Introduction to SIGGRAPH and Interactive Computer Graphics

Mike Bailey
Oregon State University

mjb@cs.oregonstate.edu
Mike Bailey

Professor of Computer Science, Oregon State University

PhD from Purdue University

Has worked at Sandia Labs, Purdue University, Megatek, San Diego Supercomputer Center (UC San Diego), and OSU

mjb@cs.oregonstate.edu
Course Goals

- Understand the different venues at SIGGRAPH, and how to strategically attend the ones that are best for you
- Provide a background for papers, panels, and other courses
- Help appreciate the Animation Festival
- Get more from the vendor exhibits
- Give our take on where the future is
- Provide pointers for further study
Topics

- How to Attend SIGGRAPH 2008
- The Graphics Process
- Graphics Hardware
- GPU programming
- Graphics Input Devices
- Finding More Information
Introduction to SIGGRAPH and Interactive Computer Graphics

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu
How to Attend SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

You can’t see it all, so …
Think Strategically -- Make a Plan, Make a Schedule, and set Priorities!
Courses

- Monday – Friday
- Some are “standard knowledge” (e.g., this one…)
- Some are cutting edge
- Some are new topics
- You got the full set of notes on the courses DVD

Courses Strategy

- Choose courses that are useful
- Choose courses that are meaningful
- Chose courses where you really want to see the speaker(s)
- Choose courses for which there will be great visual presentations which cannot be replicated in the notes
- Hop around between courses to catch the best topics and speakers
Papers

- Tuesday - Friday
- Deep technical information
- Printed in the proceedings and on the conference DVD
- Strategy: attend when an animation or interactive technique is being discussed
- Strategy: attend when you think you will not understand the topic from reading about it alone

Panels

- Monday - Friday
- General discussion, opinion, sometimes controversial
- Important point: sometimes transcribed, usually not
- Strategy: attend when a topic, speaker, or animation you really want to see is being discussed
Talks

- Tuesday - Friday
- Paper-like content, but shorter and less formal
- Some of the work has been completed
- Some of it is “Work in Progress”
- Oftentimes, the source of some really cool ideas

Special Events

- Fast-Forward Papers review: Monday 6:00-8:00
- ACM Student Research Competition: Friday, 10:30 – 12:15
Exhibition

• Tuesday – Thursday

• Many vendors time their hottest product releases for this week!

• But, it’s big, so go with a strategy!

Exhibition Strategy

• The Exhibition is open 9:30-6:00 on Tuesday and Wednesday, and 9:30-3:30 on Thursday

• The vendors are all listed in the Program and Buyer’s Guide

• They are listed both by alphabetical name and by product category
Exhibition Strategy

• Look at the list of vendors in the Program and Buyers Guide
• Make a list of the ones you really must see and sort the list by booth number
• Booth numbers are XXYY, where XX is the Aisle # and YY is \((\frac{1}{2}) x\) the number of feet from the front
• Even-numbered booths are on the left (looking from the front of the hall), odd numbers are on the right
• For example, Natural Point = booth 0733, which is Aisle 7; 165 feet from the front, on the right
• Start at one end of the floor and work your way across
Exhibition Warning

The Exhibition closes at 3:30 on Thursday afternoon, not 6:00 !!!

Exhibitor Tech Talks

• Tuesday - Thursday
• 2-hour demos, tutorials, and interactive instruction
• Sometimes a raw sales pitch, but usually (hopefully) more informative than that
• Great chance to ask questions in a quieter setting than the Exhibition
Computer Animation Festival

- Monday – Thursday
- Computer Graphics’s greatest animation hits for the past year
- It is considered cool to see it early in the week
- This year, it’s divided into:
  - Talks
  - Stereoscopic
  - Production

Reception!

- Thursday, 7:00 – 10:00
Art and Design Galleries

- Monday – Friday
- Always fun and interesting!
- Be sure to see it, even if you just stroll through without stopping

New Tech Demos

- Monday - Friday
- Speculative research on graphics, visualization, robotics, music, audio, displays, haptics, sensors, gaming, web, AI, and entertainment
The Studio

• Monday - Friday
• Working studio for 2D and 3D graphics, and 3D prototyping

Job Fair

• Tuesday – Thursday
• Good place to find information on who is hiring
• Also a good place to post your resume
Birds of a Feather

- Opportunity for impromptu gatherings centered around different topics
- All week
- Check the BOF sign-up board

Good Luck, and have fun!
How to Attend SIGGRAPH 2008

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu
The Graphics Process

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

The Graphics Process
The Graphics Process: Geometric Modeling

- 3D Scanning
- Interactive Geometric Modeling
- Model Libraries
- Displacement Mapping

3D Geometric Models

Rendering
The Graphics Process: 3D Animation

- Motion Design
- Motion Computation
- Motion Capture
- Dynamic Deformations

3D Animation Definition

Rendering

The Graphics Process: Texturing

- Scanned Images
- Computed Images
- Painted Images

Texture Information

Rendering
The Graphics Process: Rendering

- 3D Geometric Models
- Rendering
  - Transformation, Clipping, Perspective
  - Image Generation
- Image Storage and Display
- Texture Information

The Graphics Process: Image Storage and Display

- Rendering
  - Hardware Framebuffer
- Disk File
- Film Recorder
- Video Recorder
The Graphics Process: Summary

The Graphics Process

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu
Computer Graphics Hardware

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

The Generic Graphics Process

3D Geometric Models
3D Animation Definition
Lighting Information
Rendering
Texture Information
Image Storage and Display
The Human

- Acuity: 1 arc-minute for those with 20/20 vision
- Required refresh rate: 40-80 refreshes/second
- Required update rate: 15+ frames/second
- Red, Green, Blue receptors in the eye

- How many colors are we able to detect?
The Computer Graphics Monitor(s)

Video Driver

Displaying Color on a Computer Graphics CRT Monitor

- 3 color guns
- Red-green-blue phosphors
- Gun voltage \(\approx\) color brightness
Displaying Color on a Computer Graphics Plasma Monitor

- Gas cell
- Phosphor
- Grid of electrodes

Source: http://electronics.howstuffworks.com

Displaying Color on a Computer Graphics LCD Monitor

- Grid of electrodes
- Color filters

Source: http://electronics.howstuffworks.com
Additive Color (RGB)

Display Resolution

- **Pixel** resolutions (1024x768 - 1600x1200 are common)
- Screen size (13", 16", 19", 21" are common)
- Human acuity: 1 arc-minute is achieved by viewing a 19" monitor with 1280x1024 resolution from a distance of ~40 inches
- FYI: HDTV is about resolutions in the 2048x1152 range
The Video Driver

- **N refreshes/second** (N is usually between 40 and 80)
- Framebuffer contains the R,G,B that define the color at each pixel
- Cursor
  - Appearance is stored near the video driver in a “mini-framebuffer”
  - x,y is given by the CPU
- Video input
The Framebuffer

- Fragment Processor
  (custom code can go here)

- Z-Buffer

- Double-buffered Framebuffers

# Bits/color # Shades per color

- Direct color

# Bits/pixel Total colors:

<table>
<thead>
<tr>
<th># Bits/pixel</th>
<th>Total colors:</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>$2^{12} = 4K$</td>
</tr>
<tr>
<td>18</td>
<td>$2^{18} = 256K$</td>
</tr>
<tr>
<td>24</td>
<td>$2^{24} = 16.7M$</td>
</tr>
</tbody>
</table>

# Bits/color # Shades per color

<table>
<thead>
<tr>
<th># Bits/color</th>
<th># Shades per color</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$2^4 = 16$</td>
</tr>
<tr>
<td>6</td>
<td>$2^6 = 64$</td>
</tr>
<tr>
<td>8</td>
<td>$2^8 = 256$</td>
</tr>
</tbody>
</table>

Oregon State University
Computer Graphics
What’s New?

• 16- and 32-bit floating point for each color component

Why?

Many modern algorithms do arithmetic on the framebuffer color components. They need the extra precision.

The Framebuffer

• Alpha values
  – Transparency per pixel
    \( \alpha = 0 \text{ is invisible} \)
    \( \alpha = 1 \text{ is opaque} \)
  – Represented in 8-32 bits
    (integer or floating point)
  – Alpha blending equation:
    \[
    \text{Color} = \alpha C_1 + (1 - \alpha) C_2
    \]
    \( 0.0 \leq \alpha \leq 1.0 \)
The Framebuffer

- **Z-buffer**
  - Used for hidden surface removal
  - Holds pixel depth
  - Typically 16, 24, or 32 bits deep
  - Integer or floating point

<table>
<thead>
<tr>
<th># Bits / Z</th>
<th>Total Z Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>(2^{16} = 65K)</td>
</tr>
<tr>
<td>24</td>
<td>(2^{24} = 17M)</td>
</tr>
<tr>
<td>32</td>
<td>(2^{32} = 4B)</td>
</tr>
</tbody>
</table>

Z-Buffer Steps

1. Compare the incoming Z value with the Z value already in that pixel
2. If the incoming Z value is closer to the viewer than what is already there, open the gates
3. Write the new pixel color and Z value
Z-Buffer Operation

On a pixel-by-pixel basis:

Incoming

<table>
<thead>
<tr>
<th>Z</th>
<th>Allow</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Existing

<table>
<thead>
<tr>
<th>Z</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

The Framebuffer

*Double-buffering*: Don't let the viewer see *any* of the scene until the entire scene is drawn.
The Fragment Processor

- Takes in all information that describes this pixel
- Produces the RGBA for that pixel’s location in the framebuffer
The Rasterizer

 Pipeline Processor
 (custom code can go here)

 Rasterizer

 Fragment Processor
 (custom code can go here)

Rasterization

- Turn screen space vertex coordinates into pixels that make up lines and polygons
- A great place for custom electronics
Rasterizers Interpolate:

- X and Y
- Red-green-blue values
- Alpha values
- Z values
- Intensities
- Surface normals
- Texture map coordinates
- Custom values given by the shaders

Anti-Aliasing
Texture Mapping

- “Stretch” an image onto a piece of geometry
- Image can be generated by a program or scanned in
- Useful for realistic scene generation
Oregon State University
Computer Graphics

Repeating Texture
Bilinear Texture Interpolation
Decal Mode
Modulate Mode

Something New: Write-Your-Own Fragment-Processor Code

Rasterizer
Fragment Processor
(custom code can go here)
Texture Memory

Bump Mapping
Line Integral Convolution

Referred to as:
Pixel Shaders or Fragment Shaders
Pipeline Processor

- Coordinates enter in world (application) coordinate space
- Coordinates leave in screen (pixel) coordinate space
- Another great place for custom electronics

The Pipeline Processor

Diagram:
- CPU
- Display List
- Pipeline Processor (custom code can go here)
- Rasterizer
Pipeline Processor: Transformations

- Used to correctly place objects in the scene
  - Translation
  - Rotation
  - Scaling

Pipeline Processor: Windowing and Clipping

- Declare which portion of the 3D universe you are interested in viewing
  - This is called the **view volume**
  - Clip away everything that is outside the viewing volume
Pipeline Processor: Projection

- Turn 3D coordinates into 2D
  - Parallel projection
    Parallel lines remain parallel
  - Perspective projection
    Some parallel lines appear to converge
Something New: Write-Your-Own Pipeline Code

CPU

Display List

Pipeline Processor
(custom code can go here)

Rasterizer

Referrred to as:
Vertex Shaders

Wireframe
Teapot Dome
Projection

Mars
Panoram
Dome
Projection

The CPU and Bus

Input Devices

Network

CPU

Display List

Pipeline Processor
(custom code can go here)

<table>
<thead>
<tr>
<th>Type of Board</th>
<th>Speed to Board</th>
<th>Speed from Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI</td>
<td>132 Mb/sec</td>
<td>132 Mb/sec</td>
</tr>
<tr>
<td>AGP 8X</td>
<td>2 Gb/sec</td>
<td>264 Mb/sec</td>
</tr>
<tr>
<td>PCI Express</td>
<td>4 Gb/sec</td>
<td>4 Gb/sec</td>
</tr>
</tbody>
</table>
Display List

- A list of graphics instructions created ahead of time and then “played back” when needed
- May or may not be editable once it is created
- Sometimes the structure of display lists is in the form of a scene graph
- Display lists can live in CPU memory or on the graphics card (display objects)

The Limitations of using NTSC Video

- Cannot display saturated colors well
- Expect an effective resolution of (at best) ~640x480
- Do not use single-pixel thick lines
- Stay away from the edges of the screen
- Some colors have better video resolution than others
NTSC Cycles of Encoding per Scanline

<table>
<thead>
<tr>
<th>What</th>
<th>Cycles/Scanline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity</td>
<td>267</td>
</tr>
<tr>
<td>Orange-Blue</td>
<td>96</td>
</tr>
<tr>
<td>Purple-Green</td>
<td>35</td>
</tr>
</tbody>
</table>

All Together Now!

- CPU
- Display List
- Vertex Processor
- Rasterizer
- Fragment Processor
- Z-Buffer
- Video Driver
- Texture Memory
- Double-buffered Framebuffers

**Coordinate Systems**
- MC = Model Coordinates
- WC = World Coordinates
- EC = Eye Coordinates
- CC = Clip Coordinates
- NDC = Normalized Device Coordinates
- SC = Screen Coordinates

**Colors**
- RGBA

**Pixel Parameters**
- RGBA Pixels
- Texels

**Varying Variables**
- Varying variables

**Abbreviations**
- MC = Model Coordinates
- WC = World Coordinates
- EC = Eye Coordinates
- CC = Clip Coordinates
- NDC = Normalized Device Coordinates
- SC = Screen Coordinates
- TC = Texture Coordinates
The Whole Process on a Card

Computer Graphics Hardware

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu
GPU Programming

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

GPU Shader Programming

- Allows programmers to load their own code into parts of the hardware graphics pipeline
- Gives a unique combination of control and speed
- This is a hot, new area in computer graphics
- These notes will focus on what can be done this way, not on how to do it (that would take lots more time)
- If you want to know more, catch me during a break!
A GLSL Vertex Shader Replaces These Operations:

- Vertex transformations
- Normal transformations
- Normal normalization
- Handling of per-vertex lighting
- Handling of texture coordinates

A GLSL Fragment Shader Replaces These Operations:

- Color computation
- Texturing
- Color arithmetic
- Handling of per-pixel lighting
- Fog
- Blending
- Discarding fragments
Bump Mapping with Shaders

Changing the Bump Height
Cube Mapping with Shaders

Cube Map of Nvidia’s Lobby
Using Noise in Shaders

Positional Noise
Gradient Noise

Image Representation of 2D Noise

1 Octave

4 Octaves
**How to Apply Noise**

1. Have an equation to describe color assignment
2. Have actual coordinates at a pixel
3. Add Noise to the actual coordinates to produce new coordinates
4. Use the new coordinates in the old equation to assign a color at that pixel
Noise Examples

Color Blending for Marble

Color Blending for Fire

Color Blending for Clouds

Deciding when to Discard for Erosion

Using Shaders to Manipulate Imagery
Toon Rendering for Non-Photorealistic Effects

Use the GPU to enhance scientific and engineering illustration.

Mandelbrot Set

\[ z_{i+1} = z_i^2 + z_0 \]

How fast does it converge, if ever?
Julia Set

\[ Z_{i+1} = Z_i^2 + C \]

How fast does it converge, if ever?

Spectral Effects Using Shaders
Rainbow Effects

Rainbow Strategy

1. Draw one big quadrilateral across the scene
2. Anywhere that \(0.74 \leq \cos(\Theta) \leq 0.77\), paint the right color
3. If not, discard that fragment

<table>
<thead>
<tr>
<th>Color</th>
<th>(\lambda)</th>
<th>(n)</th>
<th>(\Theta)</th>
<th>(\cos(\Theta))</th>
<th>(\Theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>650 nm</td>
<td>1.510</td>
<td>42°</td>
<td>0.743</td>
<td>50.0°</td>
</tr>
<tr>
<td>Green</td>
<td>500 nm</td>
<td>1.519</td>
<td>41°</td>
<td>0.755</td>
<td>51.5°</td>
</tr>
<tr>
<td>Blue</td>
<td>400 nm</td>
<td>1.528</td>
<td>40°</td>
<td>0.766</td>
<td>53.0°</td>
</tr>
</tbody>
</table>
Scientific Visualization using Shaders

Bump-Mapping for Terrain Visualization

Visualization by Nick Gebbie
2D Line Integral Convolution

At each fragment:
1. Find the flow field velocity vector there
2. Follow that vector in both directions
3. Blend in the colors at the other fragments along that vector

3D Line Integral Convolution

Visualization by Vasu Lakshmanan
Geometry Processing using Shaders

A Shader-eye View of the Graphics Process, with Geometry Shaders
Expanding 4 Points into a Bezier Curve with a Variable Number of Line Segments

FpNum = 5
FpNum = 25

Sphere Subdivision starting with 8 triangles

Level = 0
Level = 1
Level = 2
Level = 3
Shrinking Triangles

Geometry Silhouettes
There are a lot of GPU shader programming resources at:

http://cs.oregonstate.edu/~mjb/cs519

http://cs.oregonstate.edu/~mjb/glman

Mike Bailey, mjb@cs.oregonstate.edu
Input Devices and Interaction

SIGGRAPH 2008

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

Logical Input Device Types

- Choice
- Keyboard
- Valuators
- Locators
Choice

• Return a choice that has been made
• Examples: function keypad, button box, foot switch
• Often provide sensorial feedback: lights, clicks, feel, . . .
Keyboard

- Returns keys with specific meanings
- Letters, numbers, etc.

Valuators

- Return a value for something
- Example: knobs
- Can usually specify gain, minimum, and maximum
- All locators can also double as valuators
Dial Box

Locators

- Return the location of the screen cursor
- Examples: mouse, tablet, trackball, touchscreen
- Display-to-Input ratio
Locator Display-to-Input Ratio

*DTI Ratio*: the amount of cursor movement on the screen divided by the amount of hand movement

- Large: good for speed
- Small: Good for accuracy

Also called *Gain*

Ways to Read an Input Device

- **Sampling**: What is its input *right now*?
- **Event-based**: Wait until the user does something
3D Input devices

- Read a 3D position
- Returns 3 numbers to the program: an \((x,y,z)\) triple
- Some also return 3 more numbers to the program: three rotation angles
- Examples: digitizer, spaceball, glove

3D Electromagnetic Tracker
3D Mechanical Tracker

3D Scanners
Other 3D Input Devices

Force Feedback in 3D
Force Feedback in 3D

Force Feedback in 2D
Other Input Devices

Game Controller

Mechanical 3 DOF Orientation Tracker

Input Devices and Interaction

Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu
Where to Find More Information about Computer Graphics

Mike Bailey
Oregon State University

1. References

1.1 General Computer Graphics

SIGGRAPH Online Bibliography Database:
http://www.siggraph.org/publications/bibliography


Jim Blinn, A Trip Down the Graphics Pipeline, Morgan Kaufmann, 1996.


SIGGRAPH Conference Final program.

1.2 Math and Geometry


### 1.3 Scientific Visualization


William Schroeder, Ken Martin, and Bill Lorensen, *The Visualization Toolkit*, 3rd Edition,


### 1.4 Color and Perception


### 1.5 Rendering


### 1.6 Images


### 1.7 Animation


### 1.8 Games


http://www.gamedev.net

http://www.gamasutra.net

1.9 Virtual Reality


1.10 The Web


1.11 Shaders


http://www.clockworkcoders.com/ogsl1
1.12 Miscellaneous


2. Periodicals

*Computer Graphics and Applications*: published by IEEE
(http://www.computer.org, 714-821-8380)

*Computer Graphics World*: published by Pennwell
(http://www.cgw.com, 603-891-0123)

(http://www.akpeters.com, 617-235-2210)

*Game Developer*: published by CMP Media
(http://www.gdmag.com, 415-905-2200)
(Once a year publishes the *Game Career Guide*.)

*Computer Graphics Quarterly*: published by ACM SIGGRAPH
3. Professional organizations

ACM................ Association for Computing Machinery
http://www.acm.org, 212-869-7440

SIGGRAPH..... ACM Special Interest Group on Computer Graphics
http://www.siggraph.org, 212-869-7440

SIGCHI......... ACM Special Interest Group on Computer-Human Interfaces
http://www.acm.org/sigchi, 212-869-7440

IEEE ............... Institute of Electrical and Electronic Engineers
http://www.computer.org, 202-371-0101

NAB .............. National Association of Broadcasters
http://www.nab.org, 800-521-8624

ASME.............. American Society of Mechanical Engineers
http://www.asme.org, 800-THE-ASME

IGDA............... International Game Developers Association
http://www.igda.org, 856-423-2990

4. Conferences

ACM SIGGRAPH:
2008: Los Angeles – August 11-15
2009: New Orleans
http://www.siggraph.org/s2008

IEEE Visualization:
2008: Columbus, OH – October 19-24
http://vis.computer.org

Game Developers Conference:
2009: San Francisco, CA – March 23-27
http://www.gdconf.com
5. Graphics Performance Characterization

The GPC web site tabulates graphics display speeds for a variety of vendors' workstation products. To get the information, visit:

http://www.spec.org/benchmarks.html#gwpg