CS130: Computer Graphics
Spring 2012

Tamar Shinar
Computer Science & Engineering
UC Riverside
Welcome to CS130!

Talton et al., 2011

Schröder, 2000

LLNL

Hong et al. 2007

ILM

Pixar
Today’s agenda

• Course logistics
• Introduction: graphics areas and applications
• Introduction to OpenGL
• Math review
Course Overview

• Learn fundamental 3D graphics concepts
• Implement graphics algorithms
  • make the concepts concrete
• expand your abilities and confidence for future work
Course Logistics

- Instructor: Tamar Shinar
- TA: Nam Nguyen
- Website: [http://www.cs.ucr.edu/~nnguyen/cs130](http://www.cs.ucr.edu/~nnguyen/cs130)
- Lectures: MWF 8:10-9am
- Lab: M 2:10-5:00pm, WCH 127
- announcements (assignments, etc.) made in class and on course website
Course Logistics

- Grading
  - 10% labs
  - 10% homework
  - 30% assignments (2 assignments, 15% each)
  - 50% tests (2 midterms, 1 final)
- Detailed schedule on class website
## Course schedule

tentative; see course website for up-to-date schedule

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Date</th>
<th>Topic</th>
<th>Reading</th>
<th>Assigned</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4/2</td>
<td>Introduction</td>
<td>Chapters 1-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4/4</td>
<td>Graphics Pipeline</td>
<td>Chapter 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4/6</td>
<td>Graphics Pipeline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab 1</td>
<td>4/2</td>
<td>Introduction to OpenGL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4/9</td>
<td>2D lines and circles</td>
<td>Chapter 8.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4/11</td>
<td>Modeling and Rendering Curves</td>
<td>Chapter 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4/13</td>
<td>Modeling and Rendering Curves</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab 2</td>
<td>4/9</td>
<td>Line Rasterization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4/16</td>
<td>Transformations</td>
<td>Chapters 5-6</td>
<td>Assignment 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4/18</td>
<td>Transformations (cont.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4/20</td>
<td>Transformations (cont.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab 3</td>
<td>4/16</td>
<td>Transformations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4/23</td>
<td>Polygons</td>
<td>Chapters 2.7, 8.1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4/25</td>
<td>Polygons (cont.) and Review</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>4/27</td>
<td>Test 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab 4</td>
<td>4/23</td>
<td>Modeling with Maya</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4/30</td>
<td>Rotations</td>
<td>Chapter 17.2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>5/2</td>
<td>Rotations (cont.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>5/4</td>
<td>Projections</td>
<td>Chapter 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lab 5</td>
<td>4/30</td>
<td>SLERP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5/7</td>
<td>Projection (cont.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5/9</td>
<td>Shading</td>
<td>Chapter 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>5/11</td>
<td>Shading (cont.)</td>
<td></td>
<td>Assignment 1</td>
<td></td>
</tr>
<tr>
<td>Lab 6</td>
<td>5/7</td>
<td>Programmable Shading</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Textbook

Fundamentals of Computer Graphics
Shirley and Marschner

Additional books

if you like using a book
- red book older version online: http://fly.cc.fer.hr/~unreal/theredbook/
And if you prefer -- all material is online in one form or another -- you don’t have to buy a book but it can be useful for a coherent presentation
About your instructors

- B.S., University of Illinois in Urbana-Champaign, Mathematics, Computer Science, Art
- Ph.D., 2008, Stanford University on simulation methods for computer graphics
- Started at UCR in the fall
- Work in graphics simulation and biological simulation

http://www.cs.ucr.edu/~shinar
http://www.cs.ucr.edu/~nnguyen
Introduction
Graphics applications

- 2D drawing
- Drafting, CAD
- Geometric modeling
- Special effects
- Animation
- Virtual Reality
- Games
- Educational tools
- Surgical simulation
- Scientific and information visualization
- Fine art
Graphics areas

- **Modeling** - mathematical *representations* of physical objects and phenomena
- **Rendering** - creating a *shaded image* from 3D models
- **Animation** - creating motion through a sequence of images
- **Simulation** - physics-based models for modeling dynamic environments

Think about which area interests you, dovetails with your present or future research, or that you want to learn more about.

*Modeling* and *rendering* are separate stages:
- first design and position objects -- *modeling*
- then add lights, materials properties, effects -- *rendering*
- subdivision surface – Siggraph course notes 2000
- Teddy: sketch based interface for 3D modeling
- Talton et al. – procedural modeling – for games, virtual worlds, design, etc.
  - combine machine learning and graphics
- Bronstein – reasoning about geometric models for search
Rendering

- opengl – 3D graphics (z-buffer) rendering
- teapot – image-based lighting – illuminated by a high dynamic range environment – metal, glass, diffuse, and glossy
- subsurface scattering – to capture translucent materials such as skin and marble
- rendering a emissive material such as fire – participating medium – scattering, absorption
- local vs global illumination

Hong et al. 2007

d’Eon and Irving, 2011
- direct vs. global illumination
- direct vs. global illumination
Animation

Sleeping Beauty, Disney, 1959

Adventures of Tintin, Weta 2011
Animation

Sleeping Beauty, Disney, 1959

Adventures of Tintin, Weta 2011
Simulation

ILM

Pixar

Weta

© Disney
Firestorm
Harry Potter and the Half Blood Prince
Industrial Light + Magic
fluid simulation in Pixar’s *Ratatouille*
fluid simulation in Pixar’s Ratatouille
Introduction to OpenGL
Introduction to OpenGL

- Open Graphics Library, managed by Khronos Group
- A software interface to graphics hardware
- Standard API with support for multiple languages and platforms, open source
- ~250 distinct commands
- Main competitor: Microsoft’s Direct3D

http://www.opengl.org/wiki/Main_Page

- used to produce interactive 3D graphics
- sits between programmer and 3D accelerators and hardware
- **standard** requires support for feature set for all implementations
- Both OpenGL and Direct3D support feature sets -- they take advantage of hardware acceleration or use software emulation when a feature is unavailable in hardware
- Direct3D is proprietary
- OpenGl and Direct3D both implemented in the display driver
Open GL - Software to Hardware

• Silicon Graphics (SGI) revolutionized the graphics workstation by putting graphics pipeline in hardware (1982)
• To use the system, application programmers used a library called GL
• With GL, it was relatively simple to program three dimensional interactive applications
The success of GL lead to OpenGL (1992), a platform-independent API that was
- Easy to use
- Close to the hardware - excellent performance
- Focus on rendering
- Omitted windowing and input to avoid window system dependencies
OpenGL: Conceptual Model

Real Light

Real Object

Human Eye
OpenGL: Conceptual Model

Real Object → Real Light → Human Eye

Real Object → Real Object → Human Eye

Real Object → Synthetic Light Source → Synthetic Model → Synthetic Camera → Display Device → Human Eye

Graphics System
What can OpenGL do?
Examples from the OpenGL Programming Guide ("red book")
- **Wireframe** models
  - shows each object up of polygons
- the **lines are are the edges** and the **faces of the polygons make up the object surface**
Plate 3. The same scene with **antialiased lines** that **smooth the jagged edges**. See [Chapter 7](#).

when you approximate smooth edges using pixels, this leads to jagged lines especially with near vertical and near horizontal lines
Plate 4. The scene drawn with flat-shaded polygons (a single color for each filled polygon). See Chapter 5.

“unlit scene”
Plate 5. The scene rendered with lighting and smooth-shaded polygons. See Chapter 5 and Chapter 6.
Plate 6. The scene with **texture maps and shadows added**. See Chapter 9 and Chapter 13.
Plate 7. The scene drawn with one of the objects motion-blurred. The accumulation buffer is used to compose the sequence of images needed to blur the moving object. See Chapter 10.
Plate 8. A close-up shot - the scene is rendered from a new viewpoint. See Chapter 3.
OpenGL state machine

- put OpenGL into various states
  - e.g., current color, current viewing transformation
  - these remain in effect until changed
- glEnable(), glDisable(), glGet(), glIsEnabled()
- glPushMatrix(), glPopMatrix() to temporarily modify some state
OpenGL command syntax

- commands: `glClearColor();`
- `glVertex3f()`
- constants: `GL_COLOR_BUFFER_BIT`
- types: `GLfloat`, `GLdouble`, `GLshort`, `GLint`, `GLfloat`, `GLdouble`, `GLshort`, `GLint`
Simple OpenGL program

#include <whateverYouNeed.h>

main() {

    InitializeAWindow();

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f(0.25, 0.25, 0.0);
        glVertex3f(0.75, 0.25, 0.0);
        glVertex3f(0.75, 0.75, 0.0);
        glVertex3f(0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}

OpenGL Programming Guide, 7th Ed.

- blue are placeholders for windowing system commands
- clear color, actual clear
- Ortho – the coordinate system
- flush executes the commands
OpenGL Libraries

• OpenGL core library (gl.h)
  - OpenGL32 on Windows
  - GL on most unix/linux systems

• OpenGL Utility Library - GLU (glu.h)
  - avoids having to rewrite code

• OpenGL Utility Library - GLUT (glut.h)
  - Provides functionality such as:
    • Open a window
    • Get input from mouse and keyboard
    • Menus

- GL
  - no windowing commands
  - no commands for higher-level geometry - you build these using primitives (points, lines, polygons)
- GLU - standard in every implementation
- OpenGL Utility library provides modeling support
  - capabilities range from NURBS to perspective foreshortening
Software Organization

application program

OpenCL Motif widget or similar

GLUT

GLU

GL

X windows

software and/or hardware
Simple OpenGL program

```c
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f(0.25, 0.25, 0.0);
        glVertex3f(0.75, 0.25, 0.0);
        glVertex3f(0.75, 0.75, 0.0);
        glVertex3f(0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

OpenGL Programming Guide, 7th Ed.

- blue are placeholders for windowing system commands
- can replace blue code with calls to **glut**
# Simple OpenGL program

```c
#include<GL/glut.h>

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f(0.25, 0.25, 0.0);
        glVertex3f(0.75, 0.25, 0.0);
        glVertex3f(0.75, 0.75, 0.0);
        glVertex3f(0.25, 0.75, 0.0);
    glEnd();
    glFlush();
}

main() {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (FB_WIDTH, FB_HEIGHT);
    glutCreateWindow ("Test OpenGL Program");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
}
```

- blue are placeholders for windowing system commands
- can replace blue code with calls to **glut**