Computer Graphics and Visual Computing: Introduction and Overview

Hong Qin
Center for Visual Computing (CVC)
Stony Brook University (SUNY Stony Brook)

Presentation Outline

- Computer graphics as a basic computational tool for visual computing
- Various applications
- 3D graphics pipeline
- Programming basics

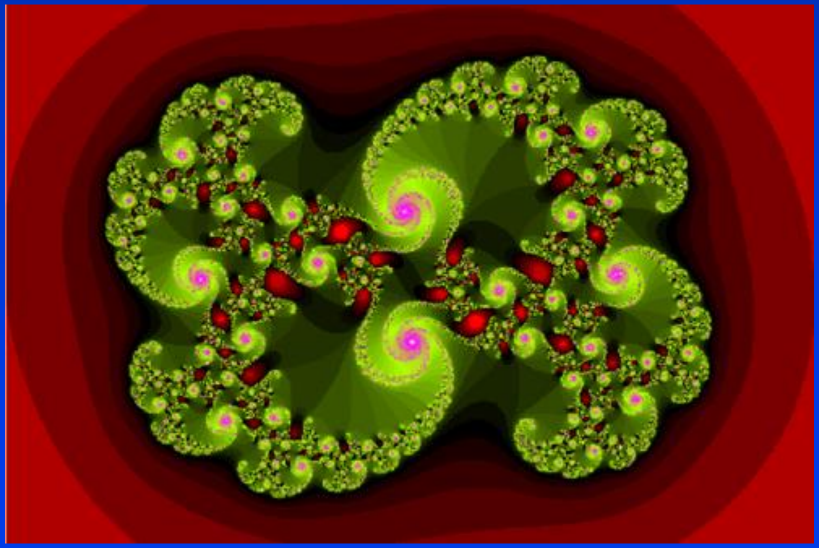
What is Computer Graphics?

- The creation of, manipulation of, analysis of, and interaction with pictorial representations of objects and data using computers.

  \- Dictionary of Computing

- A picture is worth a thousand words.

  \- Chinese Proverb



1000 words (or just 94 words), many letters though…

It looks like a swirl. There are smaller swirls at the edges. It has different shades of red at the outside, and is mostly green at the inside. The smaller swirls have purple highlights. The green has also different shades. Each small swirl is composed of even smaller ones. The swirls go clockwise. Inside the object, there are also red highlights. Those have different shades of red also. The green shades vary in a fan, while the purple ones are more uni-color. The green shades get darker towards the outside of the fan …

Computer Graphics Definition

- What is Computer Graphics?
  - (Realistic) Pictorial synthesis of real and/or imaginary objects from their computer-based models (or datasets)
- Fundamental, core elements of computer graphics
  - Modeling: representation choices, geometric processing
  - Rendering: geometric transformation, visibility, simulation of light
  - Interaction: input/output devices, tools
  - Animation: lifelike characters, natural phenomena, their interactions, surrounding environments
- So, we are focusing on computer graphics hardware, software, and mathematical foundations
- Computer Graphics is computation
  - A new method of visual computing
- Why is Computer Graphics useful and important?

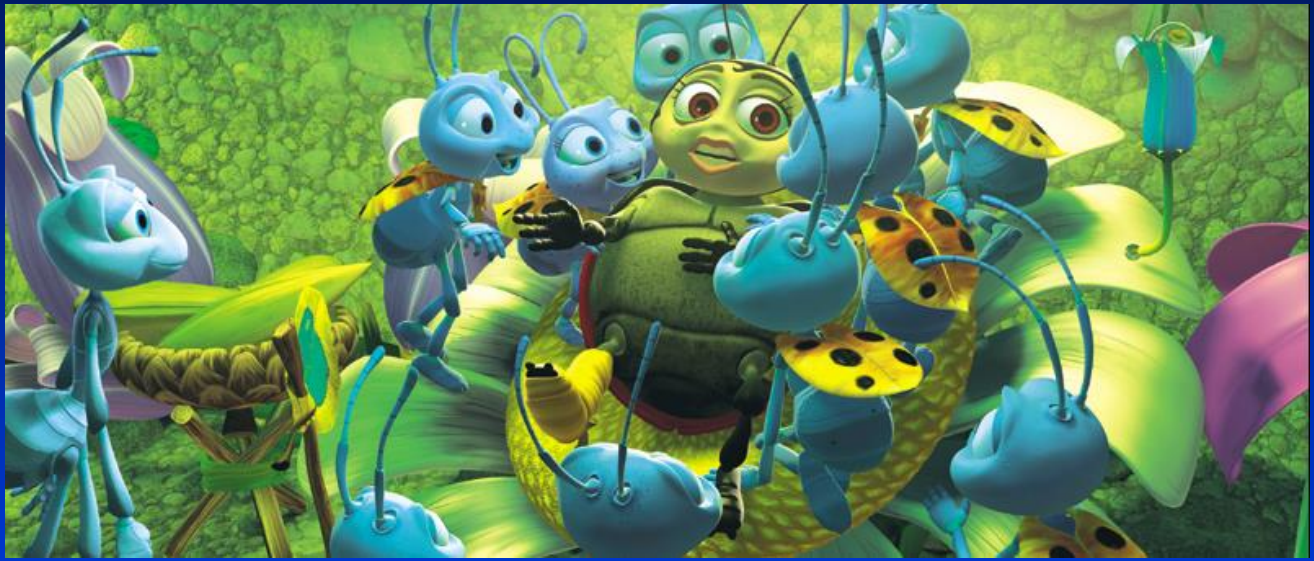
Why Computer Graphics?

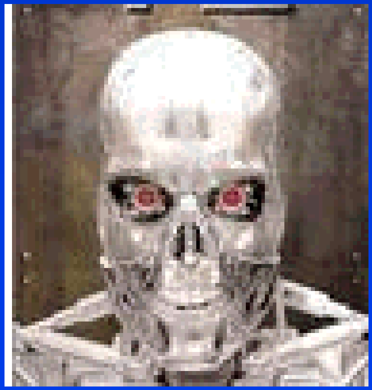
- About 50% of the brain neurons are associated with vision
- Dominant form of computer output
- Enable scientists (also engineers, physicians, and general users) to observe their simulation and computation
- Enable them to describe, explore, and summarize their datasets (models) and gain insights
- Enrich the discovery process and facilitate new inventions

Why Computer Graphics?

- Applications (In essence, computer graphics is application-driven)
  - Entertainment: Movies, Video games
  - Graphical user interface (GUI)
  - Computer aided design and manufacturing (CAD/CAM)
  - Engineering analysis and business
  - Medical applications
  - Computer Art
  - Engineering Analysis
  - Scientific visualization / simulation
  - Virtual Reality
  - others

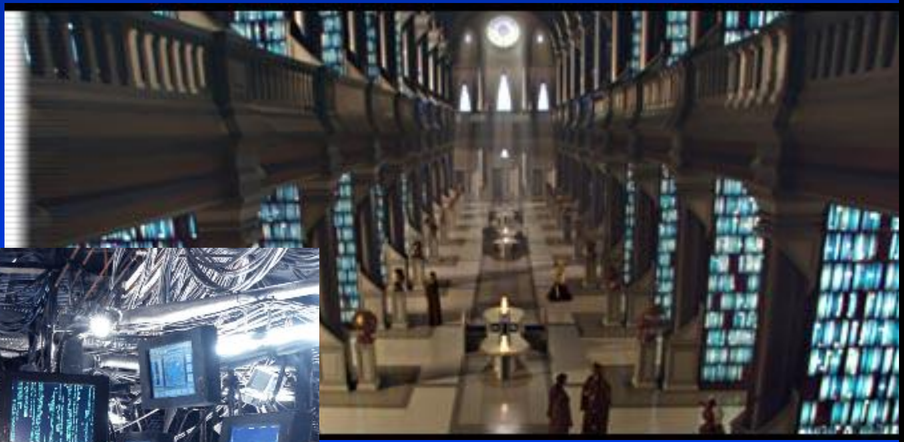
Digital Entertainment





Movies

- If you can image it, it can be done with computer graphics!
- More than one billion dollars on special effects.
- No end in sight for this trend!



Movies

"The Day After Tomorrow"

Video Games

- Important driving force
- Focus on interactivity
- Try to avoid computation and use various tricks

Games

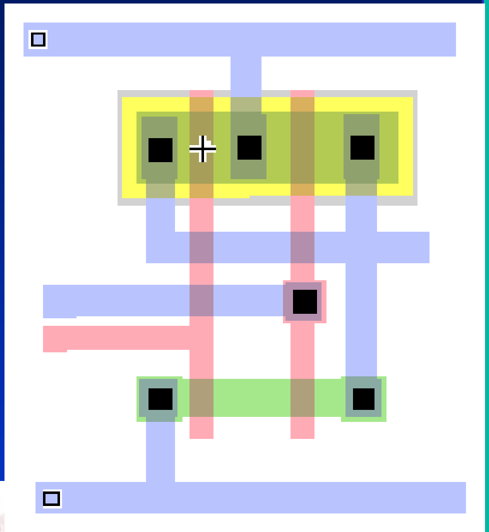
Quake III

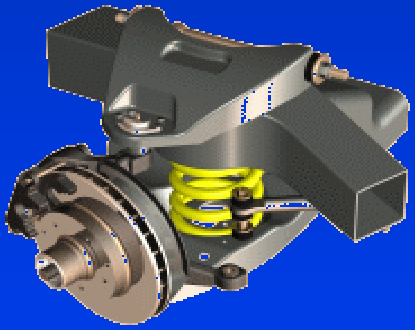
Metroid Pride

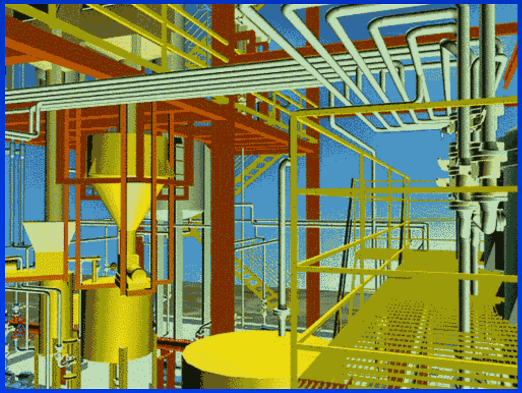
Doom

Computer-Aided Design

- Significant impact on the design process
- Mechanical, electronic design
  - entirely on computer
- Architectural and product design
  - Migrate to the computer



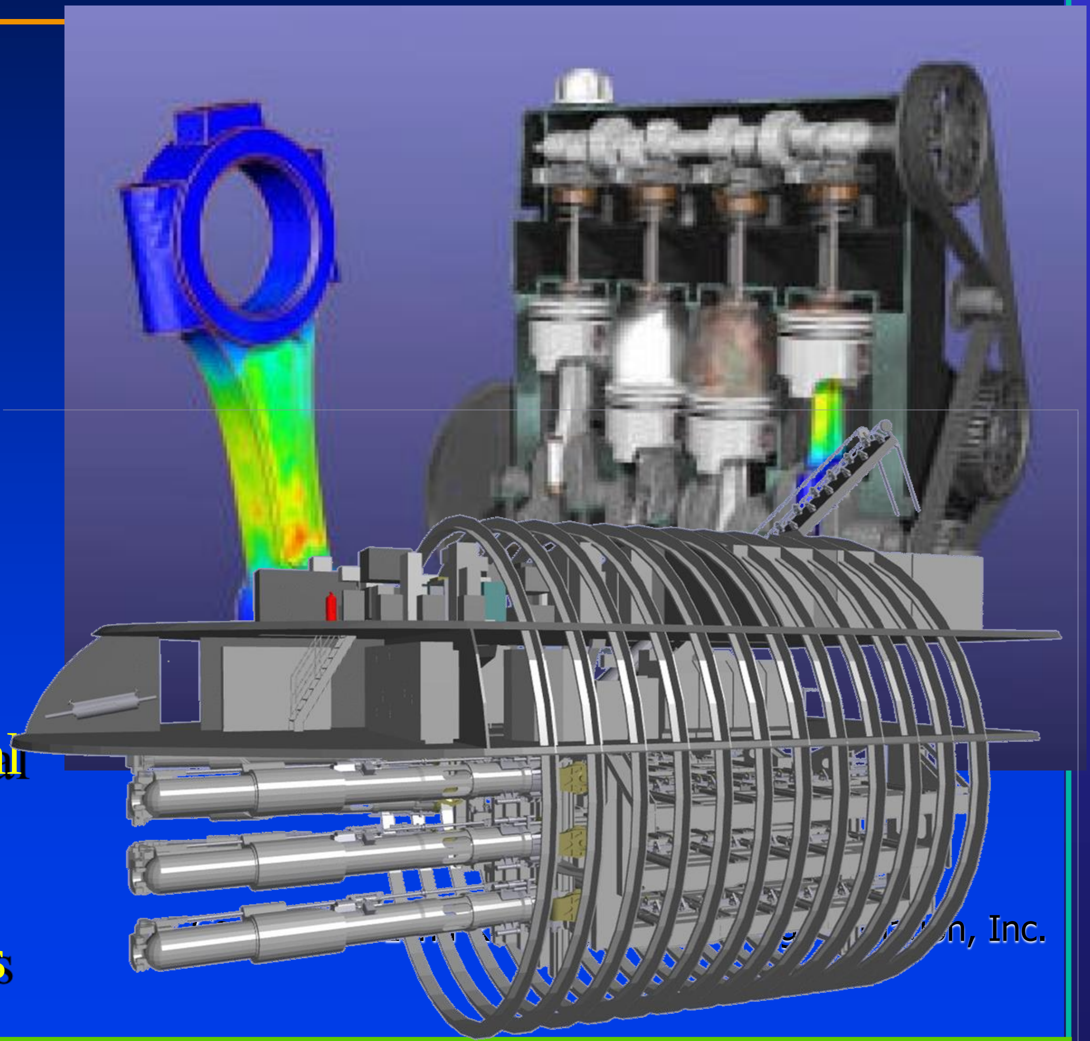




UGS: towards virtual manufacturing

Engineering Design

- Engineering & Architecture Software
- Buildings, aircraft, automobile, computers, appliances, etc.
- Interactive design (mesh editing, wire-frame display, etc.)
- Standard shape database
- Design of structural component through numerical simulation of the physical operating environment
- Testing: real-time animations



Architectural Design

- Architecture, Engineering, Construction
- Final product appearance: surface rendering, realistic lighting
- Construction planning: architects, clients can study appearance before actual construction

Courtesy of Craig Mosher & Ron Burdock, Peripheral Vision Animations

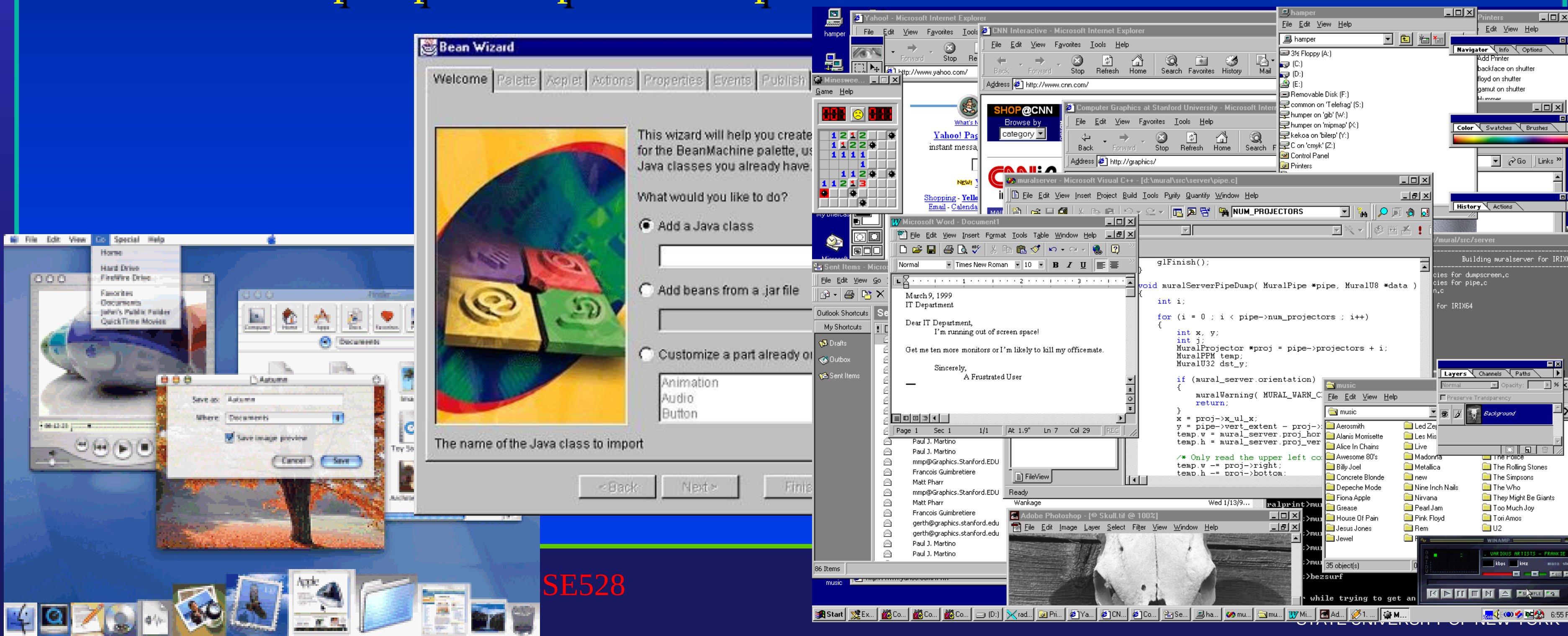
Textile Industry

- Fashion design
- Real-time cloth animation
- Web-based virtual try-on applications

Courtesy of Thalmann, Switzerland

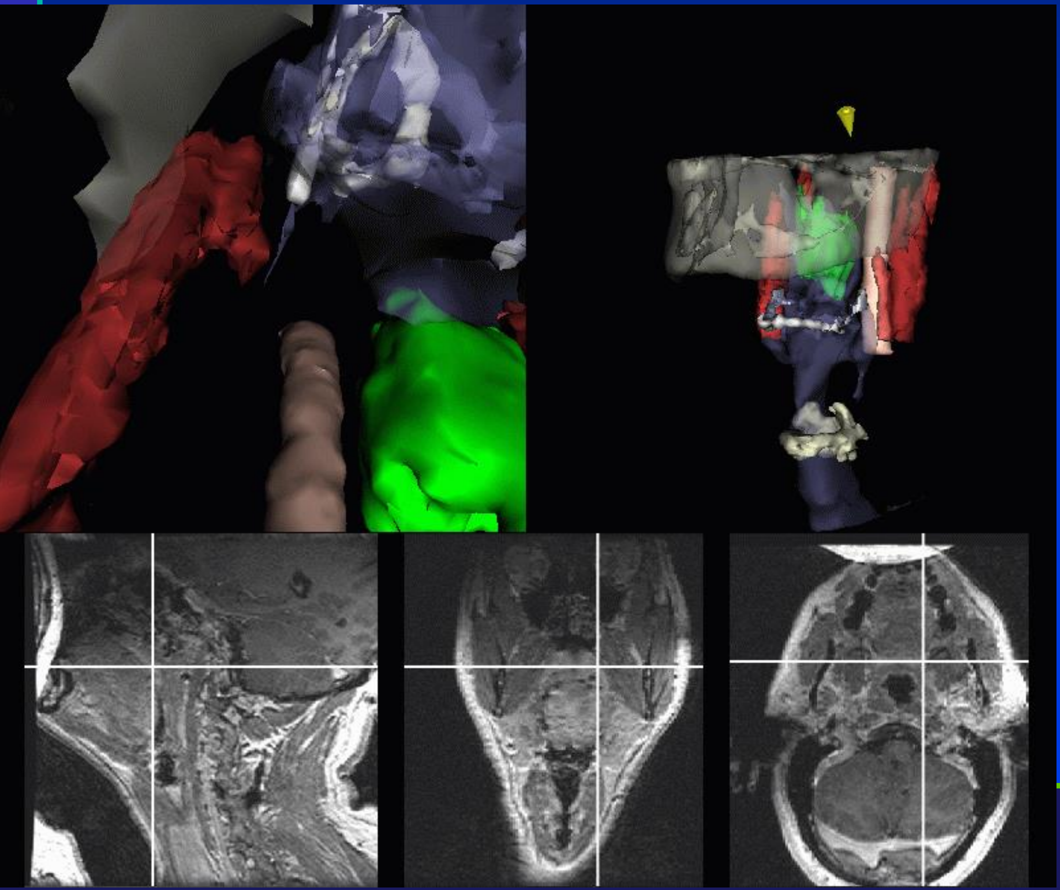
Graphical User Interface: GUI

- Integral part of everyday computing
- Graphical elements everywhere
  - Windows, cursors, menus, icons, etc.
- Nearly all professional programmers must have an understanding of graphics in order to accept input and present output to users.



Medical Applications

- Significant role in saving lives
- Training, education, diagnosis, treatment

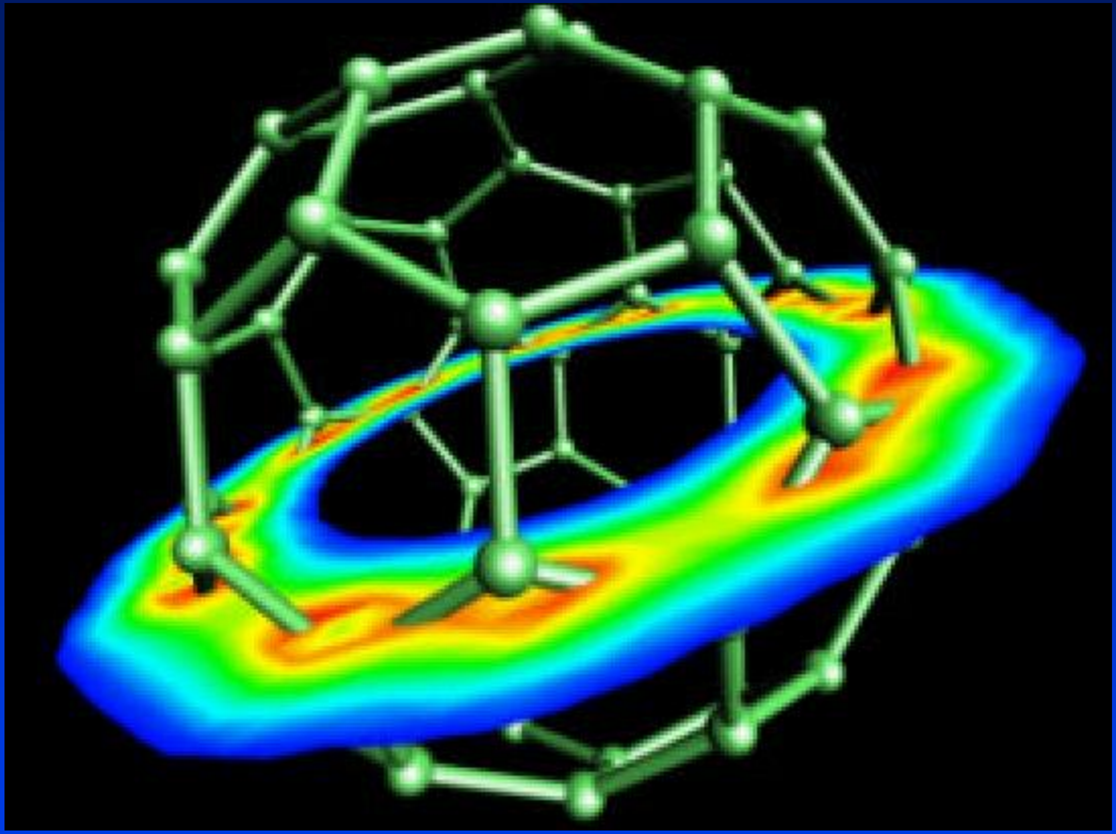


The Visible Human Project

Creation of complete, anatomically detailed 3D representation of human bodies.

Scientific Visualization

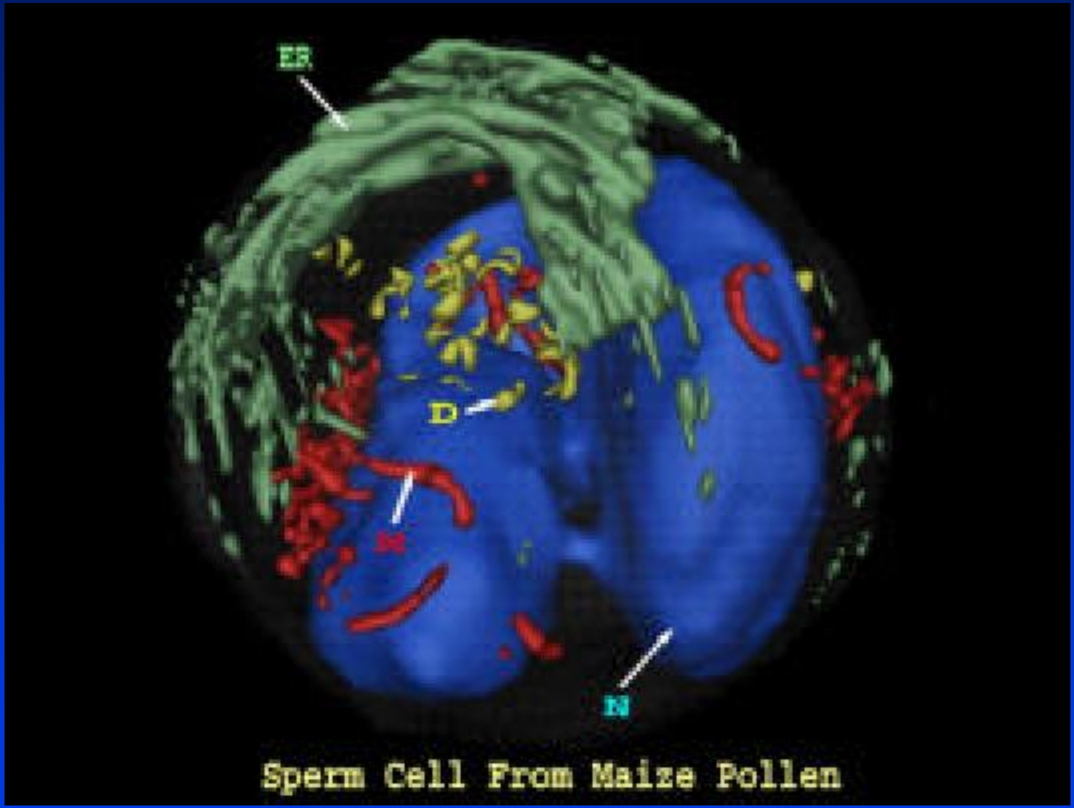
- Scientific data representation
- Picture vs. stream of numbers
- Techniques: contour plots, color coding, constant value surface rendering, custom shapes



Display of a 2D slice through the total electron density of C-60; Created by Cary Sandvig of SGI

Scientific Visualization

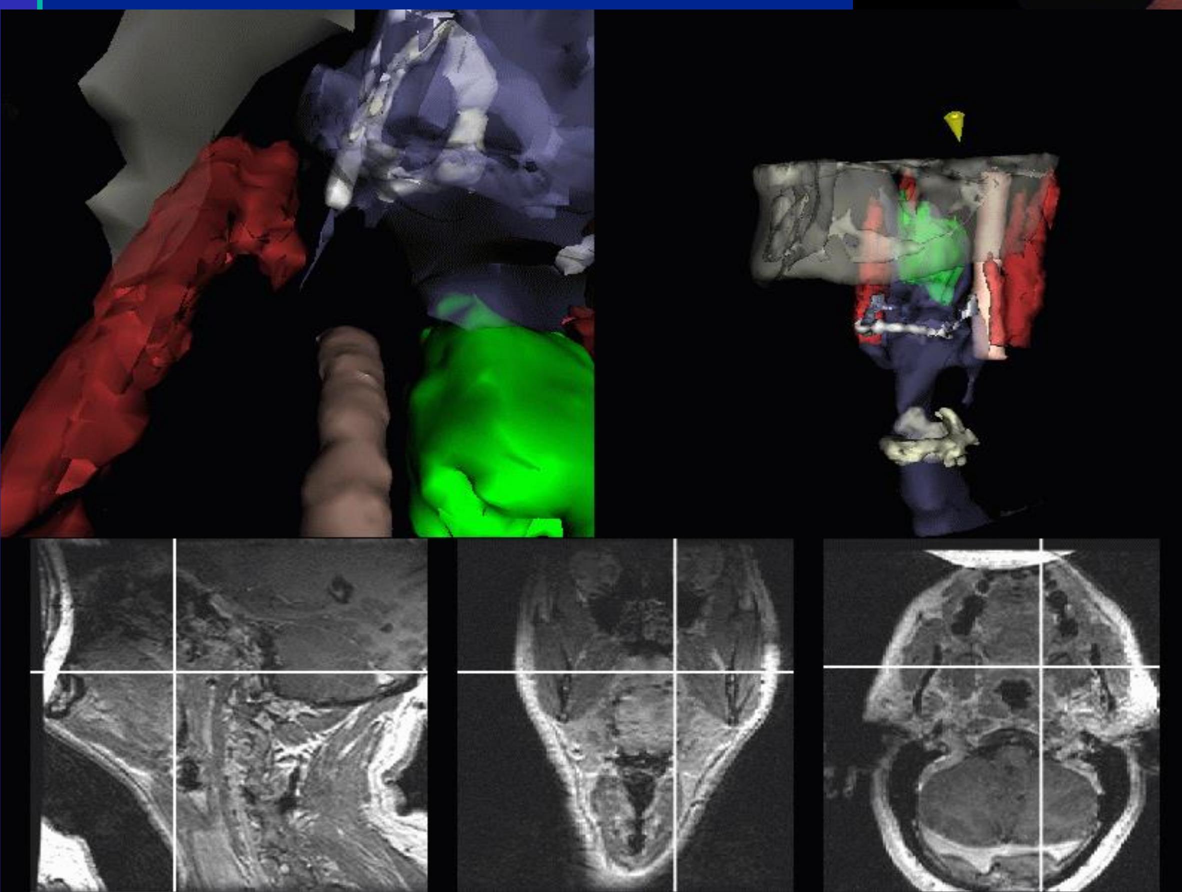
- Life Sciences
- Providing quantitative, three dimensional electron microscopy.
- Scientists can see structures as they were before being sectioned for viewing in the electron microscope.

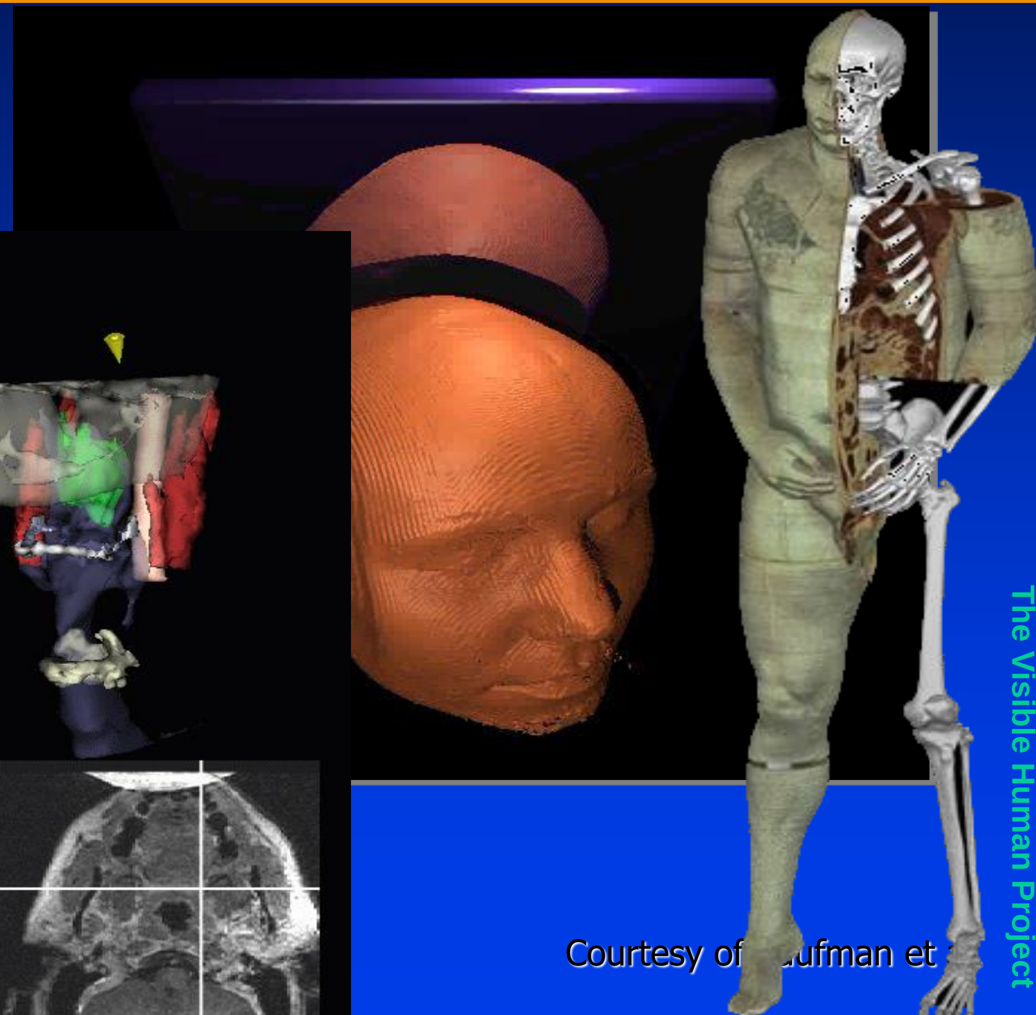


Courtesy of H. Lloyd Mogensen, Northern Arizona University

Scientific Visualization

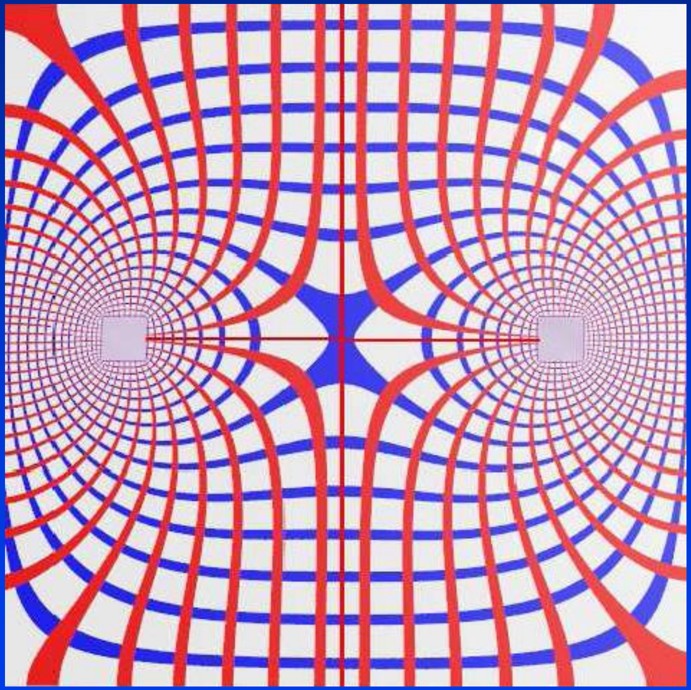
- Medical imaging & visualization



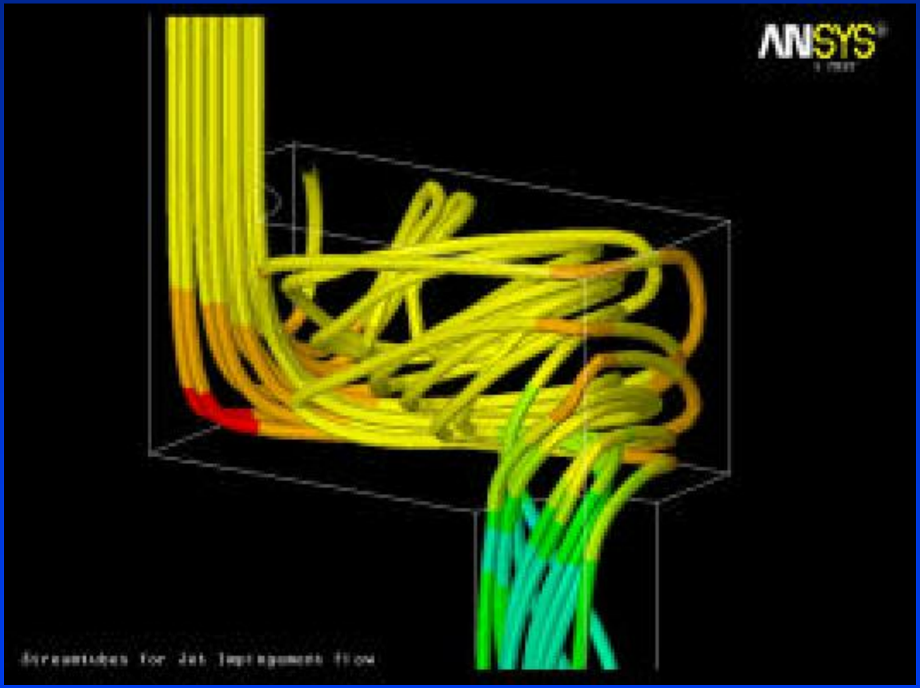


Scientific Visualization / Simulation

Electromagnetic potential field



Computational Fluid Dynamics (CFD)

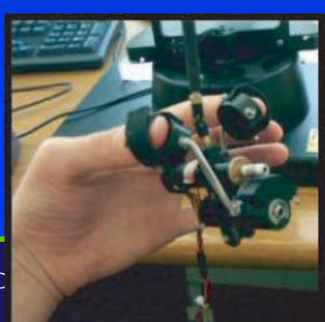


Courtesy of Mark Toscinski and Paul Tallon

Virtual Reality

- User interacts with objects in a 3D scene
- Special devices (input, output)
- Virtual walkthroughs
- Equipment training (pilots, surgeons, etc.)

Force reflecting gripper



Haptic devices

Force feedback exoskeleton

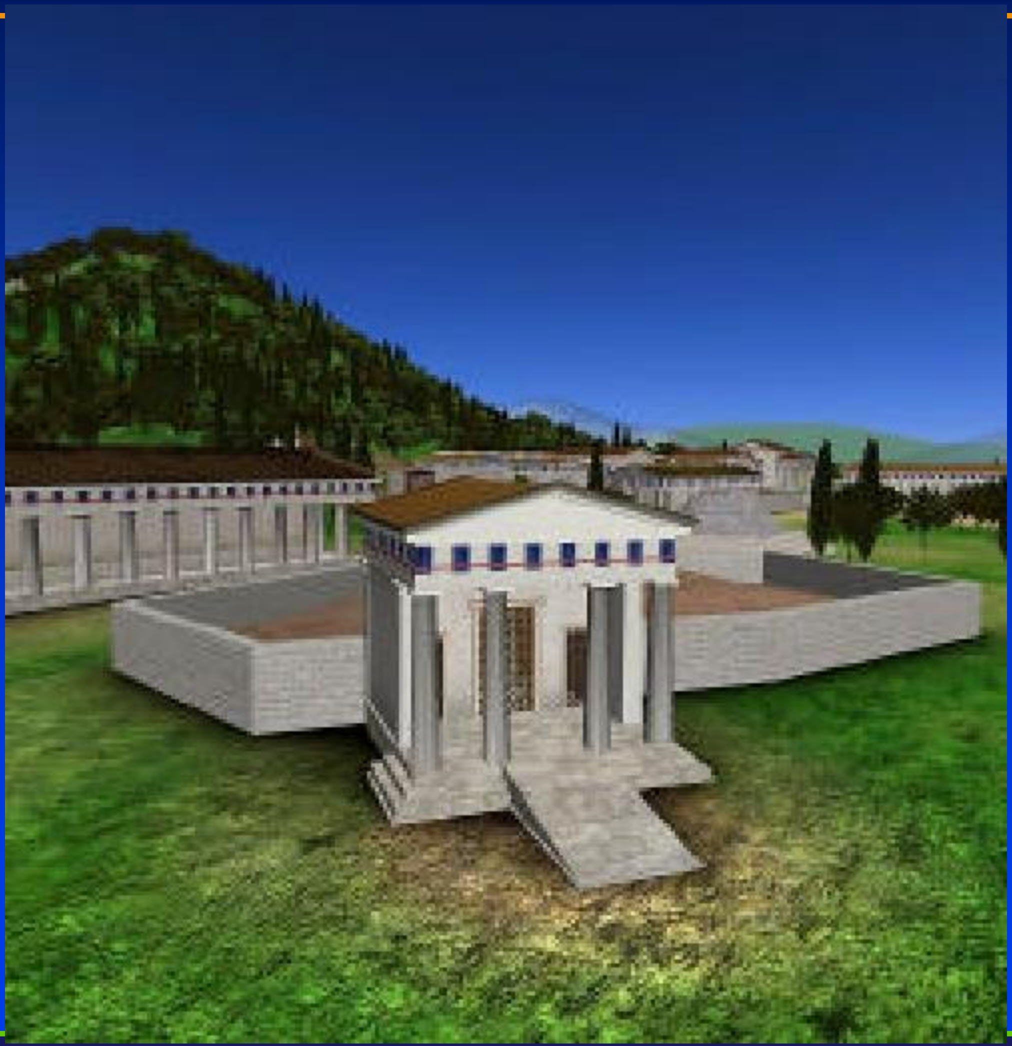
Haptic workstation

Virtual Reality

- Education using computer-generated system & process models
- Visual simulation:
  - Aircraft simulator
  - Spacecraft simulator
  - Naval craft simulator
  - Automobile simulator
  - Heavy machinery simulator
  - Surgery simulator
- Special hardware required

Virtual Reality

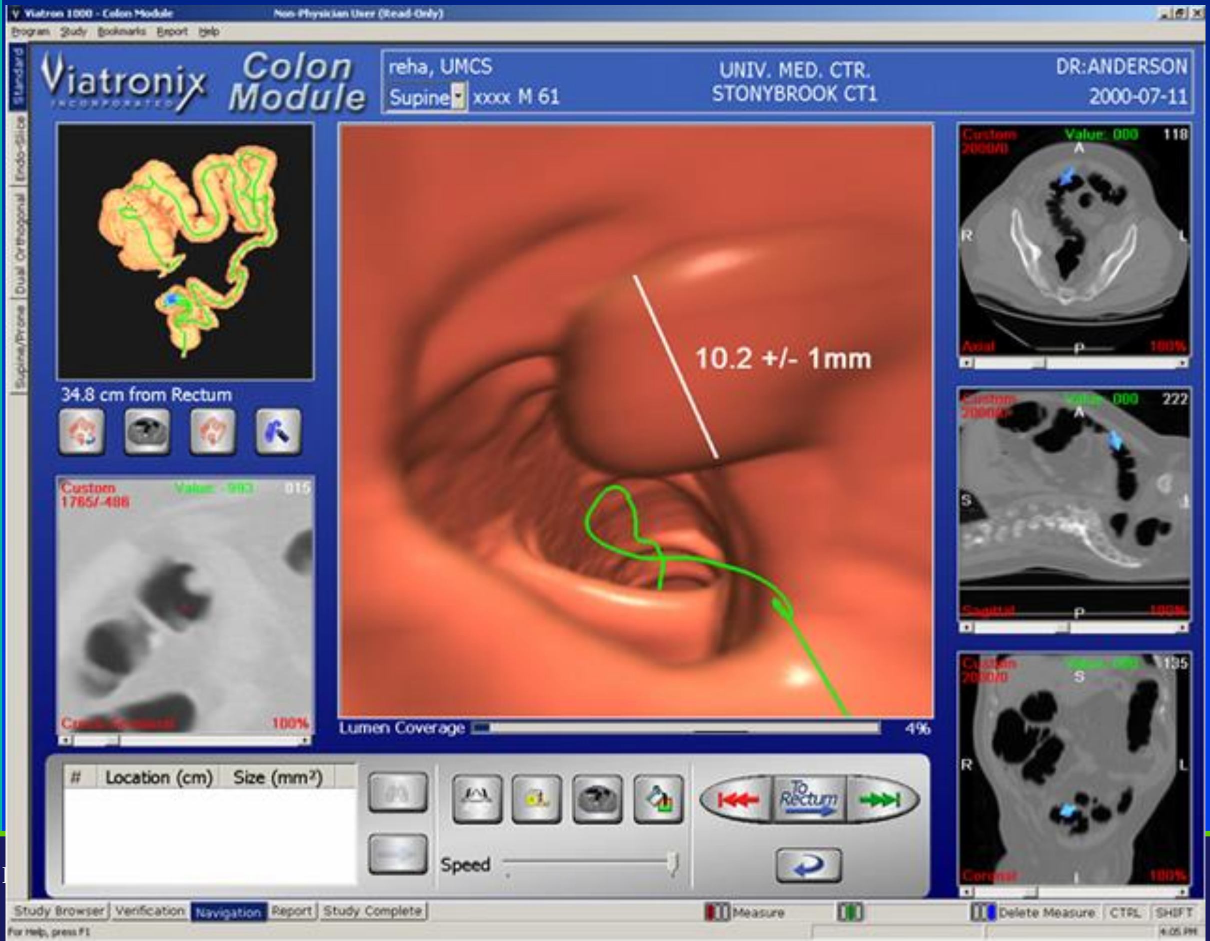
- Virtual tour of historical remains

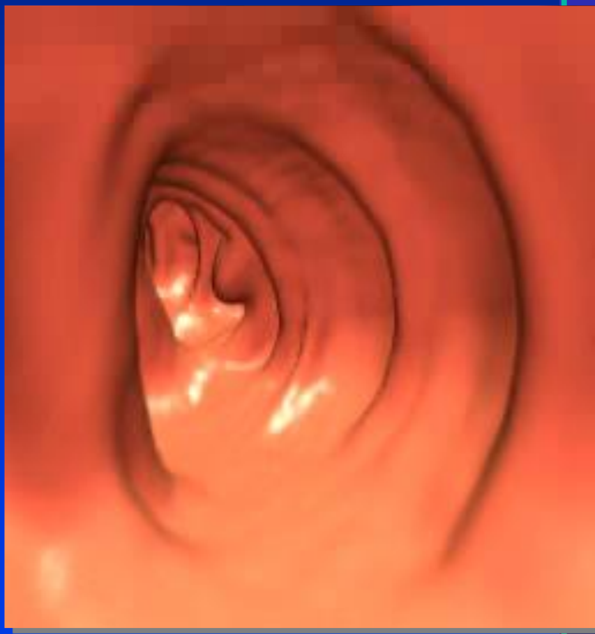


Virtual tour of Ancient Olympia, Courtesy of SGI

Virtual Reality

- Virtual colonoscopy

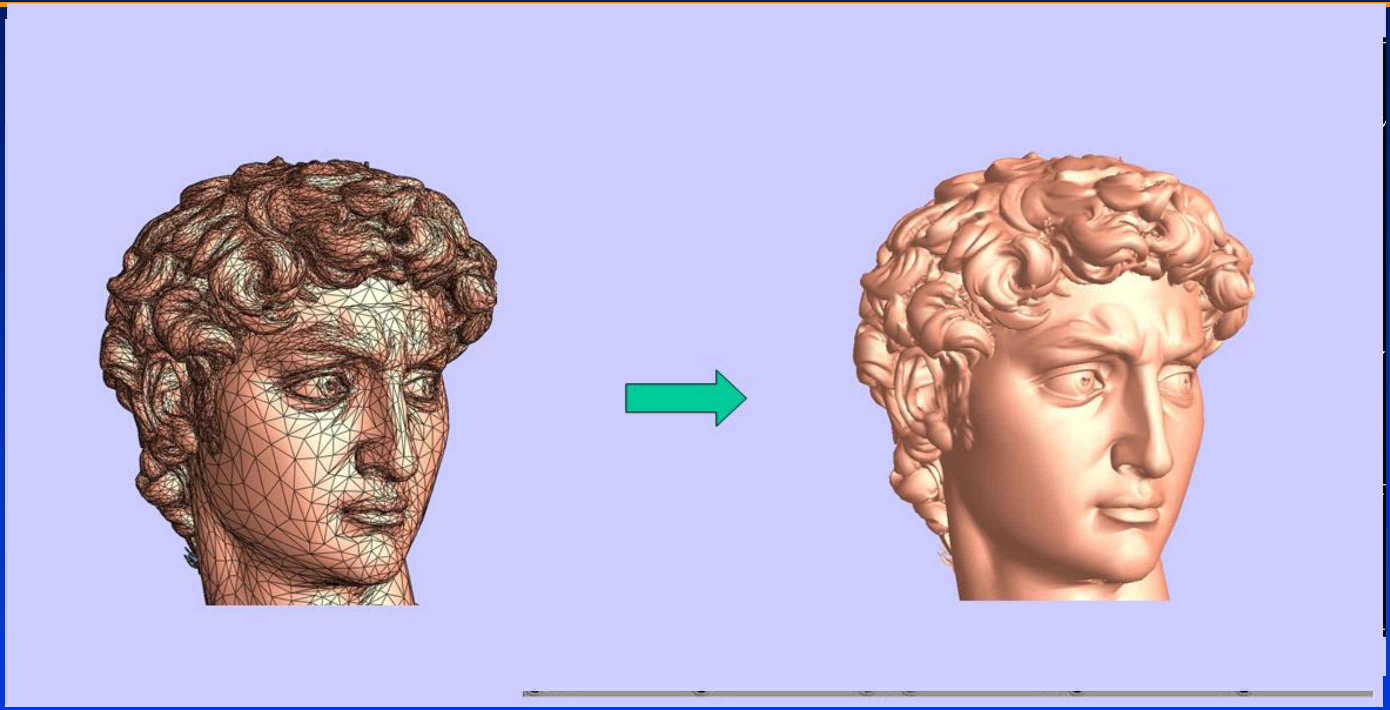


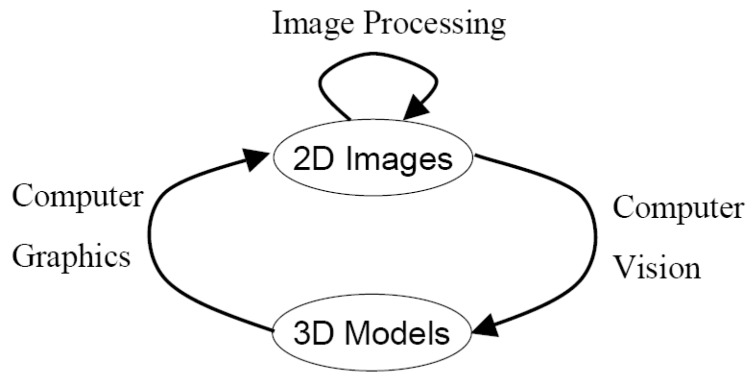


Courtesy of Kaufman et al.

STONY BROOK
STATE UNIVERSITY OF NEW YORK

Image Processing, Analysis, and Synthesis





Computer Art

- Escher Drawing
  - Combine interlocking shapes with tessellation to convey the beauty in structure and infinity

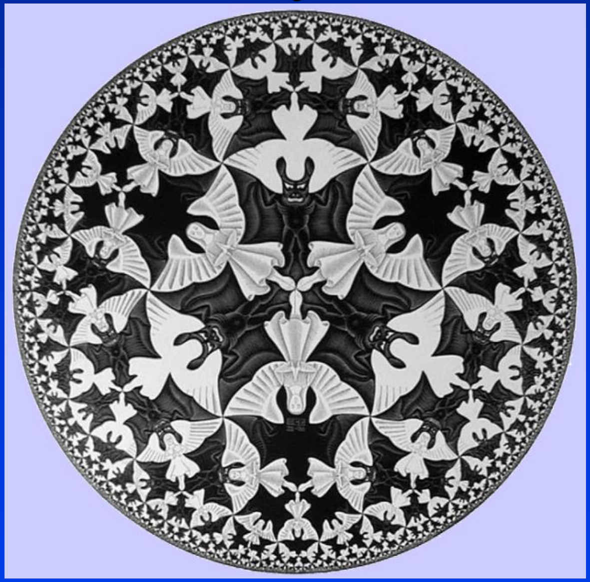
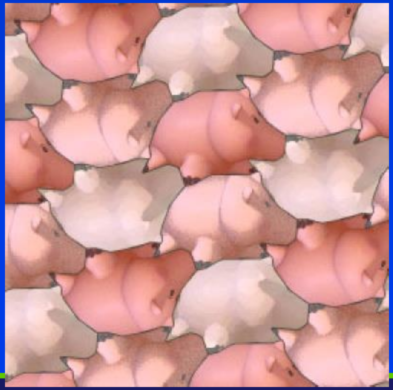


Image courtesy of Escher



Computer Art

- Fine arts, commercial art
- Artistic tools for digital art:
  - Mathematical software (Matlab, Mathematica)
  - CAD software
  - Sculpting, painting, calligraphy systems
- Graphical user interfaces
- Special input devices (pressure-sensitive stylus, graphical tablet, etc.)

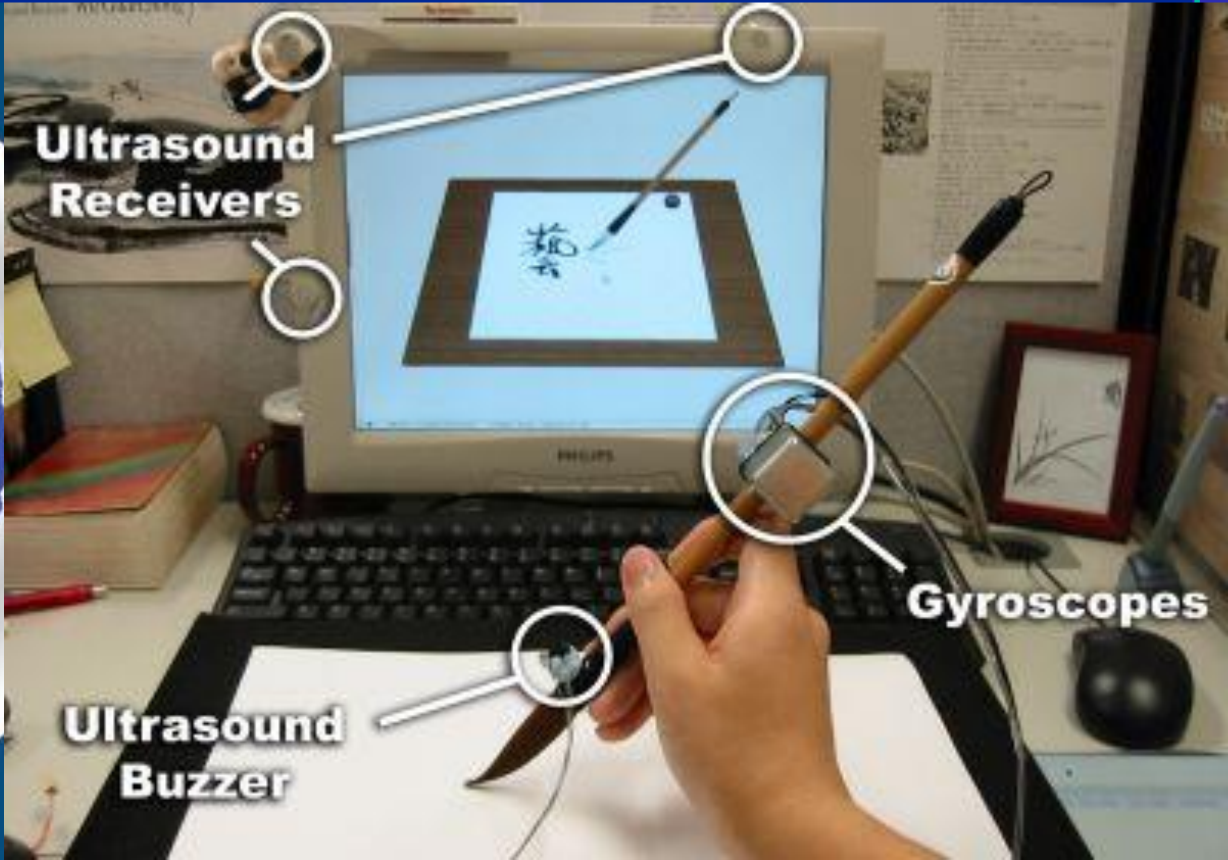
Baxter and Scheib demonstate their haptic art kit, at UNC

Computer Art

- Digital Sculpting

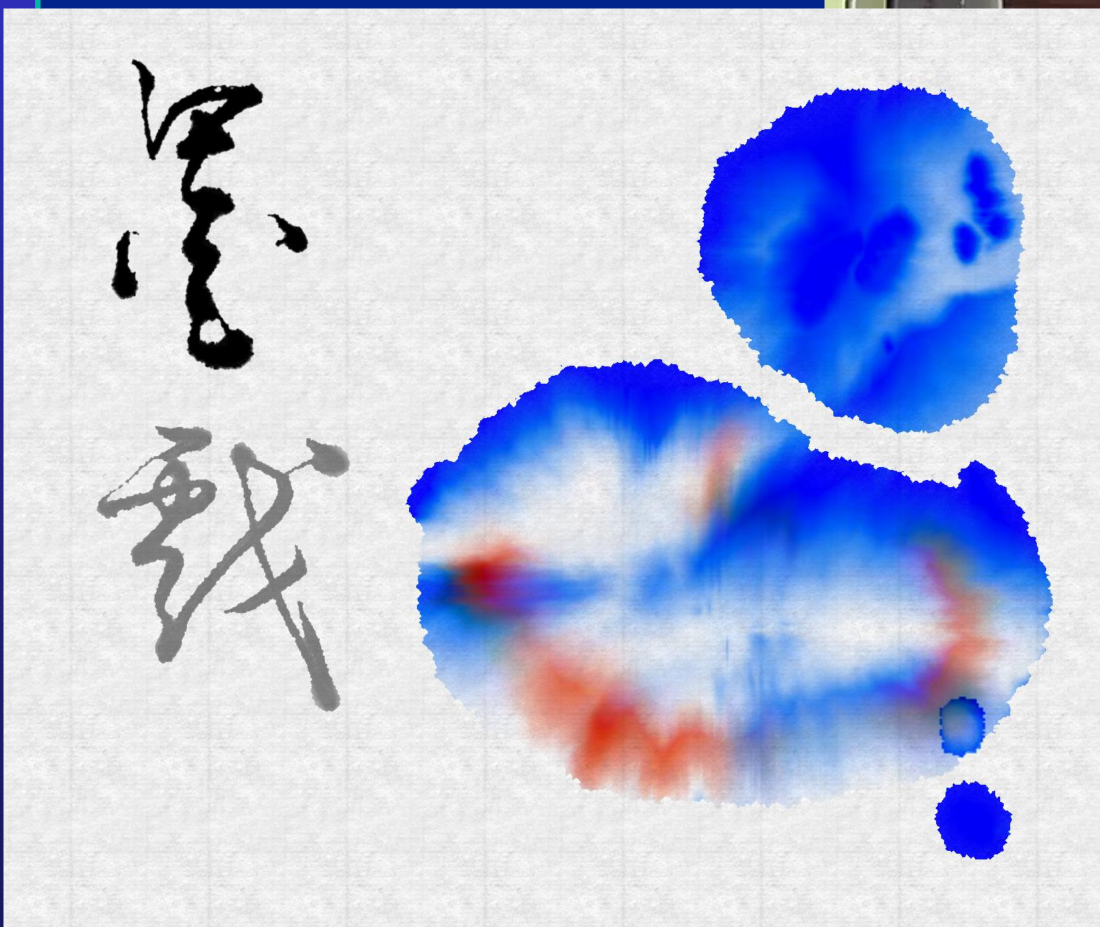
Computer Art

- Digital Painting



Computer Art

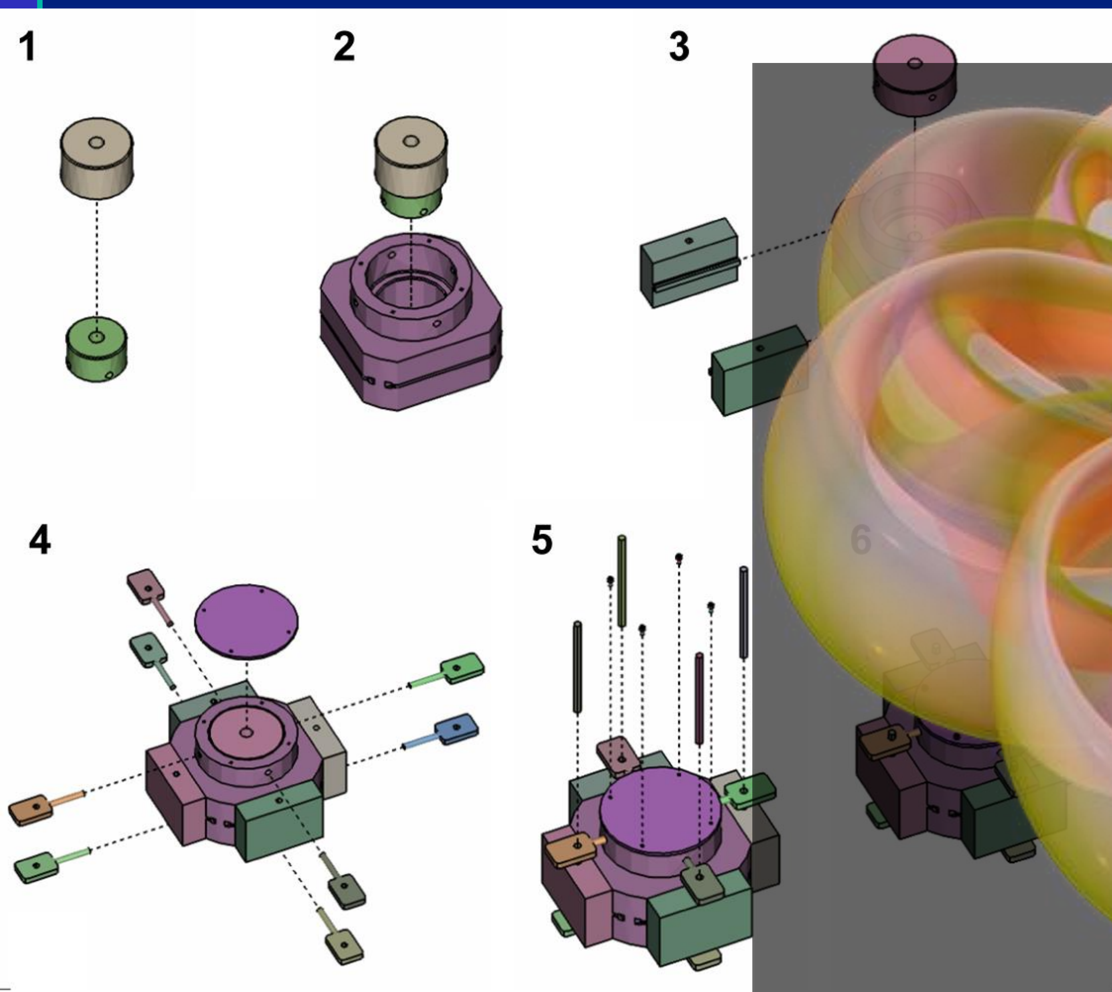
- Digital Calligraphy

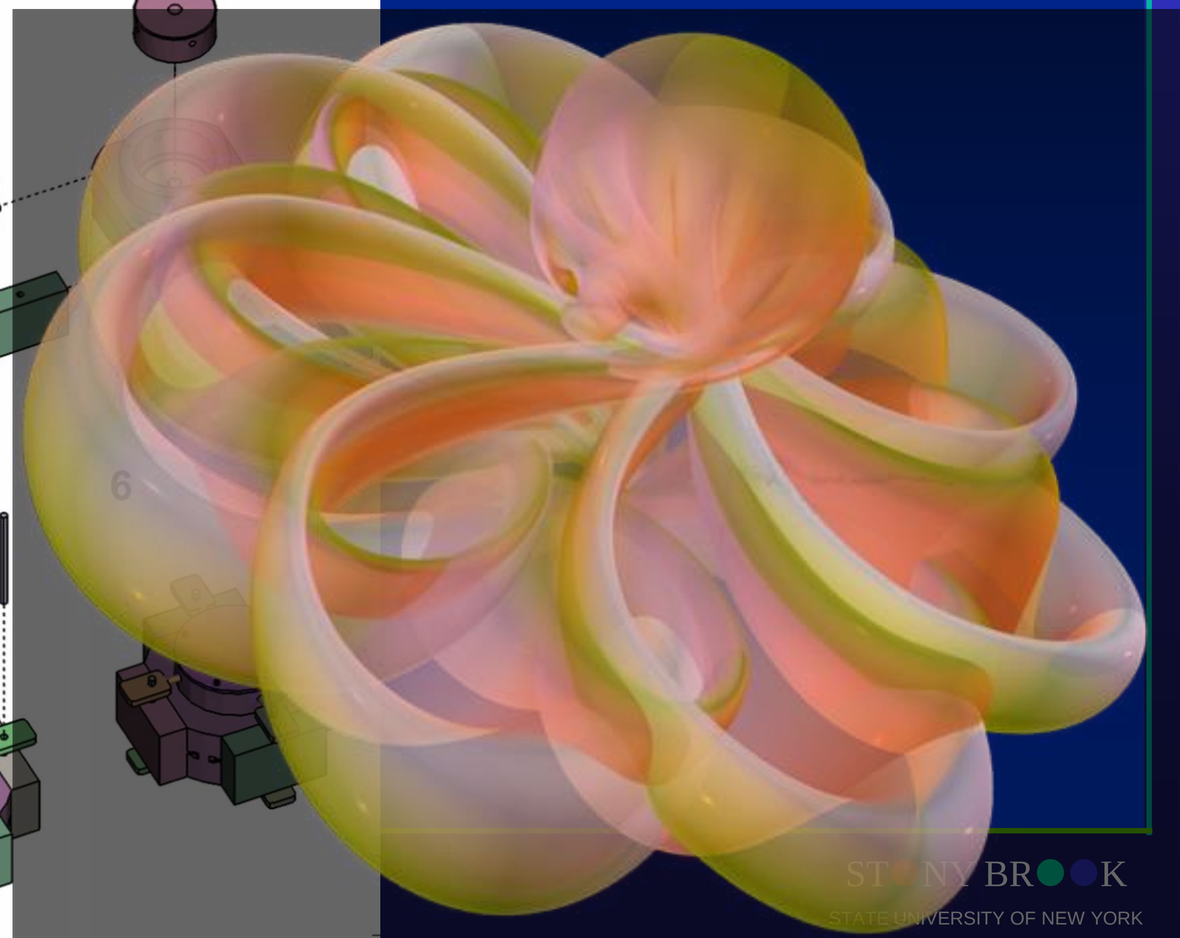


Courtesy of Tai et al.

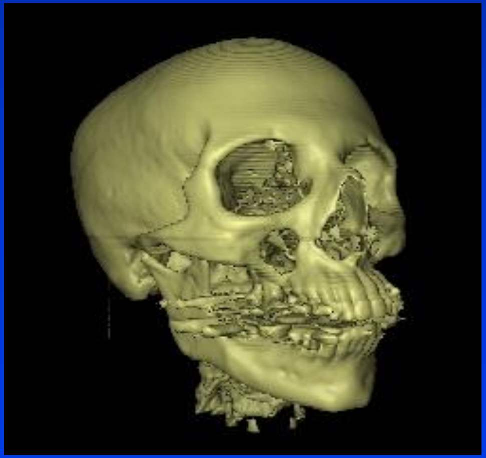
Graphics Applications

- Training and education





Graphics Examples

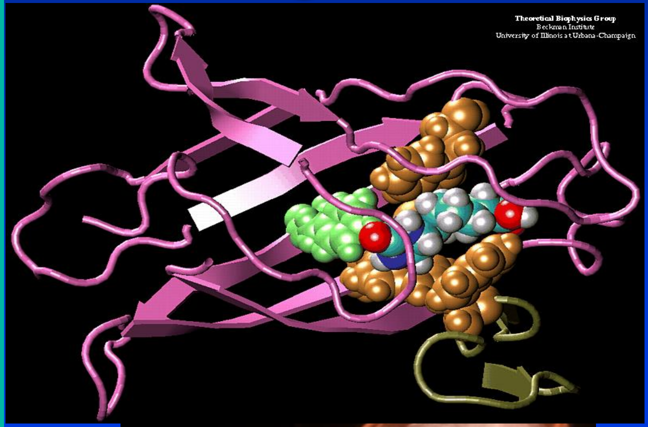


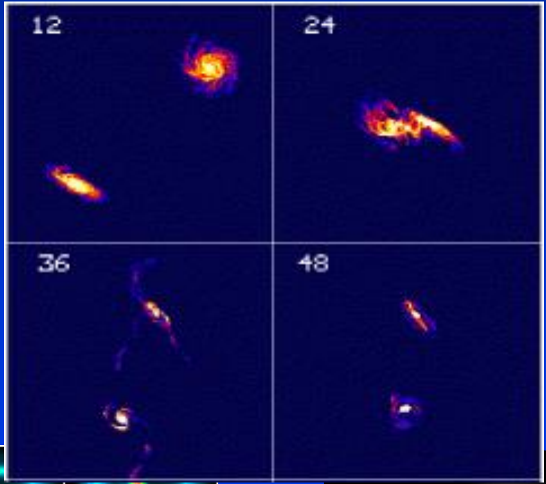
Course Facts

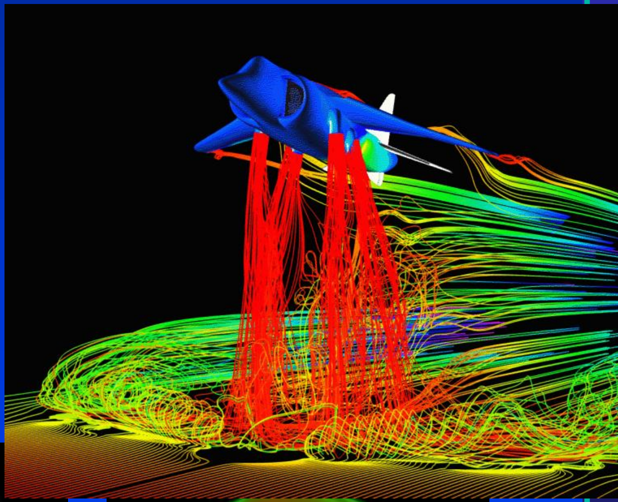
- This is an entry-level graduate course for both MS and PhD students (a quals course for PhD students)!!!
- Can I take this course? YES, if YOU
  - are a graduate student with CS background, have skills in calculus and linear algebra, or talk to the instructor
- You do NOT need to take CSE328 prior to this course
- However, if you had taken CSE328, or CSE332, or equivalent courses elsewhere, it would definitely help!
- One required textbook, several suggested references
- Lecture notes are important!!! Class attendance is critical!!!

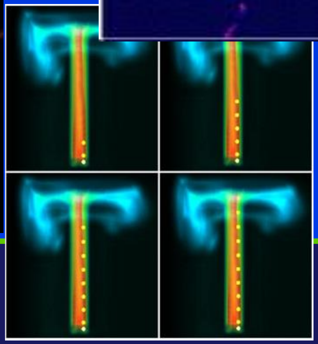
Why Visualization

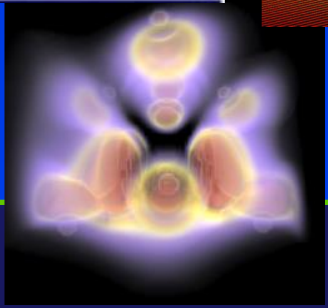
Visualization is a method of extracting meaningful information from complex or voluminous datasets through the use of interactive graphics and imaging

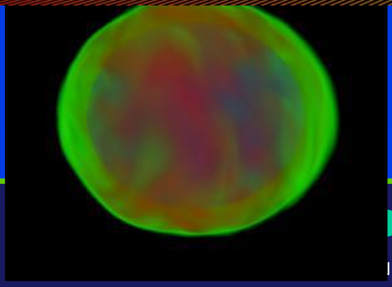


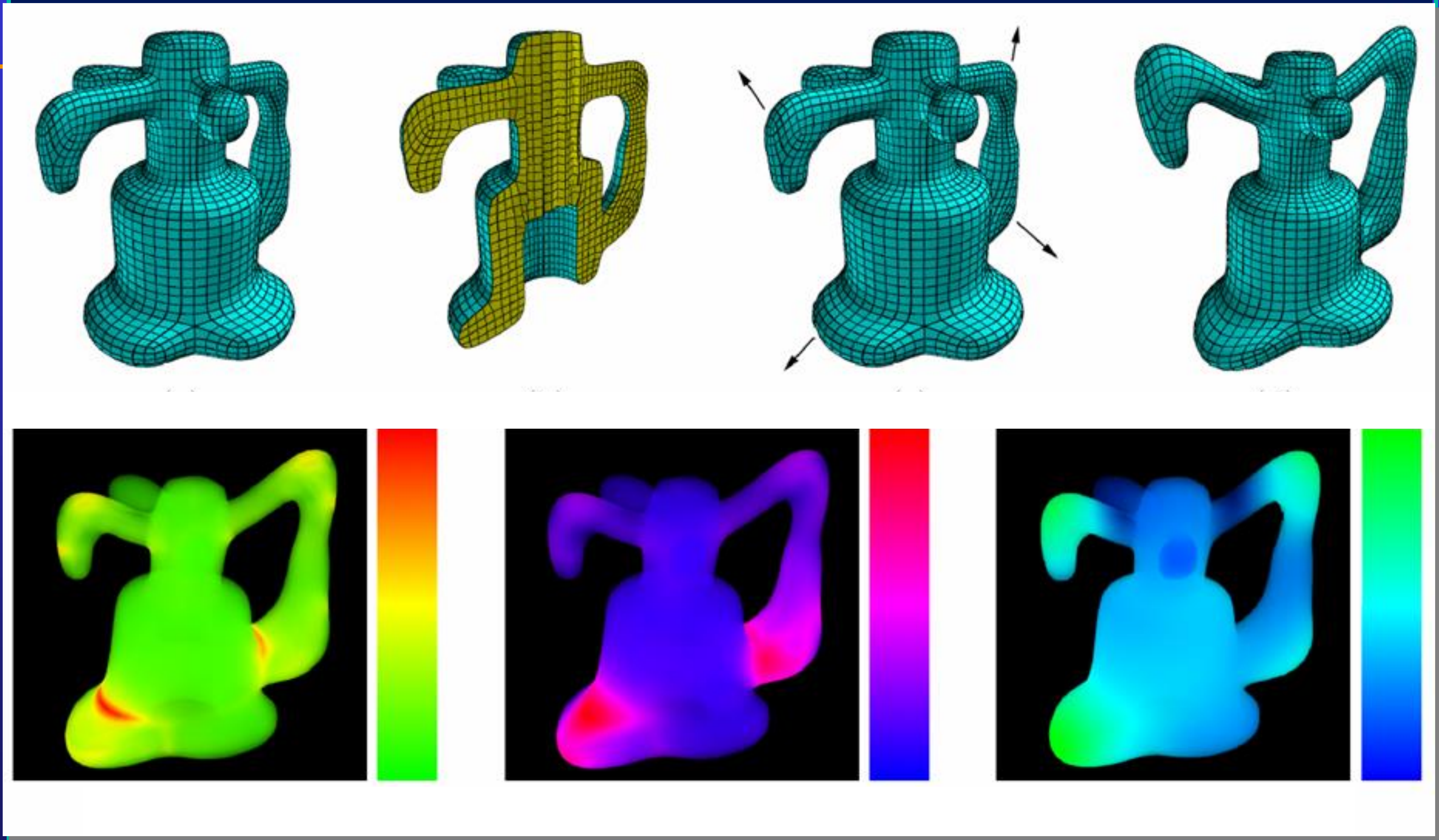












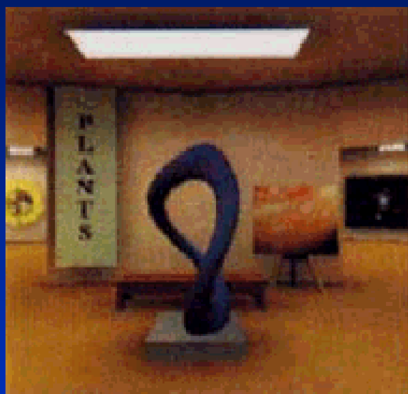
Why Visualization

- Enable scientists (also engineers, physicians, general users) to observe their simulation and computation
- Enable them to describe, explore, and summarize their datasets (models) and gain insights
- Offer a method of SEEING the UNSEEN
- Reason about quantitative information
- Enrich the discovery process and facilitate new inventions

Why Visualization

- Analyze and communicate information
- Revolutionize the way scientists/engineers/physicians conduct research and advance technologies
- About 50% of the brain neurons are associated with vision
- The gigabit bandwidth of human eye/visual system permits much faster perception of visual information and identify their spatial relationships than any other modes
  - Computerized human face recognition

Graphics Examples: Representation



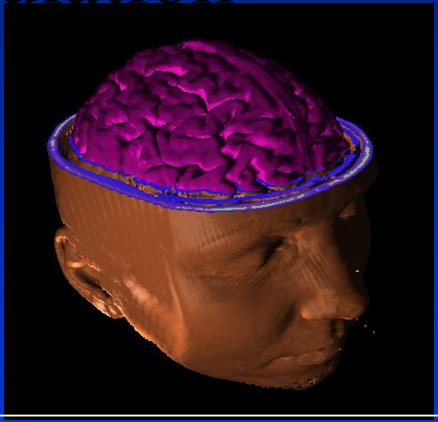
Images

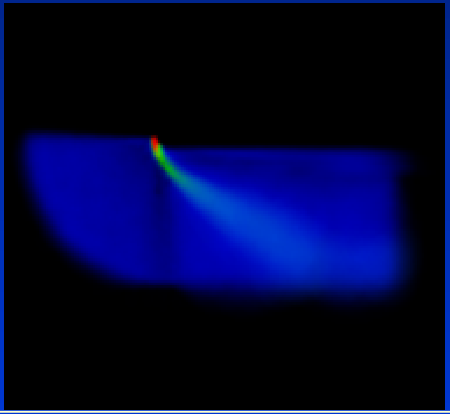
Points

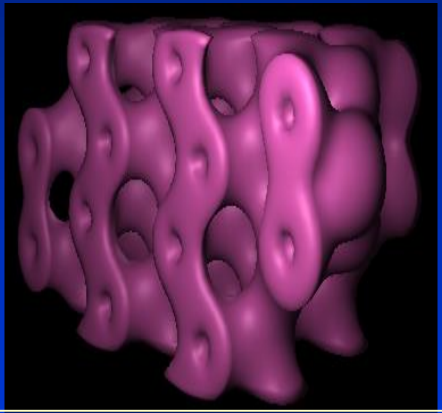
Volumes

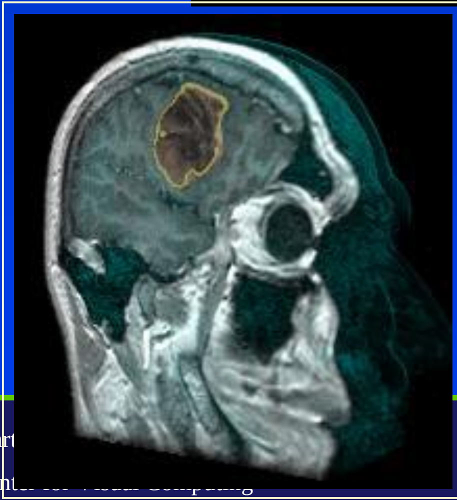
More Examples

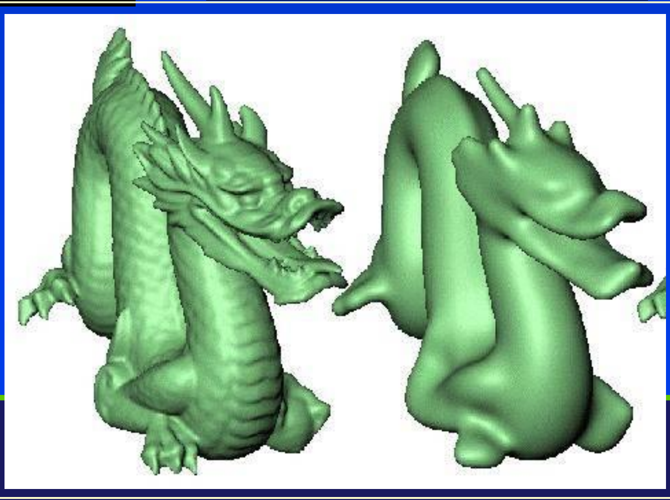
- Medical Imaging, geometry processing, physical simulation

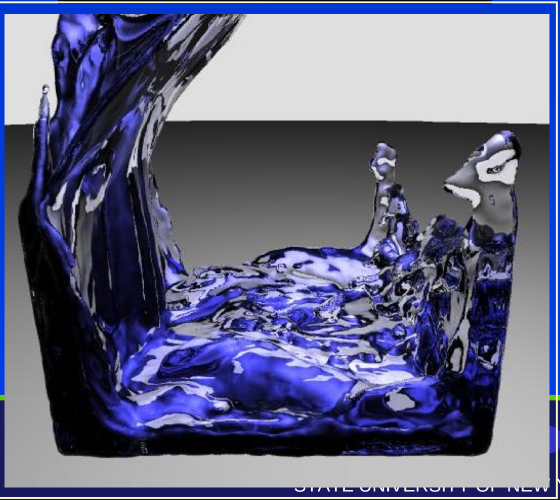










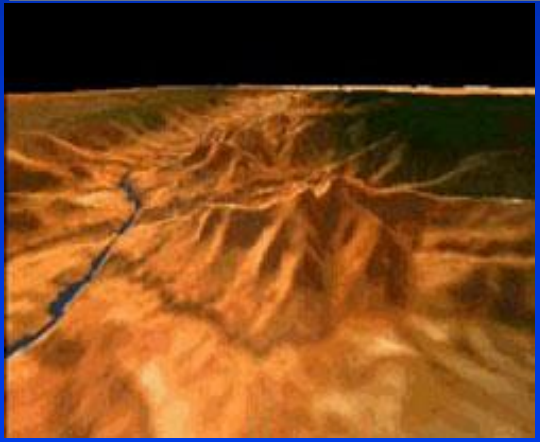


Terrain Modeling and Rendering

Medicine and Health-care

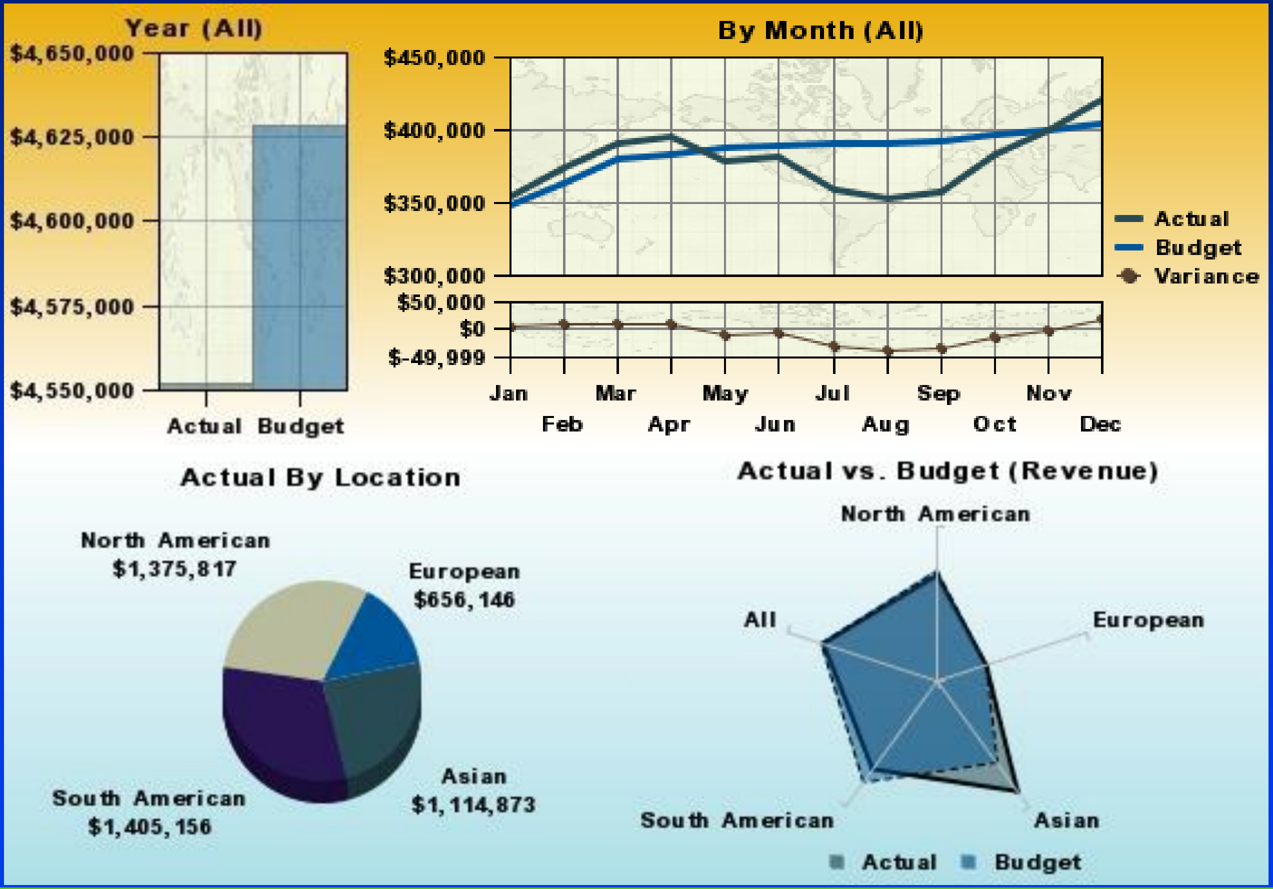
National Security



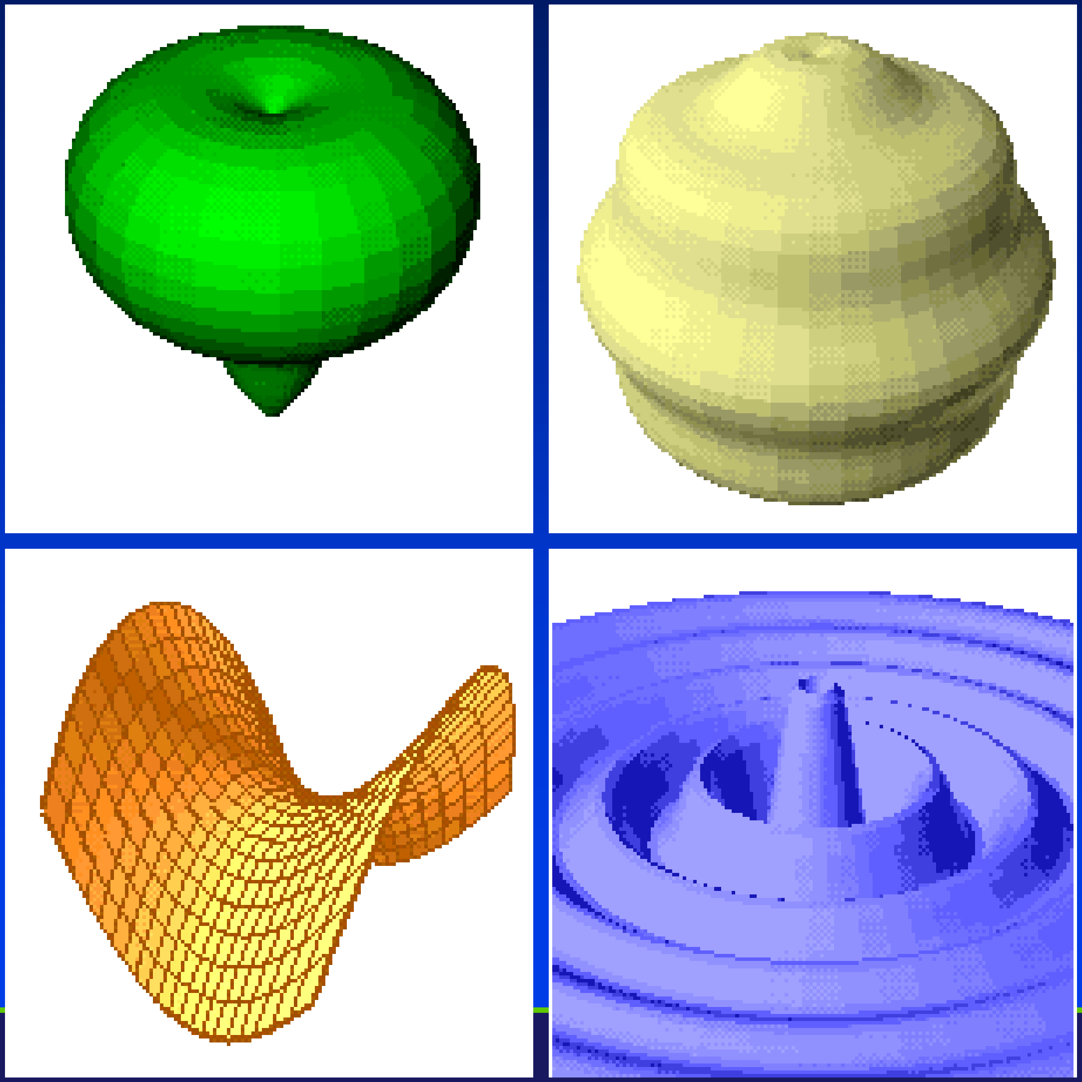



Earlier Days of Computer Graphics

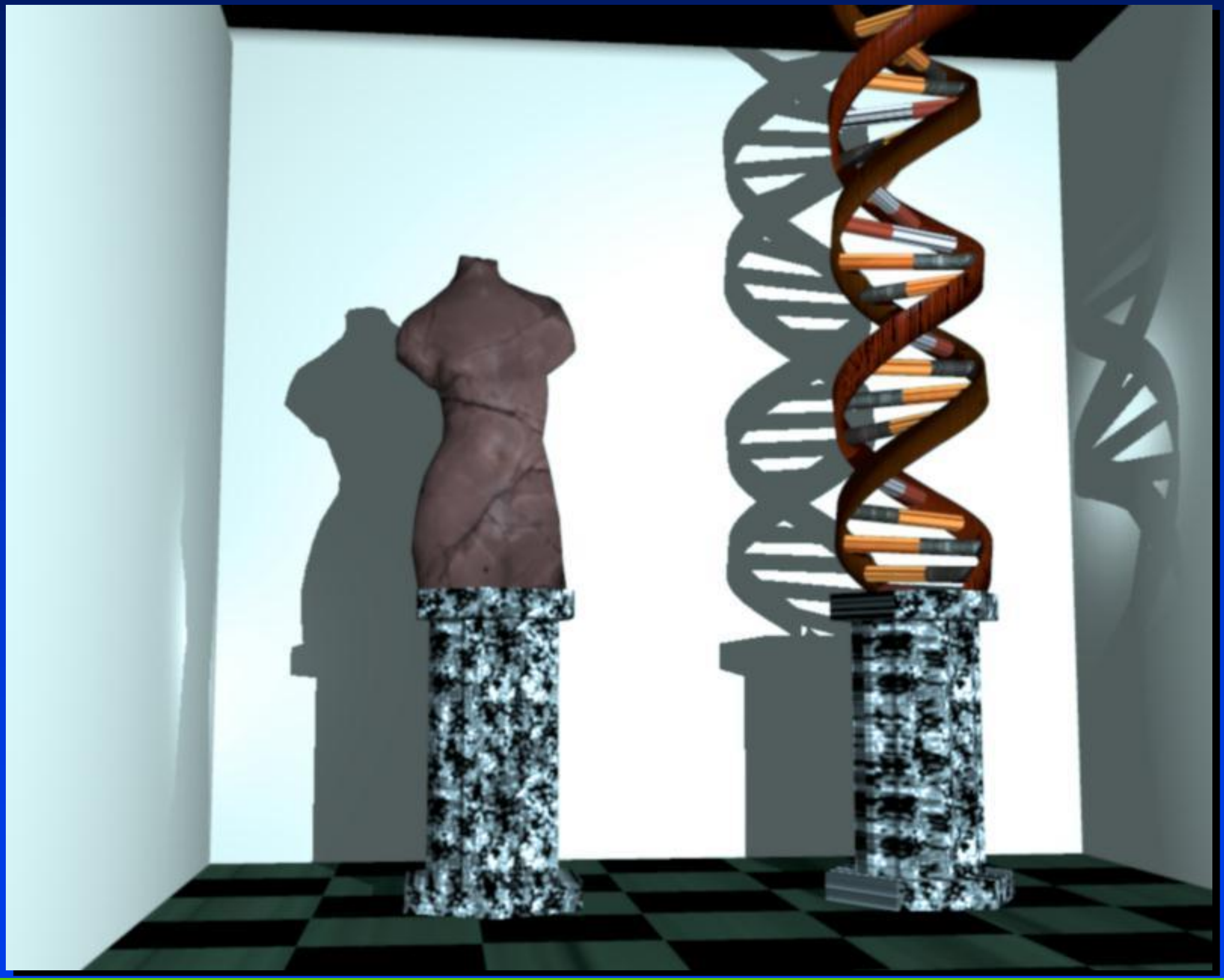
- Visual display of data (graphs and charts)



Mathematical Function Plots

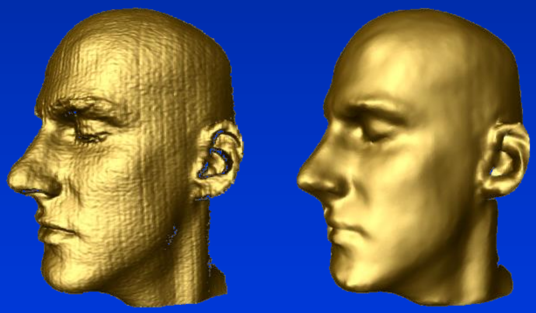


Computer Graphics Components



Geometric Processing

- **A**pplying signal and image processing algorithms to geometry

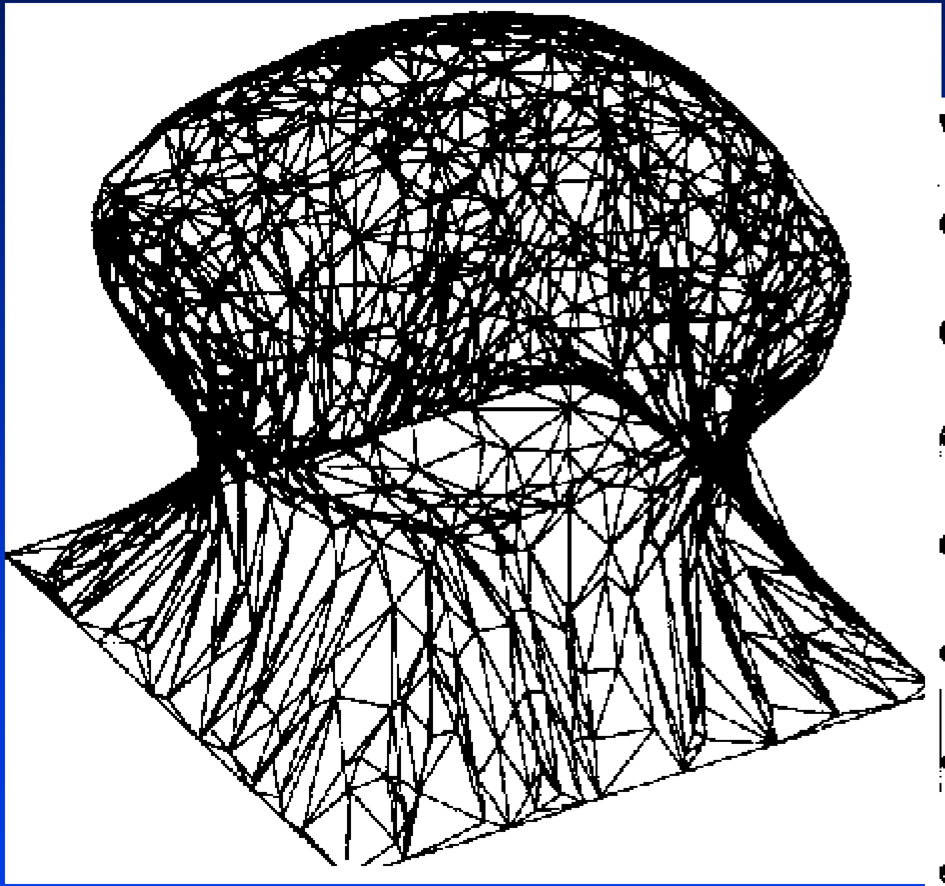


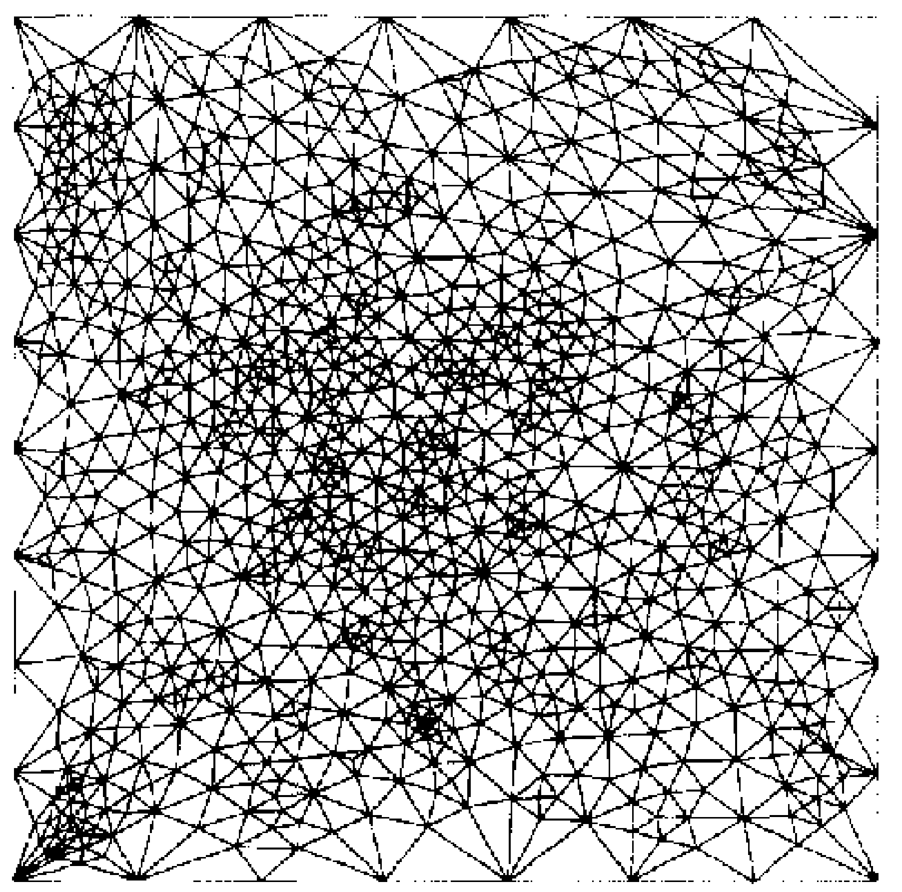
De-noising

Enhancement

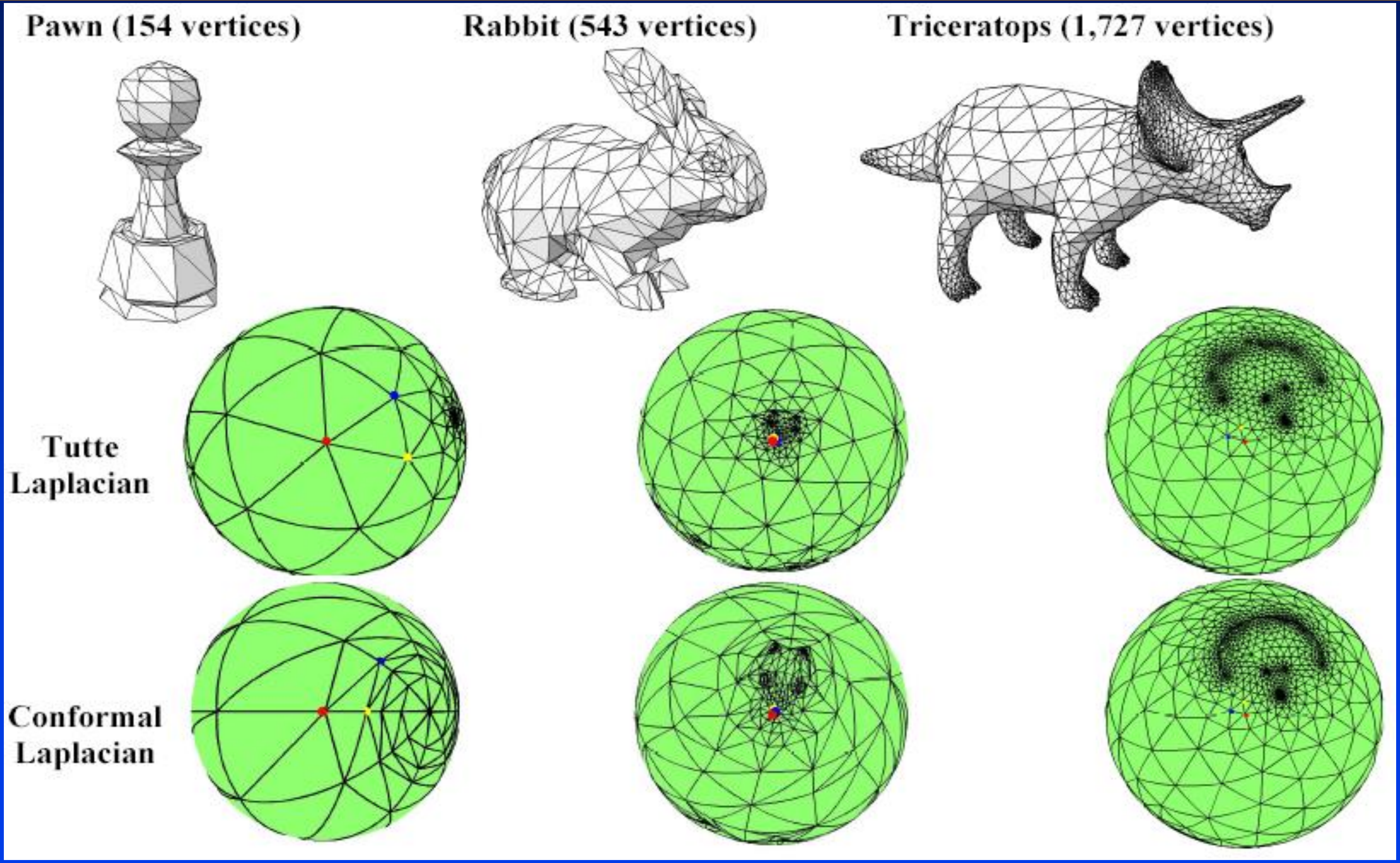
(Possible) Project Topics

Parameterization based on PDEs

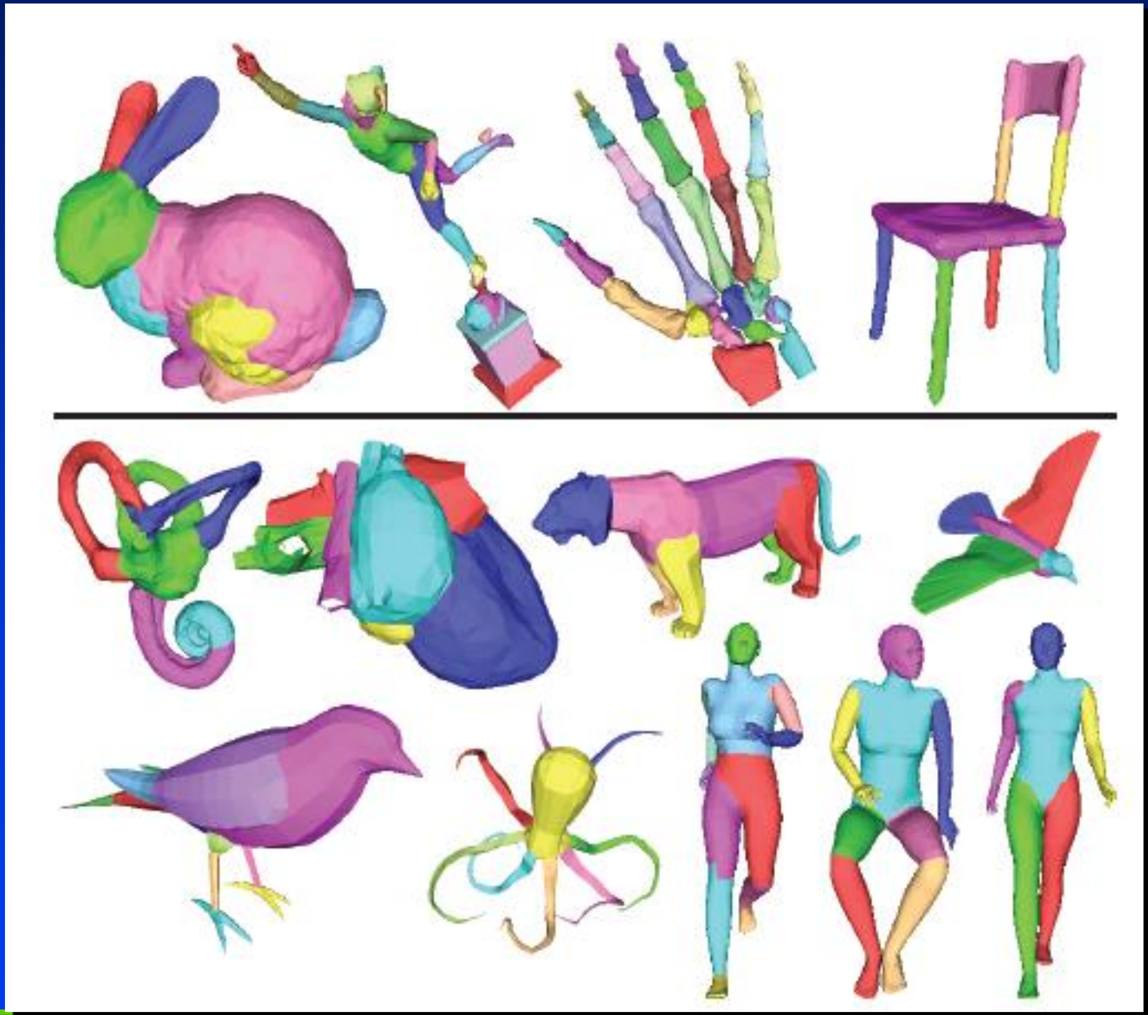




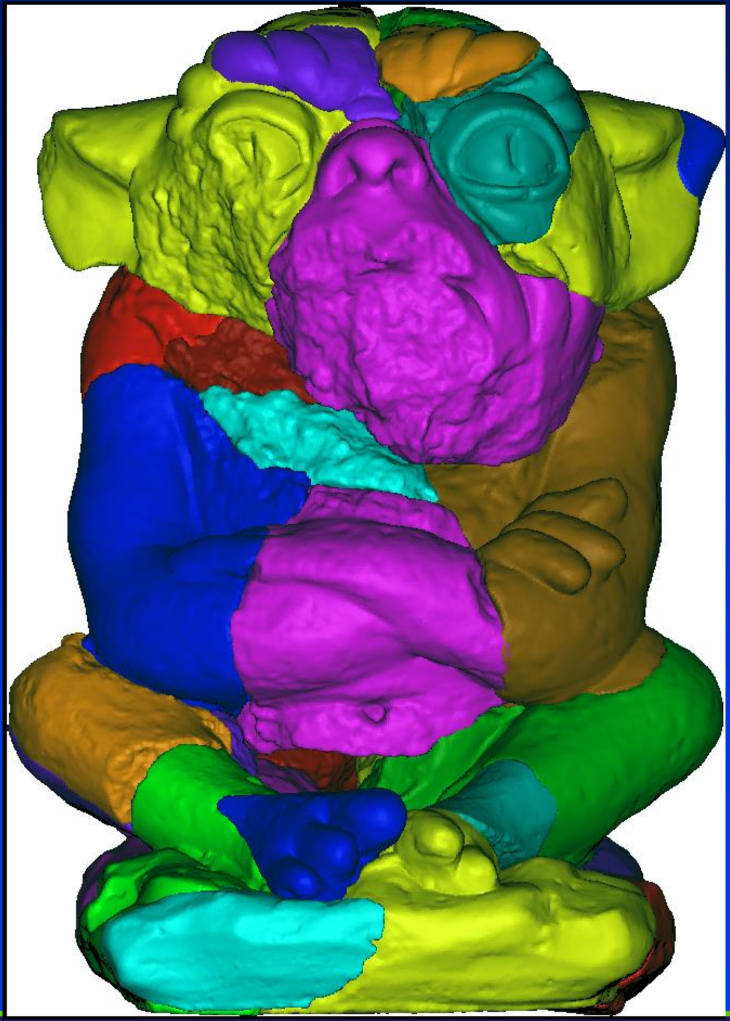
Spherical Parameterization



Model Segmentation



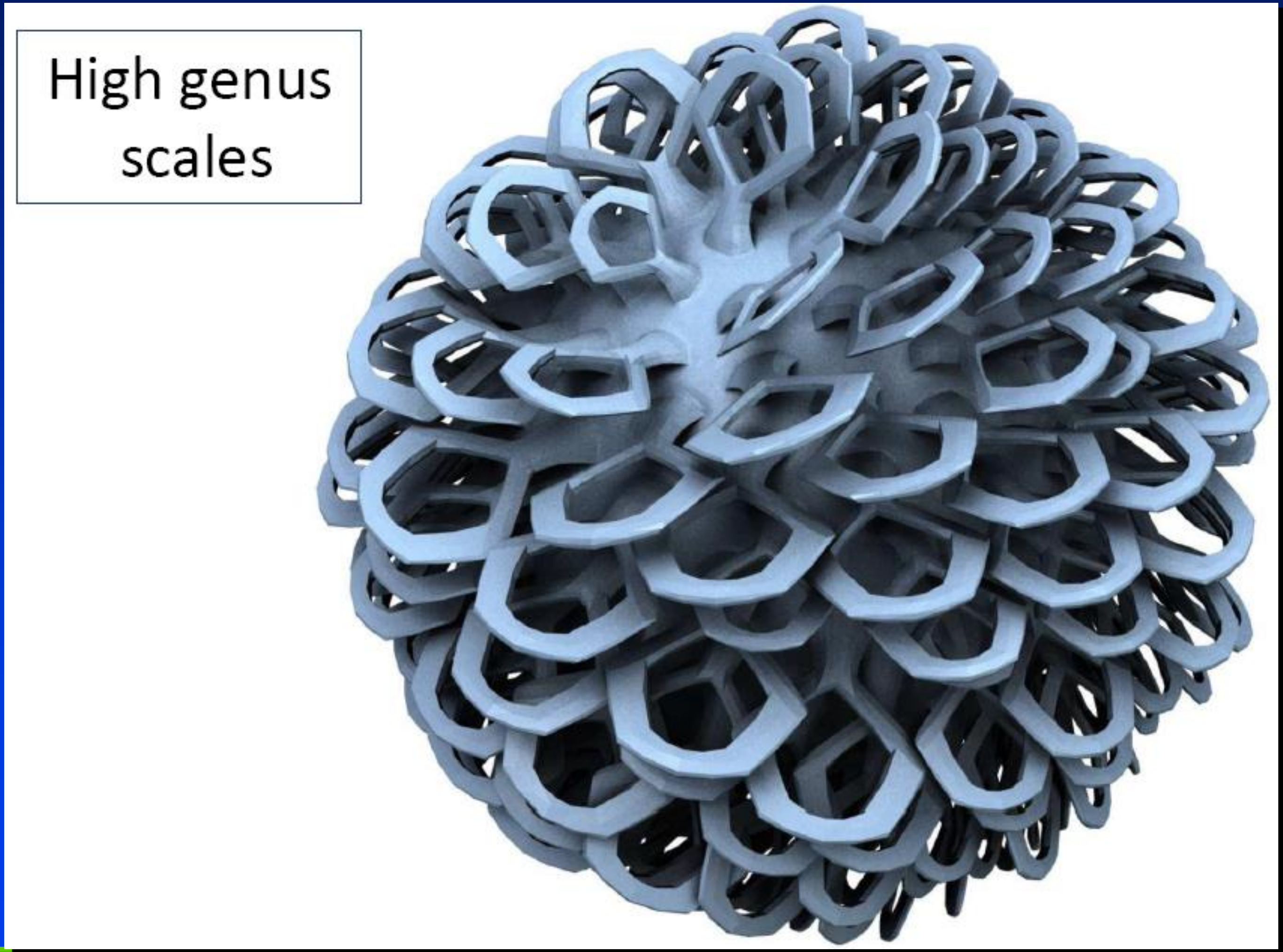
Shape Matching



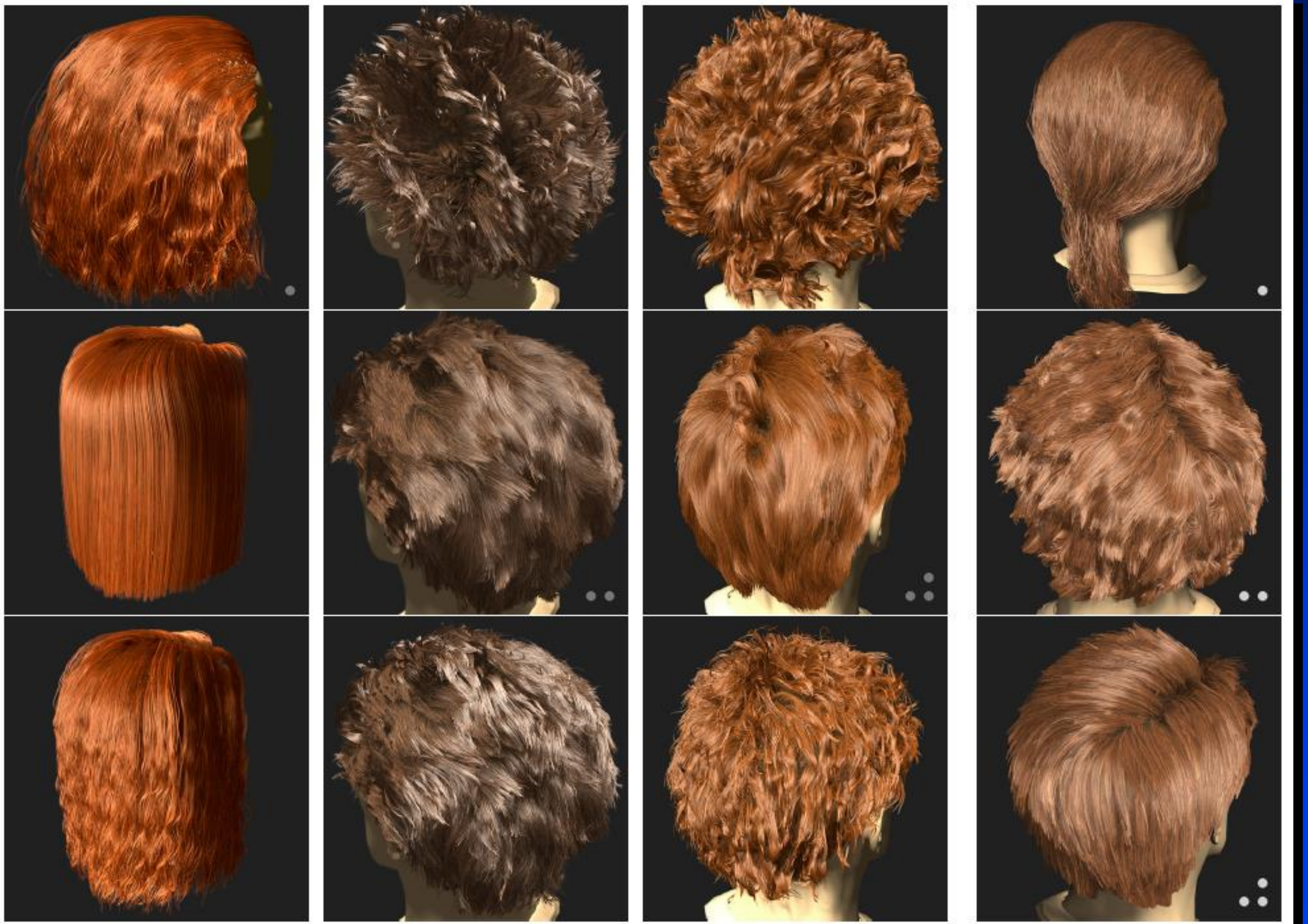
Building Reconstruction

Figure 11: *Additional reconstruction results using SmartBoxes. From left to right: real photograph, LiDAR scan, 3D reconstruction, and its textured version for a visual comparison with the photograph. The examples show reconstruction of complex buildings with some irregularity. Grouping and contextual force during drag-and-drop allow the reconstruction to deal with large-scale missing data (bottom row).*

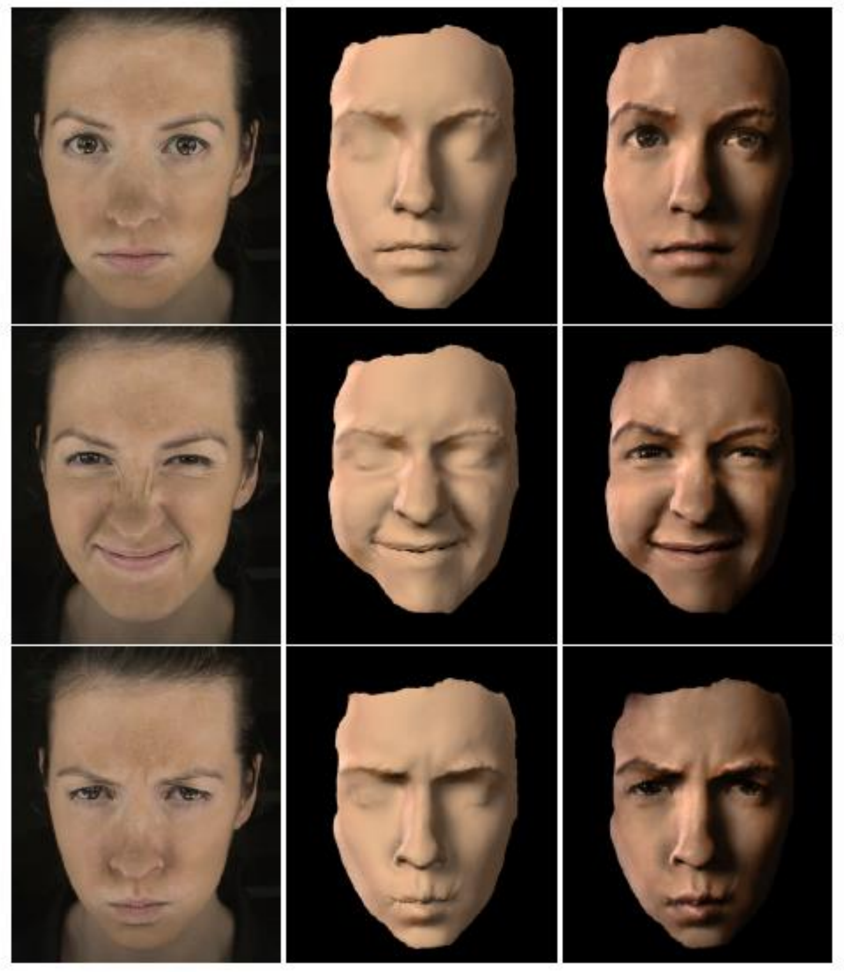
Geometry Texture Synthesis



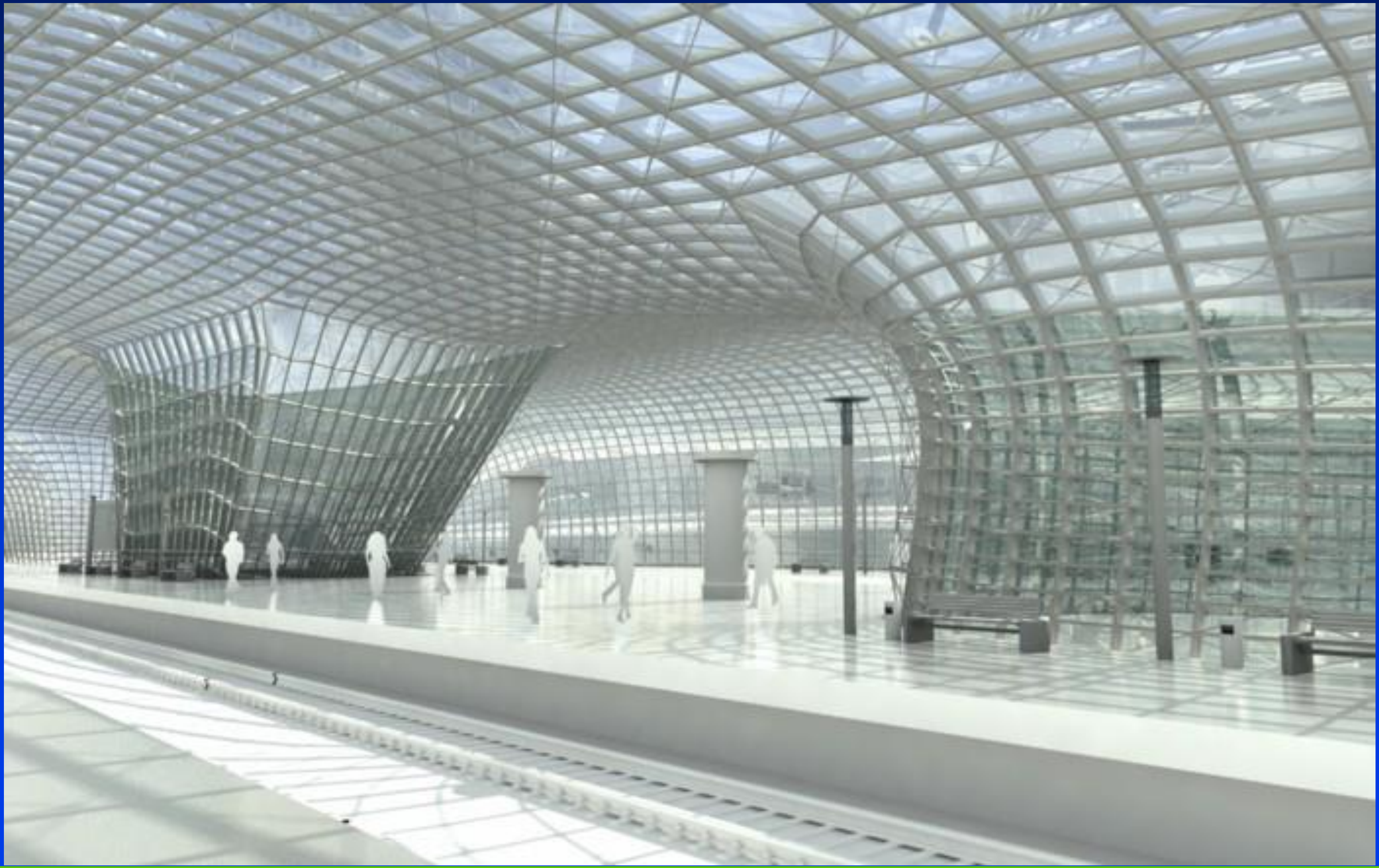
Geometry Synthesis of Human Hair


(a) curly + long straight (b) short spikes + wavy (c) puffy + straight (d) combing

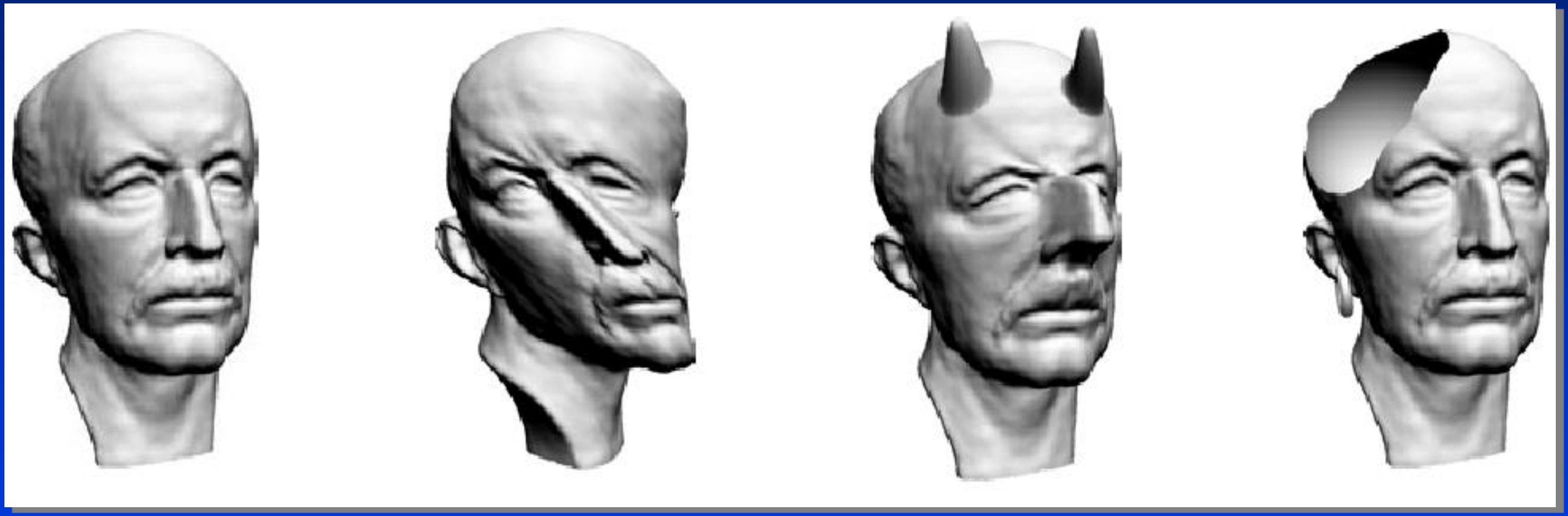
Facial Expression Acquisition and Synthesis



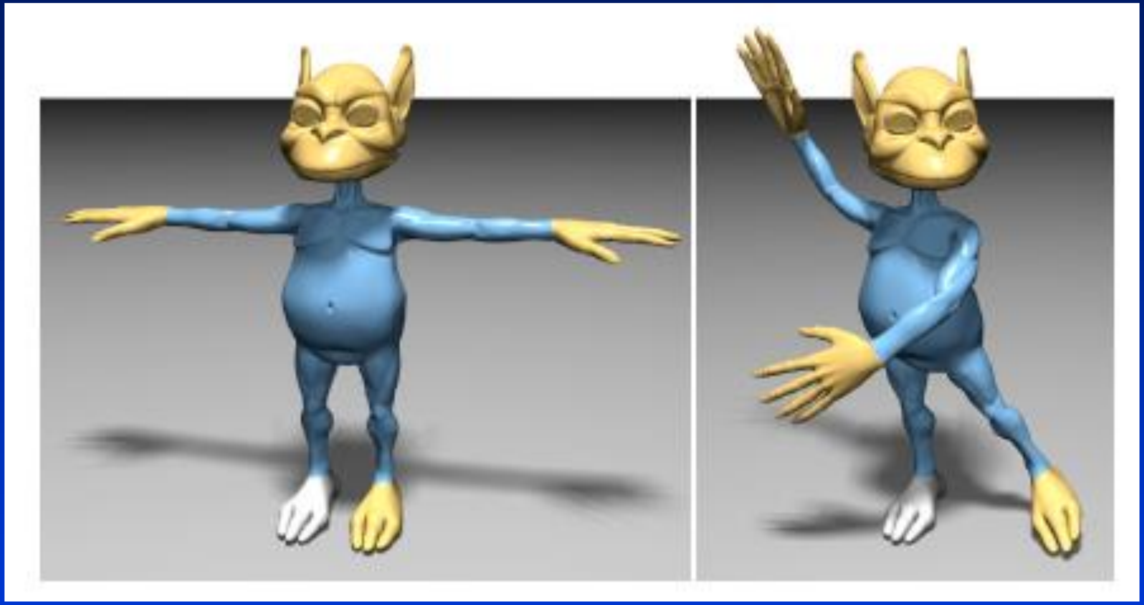

Architectural Geometry

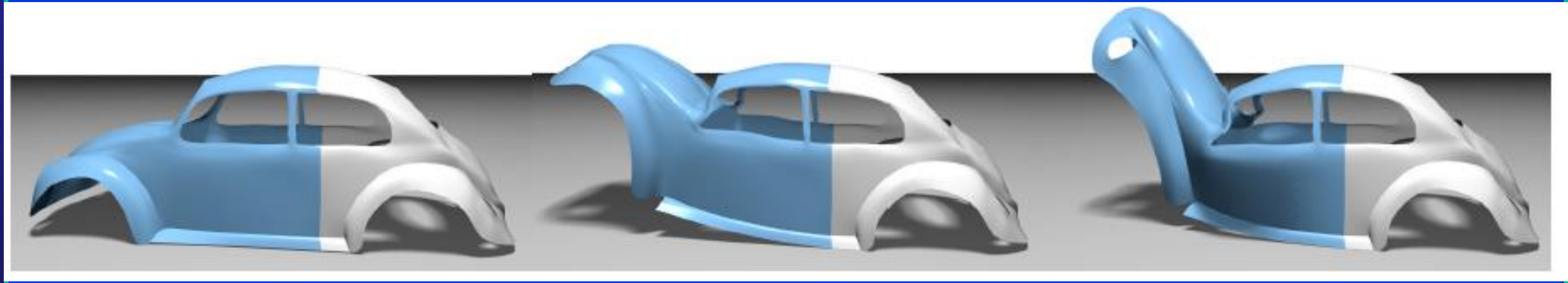


Shape Deformation and Editing

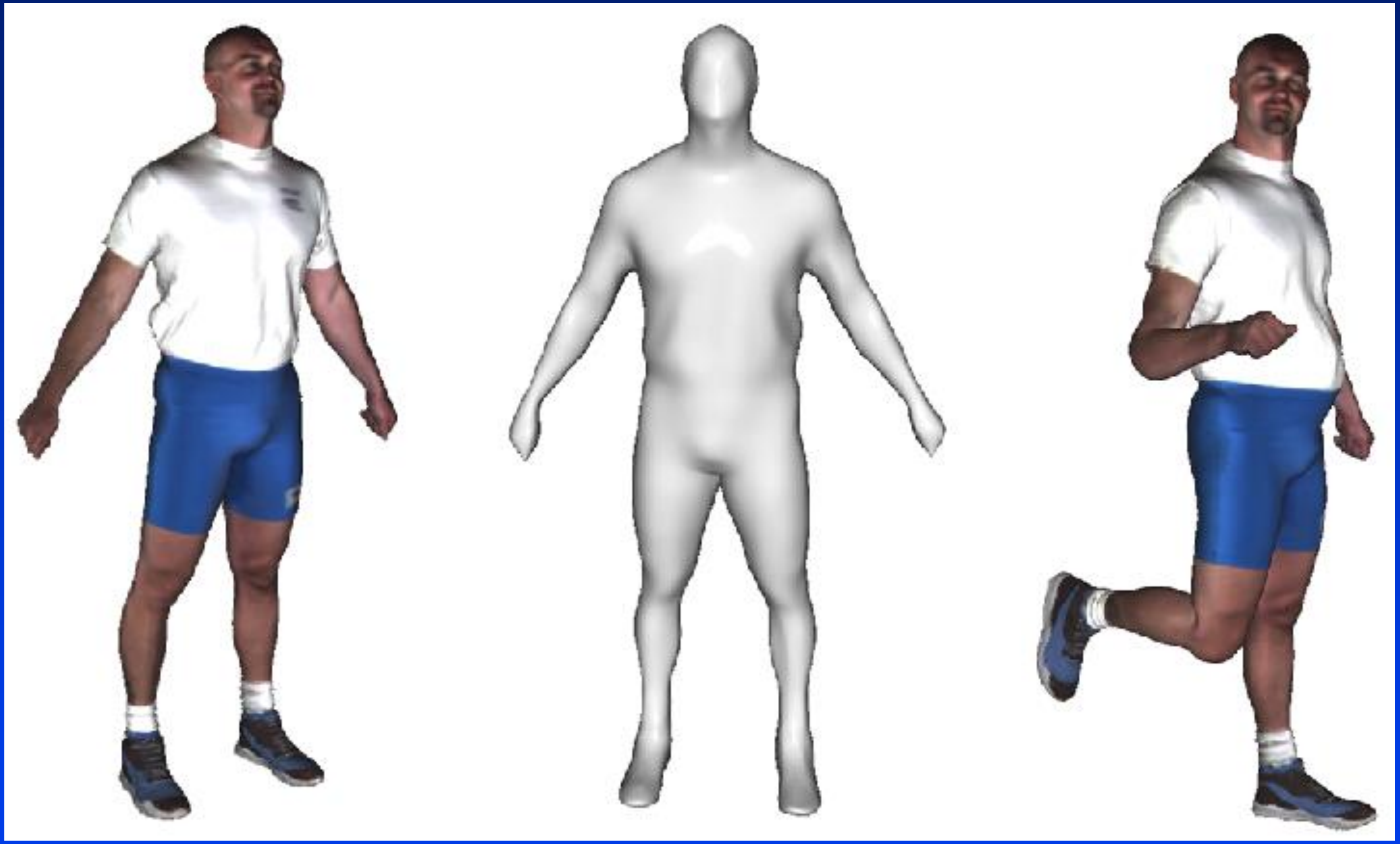


Shape Deformation





Motion Synthesis (Animation)



Tree Simulation

Finite Element Simulation

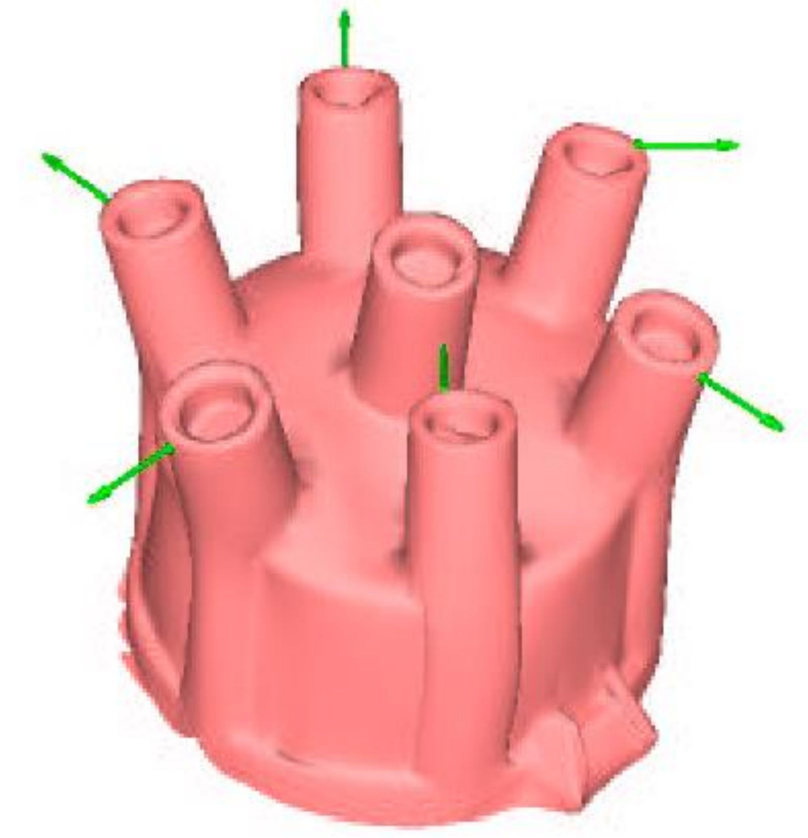
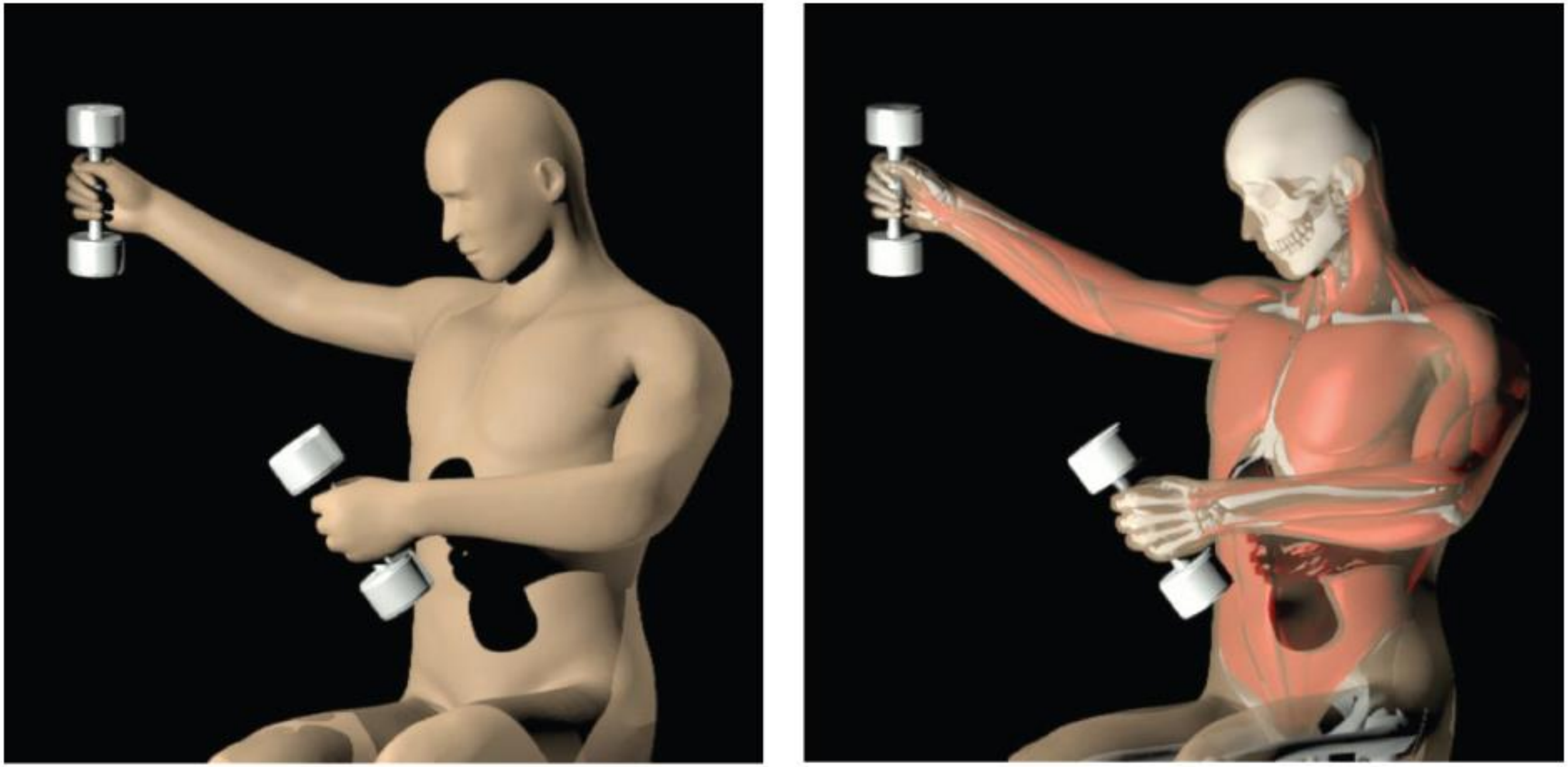


Figure 1: Distributor Cap

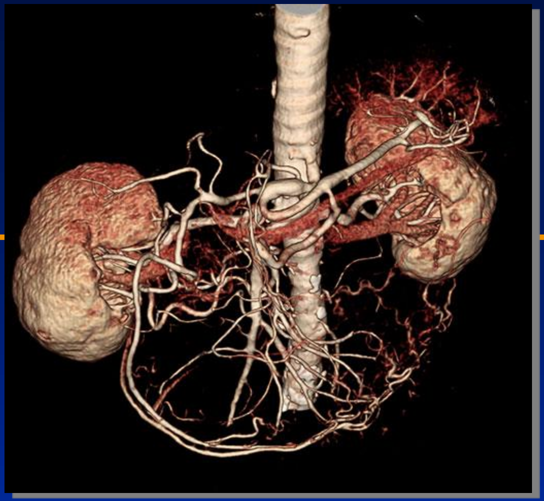
Biomechanical Modeling of Human



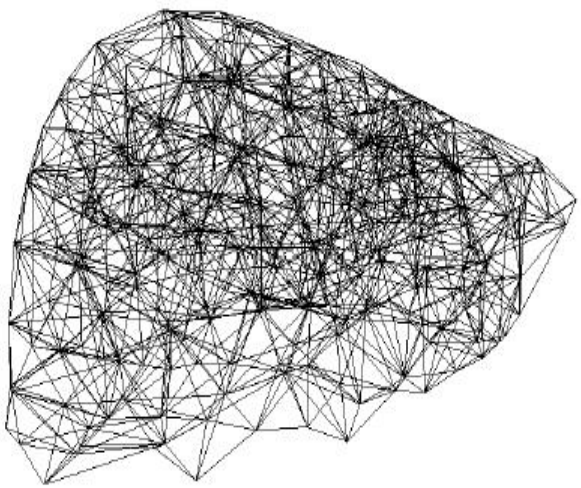
(a) (b)

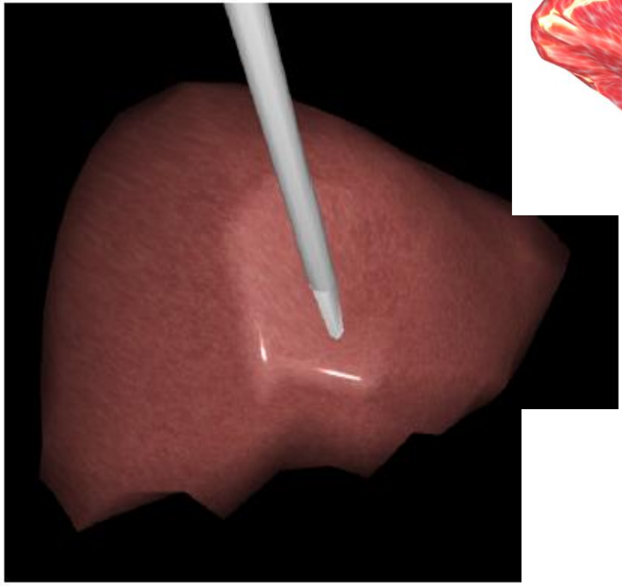
Fig. 13. The soft tissue simulator produces realistic deformations of (a) the visualization geometry, and (b) embedded volumetric muscles.

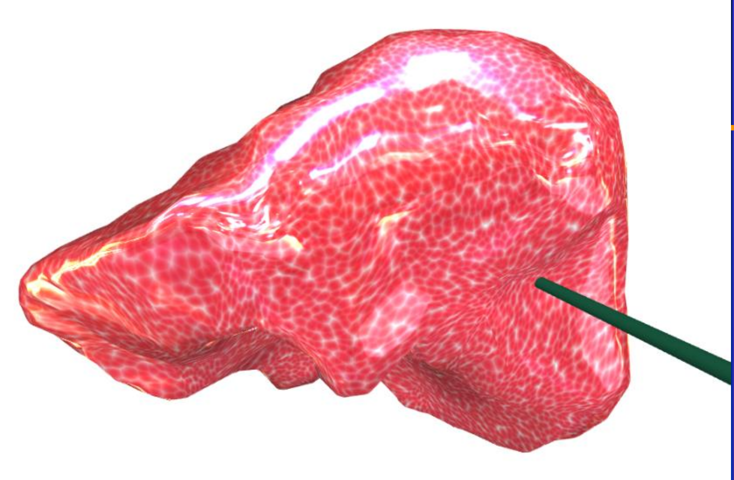
Biomedical Applications

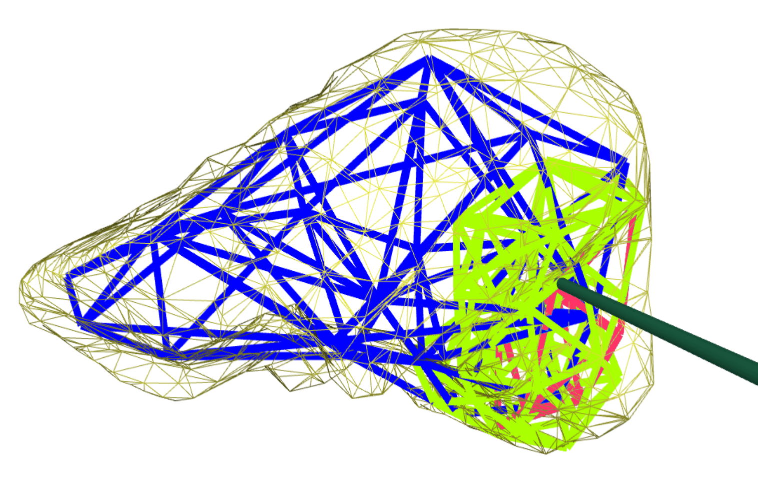


Organ Deformation







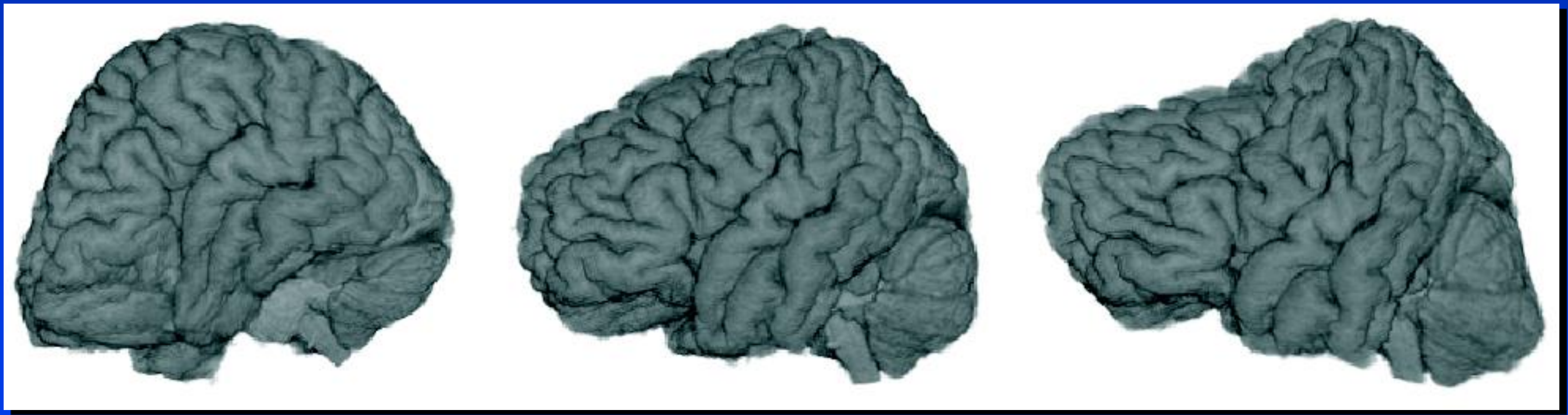


PDE-driven Texture Synthesis

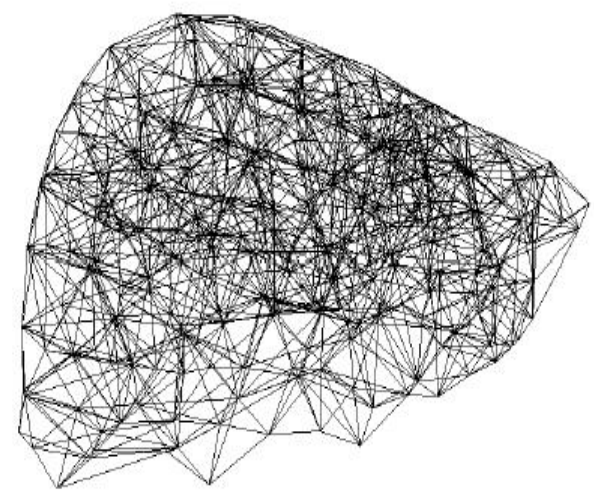
Turk 91

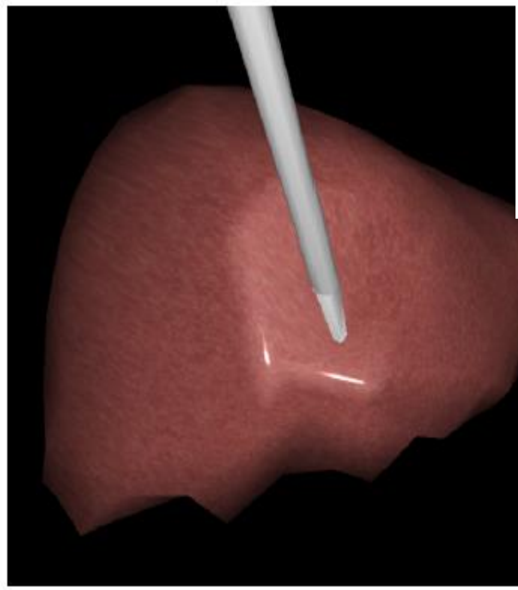
Brain Deformation

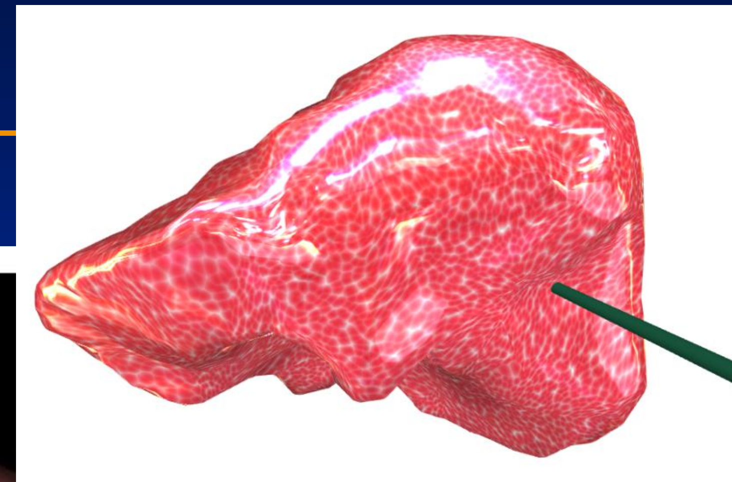
- Medicine
- Simulation
- Modeling
- Entertainment

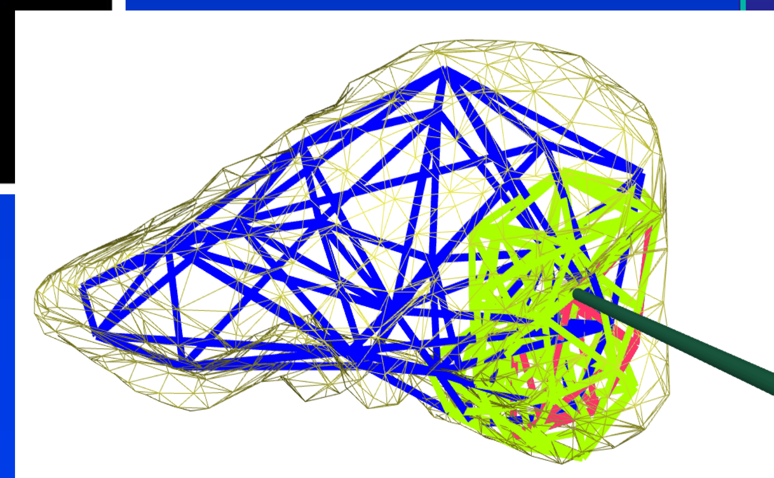


Organ Deformation

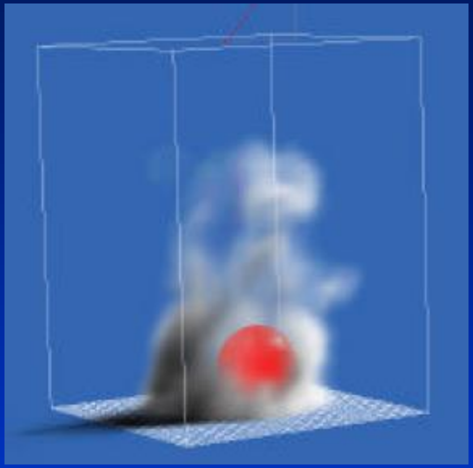


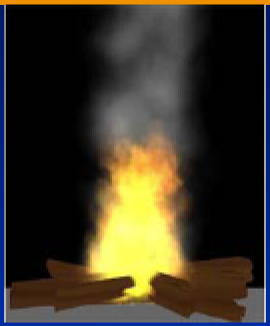


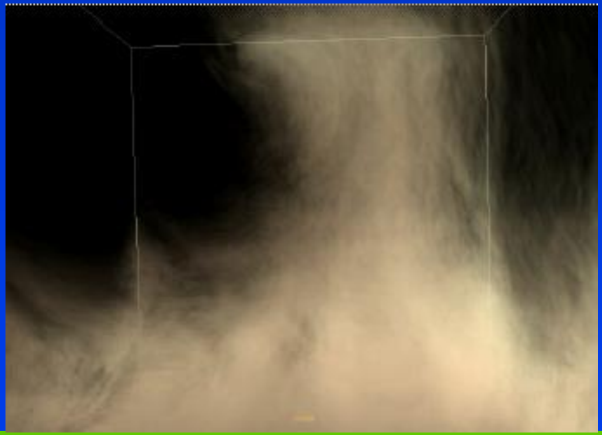


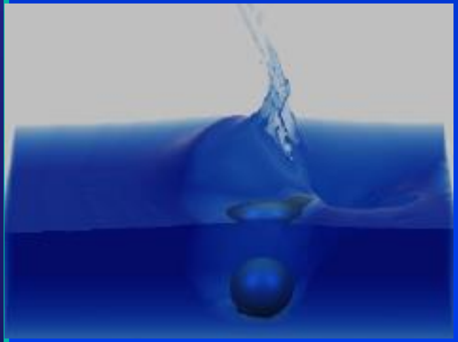


Fluid Simulation



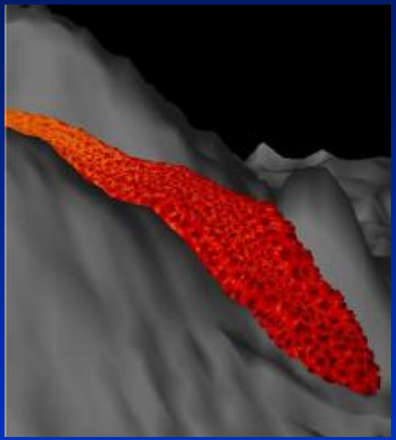




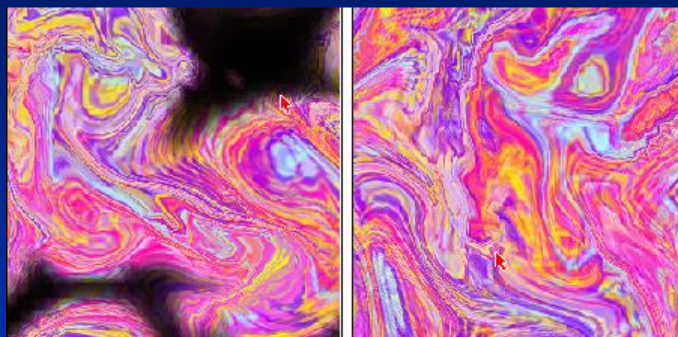
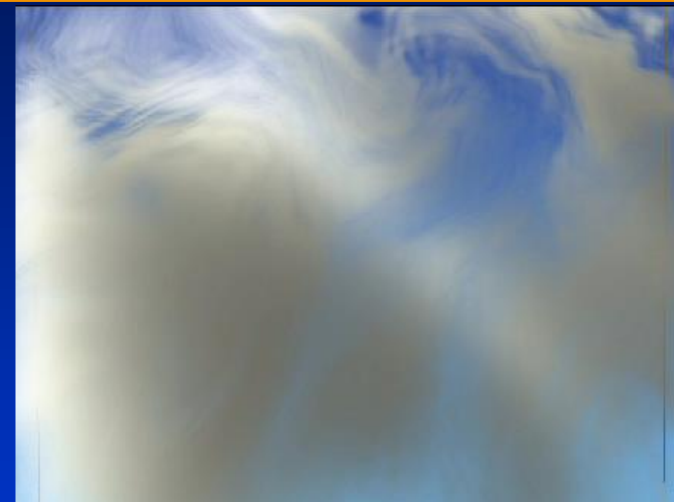


Natural Phenomena

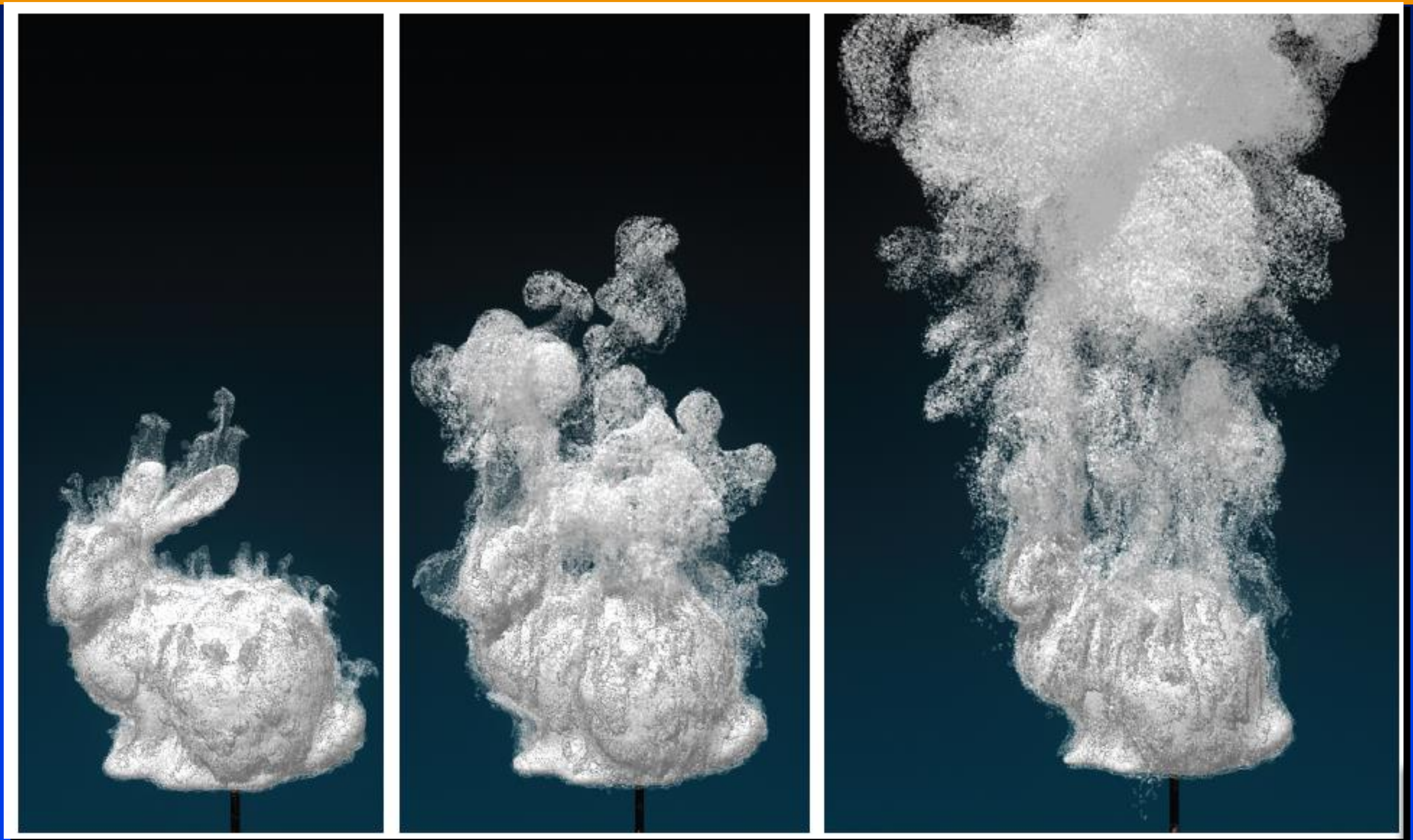




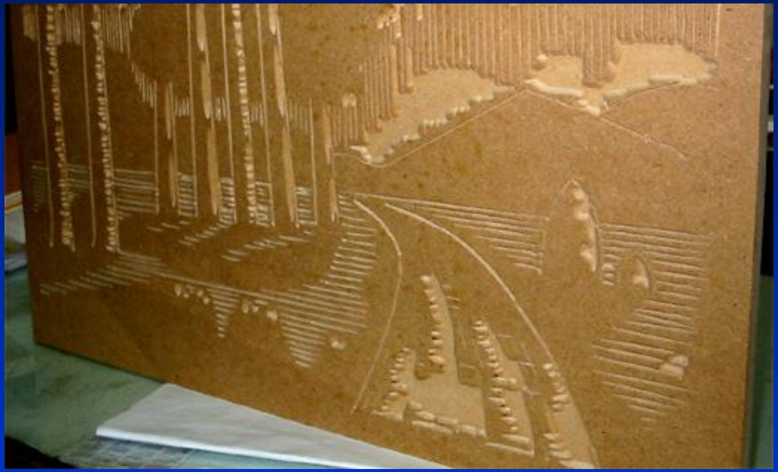

Flow Simulation (Navier-Stokes Equation)

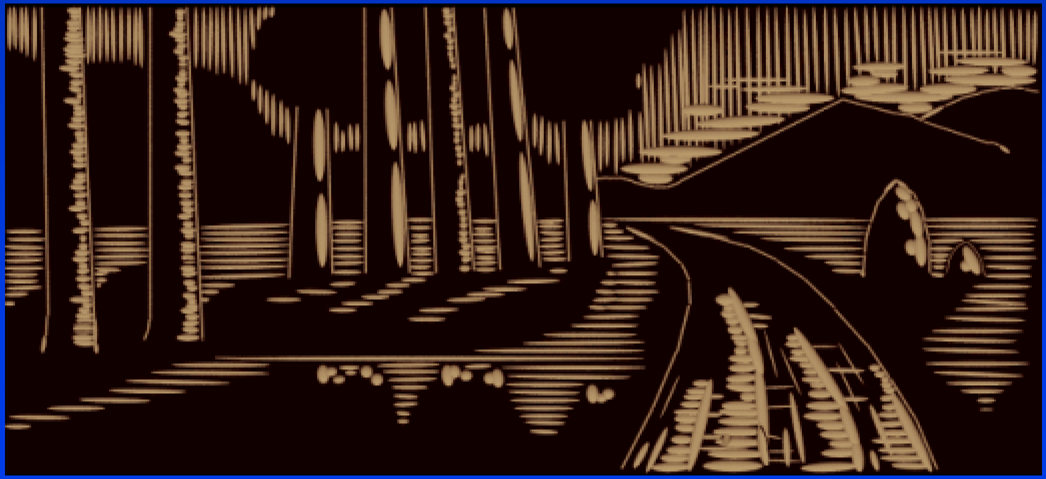



Simulation of Bubble Flow

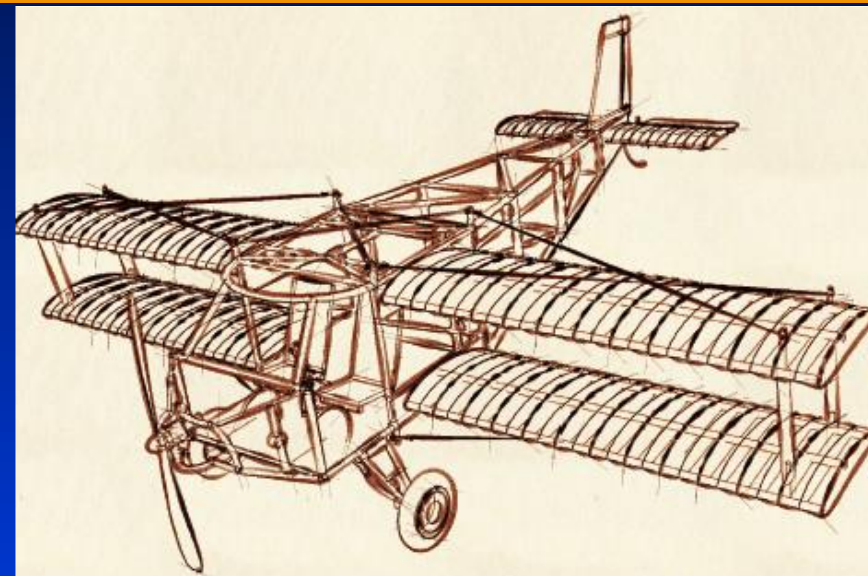


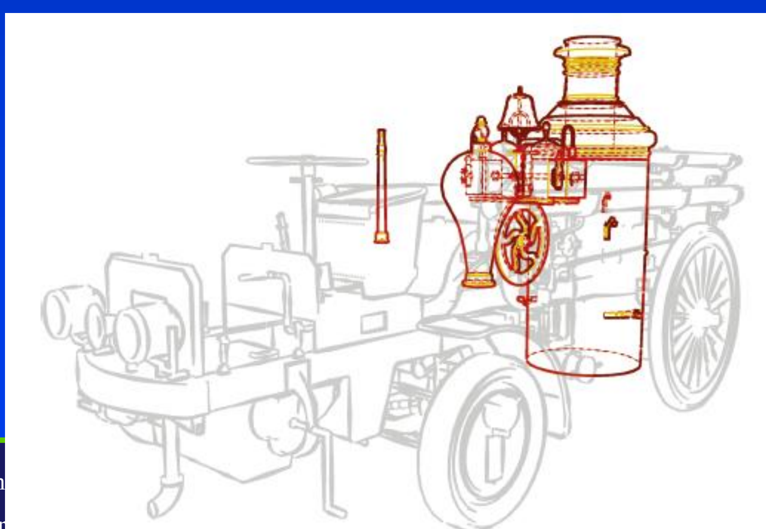
Computer Art with Physical Interface



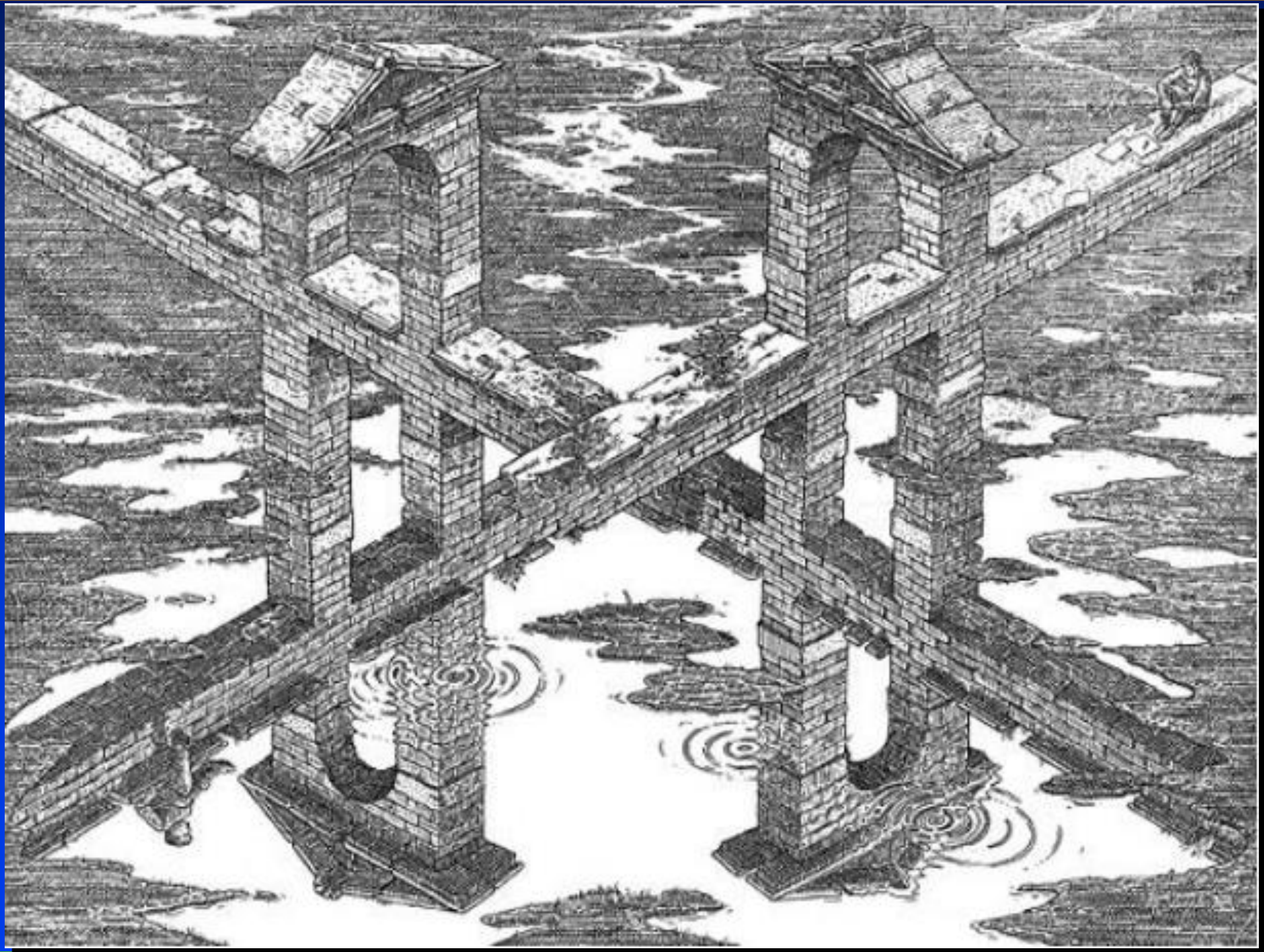


Non-Photorealistic Rendering





Impossible Figures

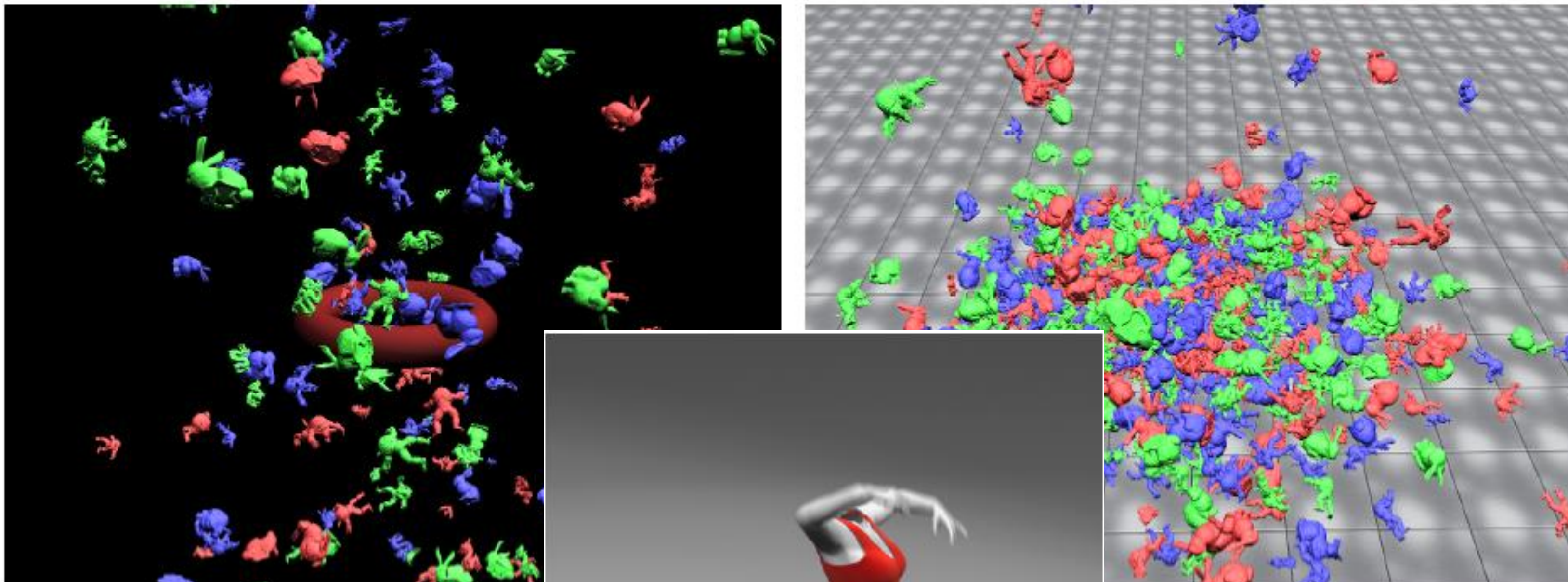


Computer Art

(a) Hebe, David, and Athena (b) Dama (c) Horse
(d) Bunny (e) Hippo (f) Zoom-in/out of landscape

Generating New Models from Examples

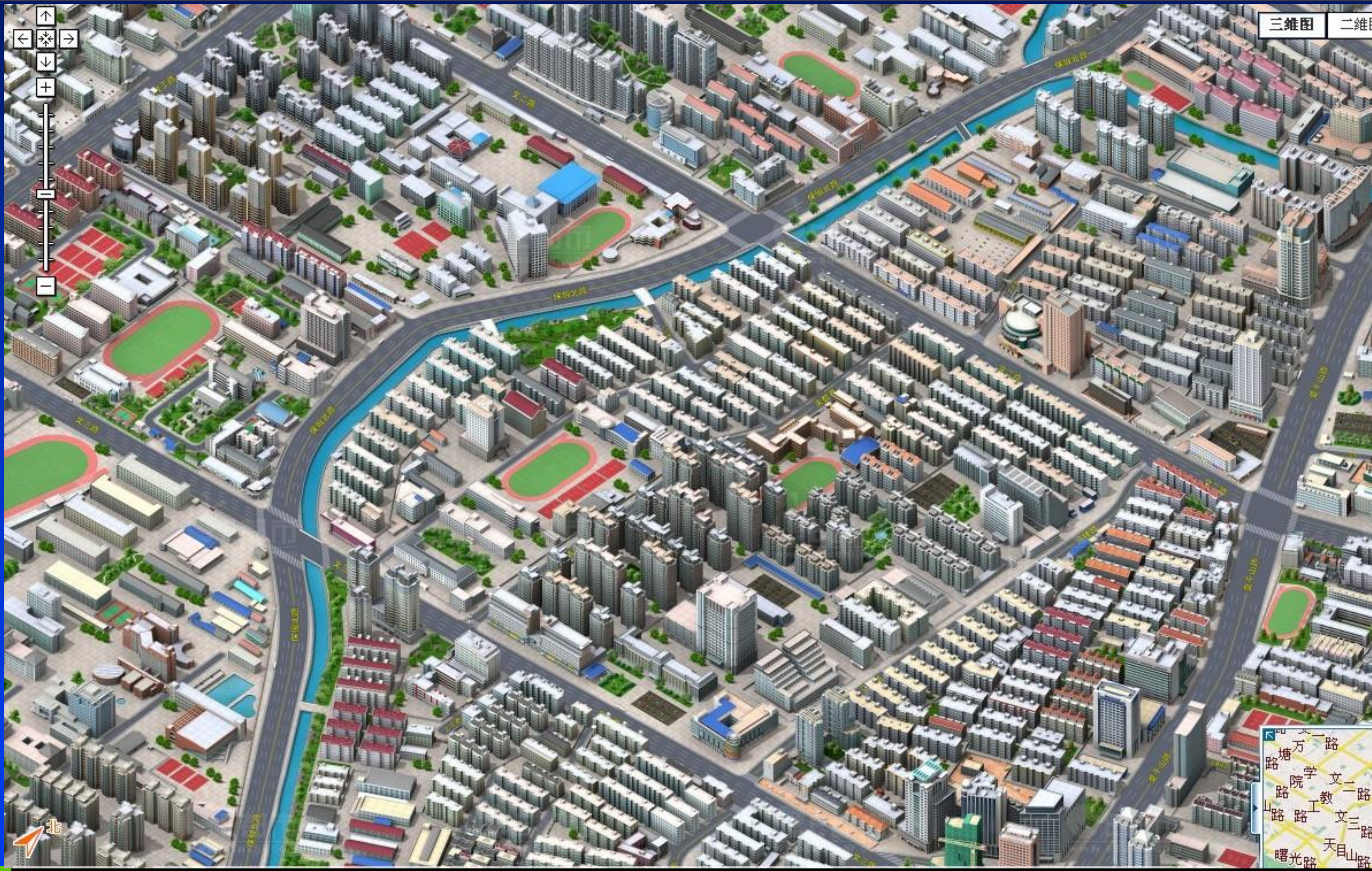
Collision Handling



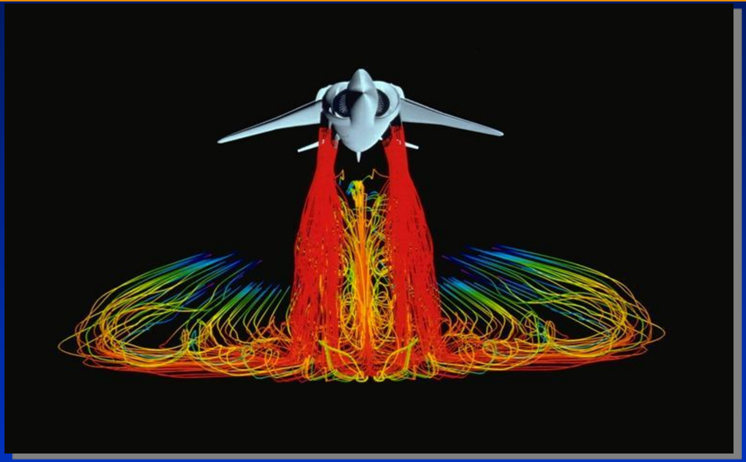
(a) (b)

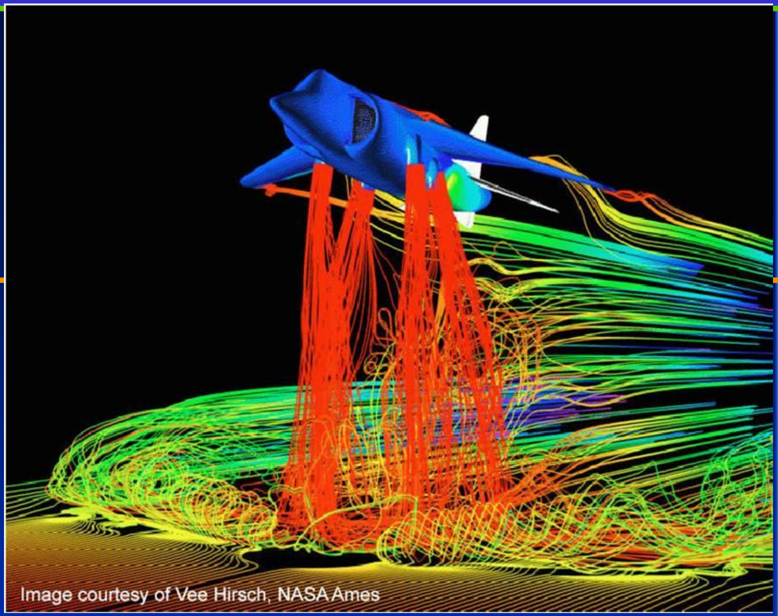
Figure 5: *Screenshots o* ... *) Plane Simulation.*

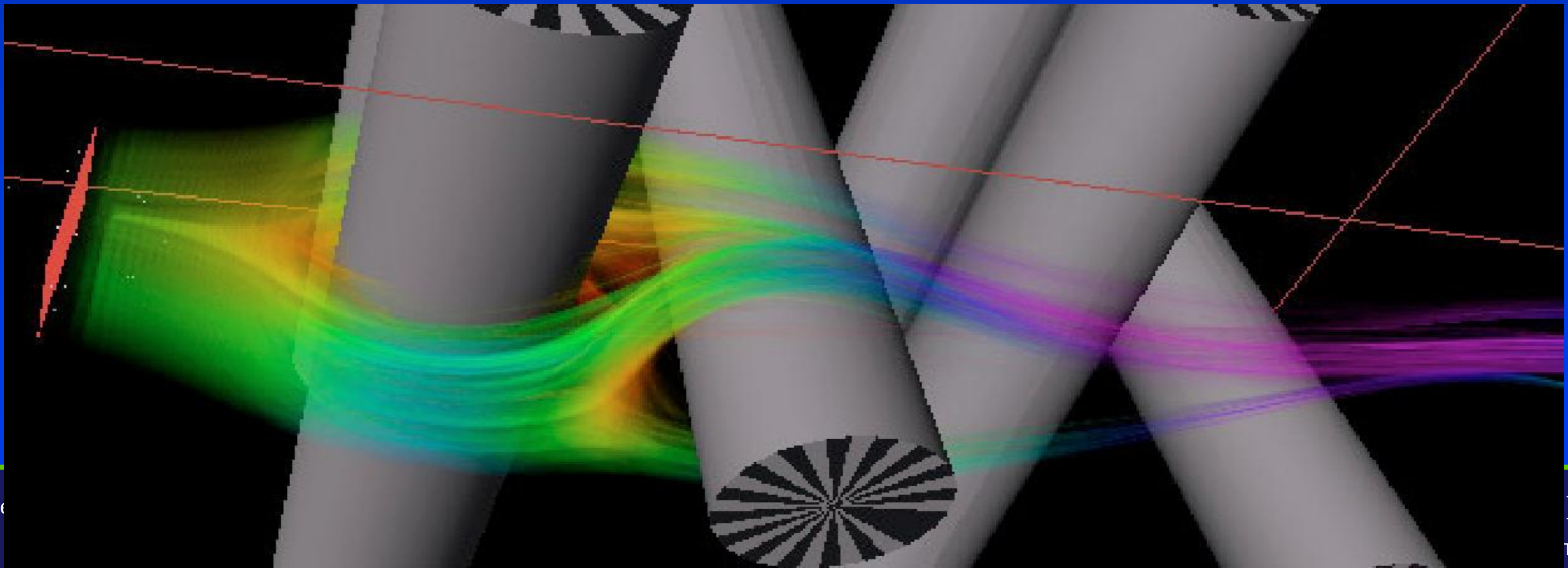
Urban Modeling



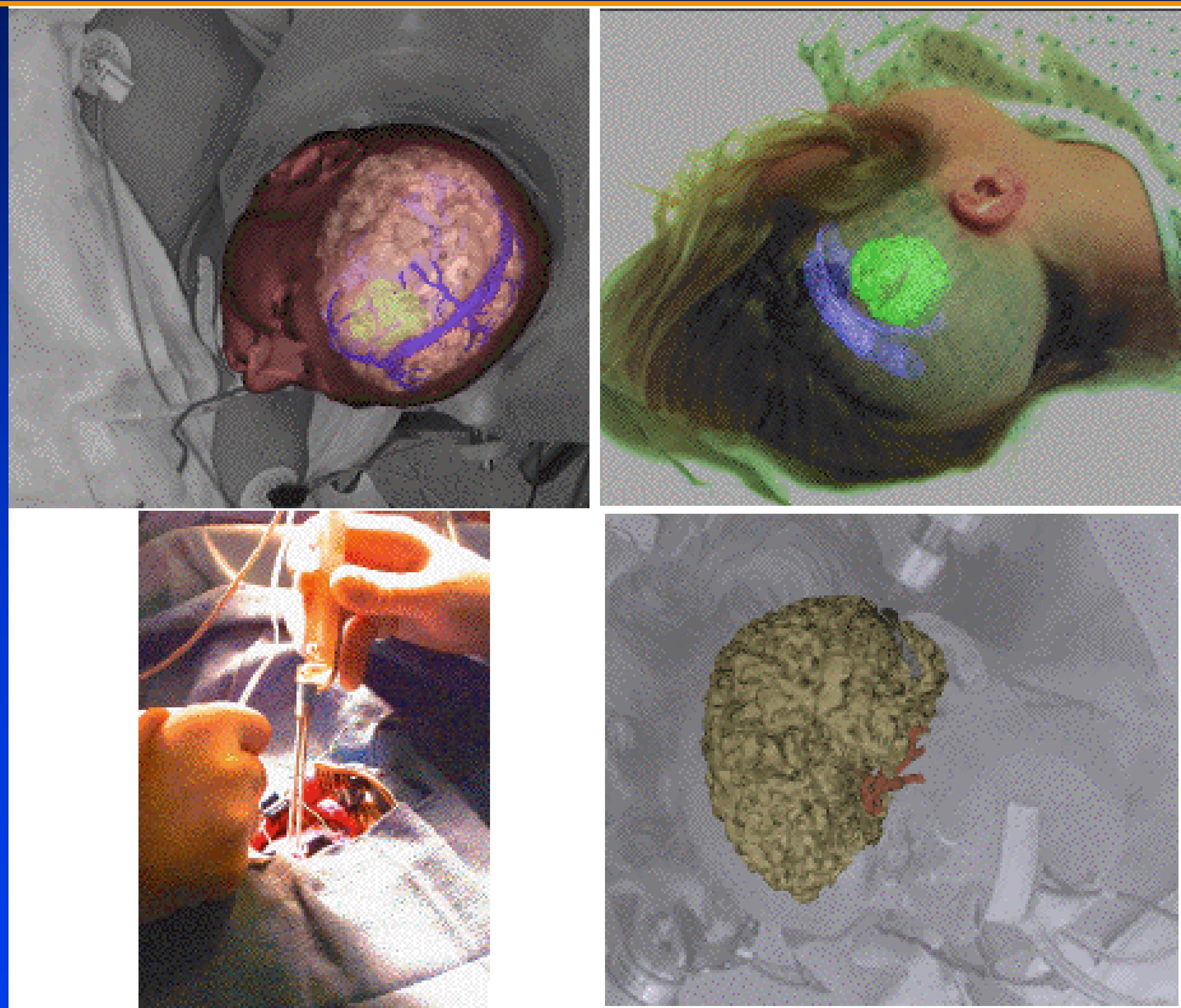
Flow Simulation





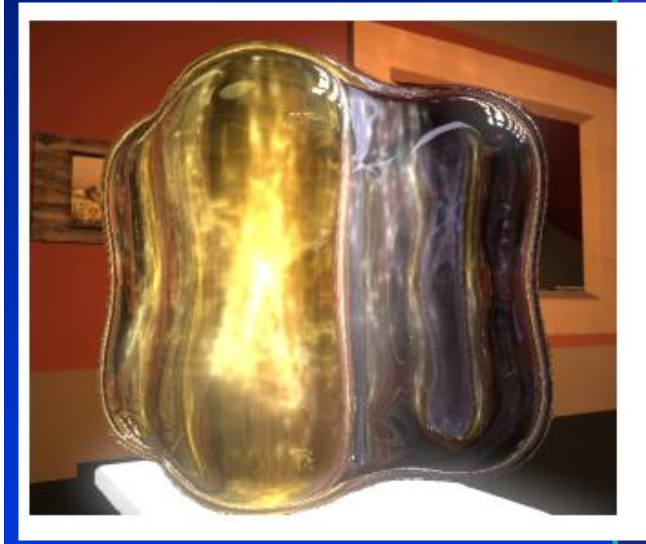


Augmented Reality in Neurosurgery



Light Transport





Different Perspectives

- Application-oriented
  - Motivation, driven by real problems
  - E.g. scientific visualization, simulation, animation, virtual reality, computer-aided design, ….
- Mathematics-oriented
  - Mathematical elements
  - E.g. computational geometry, differential geometry, PDEs, ….
- Programming-oriented
  - Modeling and rendering primitives: triangle mesh, point clouds, splines, ….
  - Basic procedural routines: edge flip, edge collapse, subdivision routines, ….
- System-oriented
  - Architecture, hardware, and software components
  - E.g. workstation, cluster, GPU, ….

What’s computer graphics course all about?

Not!

Paint and Imaging packages (Adobe Photoshop)

Cad packages (AutoCAD)

Rendering packages (Lightscape)

Modelling packages (3D Studio MAX)

Animation packages (Digimation)

Presentation Outline

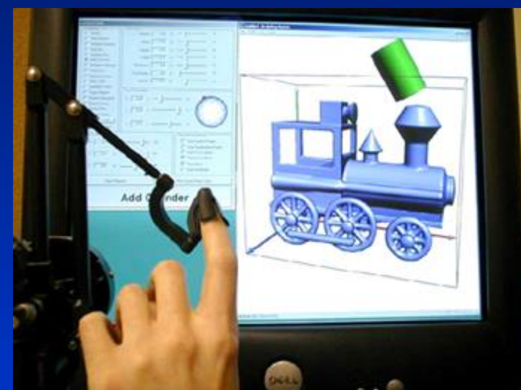
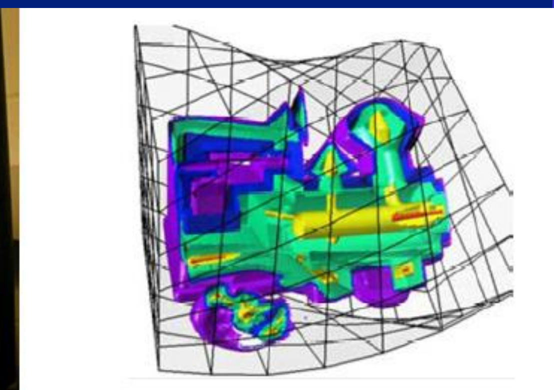
- 3D graphics pipeline

Basic Elements of Computer Graphics

- Graphics modeling: representation choices
- Graphics rendering: geometric transformation, visibility, discretization, simulation of light, etc.
- Graphics interaction: input/output devices, tools
- Animation: lifelike characters, their interactions, surrounding environments

Two Basic Questions

- What to render?
  - Scene representation
  - Modeling techniques
  - Animation, simulation
  - ...

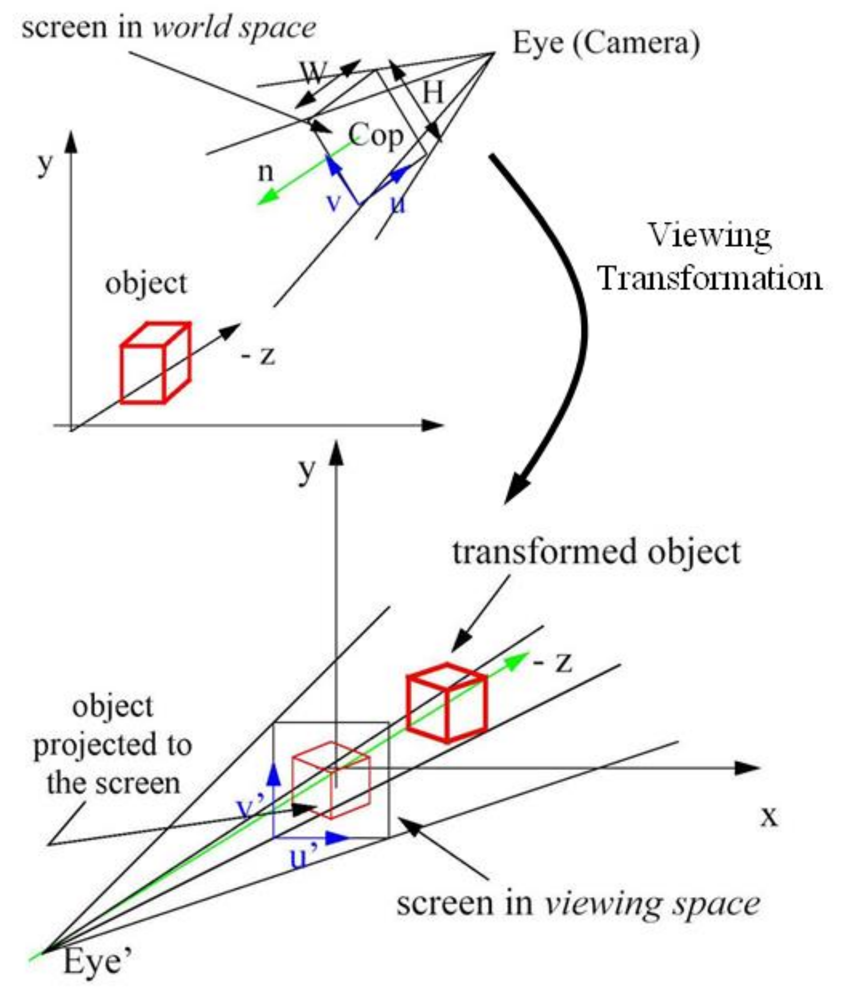
- How to put it on the screen?
  - Projection
  - Visibility
  - Illumination and shading
  - ....


Basic Topics – Undergraduate

- Hardware, system architecture
  - Basic display devices
  - Raster-scan system (rasterization)
  - Input / output devices: keyboard, mouse, haptics, data glove, scanner, ….
  - Software packages: standards, APIs, special-purpose software

Basic Topics – Undergraduate

- 2D / 3D transformation and viewing
  - 3D viewing pipeline
  - Multiple coordinate system and their transformation
  - Projection: parallel, perspective
  - Mathematical (matrix) representations

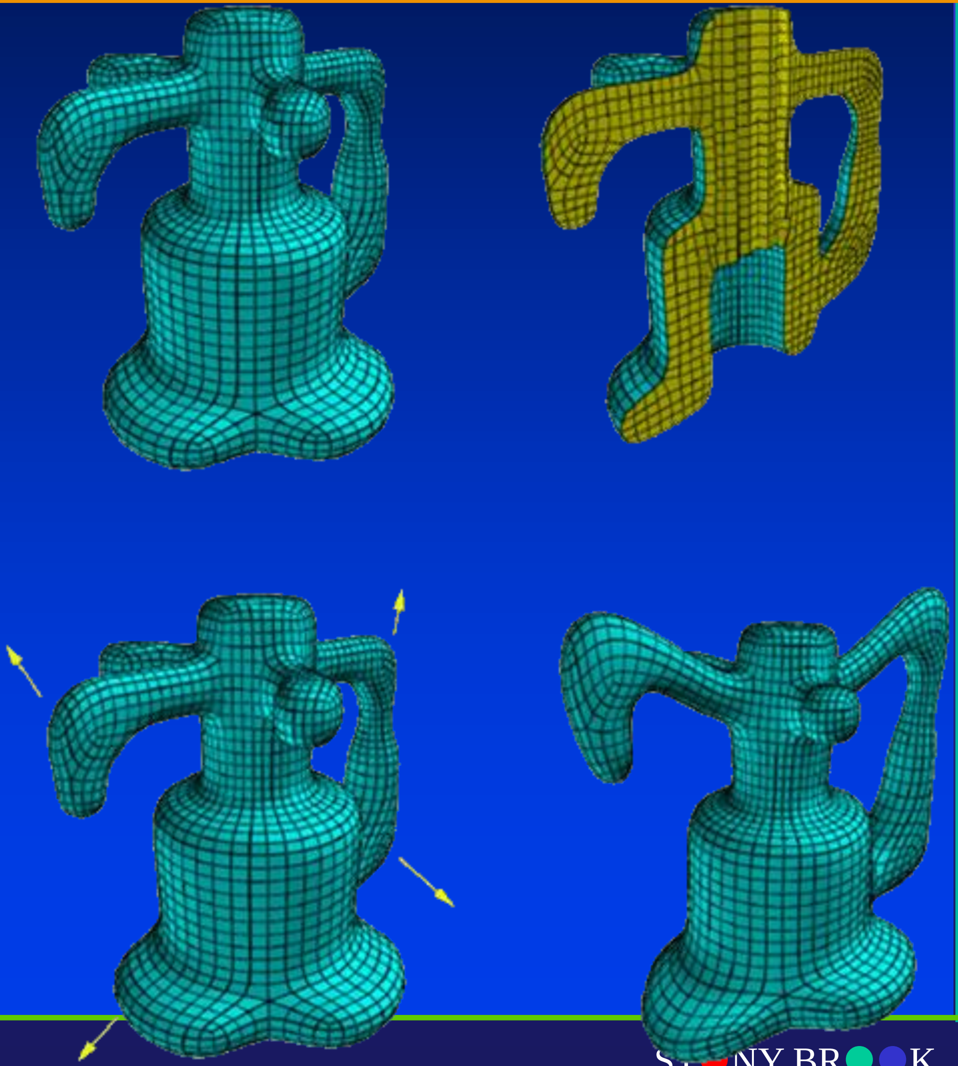


Basic Topics – Undergraduate

- Ray-casting and ray-tracing
  - Creating photorealistic rendering images

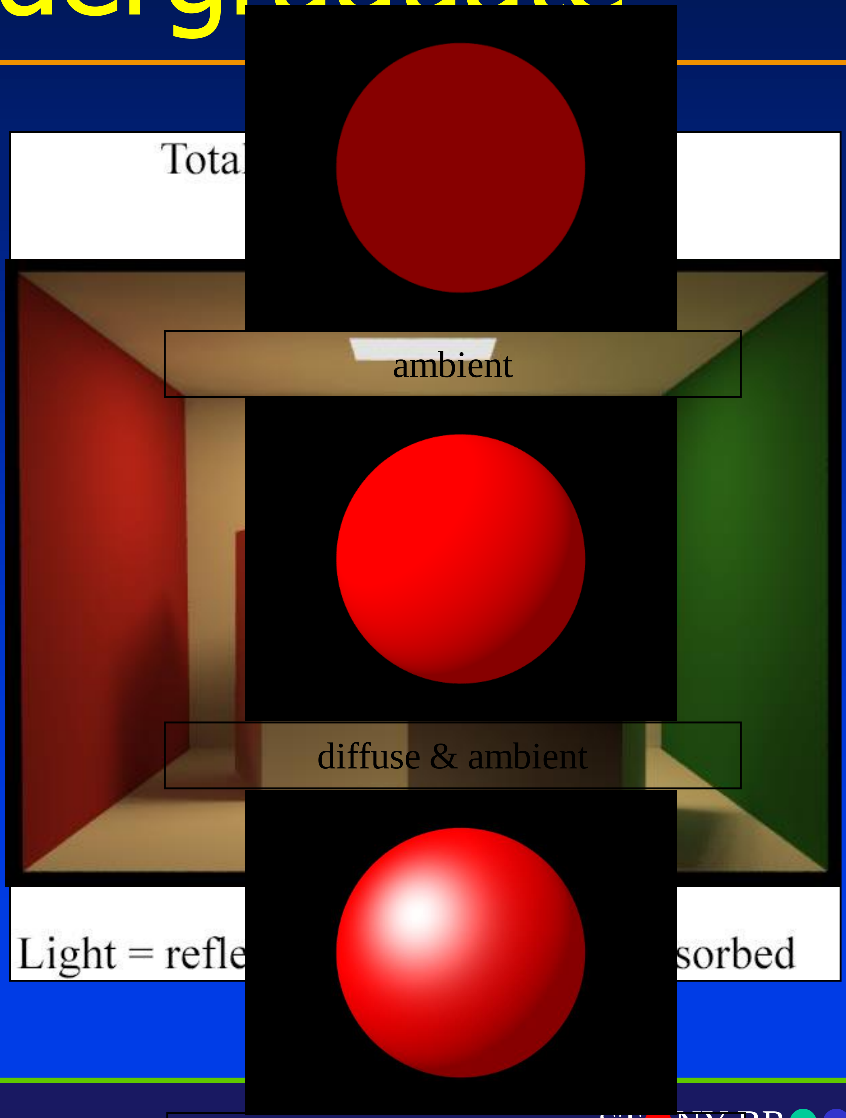
Basic Topics – Undergraduate

- Geometric models
  - Curves, surfaces, solids
  - Polygonal models
  - Parametric representations
  - Implicit representations
  - Boundary representations
  - Boolean operations (union, subtraction, ….)
  - Editing, Deformation



Basic Topics – Undergraduate

- Illumination and Shading
  - Light properties, light simulation
  - Local illumination (ambient, diffuse, specular)
  - Global illumination (ray-tracing)

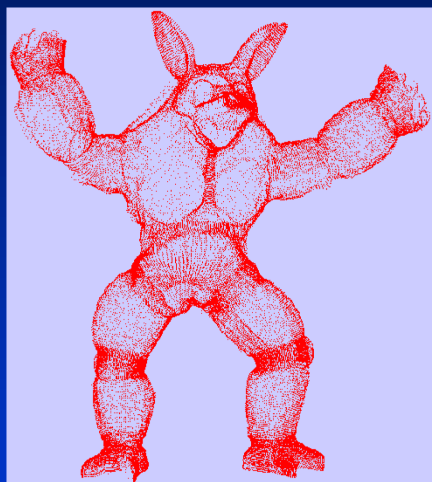


ambient

diffuse & ambient

Specular & diffuse & ambient

3D Graphics Pipeline



3D Model Acquisition → Geometric Modeling → Animation & Rendering

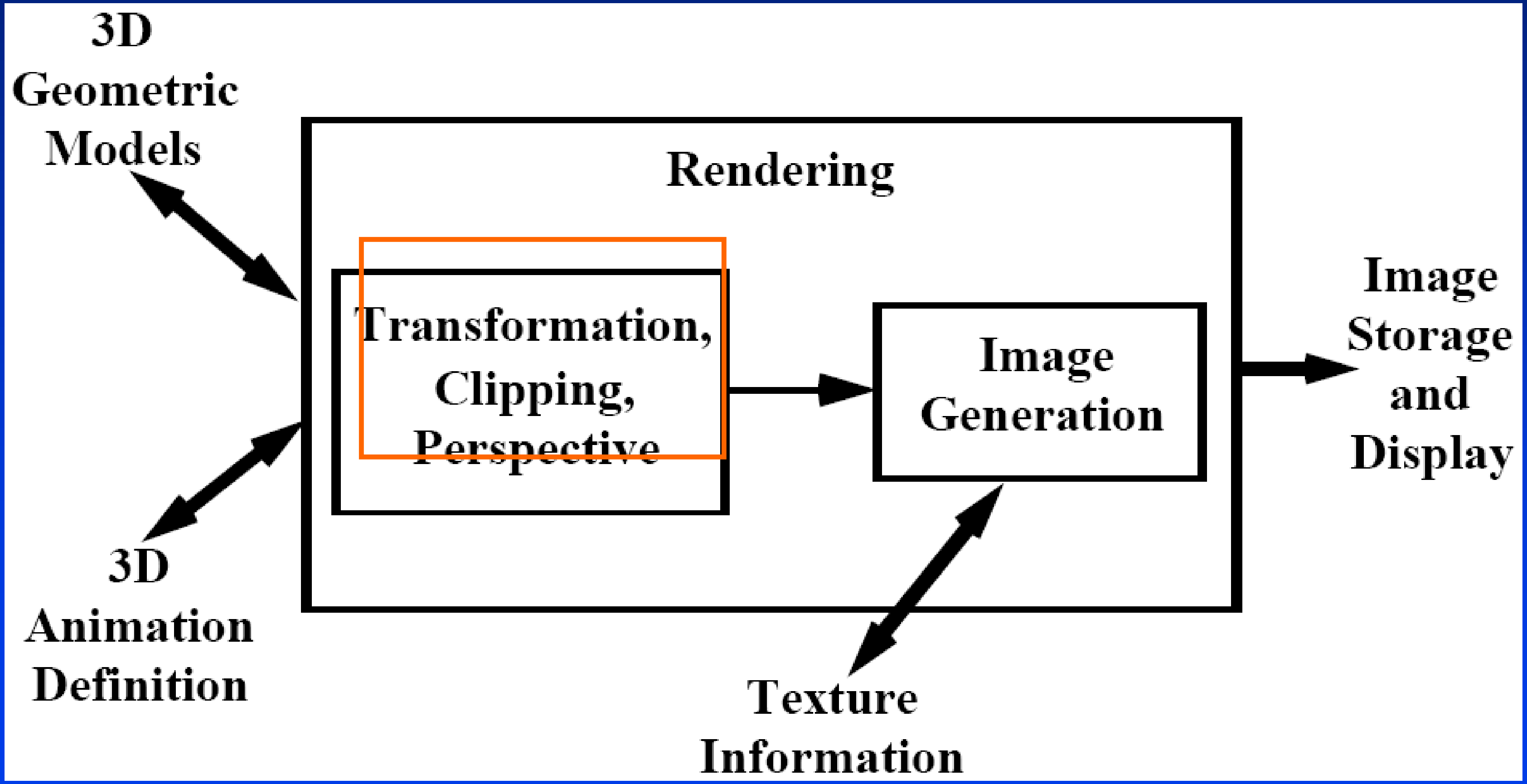
Point clouds

Curves & surfaces
Digital geometry processing
Multi-resolution modeling
…

Ray tracing
Texture synthesis
Appearance modeling
Physics-based simulation
…

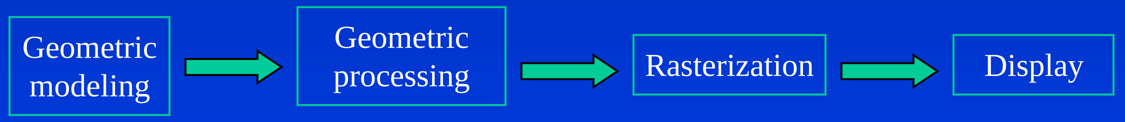
Graphics Rendering

- Conversion of a 3D scene into a 2D image



Rendering Pipeline

- Build a pipeline
- Process 3D information in a series of steps
- Each step generates results for the next one



Create 3D objects

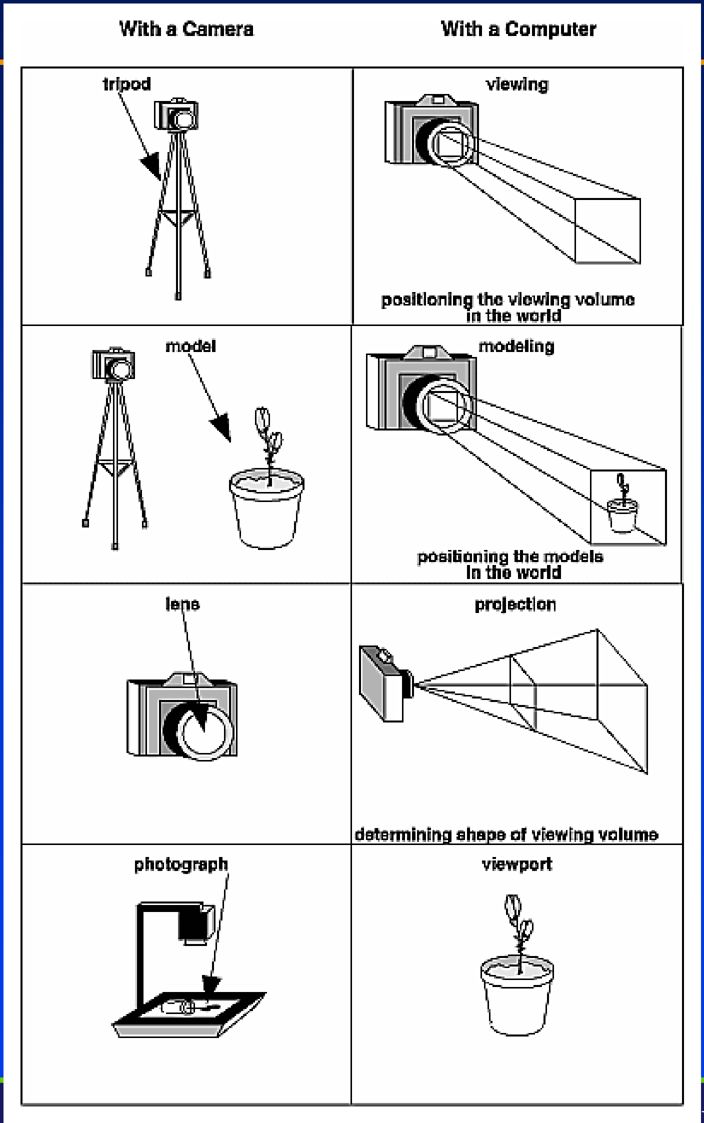
Determine which objects appear and how

Convert floating point geometry to integer pixel

Adjust pixels for special results

The Camera Analogy

Viewing:	position camera	position viewing volume
Modeling:	position model	position model
Projection:	choose lens	choose v.v. shape
Viewport:	choose photo size	choose portion of screen



Geometric Primitives

- Point
  - a location in space, 2D or 3D
  - sometimes denotes one pixel
- Line
  - straight path connecting two points
  - infinitesimal width, consistent density
  - beginning and end on points

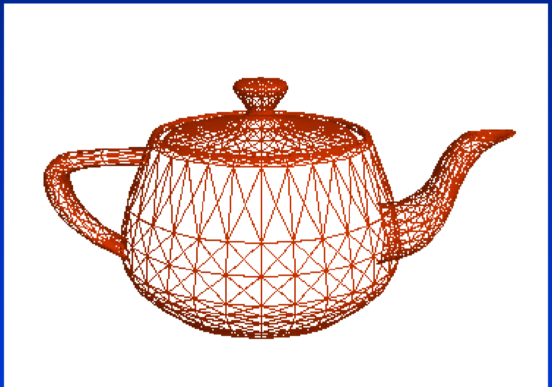
Geometric Primitives

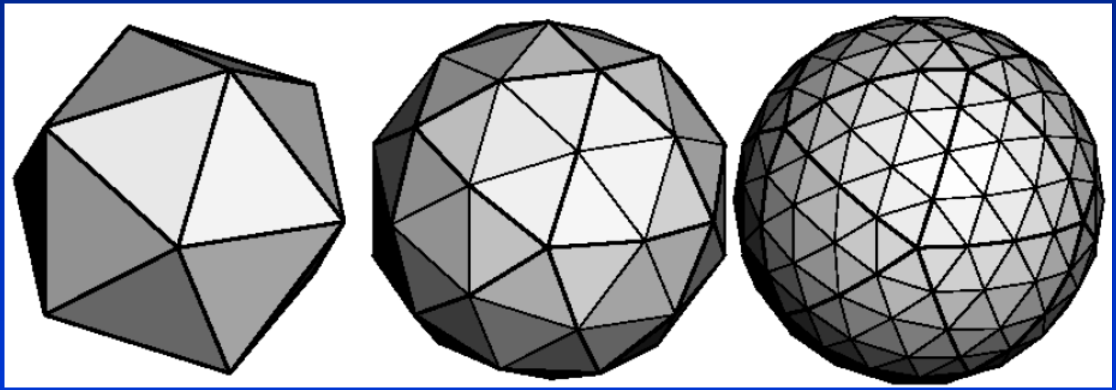
- Vertex
  - point in 3D
- Edge
  - line in 3D connecting two vertices
- Polygon/Face/Facet
  - arbitrary shape formed by connected vertices
  - fundamental unit of 3D computer graphics

3D Models

- Arbitrary shapes can be triangulated!

Polygonal approximation of surfaces

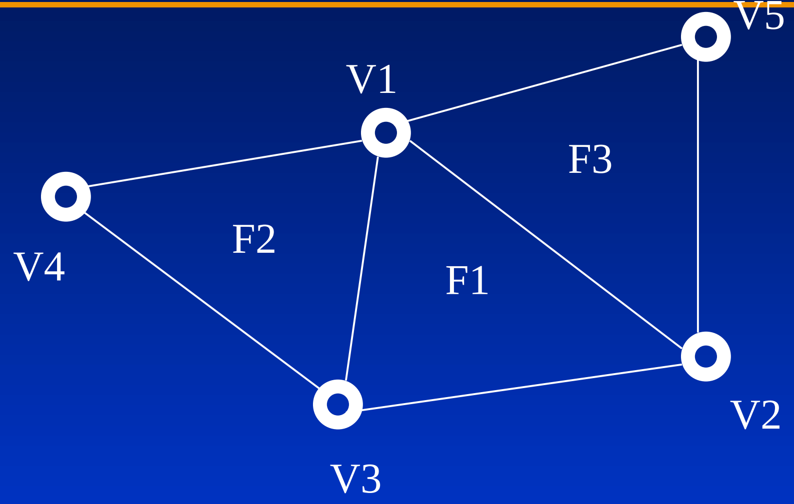




Any 2D shape (or 3D surface) can be approximated with locally linear polygons. To improve, we only need to increase the number of edges

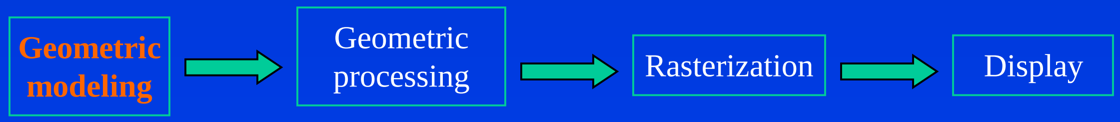
Geometric modeling → Geometric processing → Rasterization → Display

How Do We Represent Triangles?

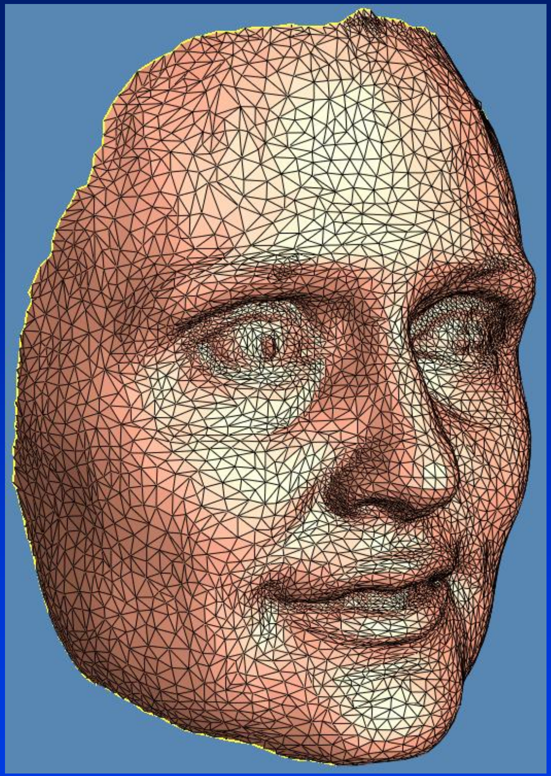


Vertex table	
V1	(x1,y1,z1)
V2	(x2,y2,z2)
V3	(x3,y3,z3)
V4	(x4,y4,z4)
V5	(x5,y5,z5)

Face table	
F1	V1,V3,V2
F2	V1,V4,V3
F3	V5,V1,V2



How Do We Represent Triangles?



mesh with 10k triangles

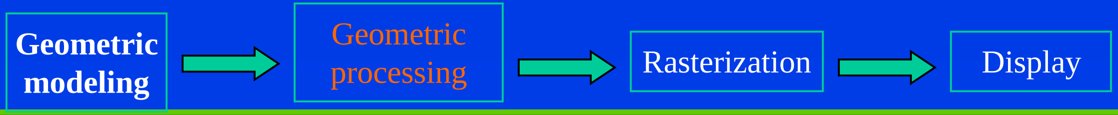
```
Vertex 1 0.6036570072 0.4613159895 0.07038059831
Vertex 2 0.6024590135 0.4750890136 0.07134509832
Vertex 3 0.6083189845 0.4888899922 0.07735790312
Vertex 4 0.611634016 0.5039420128 0.08098520339
Vertex 5 0.6236299872 0.5097290277 0.09412530065
Vertex 6 0.633580029 0.5194600224 0.1063940004
Vertex 7 0.6350849867 0.5272089839 0.1108580008
Vertex 8 0.6459569931 0.5308039784 0.1247610003
Vertex 9 0.6456980109 0.5446619987 0.1324290037
Vertex 10 0.6566579938 0.5420470238 0.1465270072
Vertex 11 0.6629710197 0.5443329811 0.1586650014
Vertex 12 0.671701014 0.541383028 0.1747259945
Vertex 13 0.6746420264 0.5451539755 0.1851660013
Vertex 14 0.6825680137 0.5424500108 0.206724003
Vertex 15 0.6884790063 0.5414119959 0.2314359993
Vertex 16 0.6935830116 0.5439419746 0.2590880096
Vertex 17 0.6981750131 0.5425440073 0.2817029953
Vertex 18 0.7026360035 0.5316519737 0.2960689962
Vertex 19 0.7058500051 0.5267260075 0.3085480034
Vertex 20 0.7095490098 0.5337790251 0.3253619969
Vertex 21 0.7104460001 0.5344949961 0.3296009898
Vertex 22 0.7158439755 0.5286110044 0.3463560045
Vertex 23 0.7237830162 0.5144050121 0.3689010143
Vertex 24 0.7282400131 0.5028949976 0.3827379942
```

```
Face 1 63 3 4
Face 2 64 63 4
Face 3 5 64 4
Face 4 65 5 6
Face 5 7 65 6
Face 6 8 65 7
Face 7 9 66 8
Face 8 10 66 9
Face 9 67 66 10
Face 10 11 67 10
Face 11 12 67 11
Face 12 14 75 13
Face 13 68 76 15
Face 14 16 68 15
Face 15 17 68 16
```

Geometric modeling → Geometric processing → Rasterization → Display

Modeling Transformation

- 3D scene
  - Many 3D models
  - Each one has its own coordinate system – object/model coordinates
- Modeling transformation
  - Place the objects in the world coordinate system
  - Translation, scaling, shearing, and rotation
- Result:
  - Object/model coordinates (local) → world coordinates (global)
  - All vertices of scene in shared 3-D “world” coordinate system

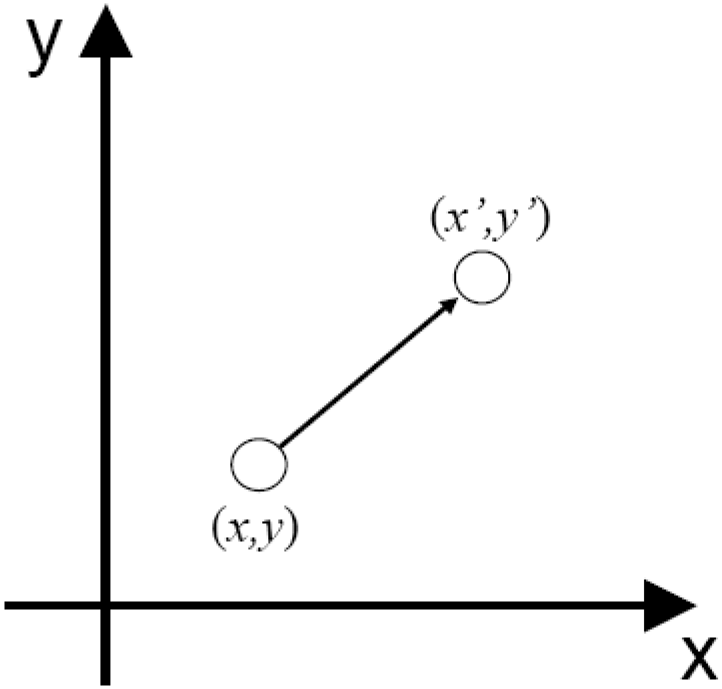


Modeling Transformation: 2D Example

- Translation

- $x' = x + t_x$
- $y' = y + t_y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

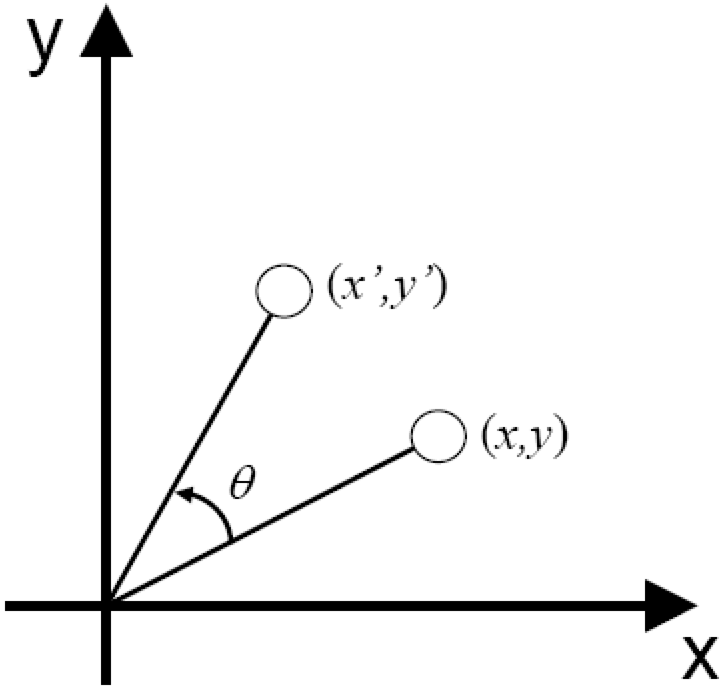


Modeling Transformation: 2D Example

- Rotation

  - $x' = x \cdot \cos\theta - y \cdot \sin\theta$
  - $y' = x \cdot \sin\theta + y \cdot \cos\theta$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

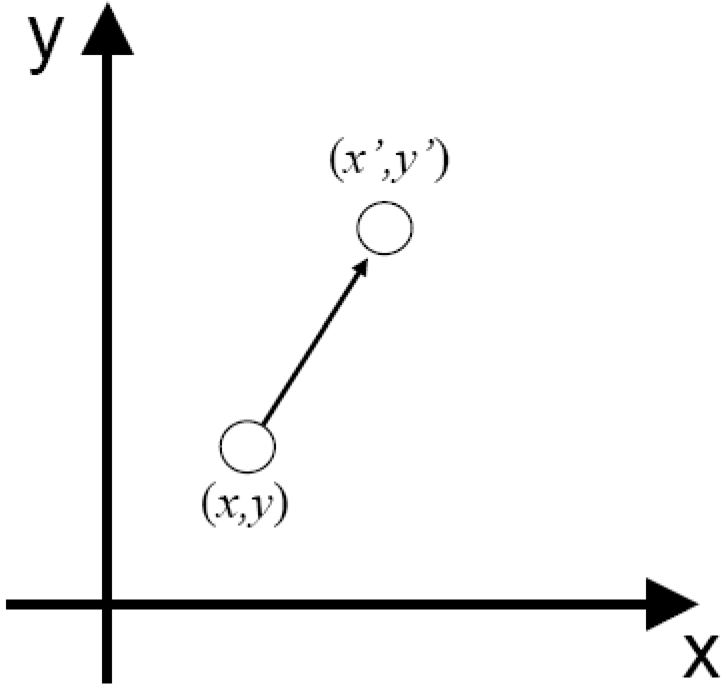


Modeling Transformation: 2D Example

- Scaling

- $x' = S_x \cdot x$
- $y' = S_y \cdot y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

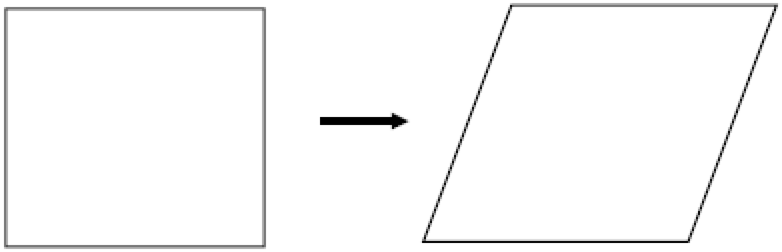


Modeling Transformation: 2D Example

- Shearing

- $x' = x + h_x \cdot y$
- $y' = y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & h_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Modeling Transformation: 2D Example

- Translation $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$
- Rotation $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
- Scaling $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
- Shearing $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & h_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
- Can we represent the above transformations in a unified format?

Homogeneous Coordinates

- Each point (x, y) is represented as (x, y, 1)
  - Append a 1 at the end of vector!
- All transformations can be represented as matrix multiplication!
- Composite transformation becomes much easier

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

Conventional coordinate

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous coordinate

Homogeneous Coordinates

- All transformations can be represented as matrix multiplication!
- Composite transformation becomes much easier!

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

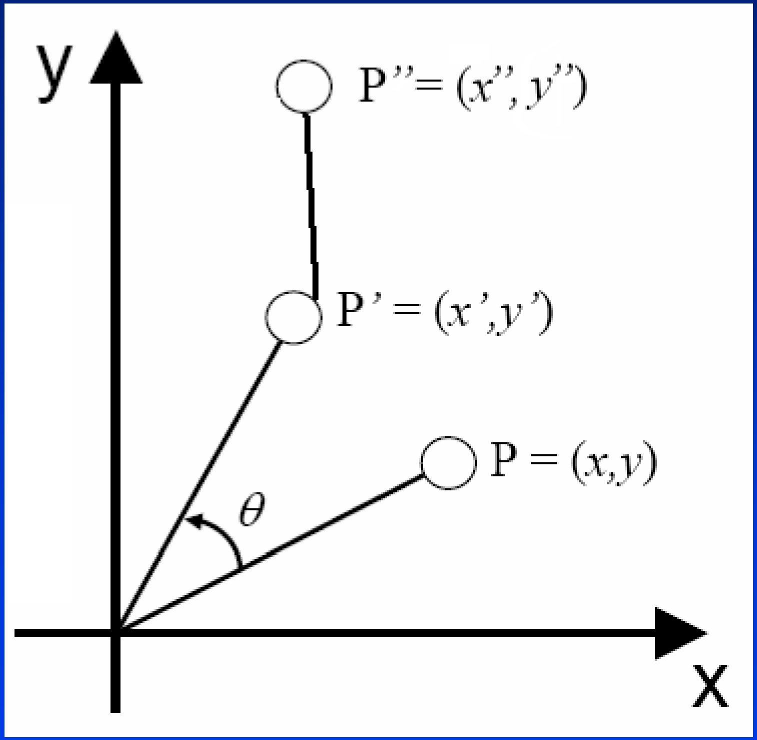
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- Composite transformation



$$P' = R(\theta) \cdot P$$

$$P'' = T(t_x , t_y) \cdot P'$$

$$P'' = T(t_x , t_y) \cdot R(\theta) \cdot P$$

- Matrix multiplication

Homogeneous Coordinates

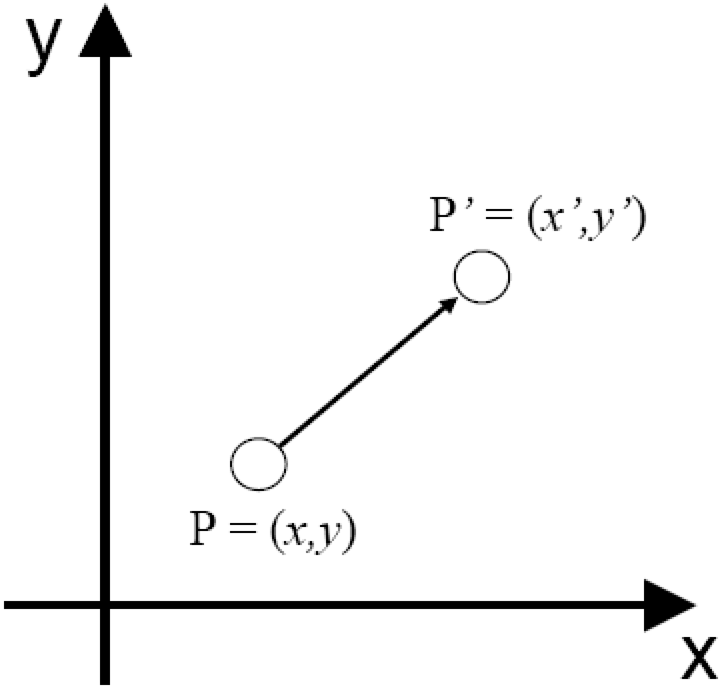
- Transformation in homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- $x' = x + t_x$
- $y' = y + t_y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{T}(t_x, t_y) \bullet \mathbf{P}$$



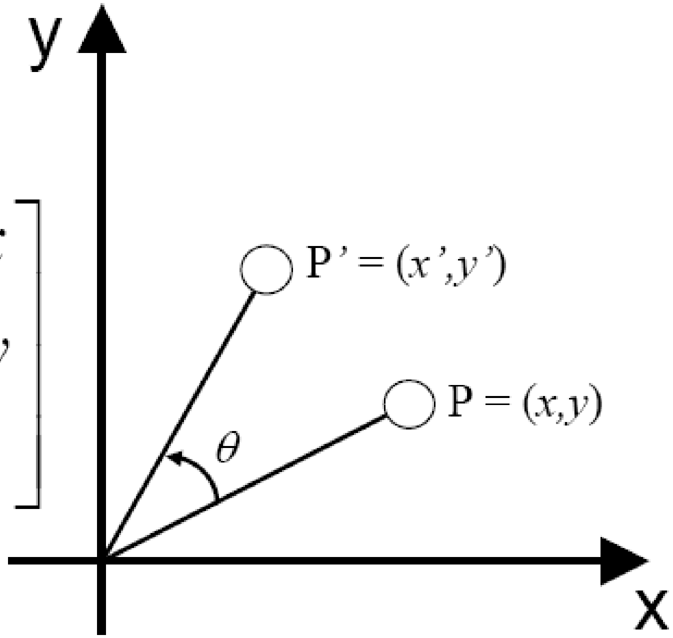
Homogeneous Coordinates

- Rotation in homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- $x' = x \cdot \cos\theta - y \cdot \sin\theta$
- $y' = x \cdot \sin\theta + y \cdot \cos\theta$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



$$\mathbf{P}' = \mathbf{R}(\theta) \cdot \mathbf{P}$$

Homogeneous Coordinates

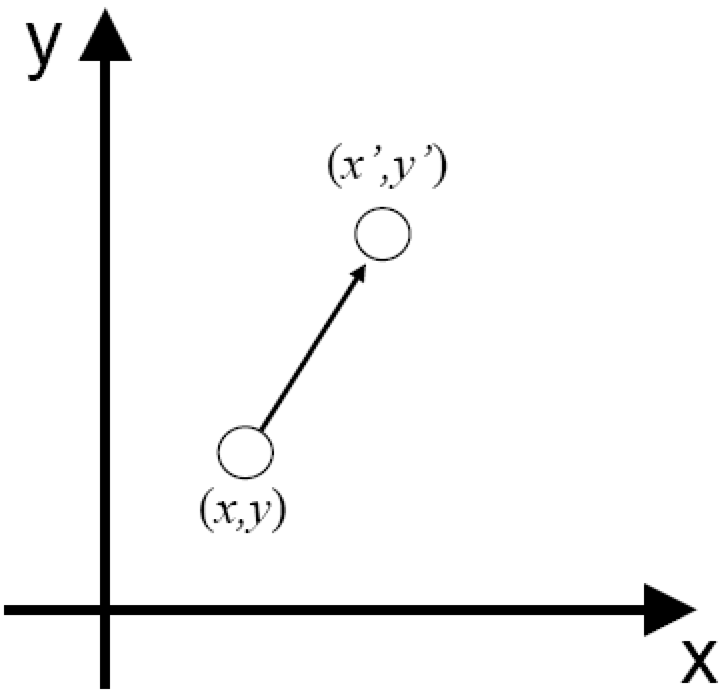
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Scaling in homogeneous coordinates

- $x' = s_x \cdot x$
- $y' = s_y \cdot y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S}(s_x\ ,\ s_y) \cdot \mathbf{P}$$



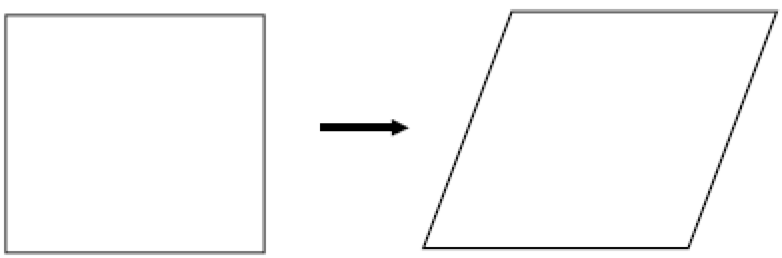
Homogeneous Coordinates

- Shearing in homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & h_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- $x' = x + h_x \cdot y$
- $y' = y$

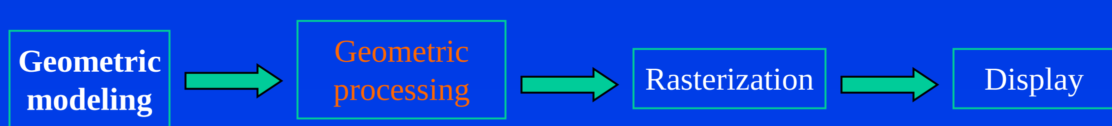
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



$$\mathbf{P}' = \mathbf{SH}_x \cdot \mathbf{P}$$

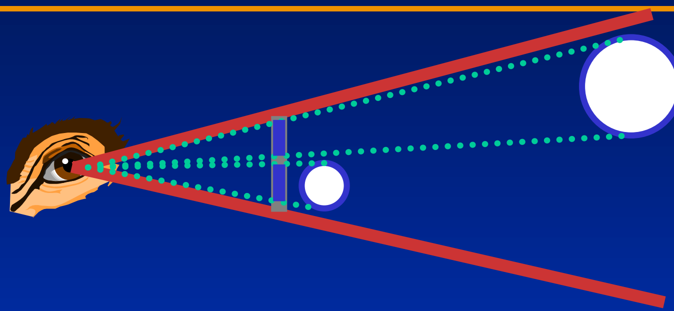
Viewing Transformation

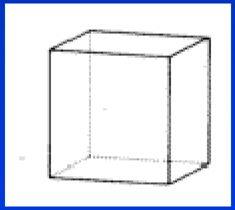
- Rotate & translate the world to lie directly in front of the camera
  - Typically place camera at origin
  - Typically looking down -Z axis
- Result:
  - World coordinates → view coordinates
  - Scene vertices in 3-D “view” or “camera” coordinate system

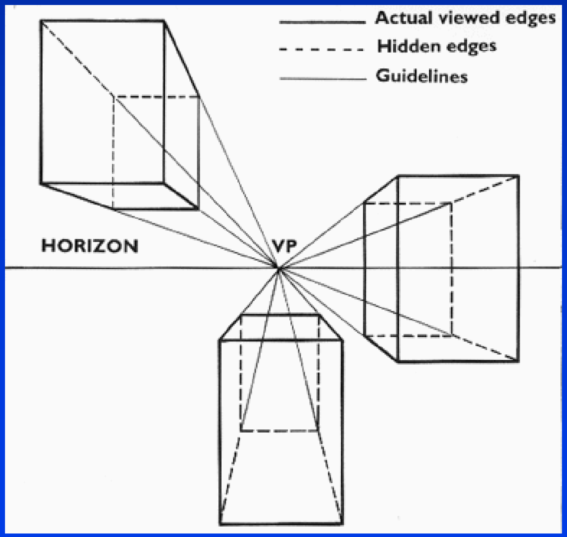


Projection

- Projection transform
  - Perspective projection
  - Orthographic projection
- Results
  - View coordinates → screen coordinates
  - 2-D screen coordinates of clipped vertices



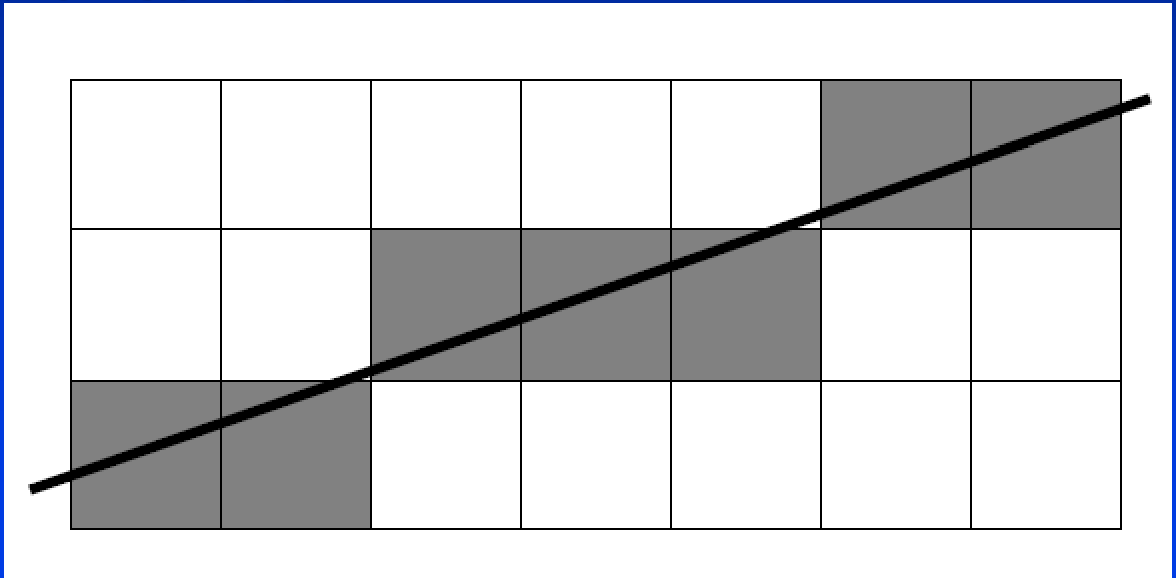


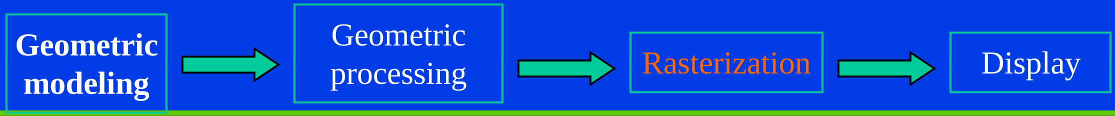


Geometric modeling → Geometric processing → Rasterization → Display

Rasterization & Display

- Convert a vertex representation in the view coordinate system to a pixel representation on computer screen



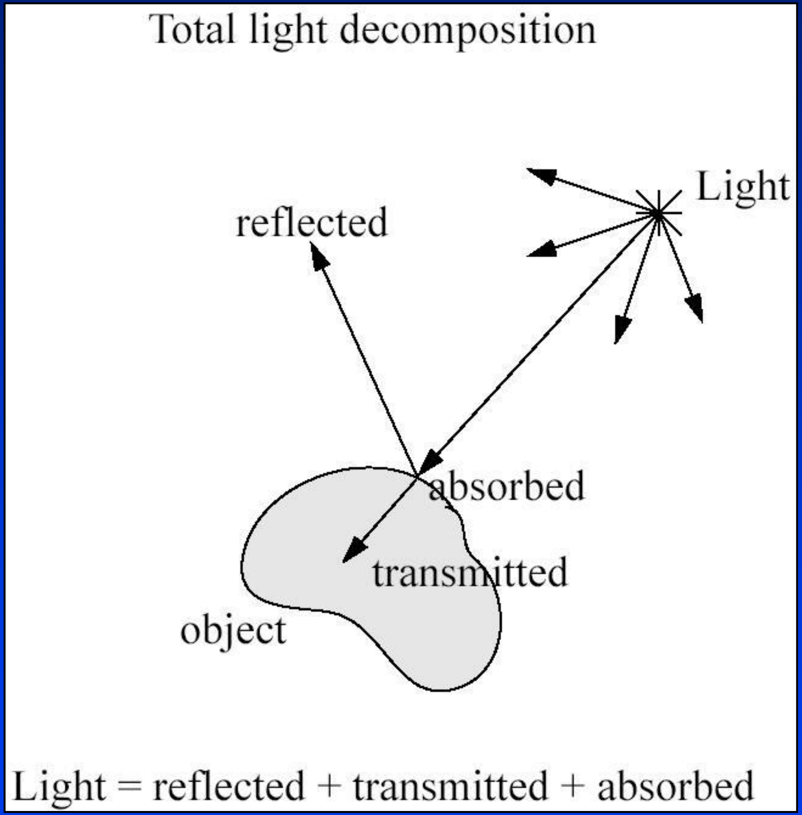


Basic Topics – Undergraduate

- Hardware, system architecture, raster-scan graphics (rasterization)
- 2D / 3D transformation and viewing
- Ray-casting and ray-tracing
- Interface
- Geometric models
- Color representations
- Hidden object removal
- Illumination models

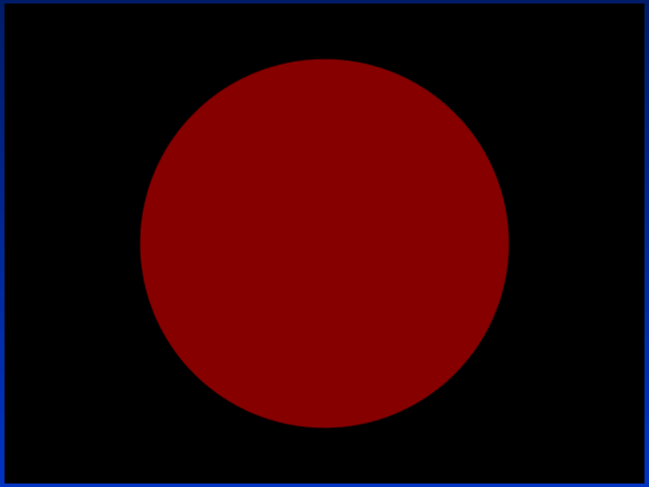
Illumination and Shading

- Now we'll look at how to shade **surfaces** to make them look 3D
- We'll see different *shading models*, or frameworks that determine a surface's color at a particular point
- These shading models can be easily modified to incorporate illumination and shading into the volume rendering pipeline
- A shading model checks what the lighting conditions are and then figures out what the surface should look like based on the lighting conditions and the surface parameters:
- Amount of light reflected (and which color(s))
- Amount of light absorbed
- Amount of light transmitted (passed through)
- Thus, we can characterize a surface's shading parameters by how much incoming light that strikes a surface is reflected to the eye, absorbed by the object, and transmitted



Ambient Reflection

- *Ambient reflection* refers to reflected light that originally came from the "background" and has no clear source
- Models general level of brightness in the scene
- Accounts for light effects that are difficult to compute (secondary diffuse reflections, etc)
- Constant for all surfaces of a particular object and the directions it is viewed from
- Directionless light
- One of many hacks or kludges used in computer graphics since every ray of light or photon has to come from somewhere!

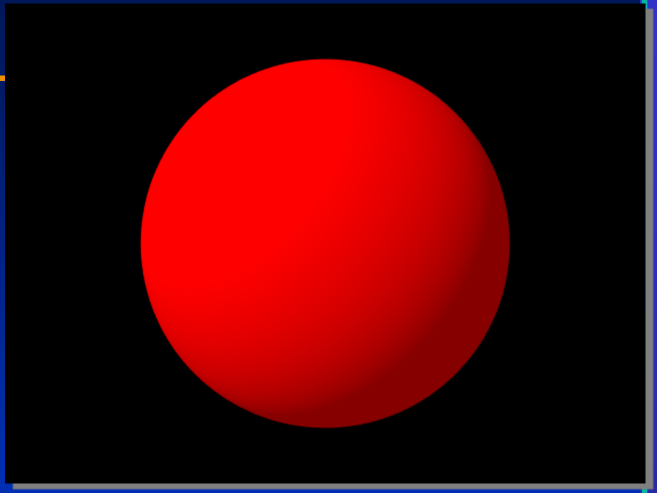


Ambient-lit sphere

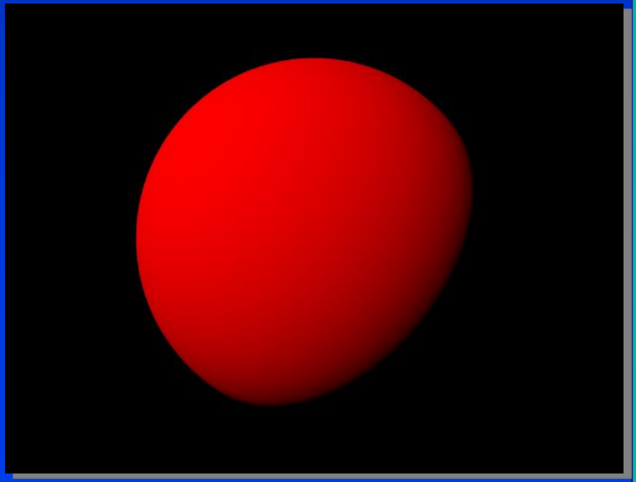
- Imagine yourself standing in a room with the curtains drawn and the lights off
- Some sunlight will still get through, but it will have bounced off many objects before entering the room
- When an object reflect this kind of light, we call it *ambient reflection*
- $I_a = k_a \cdot I_A$ I_A = ambient light k_a = material's ambient reflection coefficient

Diffuse Reflection

- Models dullness, roughness of a surface
- Equal light scattering in all directions
- For example, chalk is a diffuse reflector
- Unlike ambient reflection, diffuse reflection is dependent on the location of the light relative to the object
- So, if we were to move the light from the front of the sphere to the back, there would be little or no diffuse reflection visible on the near side of the sphere
- Compare with ambient light, which has no direction
- With ambient, it doesn't matter where we position the camera since the light source has no true position
- Computer graphics purists don't use ambient lights and instead rely on diffuse light sources to give some minimal light to a scene



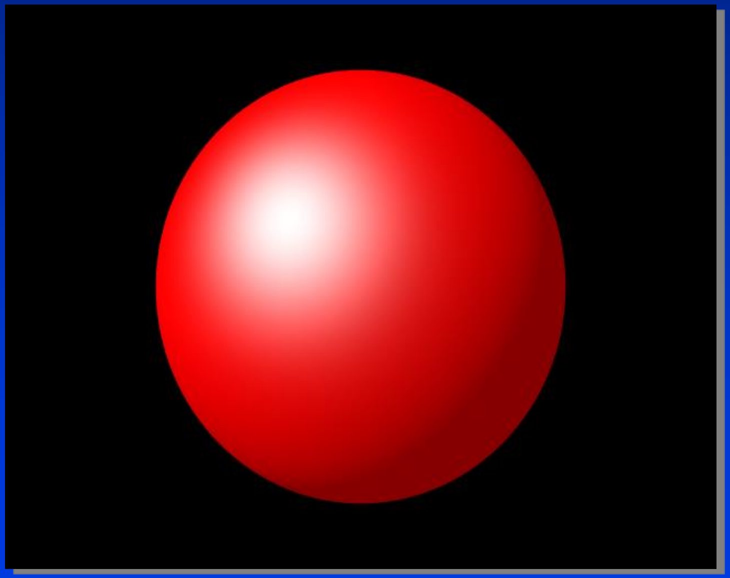
Ambient & diffuse



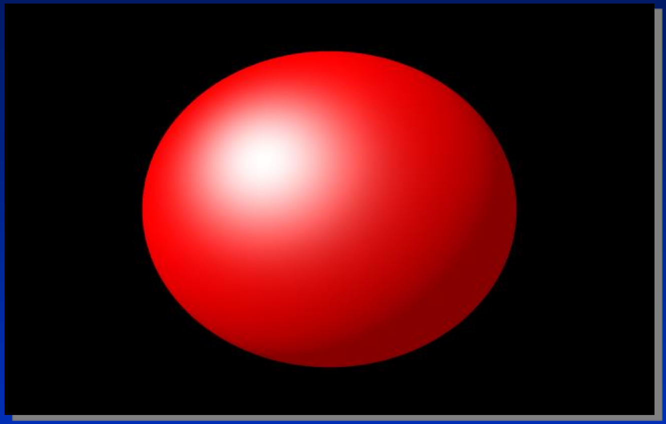
Diffuse only

Specular Reflection

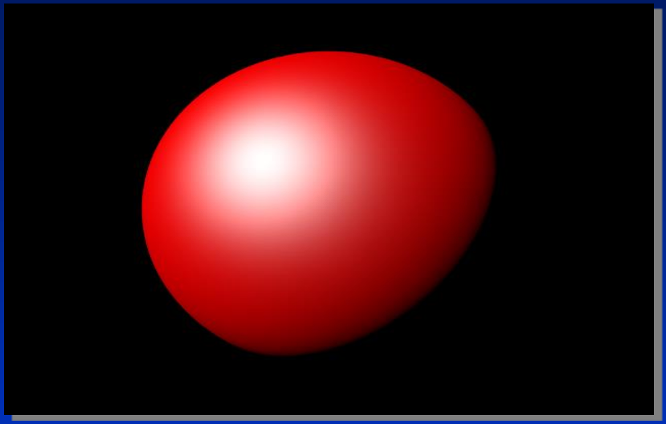
- Models reflections on shiny surfaces (polished metal, chrome, plastics, etc.)
- Specular reflection is *view-dependent* – the specular *highlight* will change as the camera's position changes
- This implies we need to take into account not only the angle the light source makes with the surface, but the angle the viewing ray makes with the surface
- Example: the image you perceive in a mirror changes as you move around
- Example: the chrome on your car shines in different ways depending on where you stand to look at it



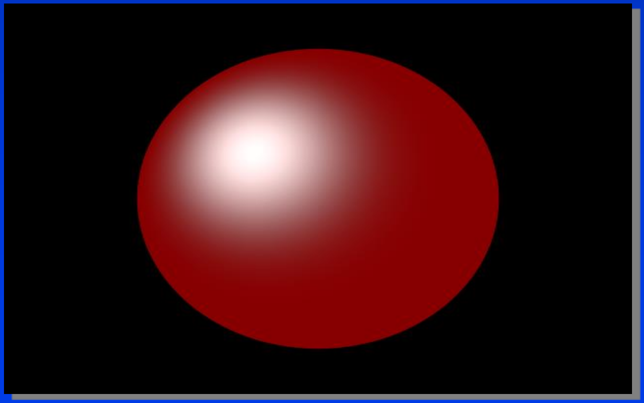
Specular Reflection



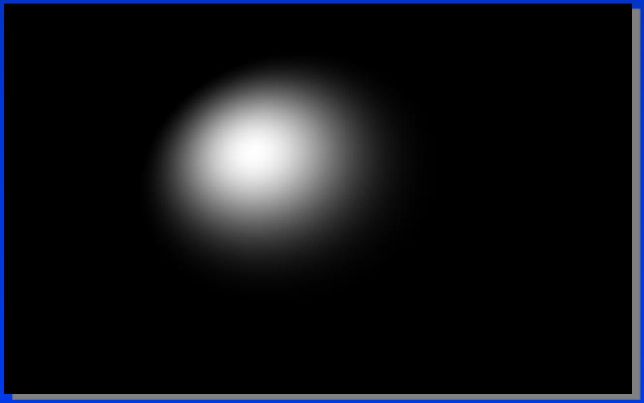
Specular & diffuse & ambient



Specular & diffuse



Specular & ambient



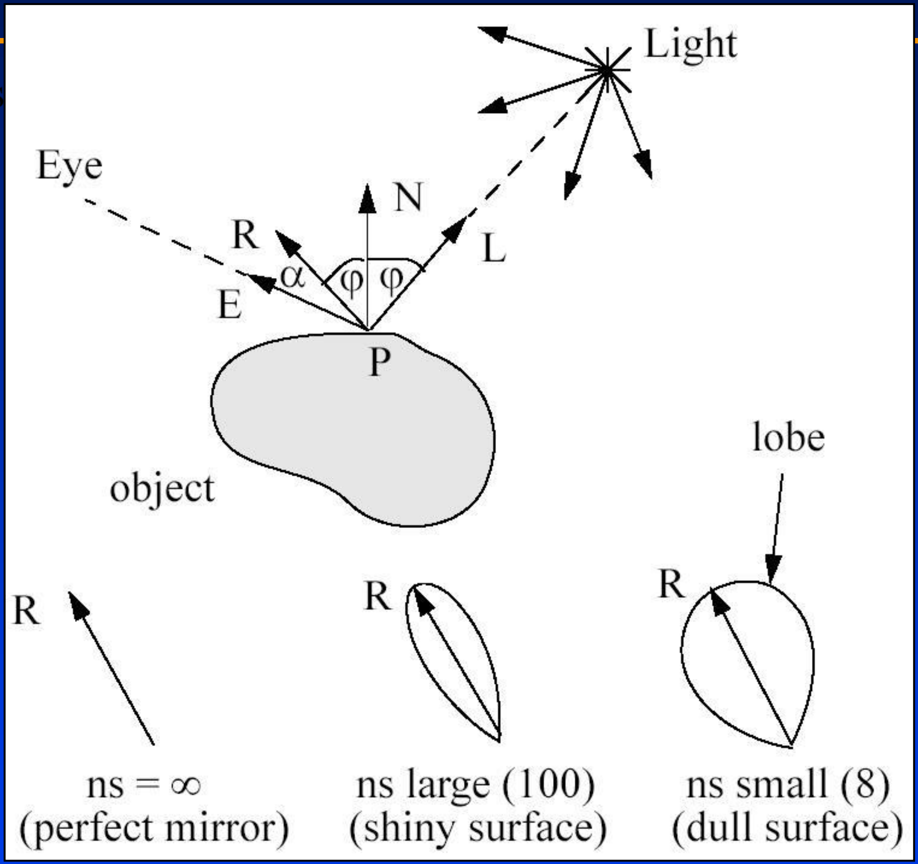
Specular only

Specular Reflection

- Ideal specular reflector (perfect mirror) reflects light only along reflection vector R
- Non-ideal reflectors reflect light in a lobe centered about R
- Phong specular reflection model:

$$I_s = k_s\, I_L\, (\cos \alpha)^{ns} = k_s\, I_L\, (E \cdot R)^{ns}$$

- cos(α) models this lobe effect
- The width of the lobe is modeled by Phong exponent ns, it scales cos(α)
- I_L: intensity of light source
- L: light vector
- R: reflection vector = 2 N (N·L) - L
- E: eye vector = (Eye-P) / |Eye-P|
- α: angle between E and R
- ns: Phong exponent
- k_s: specular reflection coefficient



increasing *ns* value

Presentation Outline

- Programming basics

Programming in Graphics

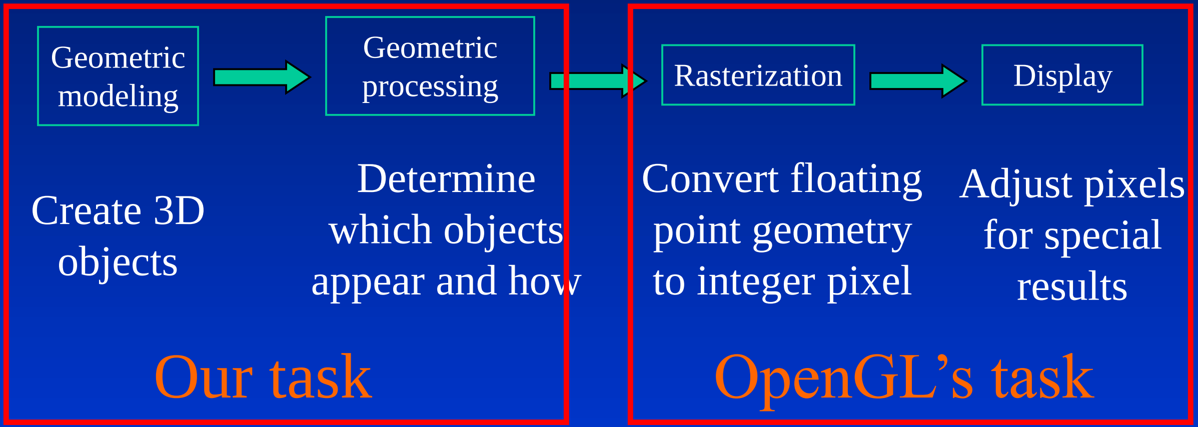
- Programming languages
  - C/C++, JAVA
- Graphics library -- a software interface to graphics hardware
  - Easy to use
  - Programs run efficiently
  - Hardware-independent
- Examples:
  - OpenGL
  - DirectX (Microsoft)
  - Java3D

OpenGL

- Contains a library of over 200 functions
- Portable
  - Implementations available for nearly all hardware and operating systems
- Portability → input or windowing are *not* included
  - Options for Windows: GLUT or MFC
  - GLUT = OpenGL Utility Toolkit
  - Implementations of GLUT exist for most computing environments
  - GLUT is portable
- Controlled by the OpenGL Architectural Review Board
  - SGI, IBM, NVIDIA, ATI, … -- some major players in CG
- www.opengl.org

Major Elements in OpenGL Programming

- Let us recall the rendering pipeline (which is shown earlier)



- Our focus now becomes: geometric modeling and processing
- Rasterization & display operations are mostly done for us by OpenGL (it also supports certain special rendering effects such as texture mapping and anti-aliasing)

Major Elements in OpenGL Programming

- Geometric primitives
  - Points, lines, polygons
  - Smooth curves and surfaces rendered in a discrete form
- Appearance
  - Color and material
  - Definition of geometric objects is separate from definition of appearance

OpenGL Commands: A Quick Look

- Just function calls:

  glColor3f(1.0, 1.0, 1.0);

  - GL prefix: gl
  - command name: Color
  - Number of arguments (if variable): 3
  - type suffix (if variable), can also end with "v": f

- Same command, different arguments:

  glColor3b(255,255,255); -- same result

Draw Geometric Primitives

- **Example**

```
glBegin(mode);

        glColor3f(1,0,0);

        glVertex3f(0,1.5,-2);
        glVertex3f(0,0.8,0);
        ......

glEnd(void);
```

glBegin(mode); ← Specify geometric primitivies

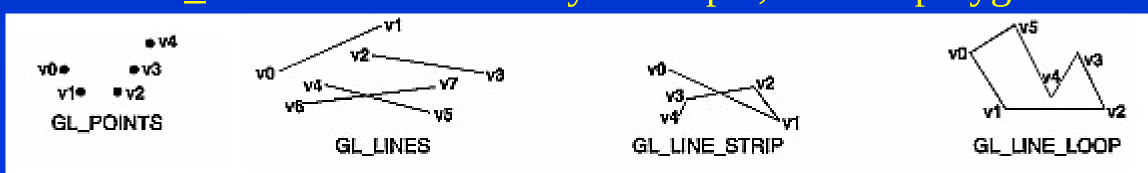
glColor3f(1,0,0); ← Specify appearance

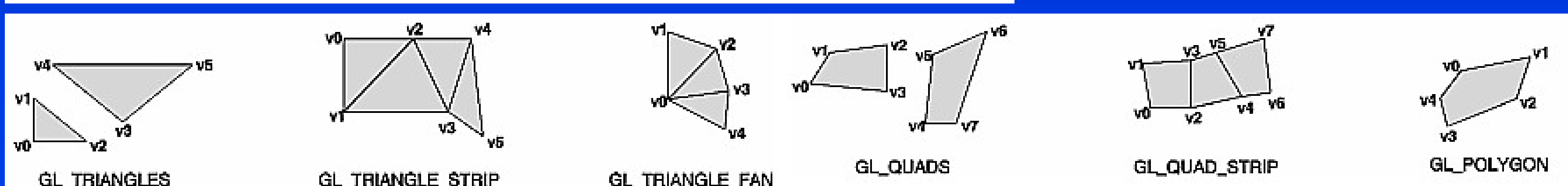
glVertex3f(0,1.5,-2); ← Specify vertices

glEnd(void); ← End OpenGL drawing

Geometric Primitives Names

- GL_POINTS: individual points
- GL_LINES: pairs of vertices interpreted as individual line segments
- GL_LINE_STRIP: series of connected line segments
- GL_LINE_LOOP: similar to above, with a segment added between last and first vertices
- GL_TRIANGLES: triples of vertices interpreted as triangles.
- GL_TRIANGLE_STRIP: linked strip of triangles.
- GL_TRIANGLE_FAN: linked fan of triangles.
- GL_QUADS: quadruples of vertices interpreted as four-sided polygons
- GL_QUAD_STRIP: linked strip of quadrilaterals
- GL_POLYGON: boundary of simple, convex polygon

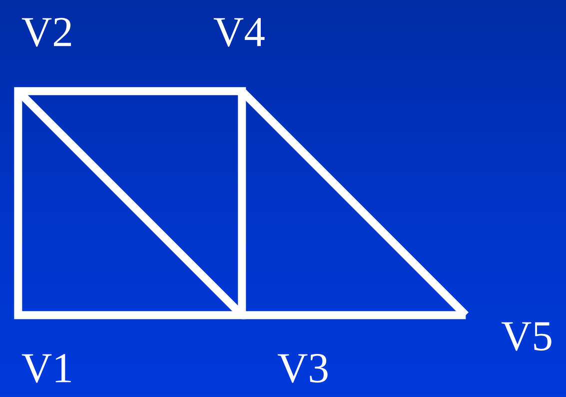




OpenGL Primitives

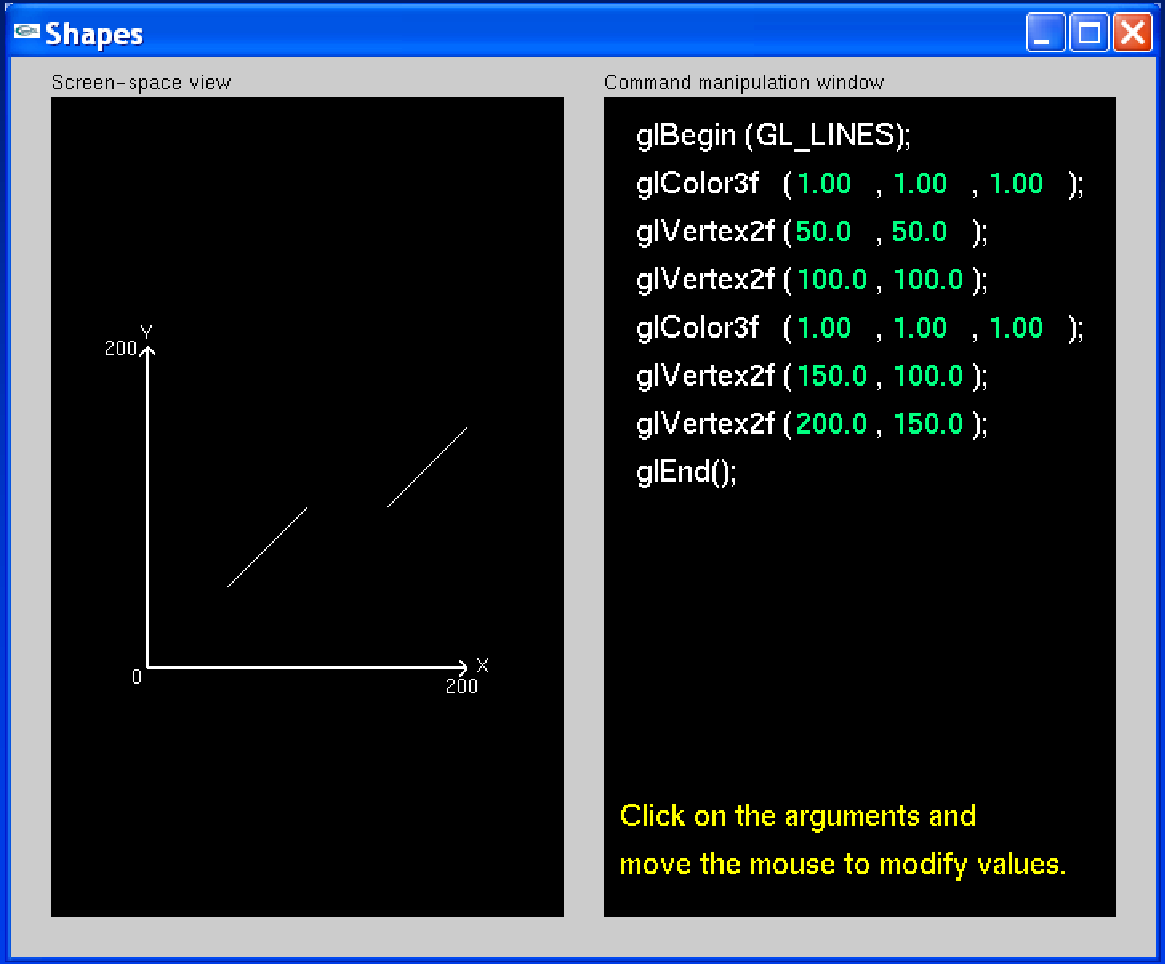
- **Example**

```
glBegin(GL_TRIANGLE_STRIP);
        glColor3f(1,1,1);  // color
        glVertex2f(0,0);   // v1
        glVertex2f(0,1);   // v2
        glVertex2f(1,0);   // v3
        glVertex2f(1,1);   // v4
        glVertex2f(2,0);   // v5
glEnd();
```



OpenGL Primitives

- Demo



OpenGL Geometric Processing

- Viewing: specify the view point (camera)
  - gluLookAt
- Modeling: place the models
  - glTranslate, glRotate
- Projection: set the lens
  - gluPerspective, gluOrtho2D
- Viewport: set the size of the photos
  - gluViewport

OpenGL Geometric Processing

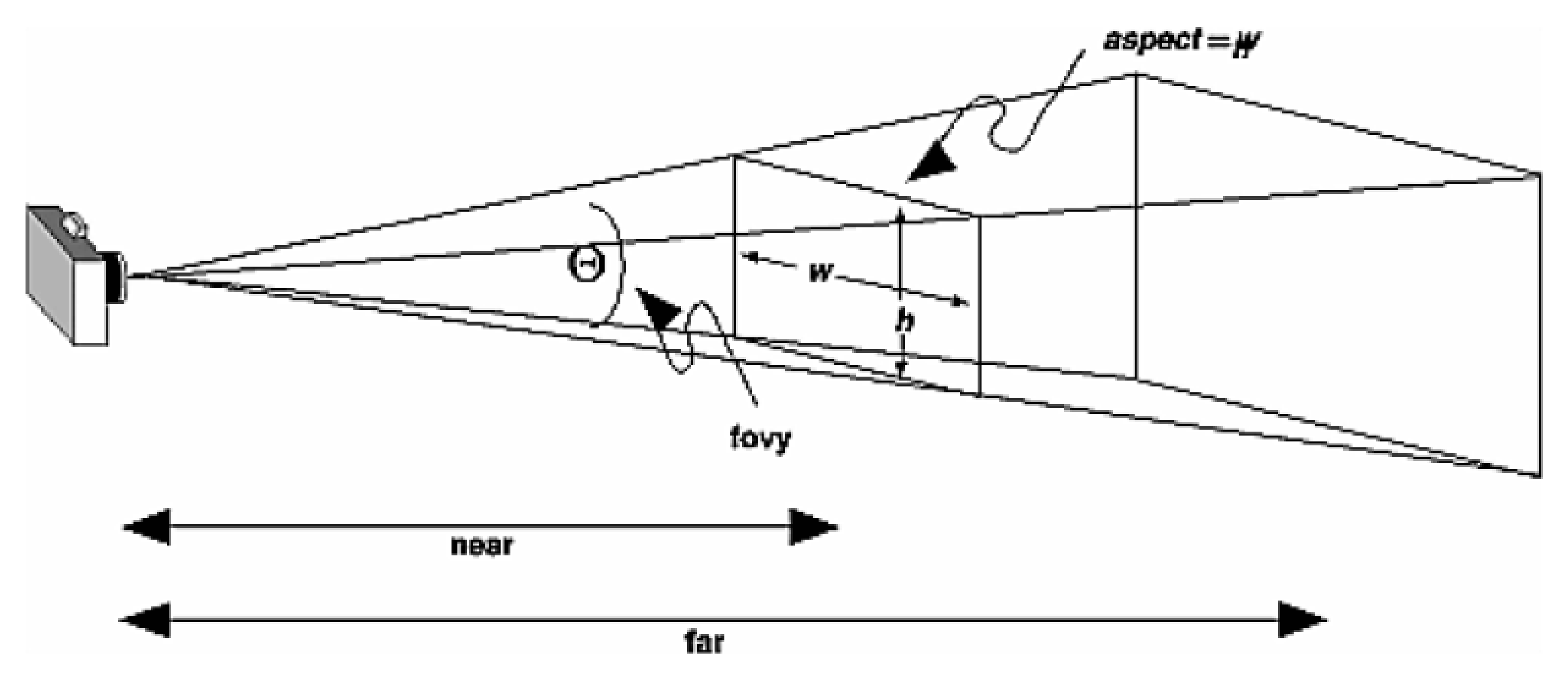
- Place the camera
  - gluLookAt(eye_x, eye_y, eye_z, // view point
    cen_x, cen_y, cen_z, // center point
    up_x, up_y, up_z); // up vector

OpenGL Geometric Processing

- Set the lens
  - gluPerspective (fovy, // view angle in degrees

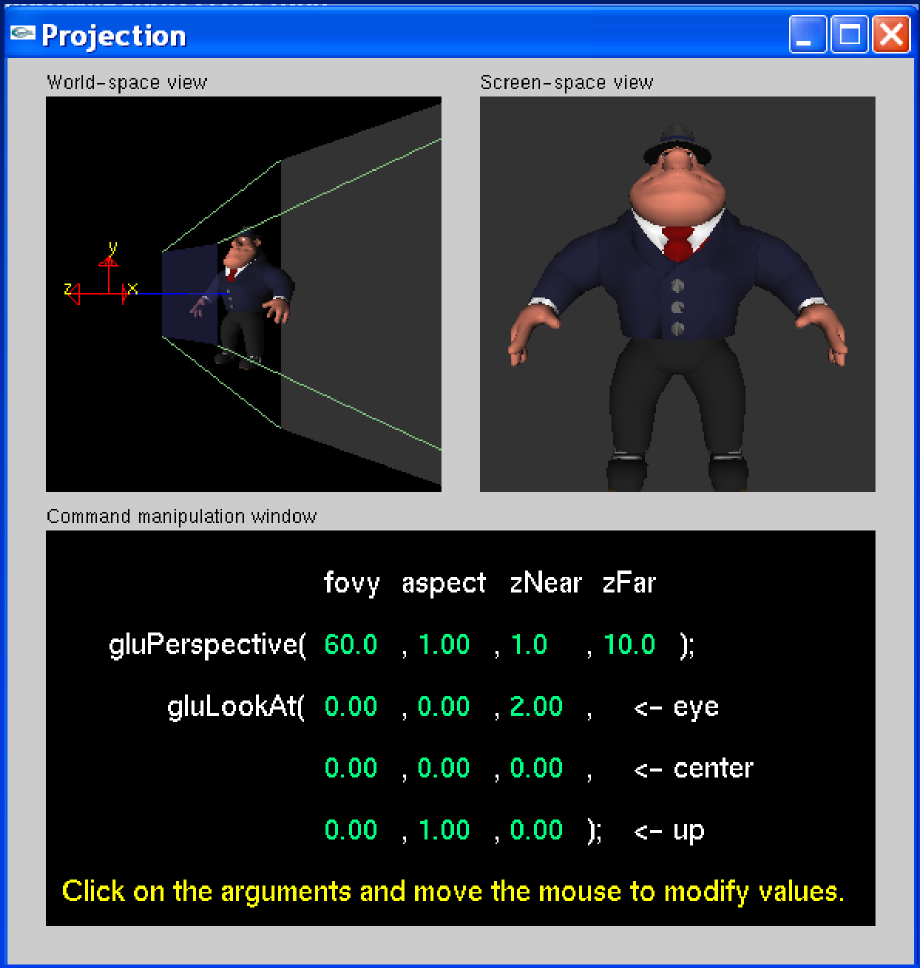
    aspect, // aspect ratio of x (width) to y (height)

    zNear, zFar); // near and far clipping plane



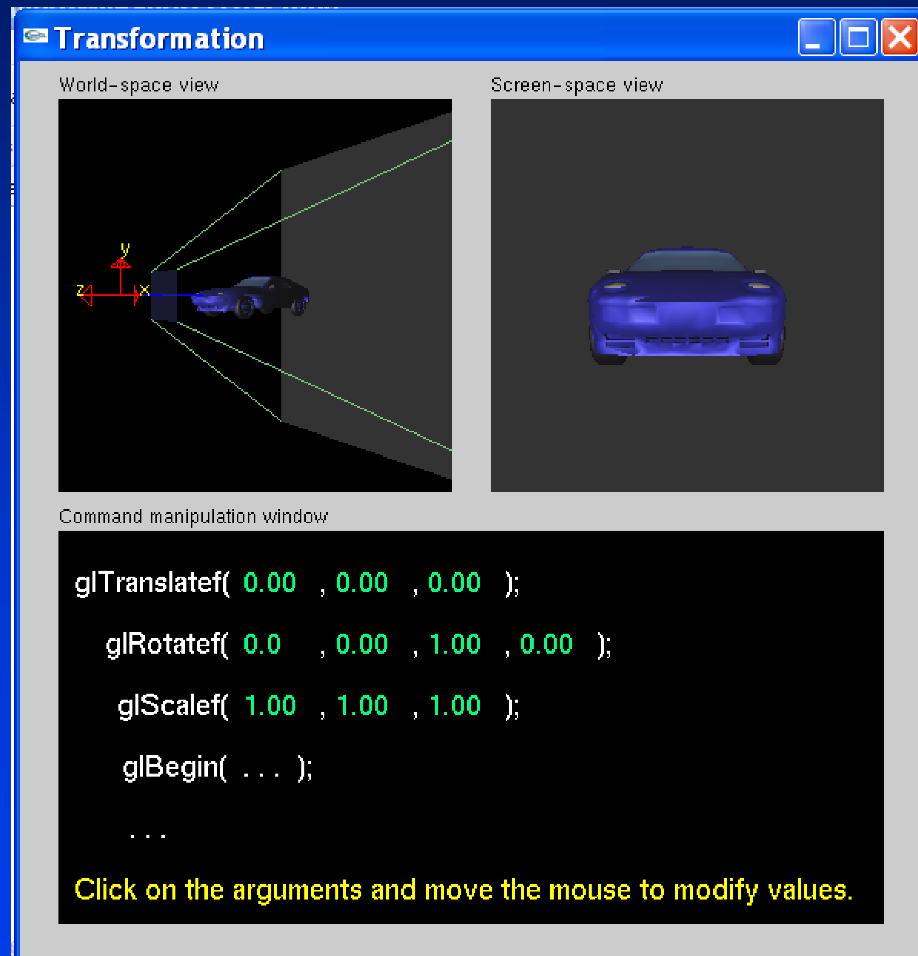
OpenGL Geometric Processing

- Demo



OpenGL Geometric Processing

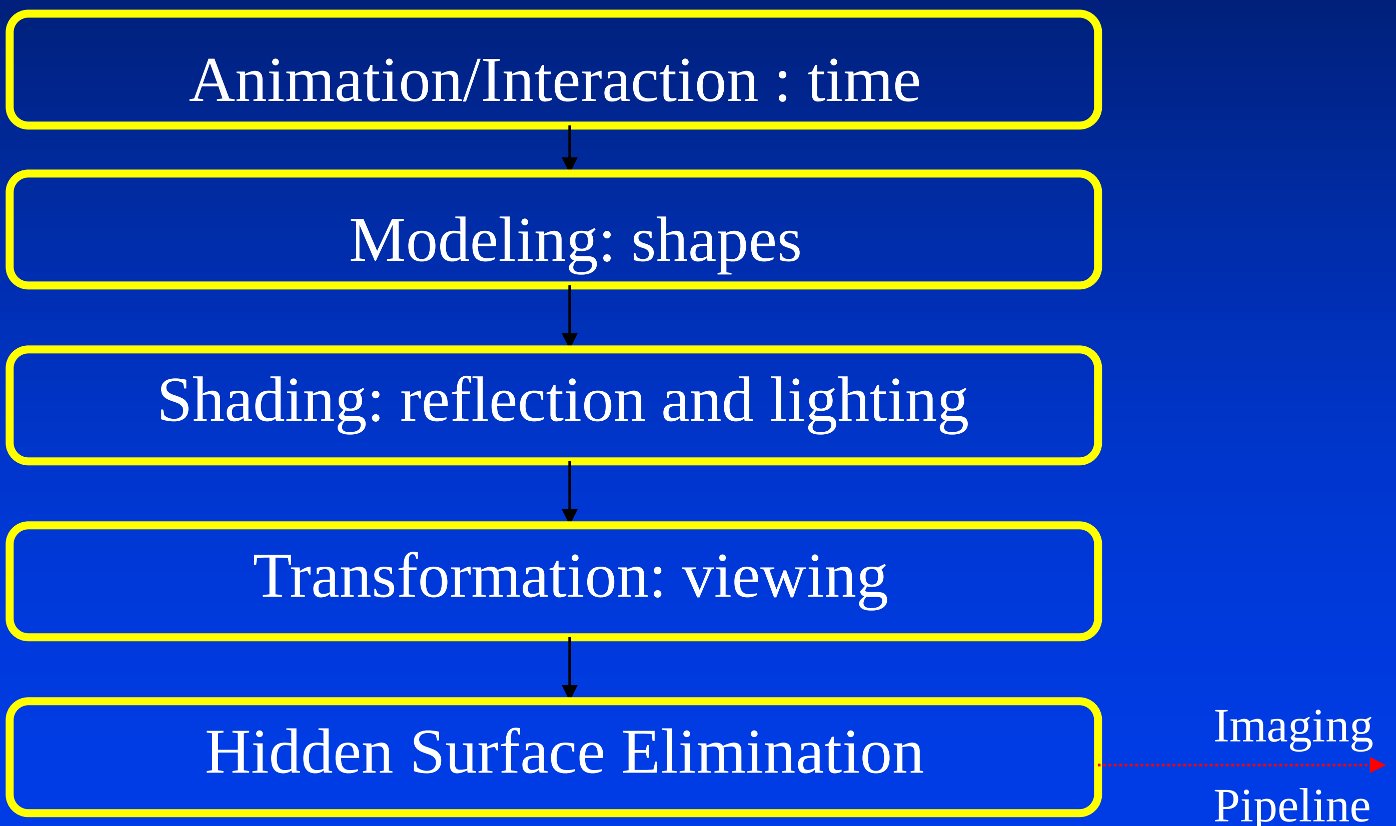
- Demo



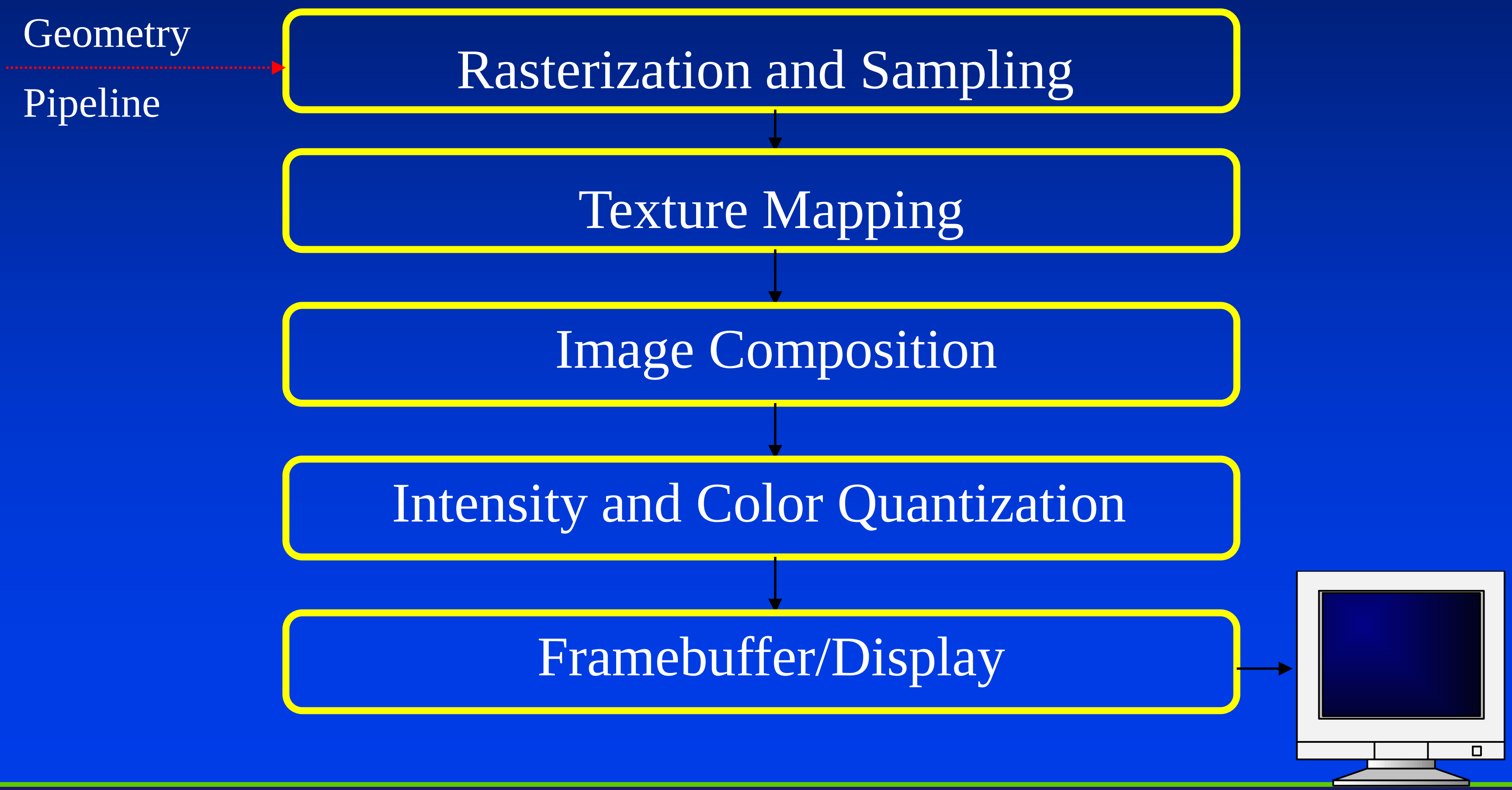
Graphics Pipelines

- Graphics processes generally execute sequentially
- Typical ‘pipeline’ model
- There are two ‘graphics’ pipelines
  – The Geometry or 3D pipeline
  – The Imaging or 2D pipeline

Geometry Pipeline

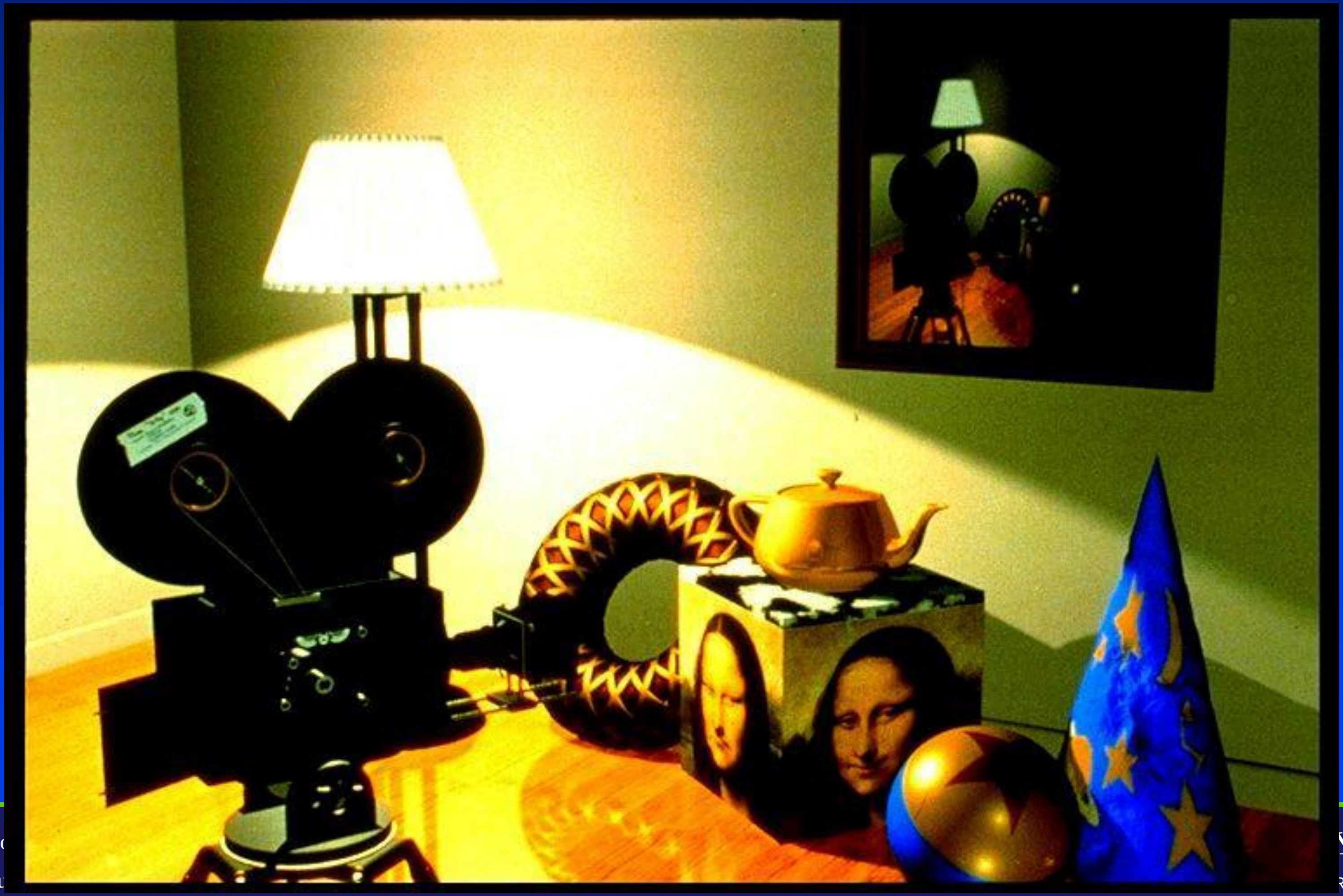


Imaging Pipeline

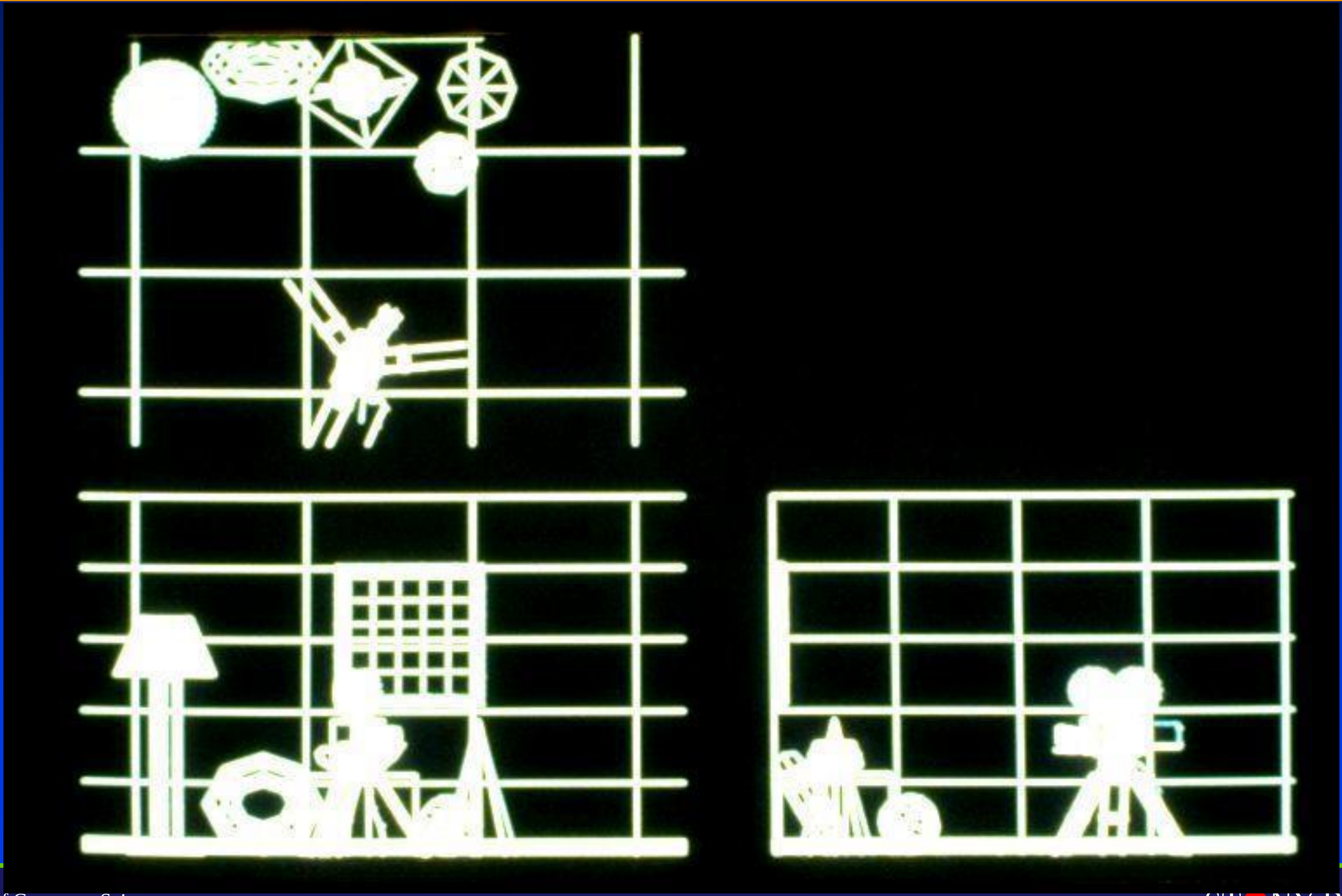


An Example through the Pipeline…

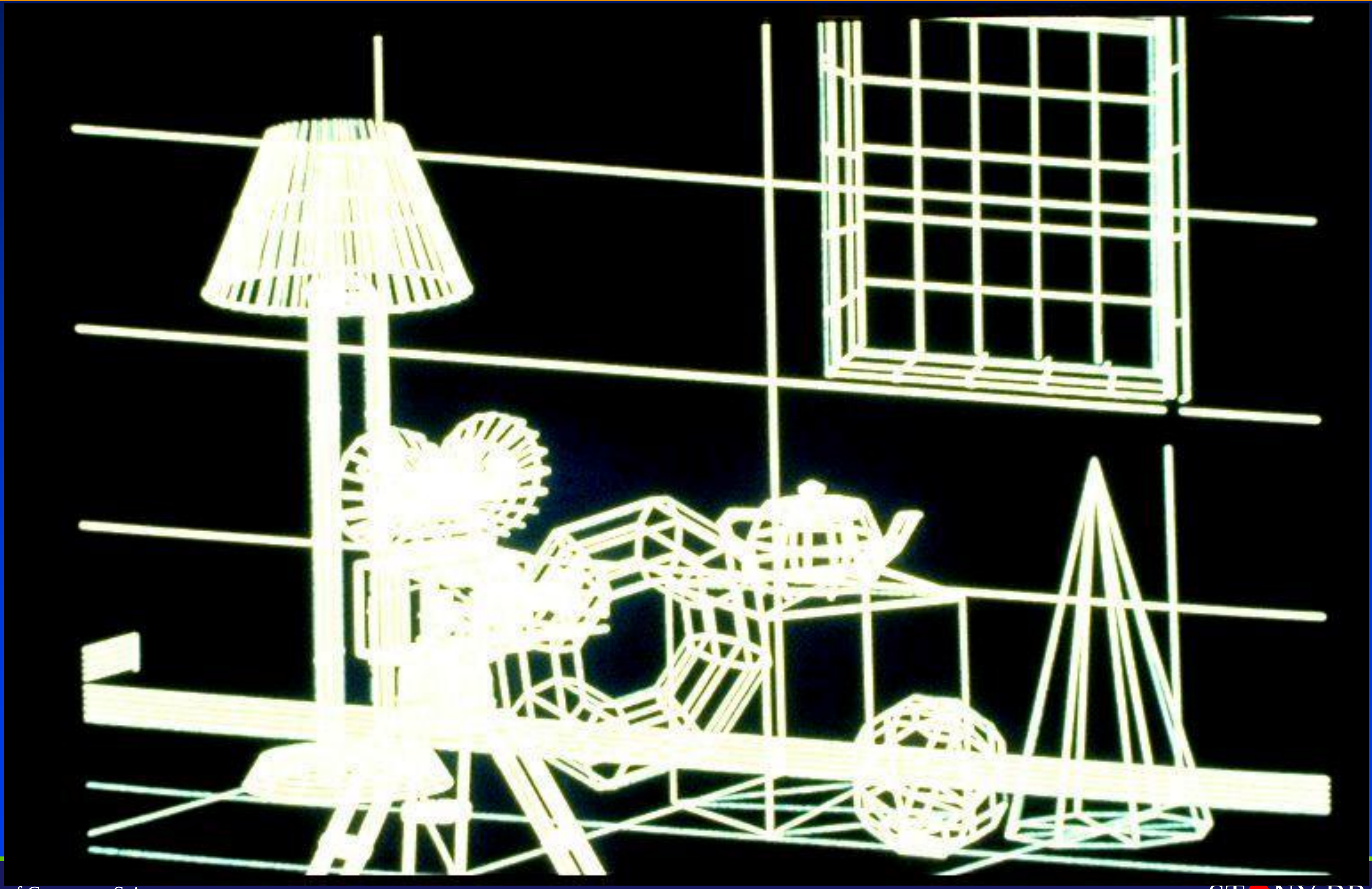
The scene we are trying to represent:



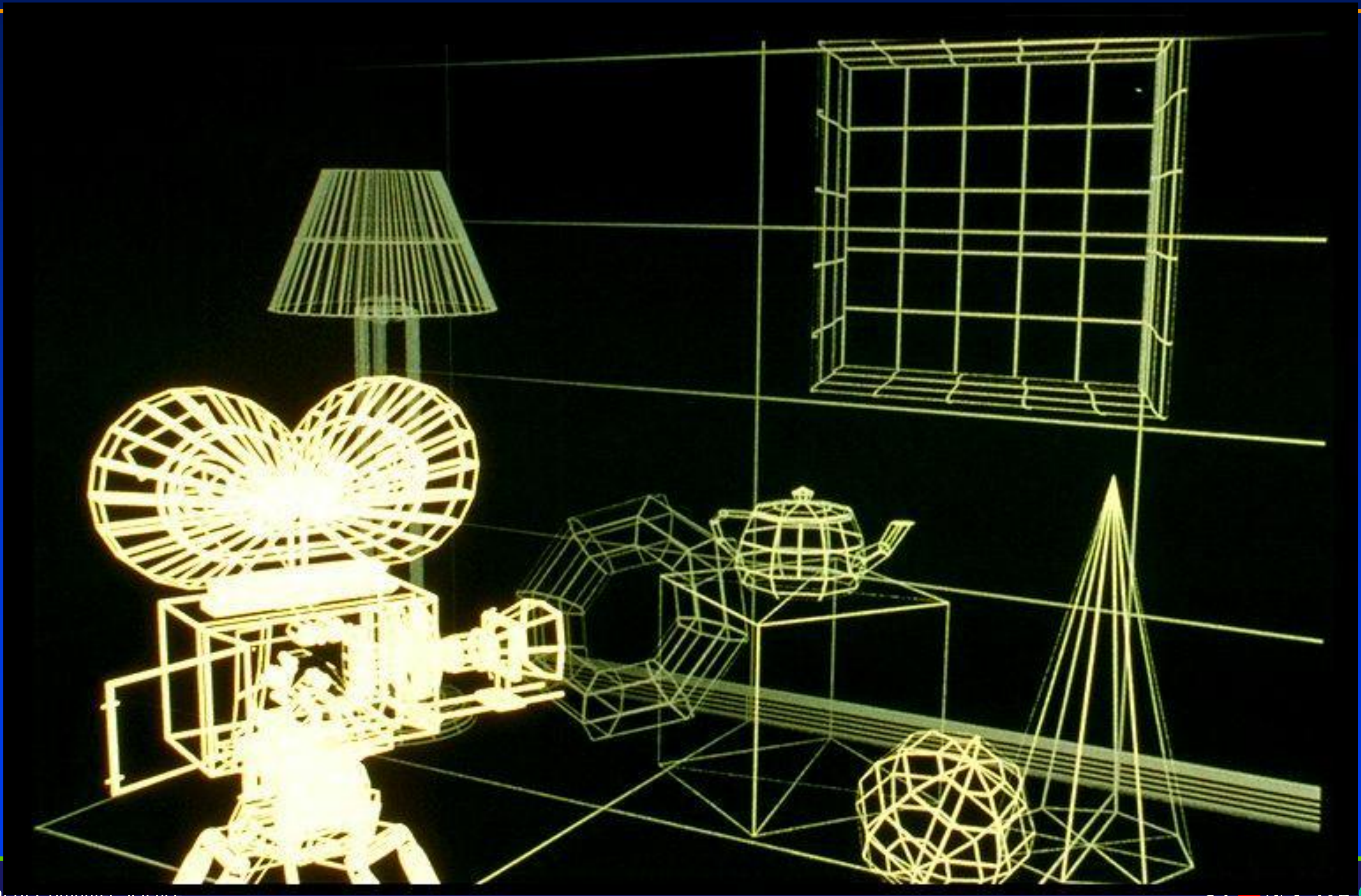
Wireframe Model – Orthographic Views



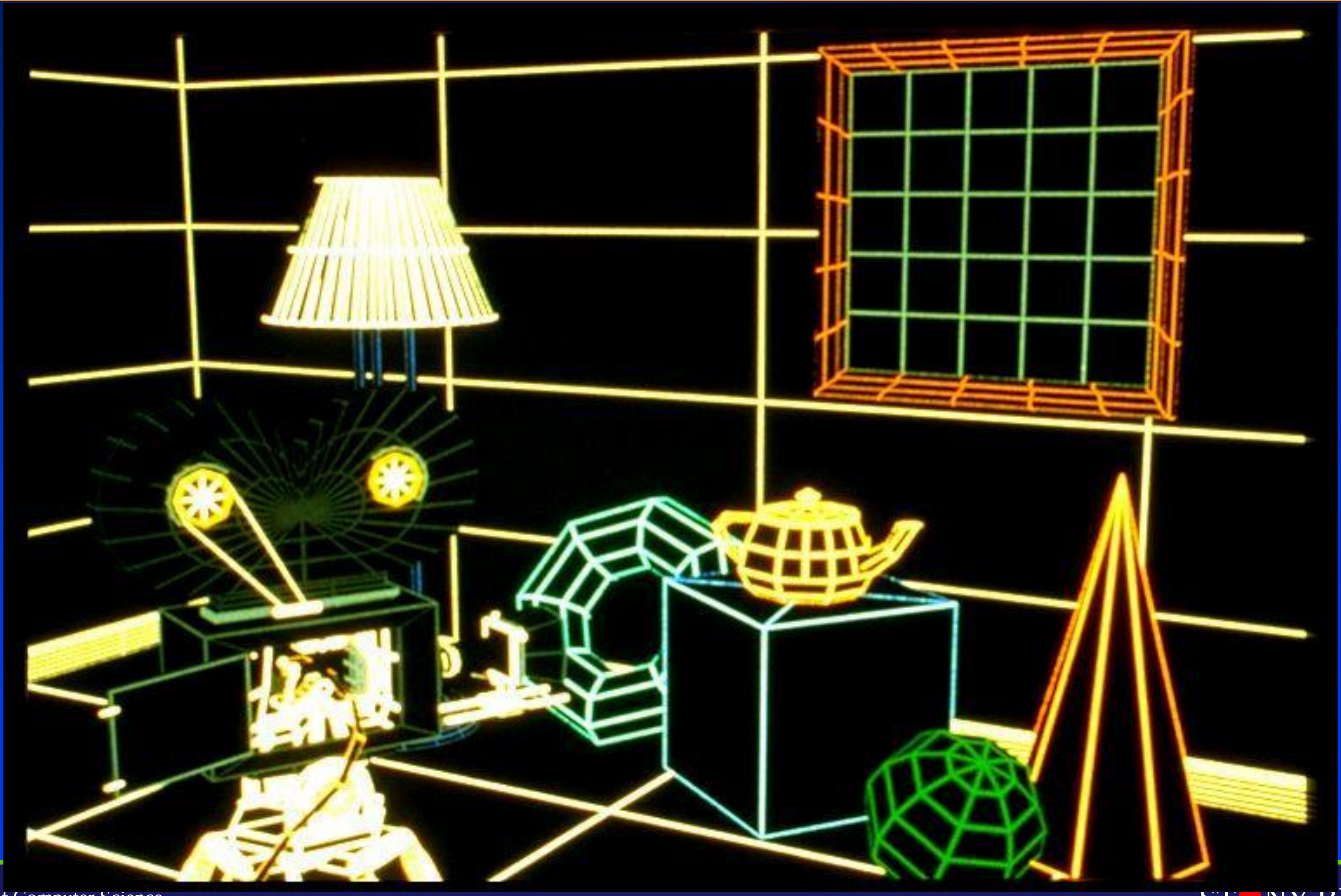
Perspective View



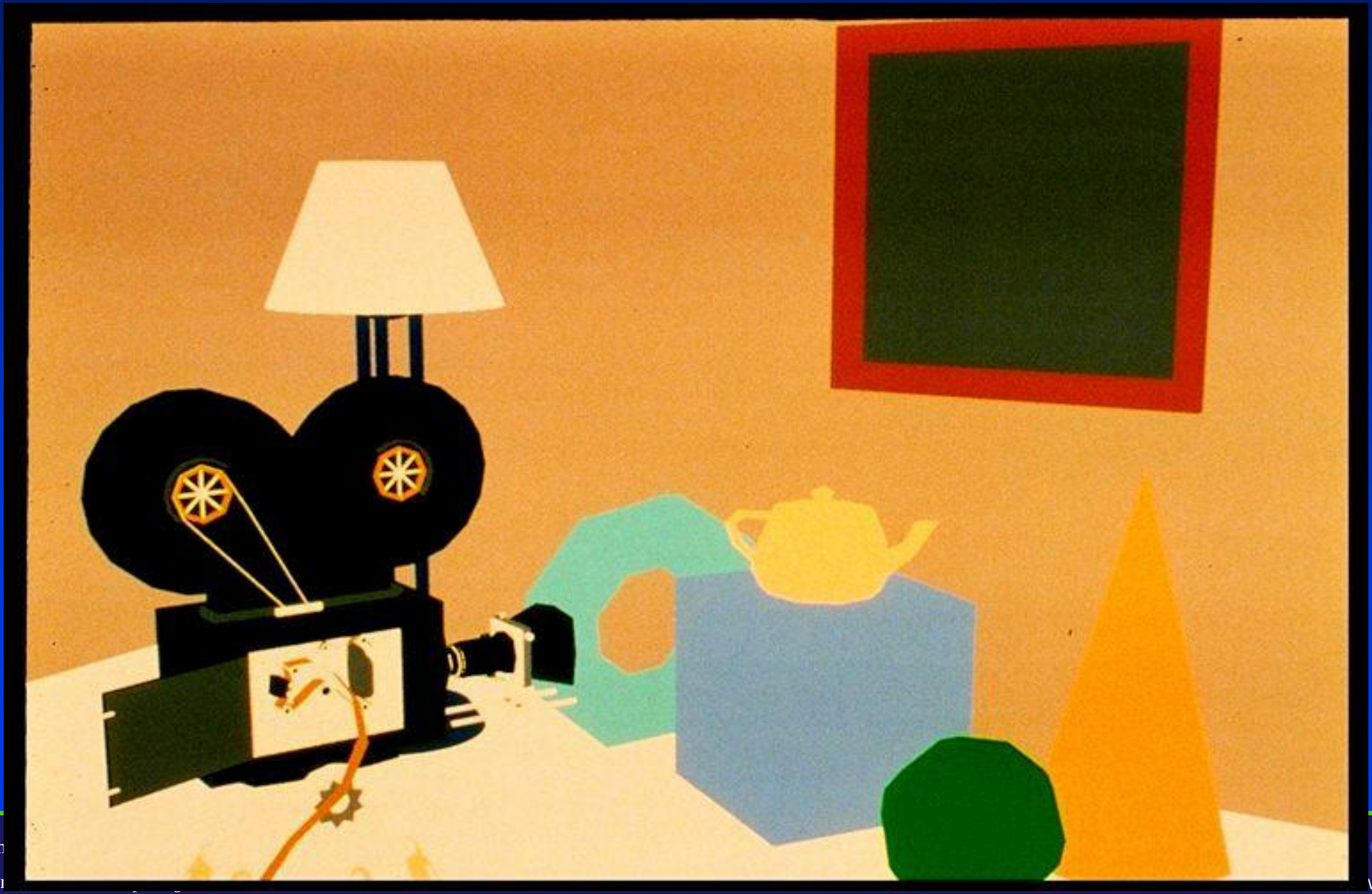
Depth Cue



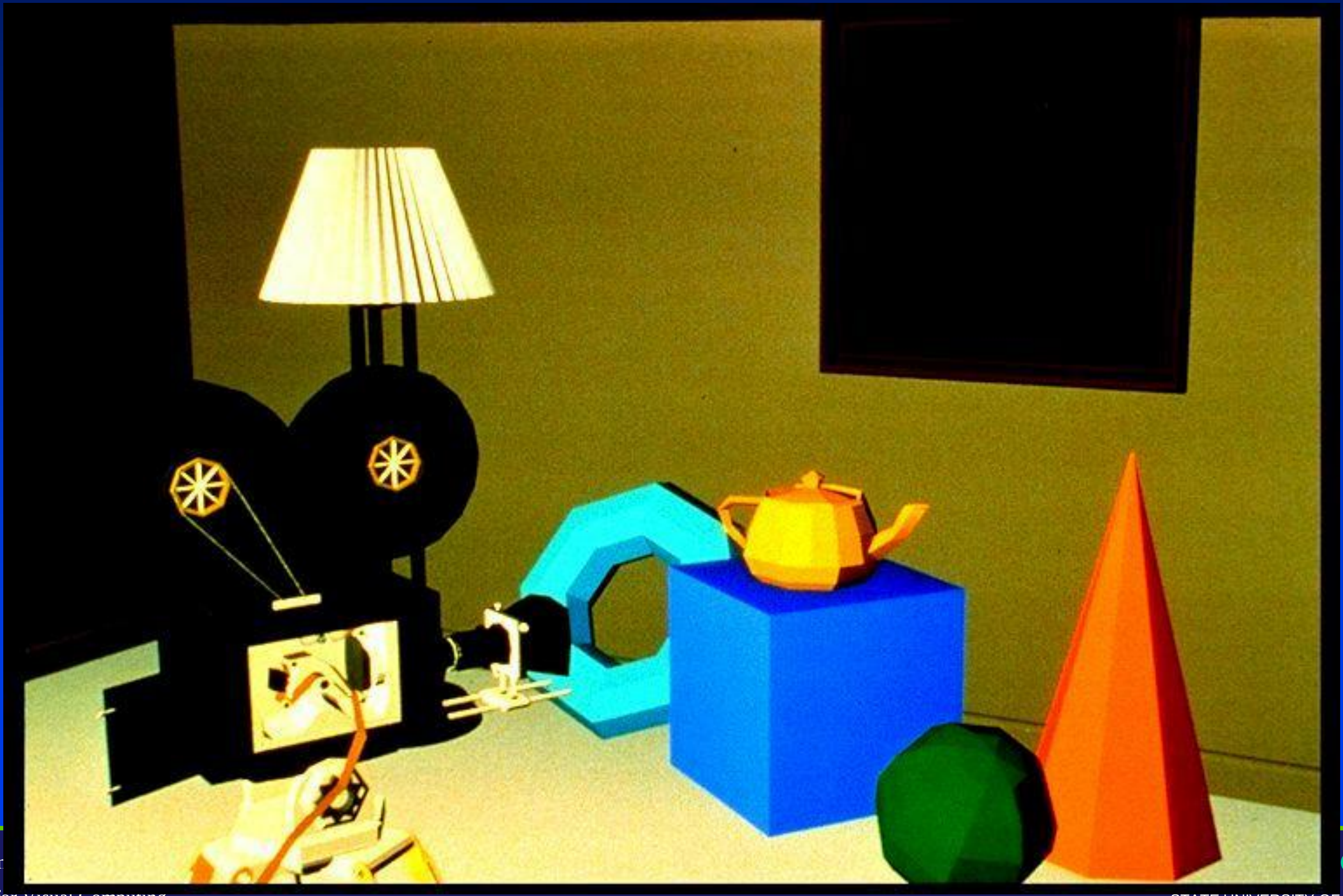
Hidden Line Removal – Add Color



Constant Shading - Ambient



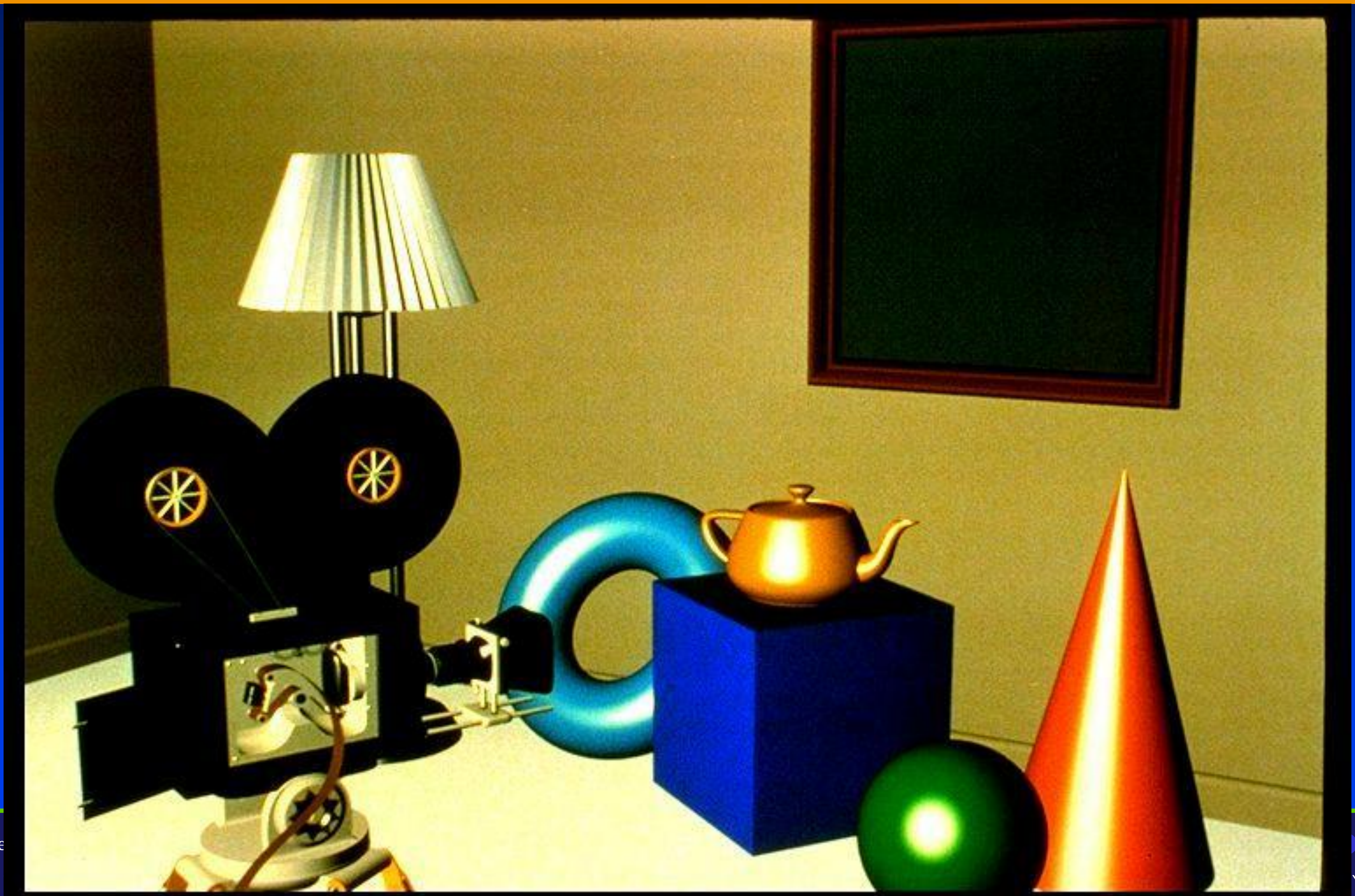
Faceted Shading - Flat



Gouraud Shading, No Specular Highlights

Specular Highlights

Phong Shading



Texture Mapping

Texture Mapping

Reflections, Shadows & Bump mapping

OpenGL Reference Books

1. OpenGL Programming Guide, 4th Edition: The Official Guide to Learning OpenGL, Version 1.4, Addison-Wesley, 2004.
2. OpenGL Reference Manual, 4th Edition: The Official Reference Document to OpenGL, Version 1.4, Addison-Wesley, 2004.

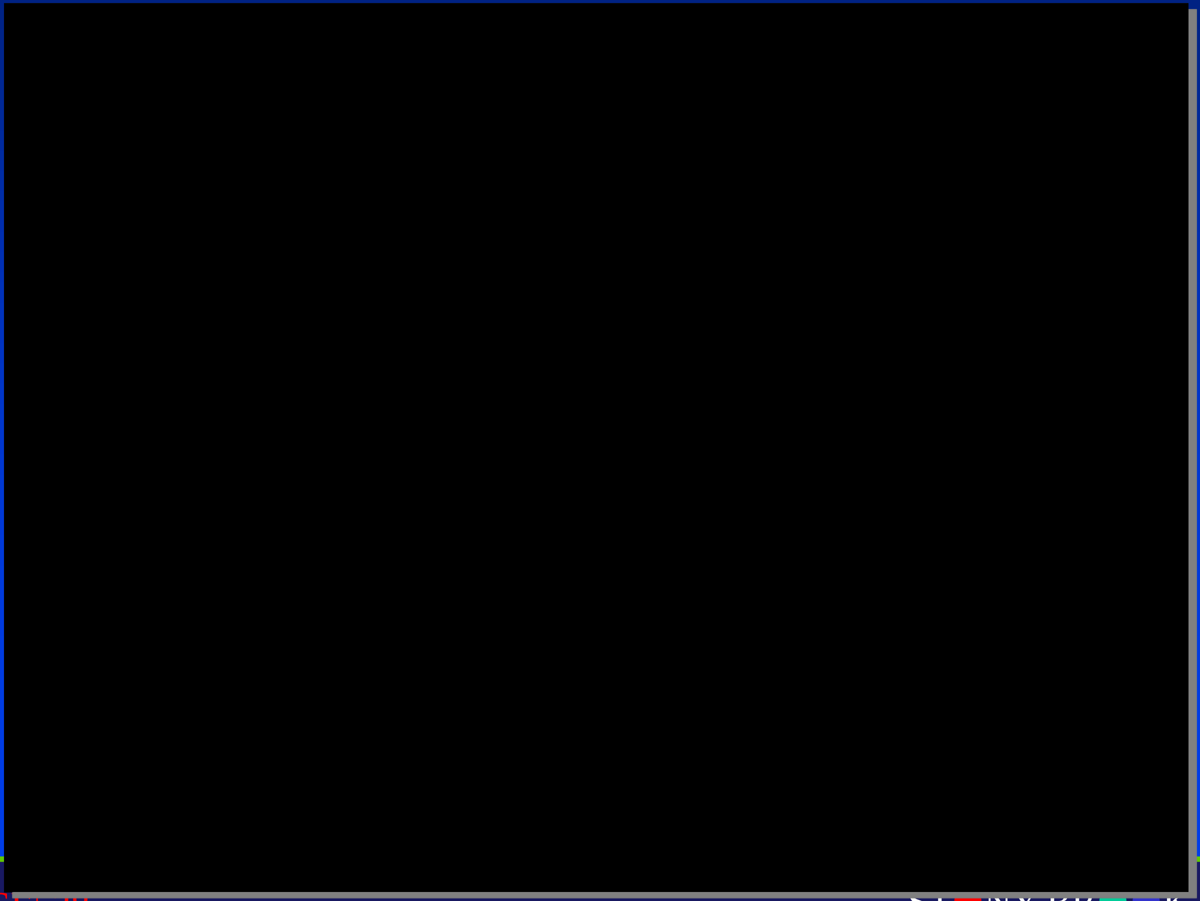
Advanced Topics

- Geometric Modeling & Processing
  - Editing & deformation
    - Interactive
    - Intuitive
    - Natural
  - Variety of tools
    - Boolean
  - User interface
    - 2D sketch
  - Other topics
    - Reconstruction
    - Parameterization
    - ….

Courtesy of Microsoft Research Asia

Advanced Topics

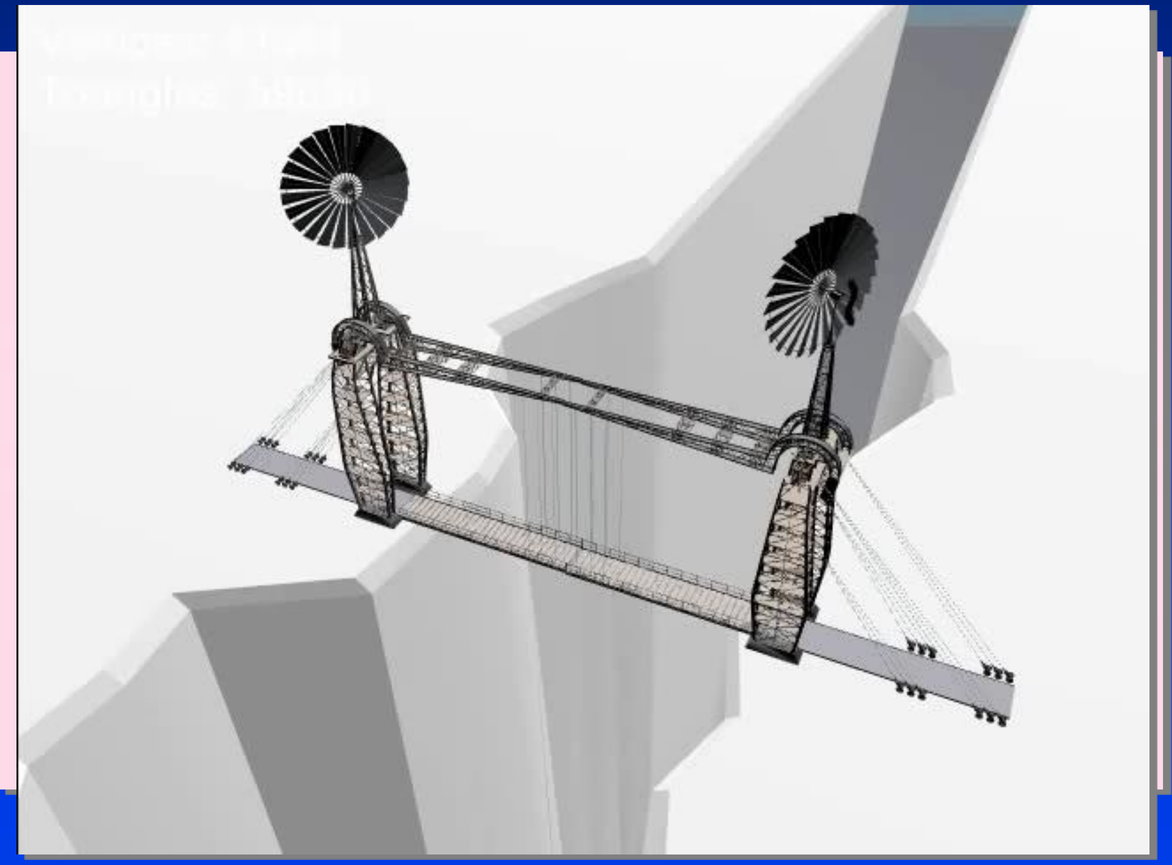
- Computer Animation & Simulation
  - Solving PDEs
  - Speed vs. accuracy
  - Physics/semi-physics
  - Numerical stability
  - Solid
    - Linear: fast, distortion
    - Nonlinear: slow, accurate
  - Fracture
    - Connectivity
    - Topology
  - Fluid



Courtesy of Muller, O'brien, Fedkiw et al.

Advanced Topics

- Human-Computer Interaction, Virtual Reality
  - Dynamic manipulation
  - Computational power
  - Low-end devices



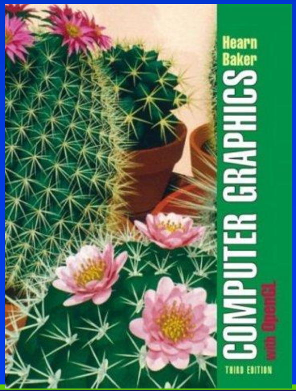
Courtesy of James, Igarashi et al.

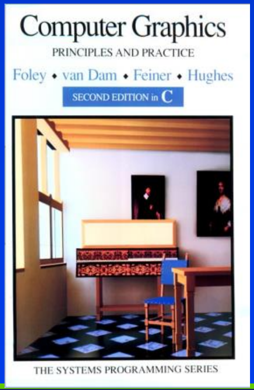
Other Advanced Topics

- Programmable graphics hardware
- Visualization
- Medical Imaging
- Non-photorealistic rendering
- Image-based rendering
- ….
- Each topic can be a course of its own!!!

Graphics Textbooks

- If you want to study computer graphics seriously:
- *Computer Graphics with OpenGL*, 3rd Edition, Donald Hearn and M. Pauline Baker, Prentice Hall, 2004.
- *Computer Graphics: Principles and Practice*, 2nd edition, Foley, van Dam, Feiner, and Hughes, Addison-Wesley Professional, 1995
- Many other textbooks and/or reference books are available in bookstores…





Presentation Outline

- Modern Approach for Computer Graphics

What Are Our Ultimate Goals?

- A large variety of datasets (acquired via scanning devices, super-computer simulation, mathematical descriptions, etc.)
- A pipeline of data processing that consists of data modeling (reconstruction), representation, manipulation (rigid transformation or deformation), classification (segmentation), feature extraction, simulation, analysis, visual display, conversion, storage, etc.
- Visual information processing

What Are Our Ultimate Goals?

- Datasets that are huge, multi-dimensional, time-evolving, unstructured, multi-attributes (geometric info. + material distributions), scattered (both temporal and spatial)…
- We are investigating mathematical tools and computational techniques for data modeling, reconstruction, manipulation, simulation, analysis, and display

Challenges

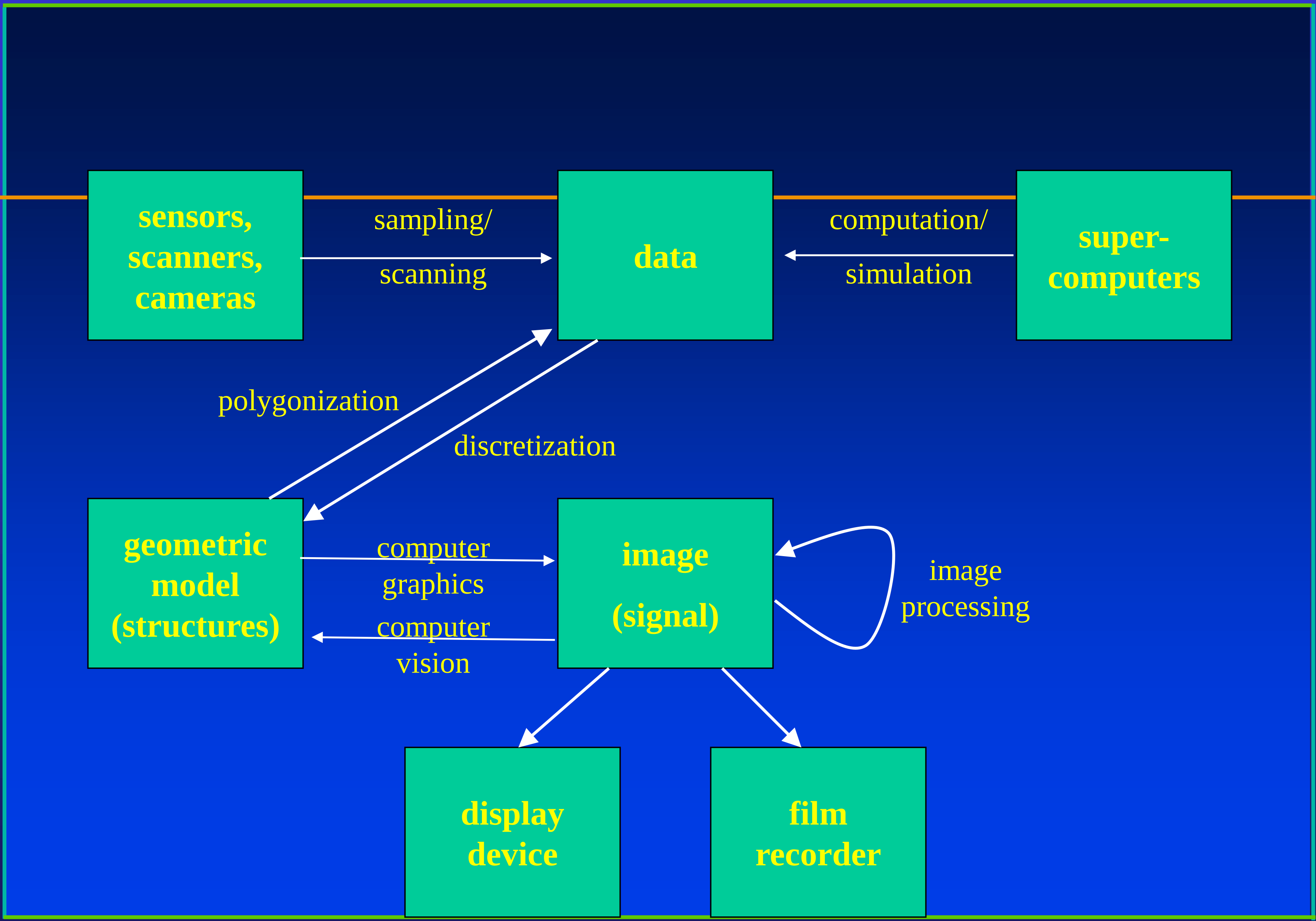
- TOO MUCH data
- The number of data sources keeps increasing
- Sensor quality and resolution are increasing
- Existing instruments are still available
- The speed of supercomputer is faster than ever
- We must do something (besides collecting and storing the datasets)
- We must deal with the huge datasets effectively
- Visual communication, improve our visual interaction with data

Challenges

- Data-driving, scientific computing to steer calculations
- Real-time interaction with computer and data experimentation
- Drive and gain insight into the scientific discovery process

Related Fields

- Computer graphics (image synthesis)
  - Generate images from complex multivariate datasets
- Image processing, signal processing
- Image understanding (pattern recognition)
  - Interpret image data
- Computational vision
- Human-computer interaction
  - Mechanisms to communicate, use, perceive visual information
- Computer-aided design
- Neurological/physiological studies on human brain and our visual system


sensors, scanners, cameras
sampling/
scanning
data
computation/
simulation
super-computers
polygonization
discretization
geometric model (structures)
computer graphics
computer vision
image (signal)
image processing
display device
film recorder

Computer Graphics Pipeline

- Data acquisition and representation
- Modeling data and their (time-varying) behaviors (e.g., physical experiments or computational simulations)
- Graphics system for data rendering
- Image-based techniques

Data Sources

- Scanned, computed, modeled data
- The first process is data-gathering
- Large variety of data sources
- Extremely large-scale datasets

Data Acquisition and Processing

- Pixels and voxels
- Regular & irregular grids
- Numerical simulations
- Surface or volumetric data
- Scalar, vector, tensor data with multiple attributes
- Higher-dimensional and/or time-varying data
- Popular techniques
  - Contouring, iso-surfaces, triangulation, marching cubes, slicing, segmentation, volume rendering, reconstruction
- Image-based processing techniques
  - Sampling, filtering, anti-aliasing, image analysis and manipulation

Information Domain

- Sciences (e.g., statistics, physics)
- Engineering (e.g., empirical observations for quality control)
- Social events (e.g., population census)
- Economic activities (e.g., stock trading)
- Medicine (e.g., computed tomograph (CT), magnetic resonance imaging (MRI), X-rays, ultrasound, various imaging modalities)
- Geology

Information Domain

- Biology (e.g., electronic microscopes, DNA sequences, molecular models, drug design)
- Computer-based simulations (e.g., computational fluid dynamics, differential equation solver, finite element analysis)
- Satellite data (e.g., earth resource, military intelligence, weather and atmospheric data)
- Spacecraft data (e.g., planetary data)
- Radio telescope, atmospheric radar, ocean sonar, etc.
- Instrumental devices recording geophysical and seismic activities (e.g., earthquake)

Graphics and Visualization

- Data acquisition, representation, and modeling
- Imaging processing
- Visualization (displaying) methods and algorithms
- More advanced research topics

Pathway to Success

- Highly-motivated
- Hard-working
- Start as soon as possible
- Communicate with the instructor on a regular basis
- Actively interact with your fellow students
- Visit university libraries frequently
- Read as many papers as possible

Computer Graphics

- "The purpose of scientific computing is insight, not numbers," by Richard Hamming many years ago
- These fields are all within computer science and engineering, yet computer graphics spans multi-disciplines
- Computer Graphics (another definition)
  - Application of computers to the disciplines of sciences/engineering

Computer Graphics

- Computer Graphics is application-driven, so what are its applications?

Applications

- Simulation and training: flight, driving
- Scientific visualization: weather, natural phenomena, physical process, chemical reaction, nuclear process
- Science: Mathematics, physics (differential equations) biology (molecular dynamics, structural biology)
- Environments sciences
- Engineering (computational fluid dynamics)
- Computer-aided design/manufacturing (CAD/CAM): architecture, mechanical part, electrical design (VLSI)

Applications

- Art and Entertainment, animation, commercial advertising, movies, games, and video
- Education, and graphical presentation
- Medicine: 3D medical imaging and analysis
- Financial world
- Law
- WWW: graphical design and e-commerce
- Communications, interface, interaction
- Military
- Others: geographic information system, graphical user interfaces, image and geometric databases, virtual reality, etc.

Conclusions

- Bigger picture about Computer Graphics
  - Animation, computer-aided design, medical application, entertainment, and other applications relevant to Computer Graphics
  - Key components for undergraduates
  - Advanced topics for senior undergraduates, and graduate research
- Graphics rendering pipeline
  - Geometric modeling
  - Modeling/viewing transformation
  - Rasterization & Display
- Programming basics
  - OpenGL

Key Components

- Modeling: representation choices of different models
- Rendering: simulating light and shadow, camera control, visibility, discretization of models
- HCI (human-computer interface): specialized I/O devices, graphical user interfaces
- Animation: lifelike characters, natural phenomena, surrounding environments

Questions?

Thank You!