# CSE167
Introduction to Computer Graphics

Matthias Zwicker
University of California, San Diego
Spring 2008

## Course staff

**Instructor**
- Matthias Zwicker
- Assistant professor at UCSD CSE since 2006

**Teaching Assistant**
- Neel Joshi
- 4th year CSE PhD student

## Today
- Course organization
- Course overview
- Linear algebra review

## Course organization

### Instructor
- Matthias Zwicker ([mzwicker@cs.ucsd.edu](mailto:mzwicker@cs.ucsd.edu))

### Teaching Assistant
- Neel Joshi ([njoshi@cs.ucsd.edu](mailto:njoshi@cs.ucsd.edu))

### Lecture
- Tue/Thur, 5:00pm-6:20pm, WLH 2111

### Lab hours
- Wed, 1:30pm-4:30pm, EBU3B B210
- Additional lab hours, schedule tbd
Course organization

Class web page
- [http://graphics.ucsd.edu/courses/cse167_s08](http://graphics.ucsd.edu/courses/cse167_s08)
- Schedule, slides, reading, project descriptions, etc.

Course organization

Webboard
- [http://webboard.ucsd.edu/](http://webboard.ucsd.edu/)
- Use your network user/PID

Textbooks

Textbooks
- Fundamentals of Computer Graphics, Peter Shirley, 2nd edition (required)
- OpenGL Programming Guide, Shreiner, Woo, Neider, Davis, 5th edition (recommended)

Programming Projects

Build your own 3D rendering engine
- **Project 1**: Matrices, Vectors, and Coordinate Transformations
- **Project 2**: Interactive Viewing
- **Project 3**: Rasterization
- **Project 4**: Lighting and Texturing
- **Project 5**: Scene Graphs
- **Project 6**: Shader Programming
- **Final Project**

Programming Projects

Assignments and schedule on class webpage
- Base code (for Windows platform) and documentation on class webpage
- Use EBU3B 2xx labs or your own PC
- Turn-in by demonstration to TA during lab hours
- Additional lab hours
  - TA presents new project
  - TA provides individual assistance

Tests

Midterm
- In class
- Tentatively Tue 05/06
Final
- Finals week, TBD
**Grading**
- Project 1-6: 10% each
- Final project: 15%
- Midterm: 10%
- Final exam: 15%
- Late policy for projects
  - 75% of original grade if you present your project the following week

**Prerequisites**
Basic familiarity with
- Linear algebra
- C++ (if you know Java you’ll be able to adapt)
- Object oriented programming

---

**Questions?**

---

**Today**
- Course organization
  - Course overview
  - Linear algebra review

---

**What is computer graphics?**

**Applications**

---

**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

### 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
Artistic rendering

- Stylized
- Artwork, illustrations, data visualization

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...

Modeling

- Procedurally, i.e. by writing programs
- Scanning real-world objects

Modeling

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

- Deforming or editing the data
- Change over time
- Faces, articulated characters, ...
- CSE169

Physics simulation

In this class
The Basics...
- Rendering 3D models
  - Camera simulation
  - Interactive viewing
  - Lighting, shading
- Modeling
  - Triangle meshes
  - Smooth surfaces
- Experience with linear algebra, C++, OpenGL
- Background for advanced topics (CSE168, CSE169)

Questions?

Today
- Course organization
- Course overview
- Linear algebra review
Linear algebra review
Why linear algebra?
• Need to describe 3D scenes
  - Position, orientation, motion of objects
  - Relation of objects to virtual camera
  - Projection of scene onto image plane
• Linear algebra provides mathematical tools

Topics today
• Vectors
• Coordinate systems
• Vector arithmetic
• Vector magnitude
• Dot product

Vectors
• Direction and length in 3D
• Vectors can describe
  - Difference between two 3D points
  - Speed of an object
• Vectors are in bold-face

Vectors
Multiplication by scalar

Vectors
Addition

Vectors
Addition
**Vectors**

Linear combination

\[ sa + tb, \quad s, t \in \mathbb{R} \]

\[ \sum_{i=1}^{n} s_i a_i, \quad s_i \in \mathbb{R} \]

Linearly dependent vectors

- A set of vectors \( a_i, i = 1 \ldots n \) is linearly dependent if there exist scalars \( s_i \) such that

\[ a_j = \sum_{i=1,i\neq j}^{n} s_i a_i \]

- Otherwise, they are linearly independent

**Coordinate systems**

- Describe any vector with respect to three basis vectors \( x, y, z \)

\[ a = a_x x + a_y y + a_z z \]

- The basis vectors form a coordinate system

**Coordinate systems**

- Any three vectors that are linearly independent could be used as a basis
  - Different lengths
  - Not perpendicular to each other

- Why linearly independent?
- Why exactly three vectors?
- Other coordinate systems?

**Coordinate systems**

Euclidean coordinate systems

- Basis vectors
  - Have unit length
  - Are perpendicular to each other
- Orthonormal
Coordinate Systems

Handedness

Right handed

Left handed

Vector arithmetic using coordinates

\[
\begin{align*}
\mathbf{a} &= \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, & \mathbf{b} &= \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}, \\
\mathbf{a} + \mathbf{b} &= \begin{bmatrix} a_x + b_x \\ a_y + b_y \\ a_z + b_z \end{bmatrix}, & \mathbf{a} - \mathbf{b} &= \begin{bmatrix} a_x - b_x \\ a_y - b_y \\ a_z - b_z \end{bmatrix} \\
-\mathbf{a} &= \begin{bmatrix} -a_x \\ -a_y \\ -a_z \end{bmatrix}, & \mathbf{s}\mathbf{a} &= \begin{bmatrix} sa_x \\ sa_y \\ sa_z \end{bmatrix}
\end{align*}
\]

Questions?

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=1.0 is called a unit vector
- We can also normalize a vector to make it a unit vector
  \[ \mathbf{v} \rightarrow \frac{\mathbf{v}}{|\mathbf{v}|} \]
- Unit vectors are often used as surface normals

Dot Product

- The dot product is a scalar value that tells us something about the relationship between two vectors
- Angles between vectors
- Lengths of vectors

Dot product

- If \( \mathbf{a} \cdot \mathbf{b} > 0 \) then \( \theta < 90^\circ \)
  - Vectors point in the same general direction
- If \( \mathbf{a} \cdot \mathbf{b} < 0 \) then \( \theta > 90^\circ \)
  - Vectors point in opposite direction
- If \( \mathbf{a} \cdot \mathbf{b} = 0 \) then \( \theta = 90^\circ \)
  - Vectors are perpendicular
  - (or one or both of the vectors is degenerate (0,0,0))
Dot Product

- Using coordinates
  \[ \mathbf{a} \cdot \mathbf{b} = \sum a_i b_i \]
  \[ \mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z \]
  \[ \mathbf{a} \cdot \mathbf{b} = |a||b| \cos \theta \]

Next class

- Matrices and transformations
- Lab tomorrow: introduction to the base code
  - If you can’t find Neel, make sure to check in all labs EBU3B 2xx