CS148: Introduction to Computer Graphics and Imaging

Drawing Machines

Electric Sheep screensaver by Scott Draves
Image created collectively by users on the internet

Pablo Picasso
Patrick Tressett

www.tenderpixel.com/aikon.html

Alexander Weber

tinkerlog.com/2011/09/02/der-kritzler/
Graphtec Pen Plotter

HPGL
PU100,100; // Pen Up
PD200,100; // Pen Down
PD200,200;
PD100,100;
PD100,100;

Roman Verostko
Laser Cutter

Laser Projector

http://elm-chan.org/works/vlp/report_e.html
Pink Floyd Laser Show

SAGE Display for Whirlwind Computer

www.thesagesite.org

CS148 Lecture 2
Pat Hanrahan, Fall 2011
Oscilloscope

Inputs: X and Y

Cathode Ray Tubes: Electron Plotters

www.tubeclockdb.com
Raster Displays (Television)

**NTSC**

Continuously Refresh Display

DAC = Digital to Analog Convertors

Image = 2D array of colors
Equivalent Program

```c
#define XRES 1440
#define YRES 900
#define FPS 60 // Hz
typedef struct { byte R, G, B, A; } RGBA;
typedef RGBA image[YRES][XRES];
for( y = 0; y < YRES; y++ )
    for( x = 0; x < XRES; x++ ) {
        DAC.R = image[y][x].R;
        DAC.G = image[y][x].G;
        DAC.B = image[y][x].B;
        hold(1.0/(FPS*XRES*YRES));
    }
```

"Frame Buffer"

Store Image in a Buffer to Separate Refresh Rate from Drawing Rate
OpenGL ≡ Abstract Drawing Machine

- **Vertices**
  - OpenGL commands
  - Per-vertex ops
    - Rasterizer
      - Texturing
        - Per-fragment ops
          - Frame buffer ops
            - Pixels
              - triangles, lines, points, images

- **Transformed vertices**
- **Fragments**
- **Shaded fragments**

---

CS148 Lecture 2

Pat Hanrahan, Fall 2011

- `glBegin(GL_TRIANGLES)`
- `glVertex3fv(v1);`
- `glVertex3fv(v2);`
- `glVertex3fv(v3);`
- `glEnd()`
Today’s Outline

The OpenGL drawing machine
  Input primitives
  State and attributes
  Windows

Performance and the need for hardware

Goal: Understand the graphics pipeline and learn how to create pictures using OpenGL

OpenGL Examples

rect.c
Framebuffer

My Macbook Pro Framebuffer: 1440 x 900

All coordinates are integers; they refer to pixel locations in the framebuffer

Window

My Macbook Pro Framebuffer: 1440 x 900

The window is the portion of the display usable by the application (under control of the “window system”)

Window (512 x 512)

All coordinates are integers; they refer to pixel locations in the framebuffer
Viewport

My Macbook Pro Framebuffer: 1440 x 900

The viewport is the portion of the window that can be drawn in, no pixels will appear outside the viewport.

All coordinates are integers; they refer to pixel locations in the framebuffer.

Window = Virtual Framebuffers

Like virtual memory enables us to share the physical memory, virtual framebuffers allows us to share the physical framebuffer.

Abstract properties
- Location and size on the screen
- Stacking order
Exposure

Parts of the window that are not visible can become visible

- When opening
- When unhiding
- When brought to front

How to update the visible parts of the window?

- Redraw entire window (or a part of it)
- Save the drawn window in a “backing store”, and copy onto screen when needed

Two Interpretations of Window

http://www.imaginativeinteriors.co.uk/trompe.shtml

Window on the Display (Virtual Framebuffer)

Window into a Virtual World
Window Coordinate Systems

Each window has a user coordinate system. A 2D coordinate system is specified by assigning coordinates to the edges of the window. The left need not be less than the right.

Pixel Coordinates

Window edges at integers

Pixels inside window

(0,0) (1,0) (0,1) (1,1) (4,3)
Pixel Coordinates

OpenGL: Pixel centers correspond to half-integer coordinates

Note: Other graphics packages may use a different convention

OpenGL Examples

reshape1.c, reshape2.c, reshape3.c
Buffering

Single-buffer
■ Draw into display buffer directly
■ *May see picture being drawn*

Double-buffer
■ Display “front” buffer
■ Draw into “back” buffer (can’t see drawing)
■ Swap front and back (wait until vertical sync)

Triple buffer
■ Avoid waiting for vertical sync

OpenGL Examples

single.c, double.c
OpenGL Drawing Primitives

Geometric Shapes

Bitmaps

Images

OpenGL Shape Primitives
OpenGL Examples

primitives.c, concave.c

Hardware
Performance

Limited by fragment drawing rate

\[
\text{\#fragments / second} = |\text{Image}| \times \text{FPS} \times \text{depth complexity}
\]

\[
1440 \times 900 \times 60 \times 4 = 32 \text{ Gigafragments/sec}
\]

Note: depth complexity is average “overdraw”

Bandwidth

\[
\text{bandwidth} = \text{\#fragment/second} \times \text{\#bytes/fragment}
\]

\[
32 \text{ Gigafragments/sec} \times 4 = 96 \text{ GB/s}
\]

#triangles/sec

\[
\text{\#triangles/sec} = (\text{\#fragments/sec}) / \text{avg(|triangle|)}
\]

Triangles are small!
Performance

\[
\text{#triangles/sec} = \frac{\text{#fragments/sec}}{\text{avg}(|\text{triangle}|)}
\]

Assume 16 pixels

\[
= \frac{32 \ \text{GigaFragments}}{16 \ \text{pixels/triangle}}
= \frac{2 \ \text{GigaTriangles}}{s}
\]
### NVIDIA GPU Performance

<table>
<thead>
<tr>
<th>Year</th>
<th>Product</th>
<th>Triangle/s</th>
<th>Fragment/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>Riva ZX</td>
<td>3m</td>
<td>100m</td>
</tr>
<tr>
<td>1999</td>
<td>Riva TNT2</td>
<td>9m</td>
<td>350m</td>
</tr>
<tr>
<td>2000</td>
<td>GeForce2 GTS</td>
<td>25m</td>
<td>664m</td>
</tr>
<tr>
<td>2001</td>
<td>GeForce3</td>
<td>30m</td>
<td>800m</td>
</tr>
<tr>
<td>2002</td>
<td>GeForce Ti 4600</td>
<td>60m</td>
<td>1200m</td>
</tr>
<tr>
<td>2003</td>
<td>GeForce FX</td>
<td>167m</td>
<td>2000m</td>
</tr>
<tr>
<td>2004</td>
<td>GeForce 6800 Ultra</td>
<td>170m</td>
<td>6800m</td>
</tr>
<tr>
<td>2005</td>
<td>GeForce 7800 GTX</td>
<td>940m</td>
<td>10300m</td>
</tr>
<tr>
<td>2006</td>
<td>GeForce 7900 GTX</td>
<td>1400m</td>
<td>15600m</td>
</tr>
<tr>
<td>2007</td>
<td>GeForce 8800 GTX</td>
<td>1800m</td>
<td>36800m</td>
</tr>
<tr>
<td>2008</td>
<td>GeForce GTX 280</td>
<td></td>
<td>48160m</td>
</tr>
<tr>
<td>2010</td>
<td>GeForce GTX 480</td>
<td></td>
<td>42000m</td>
</tr>
<tr>
<td>2011</td>
<td>GeForce GTX 580</td>
<td></td>
<td>49400m</td>
</tr>
</tbody>
</table>

### Modern PC

- 3.0 Ghz Intel Core2 Duo
- Core 1
- Core 2
- 4MB L2 Cache
- 2GB main memory (DDR2)
- 12.8 GB/sec
- PCIe Bus (v1 = 4 GB/sec)
- NVIDIA GeForce 8800 GTX (575 MHz)
- 512MB video Memory (GDDR3)
- 84 GB/sec
- System board (Intel D975)

CS148 Lecture 2
Pat Hanrahan, Fall 2011
Game Machines – Xbox 360

- 3.2 Ghz PowerPC CPU
  - Core 1
  - Core 2
  - Core 3
  - L2 Cache
- 500 Mhz ATI GPU
  - 48 3D Cores
  - Frame buffer
  - Video out
- 512 MB memory
- IO Chip
- Display (TV)

Game Machines – PS3

- 3.2 Ghz Cell
  - PPC Core
  - SPU0
  - SPU1
  - SPU2
  - SPU3
  - SPU4
  - SPU5
  - SPU6
  - SPU7
  - L2 Cache
- 550 Mhz NVIDIA RSX GPU
  - Multiple 3D cores
  - Video out
- 256 MB Memory (XDR)
- 256 MB video Memory (GDDR3)
- IO Chip
- Display (TV)
Hybrid CPU-GPUs

Intel Sandybridge

Apple A5

Summary

Drawing machines
- Why framebuffers and raster scan displays?

OpenGL as an abstract drawing machine
- Pipeline
- Primitives
- State
- Windows and viewports
- Buffering

Hardware
- Performance limits