

Computer Graphics Hardware Acceleration for Embedded Level Systems

Brian Murray 02-15-2002

Topics

- Why?
- Introduction to Computer Graphics
 - Object Representation
 - Object Rasterization
- Graphics Hardware
 - Goals
 - Problems
 - Architecture
- Embedded System Optimizations

Object Representation

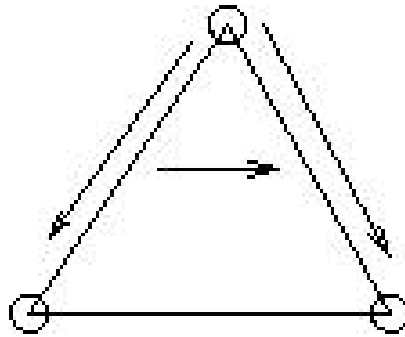
- Goals
 - Display images to user
 - Be able to represent images in abstract form
- Solution
 - Use specialized algorithms that take primitives and convert them into “pixels”.

Primitive Transformations

- Geometric Transformation
 - Move Primitives/Polygons (Triangles) to correct positions in 3D space.
 - Project 3D points onto 2D points
- Lighting
 - Use lights to correctly color and shade the triangles
- Rasterization
 - Convert the resulting objects (triangles) into pixels

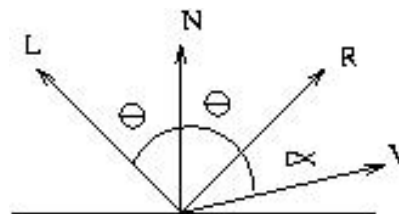
Rasterization - Interpolation

- Interpolation of Points
 - Use the coordinates at each point and find the points between.
 - Find the points between the points on the lines.



Rasterization – Lighting/Shading

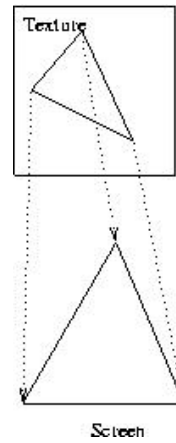
- Phong Illumination
 - Position of Lights
 - Orientation of polygon
 - Position of viewer
- Gouraud Shading
 - Interpolate colors of at the points of the triangle.



$$I = I_a k_a + \sum_{n=0}^{\text{lights}} [I_p (k_d \cos\theta + k_s \cos^n\alpha)]$$

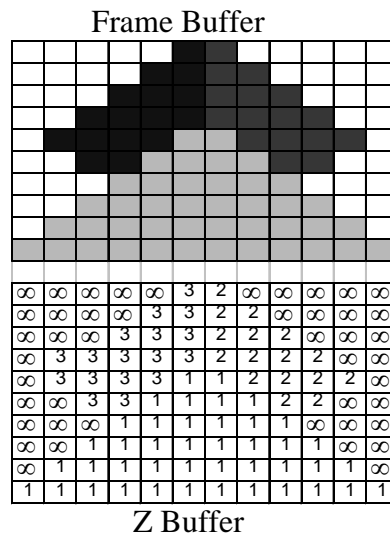
Rasterization - Texturing

- Map the points of a Texture on the points of a Triangle.
 - Each triangle point has a screen coordinate (x,y)
 - Each triangle point has a texture coordinate (u,v)
 - Use interpolation to find the points between.



Rasterization – Depth Checking

- Z-Buffer
 - Stores Z values of points as it is rasterizing
 - Check against the Z-Buffer to find out if you are near or farther from the screen.
 - If closer, write to Z-Buffer and Frame-Buffer (screen)



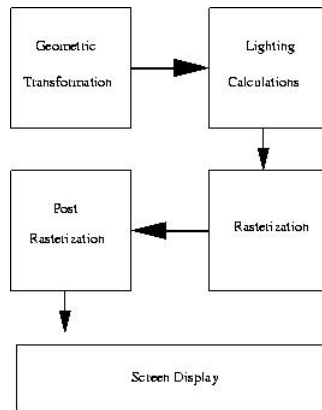
Graphics Hardware

- Goals
 - Implement the functionality of graphics processing in hardware.
 - Speed is usually the main factor
 - Can be very expensive

Graphics Hardware - Problems

- Problems
 - Many Many Memory Accesses
 - Large amounts of Floating point math
 - Must be efficiently pipelined and fast
 - Requires large amounts of memory, especially for multiple frame-buffers.

Graphics Hardware – General Architecture



Embedded System Concerns

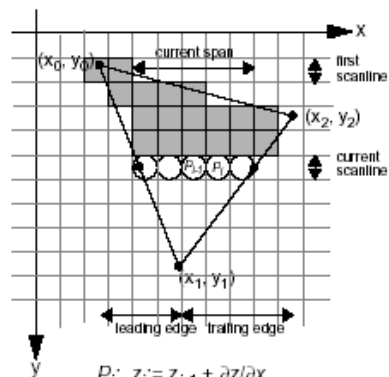
- Space
 - Cores must take up minimal area
- Power
 - Power constraints are tight
- Speed
 - Must be fast enough to be useful
- Memory
 - Memory is limited
- Heat
 - Speed and power demands generate heat, to which many systems are not tolerant

Embedded Solutions

- Off load Geometric Transformations to CPU
- Disregard Post-Rasterizer Functions
- Use an asynchronous pipeline
- Shut off portions not in use or already done
- Try to make maximum usage of memory available
- Do as many functions as possible in fixed point
- Simplify architecture and make system scalable
- Eliminate extra precision to minimize hardware

Rasterization Solution - DDS

- Sort all vertices from top to bottom
- Determine the x/y “slope” of each edge
- Determine the x starting point ($x = \text{slope} + \text{ceil}(y - y_1)$)
- Add to x the slope at each y while moving down the screen to get beginning and ending points.
- Interpolate between those points

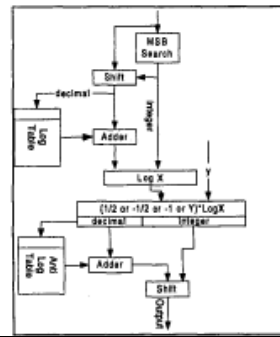
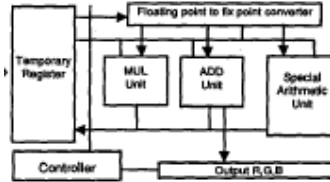


$$\begin{aligned}
 P_i: z_i &= z_{i-1} + \partial z / \partial x \\
 r_i &= r_{i-1} + \partial r / \partial x \\
 g_i &= g_{i-1} + \partial g / \partial x \\
 b_i &= b_{i-1} + \partial b / \partial x
 \end{aligned}$$

Kugler, A. "The Setup for Triangle Rasterization",
*Proceedings of the 11th Eurographics Workshop on
 Graphics Hardware*, Poitiers, France, August 1996, pp.
 49--58.

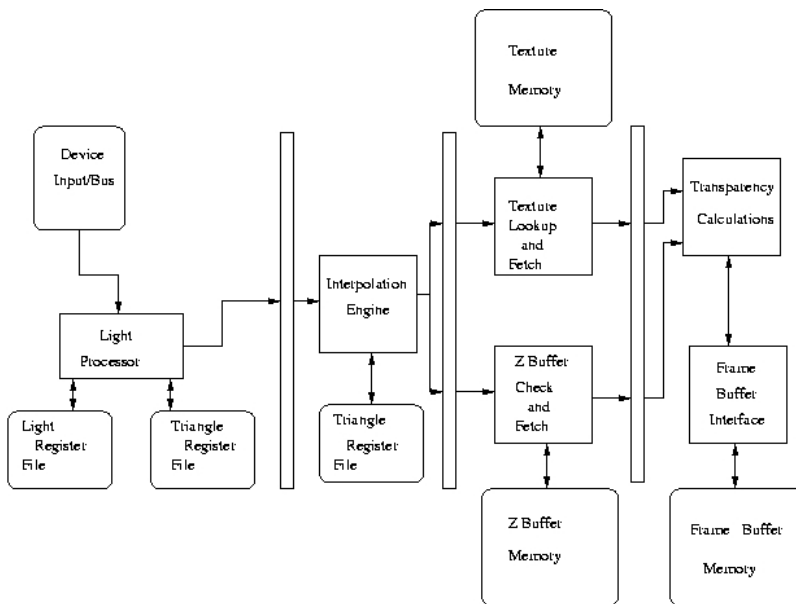
Fixed Point Lighting

- Convert all vector math to $\text{Log}(2)$ domain
- Use 16bit vectors
- Less than 1 percent color error opposed to floating point

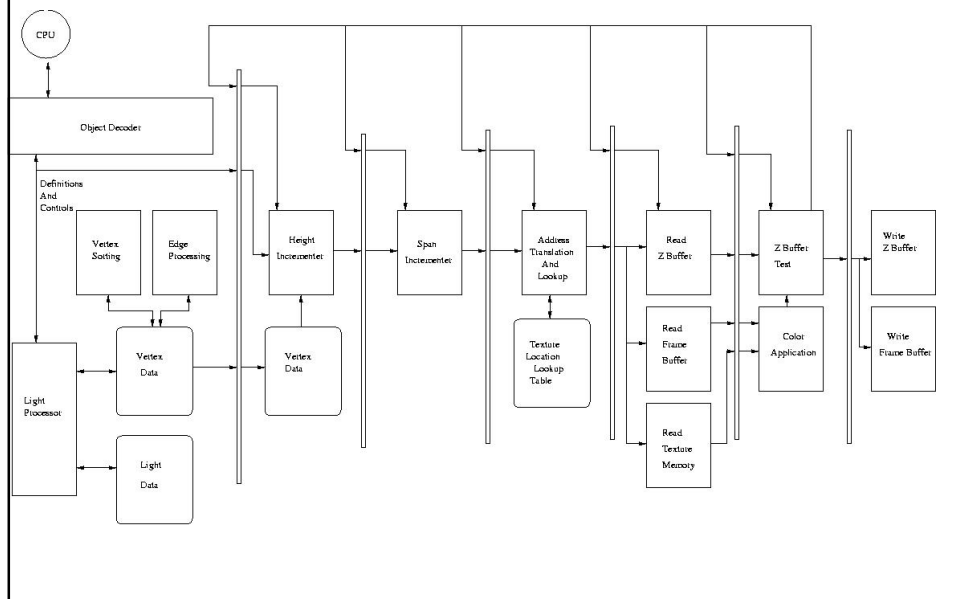


Chen, C-H; Lees, C-Y, "A cost effective lighting processor for 3D graphics application", Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on, Volume: 2, 1999, pp2 792-796.

Embedded Architecture 1



Embedded Architecture2



Verification Procedures

- Verification
 - Simulate memory accesses for cache optimization
 - Simulate Architecture for flow and control verification
 - Simulate Rasterization with fixed and floating to show possible error rates
 - Create a hardware model and verify with FPGA

Future Work

- Enhance an Embedded processor for geometric transformations
- Scale the system with a parallel architecture (i.e. mini-frame buffers)
- Add modern features such as mip-mapping, per-pixel operations, and z-buffer compression (or pyramidal)
- Cache implementations for the system : new localities and enhanced memories