INTRODUCING GVDB SPARSE VOLUMES

Rama Hoetzlein, NVIDIA Corporation
Ken Museth, Dreamworks Animation & SpaceX

July 27th 2016
SIMULATION IN MOTION PICTURES

*Increasing detail and complexity.*

Property of DreamWorks Animation
THE SOLUTION: VOXELS

Simulations are easier to perform on voxels
OPENVDB: SPARSE VOLUMES
Ken Museth, Lead Developer of OpenVDB

Ken Museth, VDB: High-resolution sparse volumes with dynamic topology, Transactions on Graphics, 2013
INTRODUCING
NVIDIA® GVDB SPARSE VOLUMES

Volume rendering 5GB surface in GVDB in 5 seconds

Data Property of DreamWorks Animation
INTRODUCING

NVIDIA® GVDB SPARSE VOLUMES

Volume rendering 5GB surface in GVDB in 5 seconds

Data Property of DreamWorks Animation
INTRODUCING
NVIDIA® GVDB SPARSE VOLUMES

Each Blue Brick is $8^3 = 512$ voxels. Total Size: $3344 \times 568 \times 3384 = 5.5$ GB (24 GB dense)
What is GVDB?

NVIDIA® GVDB is a GPU-based framework for VBD data structures inspired by the award-winning software library OpenVDB used for motion picture visual effects and modelling, with tools to enable full volume compute operations and high quality raytracing.
High Quality Raytracing

NVIDIA® GVDB integrates with NVIDIA® OptiX to deliver efficient, generalized raytracing of sparse volumes with global illumination.

GVDB Raytracing on GPU is 10x-30x faster than CPU rendering
NVIDIA® GVDB with NVIDIA® OptiX integration enables interactive editing of materials and lighting of volumes.
NVIDIA® GVDB direct raytracing of level set surfaces and volumetric data with CUDA kernels.
NVIDIA® GVDB Sparse Volumes

Compatibility with OpenVDB

Features:

• Spatial layout and numerical values identical to OpenVDB
• Uses fast VBX cache format internally, yet able to translate to and from OpenVDB files
• Run-time configuration of VDB topology
Sparse volume compute operations are supported with CUDA using a single kernel launch over all bricks.

User-created kernels can easily access neighbors.
**NVIDIA® GVDB Sparse Volumes**

*API Library Usage*

**Example Host code:**

```c
// Optional
gvdb.SetCudaDevice ( devid );

gvdb.Initialize ();           // Start GVDB

gvdb.LoadVBX ( scnpath );    // Load volume

// Screen pixels
gvdb.AddRenderBuf ( 0, w, h, 4 );

cuModuleGetFunction ( &cuRaycastKernel,
        cuCustom, "my_raycast_kernel" )

// Screen pixels

gvdb.AddRenderBuf ( 0, w, h, 4 );

unsigned char* buf = malloc ( w*h*4 );

gvdb.ReadRenderBuf ( 0, buf );

save_png ( "out.png", buf, w, h, 4 );
```

**Example Kernel code:**

```c
#include "cuda_gvdb.cuh"

__global__ void raycast_kernel ( uchar4* outBuf )
{
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    if ( x >= scn.width || y >= scn.height ) return;

    rayMarch ( gvdb.top_lev, 0, scn.campos,
            rdir, hit, norm );        // Trace ray into GVDB

    if ( hit.x != NOHIT ) {
        float3 R= normalize ( reflect3 ( eyedir , norm ) );
        float clr = tex3D ( envmap , R.xy );
    } else {
        clr = make_float3 ( 0.0, 0.0, 0.1 );
    }

    outBuf [ y*scn.width + x ] = make uchar4(
            clr.x*255, clr.y*255, clr.z*255, 255 );
}
```

*NVIDIA® GVDB is focused on motion picture developers.*
NVIDIA® GVDB SPARSE VOLUMES

Upcoming Release

API Library with multiple samples

Based on CUDA

Integration with OpenVDB and NVIDIA® OPTIX

Open Source with BSD 3-clause License

Available in late September 2016
“GVDB is a new rendering engine for VDB data, uniquely suited for NVIDIA GPUs and perfectly complements the CPU-based OpenVDB standard while improving on performance. I am excited to take part in the future adoption of GVDB in the open-source community for visual FX.”

— Dr. Ken Museth, Lead Developer of OpenVDB (DreamWorks Animation & SpaceX)
NVIDIA® GVDB SPARSE VOLUMES

Application Areas

Scientific Visualization

Motion Pictures

3D Printing

See GTC 2016 talk: Raytracing Scientific Data in NVIDIA OptiX with GVDB Sparse Volumes