CSE 167: Introduction to Computer Graphics

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2011

Today

- Course overview
- Course organization
- Vectors and Matrices

What is computer graphics

Applications:

- Movie, TV special effects
- Video games
- Scientific visualization
- GIS (Geographic Information Systems)
- Medical visualization
- Industrial design
- Simulation
- Communication
- Etc.

What is computer graphics?

- Rendering
- Modeling
- Animation

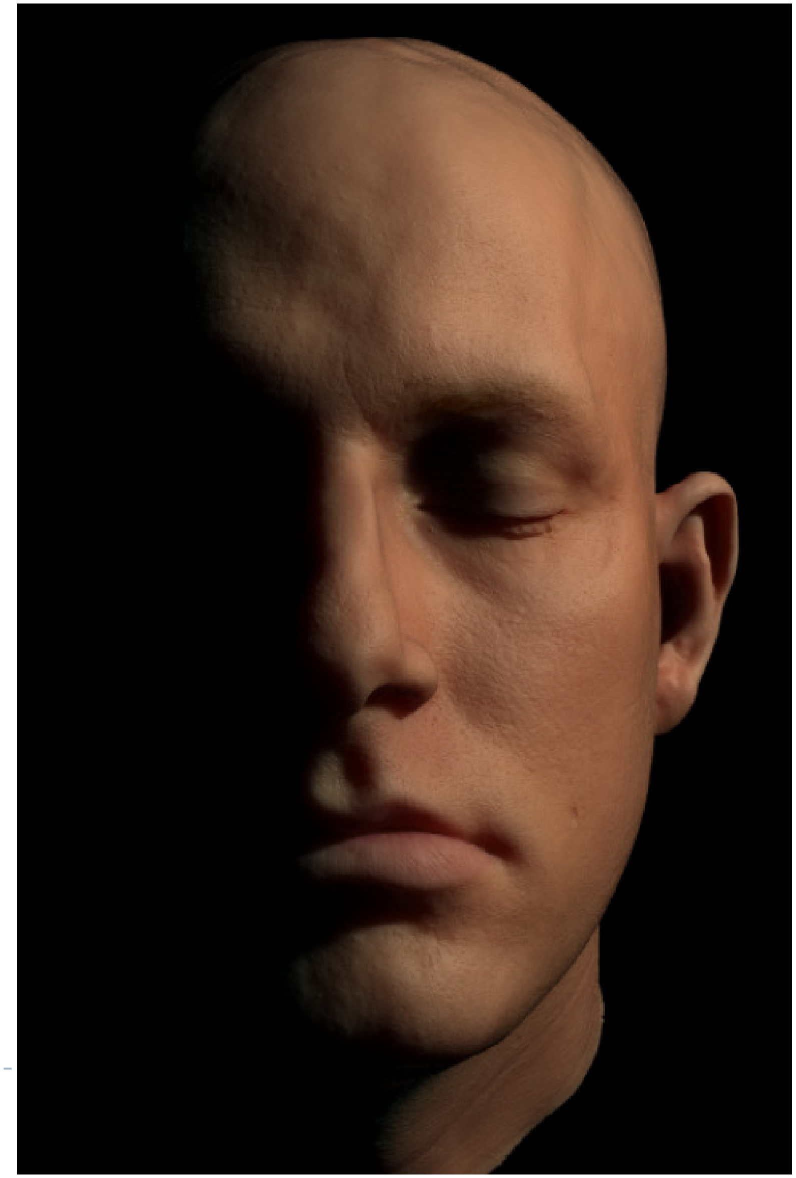
Rendering

- Synthesis of a 2D image from a 3D scene description
  - Rendering algorithm interprets data structures that represent the scene in terms of geometric primitives, textures, and lights
- 2D image is an array of pixels
  - Red, green, blue values for each pixel
- Different objectives
  - Photorealistic
  - Interactive
  - Artistic

Photorealistic rendering

- Physically-based simulation of light, camera
- Shadows, realistic illumination, multiple light bounces
- Slow, minutes to hours per image
- Special effects, movies
- CSE168: Rendering Algorithms

Photorealistic rendering



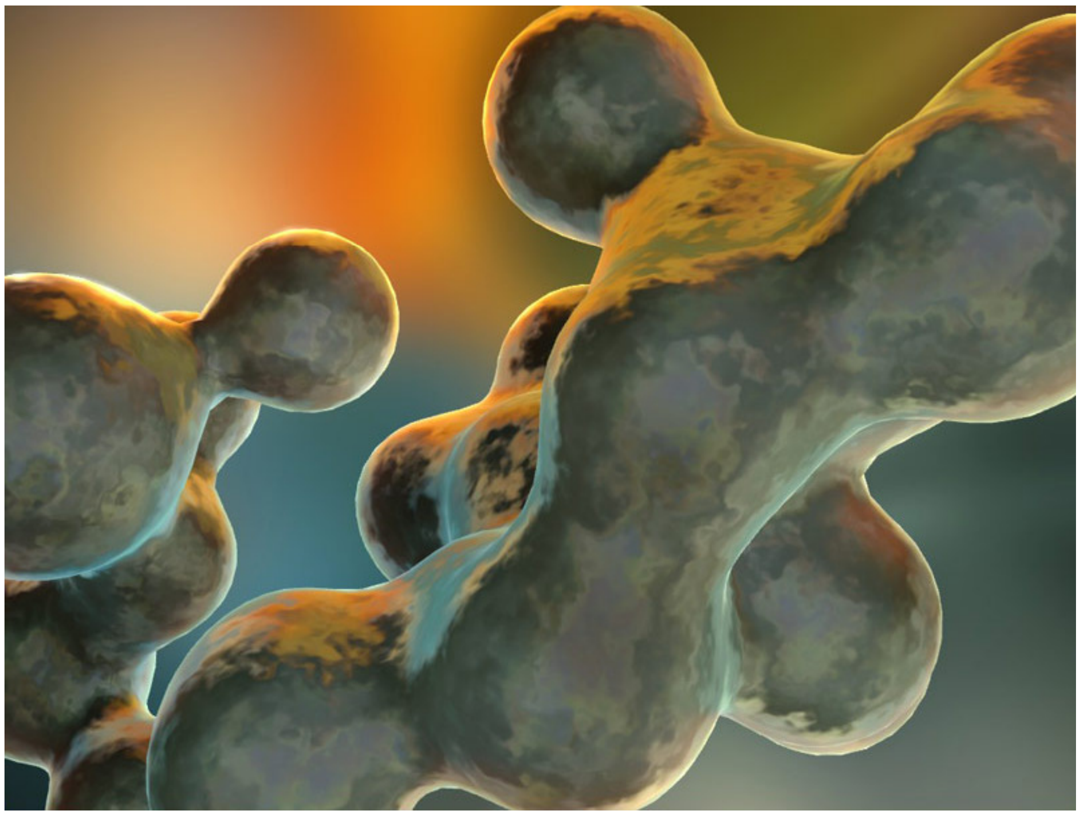
Interactive rendering

- Produce images within milliseconds
- Using specialized hardware, graphics processing units (GPUs)
- Standardized APIs (OpenGL, DirectX)
- Often "as photorealistic as possible"
- Hard shadows, fake soft shadows, only single bounce of light
- Games
- CSE167

Interactive rendering

Live Demo

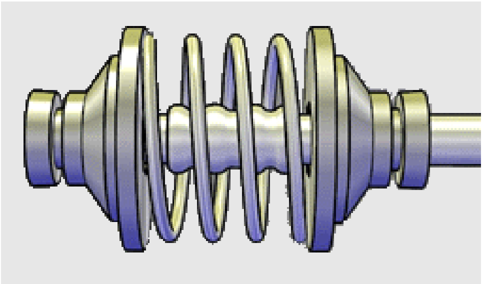
- NVIDIA Geoforms: Real-Time Rendering
  http://nzone.nvidia.com/object/nzone_geoforms_downloads.html

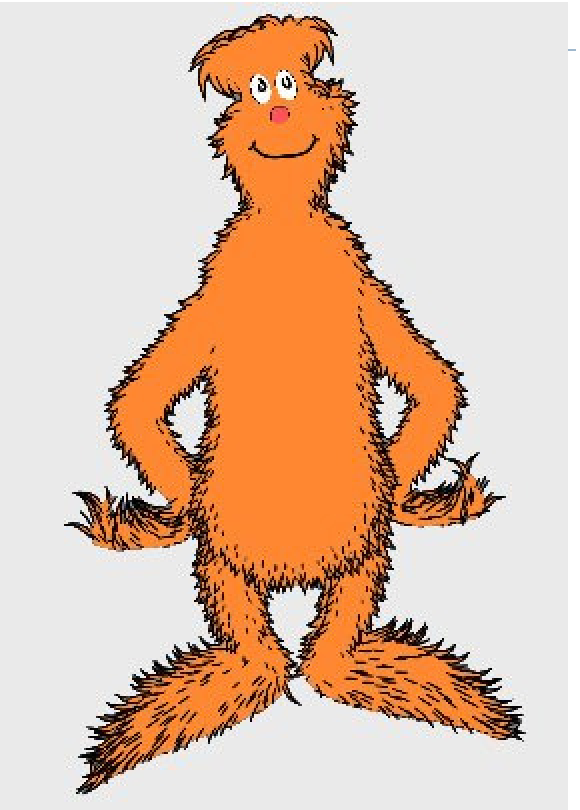


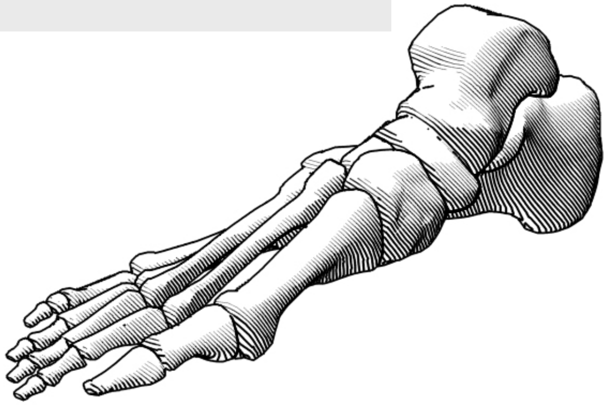
Artistic rendering

- Stylized
- Artwork, illustrations, data visualization

Artistic rendering

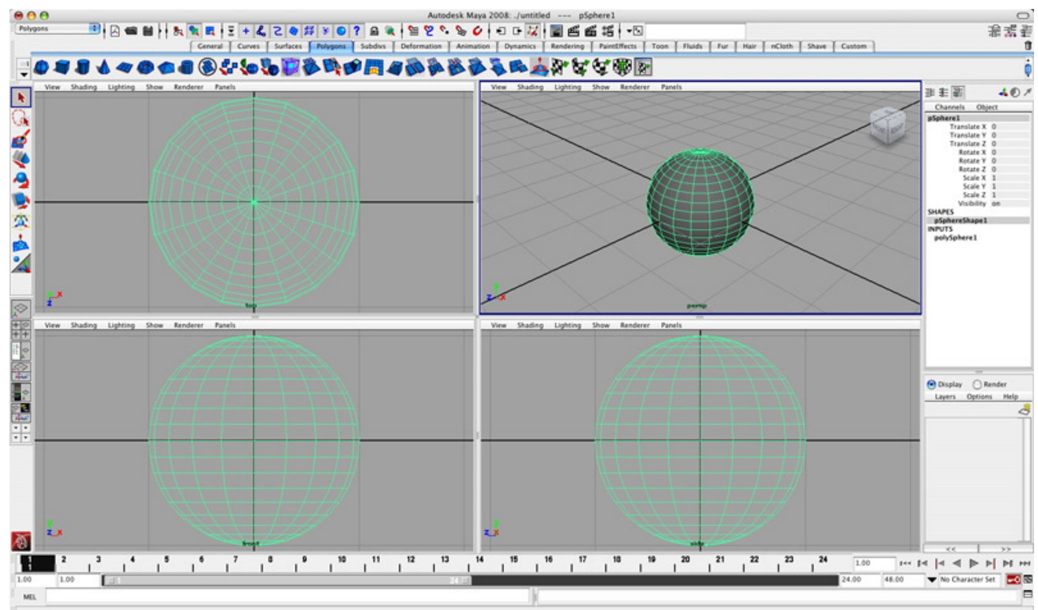






Modeling

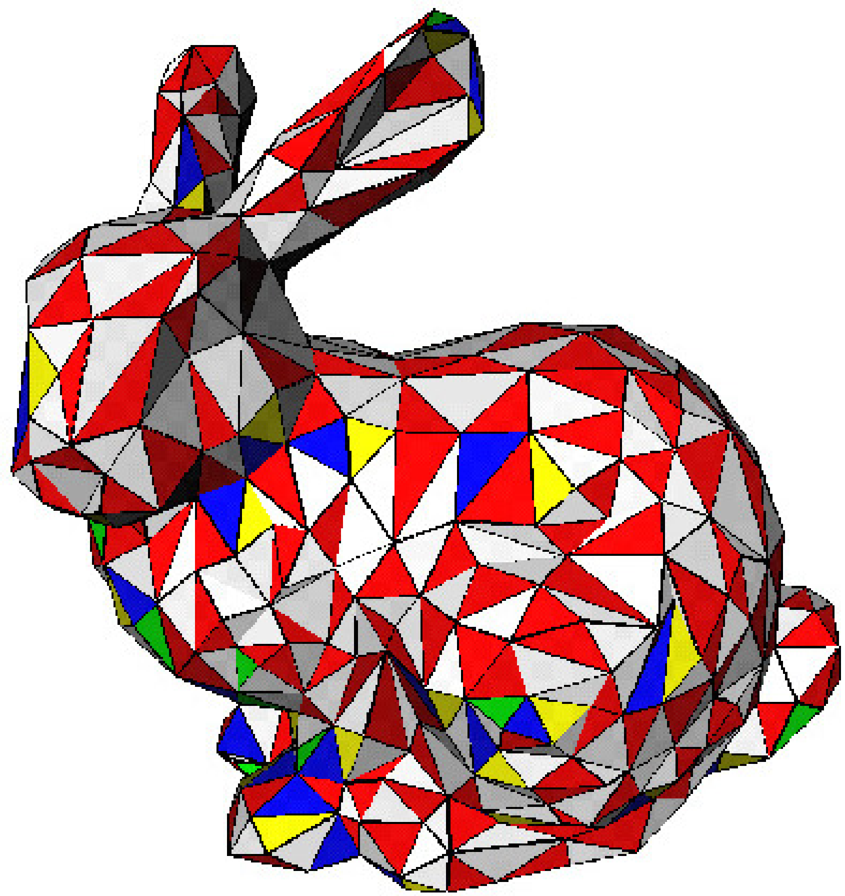
- Creating 3D geometric data
  - The "model" or the "scene"
- By hand
  - Autodesk (Maya, AutoCAD), LightWave 3D, …
- Free software
  - Blender
- Not as easy to use as Notepad…



Maya Screenshot

Modeling

- Basic 3D models consist of array of triangles
- Each triangle stores 3 vertices
- Each vertex contains
  - xyz position
  - Color
  - Etc.



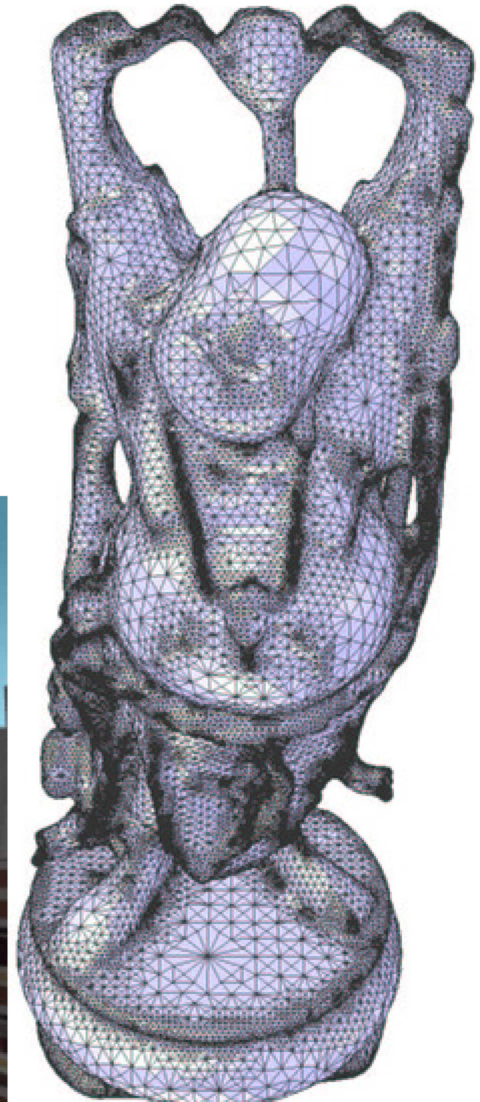
Modeling

- Procedural: by writing programs
- Scanning real-world objects

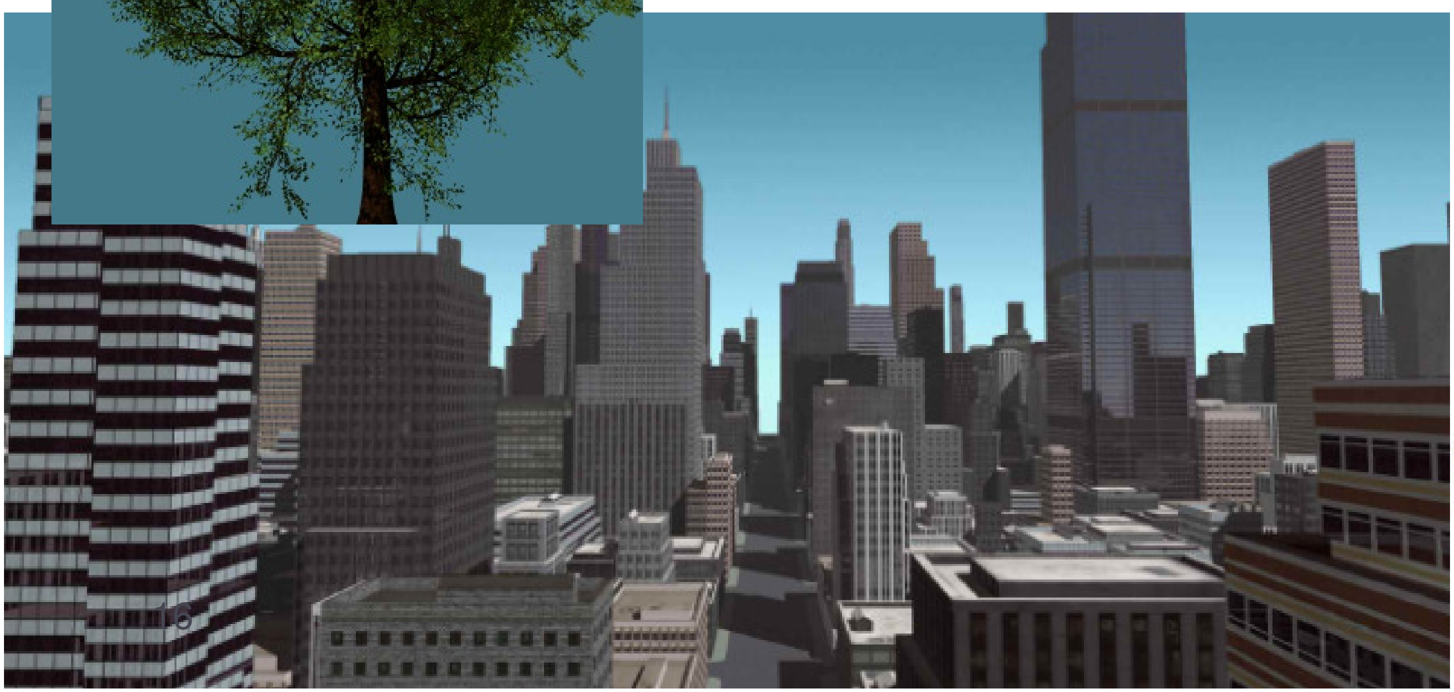
Modeling

Procedural tree

Scanned statue



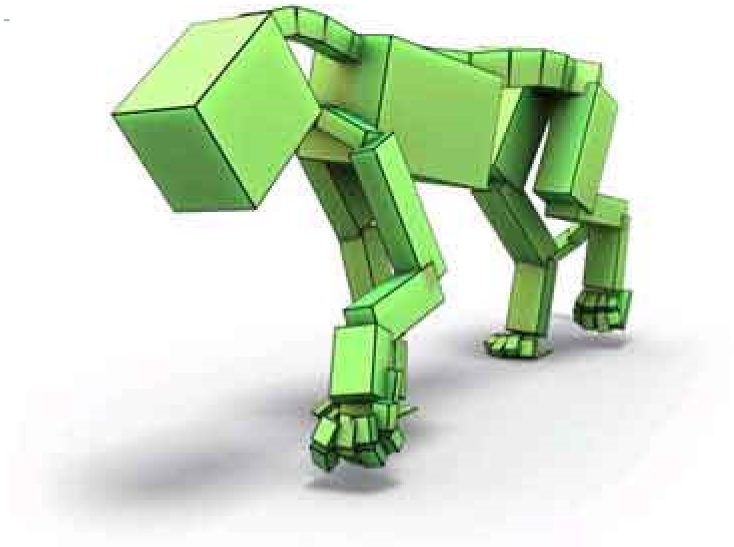
Procedural city

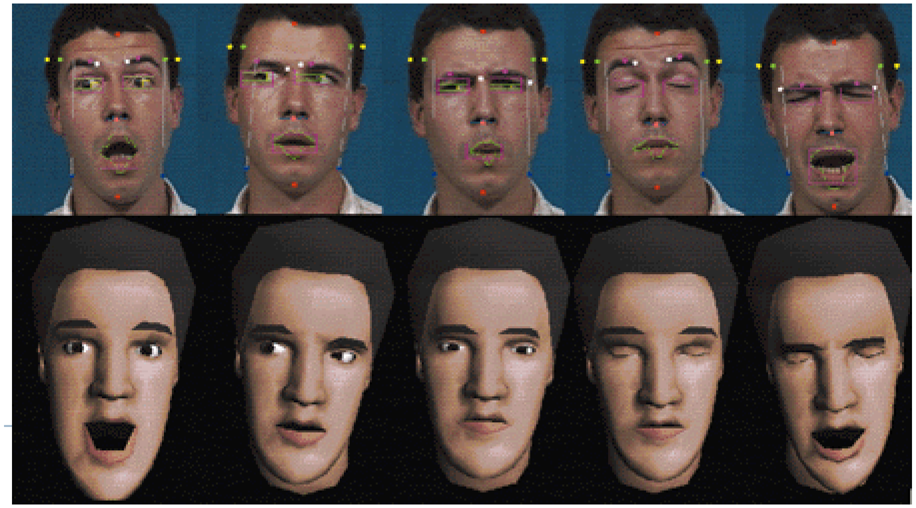


Animation

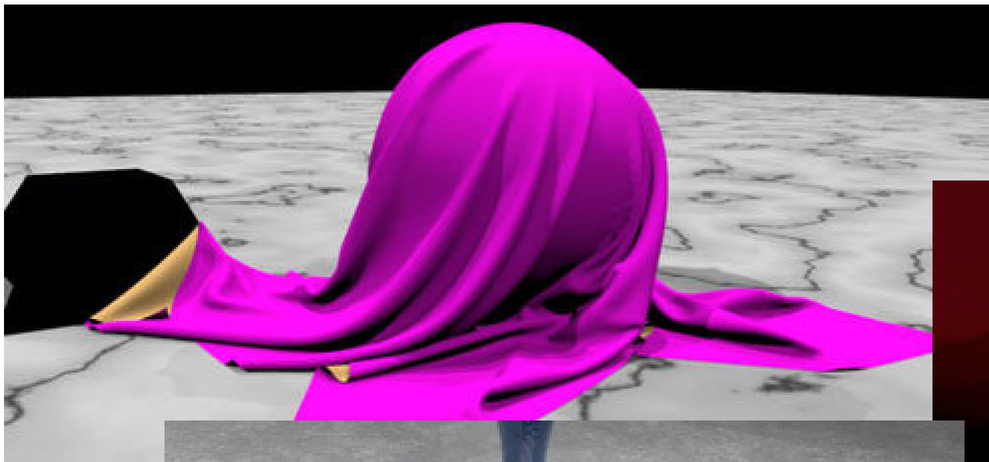
- Deforming or editing the geometry
- Change over time
- Faces, articulated characters, …
- CSE169: Computer Animation (not offered this year)

Animation





Physics simulation



Video

- SIGGRAPH 2011 Technical Papers:
  http://www.youtube.com/watch?v=JK9EEE3RsKM
- Blender Demo Reel 2011:
  http://www.youtube.com/watch?v=QbzE8jOO7_0

Today

- Course overview
- Course organization
- Vectors and Matrices

Course Staff

Instructor

- Jürgen Schulze, Ph.D.
  Lecturer in CSE, Research Scientist at Calit2

Teaching Assistants

- Gregory Long, CSE graduate student
- Jorge Schwarzhaupt, CSE graduate student

Course Organization

Lecture

- Tue/Thu, 2:00pm-3:20pm, Peterson Hall 104

Homework Grading

- Fridays (only on due dates) at 1:30pm, CSE lab 260

Instructor Office Hour

- Tue 3:30pm-4:30pm, Atkinson Hall room 2125

Office Hours in Lab 260

- Gregory Long:
  Tue 12-1p, Wed 4:30-6:30p, Thu 5-7p
- Jorge Schwarzhaupt:
  Tue 1-2p, Wed 3:30-5:30p, Thu 5-7p

Prerequisites

Familiarity with

- Linear algebra
- C++
- Object oriented programming

In this class

- Rendering 3D models
  - Camera simulation
  - Interactive viewing
  - Lighting
  - Shading
- Modeling
  - Triangle meshes
  - Parametric surfaces
- Applying linear algebra, C++, OpenGL
- Foundation for advanced graphics courses (eg, CSE168, CSE 190 on shader programming)

Ted

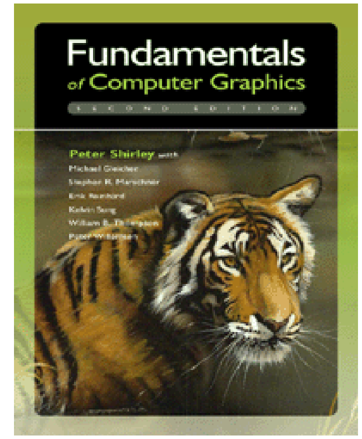
- Go to
  `http://ted.ucsd.edu`
  and select CSE167
- Log in with your Active Directory account
- Used for discussion board and grades

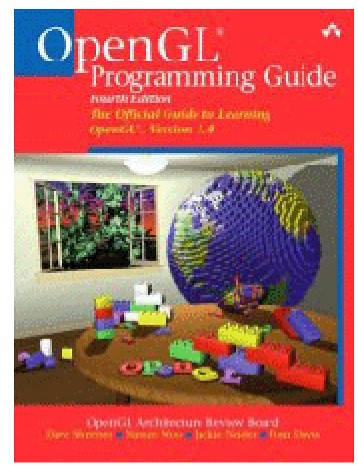
Course Web Site

- URL: http://ivl.calit2.net/wiki/index.php/CSE_167_Fall_2011
- Class schedule
- Lecture slides
- Textbooks
- Announcements
- Homework assignments
- Grading information (grades on Ted)

Textbooks

- Both textbooks are recommended, not required
- Peter Shirley: *Fundamentals of Computer Graphics*, any edition (Google Books has full text version)
- *OpenGL Programming Guide*
  Older versions available on-line





Programming Projects

- 7 programming assignments
- First and last are group projects
- Find assignments and schedule on Ted
- Base code (for Windows and Linux) and documentation on Ted
- Use EBU3B 2xx labs or your own PC/laptop
- Individual assistance by TAs during lab office hours
- Turn in by demonstration to TA during homework grading hours on Fridays. Demonstration can be done on lab PC or personal computer.
- Homework projects are due by Fridays 1:30pm.
  Late submissions possible with point deduction.

Written Tests

Two in-class written tests.

Closed book, handwritten index card is permitted.

Midterm exam:

- Thu 10/27, 2:00pm-3:20pm, Peterson Hall 104

Final exam:

- Thu 12/08, 3:00pm-6:00pm, location TBD

Grading

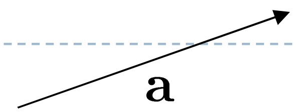
- Homework Projects 1-6: 10% each
- Written exams: 10% each
- Final project: 20%
- Late submission policy for homework projects:

  75% of original grade if you present your project within seven days of the due date

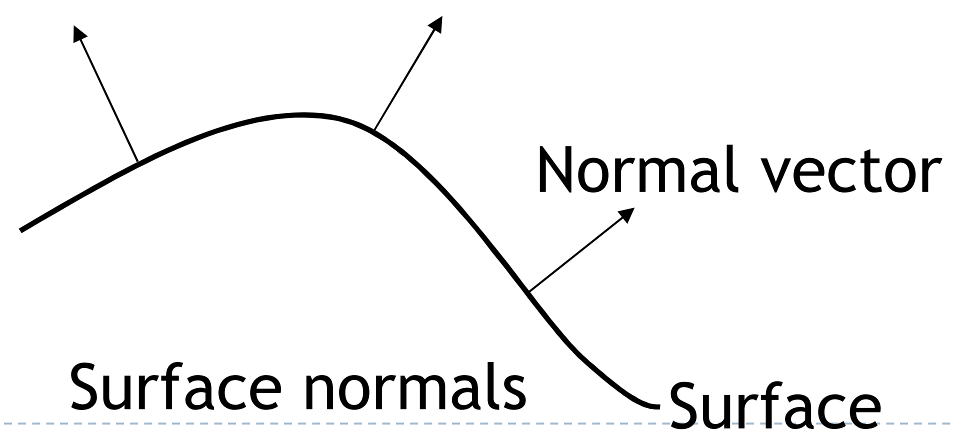
Today

- Course overview
- Course organization
- Vectors and Matrices

Vectors



- Direction and length in 3D
- Vectors can describe
  - Difference between two 3D points
  - Speed of an object
  - Surface normals (directions perpendicular to surfaces)



Vector arithmetic using coordinates

$$\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} a_x + b_x \\ a_y + b_y \\ a_z + b_z \end{bmatrix} \qquad \mathbf{a} - \mathbf{b} = \begin{bmatrix} a_x - b_x \\ a_y - b_y \\ a_z - b_z \end{bmatrix}$$

$$-\mathbf{a} = \begin{bmatrix} -a_x \\ -a_y \\ -a_z \end{bmatrix} \qquad s\mathbf{a} = \begin{bmatrix} sa_x \\ sa_y \\ sa_z \end{bmatrix}$$

where s is a scalar

Vector Magnitude

- The magnitude (length) of a vector is:

$$|\mathbf{v}|^2 = v_x^2 + v_y^2 + v_z^2$$

$$|\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

- A vector with length of 1.0 is called *unit vector*
- We can also *normalize* a vector to make it a unit vector

$$\frac{\mathbf{v}}{|\mathbf{v}|}$$

- Unit vectors are often used as **surface normals**

Dot Product

$$\mathbf{a} \cdot \mathbf{b} = \sum a_i b_i$$

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z$$

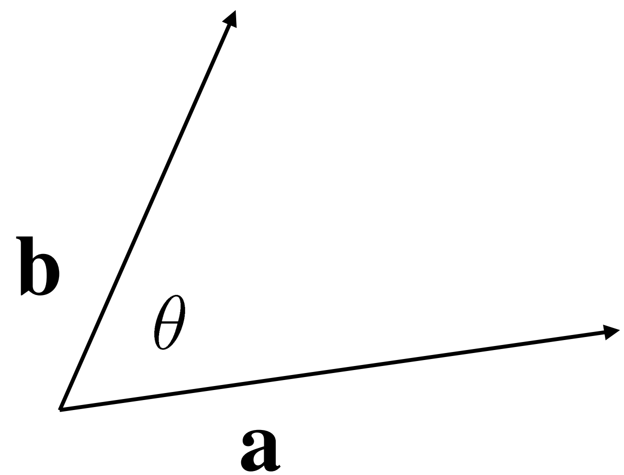
$$\mathbf{a} \cdot \mathbf{b} = |a||b| \cos \theta$$

Angle Between Two Vectors

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$

$$\cos\theta = \left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right)$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right)$$



Cross Product

$\mathbf{a} \times \mathbf{b}$ is a vector *perpendicular* to both $\mathbf{a}$ and $\mathbf{b}$, in the direction defined by the right hand rule

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}||\mathbf{b}|\sin\theta$$

$|\mathbf{a} \times \mathbf{b}| =$ area of parallelogram $\mathbf{ab}$

$|\mathbf{a} \times \mathbf{b}| = 0$ if $\mathbf{a}$ and $\mathbf{b}$ are parallel (or one or both degenerate)

Cross product

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

Sample Vector Class in C++

```
class Vector3 {
public:
    float x,y,z;
    Vector3()                                   {x=0.0; y=0.0; z=0.0;}
    Vector3(float x0,float y0,float z0)         {x=x0; y=y0; z=z0;}
    void set(float x0,float y0,float z0)        {x=x0; y=y0; z=z0;}
    void add(Vector3 &a)                        {x+=a.x; y+=a.y; z+=a.z;}
    void add(Vector3 &a,Vector3 &b)             {x=a.x+b.x; y=a.y+b.y; z=a.z+b.z;}
    void subtract(Vector3 &a)                   {x-=a.x; y-=a.y; z-=a.z;}
    void subtract(Vector3 &a,Vector3 &b)        {x=a.x-b.x; y=a.y-b.y; z=a.z-b.z;}
    void negate()                               {x=-x; y=-y; z=-z;}
    void negate(Vector3 &a)                     {x=-a.x; y=-a.y; z=-a.z;}
    void scale(float s)                         {x*=s; y*=s; z*=s;}
    void scale(float s,Vector3 &a)              {x=s*a.x; y=s*a.y; z=s*a.z;}
    float dot(Vector3 &a)                       {return x*a.x+y*a.y+z*a.z;}
    void cross(Vector3 &a,Vector3 &b)
            {x=a.y*b.z-a.z*b.y; y=a.z*b.x-a.x*b.z; z=a.x*b.y-a.y*b.x;}
    float magnitude()                   {return sqrt(x*x+y*y+z*z);}
    void normalize()                    {scale(1.0/magnitude());}
};
```

Matrices

- Rectangular array of numbers

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,n} \\ m_{2,1} & m_{2,2} & \dots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{m,1} & m_{2,2} & \dots & m_{m,n} \end{bmatrix} \in \mathbf{R}^{m \times n}$$

- Square matrix if **m = n**
- In graphics often **m = n = 3; m = n = 4**

Matrix Addition

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} & \dots & a_{1,n} + b_{1,n} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} & \dots & a_{2,n} + b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} + b_{m,1} & a_{2,2} + b_{2,2} & \dots & a_{m,n} + b_{m,n} \end{bmatrix}$$

$$\mathbf{A}, \mathbf{B} \in \mathbf{R}^{m \times n}$$

Multiplication With Scalar

$$s\mathbf{M} = \mathbf{M}s = \begin{bmatrix} sm_{1,1} & sm_{1,2} & \dots & sm_{1,n} \\ sm_{2,1} & sm_{2,2} & \dots & sm_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ sm_{m,1} & sm_{2,2} & \dots & sm_{m,n} \end{bmatrix}$$

Matrix Multiplication

$$\mathbf{AB} = \mathbf{C}, \quad \mathbf{A} \in \mathbf{R}^{p,q}, \mathbf{B} \in \mathbf{R}^{q,r}, \mathbf{C} \in \mathbf{R}^{p,r}$$

$$(\mathbf{AB})_{i,j} = \mathbf{C}_{i,j} = \sum_{k=1}^{q} a_{i,k} b_{k,j}, \quad i \in 1..p, j \in 1..r$$

Matrix-Vector Multiplication

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \quad \mathbf{A} \in \mathbf{R}^{p,q}, \mathbf{x} \in \mathbf{R}^{q}, \mathbf{y} \in \mathbf{R}^{p}$$

$$(\mathbf{A}\mathbf{x})_i = \mathbf{y}_i = \sum_{k=1}^{q} a_{i,k} x_k$$

Identity Matrix

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \in \mathbf{R}^{n \times n}$$

$$\mathbf{MI} = \mathbf{IM} = \mathbf{M}, \quad \text{for any } \mathbf{M} \in \mathbf{R}^{n \times n}$$

Matrix Inverse

If a square matrix **M** is non-singular, there exists a unique inverse **M**$^{-1}$ such that

$$\mathbf{M}\mathbf{M}^{-1} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$$

$$(\mathbf{MPQ})^{-1} = \mathbf{Q}^{-1}\mathbf{P}^{-1}\mathbf{M}^{-1}$$

OpenGL Matrices

- Vectors are column vectors
- "Column major" ordering
- Matrix elements stored in array of floats

  float M[16];
- Corresponding matrix elements:

$$\begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix}$$

Announcements

- Next Lecture
  - Tue 9/27 at 2pm
  - Topic: Homogeneous Coordinates
  - Preparation:
    Review three dimensional vector/matrix calculations

- Homework Introduction (optional):
  Introduction to base code and homework assignment #1:
  Gregory Long, CSE lab 260, Monday Sept 26th, 3pm

- Homework assignment #1 due Friday, Sept 30