CS148 Overview
Display Devices

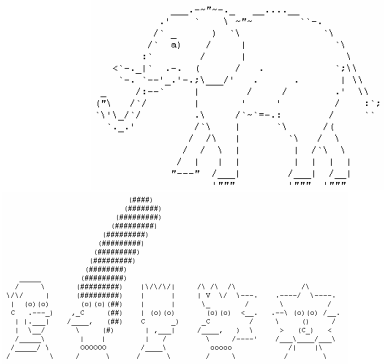
CS148: Intro to CG
Instructor: Dan Morris
TA: Sean Walker
June 21, 2005

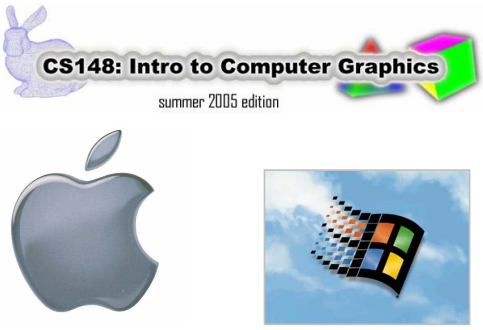
Outline for today

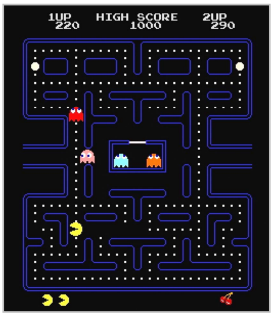
- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming

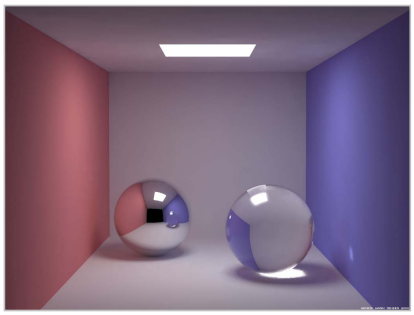
Outline for today

- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming









Computer graphics is…

- Computer-generated artwork
- Interactive 2D graphics
- Interactive 3D graphics
- Photorealistic 3D graphics
- Photorealistic 3D video
- Digital photography

One definition to rule them all…

- Computer graphics is the science of coloring pixels on a display to trick the viewer into seeing an object or a scene.

What type of computer graphics will we address this quarter?

- CS148 is (mostly) about interactive 3D graphics
  - A little bit on 2D graphics
  - A little bit on non-interactive 3D graphics
  - No image processing or digital photography
- CS448a: Computational photography
- CS448b: Visualization
- CS348a: Geometric modeling
- CS348b: Rendering
- CS223b: Computer vision

Outline for today

- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming

CS148: Intro to Graphics

- Introduces important mathematical concepts in graphics
- Primarily focused on giving you a working knowledge of OpenGL
  - What you learn will generalize to other 3D environments, e.g. DirectX, Matlab, Amira, Maya




- Not as theoretically in-depth as CS248; if you plan to continue in graphics at Stanford, you should look at CS248

Administrative Blah Blah 1: Grading

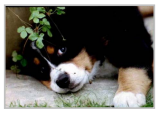
- 50% Programming Projects
- 20% Midterm
- 30% Final
- You need passing work on both exams and all projects to pass
- This part of the lecture is boring so I'm including thoroughly gratuitous pictures of cute puppies to keep you awake:

Administrative Blah Blah 2: Late Policy

- You have three "late days" for the quarter
- 25% lost per day after that
- 3 hours late is 1 late day
- For group projects, all members lose late days for late submissions
- Make your life easier and submit on time

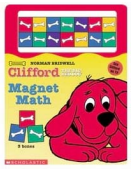
Administrative Blah Blah 3: Textbook

- *OpenGL Programming Guide v1.4, Fourth Edition* ("The Red Book") is the official text
- Second edition is online and linked from the website (85% similar)
- Handouts posted the night before class – print them out if you want them in class



Administrative Blah Blah 4: Math

- There is math in graphics
- This is not a math course
- Look over the "essential math" handout
- Get in touch with us if you have questions



Administrative Blah Blah 5: Programming

- OpenGL and GLUT (GL Utility Toolkit)
- All grading will be done on the *myth*, *firebird*, and *raptor* Linux machines in Sweet Hall
- You can develop at home if you like, and we'll provide Windows project files, but be sure to test on the Linux machines
- Subtle subtext: there is substantial programming in CS148

Wait! That's not a puppy!

Administrative Blah Blah 6: Getting in Touch

http://cs148.stanford.edu
cs148staff@cs.stanford.edu

Dan's office hours:
Tuesday, 1pm-3pm, Gates 116
Or email dmorris@cs.stanford.edu

Sean's office hours:
TBA

Summary: How to Succeed in CS148

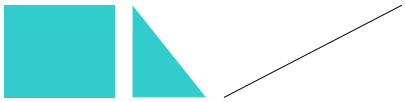
- Come to class
- Start the projects early
- The staff will not debug your programs!
- Be creative: we *want* to give you extra credit
  - Suggest optional project components or whole projects from your own work
- Submit questions for exams

Outline for today

- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming

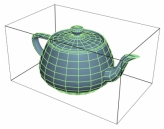
Terminology: PRIMITIVES are made of PIXELS

- Pixels are the dots that make your display, you have on the order of a million of them
- Primitives are 2D shapes… generally lines, triangles, and quads (4-sided polygons)



Terminology:
OBJECTS are made of PRIMITIVES

- Even 3D objects are made of 2D primitives
- Objects can be tens to billions of primitives
- More primitives → smoother objects


Fun With Primitives: Make an Octagon From…

Triangles

Quads

One quad + triangles (why?)

Terminology:
SCENES are made of OBJECTS

- Tens to thousands of objects per scene
- Millions of primitives in many scenes
- Speed is huge in graphics


Outline for today

- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming

An ideal graphics programming interface

Dear Monitor,

Please draw a green spaceship in which a purple alien is shooting at a blue robot.

Sincerely,

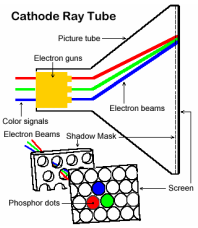
Dan

What can your monitor do?

- You can't tell your monitor "draw some spaceships"
- You can't even tell your monitor "draw some triangles"
- Your monitor only knows how to turn dots on and off.

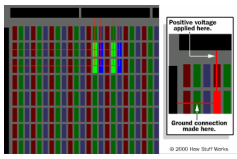
Raster-Scan Displays: CRT's

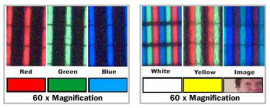
- Incoming volts turn on electron gun
- Magnetic field bends beam toward screen
- Electrons hit tiny phosphor elements to turn them on
- You see pixels
- Beam sweeps over and over at about 60Hz
  - Why so fast?



Raster-Scan Displays: LCD's

- Grid of wires puts volts on crystals
- Crystals twist to block light or let it pass
- Big white light shines behind the whole grid
- A red, green, or blue filter sits in front of each crystal
- Still scans from side to side and top to bottom





Raster-Scan Displays: Summary

- All a raster-scan display can do is scan through every pixel sequentially.
- It needs instructions about what to do for every pixel.

Doing it by hand

```
for(int i=0; i<height; i++) {
  for(int j=0; j<width; j++) {
    put out the volts for the current pixel;
    wait until it's time for the next pixel;
  }
}
```

This would not be very much fun…

Disappointment?

Dear Monitor,

Communicating with you seems terribly tedious. I'm dropping graphics and taking compilers instead.

Sincerely,
Dan

PS Don't call me either.

Solution: the framebuffer

- Graphics card takes care of talking to the monitor
- You just need to fill up an array telling the graphics card which colors should go where

Framebuffer data

1bpp black-and-white display

```
memory contents                 display screen

00300000300000030               ----------------
00J00000J00000J00               ----------------
00300001300000030               ----------------
00300000130000030               --------1-------
00300000013000030               ---------1------
00300000001300030               --1111:1111-----
00300000001300030               ---------1------
00300000013000030               --------1-------
00J00000J00000J00               -------1--------
00300000300000030               ----------------
00300000300000030               ----------------
0 pixel off                     - pixel s black
1 pixel on                      1 pixel is illuminated
```

6bpp color display

Bit values	Relative intensity
00	0 (none of this color)
01	1/3 (dim)
10	2/3 (brighter)
11	11 (brightest)

What color is 000101 in a 6bpp BGR framebuffer?

Real-world framebuffers

- Typically 24bpp or 16bpp
- Typically 1280x1024
- How much memory does this take up?

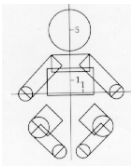
Has this made our lives easier?

- Filling the framebuffer with pixels manually isn't practical
- Even simple images have thousands of pixels
- Try drawing this baby pixel-by-pixel…

Primitives save the day

```
void circle(int cx, int cy, double radius, bool
  filled);
void rectangle(int cornerx, int cornery, double
  width, double height, double angle, bool
  filled);
```

- Now drawing the baby doesn't seem so bad…



How does all this fit into CS148?

- Thursday's class and Project 1: Turning primitives into pixels
- The rest of the course: Using primitives – supplied by OpenGL – to make pretty pictures

Outline for today

- What is computer graphics?
- Intro to CS148
- Some terminology
- Display devices
- Graphics/GUI programming

What's under the hood?

```
void circle(int cx, int cy, double radius, bool
  filled);
```

- Every video card speaks a slightly different language
- Different video cards know about different primitives
- Different video cards live in different places in hardware (PCI, AGP, USB, etc.)



Device drivers

- A device driver is a library that has low-level routines for talking directly to the hardware
- Generally released by the manufacturer
- Might contain a drawCircle() routine that does all the hard parts
- But something's still missing…

Standard Graphics API's

- All device manufacturers write their drivers with a common set of function names, so lots of programs can use them
- Multiple standards exist:
  - OpenGL
  - Direct3D
  - GDI, X, PlayStation, etc.

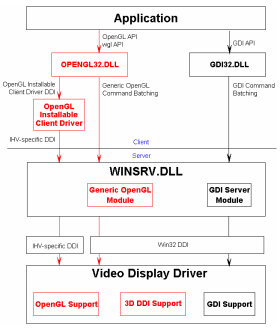
Putting it all together: device-independent programming

- Write program using OpenGL
- Compile program against empty library or OS shell library
- At runtime, the OS links your program to the device driver's version of OpenGL

Putting it all together: Linux

Your favorite game
(speaks OpenGL)
||
OpenGL library
(finds driver for you)
||
XFree86 DRI Module (aka driver)
(speaks video card's language)
||
AGP Kernel Module
(formats data for the AGP bus)
||
Linux kernel
(writes data to the AGP bus)
||
Snazzy Video Card
(speaks monitor's language)

Putting it all together: Windows



Finally, happiness…

Dear Monitor,

I'm leaving you for OpenGL. She speaks my language, you know, we can really communicate.

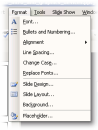
Sincerely,
Dan

PS I'm just a man.

GUI's create the same problem

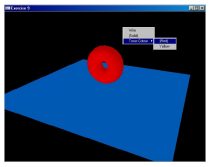
- 3D graphics would be quite dull without interactivity
- Want to get mouse and keyboard info
- Want slick GUI's
- OpenGL doesn't help us here…





Solution: GLUT

- GLUT: GL Utility Toolkit
- Standard set of function names to get simple UI features
- Plays nice with GL
- Implemented (and free) for many platforms



Sample GLUT program

```
void main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("This is a sample GLUT program");
    glutDisplayFunc(myDisplay);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKey);
    glutReshapeFunc(myReshape);
    glutMainLoop(); /* Doesn't return… */
}

// Keyboard function – called when a key is pressed
koid myKey(unsigned char key, int x, int y) { }

// Mouse function – called when a mouse button is pressed
void myMouse(int button, int state, int x, int y) { }

// Display function – called to redraw window
void myDisplay(void) { }
```

For next time

- Find us if you have questions
- Play with project 1
- Look over the essential math handout
- Next time: scan conversion