ECS 175 Introduction to Computer Graphics

Time & Place:

Tue & Thu 12:10–1:30pm
Fri 4:10–5:00pm (Discussion)
205 Olson

Instructor:

Kwan–Liu Ma
Office: Room 3025, Eng II
Office Hours: Tue 10:30–11:30am
Thu 2:00–3:00pm
(530) 752–6958
ma@cs.ucdavis.edu

TA:

Oliver Kreylos
Room 067, Eng II
Office Hours: TBA
kreylos@cs.ucdavis.edu

Prerequisite:

ECS110 or ECE73

- C, C++ programming
- Data structures
- Algorithms

Math 22A

- Linear algebra

Good problem solving skill
Good debugging skill

Course home Page:

http://www.cs.ucdavis.edu/~ma/ECS175_S00

Course newsgroups:

ucd.class.ecs175
ucd.class.ecs175.d

Textbook:

Required:

Computer Graphics, C Edition,
Hearn and Baker, Prentice Hall

Optional:

OpenGL Programming Guide, 3rd ed.
Woo, et al., Addison Wesley

Other userful references:

Notes:

UCD Graphics Research Group's class notes and C++ classes

http://graphics.cs.ucdavis.edu/Notes.html

OpenGL Tutorial

http://www.eecs.tulane.edu/www/Terry/OpenGL/Introduction.html

Tools:

Mesa3D website

3−d graphics library with OpenGL API

http://mesa3d.sourceforge.net/

Xforms websidte

http://bragg.phys.uwm.edu/xforms

Cygwin

Ports of the popular GNU development tools for Windows NT, 95, and 98.

http://sourceware.cygnus.com/cygwin/

Bloodshed Dev−C++

An IDE and C++ compiler for Win9x environment

http://www.bloodshed.nu/

Computer Graphics
What, Why, and How?

What?

- **Computer modeling and rendering of real or arteficial objects and the interaction between them.**

Why?

- **Engineering design**
- **Presentation Graphics**
- **Computer Art**
- **Entertainment**
- **Education and Training**
- **Visualization**
- **...**

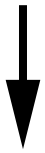
How?

- **Take 175, and other classes offered by us.**

History of 3−D Graphics

Early 1960 at MIT

1st interactive graphics program
"sketchpad" by Ivan Sutherland



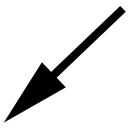
Sutherland joined the Harvard faculty
with Denny Cohen
1st 3−d flight simulator

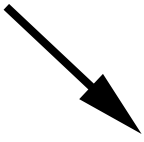
1968 Dave Evans and his students at Utah

Shaded image generation

Sutherland joined Utah faculty

The roots of 3−d graphics developed





Photo−realistic graphics

Modeling

Real−time graphics

Photo−realistic Graphics

Local Illumination

1971, Henry **Gouraud**, Utah

1975, Bui Tong **Phong**, Utah

used smooth −shading tricks to make the eye see smooth objects modeled as polygonal surfaces

1974, Ed Catmull, Utah

1st picture of mathematically smooth surfaces

1976, developed the concept of **Z−buffer**

also invented Alpha blending and texture mapping

NYIT

Pixar

Photo−realistic Graphics

Global Illumination

↓

Early 1980, Turner Whitted

Ray Tracing

Reflections and directional effects of light

↓

Mid 1980, Don Greenberg. Cornell

Radiosity

diffuse effects of light

↓

Integrating ray tracing and radiosity

↓

Photo−realism
Exciting special effects

↓

Modeling Techniques

Structural modeling

Hierarchical modeling

Curve and surface modeling

Solid modeling

Fractal modeling

Soft Object modeling

Grammar−based modeling

Procedural modeling

Texture modeling

Volume modeling

Image−based modeling

Realtime Graphics
High Performance Graphics

1970s E&S
Flight Simulators → 2000 Sony Playstation 2
2001 Microsoft XBox

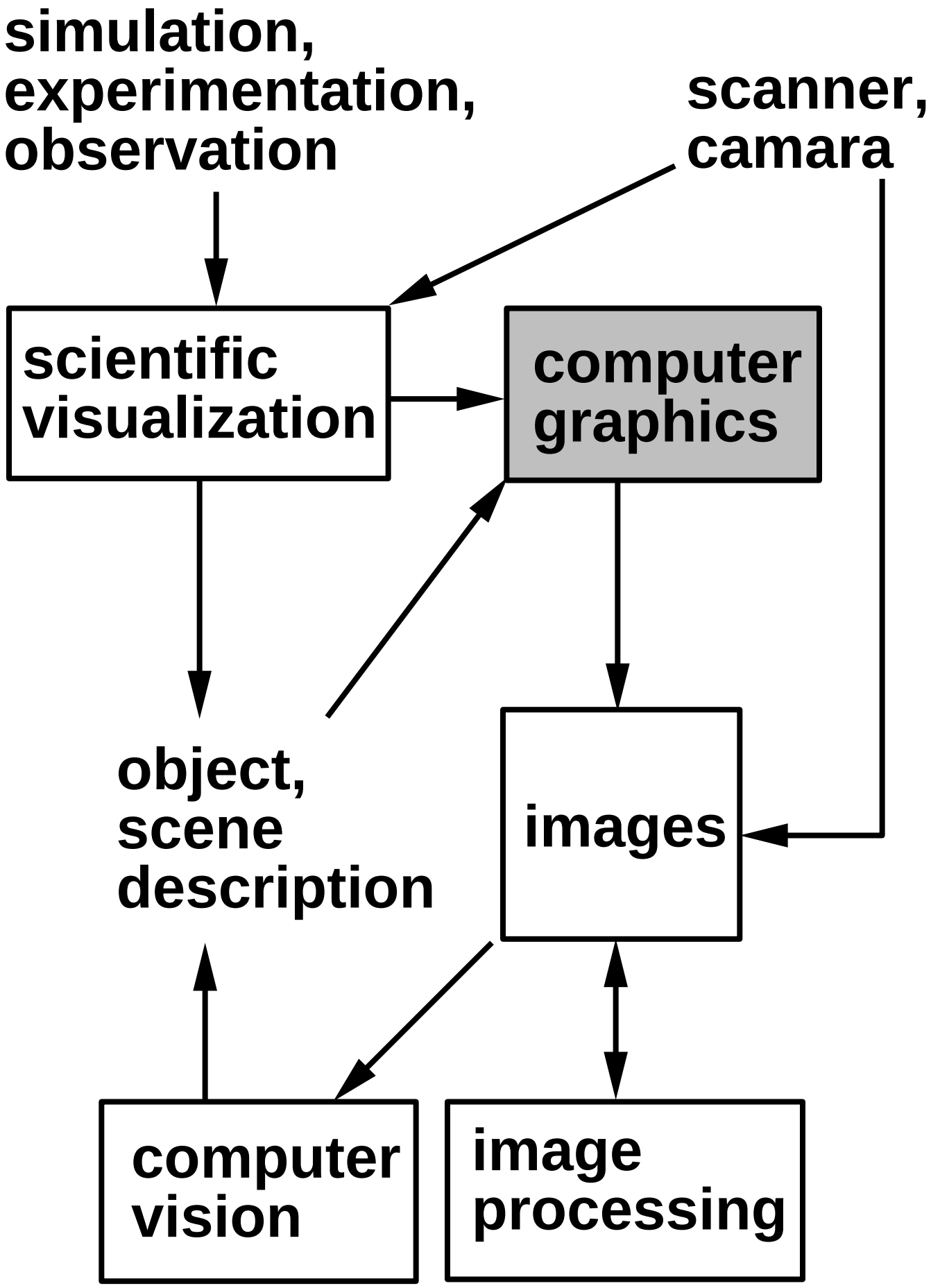
SGI (1K polygons/sec)

→ (100s million polygons/sec)

Graphics supercomputing

→ Rendering farms

→ Commodity parallel rendering


simulation,
experimentation,
observation
scanner,
camara
scientific
visualization
computer
graphics
object,
scene
description
images
computer
vision
image
processing

What will you learn in this class?

1. Computer graphics jargons
2. Transformation geometry
3. Rasterization
4. 3−d viewing
5. Shading
6. Texture mapping
7. Antialiasing
8. Hierarchical modeling
9, Curves and surfaces
10. Overview of advanced techniques
    - Ray tracing
    - Radiosity
    - Animation
    - Scientific Visualization
    - ...

Where to learn more?

275 Advanced Computer Graphics

177 Intro. to Scientific Visualization
277 Advanced Visualization
276 Advanced Volume Visualization

178 Intro. to Geometric Modeling
278 Advanced Geometric Modeling

Animation

Computational Geometry

Virtual Reality

163 Human Computer Interfaces

Research projects

Join us!!
UCD Visualization and Graphics Research Group

http://graphics.cs.ucdavis.edu

Programming Assignments

1. 2−d rasterization (15%)
2. 3−d viewing (15%)
3. Local illumination (15%)
4. Texture mapping & antialiasing (20%)

grading:

- completeness & correctness (70pt)
- interface & interaction mechanism (15pt)
- program structure & documentation (15pt)
- additinal features (up to 10pt)

No assignment will be accepted if it is incomplete!

Midterm (25%) **Quizs (10%)**

Sample Grades

90% − A 80% − B
70% − C 60% − D 50% − F

Regrade:

regrades must be turned in no later than one week aftern the graded paper were made avialable

Turn−in Procedure

The assignments are to be e−mailed to the reader/TA, according to the following instructions:

A README and Makefile is required for ECS 175. The README should describe how to compile the programs (with the Makefile), how to operate the programs, and the output of the program that we should expect. No executable files, nor ‘‘.o’’ files should be sent to the reader.

Package your files for e-mail as follows.

a) put all your source, input, README, and Makefile files in a directory named unameN, where uname is your CS login name, and N is the assignment number. (example: if your login name is joe, and you are turning in assignment 3, you want to put the files under the directory joe3) Then do the following from the shell prompt (replace unameN with the appropriate username and assignment number) :

b) tar cvf uidN.tar uidN

c) uuencode unameN.tar unameN.tar > unameN.uu

d) mail cs175r@cs.ucdavis.edu < unameN.uu

Policies, Cheating and Plagiarism

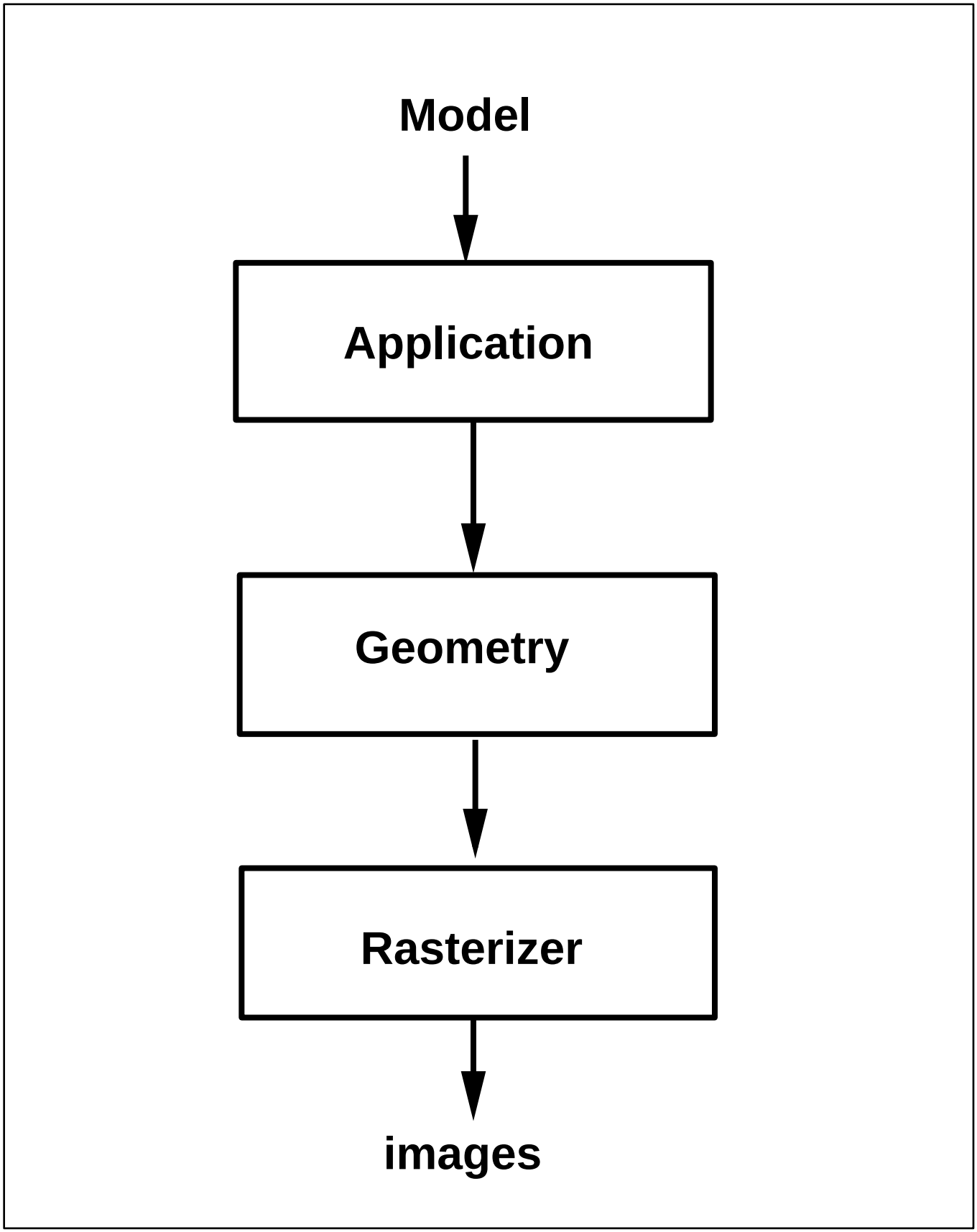
Produce your own code!!!!

Last day to drop

April 27

Waiting List

Graphics Rendering Pipeline



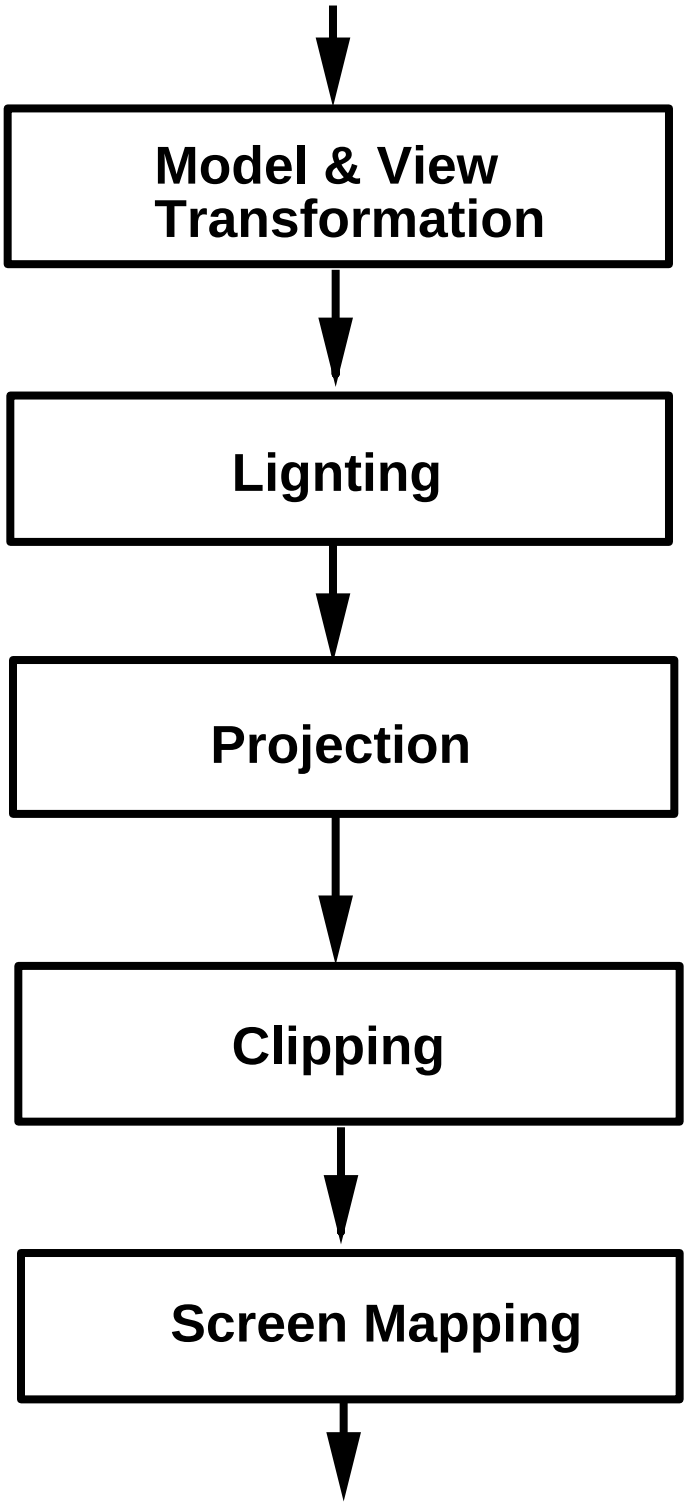
Graphics Rendering Pipeline

The Application Stage

- Interact with the model, user inputs
- Decimation
- Hierarchical view frustum culling
- Collision detection
- Geometry morphing
- Implemented in software!

Graphics Rendering Pipeline

The Geometry Stage



Graphics Rendering Pipeline

The Rasterizer Stage

- Rasterization or Scan Conversion
- Assign correct colors to the pixels
- Texturing
- Double buffer
- Z−buffer
- Transparency?
- Stencil buffer, A−buffer, ...
- Implemented in hardware!