INTRODUCTION AND TECHNIQUES
WITH NVIDIA® GVDB VOXELS

Dr. Rama Hoetzlein, Lead Architect for GVDB Voxels
MOTIVATION

Motion Pictures

Data Property of
DreamWorks Animation
MOTIVATION
Motion Pictures
MOTIVATION
3D Printing
MOTIVATION
Scientific Visualization

A Numerical Study of High-Pressure Oxygen/Methane Mixing and Combustion of a Shear Coaxial Injector, Nan Zong & Vigor Yang, AIAA 2005
MOTIVATION
Scientific Visualization
INTRODUCING
NVIDIA® GVDB VOXELS

A new, open source NVIDIA SDK for compute, simulation and rendering of sparse volumes based on CUDA
OUTLINE

1. Overview
2. GVDB Features
3. Workflow
4. Compute API: Spray Deposition
5. Rendering API: OptiX Integration
6. Dynamic API: Fluid Surface
What is GVDB Voxels?

- An SDK library for developers to create applications in Motion Pictures, 3D Printing and Scientific Sim/Vis

- Abstracts the complexity of sparseness to support compute and rendering of high resolution volumes

- Based on CUDA, for portability to embedded devices (JetsonTX), Quadro/GeForce, Tegra/GRID

- Open source, fBor developers to customize
NVIDIA® GVDB VOXELS

Overview

Dense Volume
NVIDIA® GVDB VOXELS

Overview

Dense Volume

- 8 empty steps
- 5 sample steps
Overview

- 8 empty steps
- 5 sample steps
NVIDIA® GVDB VOXELS

Overview

- Sparse Volume

- 2 DDA skip steps
- 5 sample steps
NVIDIA® GVDB VOXELS

Overview

- Sparse Volume
- 2 DDA skip steps
- 5 sample steps
NVIDIA® GVDB VOXELS

Overview

Topology

Data Atlas
Ken Museth, *Dreamworks Animation*
Transactions on Graphics, 2013
NVIDIA® GVDB VOXELS

Overview

Topology

~10 MB

CPU

GPU

Voxel Data

1 2 4 3 6 7 5 9
4 3 3 2 6 7 6 5
3 2 2 1 7 8 6 6
3 2 1 1 8 9 7 6
5 5 6 7 1 1 3 2
5 6 6 8 0 1 2 3
6 7 6 7 1 2 3 3
7 8 7 6 1 2 4 3

>1 GB
NVIDIA® GVDB VOXELS

Overview

- Memory pooling for fast topology
- 3D Textures for hardware trilinear filtering
- Indexing instead of pointers for CPU-GPU transfers

Hoetzlein, GVDB: Raytracing Sparse Voxel Database Structures on the GPU
High Performance Graphics, 2017
NVIDIA® GVDB VOXELS

Features for Developers

Compute API
Author compute kernels more easily with virtual neighbors.

Render API
Built-in and custom rendering for CUDA and OptiX Raytracing

Dynamic API
Modify topology to support dynamic simulations

See Programming Guide for additional APIs and features
Computes API

Volume Smoothing

Launch GPU kernels over all sparse voxels.

Efficient compute on very large domains.
extern "C" __global__ void kernelSmooth ( int res, float amt )
{
  GVDB_SHARED_COPY

  float v = 6.0*svox[ndx.x][ndx.y][ndx.z];
  v += svox[ndx.x-1][ndx.y][ndx.z];
  v += svox[ndx.x+1][ndx.y][ndx.z];
  v += svox[ndx.x][ndx.y-1][ndx.z];
  v += svox[ndx.x][ndx.y+1][ndx.z];
  v += svox[ndx.x][ndx.y][ndx.z-1];
  v += svox[ndx.x][ndx.y][ndx.z+1];
  v /= 12.0;

  surf3Dwrite ( v, volTexOut, vox.x*sizeof(float), vox.y, vox.z );
}

← Macro ensures neighbors are available in shared memory
← Smoothing operation (values from neighbors)
← Output value
COMPUTE API

Simulation
COMPUTE API

Simulation

Generate Rays from Source

Cast Rays into Scene

Smooth to simulate viscosity

Accumulate material at Hit points

repeat
**COMPUTE API**

**Simulation**

- Generate Rays from Source
- Cast Rays into Scene
- Smooth to simulate viscosity
- Accumulate material at Hit points

**GVDB VOXELS**

- Model Part
- Sparse Raytracing
- Voxel Smoothing
- Add Density

**Application**

- Deposited Material
- Occlusion
- Overhang
- Part
- Particle Source
**COMPUTE API**

**Simulation**

```c
void App::simulate ()
{
    numrays = 1000;
    gvdb.AllocData ( rays, numrays, sizeof(ScnRay) );

    ScnRay* ray = (ScnRay*) gvdb.getDataPtr ( 0, rays );
    for (int n=0; n < numrays; n++ ) {
        GenerateRay ( ray );
        ray++;
    }
    gvdb.CommitData ( rays );

    gvdb.Raytrace ( rays, SHADE_TRILINEAR, 0, -0.0001 );
    gvdb.SetPoints ( pntpos, pntclr );
    gvdb.InsertPoints ( rays, Vector3DF(0,0,0) );
    gvdb.ScatterPointDensity ( rays, 2.0, 1.0, Vector3DF(0,0,0) );
    gvdb.UpdateApron ();
    gvdb.Compute ( FUNC_SMOOTH, 0, 1, Vector3DF(4,0,0), true );
}
```

- Allocate ray memory (CPU and GPU)
- Generate rays
- Commit rays to GPU
- Raytracing to hit part
- Scatter new material at hit points
- Smooth result slightly
RENDERING

InteractiveGL Sample

InteractiveOptiX Sample
// Render with CUDA
gvdb.Render ( 0, SHADE_TRILINEAR, .. )

// Read result into OpenGL texture
gvdb.ReadRenderTexGL ( 0, gl_texture );

// Display texture on screen
renderScreenQuadGL ( gl_texture );

3DPrint Sample
// Set render parameters
SetExtinct ( -1.0f, 1.5f, 0.0f );
SetVolumeRange ( 0.1f, 0.0f, .1f );
SetCutoff ( 0.005f, 0.01f, 0.0f );
SetBackgroundClr ( 0.1, 0.2, 0.4, 1.0 );

// Create transfer function
LinearTransferFunc ( 0.00f, 0.25f, ... );
LinearTransferFunc ( 0.25f, 0.50f, ... );
LinearTransferFunc ( 0.50f, 0.75f, ... );
LinearTransferFunc ( 0.75f, 1.00f, ... );

// Render with CUDA
gvdb.Render ( 0, SHADE_VOLUME, .. )
OptiX: A General Purpose Ray Tracing Engine

OptiX: A General Purpose Ray Tracing Engine, Steven Parker. ACM Transactions on Graphics, Vol. 29, No. 4, July 2010
OptiX SDK

- Available for free: Windows, Linux, Mac
GVDB and OptiX
Voxel-to-Polygon Scattering

Polygon Self-Shadowing

Polygon-to-Polygon Scattering

Volume Deep Rendering

InteractiveOptiX Sample
GVDB and OptiX
Thursday, 2:00-2:25pm, S7425

3D PRINTING WITH NVIDIA
GVDB VOXELS
**Features**

- Efficient raytracing of large, sparse volumes
- CUDA based raytracing
- Native rendering of semi-transparent volumes, isosurfaces and level sets
- Integration with NVIDIA® OptiX
- Custom transfer functions and custom render kernels
- Rendering API in GVDB Library
VIDEO
PLUGINS
GVDB Performance

- **Rendering (30 frames)**
  - Maxwell: 31 sec
  - Pascal GP100: 45 msec/frame
  - +42% faster

- **Voronoi Generation**
  - Maxwell: 2.5 sec
  - Pascal GP100: 1.5 sec
  - +65% faster

- **Voxelization**
  - Maxwell: 6.3 sec
  - Pascal GP100: 3.1 sec
  - +100% faster

*Lower is better*
NVIDIA® GVDB VOXELS
Platforms

Hardware: Jetson TX1 / TX2 embedded systems 256 cores (Tegra)
          Quadro, GeForce 3840 cores (GP100)
          Tesla K80 computing cluster 4992 cores (K80)

Software: CUDA Compute 3.0 and higher
          CUDA Toolkit 6.5 and higher

Operating Systems: Windows 7/8/10 64-bit and Linux

Licensing: Open Source, BSD 3-clause
VOX3: Experimental DIY DLP/SLA 3D Printer

Thursday, 2:00-2:25pm, S7425

3D PRINTING WITH NVIDIA GVDB VOXELS
NVIDIA® GVDB VOXELS
Software

http://developer.nvidia.com/gvdb

Download from website

Windows Installer or Github

Comes with 8x samples

Detailed Programming Guide on webpage

New Forum on devtalk.nvidia.com
NVIDIA® GVDB VOXELS

Rama Hoetzlein Lead Architect for GVDB Voxels rhoetzlein@nvidia.com

Thanks to..

Andrew Page Product Manager
Prerna Dogra Product Manager, 3D Printing
Vincent Brisebois Marketing
Tristan Lorach Professional Graphics
THANK YOU