Course basics:
- Instructor – Dr. Natacha Georgieva
- Mon, Wed 11:15 am – 1:10 pm
- Materials will be available at:
  [http://cs.csi.cuny.edu/~natacha](http://cs.csi.cuny.edu/~natacha)
- 2 projects, 4 homeworks, 1 presentation, midterm and final

References

Outline and approach

Computer graphics is concerned with all aspects of producing pictures or images using a computer.

Applications of computer graphics:
- display of information
- design
- simulation
- user interfaces
Main Tasks

• Main tasks:
  – modeling: creating and representing the geometry of objects in the 3D world
  – rendering: generating 2D images of the objects
  – animation: describing how objects change in time

Why Study Computer Graphics?

• Graphics is cool
  – I like to see what I’m doing
  – I like to show people what I’m doing
• Graphics is interesting
  – Involves simulation, algorithms, architecture…
• I’ll never get an Oscar for my acting
  – But maybe I’ll get one for my CG special effects
• Graphics is fun

Graphics Applications

• Entertainment: Cinema
Graphics Applications

• Entertainment: Games

• Video Games

• Medical Visualization
Graphics Applications

- Computer Aided Design (CAD)

Graphics Applications

- Scientific Visualization

Computer Graphics Applications

Art, publicity
Outline and approach

Approaches to teaching CG:
- bottom-up
- survey
- top-down

You don't need to know what's under the hood to be literate, but unless you know how to program, you'll be sitting in the back seat instead of driving.

- Bottom up = study the engine
- Survey = hire a chauffeur
- Top Down = learn to drive
Introduction to graphic systems

Graphic system:

Pixels and the frame buffer:

Raster Images

• Common in Computer Graphics
  – incorporate images in scenes (texture mapping)
  – result of generated scene

• Bi-level image
  – pixels can have one of two values (0 and 1)
  – requires one bit to represent the intensity of each pixel
    • 0: no intensity (black)
    • 1: full intensity (white)

Gray-scale Images

• pixels can have more than two values
• classified by the number of bits needed to represent a pixel intensity level, **pixel depth**
  or number of **quantization** levels
• bits have possible gray levels
  – 2 bits/pixel = 4 gray levels
  – 4 bits/pixel = 16 gray levels
  – 8 bits/pixel = 256 gray levels
• 8 bit gray-scale images are common
Gray-scale Images

- Quantization
  - number of bits representing gray-scale values
  - more bits
    - more gray-scale values
    - higher gray-scale resolution
    - larger image size
  - useful in examining pixels within different ranges, so called window and level

Levels of Gray

- Levels of gray/colors: instead of 0/1 more digits

Levels of Gray

- Reducing the levels of gray: Original scanned picture
Levels of Gray

- Reducing the levels of gray: Three bits per pixel

Color Images

- Pixel value represents a color, RGB and Color Index (color lookup table)
  - RGB: each pixel is an ordered triple representing the intensity (amount) of red, green, and blue that are summed together (R, G, B)
  - color depth of a pixel is the total number of bits representing red, green, and blue
  - true-color images have a depth of 24 bits
    - 8-bits per color
Color palette

- Color table (palette)

<table>
<thead>
<tr>
<th>color value</th>
<th>displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>black</td>
</tr>
<tr>
<td>0.01</td>
<td>blue</td>
</tr>
<tr>
<td>0.10</td>
<td>green</td>
</tr>
<tr>
<td>0.11</td>
<td>cyan</td>
</tr>
<tr>
<td>1.00</td>
<td>red</td>
</tr>
<tr>
<td>1.01</td>
<td>magenta</td>
</tr>
<tr>
<td>1.10</td>
<td>yellow</td>
</tr>
<tr>
<td>1.11</td>
<td>white</td>
</tr>
</tbody>
</table>

Raster Displays

- Graphics systems produce raster images primarily because they will be displayed with a **raster display**
  - common display device on computers
- display surface composed of pixels
  - 480 rows, each containing 640 pixels for a total of 307,000 pixels
  - 1024 rows, each containing 1280 pixels for a total of 1,310,720 pixels

- More pixels representing a given size screen
  - higher resolution
  - larger amount of memory representing a scene
- built in coordinate system
  - relates a given pixel to a physical location on the screen
Frame buffer

- memory containing pixel and other values
- frame buffer commonly part of the graphics card
- frame buffer must be connected to the raster display device (monitor)
- pixel values are related to displayed intensity

Scan Converter

- changes digital pixel values to an analog voltage (intensity) values
- converts memory address into a physical location on the display screen
Video Monitors

- Primary output device
- Operation based on the standard cathode-ray tube (CRT)
- Beam of electrons (cathode rays) emitted by an electron gun
- Beam focused and directed by electric/magnetic deflection systems
- Phosphor coated screen emits small spot

Introduction to graphic systems

Output devices: cathode-ray tube (CRT)

Video Monitors

- The entire screen is painted once every 1/30th of a second (33 ms)
Video Monitors

- Phosphor emits a small spot of light
  - fades quickly => needs refreshing
  - refresh CRT
- Refresh (scanning) pattern proceeds from the top-left of the screen to the bottom-right
- each row is called a scanline

Introduction to graphic systems

Images: physical and synthetic

Objects and viewers, light and images
Introduction to graphic systems

Clipping rectangle

Programmer interface

Application programmer's interface (API)
need functions for:
- objects
- viewer
- light sources
- material properties

Ray tracing
Introduction to graphic systems

Human visual system

Pinhole camera

Synthetic camera model
## Graphics Programming

### Sierpinski gasket

```c
void main(int argc, char** argv)
{
    /* Standard GLUT initialization */
    glutInit(&argc,argv);
    glutInitDisplayMode (GLUT_SINGLE |
                        GLUT_RGB); /* default, not needed */
    glutInitWindowSize(500,500); /* 500 x 500 pixel window */
    glutInitWindowPosition(0,0); /* place window top left on display */
    glutCreateWindow("Sierpinski Gasket"); /* window title */
    glutDisplayFunc(display); /* display callback invoked when window opened */
    myinit(); /* set attributes */
    glutMainLoop(); /* enter event loop */
}
```

## Graphics Programming

### OpenGL API

**Graphic functions:**

- **primitive** (points, lines, polygons, pixels, text, curves, surfaces)
- **attribute** (colors, fills, type face for titles of graphs)
- **viewing** (types of views)
- **transformation** (rotation, translation, scaling)
- **input**
- **control** (communicate with window system, initialize programs, deal with errors)

## Introduction to OpenGL

- **What is an application programmer’s interface?**
  - Software library
  - Layer between programmer and graphics hardware and software
- **Where does OpenGL fit in?**
  - Between application and graphics system
  - Between high level API and system software
Why OpenGL?

- Fast
- Simple
- Well-defined architecture
- Window system independent
- Supports high-end features
- Both geometric and pixel processing
  Standard, available on many platforms

Why OpenGL

- industry standard
- stable
- reliable and portable
- evolving
- scalable
- easy to use
- well-documented API
- provides a full set of 3D graphics routines
OpenGL As a Renderer

- Renders simple geometric primitives
  - Points, lines, polygons
- Renders images and bitmaps
- Separate pipelines for geometry and pixels, linked through texture mapping
- Rendering depends on state
  - Colors, light sources, materials, texture, normals

Event driven programming

- Almost all window based programs follow the event driven paradigm
  - program waits for events to occur
  - takes appropriate action
- Events are typically stored in an **event queue**
- Application programs specify the types of events that are of interest
- The window system passes events to apps.
Event driven programming

- Appropriate action is implemented by the application calling a unique, user defined function for each event of interest
- These functions are referred to as callback functions
- Associating callback functions with events is not a rendering issue, OpenGL does not provide for this
- GLUT (OpenGL Utility Toolkit) does

Registering Events

- The function call:
  
  ```
  void main(int argc, char* argv[]) {
    open a screen window
    render into the window
    process events
  }
  ```

- The function call:
  
  ```
  glutMouseFunc(mouse)
  ```
  - associates the callback function mouse with all mouse events
  - button press/release
- When a mouse button is pressed or released in the graphics application window, the user defined function mouse is called
- Application ignores all mouse events without this function call
OpenGL Buffers

• OpenGL supports a variety of buffers that can be used for advanced rendering
  – Color buffers (front, back, right, left)
  – Depth (z) buffer
  – Accumulation
  – Stencil
• Window system interactions must be handled outside of OpenGL

OpenGL Related Libraries

• OpenGL Utility Library (GLU)
• GLX (X Window extension), glX...
• WGL (MS Windows extension), wgl…
• GLUT (OpenGL Utility Toolkit), glut…
• OpenInventor
  – C++
  – high level three-dimensional applications

Simple OpenGL Program

```c
#include <GL/gl.h>
#include<GL/glew.h>
void display(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glColor3f(1.0, 1.0, 1.0);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
    glMatrixMode (GL_MODELVIEW);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
    glEnd();
    glutSwapBuffers();
}
```
Clearing

- Clearing color has to be set
  - `glClearColor(r,g,b,alpha)`
- Buffers can be cleared all at once
  - `glClear(bitfield)`
    - Color buffer
    - Depth buffer
    - Accumulation buffer
    - Stencil buffer