GameDevelopers Conference

EVOLVE GDC>04

Optimization for DirectX9 Graphics

Ashu Rege

NVIDIA®

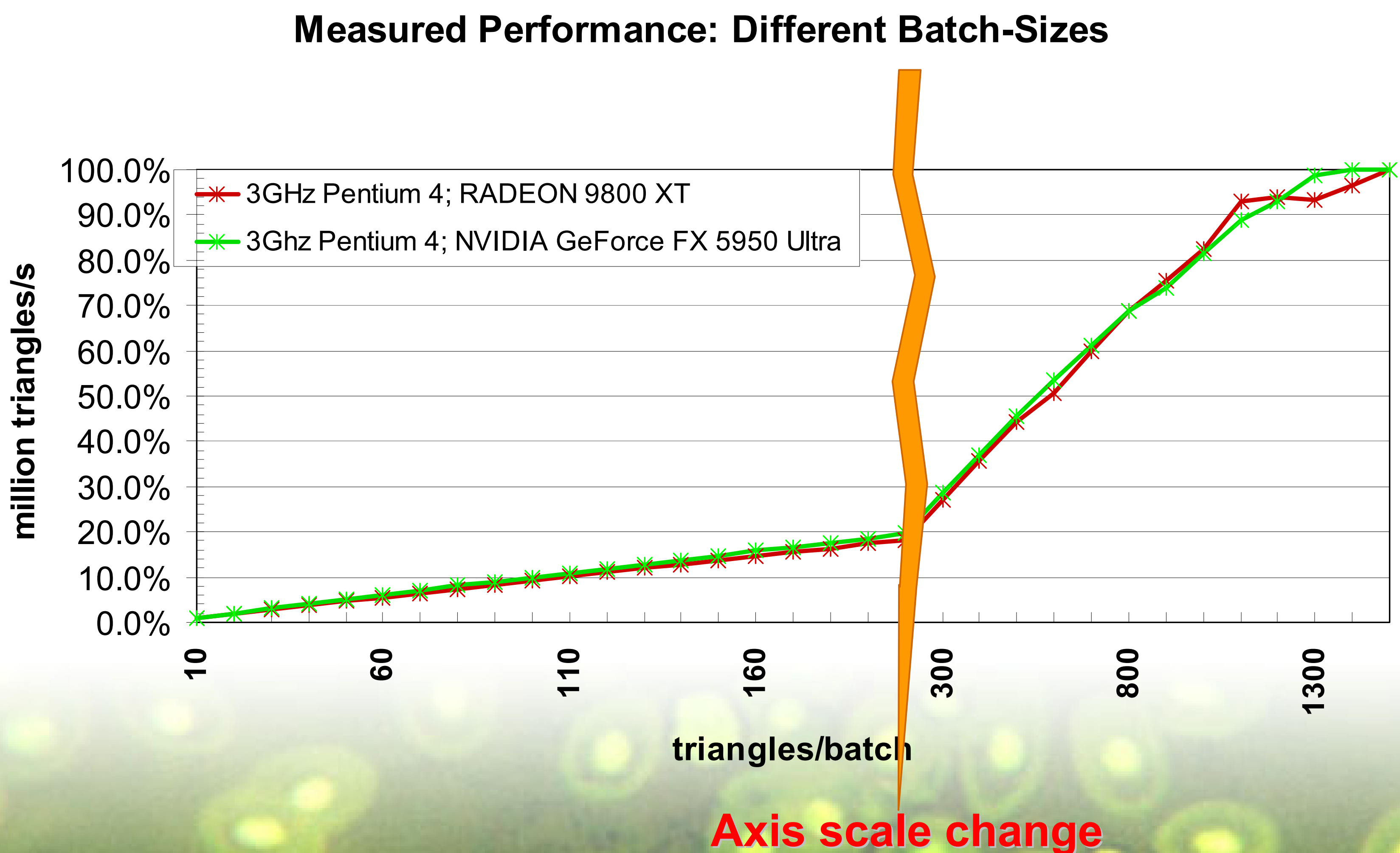
Last Year: Batch, Batch, Batch

- Moral of the story: Small batches BAD
- What is a “batch”
  - Every DrawIndexedPrimitive call is a batch
  - All render, texture, shader, ... state is same

Simple Test App

- Degenerate Triangles (no fill cost)
- Post TnL Cache Vertices (no xform cost)
- Static Data (minimal AGP overhead)
- Fixed (~100 K) Tris/Frame
- Vary Number of Batches

Last Year's Graph Updated



This Year: Son Of A Batch

- What makes an app ‘batchy’?
  - Too many state changes
- What kinds of state changes?
- Techniques to reduce batches

State Changes

- Analysis of some popular games
- Top State Changes:
  - Texture State
  - Vertex Shaders and Vertex Shader Constants
  - Pixel Shaders and Pixel Shader Constants

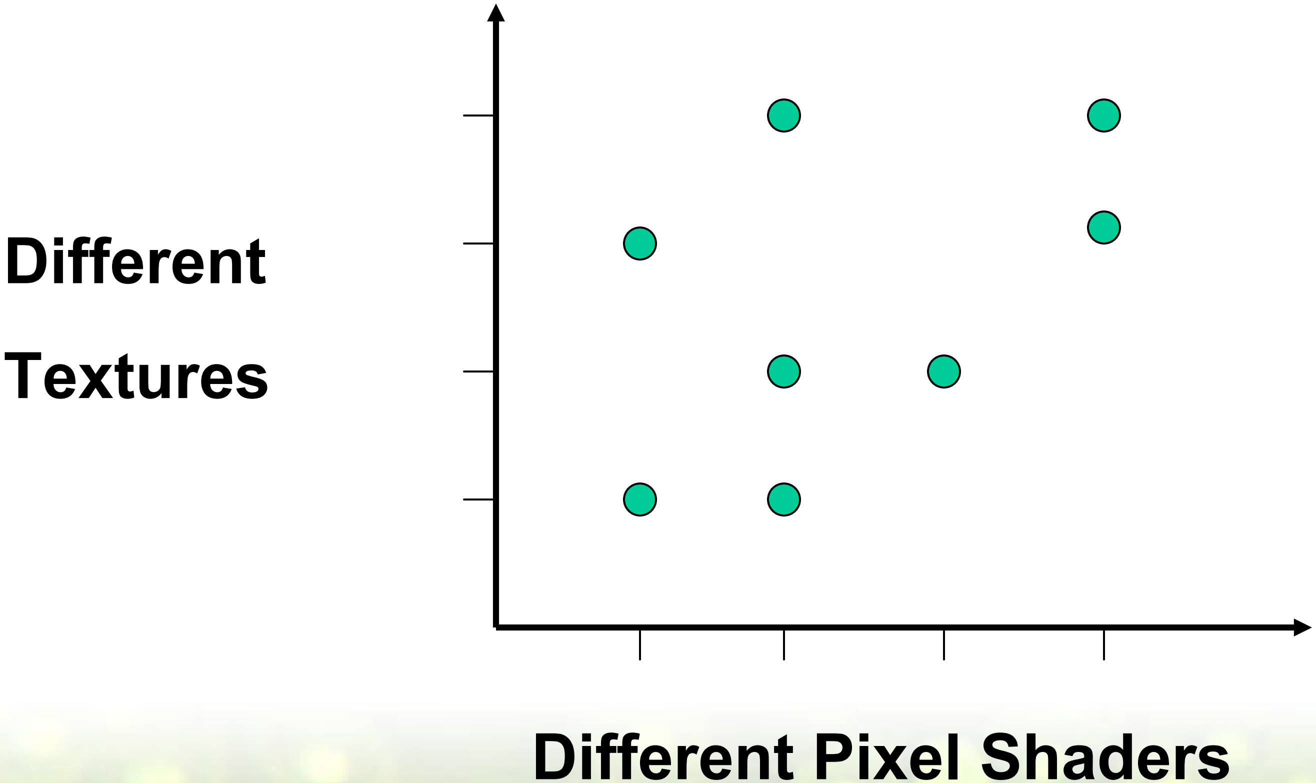
Do State Changes Really Matter?

- Cost of state changes
- Comparison with no state changes
- One state change:
  - Factor of 4 drop in fps (on average)
- Multiple state changes:
  - Another factor of 2-5 drop

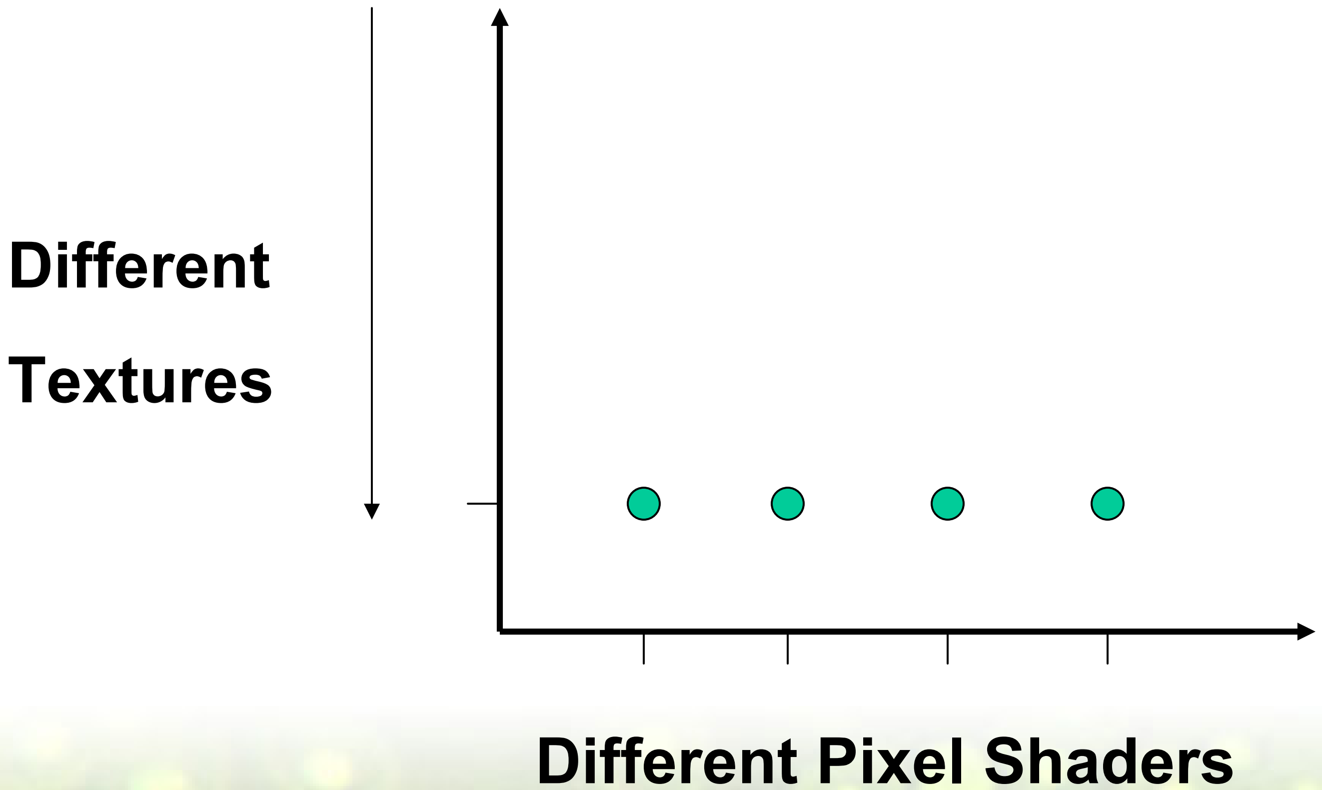
How To Sort?

- Seems like an n-dimensional problem
- Should I sort by texture, pixel shader, vertex shader, ... what?

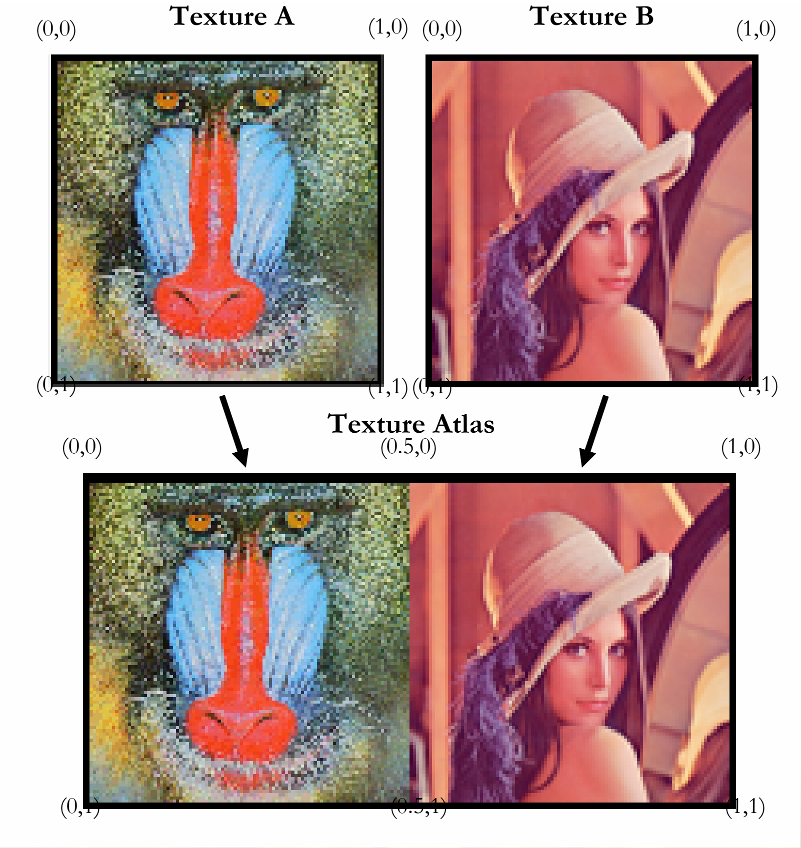
Texture v. Pixel Shader



Collapse One Of The Axes



Texture Atlases



Basic Idea

- Select batch-breaking textures
- Pack into one or more texture atlases
- Update the *uv*-coordinates of models

- Convert multiple DIP calls into one

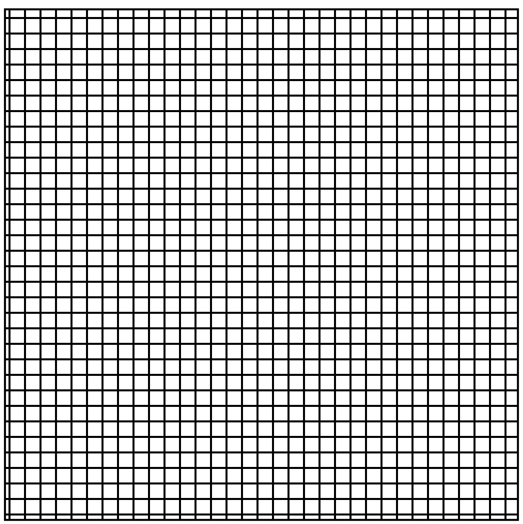
What About Mip-Maps?

- What happens to the lowest 1x1 level?
  - Smearing?
- Tool-chain should generate mip-maps before packing
- Use special purpose mip-map filters

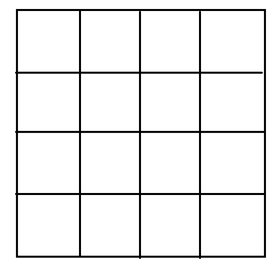
What About Lower Levels?

1 16x16 Sub-Texture

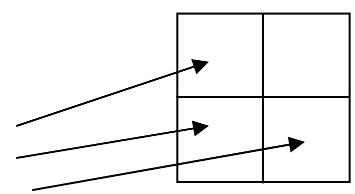
12 8x8 Sub-Textures



4x4 Level



2x2 Level

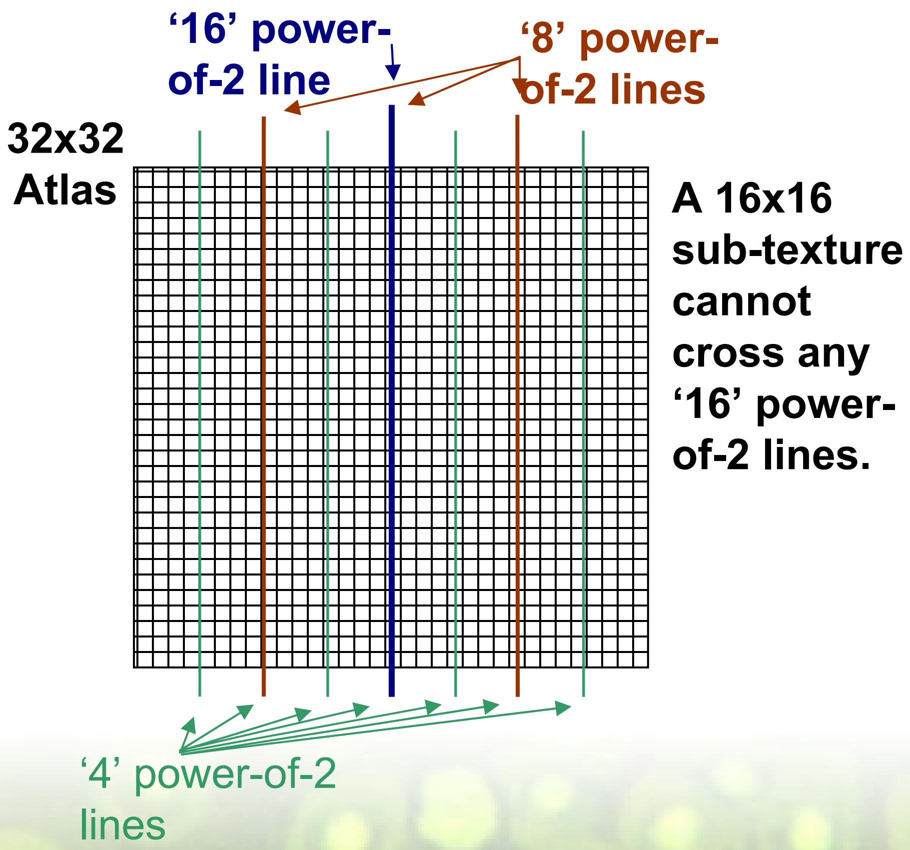


Smearing

Auto-Generation of Mip-Maps

- 2x2 Box filter can also work for power-of-2 textures
  - Both atlas and sub-textures in it are pow2
  - Textures should not cross pow2 lines

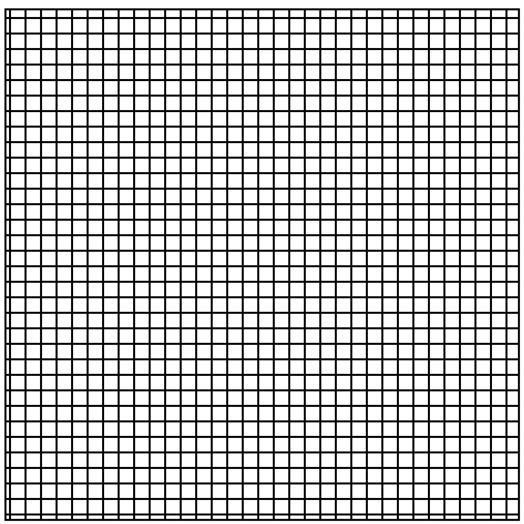
Proper Placement For Box Filter



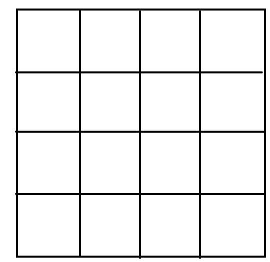
What About Lower Levels?

1 16x16 Sub-Texture

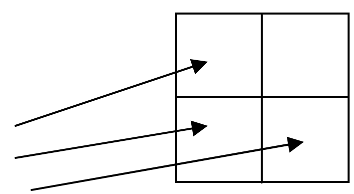
12 8x8 Sub-Textures



4x4 Level



2x2 Level



Smearing

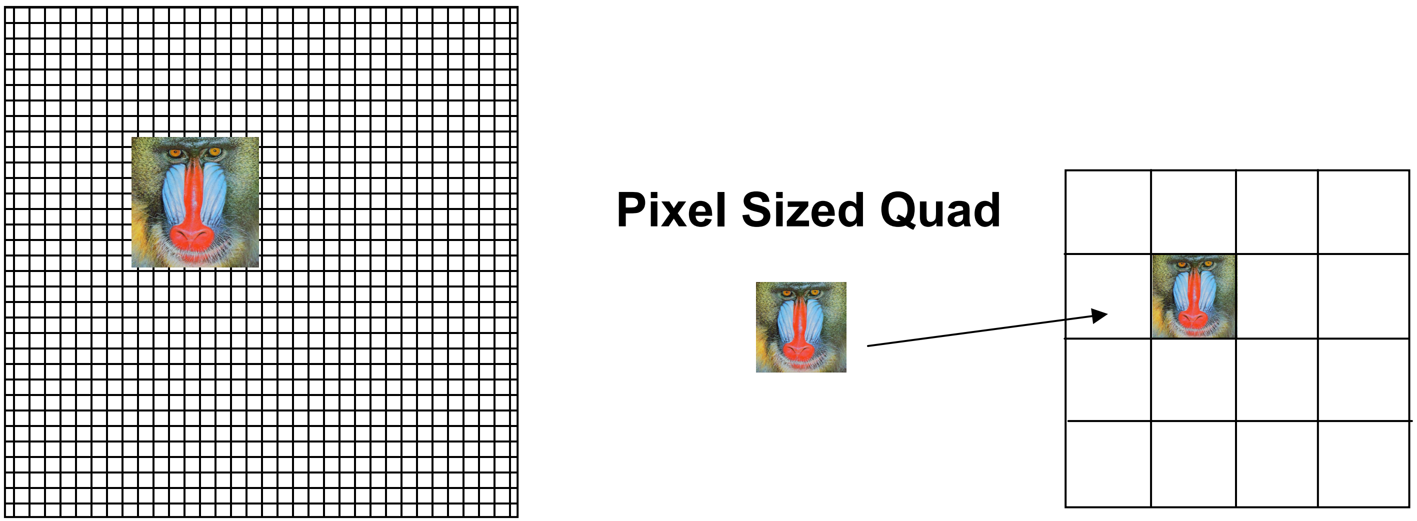
Possible Solutions

- Terminate mip chain to fit smallest sub-texture
  - Image Quality and Performance Issues
- Use only sub-textures of same size
  - May be inflexible
- But there's good news...

Cannot Access Lower Levels

- A triangle's texture coordinates never span across sub-textures
- Worst case: pixel-sized triangle spanning entire sub-texture
- Only "1-texel" level is accessed
  – Fill it with valid data

Cannot Access Lower Levels



- DirectX raster rules make it unlikely for smaller quad (or tri) to generate pixel

Other Issues

- Address modes such as clamp?
  - Use *ddx, ddy* in pixel-shader to emulate modes
- Smearing due to filtering
  - Texels on border of sub-textures get smeared
  - Aniso can help: smaller footprint
  - Do re-mapping of texcoords in pixel shaders
  - Pad textures with border texels

DirectX9 Instancing API

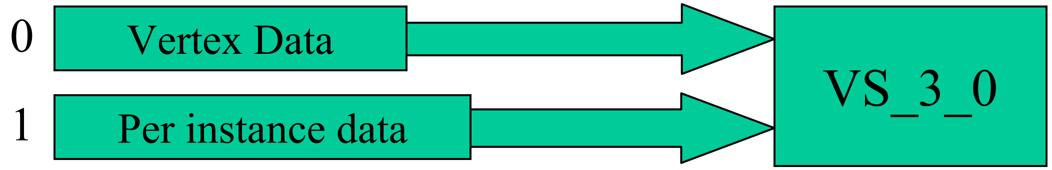
- What is it?
  - Single draw call to draw multiple instances of the same model
- Why should you care?
  - Avoid DIP calls and minimize batching overhead
- What do you need?
  - DirectX 9.0c
  - VS 3.0/PS 3.0 support

When To Use Instancing

- Many Instance of Same Model
  - Forest of trees, particle systems, sprites
- Encode per-instance data in auxiliary stream
  - Colors, texture coordinates, per-instance consts
- Not as useful if batching overhead is low
  - Fixed overhead to instancing

How Does It Work?

- Vertex stream frequency divider API

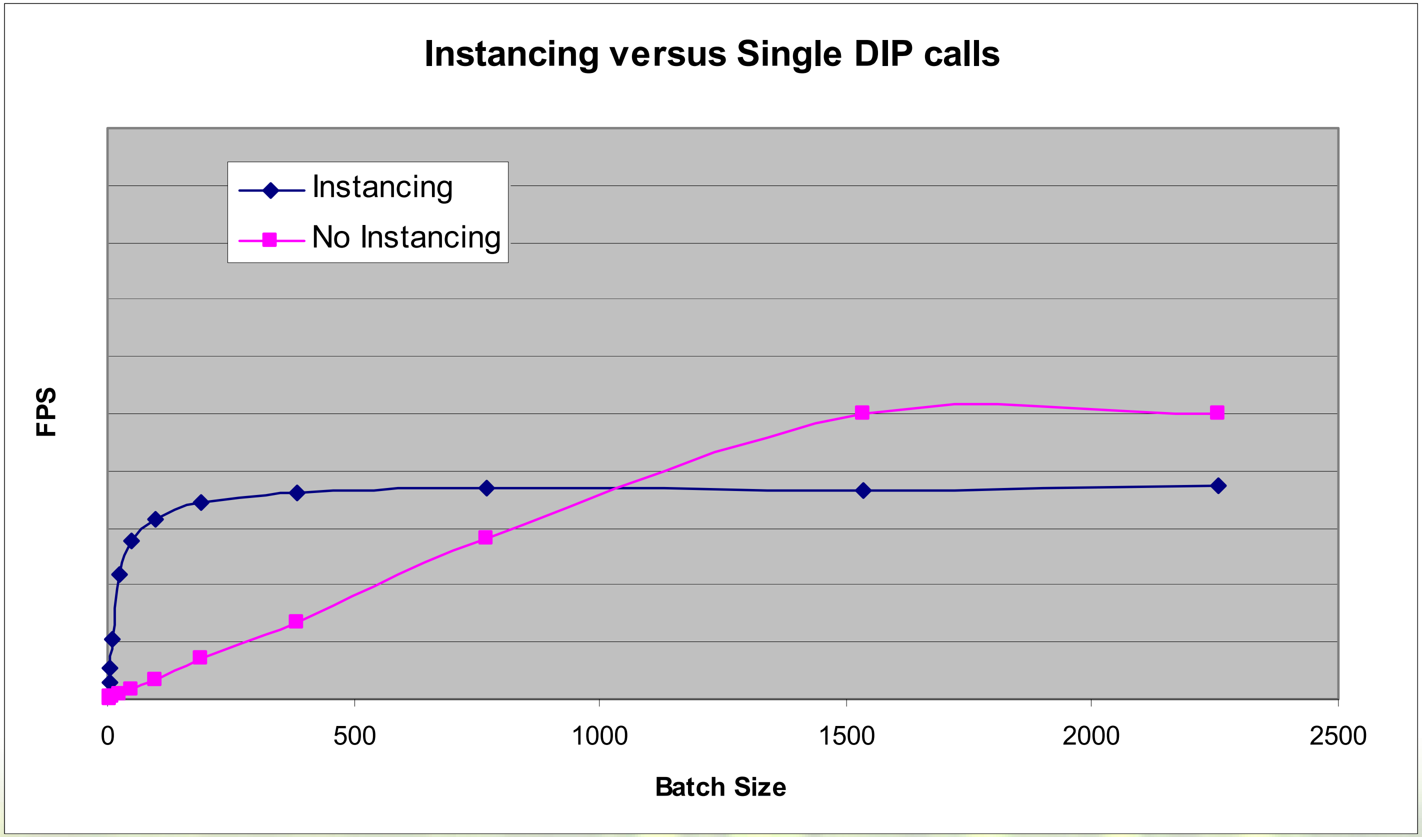


- Primary stream is a single copy of the model data
- Secondary stream: per instance data
  – pointer is advanced for each rendered instance

Simple Instancing Example

- 100 poly trees
  - Stream 0 contains just the one tree model
  - Stream 1 contains model WVP transforms
    - Possibly calculated per frame based on the instances in the view
  - Vertex Shader is the same as normal, except you use the matrix from the vertex stream instead of the matrix from VS constants
- If you are drawing 10k trees that's a lot of draw call savings!
  - You could manipulate the VB and pre-transform verts, but it's often tricky, and you are replicating a lot of data

Some Test Results



1 million diffuse shaded polys in each run

Test Summary

- Big win for small batch sizes
- Fixed overhead for instancing
- Cross-over point changes depending on CPU and GPU, engine overhead etc.

More Information

- White paper and tools soon for texture atlases on www.nvidia.com/developer
- “Profiling Your DirectX Application” in NVIDIA sponsored session on Wed.

Questions?

- Contact: arege@nvidia.com