the direction of the normal vector
  - The magnitude of d_i crudely approximates the local curvature

- Let X be the vector of all mesh vertices (one R^3 entry for each vertex)
- Let D be the vector of differential coordinates (one R^3 entry for each vertex)
- Then, can write $D = MX$ where M is a constant coefficient sparse matrix

Laplacian Mesh Editing

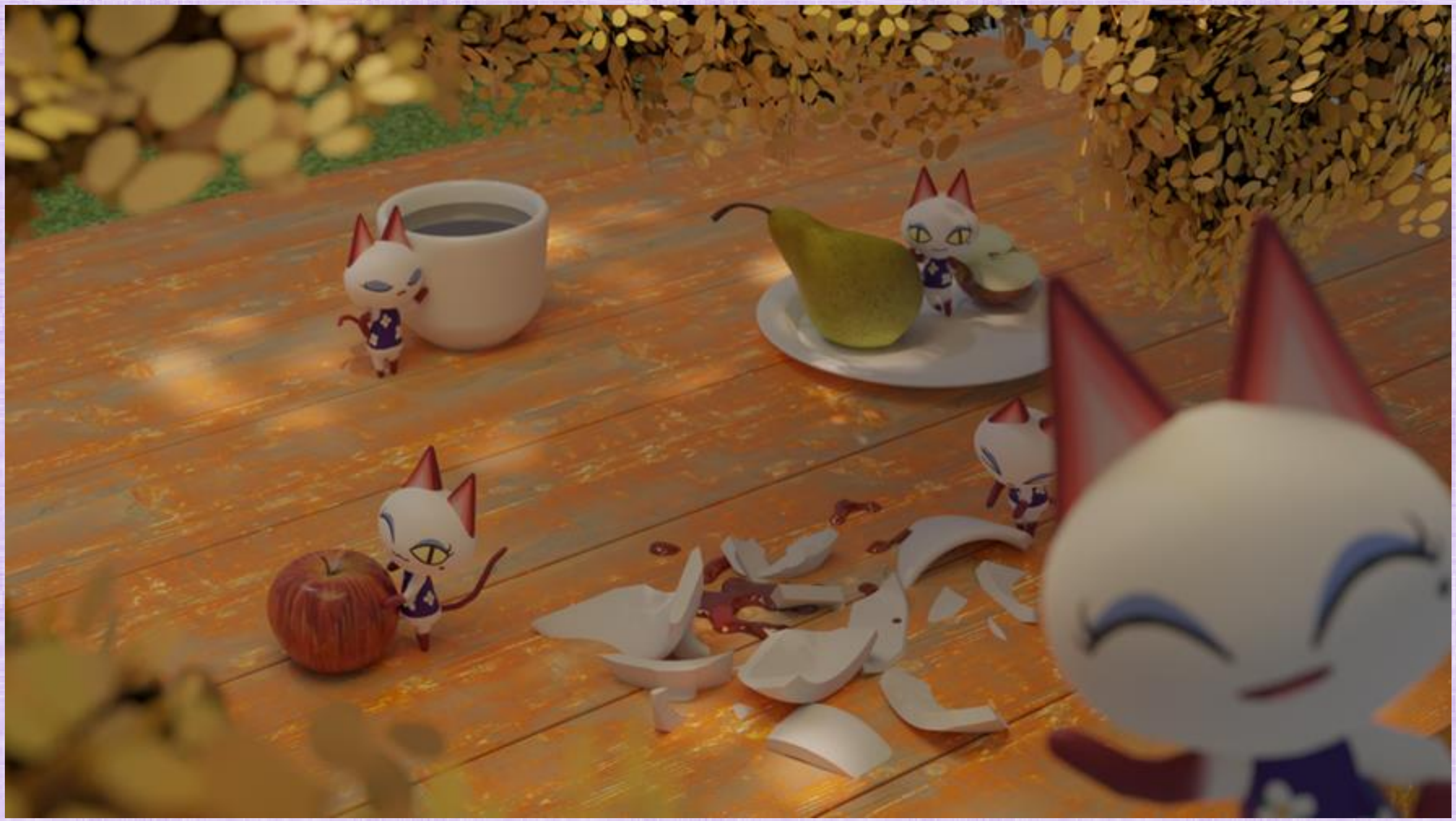
one of the ways to smoothly edit triangle mesh vertices directly

- Select vertices x_c and target positions x_{target}
- Select a region of influence Ω containing (at least) all the x_c

- Solve the sparse linear system $MX = D$ for all $x_i \in \Omega$, with soft constraints $x_c = x_{target}$
    - $MX = D$ aims to preserve the shape, while $x_c = x_{target}$ targets the deformation
    - Minimize $E(X) = \sum_{i \in \Omega} \|L(x_i) - d_i\|_2^2 + \sum_c \|x_c - x_{target}\|_2^2$

- As is typical in optimization, it can be useful to include regularization terms in addition to the data term
    - Here, regularization can be formulated by constraining a vertex and its neighbors to deform together with a rigid body transform (rotation and translation, as discussed in class 3)
    - This can be added to $E(X)$ as an additional soft constraint

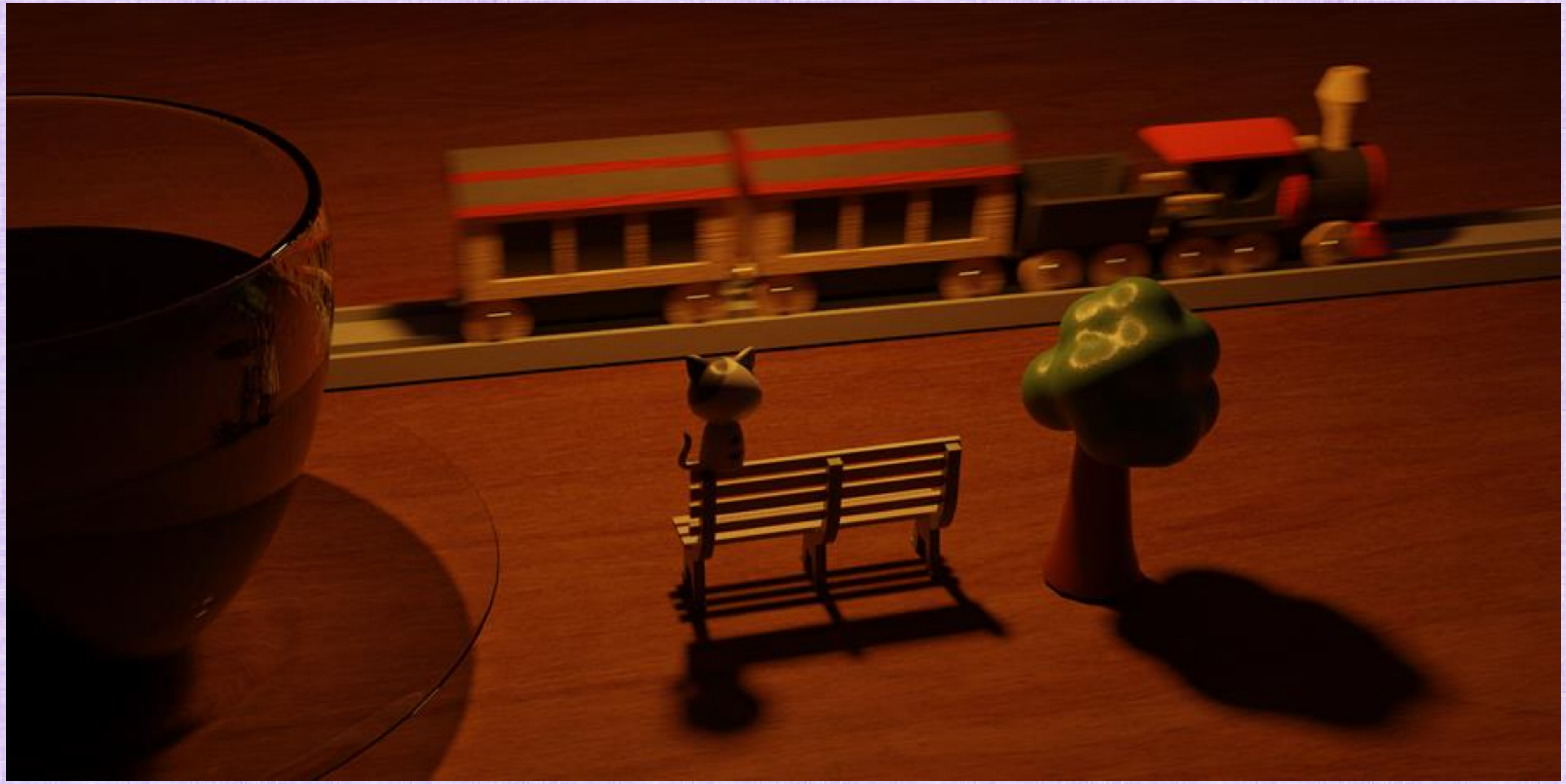
Some examples from prior students…



Blender, Clip Studio Paint, TurboSquid

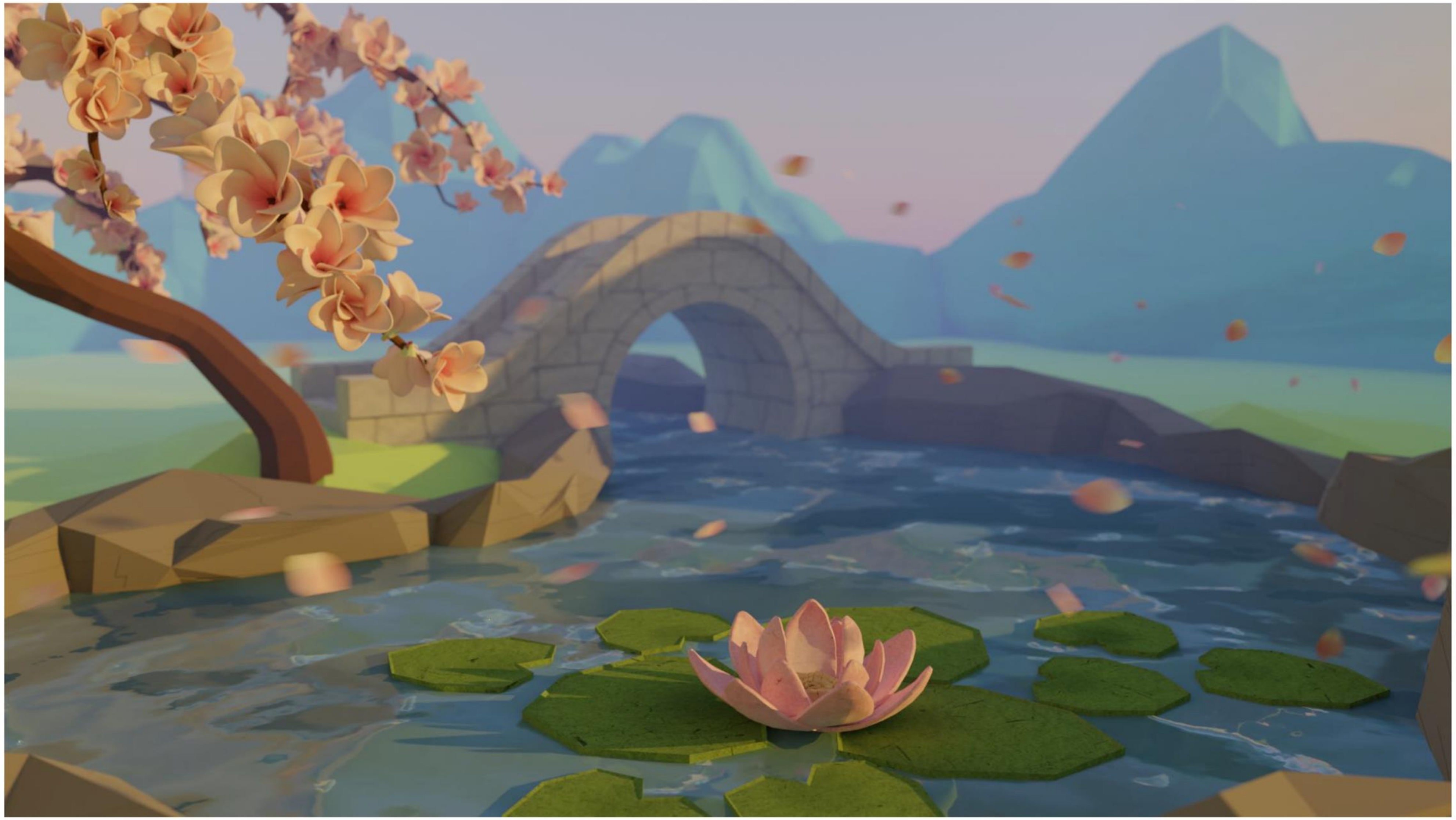
- Pear is downloaded from TurboSquid, all other geometry created from scratch
- Cat character is created through modeling from geometric primitives and sculpting
- Cup is broken through cell fracture and rigid body collisions
- Liquid simulations to model fluid inside cup
- Leaves/trees created through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

Gray Wong, CS148 2021



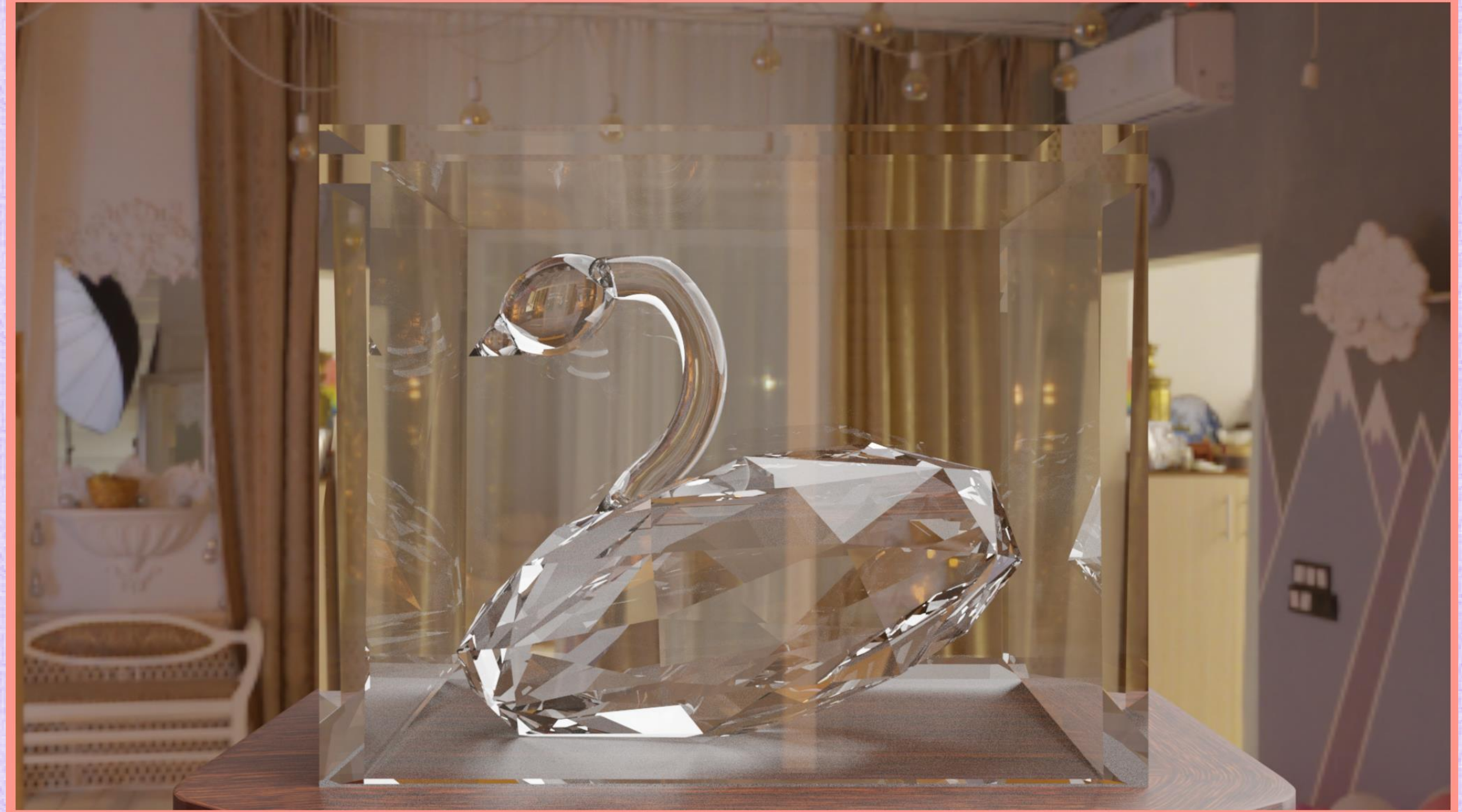
- Set train at different positions on separate keyframes to create motion blur.
- Used loop cuts to make fine-grained adjustments to the geometry of the mesh.
- Applied noise textures, bump map, and color ramp to create plastic material.
- Marked edges as sharp and used Auto Smooth to ensure that intentional 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



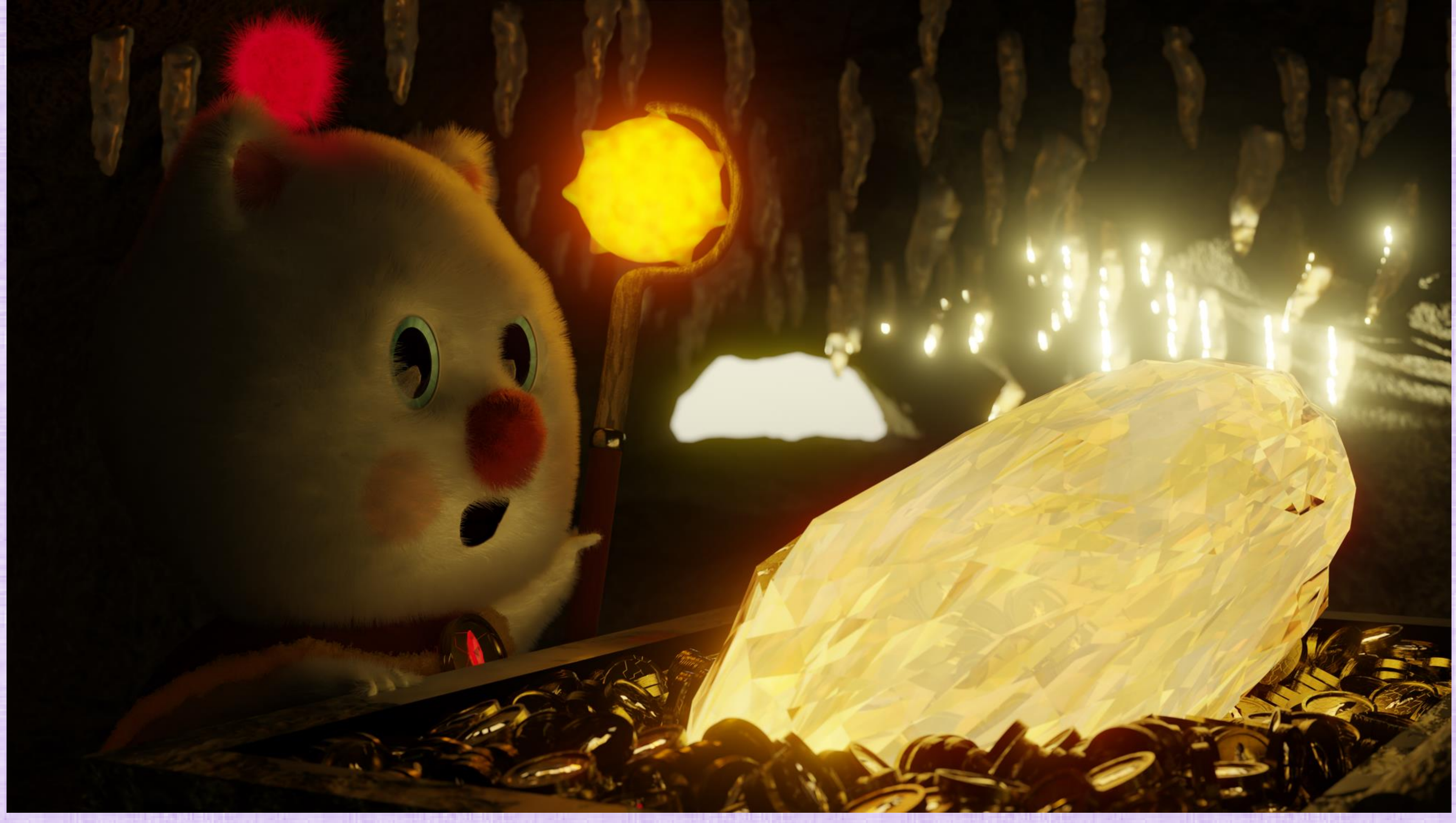
- Tree created from scratch using a skin modifier on a skeleton of vertices and edges forming the shape of a tree, and subdivided later to apply weight painting for the particle system.
- Bridge, stone, ground, and mountains created from scratch using extrusion and various low-poly modeling techniques.
- 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)

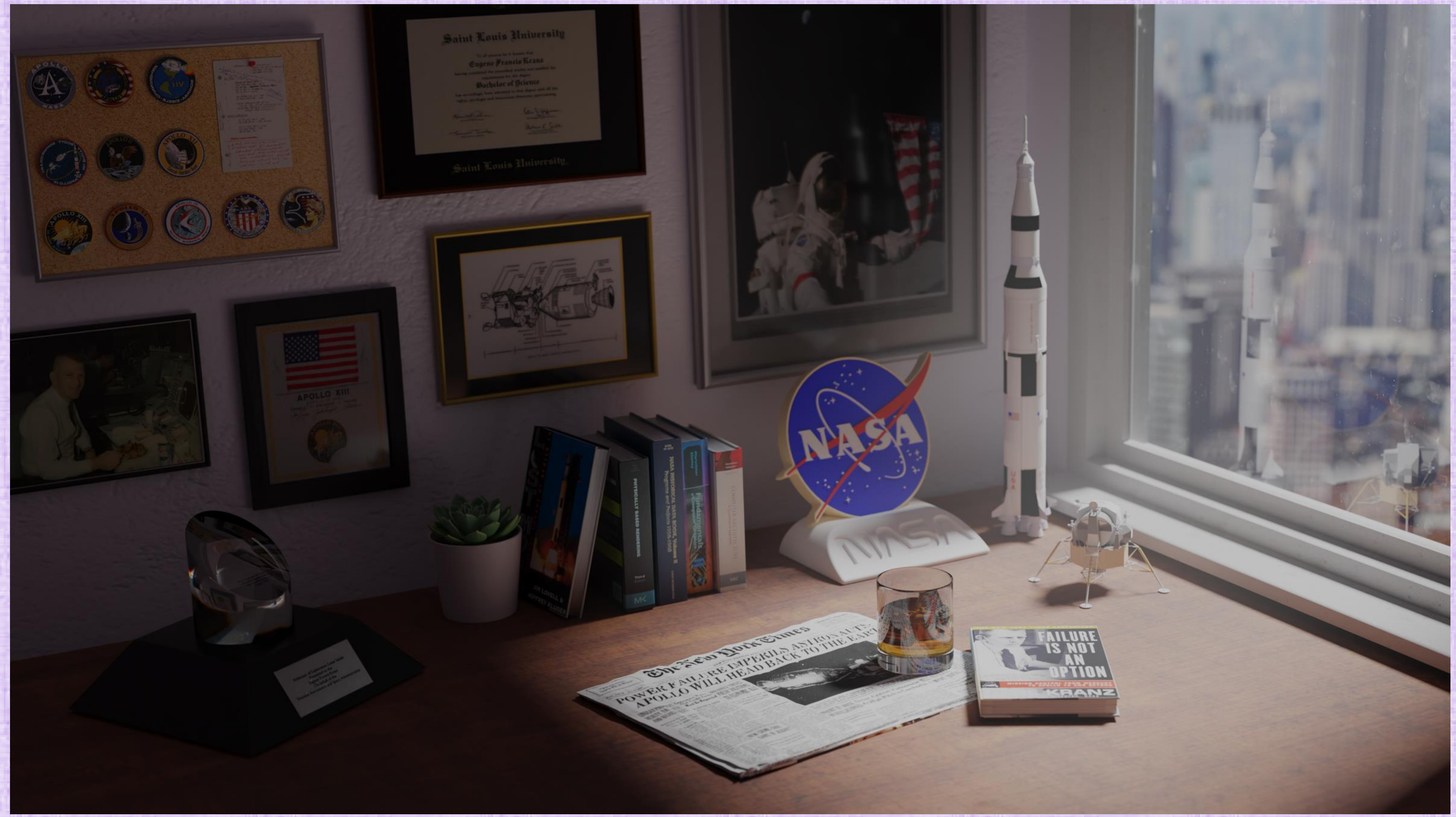
- Scaled a sphere to model the basic body shape, then applied decimate to get the "crystal" look
- Applied spin to a cylinder to model the rounded swan neck
- Used the bevel function to round the wood surface edges
- Used the "inset faces" method on the face of a cube and moved the face back to create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



Crystal Explorer by Kent Vainio (2020)

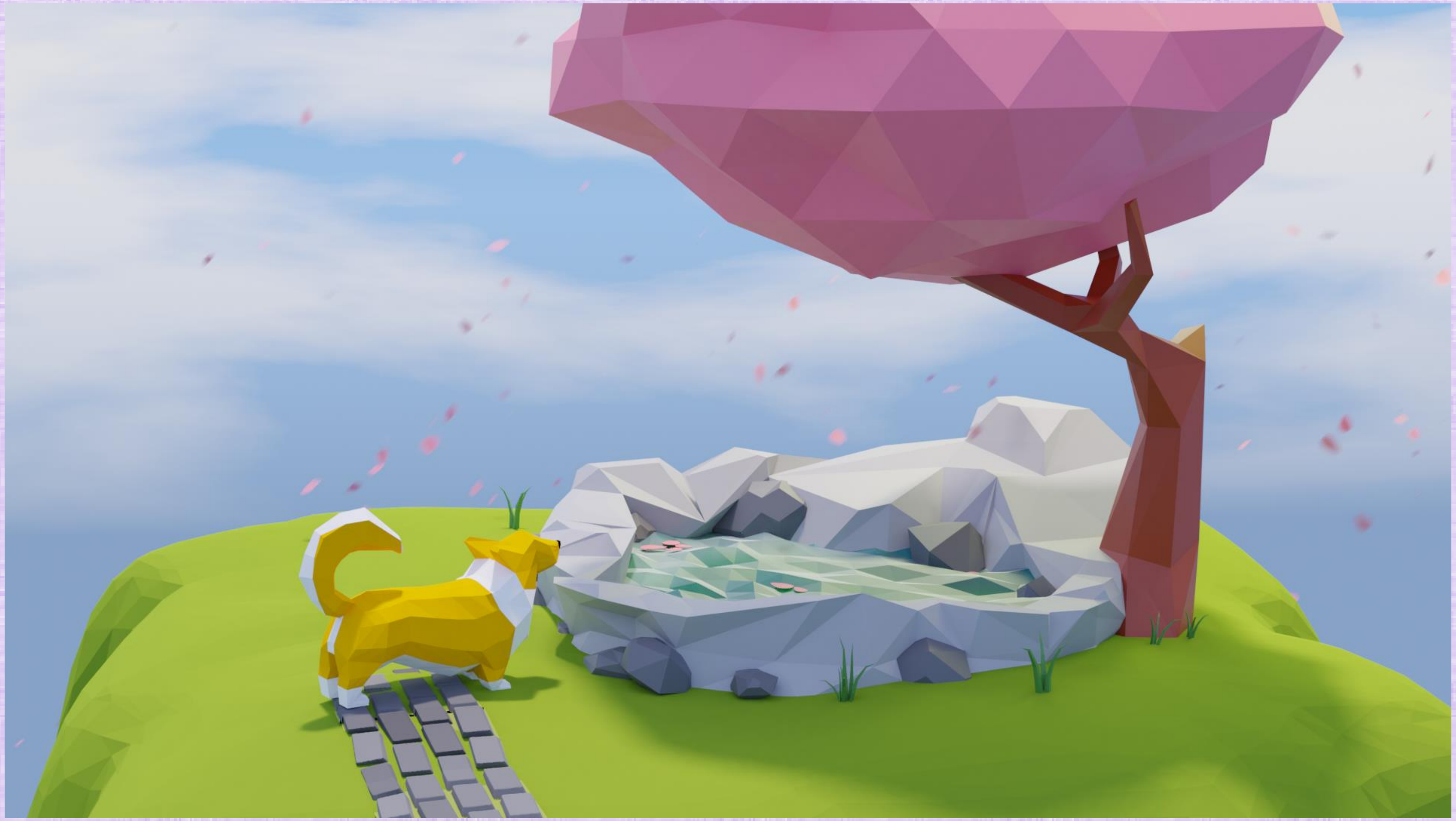
METHODS:

- 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

Yifan Wang, CS148 2020