Real-Time Graphics Architecture

Kurt Akeley
Pat Hanrahan

http://www.graphics.stanford.edu/courses/cs448a-01-fall

Performance Analysis and Characterization
Topics

1. Tracing and quantitative analysis
2. Applications and scenes
3. Triangle size and depth complexity
4. Trends, maxims and pitfalls

Readings

Required
1. J. C. Dunwoody and M. Linton, Tracing interactive 3D graphics programs
2. M. Deering, Data complexity for virtual reality: Where did all the triangle go?
Graphics Performance Analysis

Goals:
1. Characterize application workloads
2. Understand system performance under workloads
3. Simulate new architectures

Tracing

Application

OpenGL.dll

Hardware
Tracing

<table>
<thead>
<tr>
<th>OpenGL Trace</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGLT.dll</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware</td>
</tr>
<tr>
<td>glt: Kekoa Proudfoot</td>
<td></td>
</tr>
</tbody>
</table>

Comments
1. Enabled by simple function pointer interface (jump tables)
2. Must not perturb interaction

Tricks with Dynamic Libraries

Ability to insert GL filter is very useful
1. Convert to postscript
2. Realistic and non-photorealistic rendering
3. Debugging (application or architect)
4. Network transparent graphics
5. Stereo, rendering to tiled displays, caves, etc.
6. Regression testing
7. Reverse engineering
8. Cheating: player can turn opaque polygons into transparent polygons
9. Stealing models: capture scene geometry
Tracing

Scenes: Head (240 frames)

| Vertices  | 60104 |
| Triangles (3D) | 59592 |
| Triangles (2D)  | 24884 |
| Fragments    | 263369 |
| Image        | 1024×768 |
### Scenes: Light (101 frames)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>1800116</td>
</tr>
<tr>
<td>Triangles (3D)</td>
<td>900058</td>
</tr>
<tr>
<td>Triangles (2D)</td>
<td>106503</td>
</tr>
<tr>
<td>Fragments</td>
<td>1818726</td>
</tr>
<tr>
<td>Image</td>
<td>1024×768</td>
</tr>
</tbody>
</table>

### Scenes: QTVR (734 frames)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>145.8</td>
</tr>
<tr>
<td>Triangles (3D)</td>
<td>116.6</td>
</tr>
<tr>
<td>Triangles (2D)</td>
<td>94.2</td>
</tr>
<tr>
<td>Fragments</td>
<td>786431</td>
</tr>
<tr>
<td>Image</td>
<td>1024×768</td>
</tr>
</tbody>
</table>

---

**CS448 Lecture 4**

Kurt Akeley, Pat Hanrahan, Fall 2001
Scenes: Town (1338 frames)

738541 glVertex3fv
728673 glTexCoord2fv
224682 glColor4fv
206474 glNormal3fv
201074 glCallList
180574 glBegin
180574 glEnd
168356 glLoadIdentity

728673 glVertex3fv
728673 glTexCoord2fv
224682 glColor4fv
206474 glNormal3fv
201074 glCallList
180574 glBegin
180574 glEnd
168356 glLoadIdentity

Vertices | 4326.8
---|---
Triangles (3D) | 2535.3
Triangles (2D) | 939.0
Fragments | 1353892
Image | 1024×768

Scenes: Flight (123 frames)

588499 glVertex3fv
565531 glNormal3fv
487229 glTexCoord2fv
94632 glBegin
94632 glEnd
26178 glColor4fv
1985 glEnable
1971 glDisable
1368 glPopMatrix
1368 glPushMatrix
1338 glLoadIdentity
1338 glMatrixMode

588499 glVertex3fv
565531 glNormal3fv
487229 glTexCoord2fv
94632 glBegin
94632 glEnd
26178 glColor4fv
1985 glEnable
1971 glDisable
1368 glPopMatrix
1368 glPushMatrix
1338 glLoadIdentity
1338 glMatrixMode

Vertices | 3932.8
---|---
Triangles (3D) | 2843.3
Triangles (2D) | 553.0
Fragments | 1004604
Image | 1024×768
Scenes: Quake (330 frames)

Vertices | 3627.0
---|---
Triangles (3D) | 1801.5
Triangles (2D) | 937.4
Fragments | 1855471
Image | 1024×768

869677  glTexCoord2f
531658  glVertex3fv
528161  glVertex3f
351303  glColor3f
221434  glTexCoord2fv
182997  glBegin
182997  glEnd
88470  glColor3ubv
31292  glVertex2fv
10657  glRotatef
9843  glBindTextureEXT
5160  glTranslatef
4285  glTexImage2D
4188  glDisable
3643  glShadeModel
3532  glEnable
3037  glBlendFunc
3036  glDepthMask
3019  glPopMatrix
3019  glPushMatrix
1821  glScalef
1606  glColor3ub
1518  glLoadIdentity
1300  glLoadIdentity

Viewperf OpenGL Benchmark

Alias/Wavefront Advanced Visualizer AWadvs-04
DesignReview DRV-07
IBM Data Explorer DX-06
Lightscape Light-04
Parametric Technology ProCDRS-03
Pro/Engineer medMCAD-01

http://www.specbench.org/gpc/opc.static/opcview.htm
Fragment Formula

Performance: $T$ a-pixel triangles per frame

$$a \equiv \frac{F}{T} \Rightarrow F = aT$$

Parameters

- $T$ = Number of triangles
- $a$ = Average area of a triangle
- $F$ = Number of fragments

Per-frame and per-second related by fps

Triangle Area Implications

The average triangle area $a$ represents a balance point between the floating point computation needed to process a triangle independent of pixel area, and the framebuffer fill capacity.

Implications:
- Triangles with average number of pixels greater than $a$ typically will render at a rate less than $T$, because the triangles are fill-dominated.
- Triangles smaller than $a$ pixels will render at a rate no faster than $T$, as such triangles are geometry-limited.
Deering Study

150 optimized triangulations of 3D objects from the Viewpoint, these are created from hand-digitized solid objects, rendered at 700 by 700.

<table>
<thead>
<tr>
<th>Model</th>
<th>Triangles</th>
<th>F</th>
<th>mean a</th>
<th>median a</th>
</tr>
</thead>
<tbody>
<tr>
<td>85skylark</td>
<td>2116</td>
<td>263933</td>
<td>255</td>
<td>59</td>
</tr>
<tr>
<td>R85skylark</td>
<td>2116</td>
<td>304895</td>
<td>305</td>
<td>57</td>
</tr>
<tr>
<td>86taurus</td>
<td>2458</td>
<td>278340</td>
<td>230</td>
<td>62</td>
</tr>
<tr>
<td>80deloreanM</td>
<td>2770</td>
<td>302871</td>
<td>228</td>
<td>51</td>
</tr>
<tr>
<td>83cutlass</td>
<td>3028</td>
<td>245286</td>
<td>156</td>
<td>39</td>
</tr>
<tr>
<td>camaro</td>
<td>3640</td>
<td>281127</td>
<td>155</td>
<td>35</td>
</tr>
</tbody>
</table>

Main result: distribution of triangle size in model and screen space is roughly exponential in the direction of small triangles. That is, the median triangle size is smaller than the mean triangle size.

Scene: Light
Scene: Flight

Scene: QTVR
Triangle Area Histogram Implications

Motivate two-types of rasterization
- Large triangles = amortize the cost of setup
  - Maximum per-triangle; minimum per-fragment
- Small triangles
  - Minimize the cost of producing a few pixels

Triangle Size vs. Time (SGI)

<table>
<thead>
<tr>
<th>Year</th>
<th>Product</th>
<th>F</th>
<th>T</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1984</td>
<td>Iris 2000</td>
<td>46M</td>
<td>10K</td>
<td>4600</td>
</tr>
<tr>
<td>1988</td>
<td>GTX</td>
<td>80M</td>
<td>135K</td>
<td>592</td>
</tr>
<tr>
<td>1992</td>
<td>RE</td>
<td>380M</td>
<td>32M</td>
<td>190</td>
</tr>
<tr>
<td>1996</td>
<td>IR</td>
<td>1000M</td>
<td>12M</td>
<td>83</td>
</tr>
</tbody>
</table>

Peak fill rates
Triangle Size vs. Time (NVIDIA)

<table>
<thead>
<tr>
<th>Season</th>
<th>Product</th>
<th>F</th>
<th>T</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>2H97</td>
<td>Riva 128</td>
<td>20M</td>
<td>3M</td>
<td>6.67</td>
</tr>
<tr>
<td>1H98</td>
<td>Riva ZX</td>
<td>31M</td>
<td>3M</td>
<td>10.33</td>
</tr>
<tr>
<td>2H98</td>
<td>Riva TNT</td>
<td>50M</td>
<td>6M</td>
<td>8.33</td>
</tr>
<tr>
<td>1H99</td>
<td>TNT2</td>
<td>75M</td>
<td>9M</td>
<td>8.33</td>
</tr>
<tr>
<td>2H99</td>
<td>GeForce</td>
<td>120M</td>
<td>15M</td>
<td>8.00</td>
</tr>
<tr>
<td>1H00</td>
<td>GeForce2</td>
<td>200M</td>
<td>25M</td>
<td>8.00</td>
</tr>
<tr>
<td>2H00</td>
<td>NV16</td>
<td>250M</td>
<td>31M</td>
<td>8.06</td>
</tr>
<tr>
<td>1H01</td>
<td>NV20</td>
<td>500M</td>
<td>30M</td>
<td>16.67</td>
</tr>
</tbody>
</table>

Depth Complexity

Definition:

\[ d \equiv \frac{F}{I} \implies F = dI \]

Quake

Color

Depth Complexity
**Depth Complexity**

**Definition:**

\[
F \equiv \frac{F}{I} \Rightarrow F = dI
\]

Quake

---

**Z-buffer Reads and Writes**

```
If(fragment.z < z[fragment.x][fragment.y]){  
c[fragment.x][fragment.y]=blend(fragment);  
z[fragment.x][fragment.y]=fragment.z;  
}
```

Probability of a write?

\[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \ldots \frac{1}{n}\]

Knuth: Analysis of Algorithms

H(n): Harmonic numbers; asymptotically \( \sim \log(n) \)

Best case: 1; Worst case: \( n \); Random case for \( d=4 \) is 2
Fill Rates

Need a minimum fill rate (d=1) to be interesting

For example: VR has high frame rates and hence require high fill rates

Providing high fill rates has been the major challenge to graphics architects

Depth Complexity is Bounded

High-quality rendering

Movie set analogy (don’t build parts of the environment that can’t be seen)

Well-written apps have low depth complexity

Culling and level-of-detail strategies (Performer)

Adds significant complexity to the application
80 Million Triangle Scenes?

Movie quality I = 10 MP (4K by 2.5K)

\[ F = d \cdot I = 4 \times 10 \text{ MP} = 40 \text{ MF} \]

\[ a = \frac{40 \text{ MF}}{80 \text{ MT}} = 0.5 \frac{F}{T} \text{ (Nyquist limit)} \]

Scaling up to 60 Hz yields 60 I/s * 80 MT/I = 4.8 BT/s

Assumptions:
- Culling limits \( d \) to 4
- Level of detail removes really small triangles

Loren Carpenter, Rob Cook, Alvy Ray Smith @ Lucasfilm

Constrained Design Space

\[ aT = dI \]

Parameters
- \( T \) = Number of triangles
- \( a \) = Average area of a triangle
- \( F \) = Number of fragments
- \( I \) = Image size
- \( d \) = Depth complexity
Design Strategies

1. Select cost-effective memory technology
   - Fixes memory bandwidth and hence fill rate
   - Processor capability determines triangle rate
   - Triangle area determined
   - Lampson and Thacker: GA must fully utilizes bw

2. Select performance goal
   - Target polygon count and average area
   - Image size and depth complexity determines fill rate
   - Interleave memory to achieve goal

Interframe ...
Performance Limiting Factors

Fill-limited -> Memory limited
Geometry-limited -> Compute- or interface-limited
Application-limited

Most applications are application-limited
- Fundamentally compute-limited
  E.g. marching cubes
- Inefficient use of the graphics system
  E.g. vtk rendering driver
Maxims and Pitfalls

Don’t design for last year’s scenes
Old benchmarks may not use new features; this presents a challenge since new systems may not necessarily accelerate old applications

Biggest challenge is balancing the system
Very difficult to simultaneously achieve both peak fill and geometry rates
Don’t evaluate systems using single-frames; use sequences
Image-Space Work Distribution

Parke - Tiled

Fuchs - Interleaved

Rasterization Cost

- Large tiles
  - Few tasks, greater variation in work
  - → bad load balance

- Medium tiles
  - More tasks, low overlap
  - → good load balance

- Small tiles
  - High overlap/more redundancy
  - → best load balance but redundant work
The Overlap Factor

\[
O = \left( \frac{H + h}{H} \right) \left( \frac{W + w}{W} \right)
\]

Molnar-Eyles Formula

Derivation! Reference!!