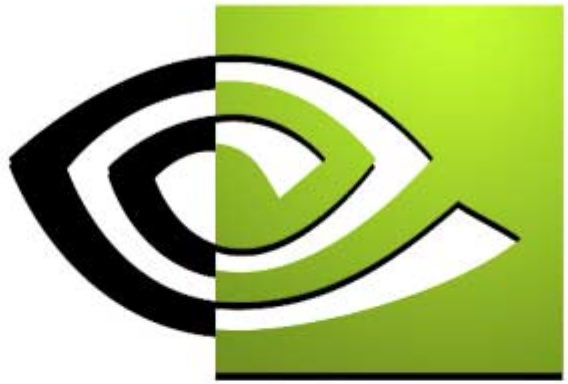


NVIDIA™

What comes after 4?

Richard Huddy, NVIDIA Corporation

RichardH@nvidia.com

The Past

- **A brief history of mine…**
- **Moore's Law cubed**
  - **Performance doubles ever 6 months**
  - **(Moore's prediction is a doubling every 18 months)**

NVIDIA.

Before we start on consumer graphics

- In 1992 Silicon Graphics (SGI) launched the “Reality Engine”.
- $1,000,000
- 8 Parallel graphics processors
  - Each with it’s own LED!
- Integrated hardware transform, lighting and rasterization (including texture mapping!)
- About the size of a domestic fridge…

NVIDIA

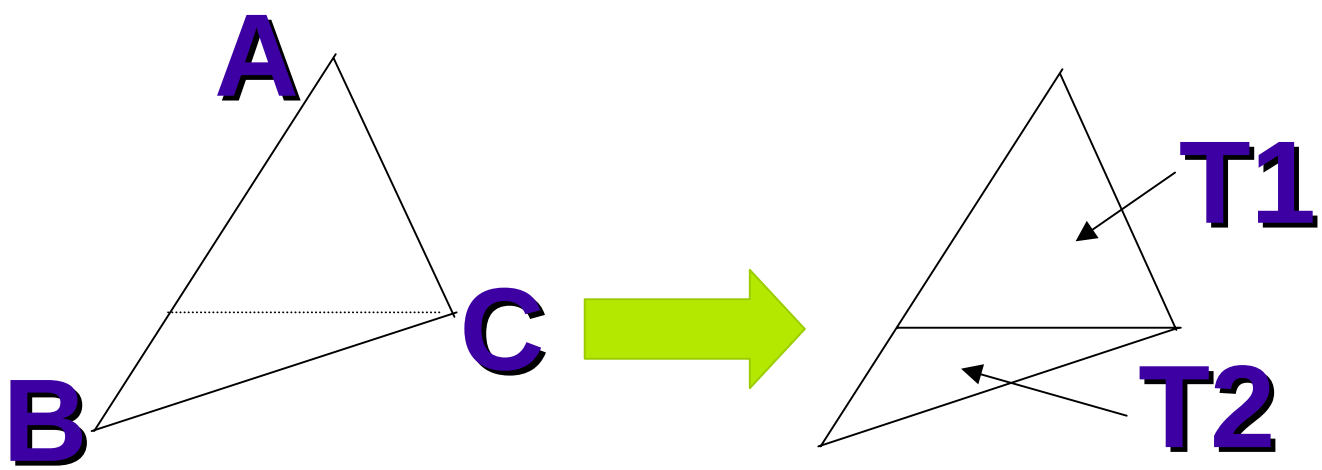
History of consumer PC graphics

- Up to 1995
  - 2D only…
    - Dominated by S3, Cirrus Logic, Tseng Labs, Trident
- 1995 Scanlines (Proprietary APIs)
- 1996 Trapezium rendering (introduction of DX3)
- 1997 Triangle rendering (… DX5)
- 1998 Triangle setup (…DX6)
- 1999 Multiply pipelined architectures (…DX7)
- 2000 Transform and lighting (…DX8)
- 2001 Programmable Shaders



Trapezium Rendering

- **Any arbitrary triangle can be split into two screen aligned triangles…**



… each of T1 and T2 is a degenerate screen aligned trapezium (which is what the hardware wanted)

NVIDIA.

Triangle Rendering (Riva128)

- No longer needs to split into screen aligned trapeziums ☺
- But… still needs the CPU to calculate all edge interpolators ☹
  - Sx, Sy, Sz, RHW, Diffuse, specular, U, V, fog, alpha
    - For all three edges ☹☹

Sx, Sy – Screen coordinates

Sz – Z buffer value

RHW for perspective correction of:

U, V, (Diffuse, Specular, fog, alpha)

NVIDIA

Triangle setup (Voodoo 2)

- Now we're getting somewhere!
- Pass in 'TLVERTEX' style data…
  - Sx, Sy, Sz, RHW, RGBA, fog, specular, U, V
  - (That's 32 bytes per vertex or 96 bytes per triangle)
- Typically this is bus/memory bandwidth limited (which makes it productive to cull on the CPU)
  - Remember this is the PCI or AGP1X generation
- The integrated setup engine determines all of the edge interpolation values

NVIDIA

Multiply Pipelined Architectures (TNT)

- TNT2 ("TwiN Texel")
  - 2 Pixel pipelines for effective 2x fill rate
  - 2 Texture units
    - e.g. Base texture + Light map
  - Architectural flexibility which gives great performance
  - Dual texturing is more efficient:
    - Write once is better than
    - Write… read, modify, write.
  - But two pixels per clock is the common case

NVIDIA

Transform and Lighting (GeForce)

Plenty of new opportunities for cleverness…

- **Clipping is non-trivial**
- **Should we use the guardband as a clipper?**
- **Where should we light?**
  - **Model space, World space, Camera space?**
- **How parallel can we make it?**
  - **Very…**
    - **Typical pipeline now has several hundred stages**
    - **Which means stalls are now *very* expensive**

NVIDIA

Transform and Lighting chips?

- **In terms of rasterization, lighting, clipping and transformation… it out performs the 1992 Reality Engine!**
- **From a $1,000,000 box to a sub $100 consumer board in 9 years...**
- **Moore's law suggests a cost of ~$15,000 on the basis of a halving in cost every 18 months**
- **But then, maybe that's partly why we have sold roughly 50 million of them...**



That 5 year period…

- Is a process of absorption
  - Taking a well-defined task away from the CPU
- Has produced a highly parallel chip
  - Typical chips now have 200+ stages
- Is nearing completion
  - The standard pipeline is now fully implemented
- Is quite unlike what's coming in the next 5 years
  - Because we've done the standard work, now it's time to add radical new technologies…

NVIDIA

The Present

- **The market is at the second “Inflection point”**
- **i.e. Unless you understand the business trends, then you cannot predict the future from the past**
- **Or “this is where interpolation breaks down”**

NVIDIA.

Current state of the art

- Hardware Transform and Lighting
  - Over 80 million vertices (triangles) per second
  - 8 hardware lights
- Hardware texture coordinate generation and manipulation
- Very high fill-rates
  - Now well in excess of a gigapixel
- Multi-texturing
  - Dual texture commonplace, more emerging

NVIDIA

Quick comparison: PC vs Console

- **Higher integration**
  - **PC: Many chips (but few graphics chips)**
  - **Console: Few chips (and only one for graphics)**
- **Costs**
  - **PC: Top graphics card costs as much as a console**
  - **Console: Graphics chip can be $10 to $50 max**
- **PlayStation 2**

4MB embedded video ram

16 pixels per clock

High programmability

No multi texture…

No bumps…

No H/W AA…

NVIDIA.

Public XBox info

- 116 Million drawn polys per second
- Over 1 gigapixel per second
- Runs DirectX 8
- Quad texture
- 64Mb UMA (effective 400MHz memory)
- Intel CPU (Pentium III at 733MHz)
- NVIDIA GPU (with integrated North Bridge)
- NVIDIA MCPX (South Bridge with integrated 3D-sound, ethernet, modem, joystick controller, USB, Dolby Digital encode, IDE, etc)

NVIDIA.

The Future

- What solutions are being examined and which problems do they specifically address?
- And, where do we want to be?

NVIDIA.

Where do we go from here?

- The tough problems:
  - Where will we find the extra memory bandwidth?
    - DDR, multi-chip solutions, tile based architectures
    - Embedded DRAM
  - How many textures are ‘enough’?
    - Games programmers have uses for 8 or more (sigh!)
  - When is fill-rate high enough?
    - [Hint: Never]
  - Pixel quality
    - Anti-aliasing, Phong shading, Blinn bump mapping
  - Programmability (the escape from conformity)

NVIDIA

Where are we trying to get to?

- If we're aiming at film quality rendering in real-time then we have some way to go…
- Toy Story took an average of 7 hours per frame to render (max ~90 hours)
- Alvy Ray Smith (MS Graphics Research Fellow & Pixar tech guy) would like 80M polys per frame
  - That's 4.8 billion polys per sec at 60Hz
- "Shrek" main characters have ~800K polys each
- And that's all Renderman rendered...

NVIDIA

DX8 - Minor refinements to the API

- Index Buffers
  - To save memory bandwidth
- Vertex Streams
  - To allow the app to Lock only what it will truly modify
    - (e.g. keep texture coordinates in a separate stream)
- Point sprites
  - Compact representation of screen aligned entities
- Resources can now all be managed by the system
  - Reduces contention between competing types

NVIDIA

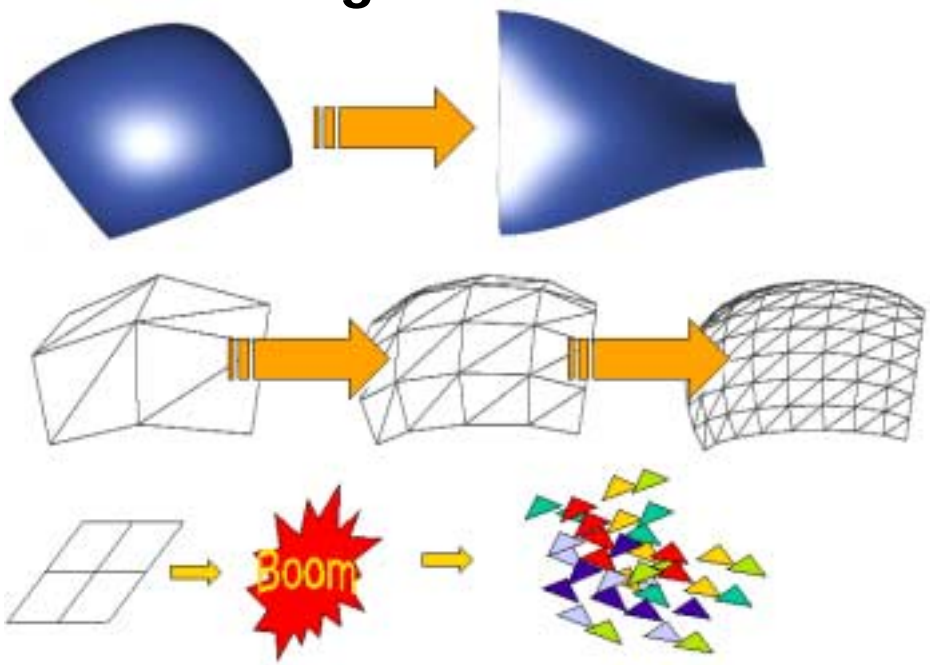
The radical new API features

- Higher Order Surfaces (Bezier patches, Nurbs patches etc)
  - Compact representation of detail
  - Supported by many modelers
  - But they introduce new problems
    - Shading continuity
    - Collision detection consistency
- Vertex Shaders
  - Custom transform and lighting functions
- Pixel Shaders
  - Custom texturing and lighting effects

NVIDIA

Higher Order Surfaces

- A practical alternative to dense polygon meshes
  - Under app control (good for animation and LOD)
  - Several types to choose from
  - Decimate to triangles



NVIDIA.

Vertex Shaders

- Take over the transform, lighting and texture coordinate manipulation.
- Can totally replace existing fixed-function code
- Gets programmers back to writing assembler!

- Extraordinary flexibility to do things like -
  - Irregular transforms
  - Custom lighting
  - Procedural geometry deformation
  - DOT3 lighting setup
  - Shadows, animation, etc.

NVIDIA

Pixel Shaders

- Replace the standard shading and texture operations with your own micro-code.
- More ‘trivial’ assembler to write
  - i.e. few instructions in each program
- Very direct control at the pixel level
- Increasingly orthogonal instruction set
  - Numerous permutations make sense
  - Does everything the previous API used to do, and much more (so no more ‘random’ pixel modes)

NVIDIA

What else will we need?

- We already have vertex and pixel processors…
- We also need to add a primitive processor:
  - With access to the connectivity information of the mesh
  - So that you can destroy primitives inside the GPU
    - e.g. Local LOD management were it counts
  - So that you can create primitives inside the GPU
    - Implementation of arbitrary HOS in your app without me having to tell my architecture team exactly what you want as much as three years before you want it

NVIDIA

Where is the industry going?

- **Towards:**
  - **Full programmability**
  - **Subdivision surfaces**
  - **Displacement maps**
  - **Very high quality pixels**
- **Away from:**
  - **Fixed function**
  - **Triangle meshes**
  - **Flat textures**
  - **Traditional lighting models**

This may take quite some time…

NVIDIA

Subdivision Surfaces…

- A very compact and flexible model description
- Advantages:
  - Amenable to hardware implementation
  - Great for dynamic LOD

A good place to start learning…

- http://research.microsoft.com/~hoppe/siggraph94/node1.html
- http://www.cs.princeton.edu/gfx/proj/dss/

NVIDIA

Displacement Maps

- Are a form of “Geometry amplification”
- Compact representation of detail
- Basic implementation amenable to hardware
- Issues…
  - Modelers don’t think that way
  - Sampling and filtering not well developed
  - Joining of adjacent primitives without gaps

NVIDIA

Displacement mapped surfaces



Control Mesh



Smooth Domain Surface



Displaced Subdivision Surface

NVIDIA.

Other features on the horizon…

- Greater color precision
  - To better support many-pass rendering
- Hardware shadow maps
  - Because stencil is CPU intensive, inefficient and doesn't usually look great
    - Shadows in the real world are soft-edged
- Higher quality texture sampling
  - Because tri-linear just can't cut it
- The eventual aim is the Renderman feature set in consumer graphics

NVIDIA

Isn’t Renderman shading slow?

- Yes so...
  - Consider laying down the Z buffer first
  - Draw your objects into front-to-back order
    - But this isn’t a per-poly sort...
- This allows you to minimise the overall cost by not spending time on unseen pixels
- But this means you pass all the vertex data thru the GPU twice per frame
  - And that means you need very fast vertex engines

NVIDIA.

Summary

- PC Graphics is a very fast moving business
  - There are no survivors from 5 years ago
  - Last years top chips are mainstream
  - Last years mainstream chips look pretty dated
- R&D goes straight into the chips from now on…
- Is currently driven by games
  - Ooooh! That's you guys!

NVIDIA

Questions?



Richard Huddy

NVIDIA Corporation

RichardH@nvidia.com