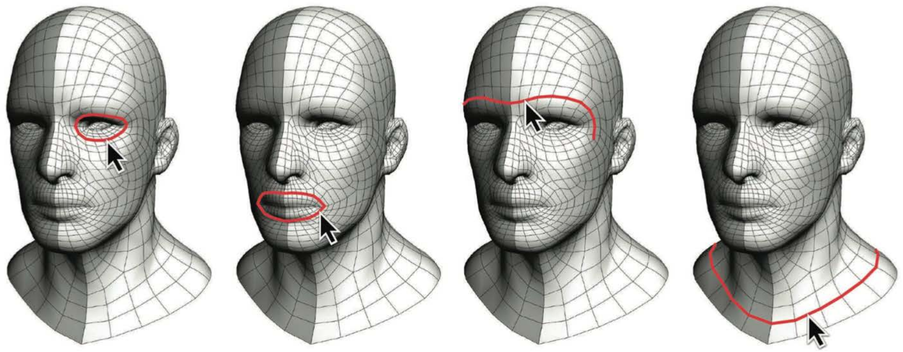
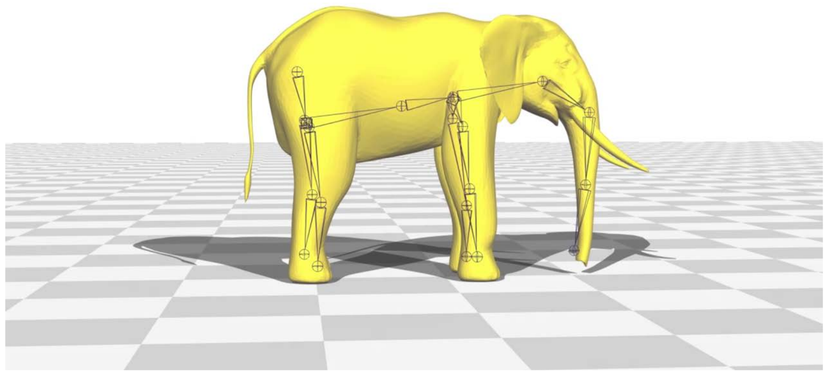
Summary

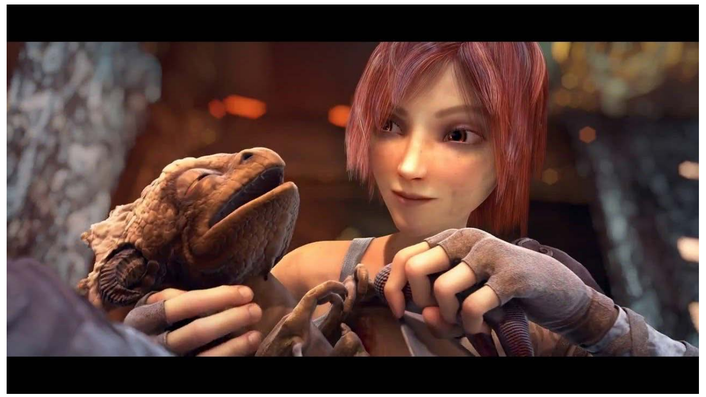
What is Computer Graphics?

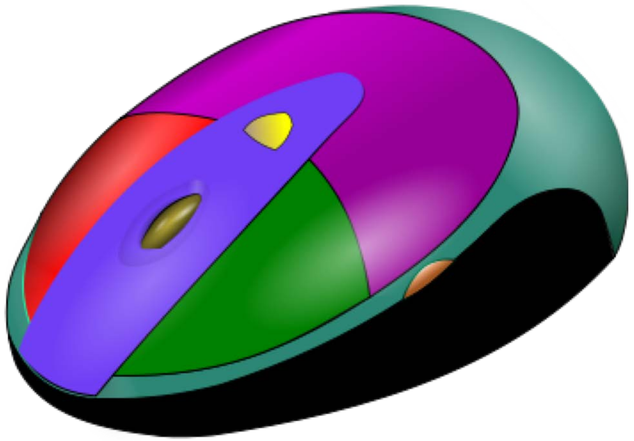


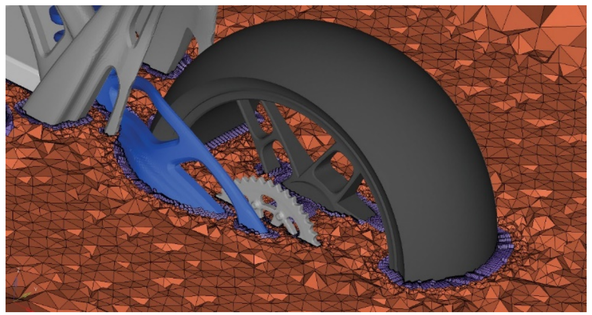


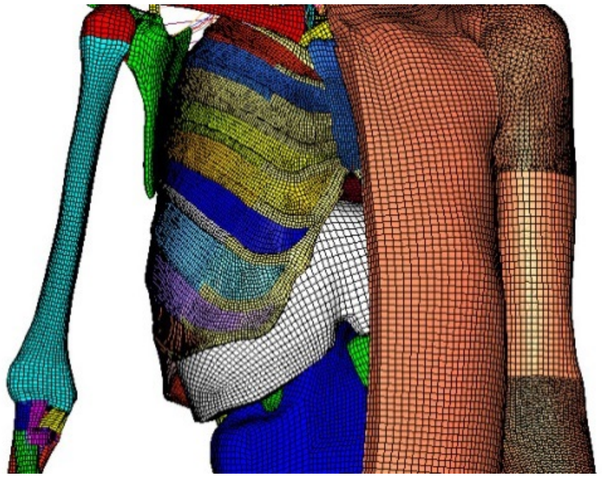
Involves the creation and manipulation of 2D and 3D data

Why is Computer Graphics?



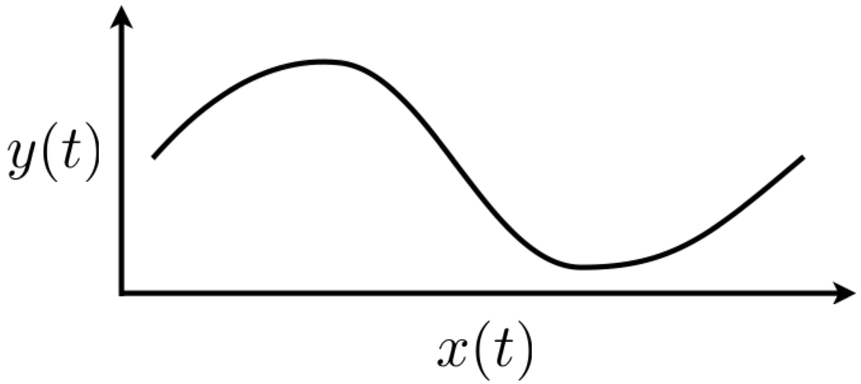


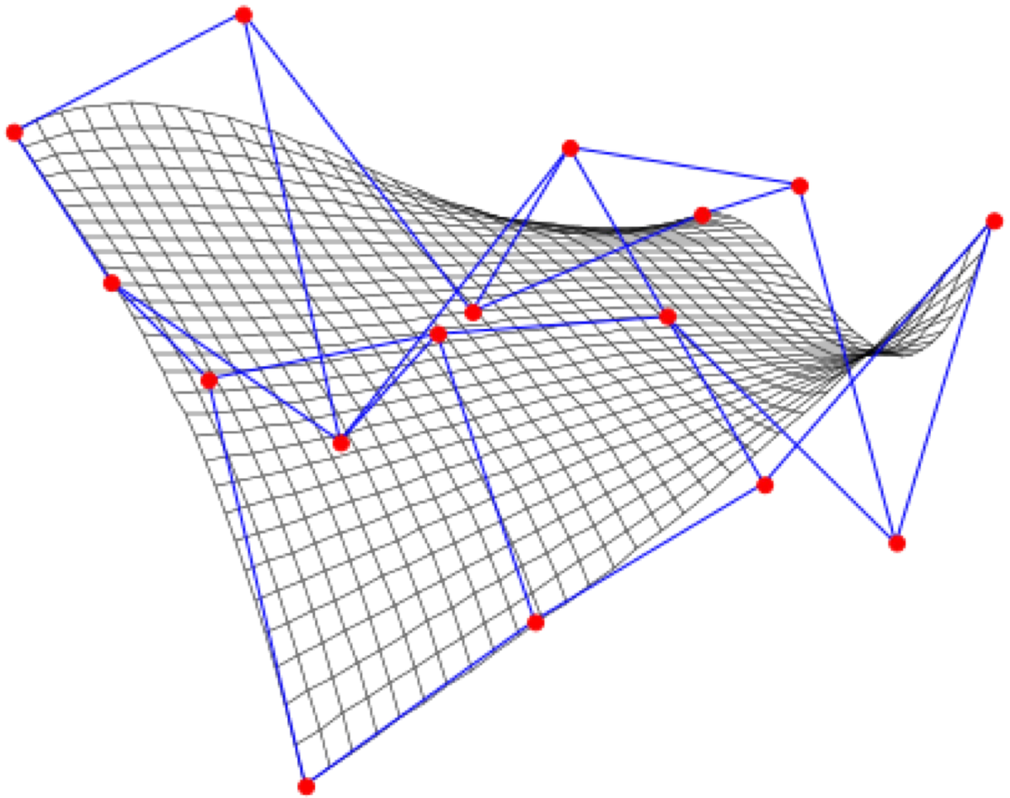




Modeling

Building blocks: curves and surfaces

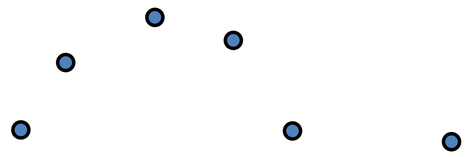




By Wojciech mula at Polish Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=9853555

Polynomial curves

- Polynomials



$$\mathbf{p}(t) = \begin{pmatrix} x_0 + x_1 t + x_2 t^2 + \cdots \\ y_0 + y_1 t + y_2 t^2 + \cdots \end{pmatrix} = \\ = \mathbf{p}_0 + \mathbf{p}_1 t + \mathbf{p}_2 t^2 + \cdots$$

- For degree d we need $d + 1$ points (“coefficients”)

Interpolation (1D)

- Polynomial fitting can be done explicitly:

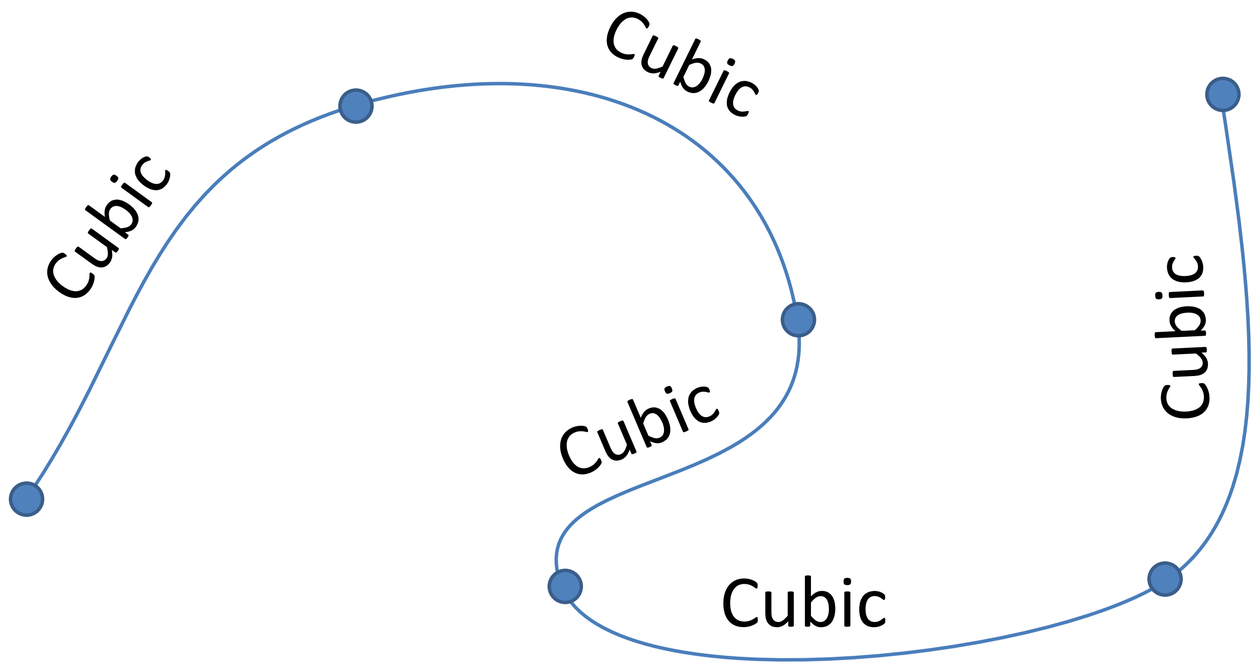
$$y(t) = \sum_i p_i \prod_{j \neq i} \frac{t - t_j}{t_i - t_j} = \sum_i p_i L_i(t)$$

- Check that $y(t_i) = p_i$

- Products $L_i(t)$ are called Lagrange polynomials

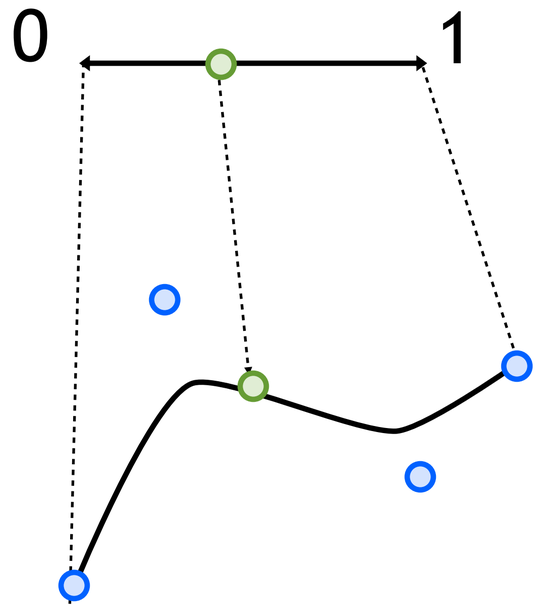
Splines

- Paste together low-degree polynomials

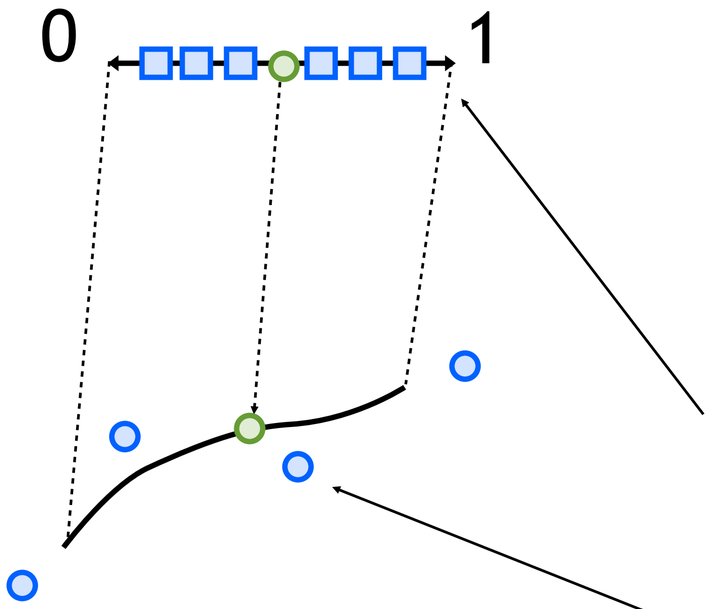


Ingredients

Bezier



B-spline



$$m = n + p + 1$$

Basis degree p

Knots Points $m + 1$

Control Points $n + 1$

Bézier Patches

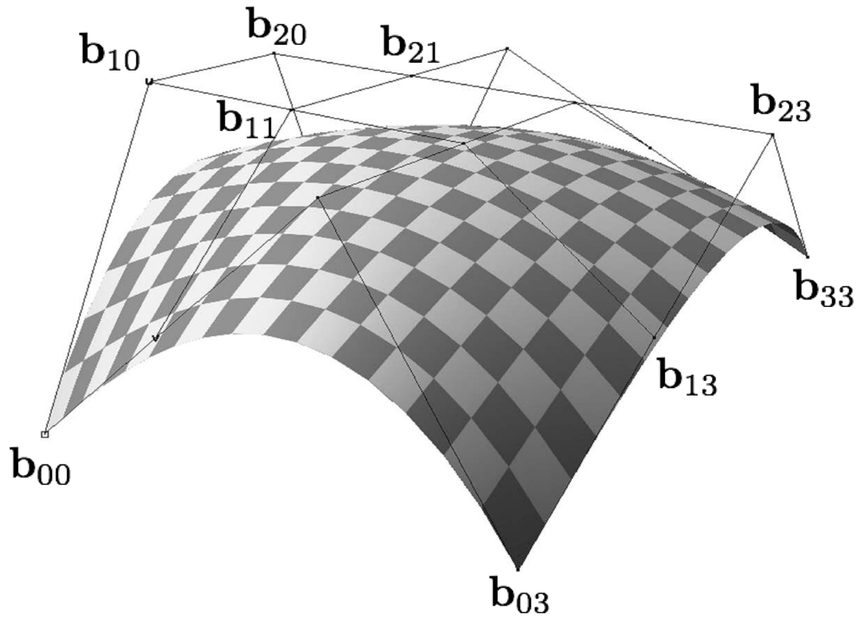
- Build on Bézier curves

$$b^m(u) = \sum_{i=0}^{m} b_i B_i^m(u)$$

- Control points as curves

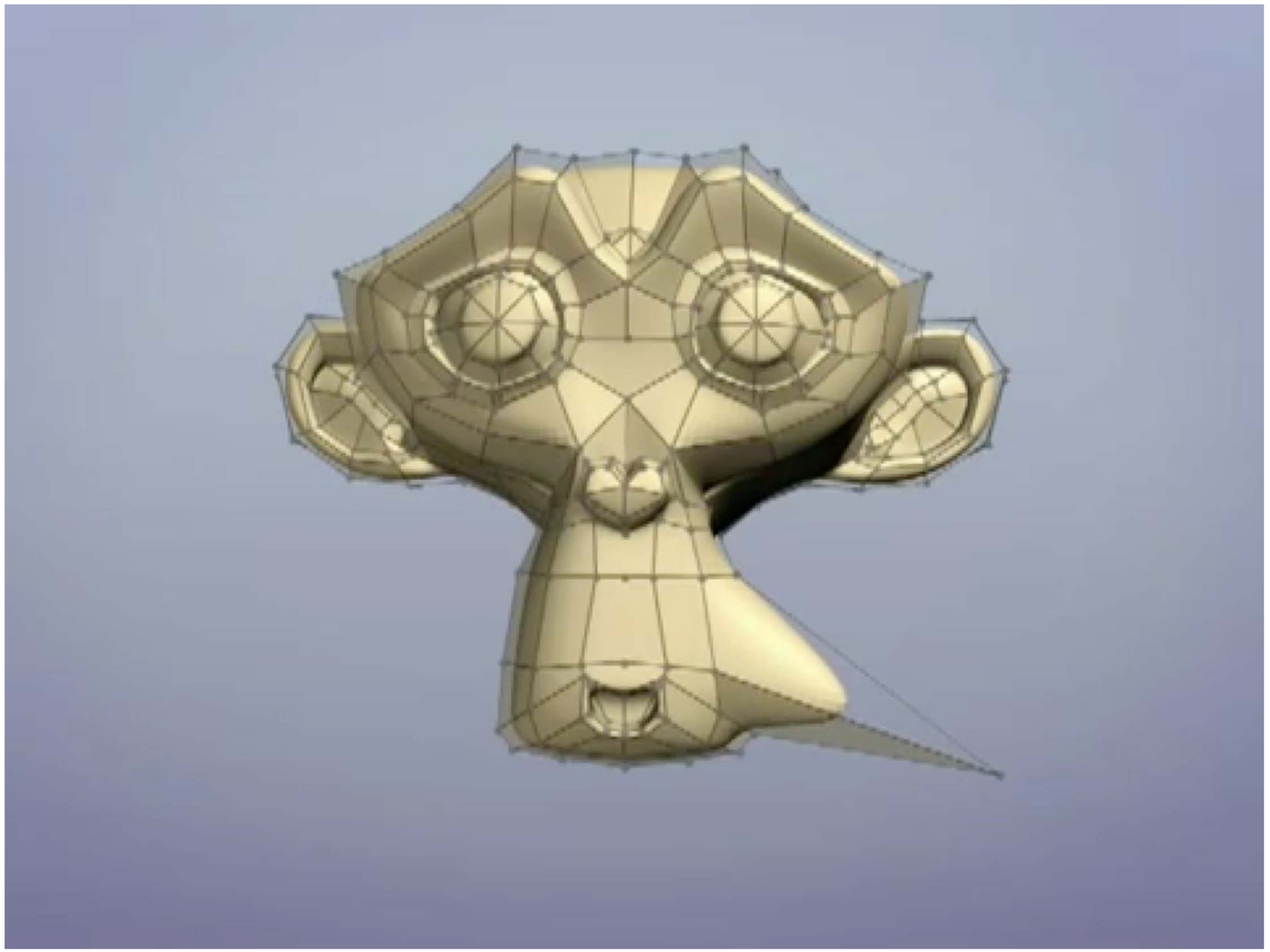
$$b_i = b_i(v) = \sum_{j=0}^{n} b_{ij} B_j^n(v)$$

$$b^{mn}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(u) B_j^n(v)$$

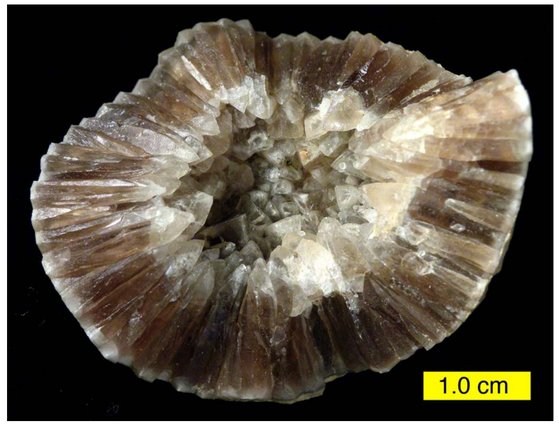


- Keep one parameter fixed: iso-parameter curves

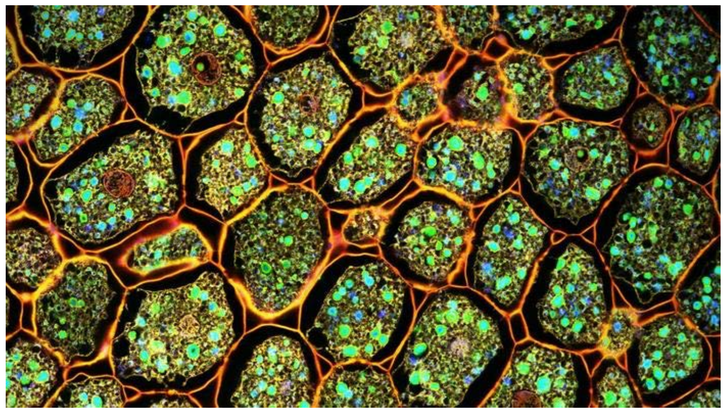
How are they used in practice?

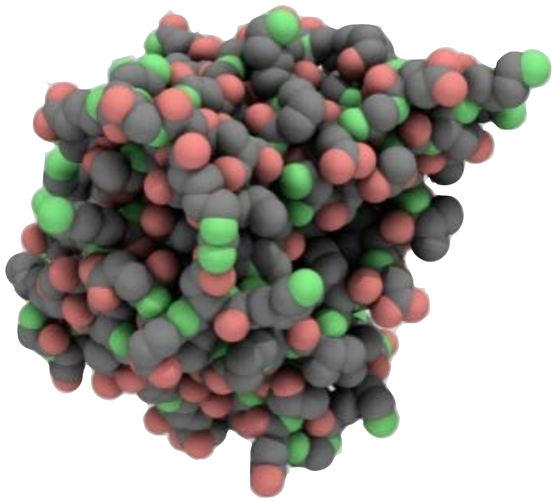


Origin of Meshes



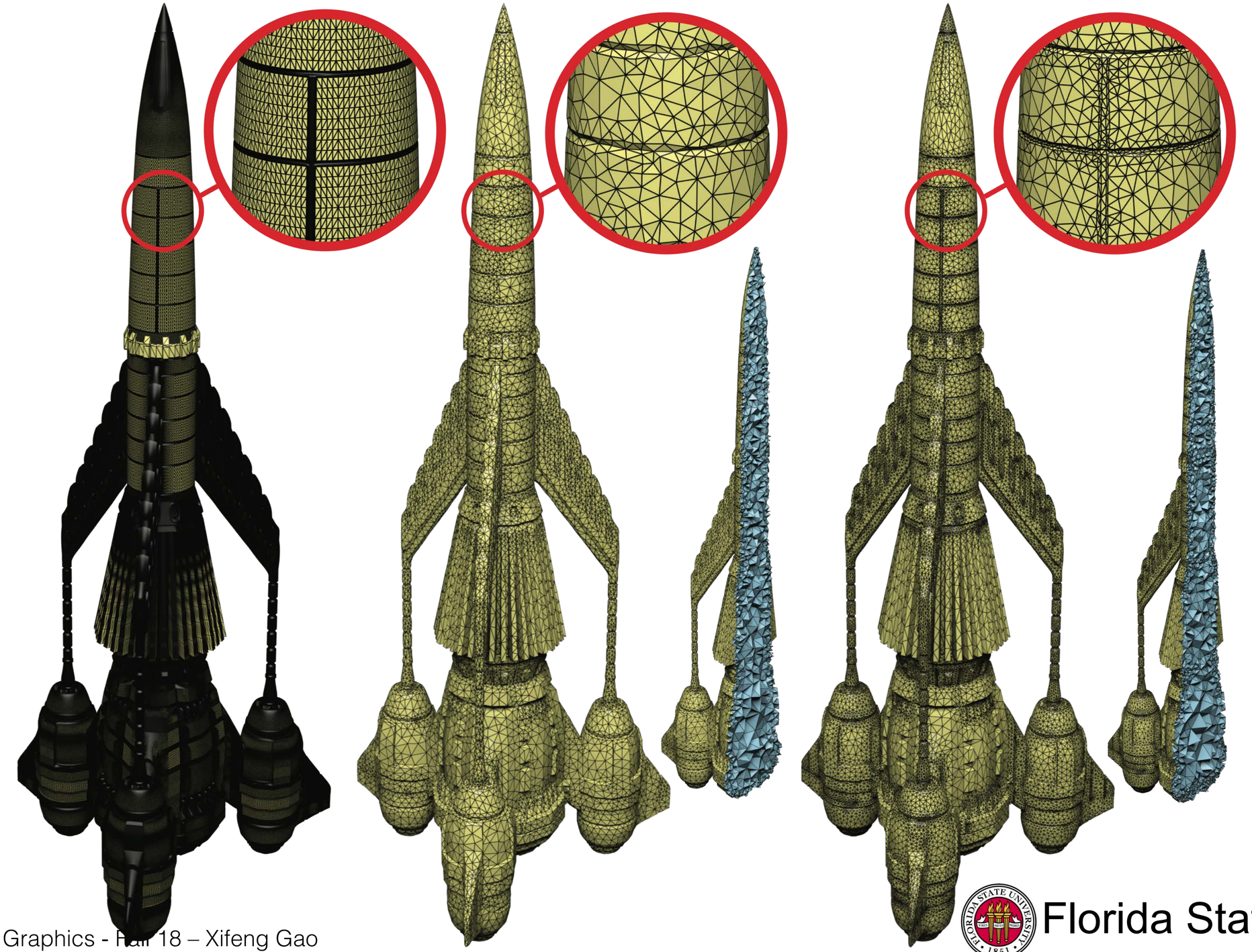
- In nature, meshes arise in a variety of contexts:
  - –Cells in organic tissues
  - –Crystals
  - –Molecules
  - –Mostly ***convex*** but ***irregular*** cells
  - –Common concept: ***complex*** shapes can be described as ***collections*** of ***simple building blocks***



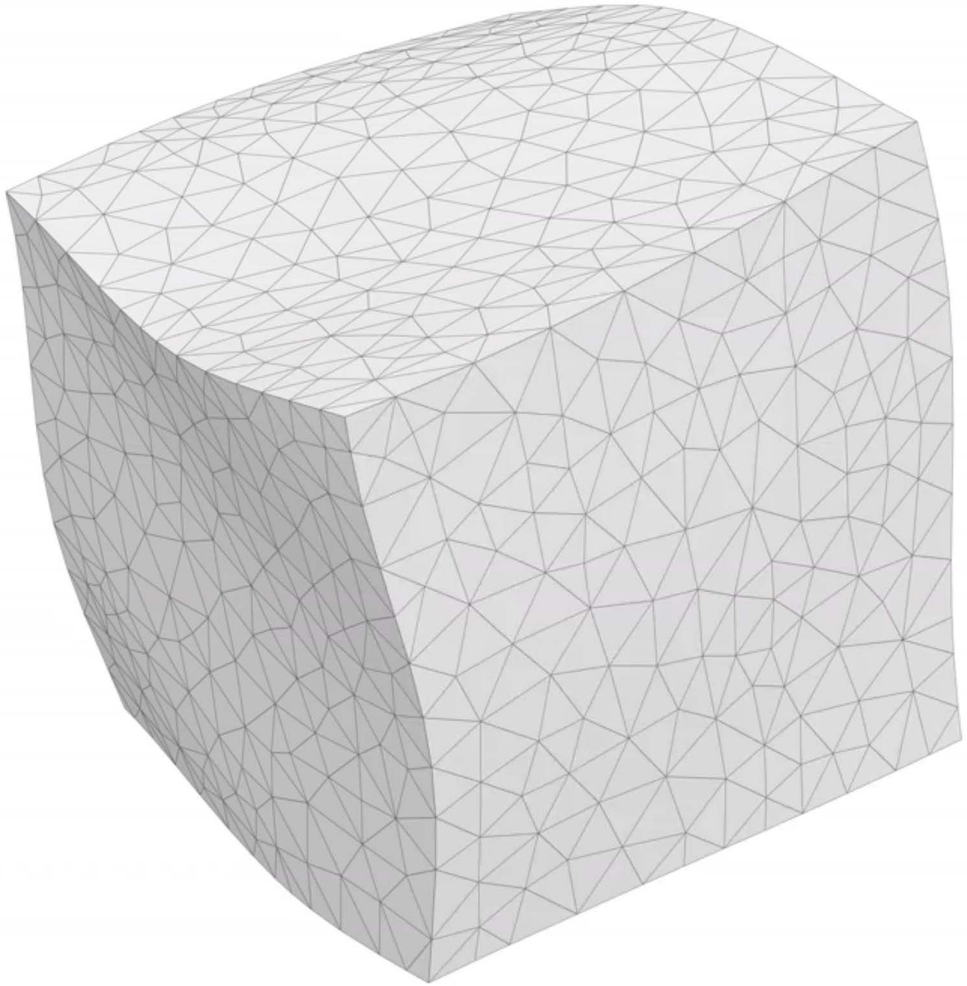


Credit: Fernan Federici Moment Getty Images

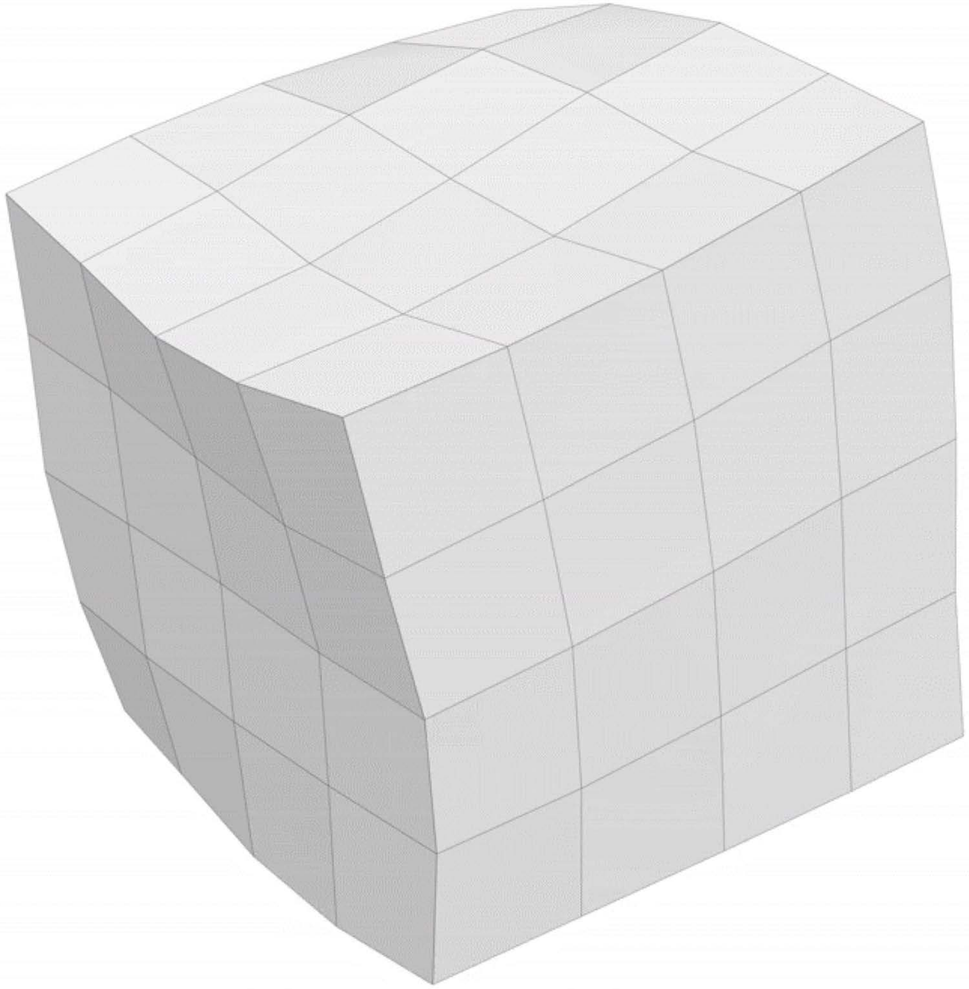
By Mark A. Wilson (Department of Geology, The College of Wooster).[1] - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=12666593



Volumetric Meshing

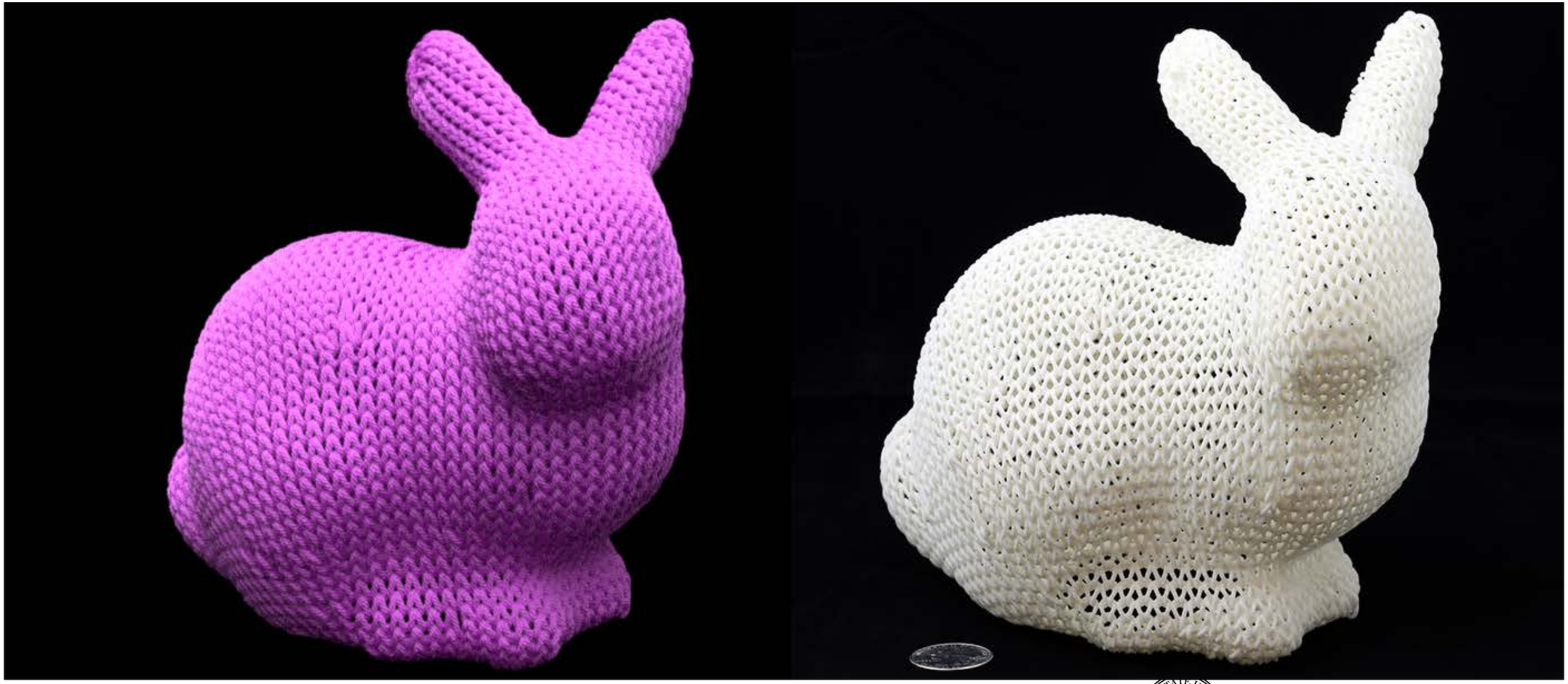


Tet-meshing

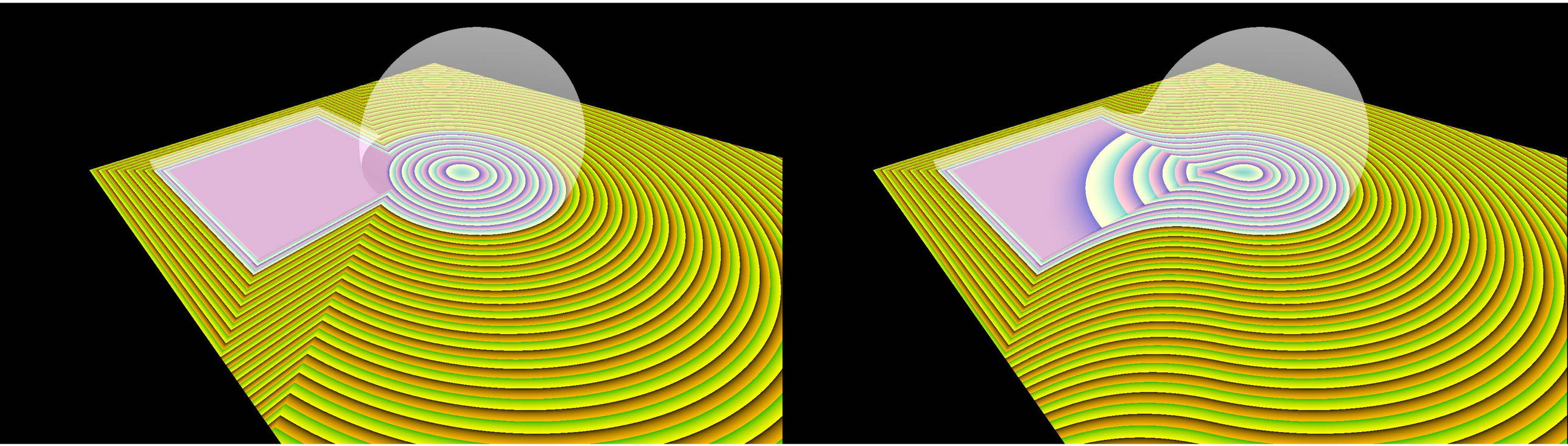


Hex-meshing

Stitch meshing

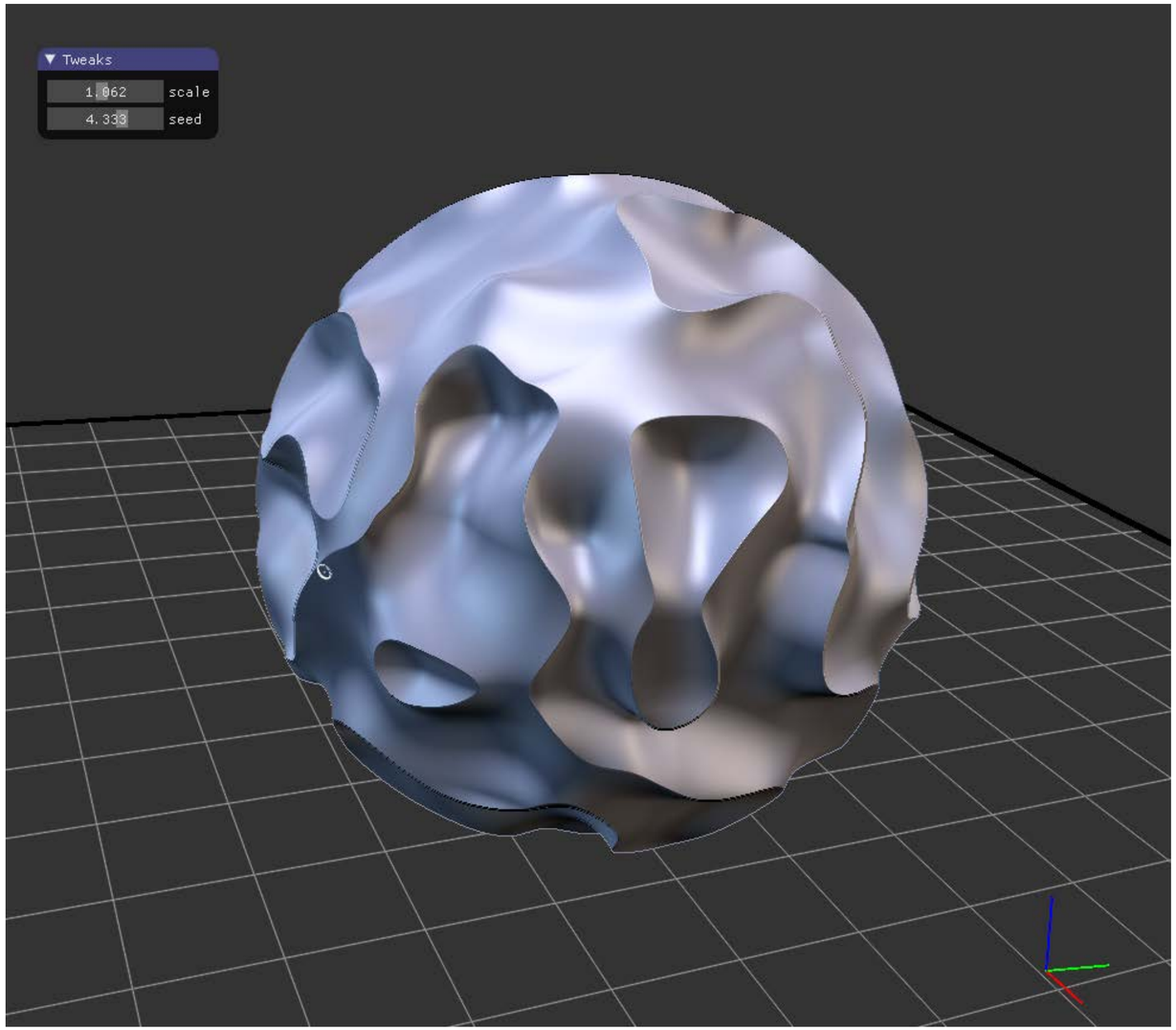


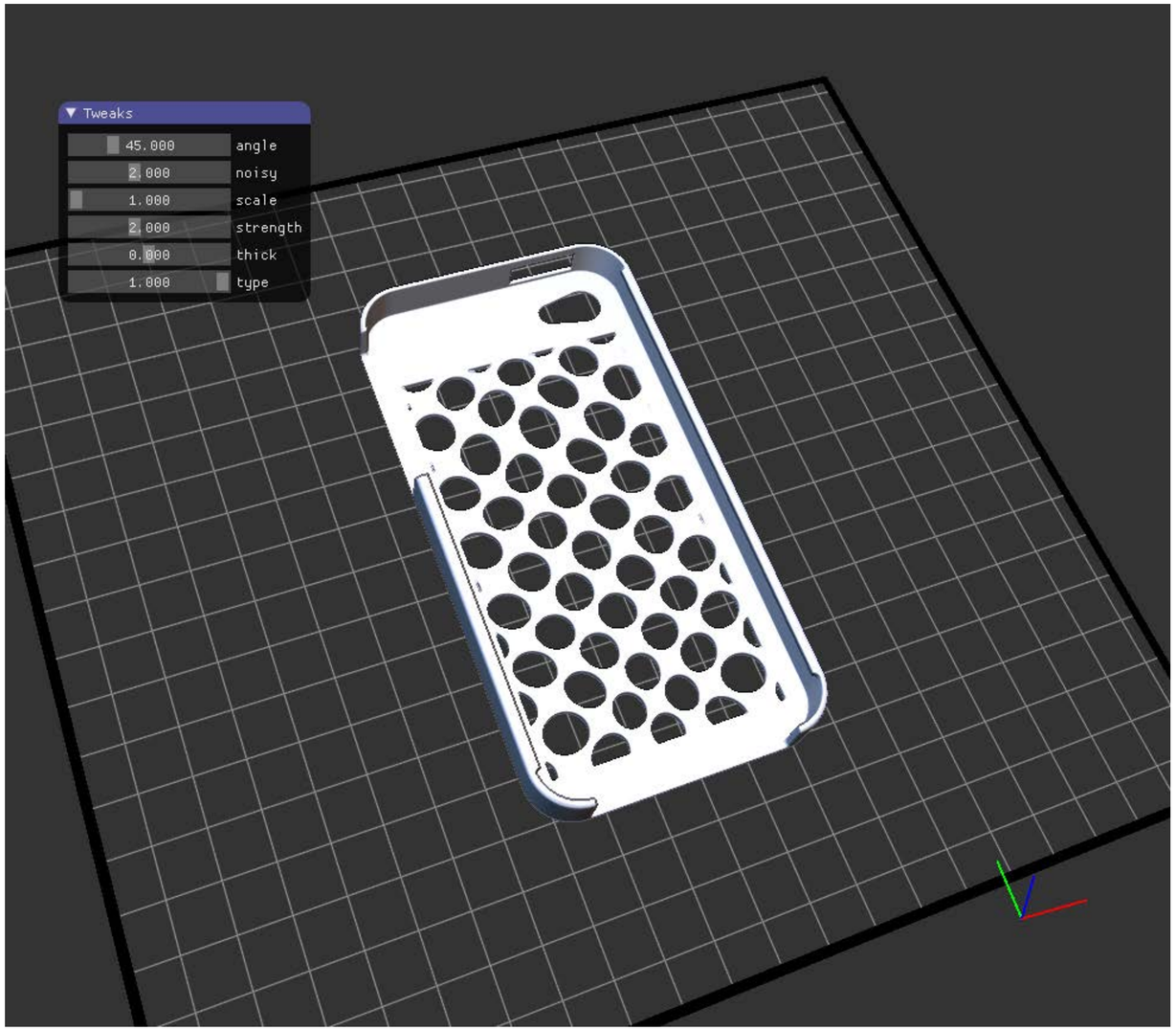
Modeling With Implicit functions



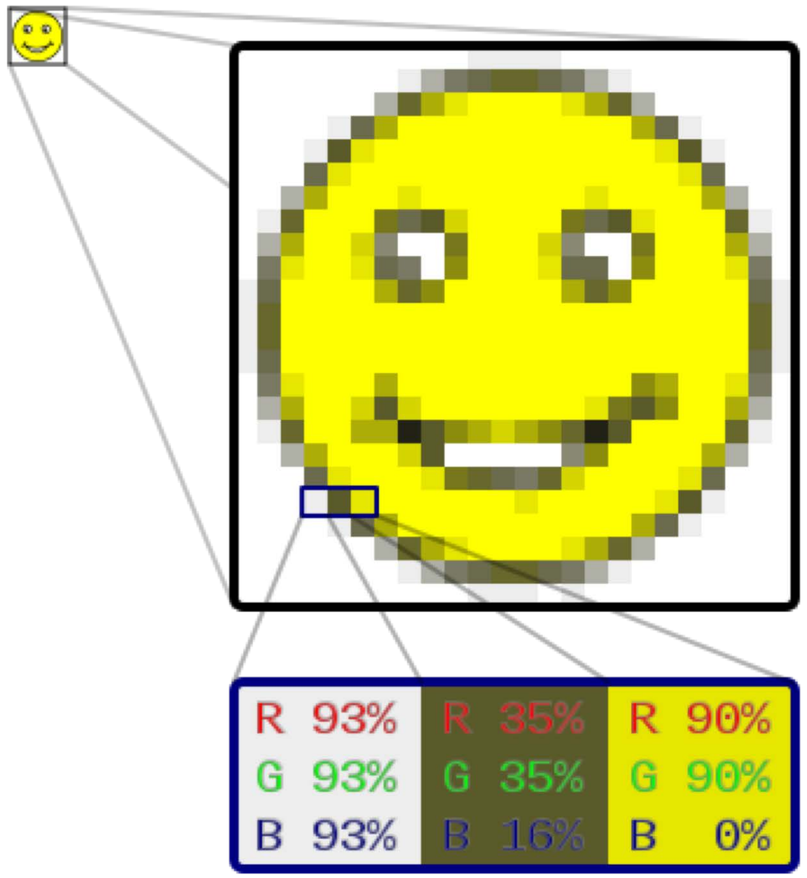
Alex Evans at SIGGRAPH 2015
https://www.mediamolecule.com/blog/article/siggraph_2015

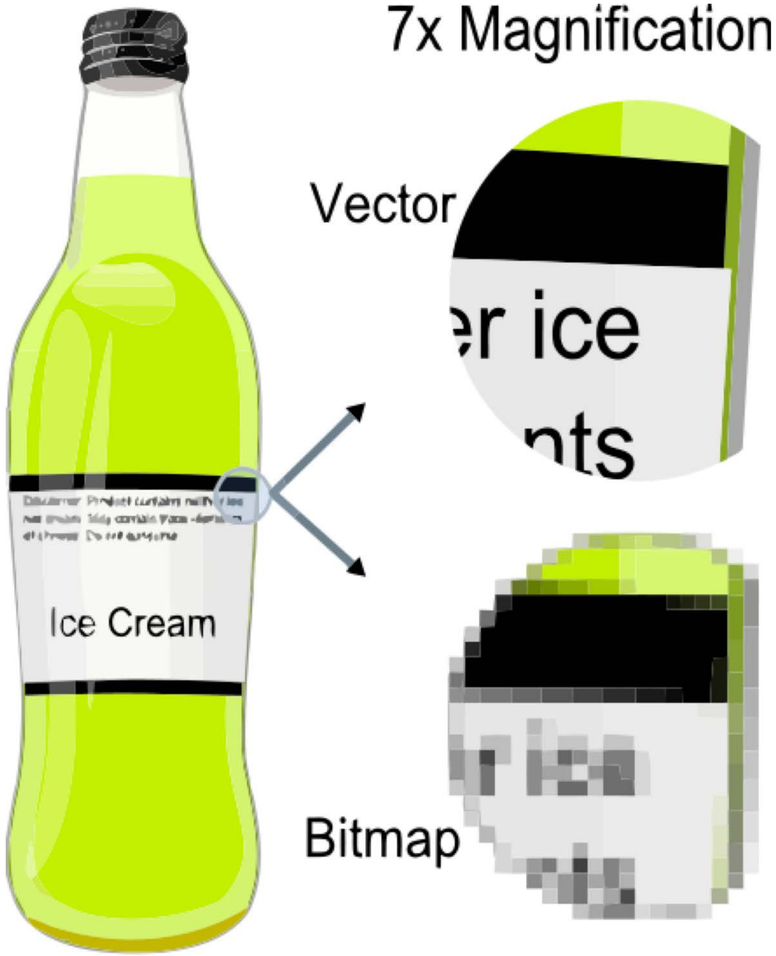
IceSL





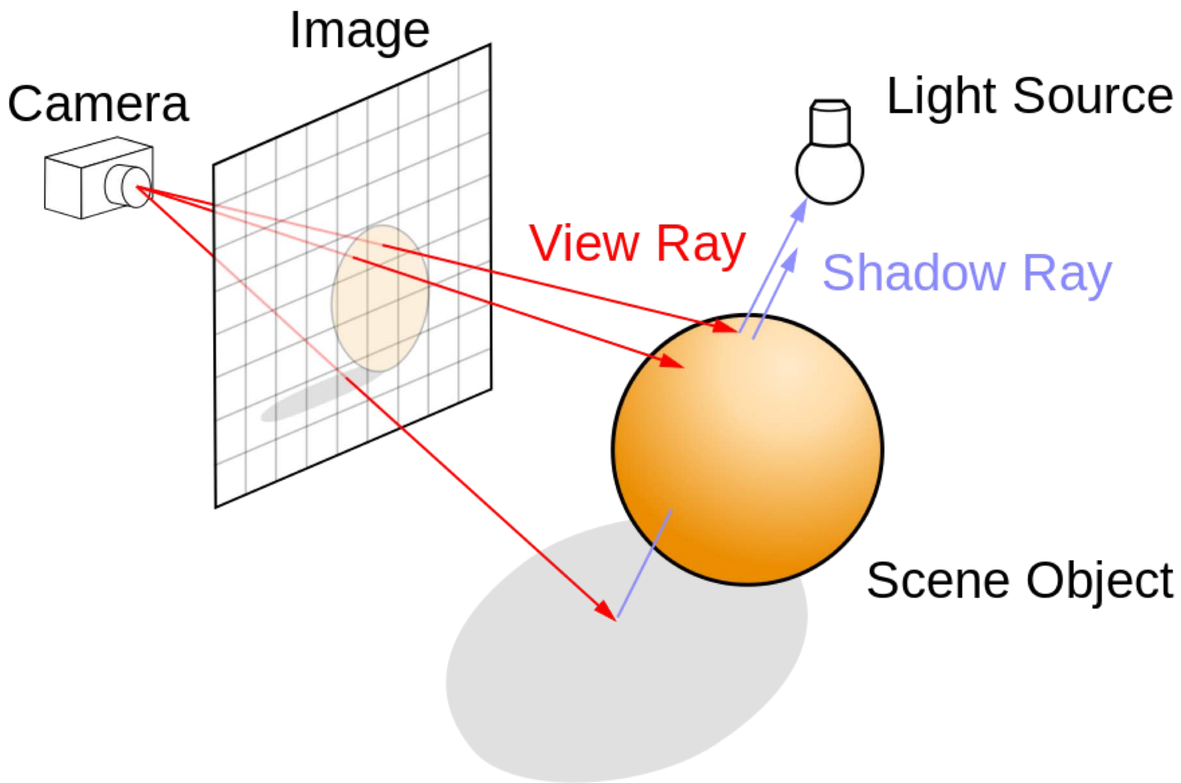
http://shapeforge.loria.fr/icesl/


R 93%
G 93%
B 93%
R 35%
G 35%
B 16%
R 90%
G 90%
B 0%


7x Magnification
Vector
Ice Cream
Bitmap

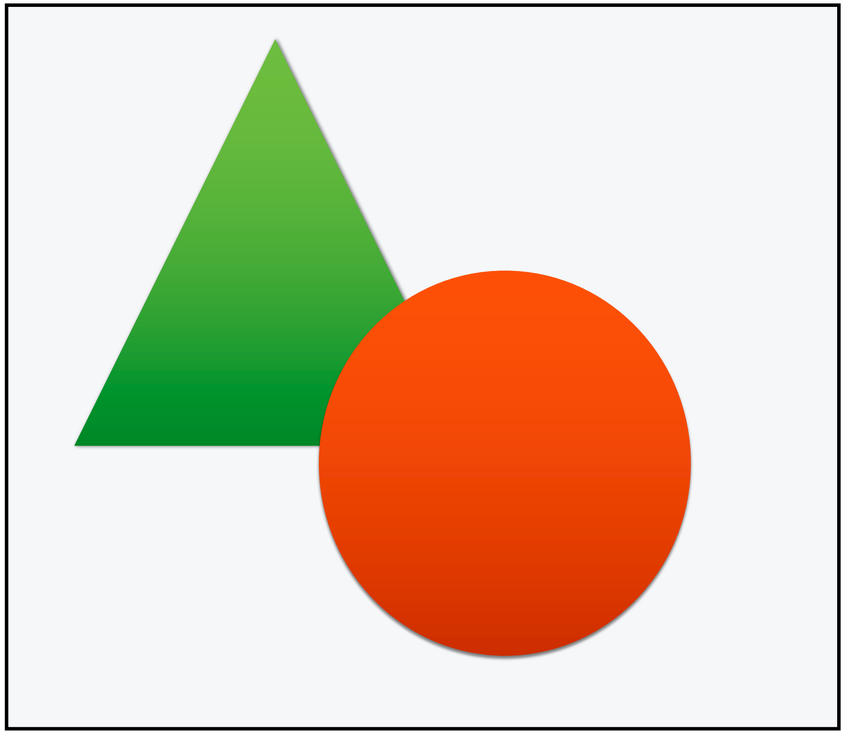
Two major approaches

Per-pixel - “Raytracing”



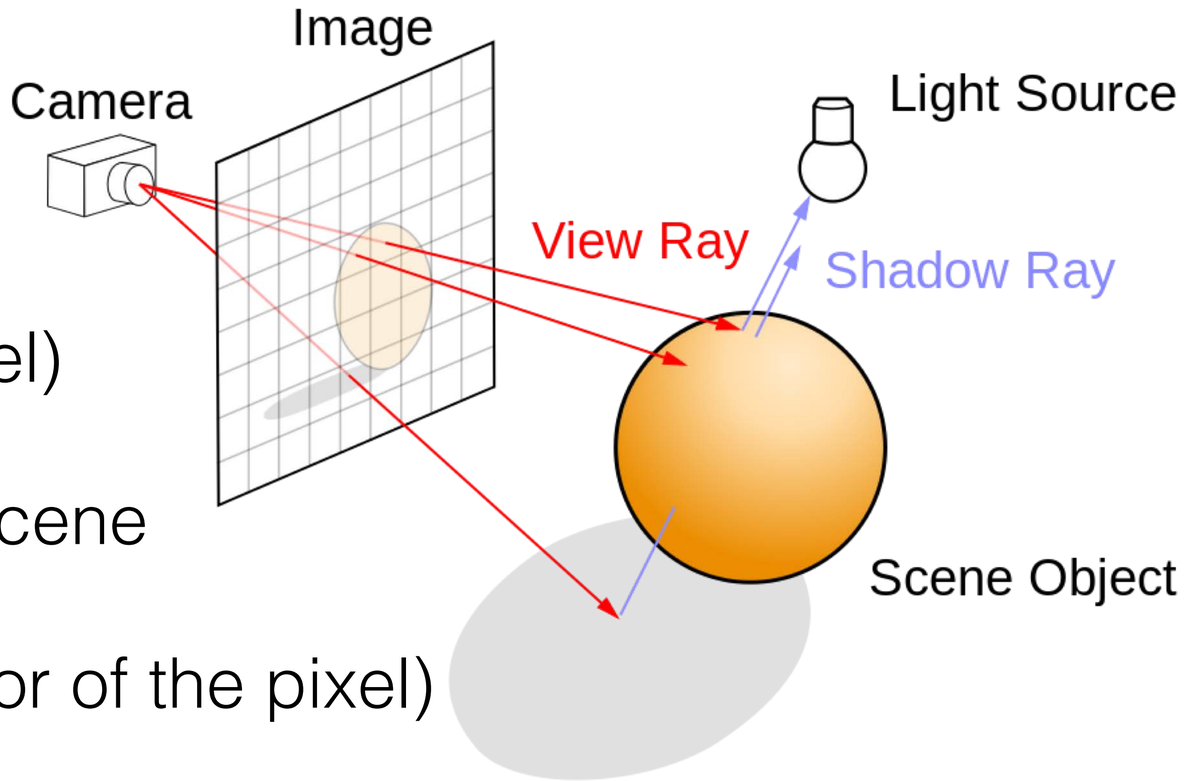
By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

Per-object - “Rasterization”



Basic Raytracing

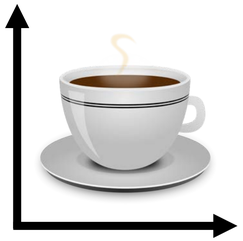
1. Generation of Rays (one per pixel)
2. Intersection with objects in the scene
3. Shading (computation of the color of the pixel)



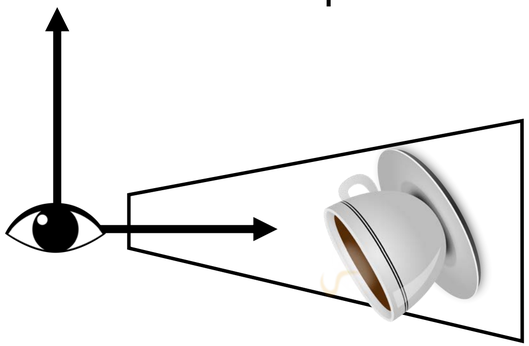
By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

Viewing Transformation

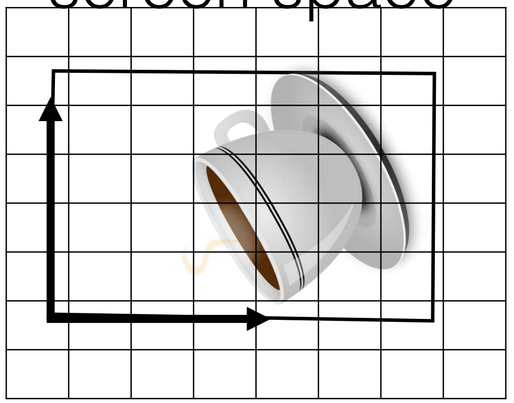
object space



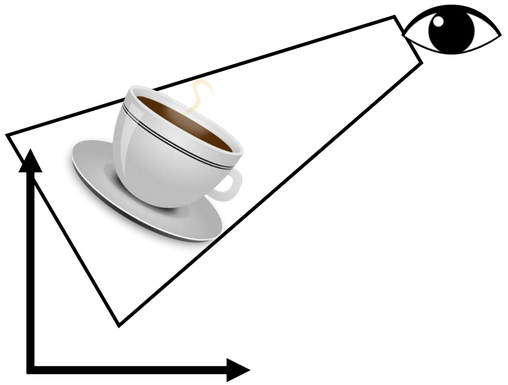
camera space



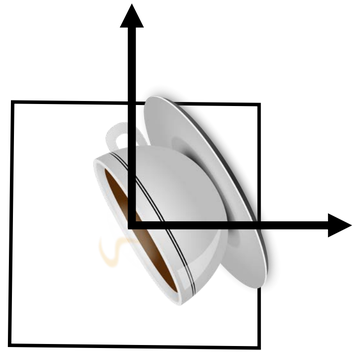
screen space



model camera projection viewport

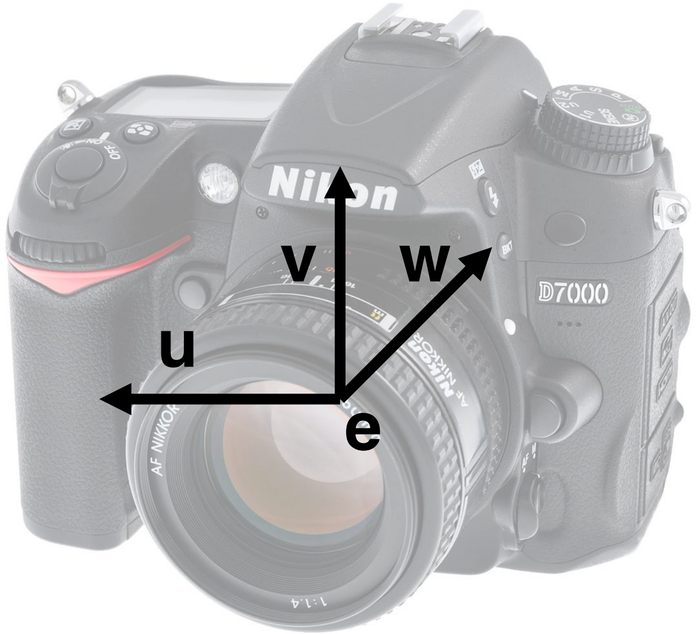


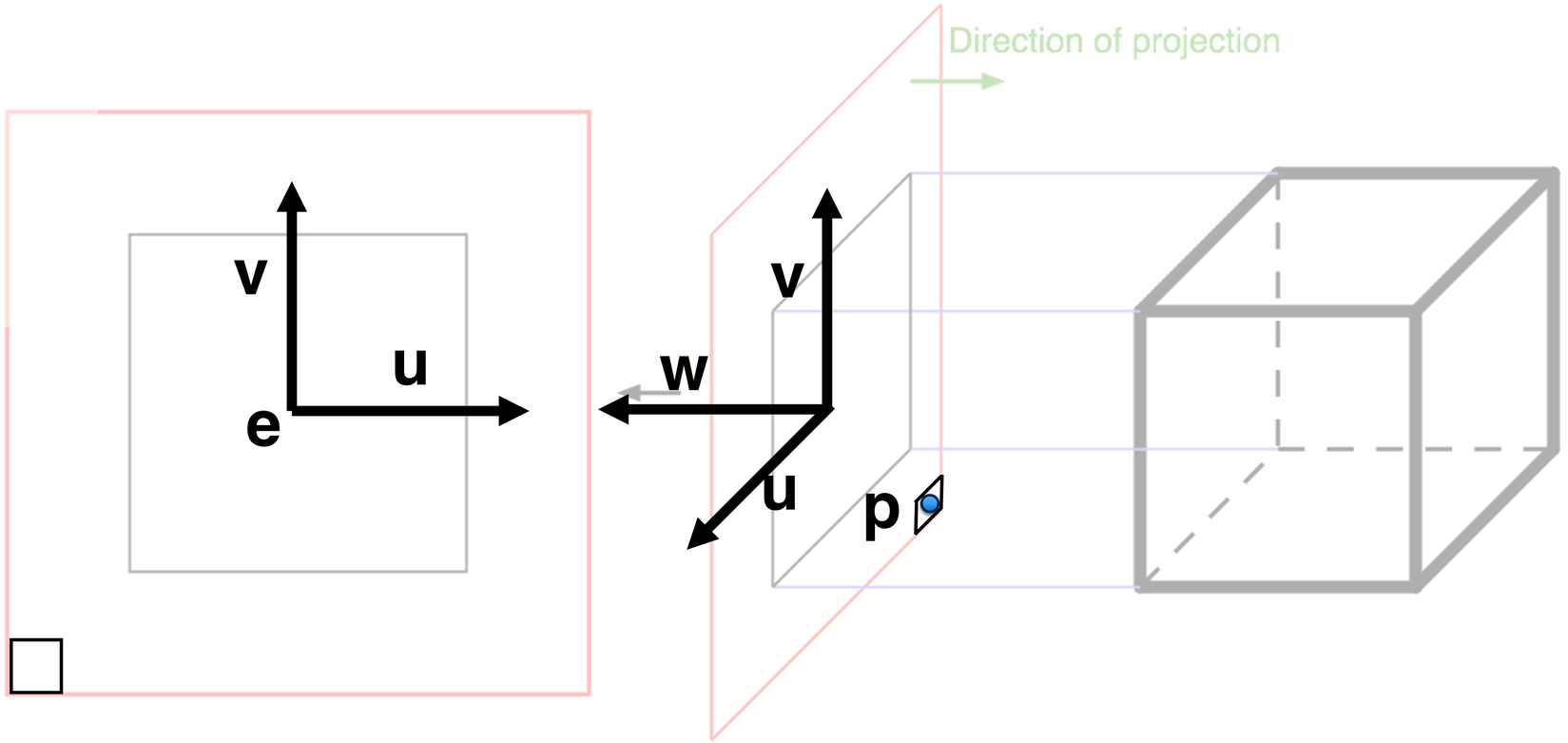
world space



canonical
view volume

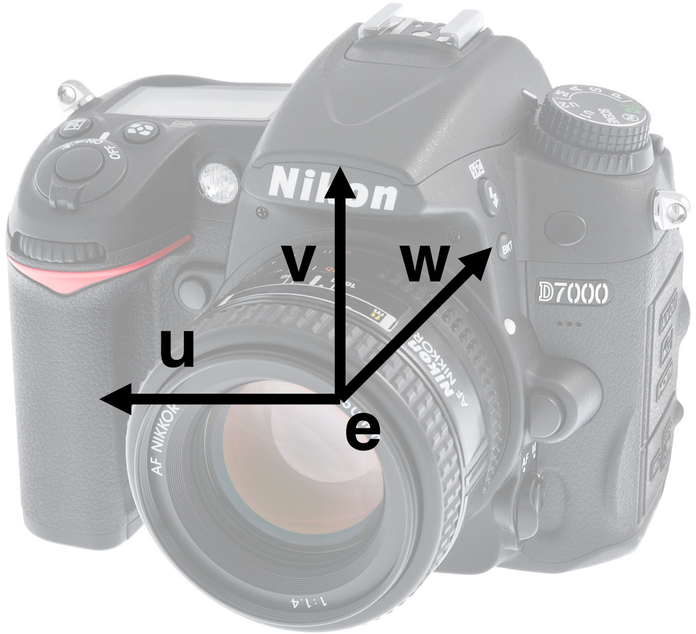
Orthographic

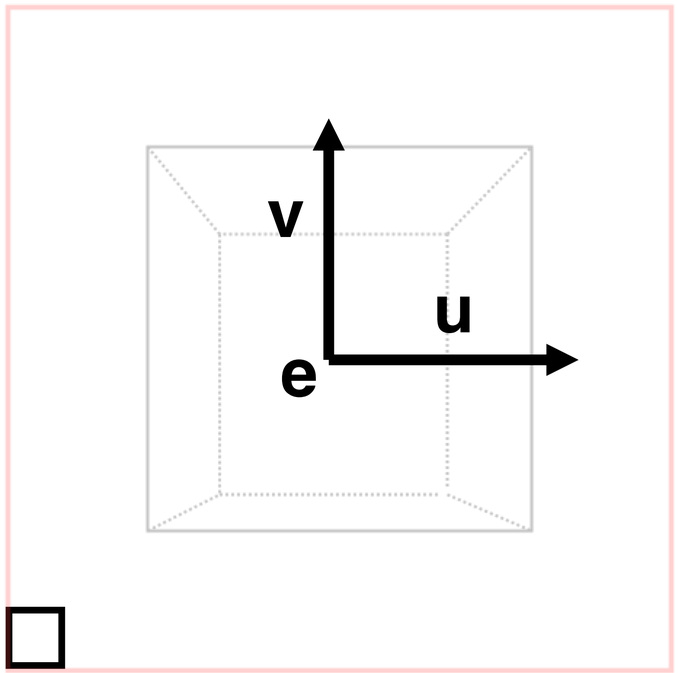


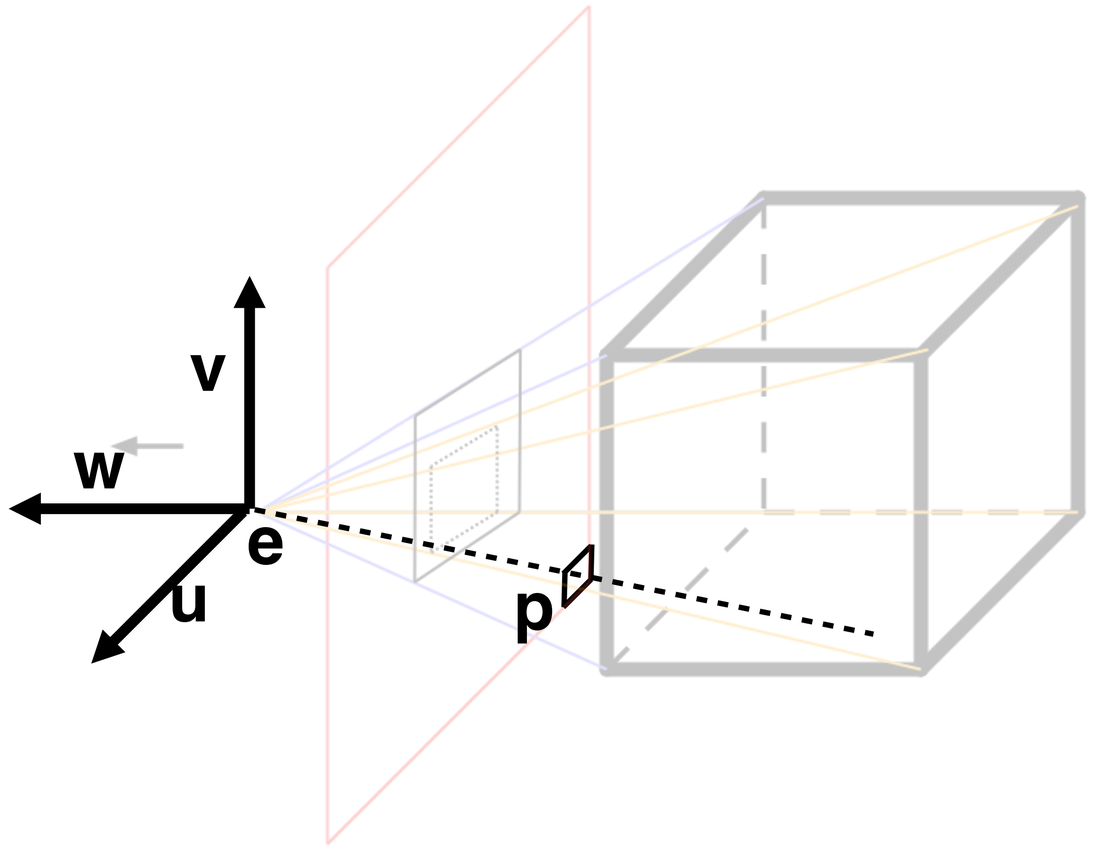


- For the ray assigned to pixel **p**:
  - Origin: **p**
  - Direction: **-w**

Perspective



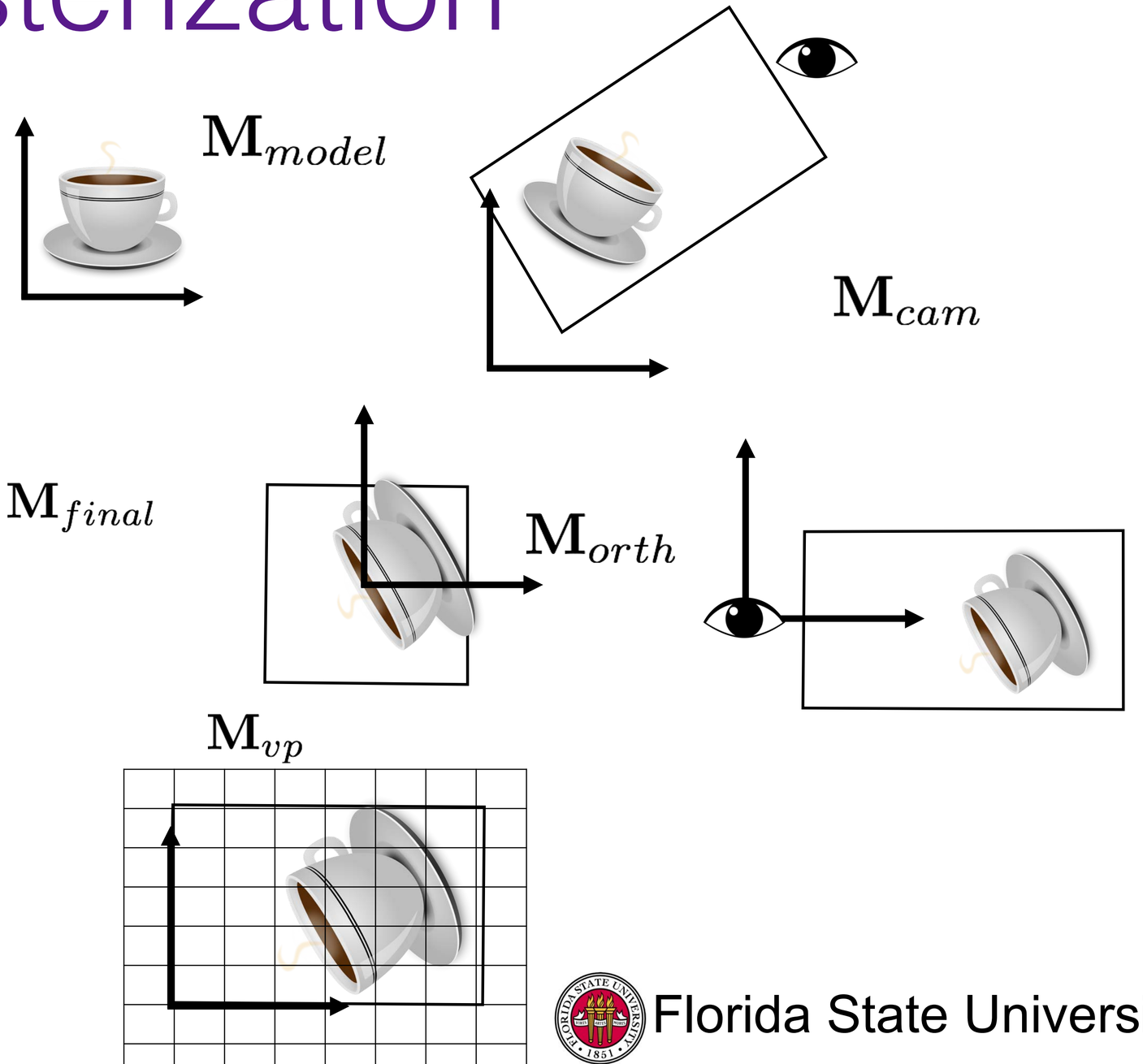


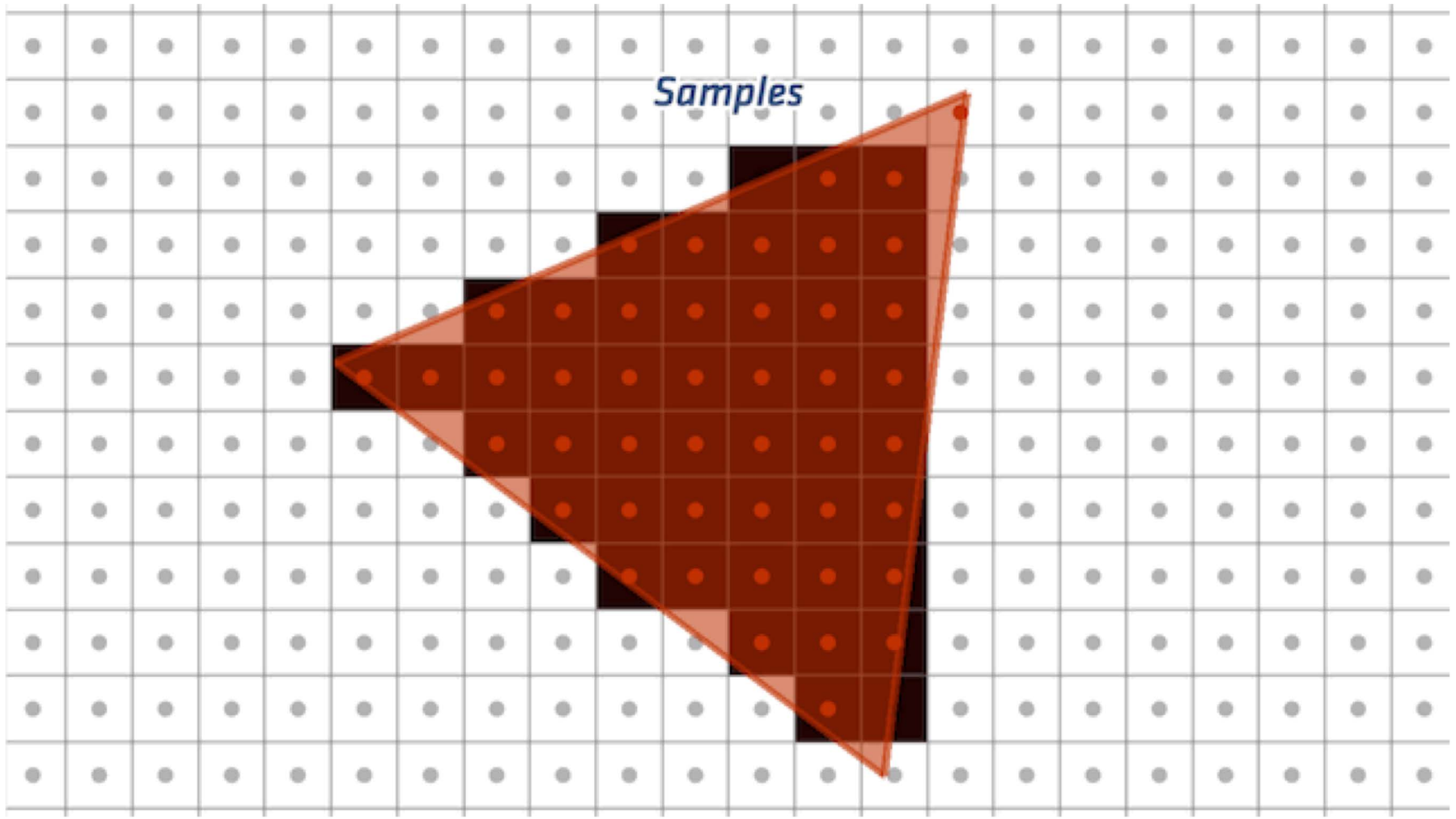


- For the ray assigned to pixel **p**:
  - Origin: **e**
  - Direction: **p - e**

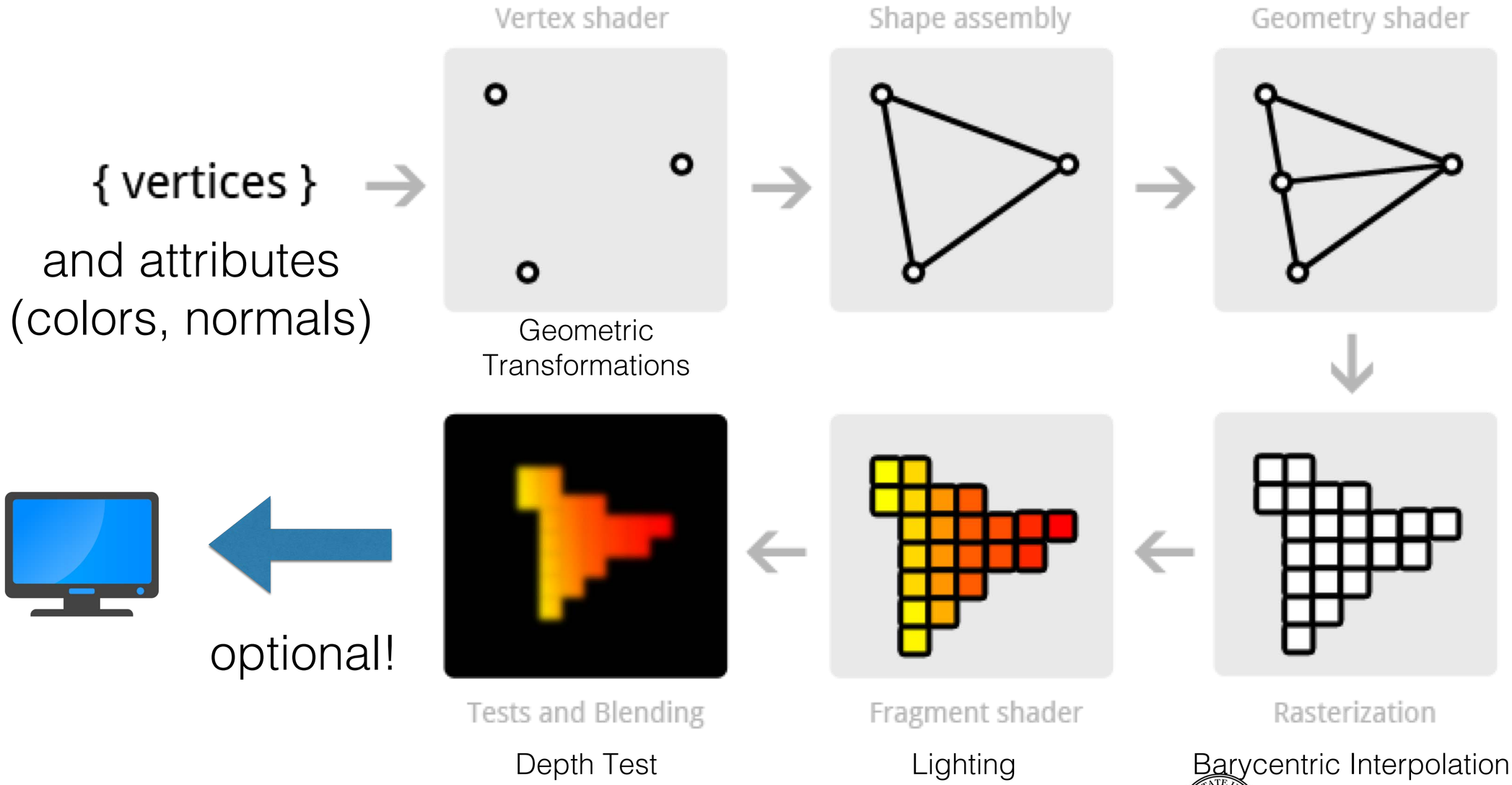
Rasterization

- Construct Viewport Matrix $\mathbf{M}_{vp}$
- Construct Projection Matrix $\mathbf{M}_{orth}$
- Construct Camera Matrix $\mathbf{M}_{cam}$
- $\mathbf{M} = \mathbf{M}_{vp}\mathbf{M}_{orth}\mathbf{M}_{cam}$
- For each model
  - Construct Model Matrix $\mathbf{M}_{model}$
  - $\mathbf{M}_{final} = \mathbf{M}\mathbf{M}_{model}$
  - For every point **p** in each primitive of the model
    - $\mathbf{p}_{final} = \mathbf{M}_{final}\mathbf{p}$
  - Rasterize the model

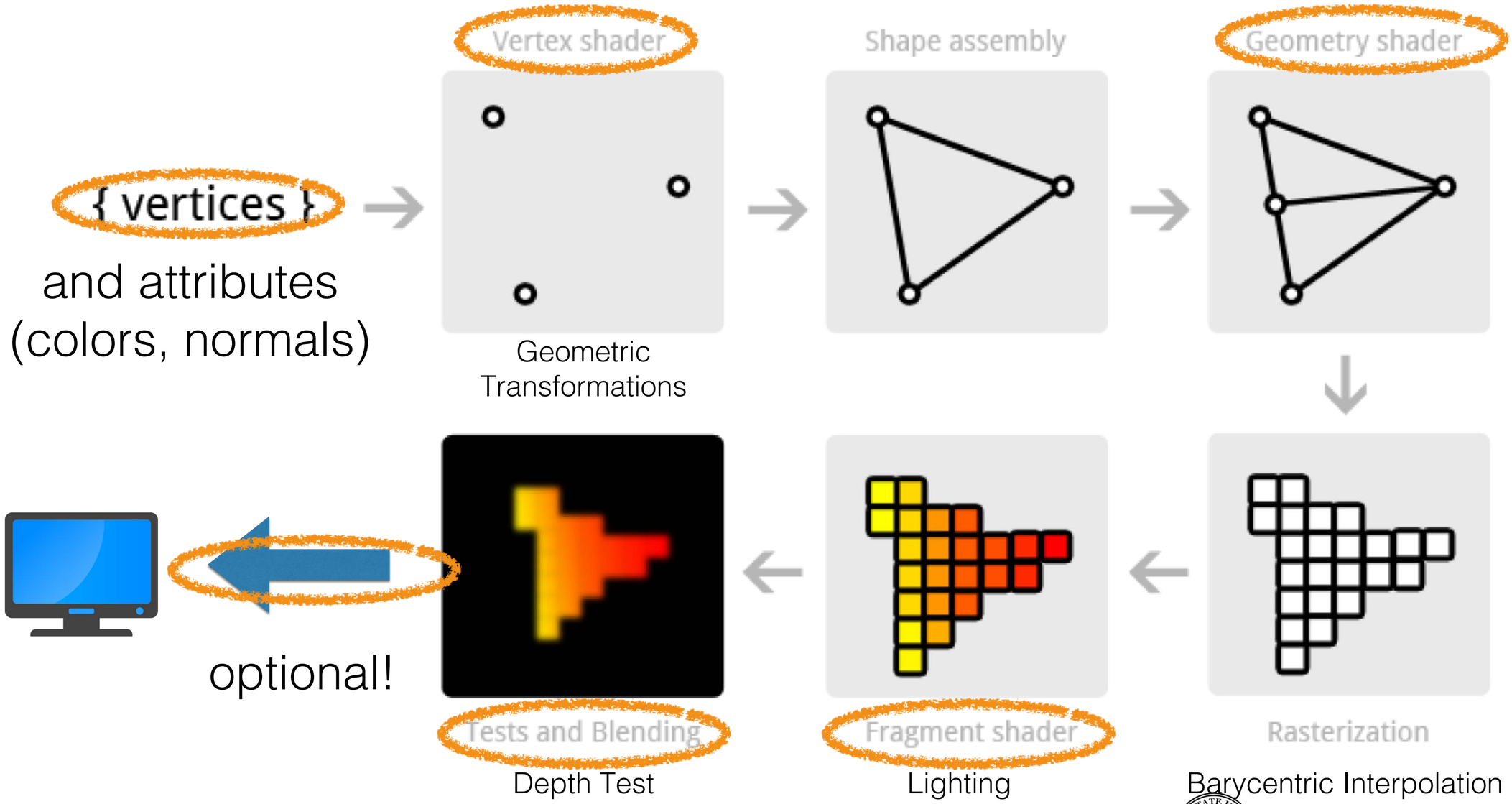



Samples

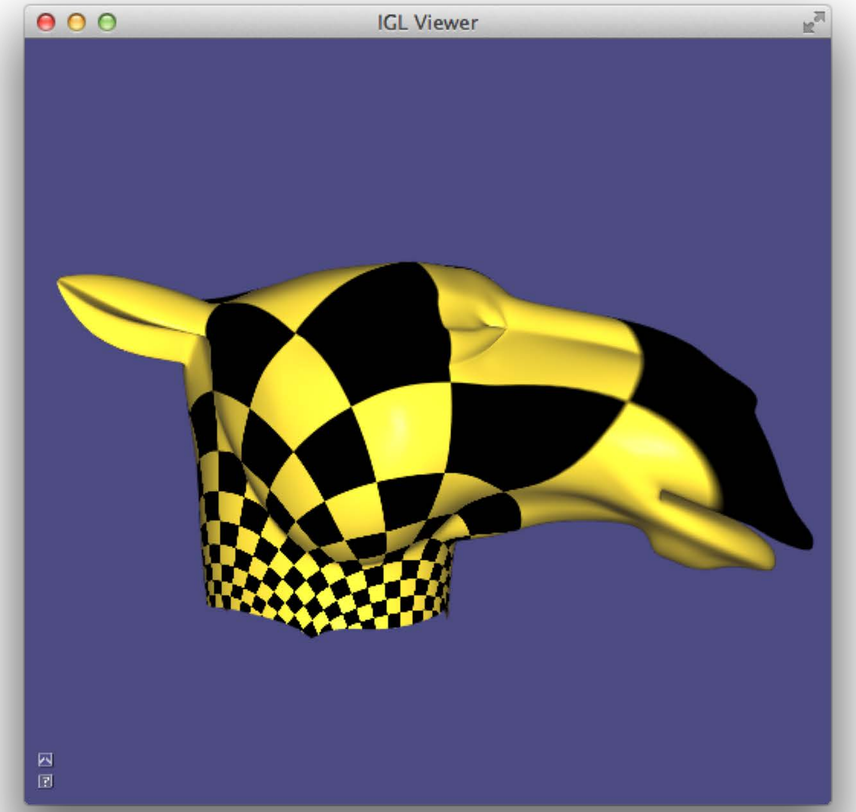
OpenGL pipeline

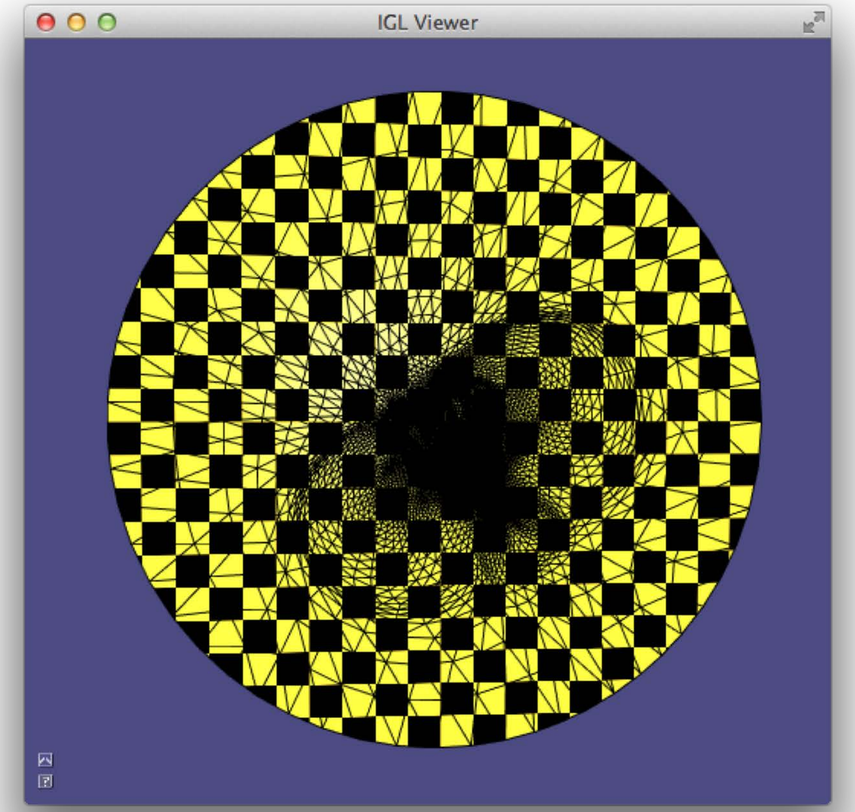


OpenGL pipeline



Demo



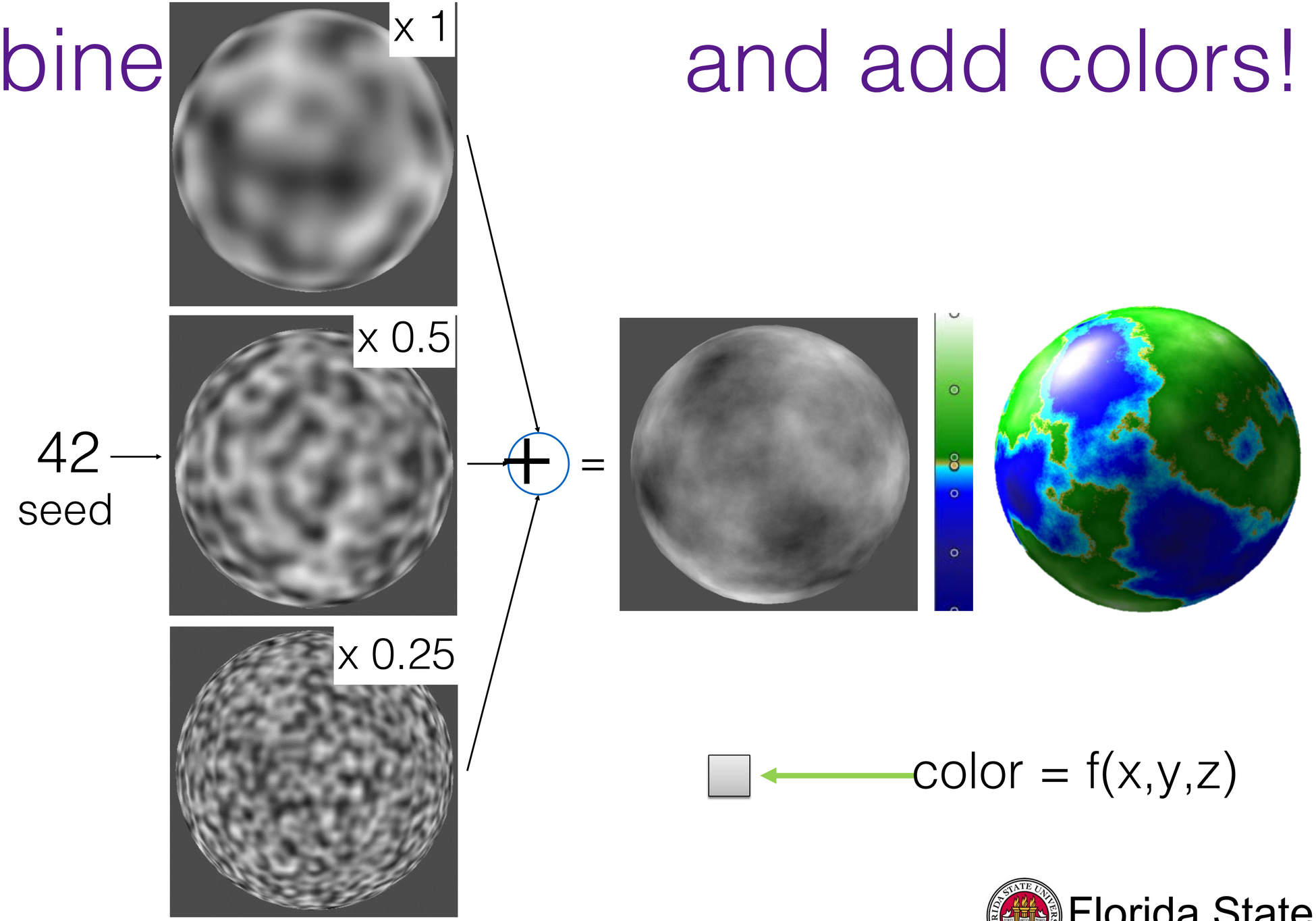


https://libigl.github.io/libigl/tutorial/tutorial.html#harmonicparametrization

“Seams” are needed for complex objects

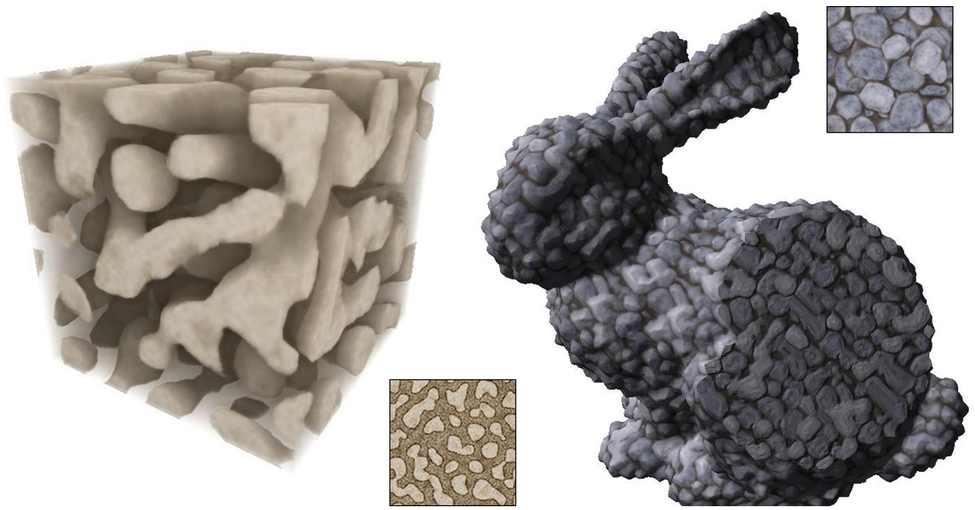
Image from Vallet and Levy, techreport INRIA

Combine and add colors!

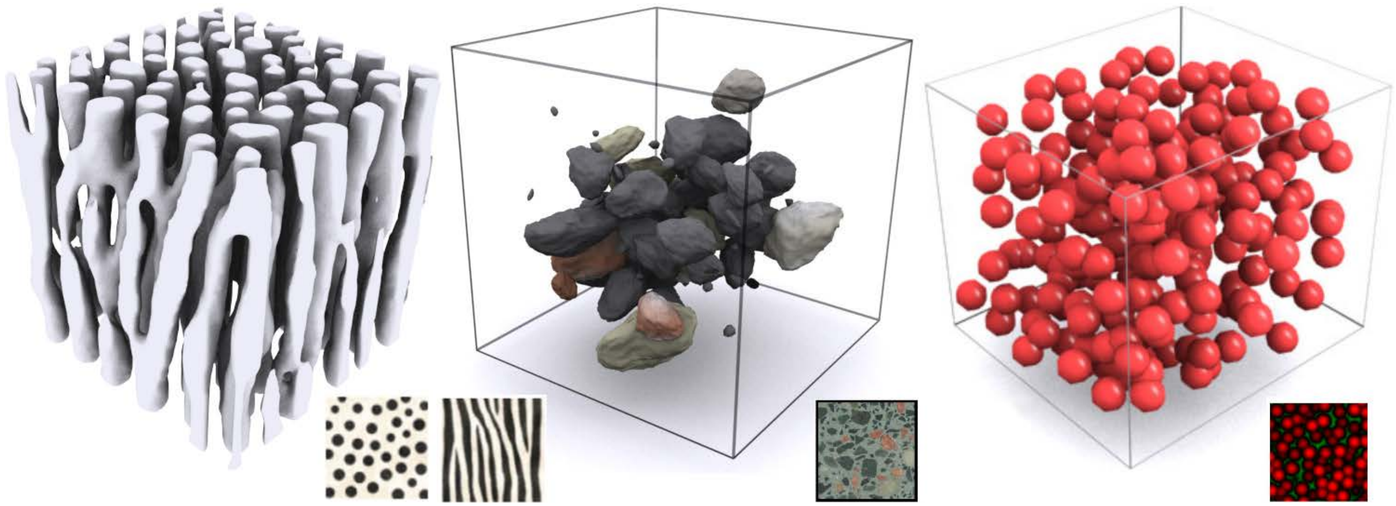


color = f(x,y,z)

By-Example Solid Textures



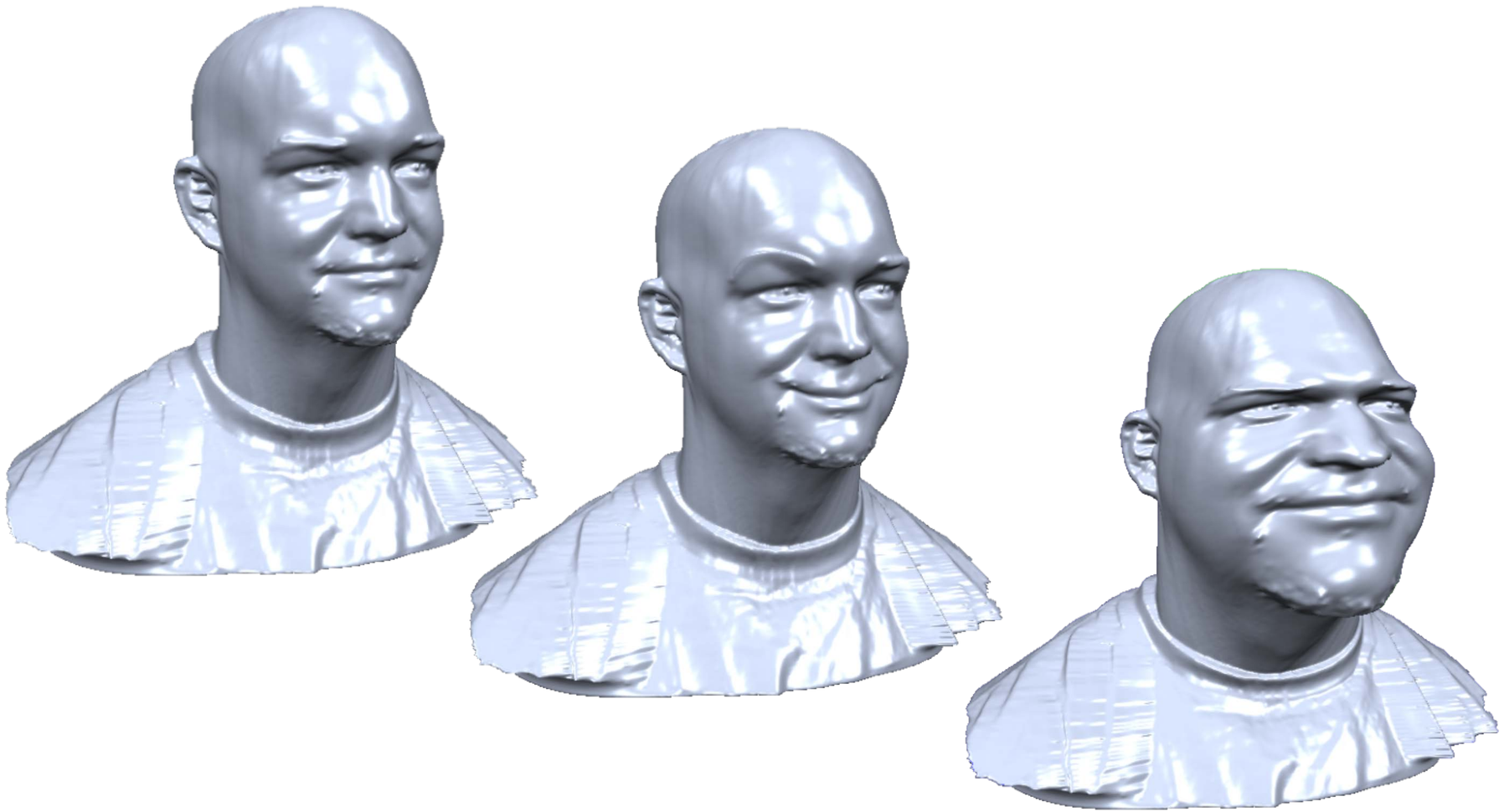
Solid Texture Synthesis [Kopf07]



Lazy Solid Texture Synthesis [Dong08]

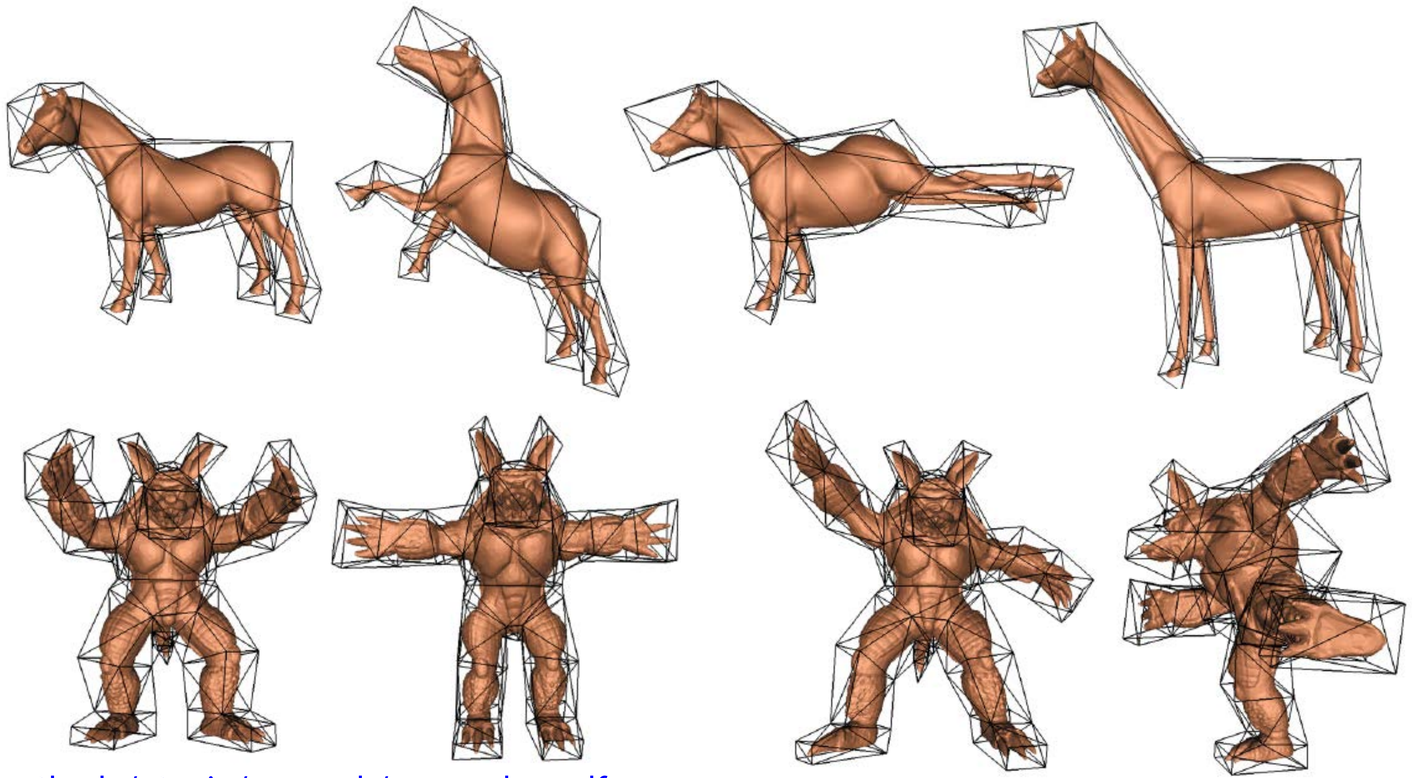
Local & Global Deformations

[Real-Time Shape Editing using Radial Basis Functions, Botsch and Kobbelt, EUROGRAPHICS 2005]



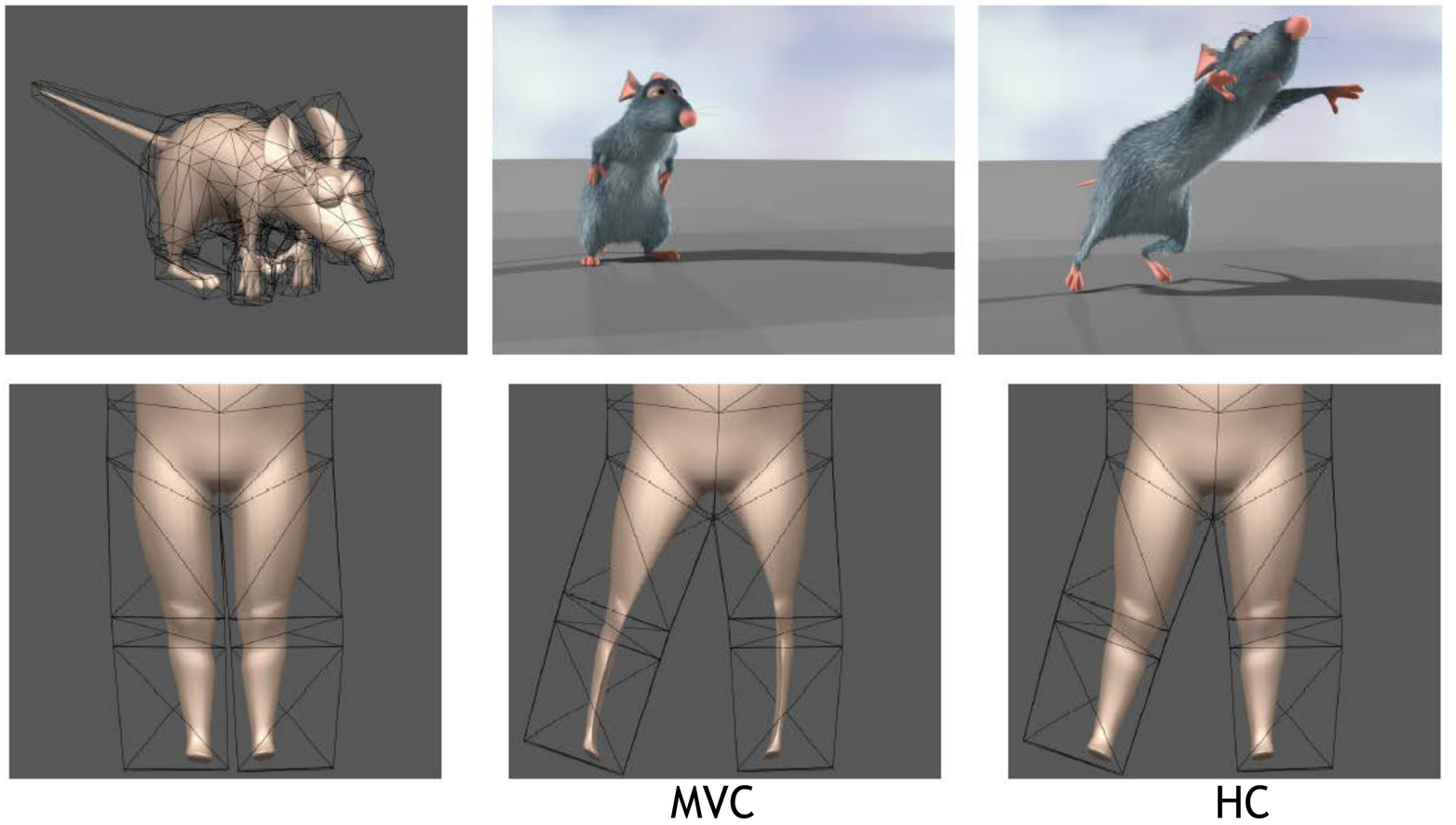
Cage-based Deformations

[Ju et al. 2005]

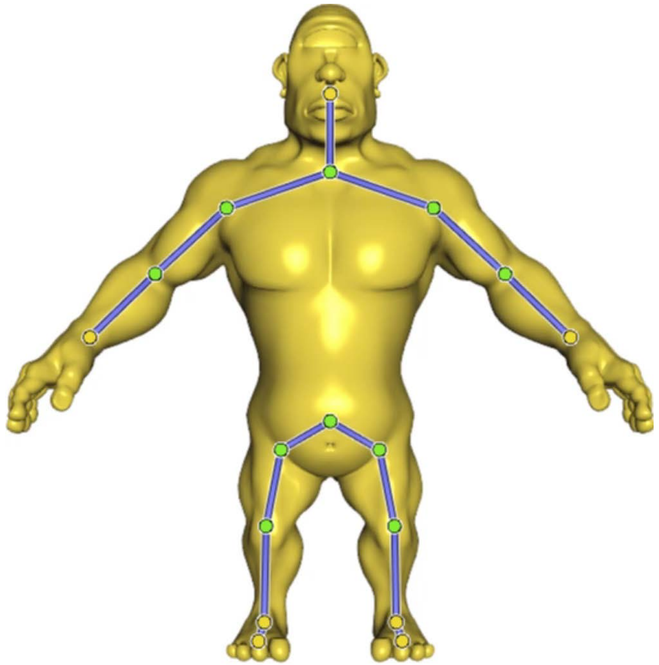


http://www.cs.wustl.edu/~taoju/research/meanvalue.pdf

Harmonic coordinates (Joshi et al. 2007)

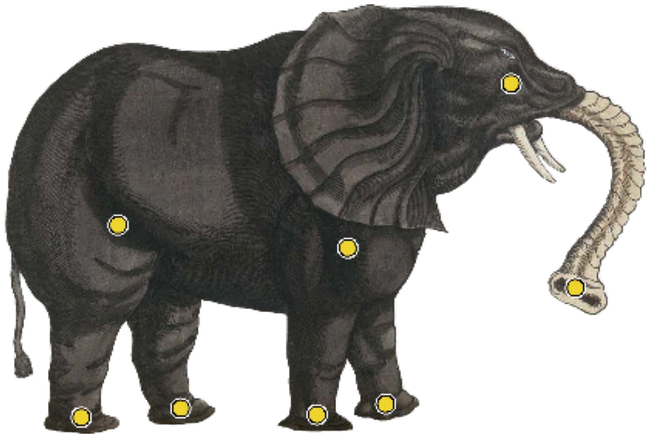


LBS generalizes to different handle types

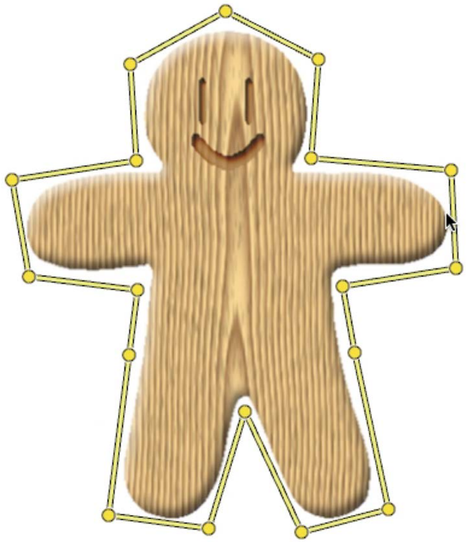


skeletons

regions

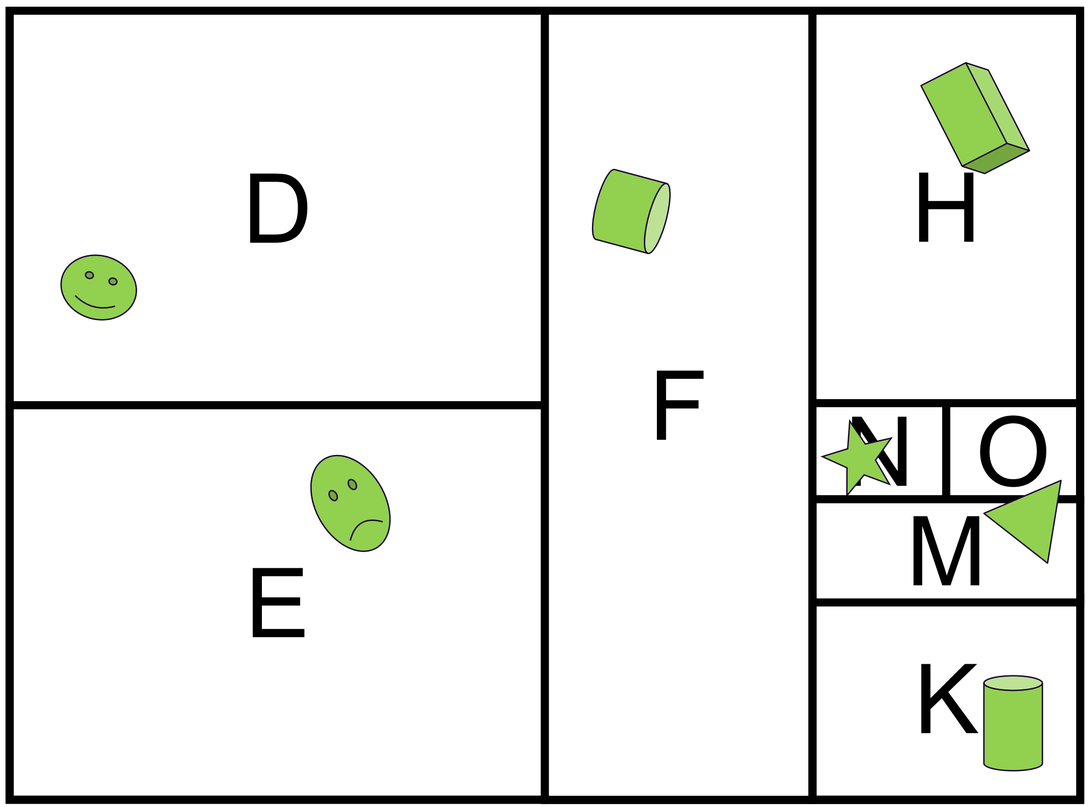


points



cages

kD-tree



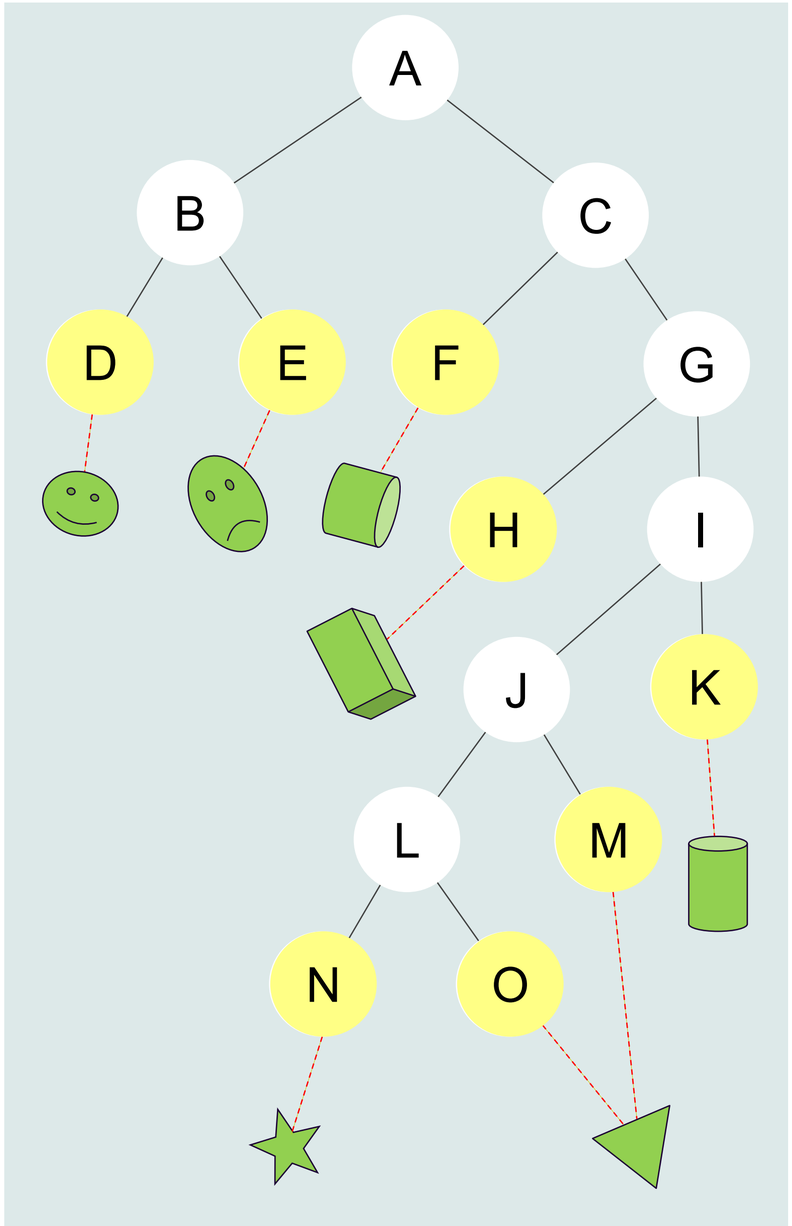
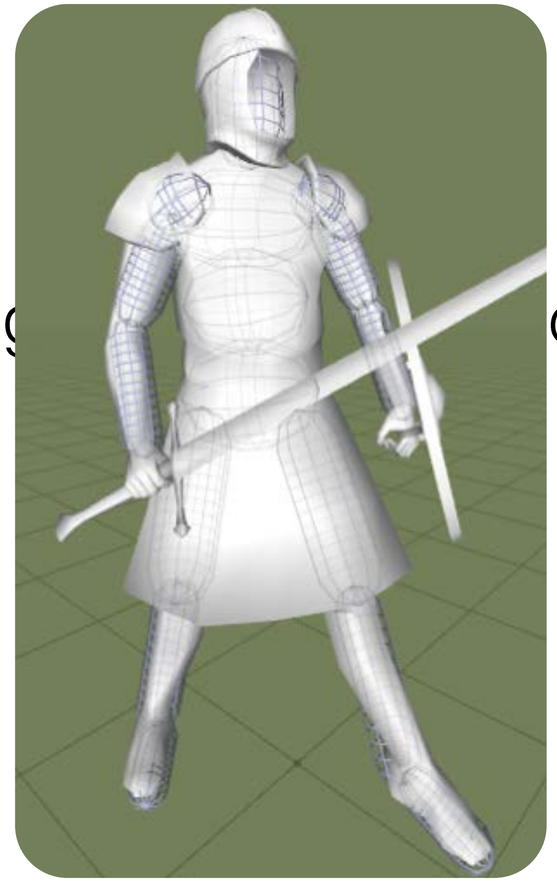
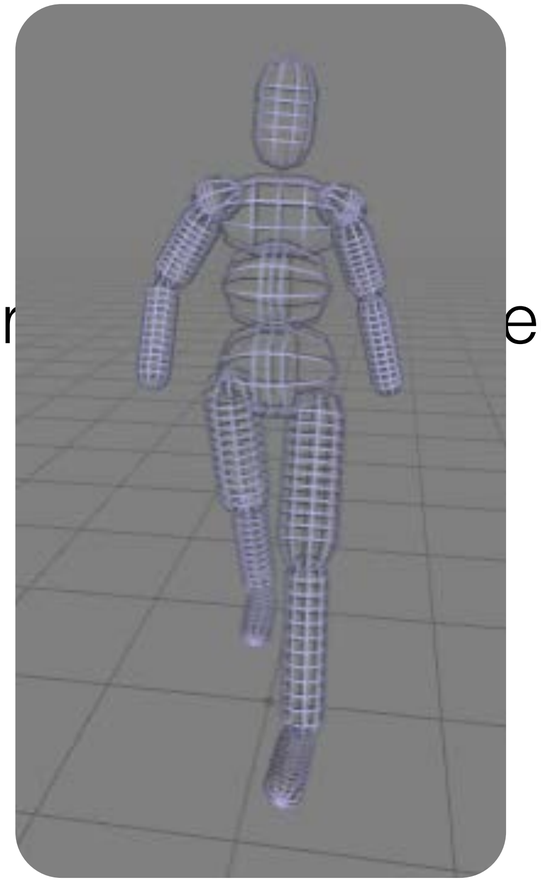


Image Copyright: Marco Tarini

Geometric Proxies

- Idea: use a g o appr ects in the scene





Final Project

- Time and location option
- Tuesday, Dec 11, LOV 103, 10:00 am – 12:00 pm
- Thursday, Dec 13, LOV 151, 2:00 pm – 4:00 pm
- Friday, Dec 14, LOV 151, 8:00 am – 9:30 am
- Submit the entire package, source code and readme, through CANVAS
- 8 mins for each presenter, in the alphabetical order of the last name