GoForce 3D: Coming to a Pixel Near You

CEDEC 2004

NVIDIA Actively Developing Handheld Solutions

- Exciting and Growing Market
- Fully Committed to developing World Class graphics products for the mobile
- Already in active development

Why Make Games For Handheld Devices?

- Emerging Market
- Ubiquitous Mobile Devices
  - 500+ Million Units Worldwide
  - 3 billion dollars in ringtone downloads
  - 45 Million of 60 Million gamers have used handset for games
- Casual Gamers and Enthusiasts

Why Make Games For Handheld Devices?

- Technological Innovation raising the bar
  - Real-time 3D
  - Wireless Connectivity
- Worldwide Potential to Make Money

Challenges...

- Limited System Resources
- Non-Homogenous Development Space
  - Diversity

... and more challenges

- Publishing and Distribution
- Digital Rights Management
- Unique Market Dynamics
  - Service Providers and ARPU
  - Increasing focus on DATA and Mobile Entertainment

Introducing... GoForce 3D

- Licensable 3D Core for Mobile Devices

- OpenGL ES / Direct3Dm compliant
- Low Power Architecture
- Integrated Unified SRAM
- Up to VGA resolution
- Modern Feature Set

- Targeted to run complex games at 30+Hz

GoForce 3D Feature Set

- Geometry Engine (float and fixed point)
- 16-bit color w/ 16-bit Z
- 40-bit color (internal)
- Multi-texturing w/ up to 4 simultaneous textures
- Bilinear / Trilinear Filtering
- Flexible Texture Formats
  - 4-bit/8-bit palletized, DXT1 compression, more
- Fully Perspective Correct (color included)
- Sub-Pixel Accuracy
- Per-Pixel Fog
- Alpha Blending

Traditional Architecture

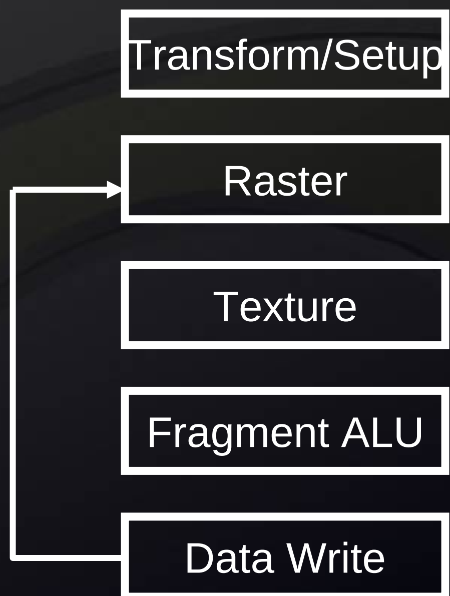
- Setup/Raster
- Texture Addr
- Texture
- Fog
- AlphaTest
- DepthTest
- AlphaBlend
- Mem Write

- **Deep pipeline (200 stages)**
- **Always have to go through all stages**
- **Optimized for OpenGL-style fast texturing**
- **Pipelines always clocking**
- **Fast, but too much power consumption**
- **~750mW per 100M pixel/sec**

(~200 pipe stages)

A Completely New Architecture

for Ultra Low Power



(~50 pipe stages)

- **Flexible Fragment ALU**
- **Raster – fragment generation and loop management**
- **Pipelines only trigger on activity**
- **Low Power**
  - **< 50 mW per 100M pixel/sec**
  - **During actual gameplay**
- **Very scalable architecture**

Why Geometry?

- Current state of Handheld Processor
  - Arm 7/Arm 9/+
  - Clock rates: 50Mhz – 400Mhz
  - No floating point
  - Host bus is shared with dram bus
  - Limited system memory
- Move as much processing onto the GPU
  - More power efficient
  - Better performance

Reduced pipeline for power savings

Depth Complexity = 1
Textured
No blend
No Z

Depth Complexity = 4
1. Textured tri, no blend
2. Textured tri, no blend with Z
3. Textured tri, no blend with Z
4. Textured tri, blend, with Z

Simple scenes don't require fog, blending, alpha test, and even depth comparison for every triangle.

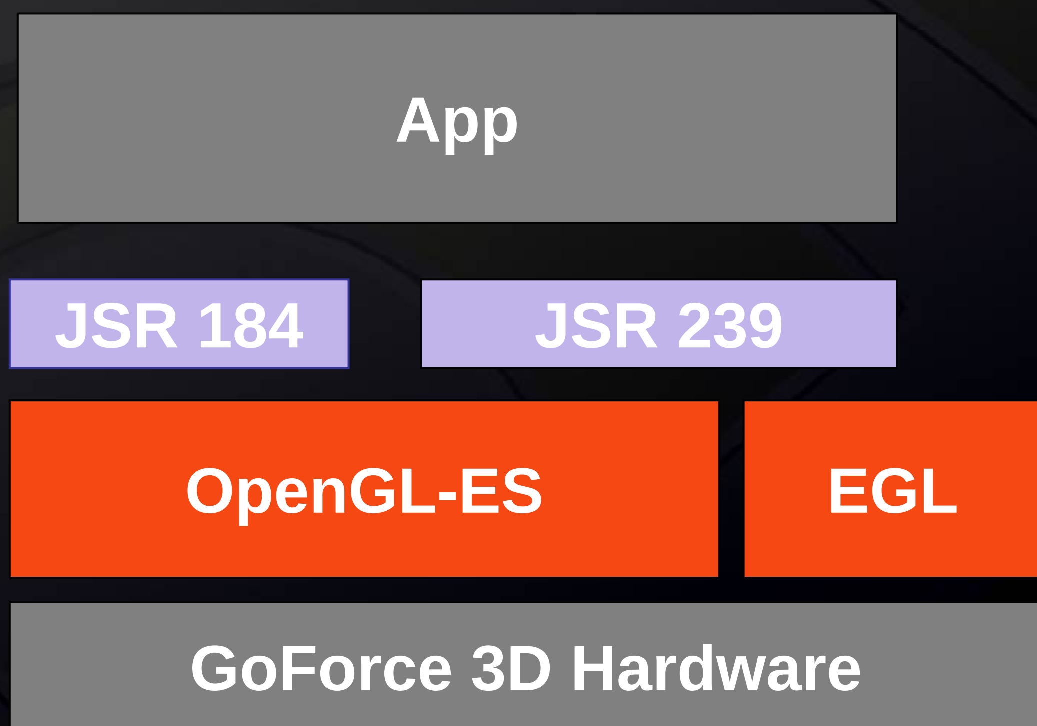
Rich 2D Features

- Solid color fill
- Source copy
- Alpha blending
  - Fixed alpha value for all pixels
- 16x16 Pattern fill
- Line draw
  - Sub-Pixel accurate
- Clipping & Transparency
  - Inside or outside clipping supported

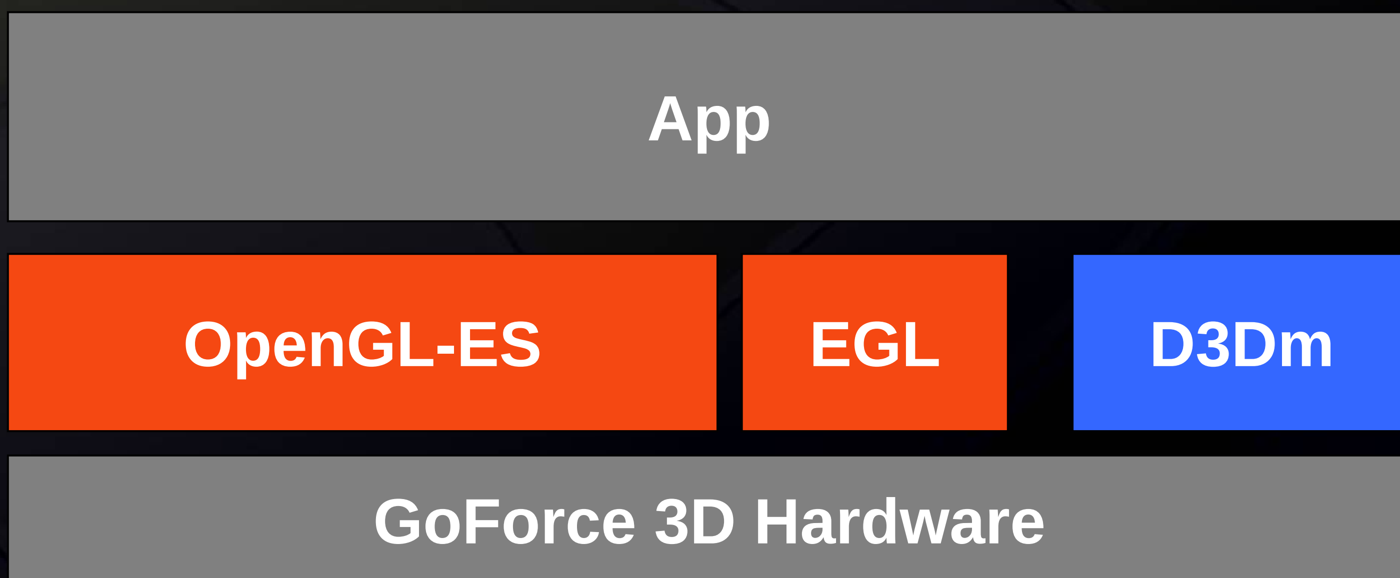
*n*Power Technology

- Automatic power-down of unused pipelines
- Normal, standby, and sleep modes Architecture-level power management
- Multiple Levels of Advanced Power Management
- Low-Voltage operation

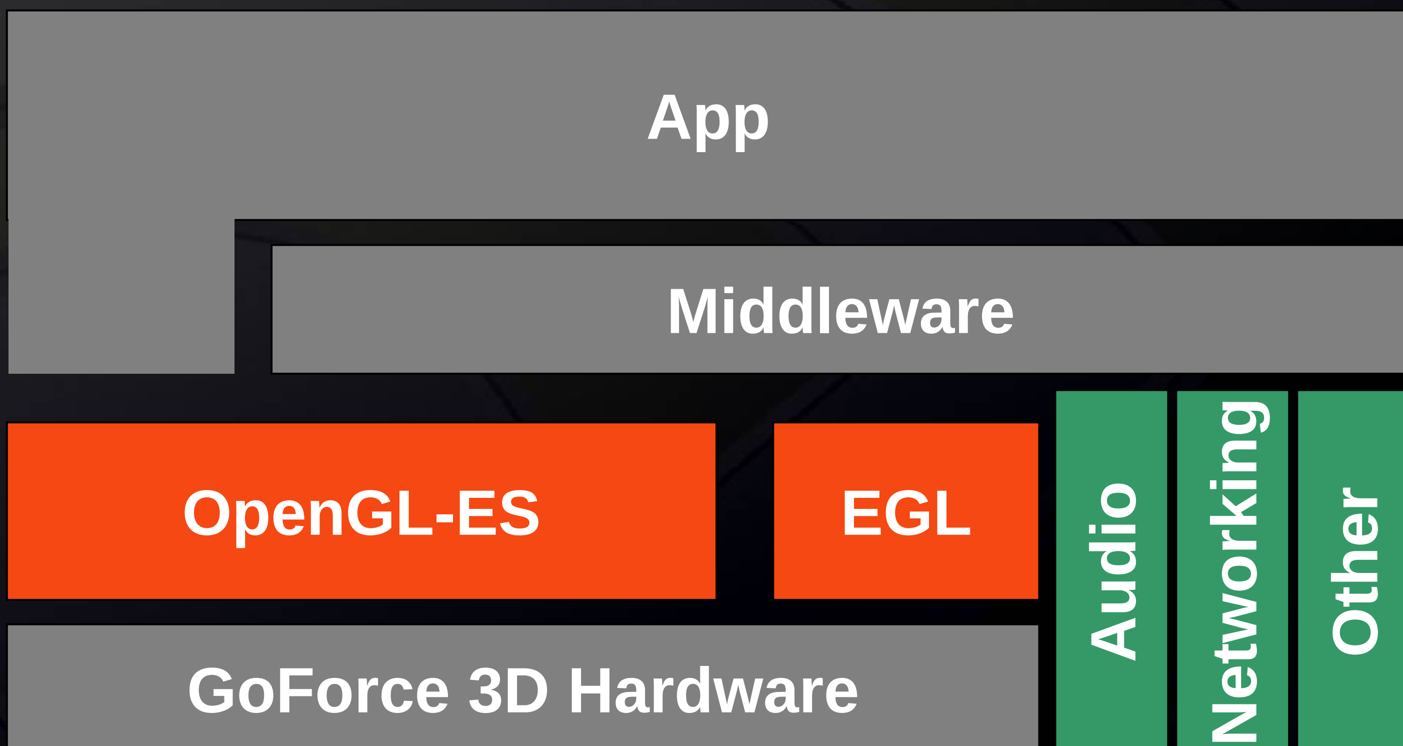
Java Programming Model



Native Programming Model



Middleware Programming Model



OpenGL ES 1.0 vs. OpenGL

- Roughly OpenGL 1.3
- Removes
  - Display List
  - glBegin/glEnd
  - Texgen
  - Environment Maps
  - Evaluators
- Adds
  - Fixed Point type/entry points
  - Byte type more universal

OpenGL ES 1.1 vs. OpenGL

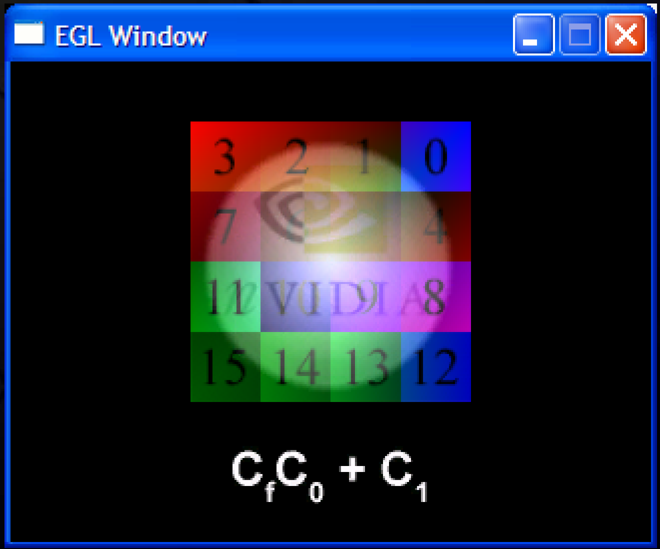
- Based on OpenGL 1.5 spec.
- Adds functionality to ES 1.0
  - Vertex Buffer Objects
  - Automatic Mipmap Generation
  - Enhanced Texture Combine Operations
  - User-defined clip planes
  - Point Sprites and Point Sprite arrays
  - Queries of dynamic states

Direct3Dm

- Not public yet
- Working w/ Microsoft

NVIDIA Handheld SDK

- Demos
  - OpenGL ES (101)
  - Feature Demos



- Porting layer
  - OpenGL-ES to OpenGL
  - Runs on PC
  - Get developers new to embedded up and running

NVIDIA HHDK (cont.)

- .NET Demo Wizard
  - Builds skeleton app w/ both x86 Windows and ARM Linux targets
- Tools and Libraries
  - DXT1 compression tools
  - DXT1 image loading library
  - Fixed point math library – optimized ARM math
- Documentation – GoForce 3D Overview

Development Kits

- Coming soon!
- Register for NVIDIA Handheld Developer Program
  - http://developer.nvidia.com
- Email
  - handset-dev@nvidia.com

Case Study: Bubble

- Originally authored for GeForce 256 desktop GPU (circa 2000)
- Deforming, Reflecting Surface
  - Spring-based physics
  - Environment mapping

- Ported to GoForce 3D
  - Goal: Understand the feasibility of implementing native graphics apps on GoForce 3D

Case Study: Bubble

- Demo

Bubble: Overview

- Sphere Model – Set of Vertices and Edges
- Set of forces
  - Impulse “Poke” Force
  - “Homeward” Force
  - Elasticity “Edge” Force
  - Outward “Swelling” Force

- Forces influence – velocity, position, and normal

Bubble: Deformation

- Simulation in Floating Point – VERY slow
- Profiler to identify problem areas
- Switched to integer math (s15.16)

Fractional bits

Integer bits

- Fixed Point – range vs. precision tradeoff
  - Alternate formats or rescaling

Bubble: Environment Mapping

- Original used Cube Mapping and Reflection Texgen
  - No support for either in ES 1.0
- Dual-Paraboloid Mapping w/ Manual Texture Coordinate Generation (fixed point)

Bubble: Texture Memory Usage

- Each scene uses 8 textures
  - 2 – 256x256 textures (mip-mapped)
  - 6 – 256x256 textures (non-mipmapped)

R5G6B5 – 16-bits/texel = 786432 + 349524 = 1.08Mb

DXT1 – 4-bits/texel = 196608 + 87381 = 0.27Mb

DXT1 is high quality and 25% the cost of R5G6B5

Bubble: Quality

Bilinear	Trilinear	Trilinear w/ LOD clamp
		
High frequency	High frequency near silhouette	Best Quality *

* Using SGIS_texture_lod

NVIDIA 3D Quality Demos

Running on OpenGL-ES wrapper for x86/Windows

Emulates what GoForce 3D hw handset graphics can generate.

NVIDIA Developer Site

- Register for NVIDIA Handheld Developer Program

developer.nvidia.com

Questions?

Handset-Dev@nvidia.com

Bubble: How it Works

- Sphere Model – Set of Vertices and Edges
  - Vertex
    - Position
    - Normal
    - Velocity
    - Average Velocity – average “neighborhood” velocity
    - Home Position – vertex “home” resting position
  - Edge
    - Pair of vertex indices
    - Home Length – initial edge length

Bubble: How it Works

- Deformation – apply forces to update model
  - Vertex
    - Position
    - Normal
    - Velocity
    - Average Velocity
    - Home Position
  - Edge
    - Pair of vertex indices
    - Home Length

Multi-Step Process...

Bubble: Deformation

- Step 1 – Updating the Velocities
- Adjust based on spring forces
  - “Homeward” force
  - “Outward” force
  - “Edge” force (i.e. elasticity)

foreach vertex

```
vel += HomeForce( home – pos )
vel += OutwardForce( normal )
```

foreach edge

```
vert[v0].vel += EdgeForce( vert[v0].pos – vert[v1].pos )
vert[v1].vel += EdgeForce( vert[v1].pos – vert[v0].pos )
```

Bubble: Deformation

- Step 2 – Filter Velocities
  - Compute Average Velocities
  - Apply Filter – vel = 0.9 * vel + 0.1 * avg
- Step 3 – Update Positions
- Step 4 – Apply Drag to Velocities
- Step 5 – Compute Normals
  - Iterate over all triangles, use cross-product of edges

Bubble: Poking

- Requires Instantaneous velocity update
- Find closest point to “pick ray”
  - Eye Pos: (0,0,0)
  - Pick Ray: (screen_x,screen_y, -near)
- Apply inward pulse force based on distance

p.vel += PulseForce(distance(closest.pos, p.pos))

where PulseForce(d) = k_1 * Pow(d, -20)

Bubble: Deformation (revisited)

- Step 2 – Filter Velocities
  - What happens if we don't filter the velocities?

Simulation becomes unstable.