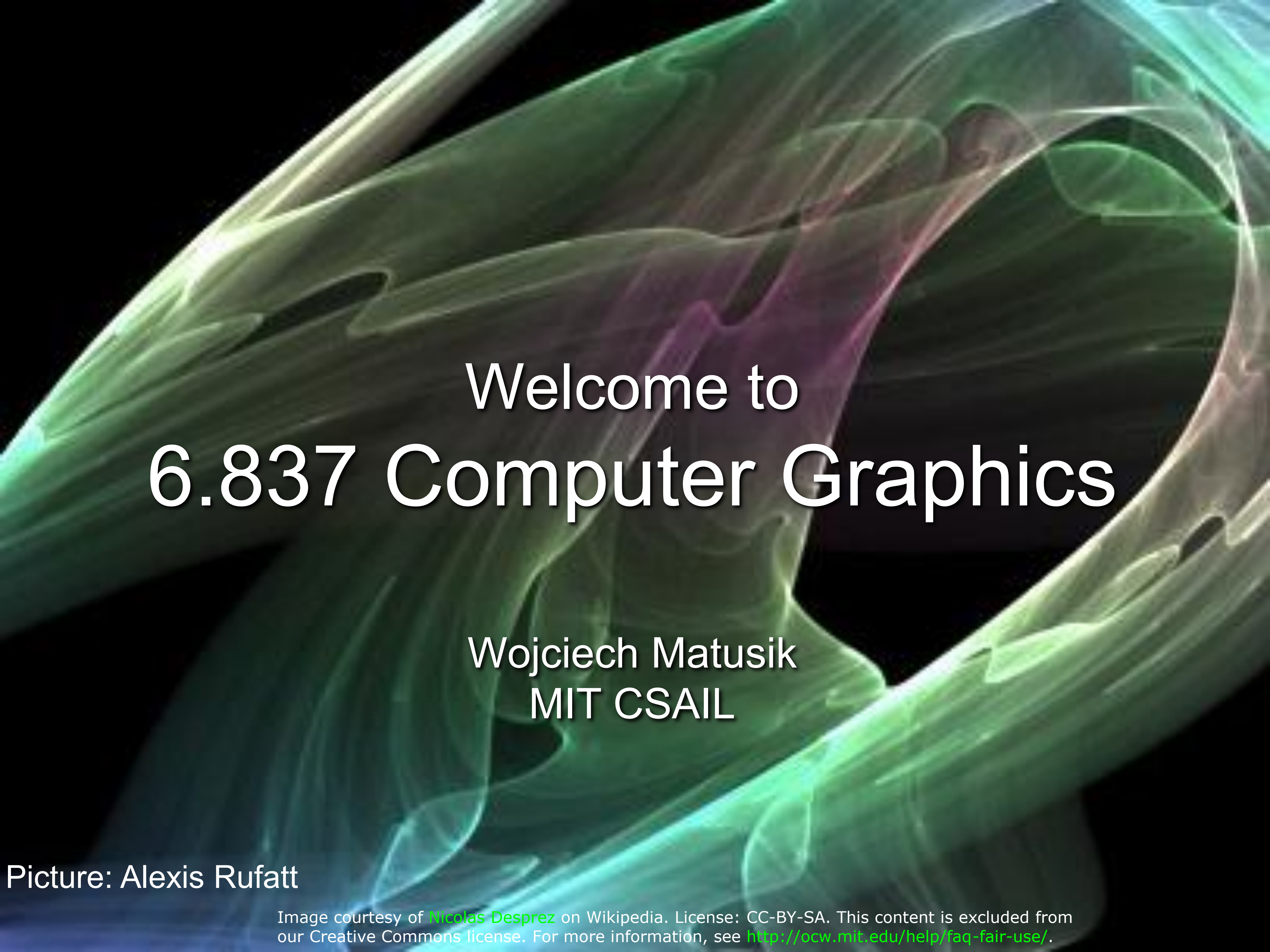


Welcome to
6.837 Computer Graphics

Wojciech Matusik
MIT CSAIL

Picture: Alexis Rufatt

Luxo Jr.

- Pixar Animation Studios, 1986
- Director: John Lasseter

Image of Pixar's Luxo Jr. removed due to copyright restrictions -- please see http://www.computerhistory.org/atchm/pixars-luxo-jr/ for further details.

Plan

- Overview of computer graphics
- Administrivia
- Overview of the semester
- Overview of assignments
- Intro to OpenGL & assignment 0

What are the applications of graphics?

Movies/special effects

Images removed due to copyright restrictions.

More than you would expect

Video clip removed due to copyright restrictions -- please see http://vimeo.com/9553622 for further details.

Video Games

Images removed due to copyright restrictions.

Simulation

Images removed due to copyright restrictions.

CAD-CAM & Design

Images removed due to copyright restrictions.

Architecture

Images removed due to copyright restrictions.

Video removed due to copyright restrictions -- please see "The Light of Mies van der Rohe" for further details, available at http://graphics.ucsd.edu/~henrik/animations/.

Virtual Reality

Images removed due to copyright restrictions.

Visualization

Images removed due to copyright restrictions.

Medical Imaging

Images removed due to copyright restrictions.

Education

Image removed due to copyright restrictions.

Geographic Info Systems & GPS

Image removed due to copyright restrictions.

Any display

- Computers go through OpenGL and DirectX to display anything
- 2D graphics, Illustrator, Flash, Fonts

Image removed due to copyright restrictions.

What do you expect to learn?

- And why?

What you will learn in 6.837

- Fundamentals of computer graphics algorithms
  - Will give a pretty good idea of how to implement lots of the things just shown
- We will concentrate on 3D, not 2D illustration or image processing
- Basics of real-time rendering and graphics hardware
- Basic OpenGL
  - Not the focus, though: Means, not the end.
- You will get C++ programming experience

What you will NOT learn in 6.837

- OpenGL and DirectX hacks
  - Most become obsolete every 18 months anyway!
  - Does not really matter either: Graphics is becoming all software again (OpenCL, Larrabee, etc.)
- Software packages
  - CAD-CAM, 3D Studio MAX, Maya
  - Photoshop and other painting tools
- Artistic skills
- Game design

How much Math?

- Lots of simple linear algebra
  - Get it right, it will help you a lot!
- Some more advanced concepts
  - Homogeneous coordinates
  - Ordinary differential equations (ODEs) and their numerical solution
  - Sampling, antialiasing (some gentle Fourier analysis)
  - Monte-Carlo integration
- Always in a concrete and visual context

Beyond computer graphics

- Many of the mathematical and algorithmic tools are useful in other engineering and scientific context
- Linear algebra
- Splines
- Differential equations
- Monte-Carlo integration
- …

Questions?

Plan

- Overview of computer graphics
- **Administrivia**
- Overview of the semester
- Overview of assignments
- Intro to OpenGL & assignment 0

Team

- Instructor
  - Wojciech Matusik

Administrivia: Website, Staff Email

- Course website
  - Announcements
  - Slides (posted soon after each lecture)
  - Assignments, both instructions and turn-in
- Message Board
- Staff Email
  - Reaches all of us, preferred method of communication

Administrivia: Grading Policy

- Assignments: 75%
  - Two-week programming assignments
  - Must be completed individually
  - *No final project*
- Quiz: 10%
  - in class
- Final Exam: 10%
  - TBA during finals week
- Participation: 5%

Administrivia: Prerequisites

- Not strictly enforced
- All assignments are in C++
  - *Optional review/introductory session*

- Calculus, Linear Algebra
  - Solving equations, derivatives, integral
  - vectors, matrices, basis, solving systems of equations
  - *Optional review/introductory session*

Administrivia: Assignments

- Turn in code and executable (Linux)
- Always turn in a README file
  - Describe problems, explain partially-working code
    Say how long the assignment took
- Coding style important
  - Some assignments are cumulative
- Collaboration policy:
  - You can chat, but code on your own
  - Acknowledge your collaboration! (in readme file)
- Late policy:
  - **The deadline is absolute: 0 if not on time**
  - Due Wednesday @ 8pm
  - Extensions only considered if requested 1 week before due date
  - Medical problems must be documented

The deadline is absolute

- I mean it.
- I do regularly give 0 for,
  - an assignment turned in half an hour late
  - turning in the wrong file
- Submit early, even before you might be fully done

Collaboration policy

- You can chat, but code on your own (we use automated plagiarism detection software!)
- Use Piazza message board
- Help others on Piazza message board (will help your grade!)
- Acknowledge your collaboration (in README)
- Talk to each other, get a community going
  - Graphics is fun!

Administrivia: Assignments

- **The assignments are a lot of work. Really.**
  - Start early!

Assignments

0: Warm up (mesh display with OpenGL)

1: Curves & surfaces

2: Hierarchical modeling, skinning

3: Physically-based simulation

4: Ray casting

5: Ray tracing

(more in later slides)

Textbooks

- No textbook is required
- Recommendations
  - 3D Computer Graphics (Watt)
  - 3D Computer Graphics: A Mathematical Introduction with OpenGL (Buss)
    - **There is a free online version** available from Books24x7
  - Real-Time Rendering, 3rd ed. (Akenine-Möller, Haines, Hoffman)
  - Fundamentals of Computer Graphics, 3rd ed. (Shirley, Marschner)

Questions?

Plan

- Overview of computer graphics
- Administrivia
- **Overview of the semester**
- Overview of assignments
- Intro to OpenGL & assignment 0

How do you make this picture?

- Modeling
  - Geometry
  - Materials
  - Lights
- Animation
  - Make it move
- Rendering
  - I.e., draw the picture!
  - Lighting, shadows, textures…

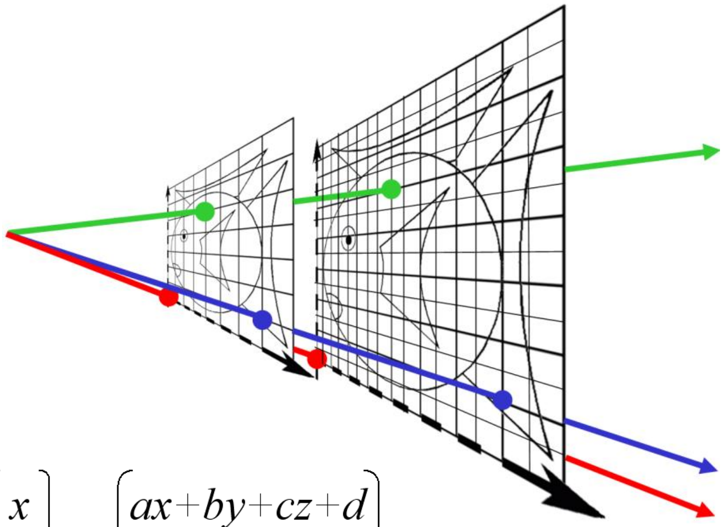
Questions?

Overview of the Semester

- Modeling, Transformations
- Animation, Color
- Ray Casting / Ray Tracing
- The Graphics Pipeline
- Textures, Shadows
- Sampling, Global Illumination

Transformations

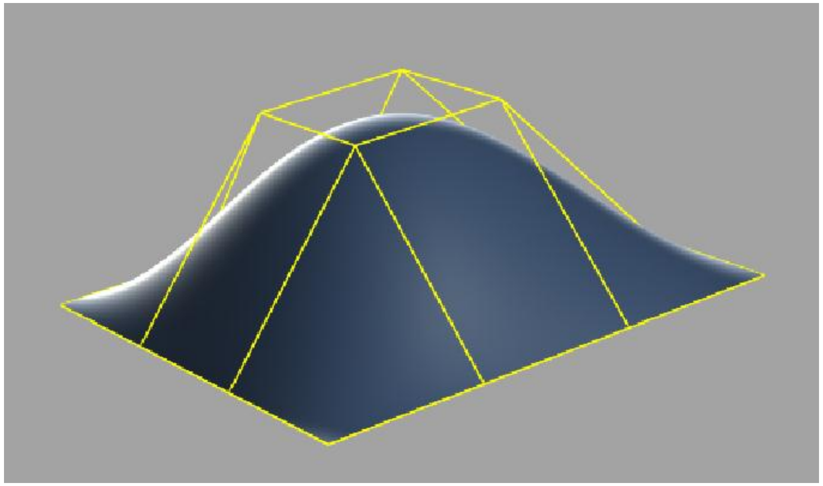
- Yep, good old linear algebra
- Homogeneous coordinates
  - (Adding dimensions to make life harder)
- Perspective

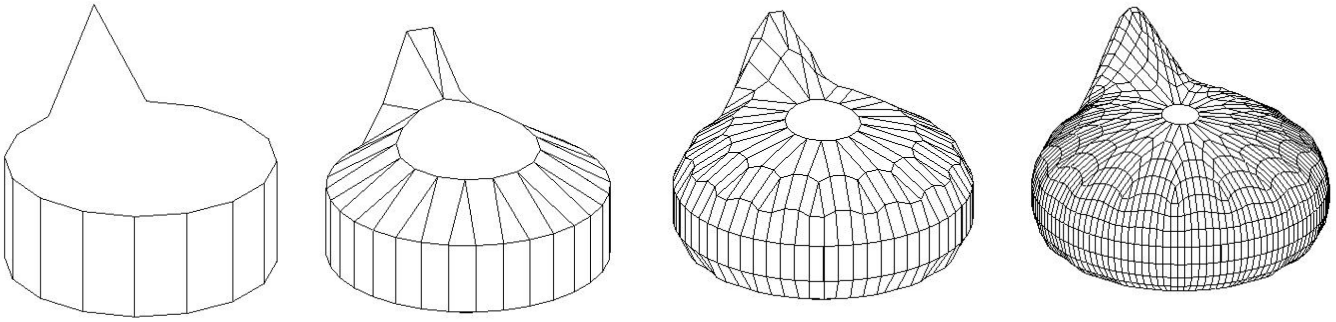


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} ax+by+cz+d \\ ex+fy+gz+h \\ ix+jy+kz+l \\ 1 \end{pmatrix}$$

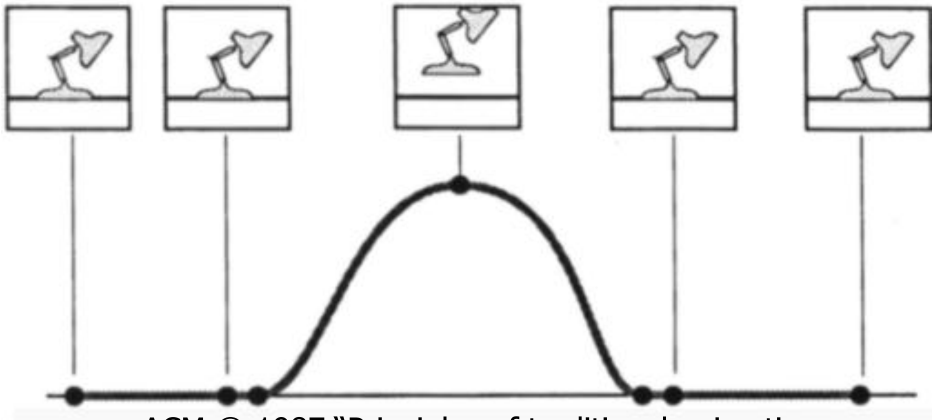
Modeling

- Curves and surfaces
- Subdivision surfaces





Animation: Keyframing



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

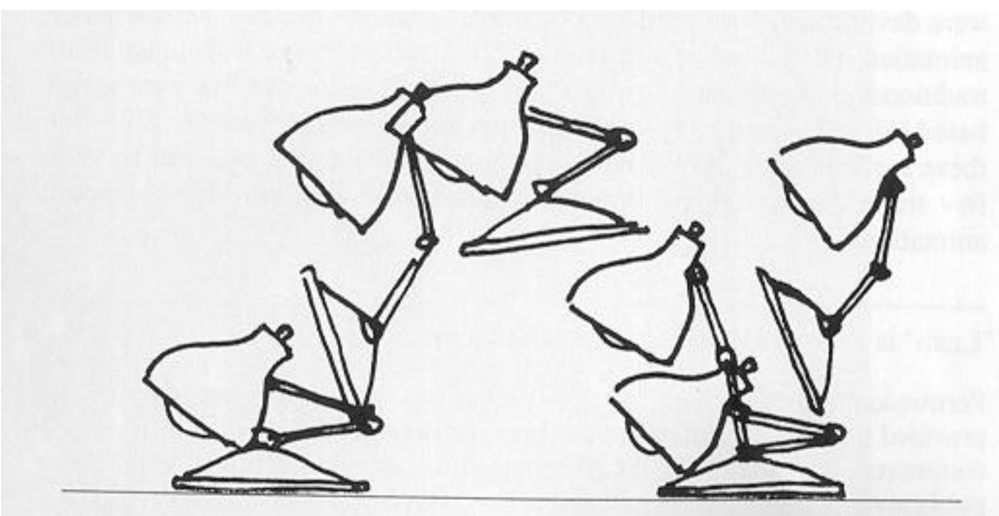
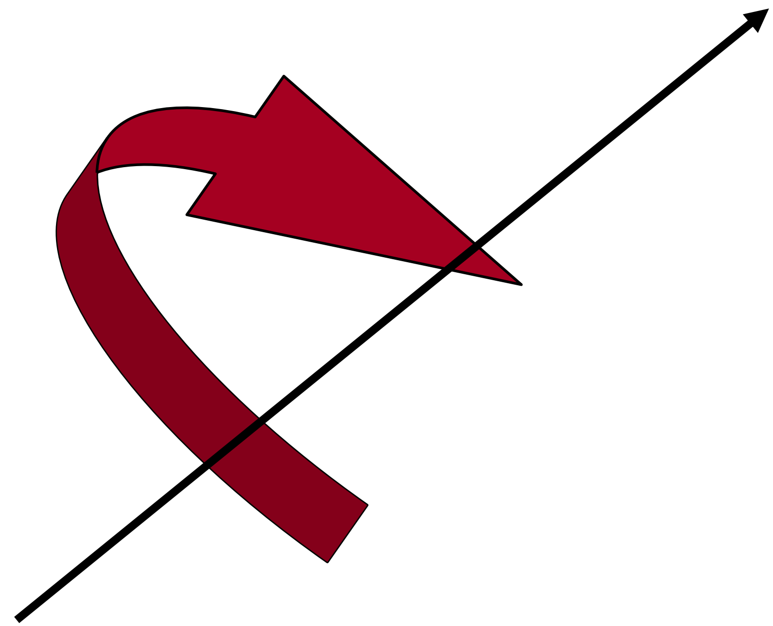
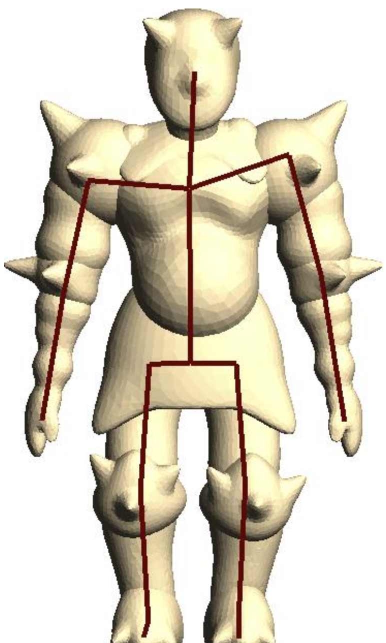


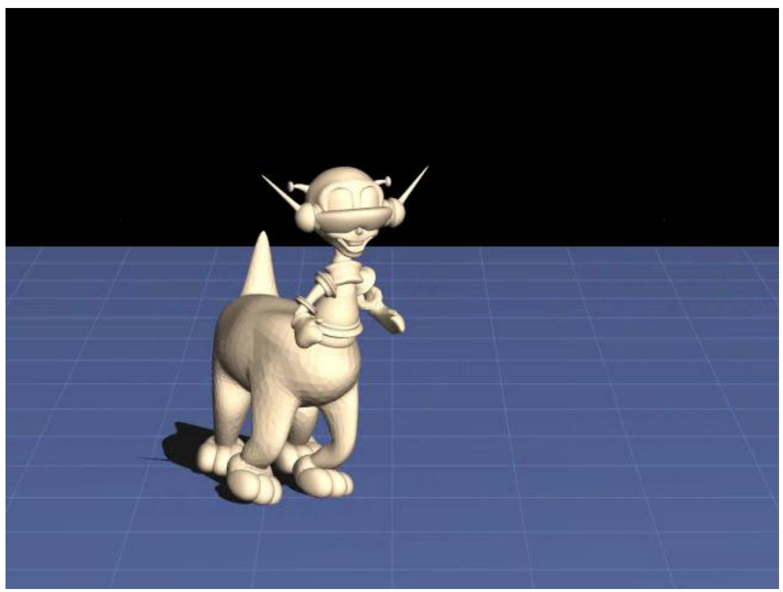
FIGURE 3. Squash & stretch in Luxo Jr.'s hop.



Character Animation: Skinning

- Animate simple “skeleton”
- Attach “skin” to skeleton
  - Skin deforms smoothly with skeleton
- Used everywhere (games, movies)



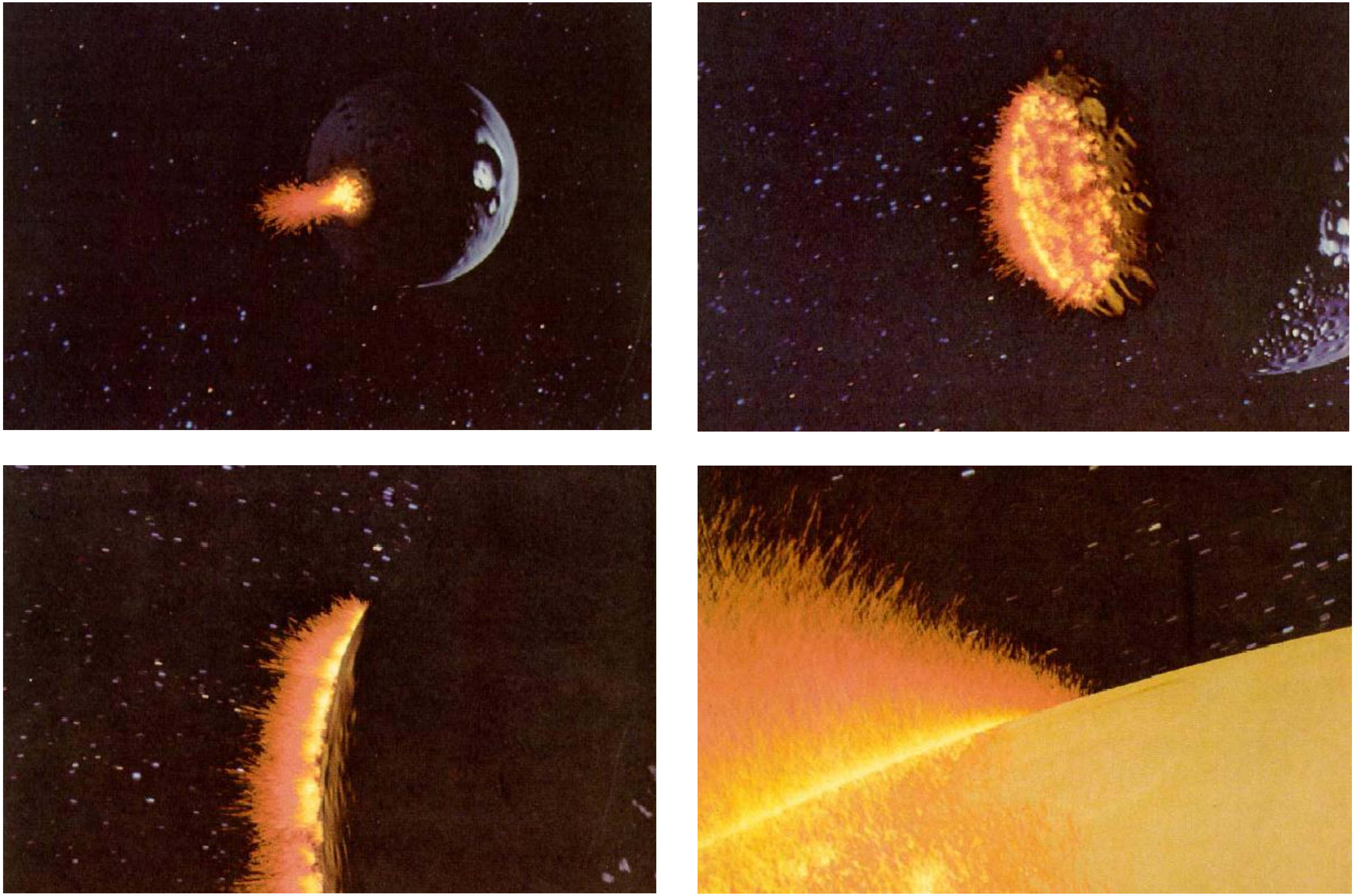


Ilya Baran

Image removed due to copyright restrictions.

Epic Games

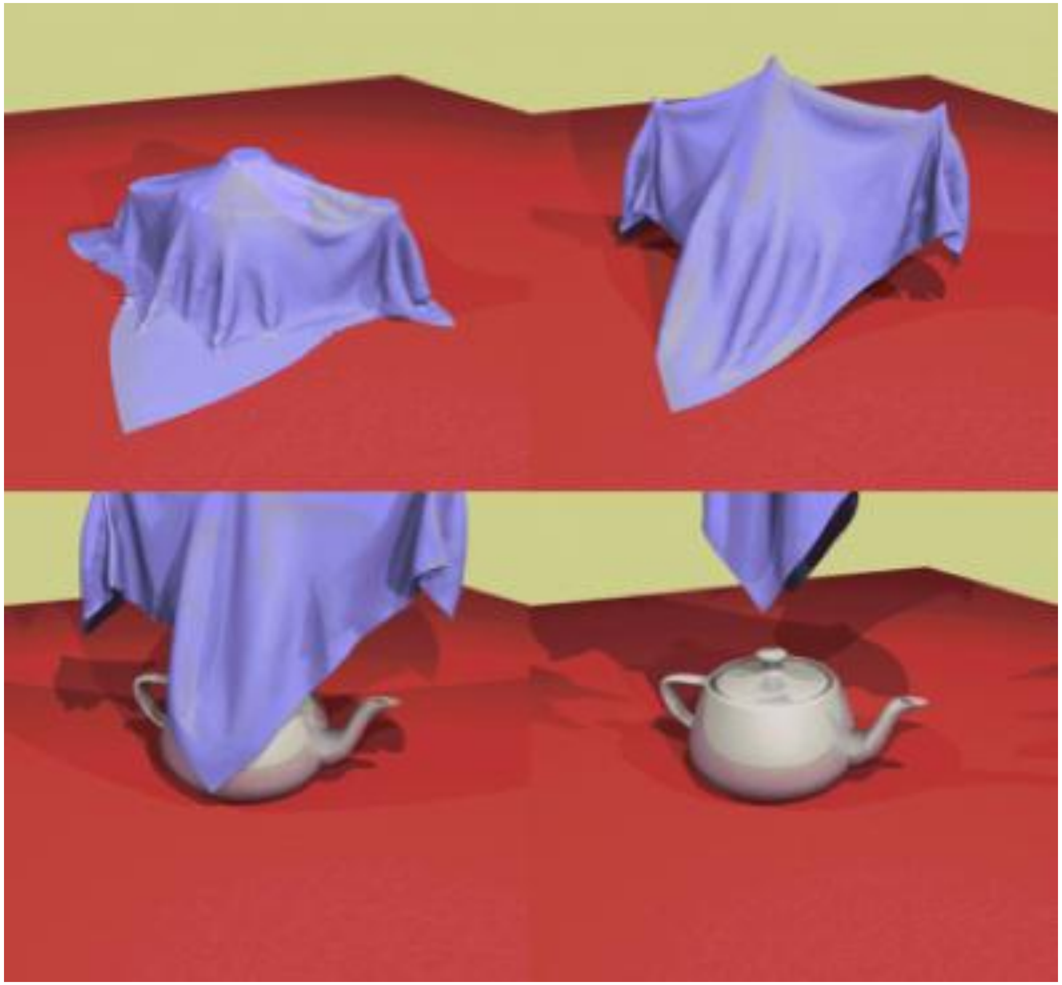
Particle system (PDE)



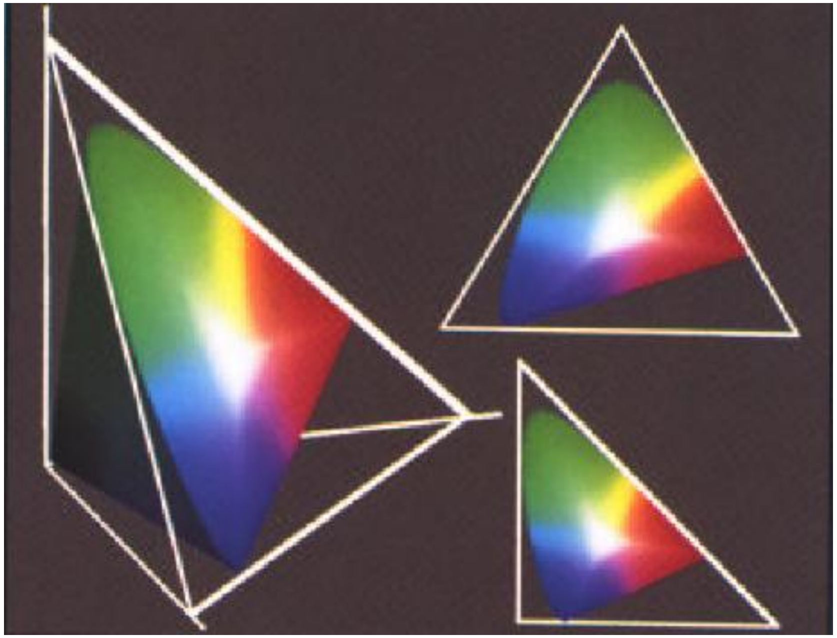
“Physics” (ODEs)

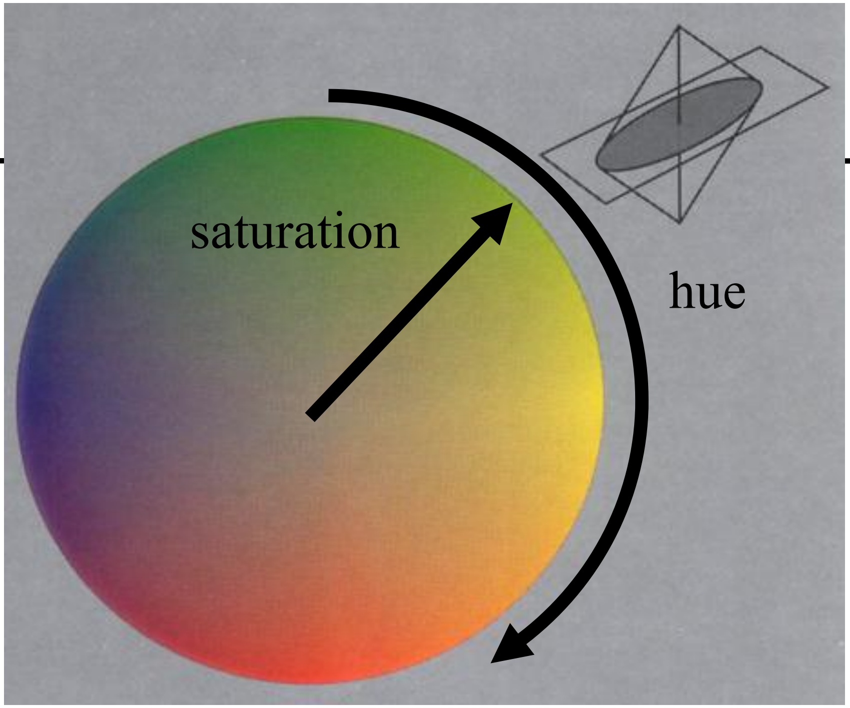
- Fire, smoke
- Cloth

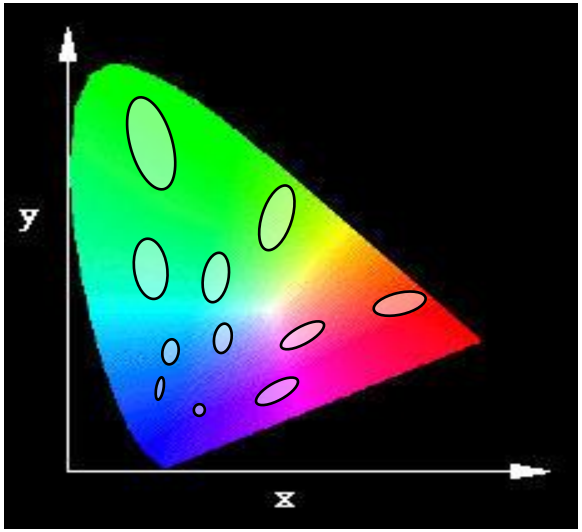
- Quotes because we do “visual simulation”

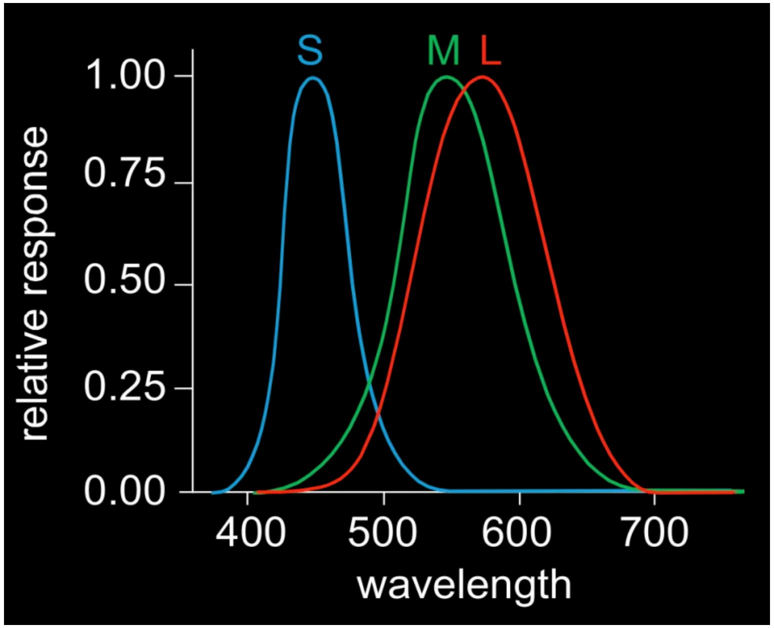


Color





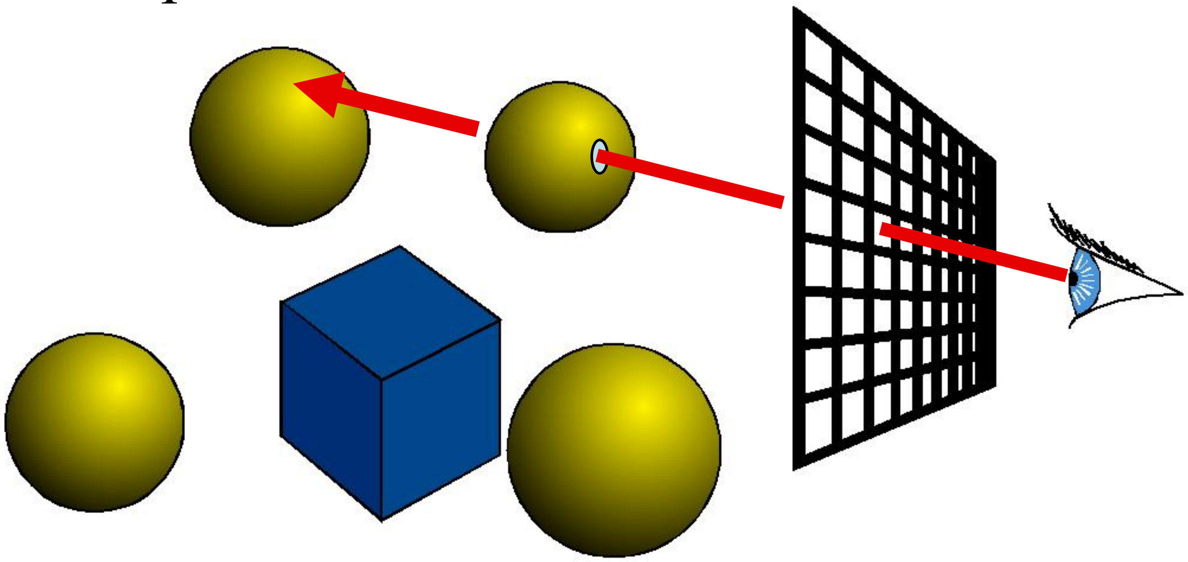




Courtesy of Victor Ostromoukhov.

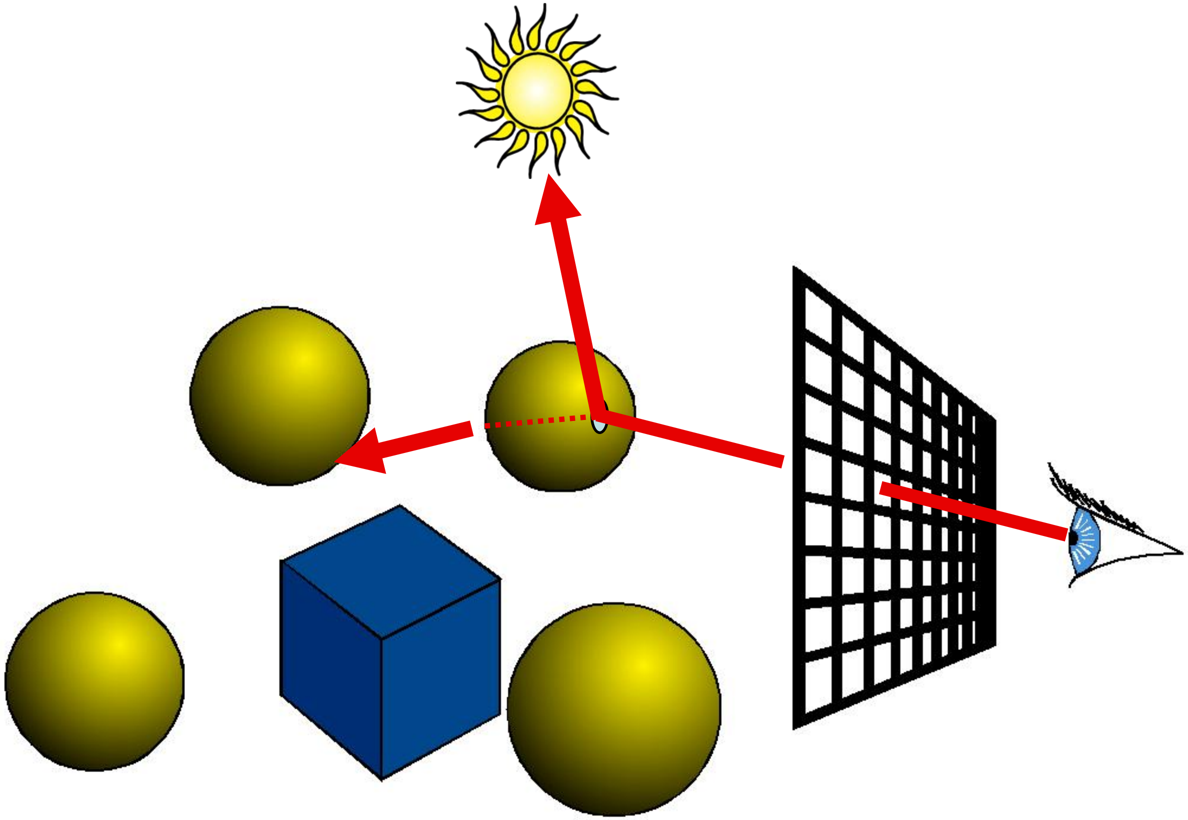
Ray Casting

- For every pixel construct a ray from the eye
  - For every object in the scene
    - Find intersection with the ray
    - Keep if closest



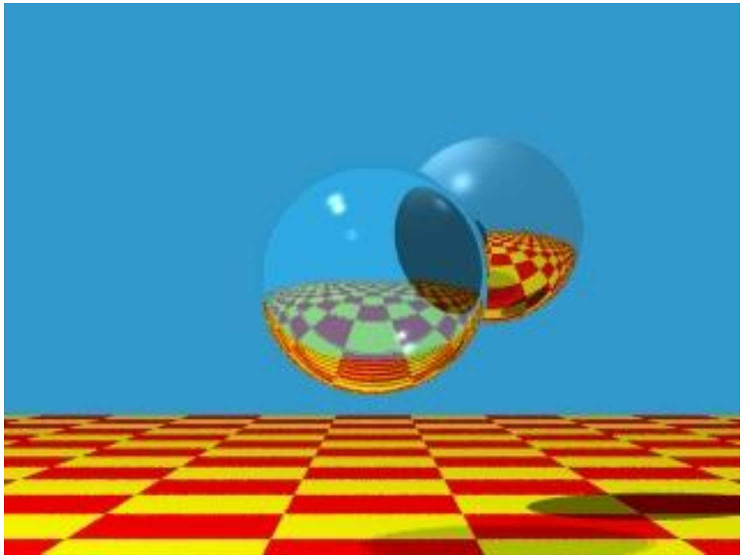
Ray Tracing

- Shade (interaction of light and material)
- Secondary rays (shadows, reflection, refraction)



Ray Tracing

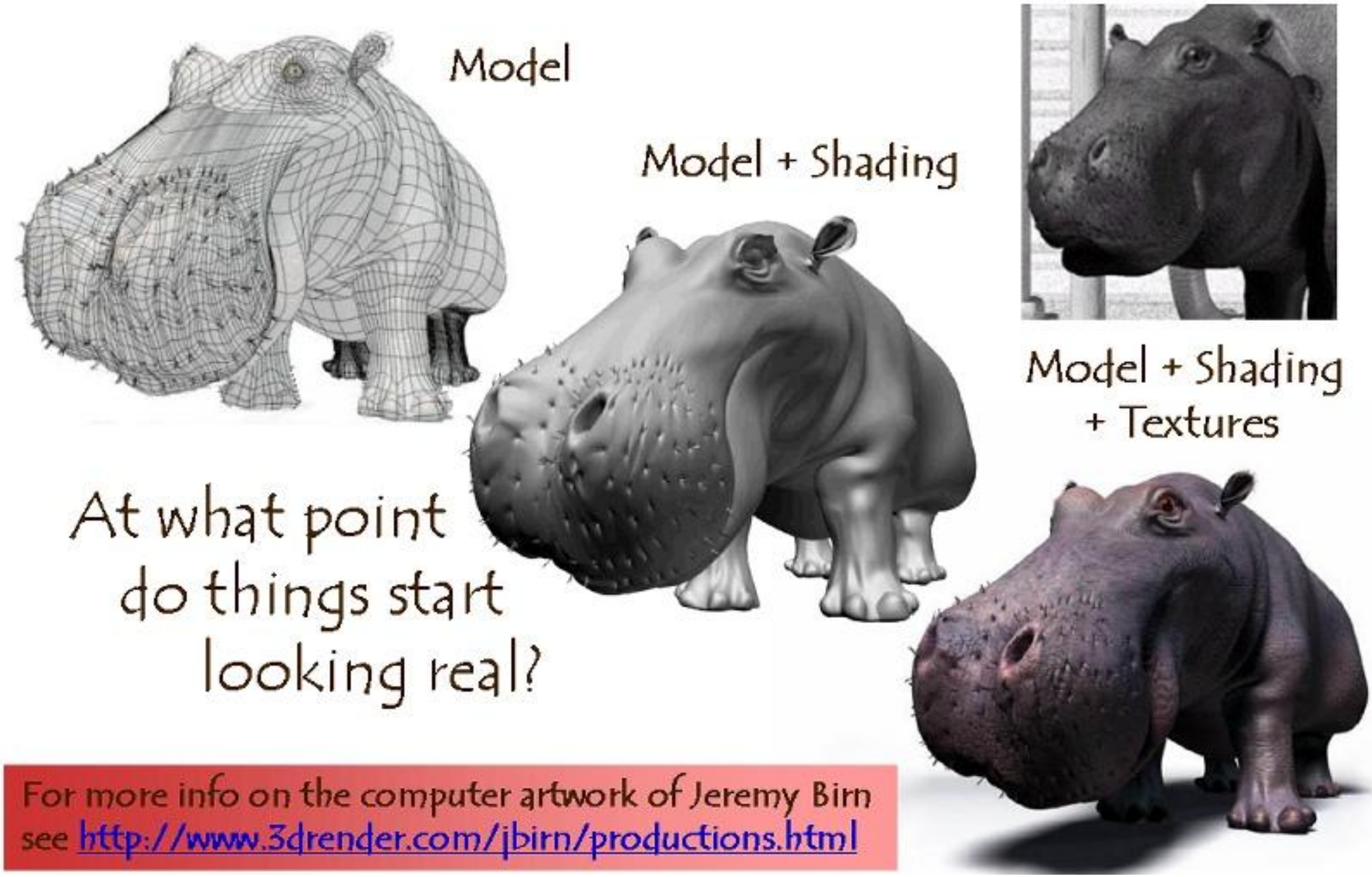
- Original Ray-traced image by Whitted



- Image computed using the Dali ray tracer by Henrik Wann Jensen
- Environment map by Paul Debevec

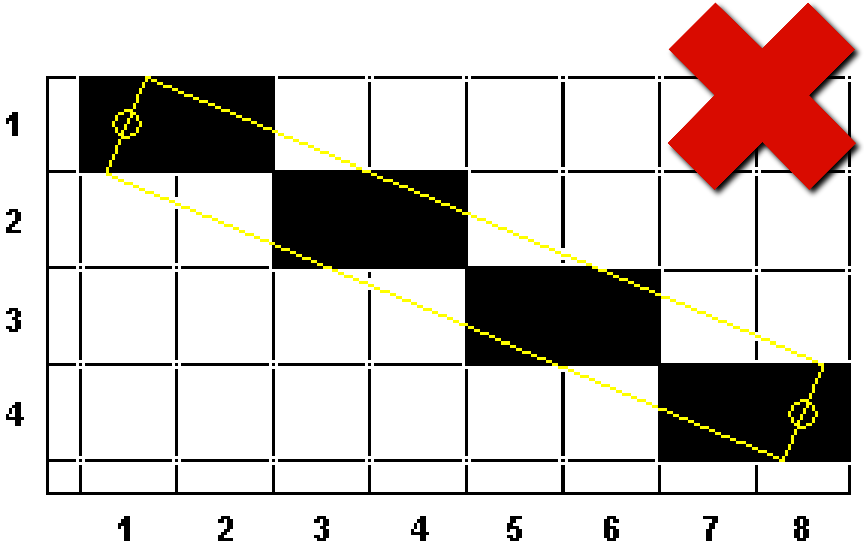
Courtesy of Henrik Wann Jensen. Used with permission.

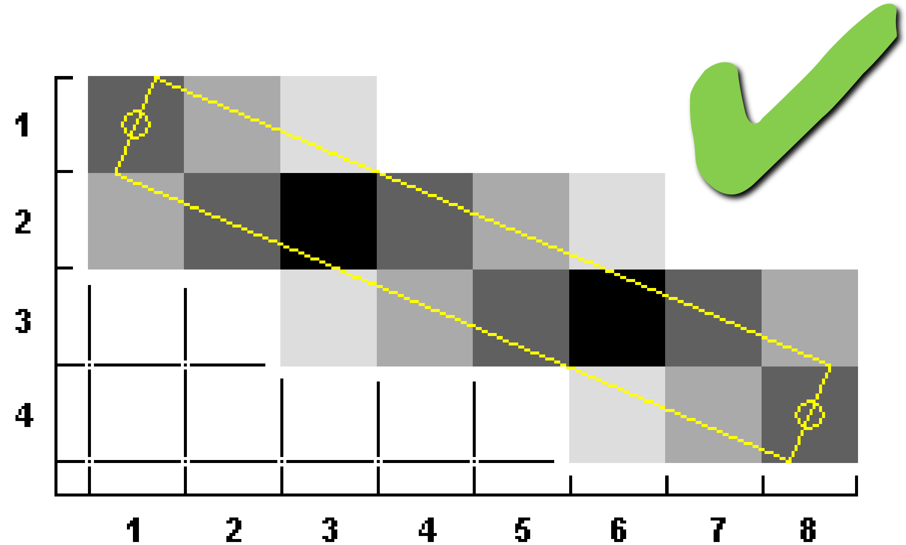
Textures and Shading

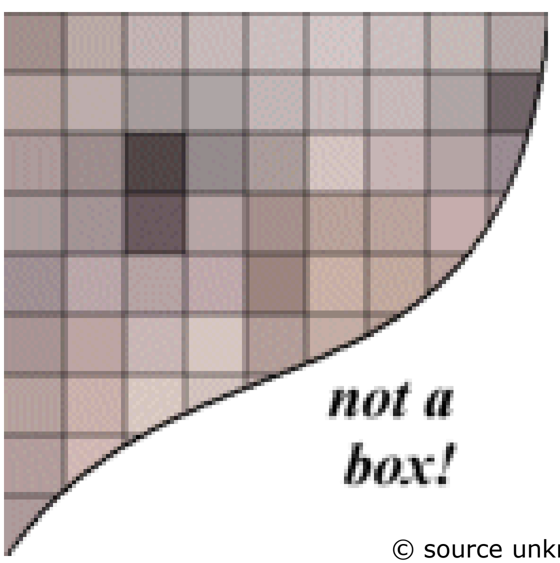


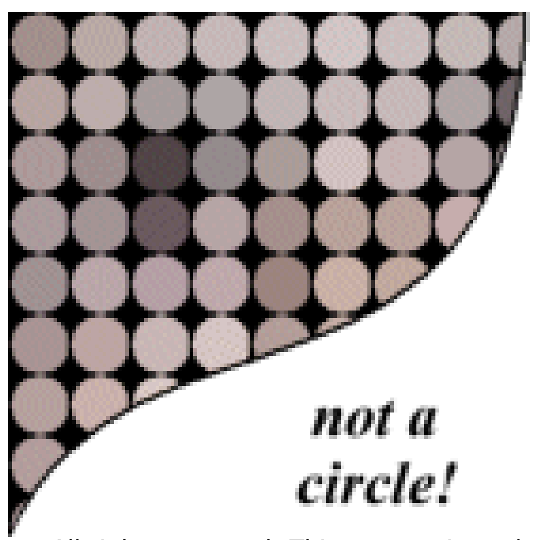
Courtesy of Jeremy Birn.

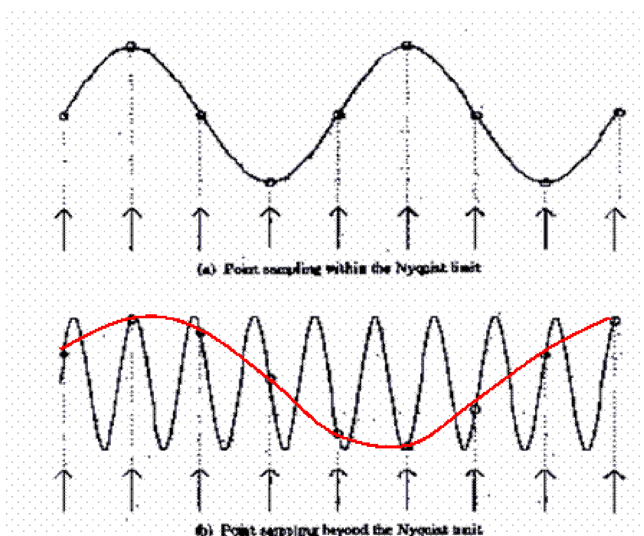
Sampling & Antialiasing











Shadows

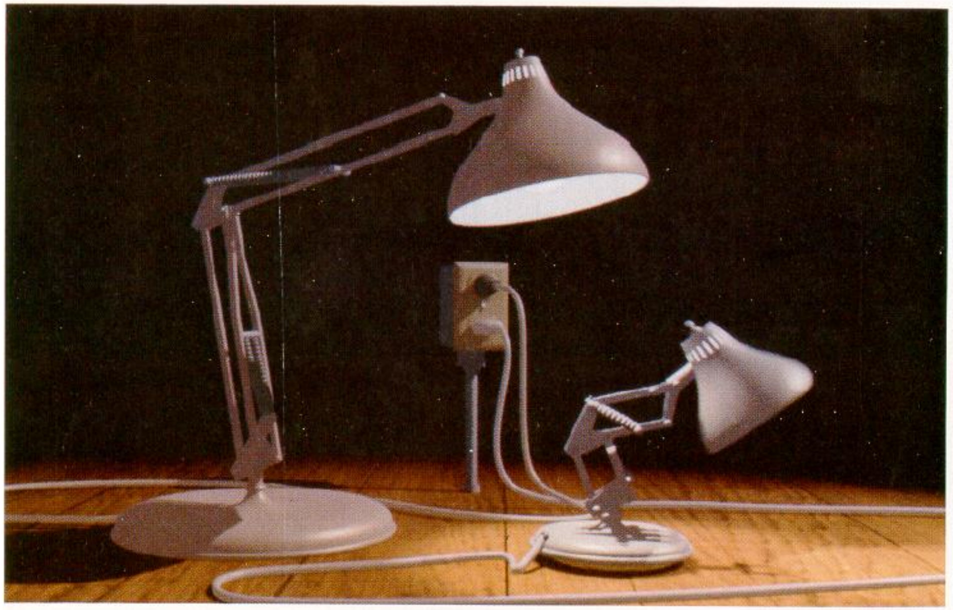


Figure 12. Frame from *Luxo Jr.*

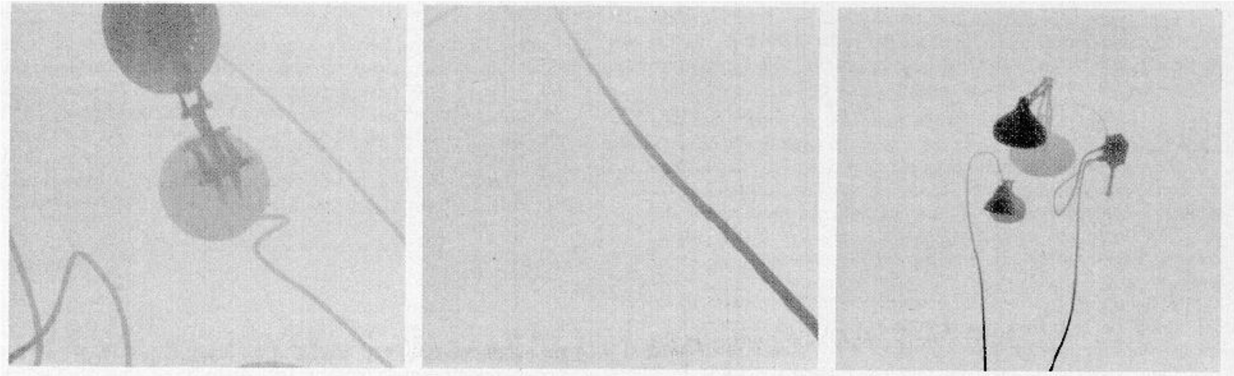
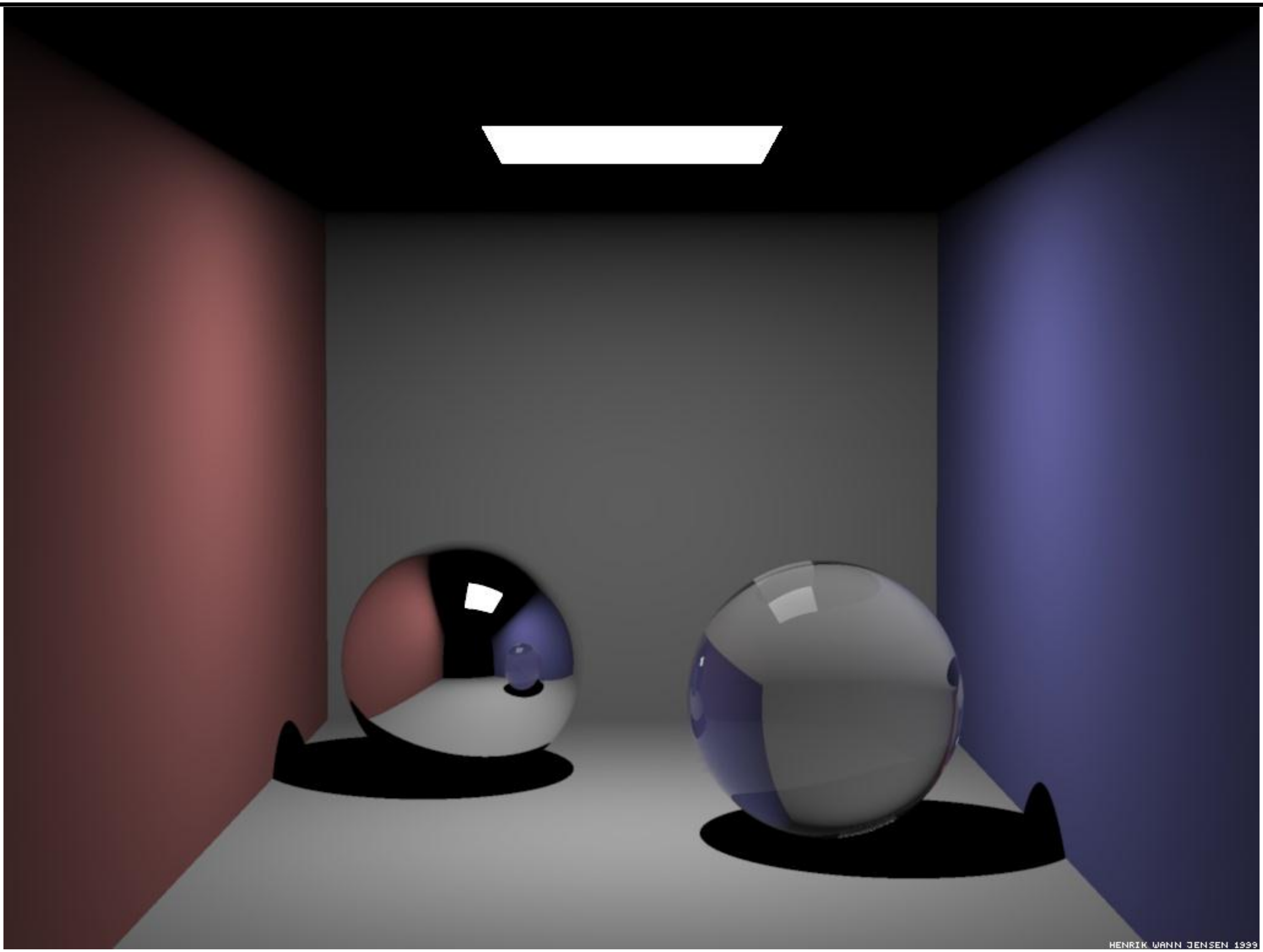


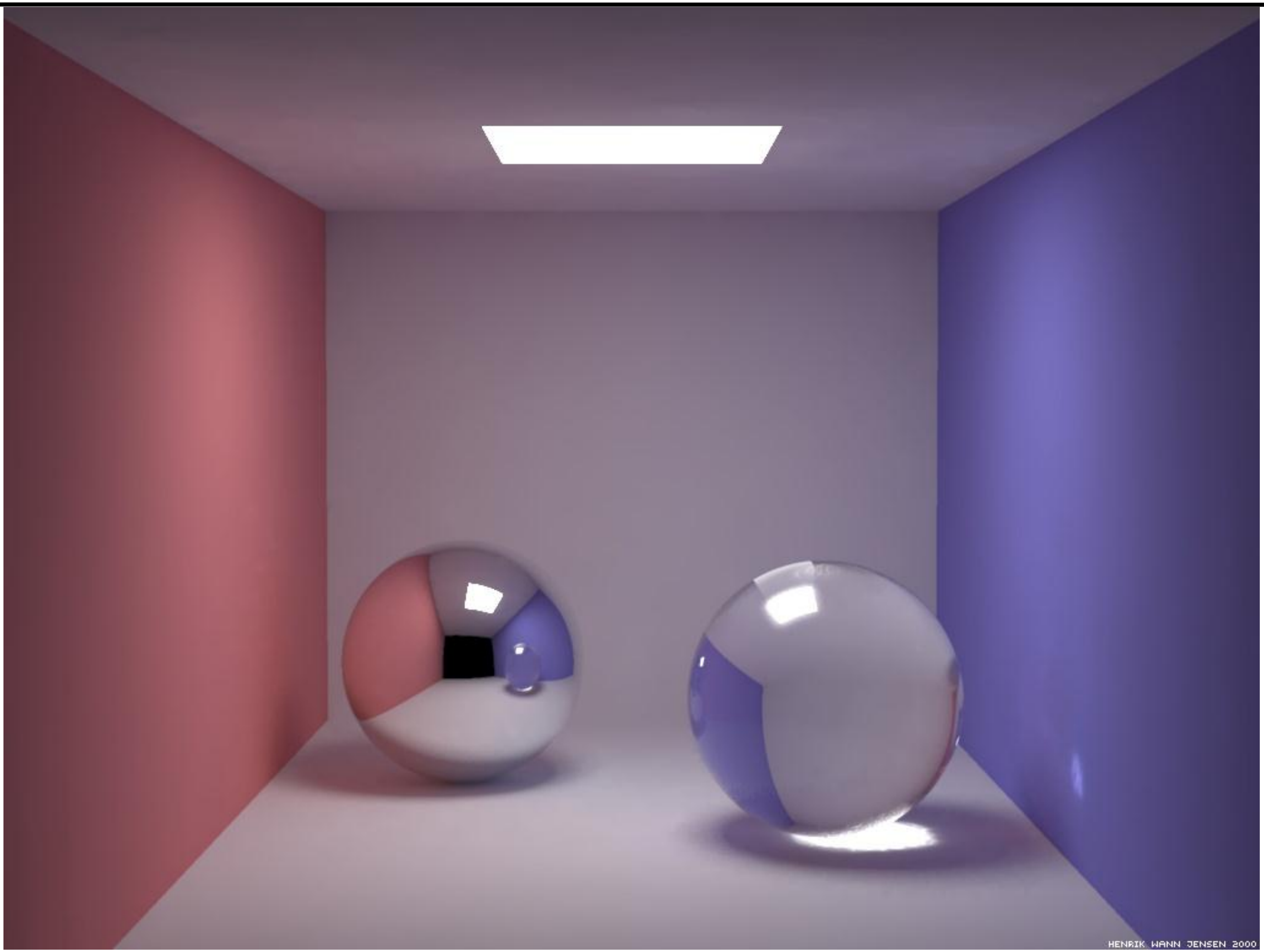
Figure 13. Shadow maps from *Luxo Jr.*

Traditional Ray Tracing



Courtesy of Henrik Wann Jensen. Used with permission.

Global Illumination

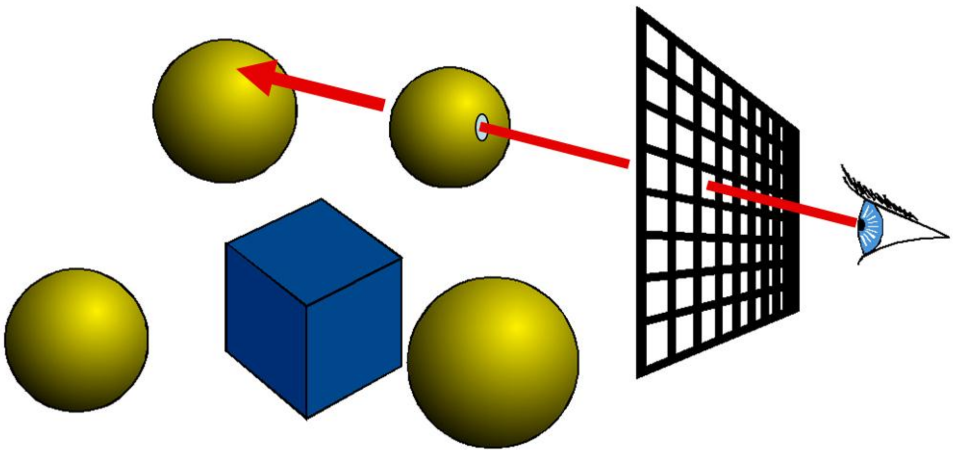


Courtesy of Henrik Wann Jensen. Used with permission.

The Graphics Pipeline

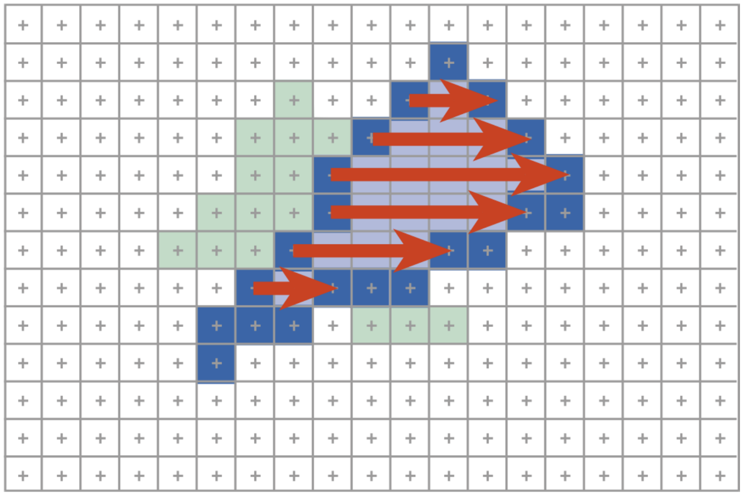
Ray Casting

```
For each pixel
    For each object
      Send pixels to scene
```



Rendering Pipeline

```
For each triangle
   For each projected pixel
       Project scene to pixels
```



The Graphics Pipeline

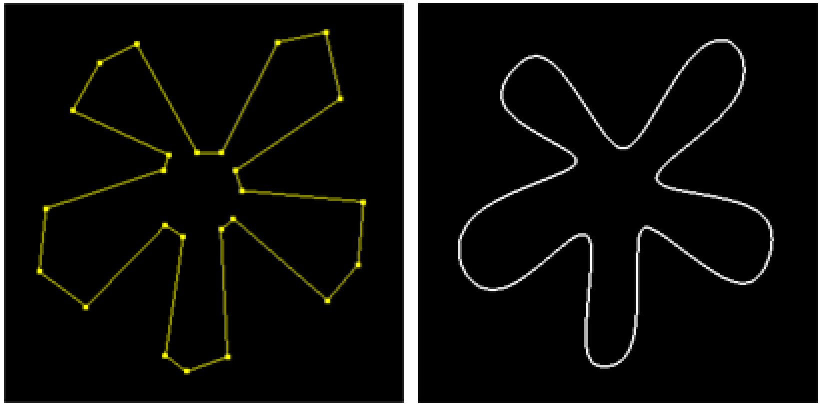
- Transformations
- Clipping
- Rasterization
- Visibility

Questions?

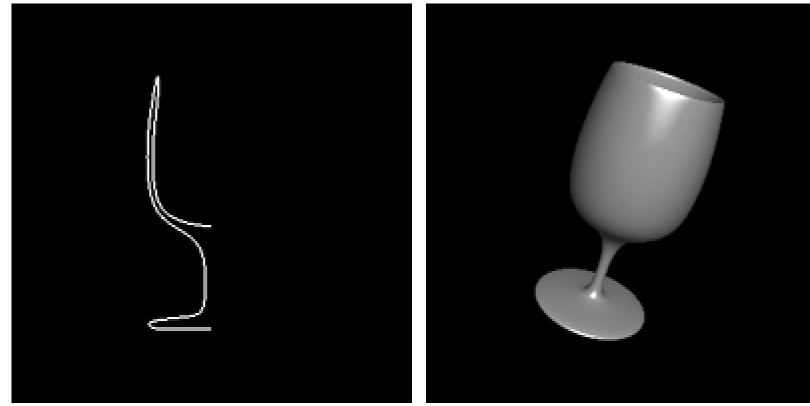
Plan

- Overview of computer graphics
- Administrivia
- Overview of the semester
- **Overview of assignments**
- Intro to OpenGL & assignment 0

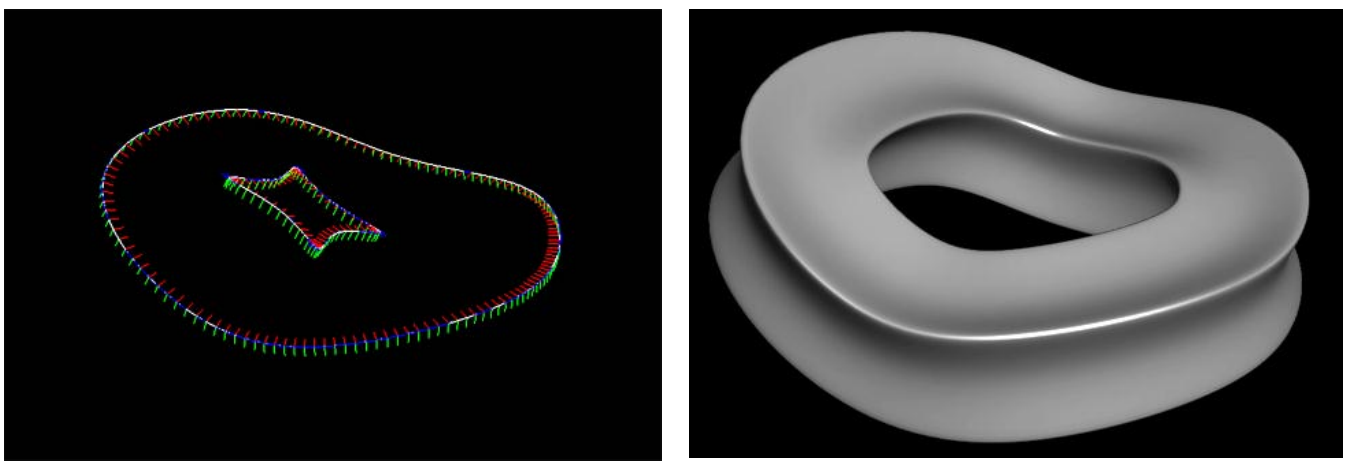
Assignment 1: curves & surfaces



Bezier curves



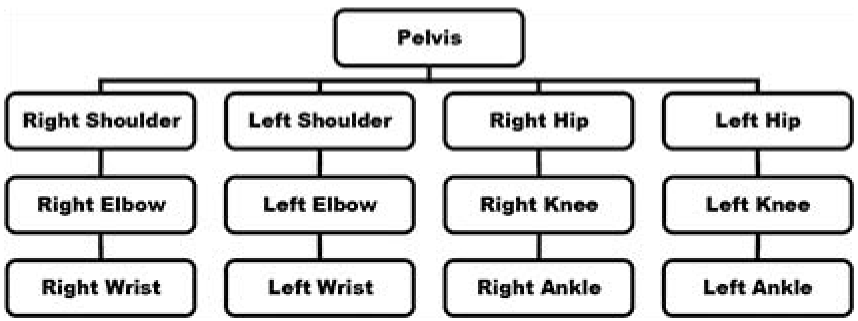
Surfaces of revolution

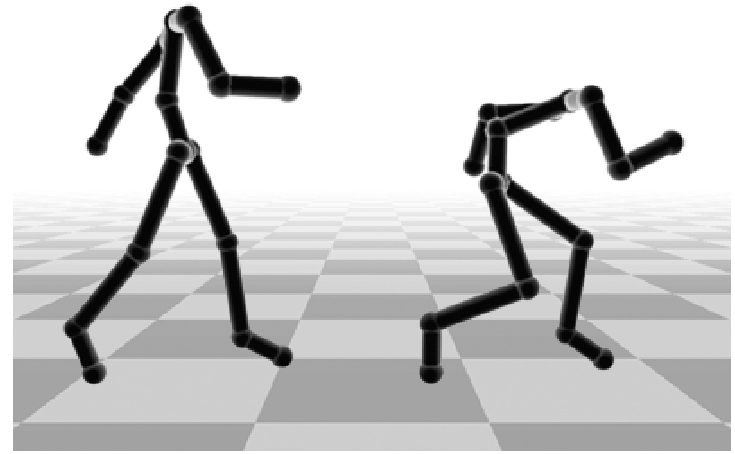


Sweep surfaces

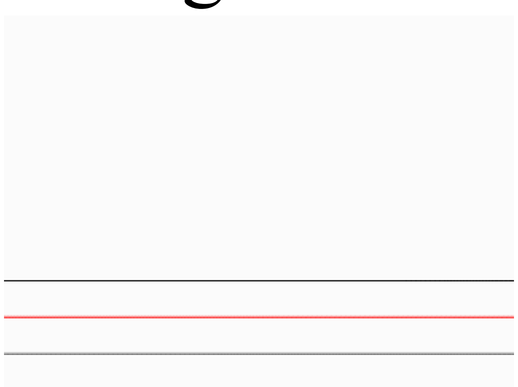
Assignment 2: hierarchical modeling

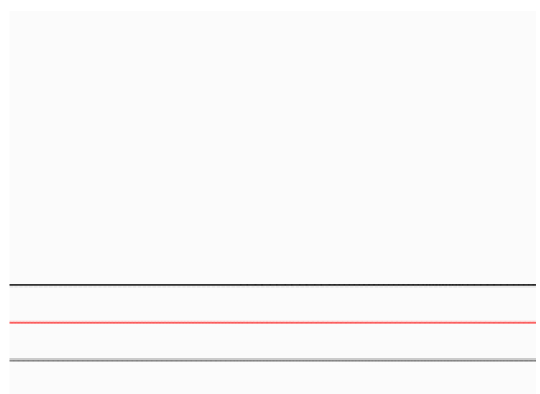
- Animate character skeleton as tree of transformations





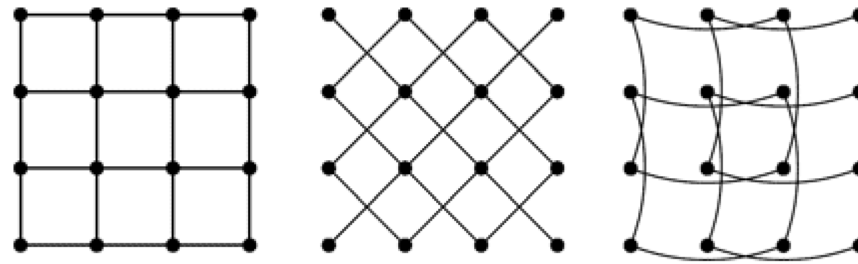
- Skinning: smooth surface deformation

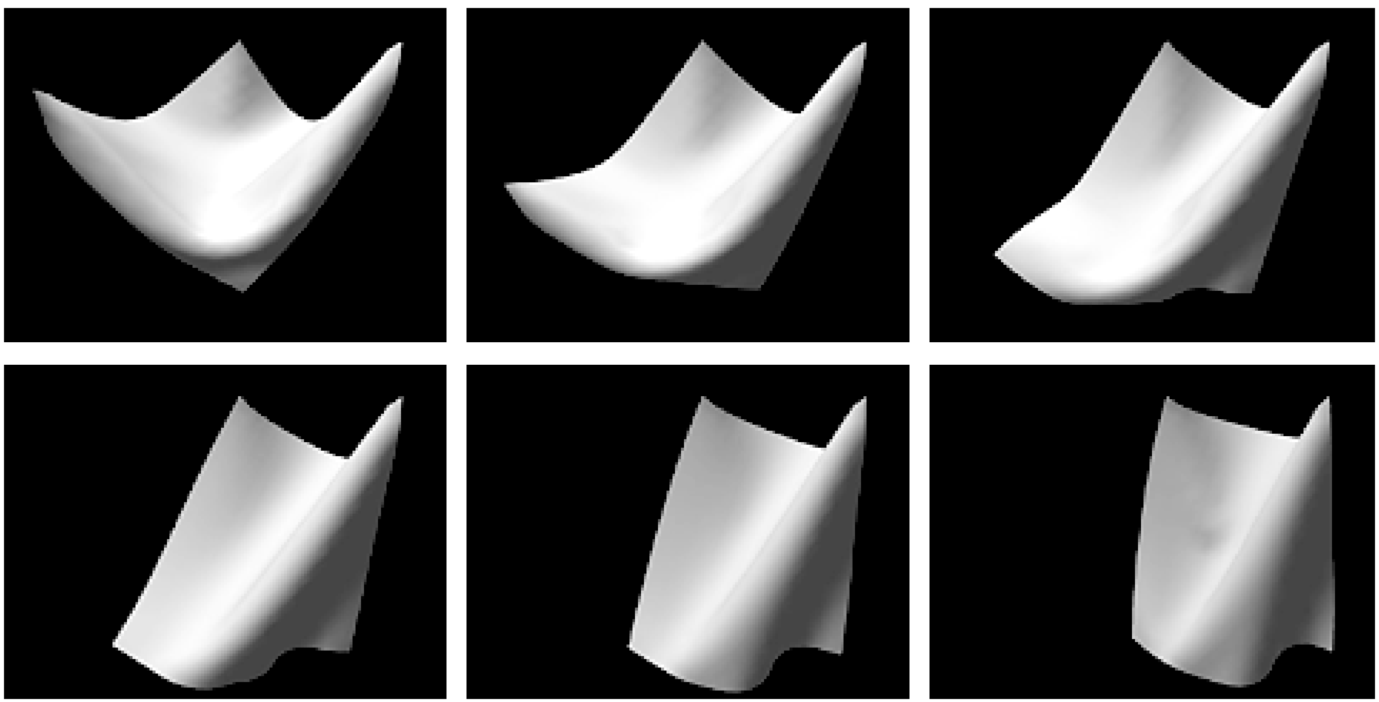




Assignment 3: physics

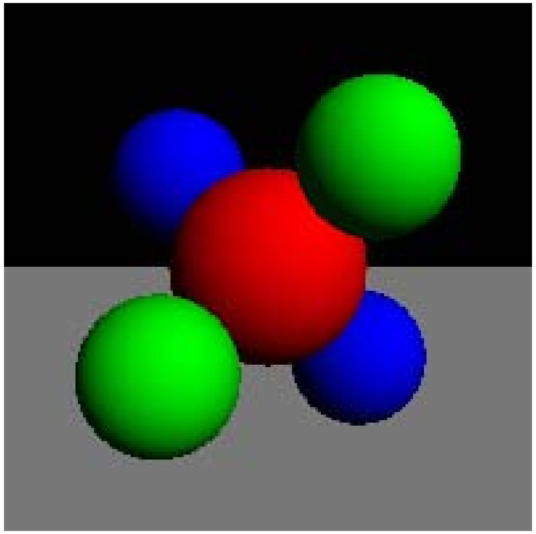
- Simulate cloth as a mass-spring network
  - ODE integration

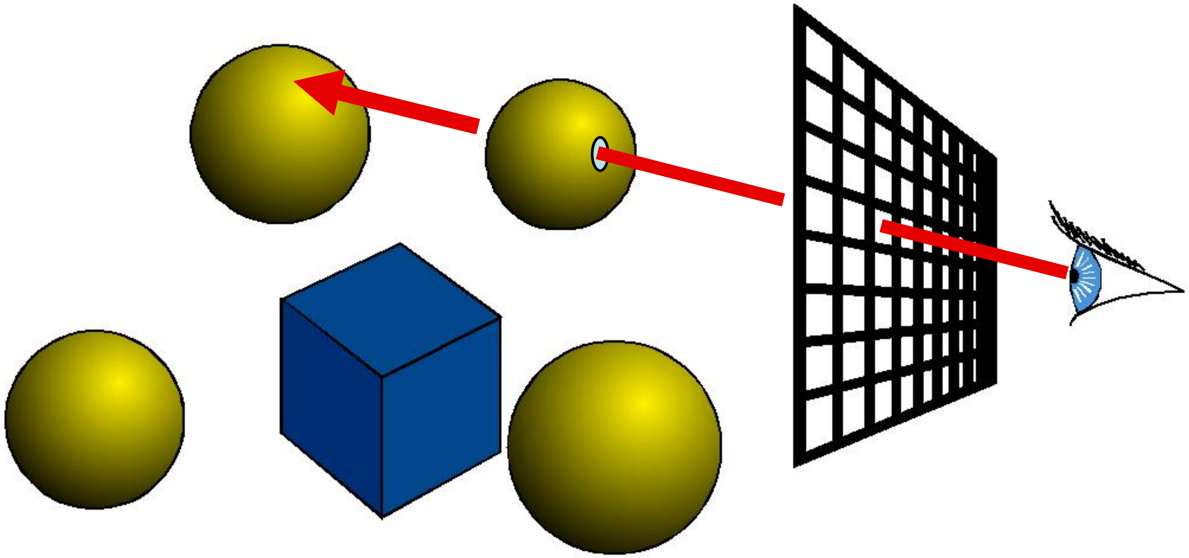




Assignment 4: ray casting

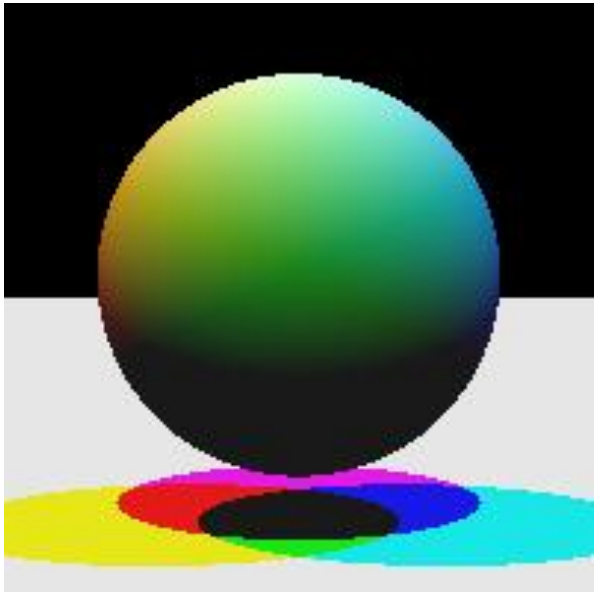
- Cast rays from the viewpoint
- Intersect with scene primitives

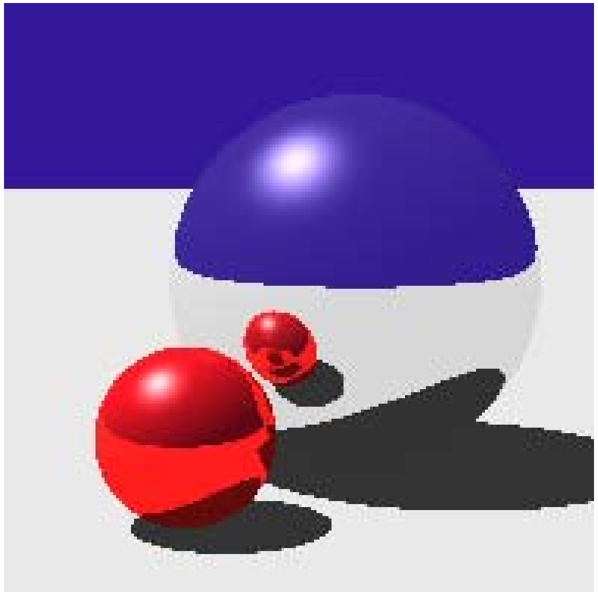


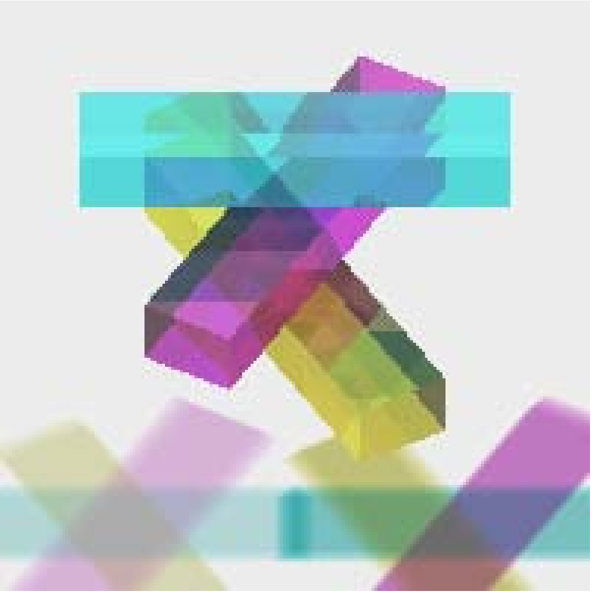


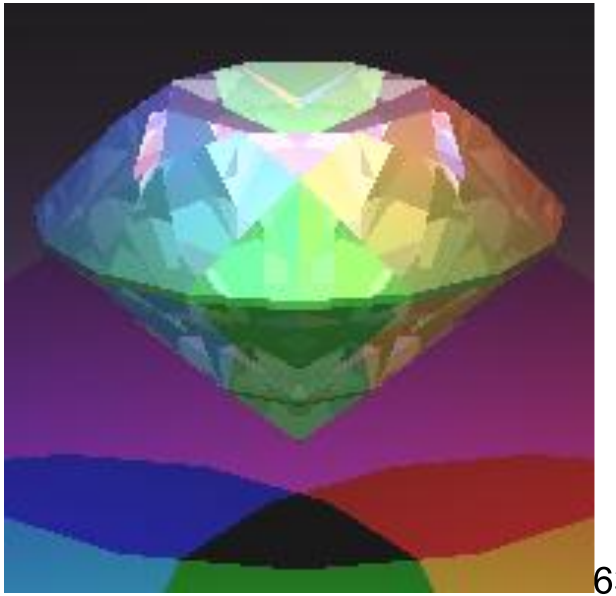
Assignment 5: ray tracing

- Shadows, reflection, refraction
- + flexible extension









Questions?

Plan

- Overview of computer graphics
- Administrivia
- Overview of the semester
- Overview of assignments
- **Intro to OpenGL & assignment 0**

Simple 3D with OpenGL

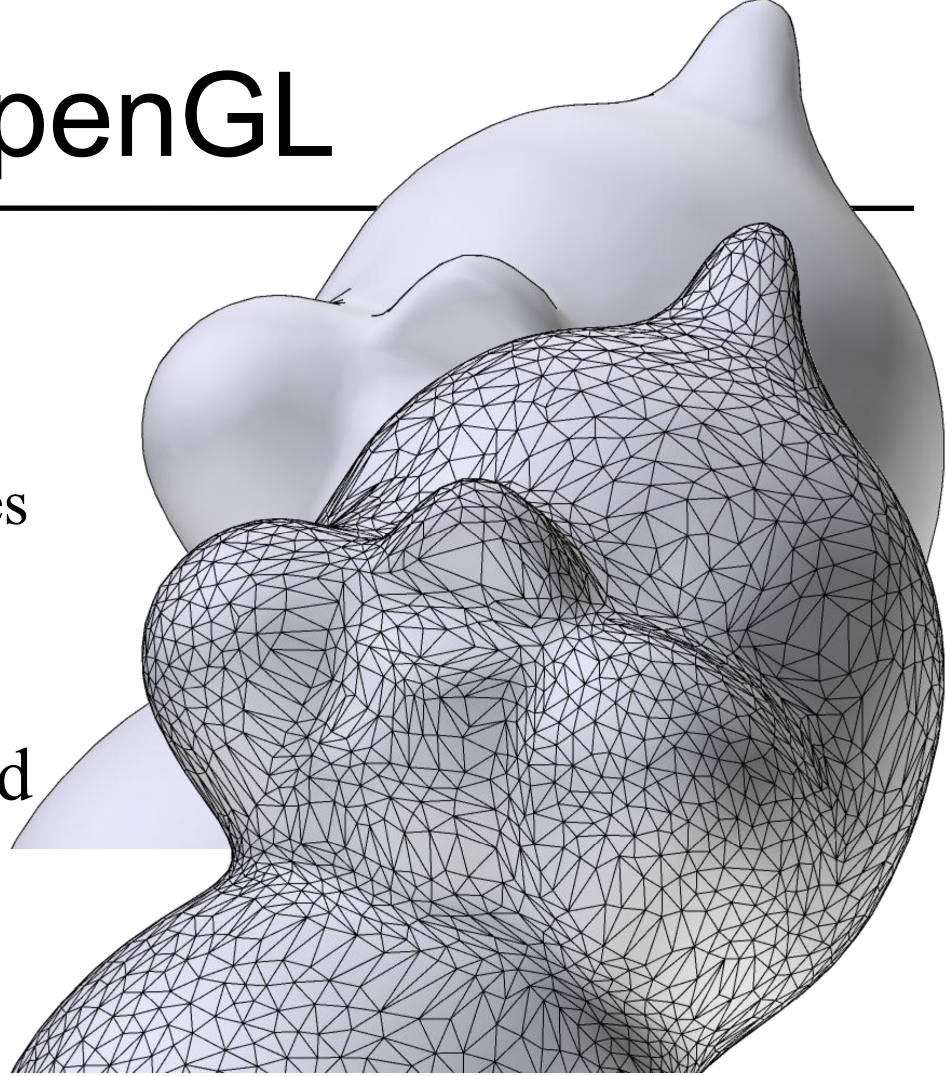
- OpenGL is an API that allows you to send commands to the graphics card to draw 2D or 3D scenes
- At the beginning of the semester, we will use OpenGL as a black box to display 3D content
- Later, we will see what is under the hood

Assignment 0

- Read a file with triangle mesh data
  - Including mesh normals
- Display it using OpenGL
  - Colors, simple movement

- Due next Wednesday!

Simple 3D with OpenGL

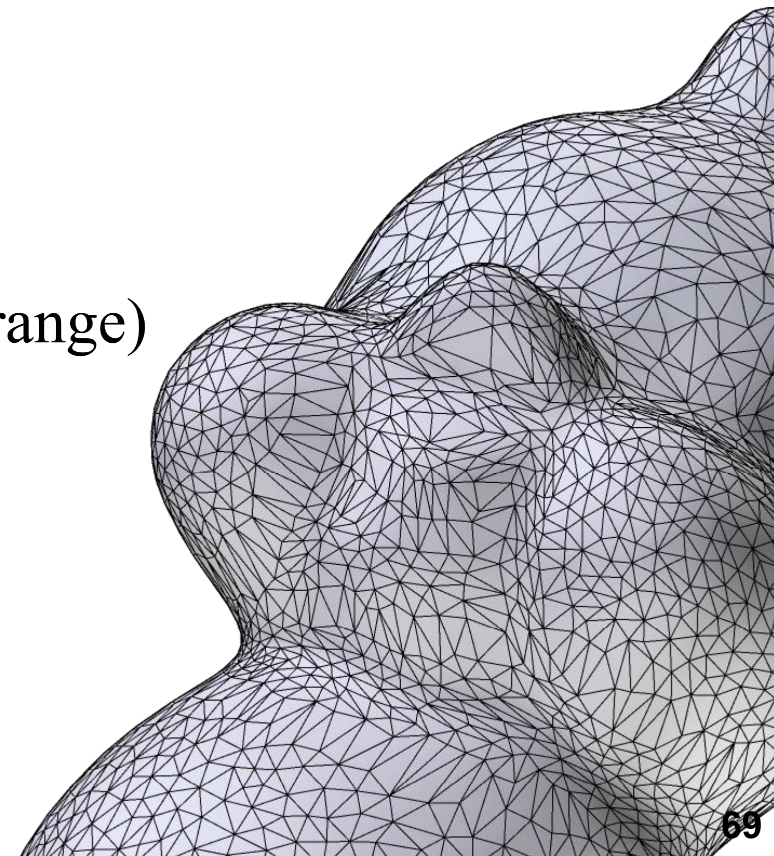


- Scene represented as triangles
  - A triangle is a set of 3 vertices
  - A vertex is a set of 3 floating point numbers (x, y, z)
- We will use OpenGL to send this to the graphics card (GPU)
  - The GPU will do its magic to display the scene from the current viewpoint (Later, we will get to see how this happens)

How to Draw?

Modelview
Object coordinates
World coordinates
View coordinates
Image coordinates
Projection

- You need to tell OpenGL
  - The geometry of the object
    - Vertex positions
    - Vertex normals
    - 3 x vertex makes a triangle!
  - Camera parameters
    - Field of view, aspect ratio, (depth range)
    - The "projection matrix"



Questions?

OpenGL high-level pseudocode

- Initialize
  - (get graphics context, etc.)
- For each frame
  - Manage UI
  - Set appropriate viewpoint
  - Set light source directions
  - For each triangle

```
For i=0 to 2
    Send vertex data
```

OpenGL Example: Viewing

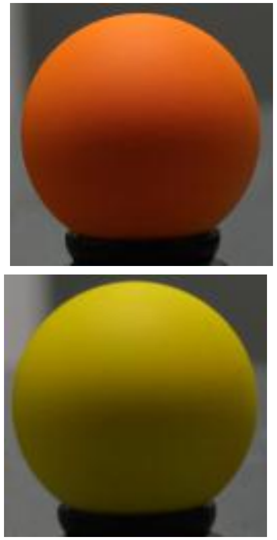
```
// Current matrix affects objects positions
glMatrixMode( GL_MODELVIEW );
// Initialize to the identity
glLoadIdentity();
// Position the camera at [0,0,5], looking at
// [0,0,0], with [0,1,0] as the up direction.
gluLookAt(0.0, 0.0, 5.0,
          0.0, 0.0, 0.0,
          0.0, 1.0, 0.0);
// Rotate by -20 degrees about [0,1,0]
glRotated(-20.0, 0.0, 1.0, 0.0);

// Draw a teapot.
glutSolidTeapot(1.0);
```

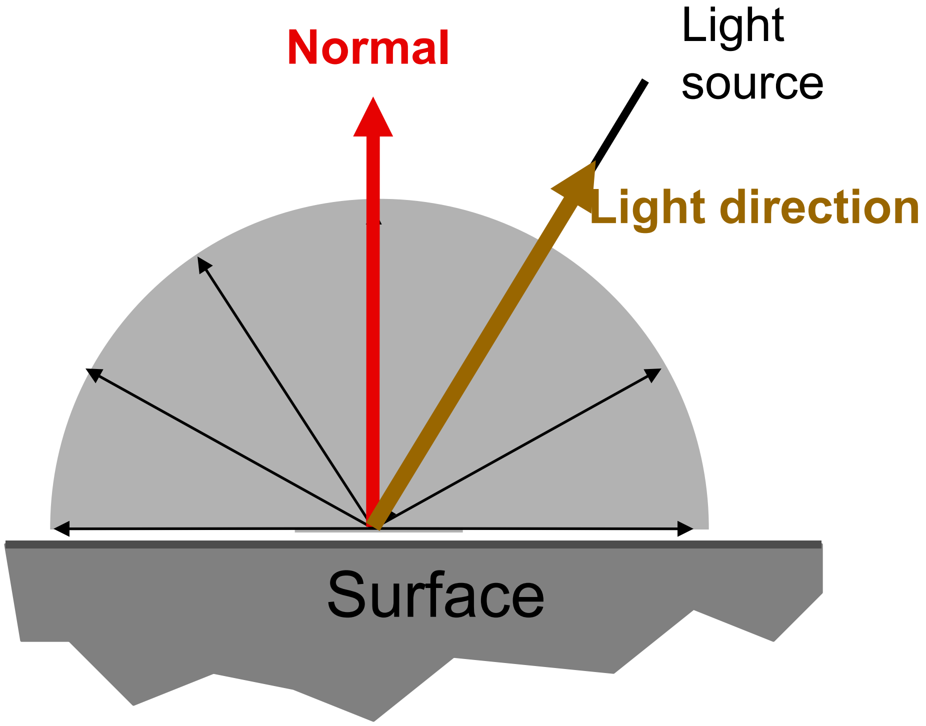
Vertex data

- What information do we need at each vertex?
  - Coordinates (3 floats)
  - Color (optional, 3 floats)
  - Normal information (optional, 3 floats)
  - Transparency (optional, 1 float)
  - More to come (texture information, shininess)

Why normals?

- To compute color as a function of light direction
- Simplest: Diffuse or Lambert model
  - Intensity = dot product (normal, light direction)





OpenGL Code

```
glBegin(GL_TRIANGLES); //what follows describes triangles
glColor3d (1,1,0); //red, green and blue components=>(yellow)
glNormal3d (0, 0, 1); //normal pointing up
glVertex3d (2,3,3); //3D position x, y, z
glColor3d (1,0,0);
glNormal3d (0, 0, 1);
glVertex3d (5,3,3);
glColor3d (1,0,1);
glNormal3d (0, 0, 1);
glVertex3d (3,6,3);
glEnd();
```

OpenGL high-level pseudocode

- Initialize
  - (get graphics context, etc.)
- For each frame
  - Manage UI
  - Set appropriate viewpoint
  - Set light source directions
  - For each triangle

```
For i=0 to 2
    Send vertex data
```

OpenGL is a state machine

- Each command changes the state
  - But `glVertex` also "pushes" data
- For example, glColor3f changes the current color.
  - The color remains valid until we call glColor*xx* again
  - Use it before each vertex to get per-vertex color.
- Other state to manage lighting and other rendering aspects
- Can make it hard to debug
- *(Note: This is conceptually simple, but not quite how you write efficient code these days.)*

Assignment 0

- Read a file with triangle mesh data
  - Including mesh normals
- Display it using OpenGL
  - Colors, simple movement

- Due next Wednesday!

What is missing?

- Shadows
- Shininess
- Texture
- Etc.

- Be patient, you will have plenty enough

Linear Algebra is Everywhere

- Vertices are 3-vectors
- Normals are 3-vectors
  - Orthogonal to surface tangent plane
  - Cross product
- Colors are 3-vectors
- Diffuse shading is a dot product
- A non-bending object moving in a scene undergoes a rigid transformation
- Changing the viewpoint is a linear transformation of the scene coordinate
- Brush up in the review session!

What Makes Graphics Fun?

- Very interdisciplinary
  - Within CS: systems, compilers, languages, computer architecture, algorithms, numerical techniques
  - Math, physics, art, perception, architecture, manufacturing
- Helps you understand why the world looks the way it does
- You can “see” the result

Image removed due to copyright restrictions.

visualparadox.com