OpenVDB Course:
Advanced Applications of OpenVDB in Production

double negative visual effects
dan bailey  (dan@dneg.com)
OpenVDB Uses

- Level Set
- Fog Volume
- Points (New)
- Vector Field
- Alpha Mask

double negative visual effects
Attributes

- OpenVDB extended with new Attribute API
- AttributeArray.h and AttributeSet.h
Points located within a Leaf Node are owned by that Leaf along with all of their attributes.
VDB Points Data Structure

- Root node
  - (unbounded)
- Internal Node 1
- Internal Node 2
- Point Data
- Leaf Node
- Attribute Set

Tile values with active/inactive states
- Active Mask
- Child Mask
- Tile values / Child pointers
- Active Mask
- Voxels
- Attribute Array

(New)
Point Data Class Structure

- Tree
  - Root Node
  - Accessor Registry
  - Internal Node
  - Background
  - Tiles
  - Point Data Leaf Node
  - Bit Mask
  - Tiles
  - Attribute Set
  - Voxels
  - Bit Mask
  - Attribute Descriptor
  - Attribute Arrays
  - Meta Map
  - Transform
    - Linear Maps
    - Frustum
Leaf Nodes can store different attributes (from each other)

However, not typically supported by tools
<table>
<thead>
<tr>
<th>Motivation</th>
<th>Convenience</th>
<th>Performance</th>
<th>Memory</th>
<th>I/O</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial Organisation</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Greater Compression</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Data Locality</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>VDB Topology and Tools</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
In-Memory Compression

Attribute Compression
(x, y, z) => (w)
3 x 32-bits => 16-bits
(Not Available for Native Houdini Points)

Uniform Value Compression
[1, 1, 1, 1, 1, 1, ...] => [1]
(Available for Native Houdini Points)

Stream Compression
(Available for Native Houdini Points in Houdini 14 but only for disk compression)
Position Storage

((-0.1, -0.3, 0.2))
Floating-point

Exponent | Mantissa
---|---
10110111 | 0101011101010101010101010

8 bits | 24 bits

1.\{mantissa\} x 2\{exponent\}

8.4M → 127

126 ← 8.4M 8.4M 8.4M 8.4M
Quantisation

Encoding

32-bit floating-point → (int) round(x * $2^{16f}$ + 0.5f) → 16-bit integer

Decoding

16-bit integer → float(x) / $2^{16f}$ → 32-bit floating-point
Position Compression

- **Float Array**
  - Memory: 1.92GB

- **No Quantisation**
  - Memory: 1.95GB

- **16-bit Quantisation**
  - Memory: 1.02GB

- **8-bit Quantisation**
  - Memory: 568MB

<table>
<thead>
<tr>
<th>Compression Type</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float Array</td>
<td>1.86</td>
</tr>
<tr>
<td>No Quantisation</td>
<td>1.71</td>
</tr>
<tr>
<td>16-bit Quantisation</td>
<td>1.60</td>
</tr>
<tr>
<td>8-bit Quantisation</td>
<td></td>
</tr>
</tbody>
</table>
Gather
Gather Rasterisation Performance

Point Index Grid / Point Partitioner
- Time: 26.2s
- Performance: 2.23x

Point Data Grid
- Time: 11.7s

256 Million Points
64 Million Voxels
In-House Dynamo Liquid Solver

Dynamo Nodes
Dynamo Data Model
Dynamo Distribution
OpenVDB
OpenVDB Points

Large Data Sets All Stored using OpenVDB
Houdini Integration

SOP “Micro-Solvers”

Pressure Visualisation
Dynamo FLIP Liquid Simulation
1 Billion Points +

double negative visual effects
Distribution Scaling

Near-Linear Scaling using OpenVDB

Nodes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Scaling factors: 3.7x, 5.5x
Point Count: 1 billion
Peak Memory: 60GB
Performance: 10-15 mins/frame
Nodes: 1 machine

Render Time: 1 hour/frame
Memory: 11.0 GB
Clarisse Integration

Intersection Testing

Ray Bundle

Hit Points + Derivatives
Intersections: 9 voxels
Tracing Rays - DDA for Points

Intersections: 19 voxels

More Intersections due to Point Radius!
Tracing Rays - Motion Blur

Intersections:
40 voxels

Even More Intersections to Introduce Motion Blur!
BVH Structure

Smallest Unit:
2 x 2 x 2 Voxels

Position at T - 0.5
Position at T
Position at T + 0.5
BVH Structure

Bounding Box at T + 0.5

Interpolate In-Between

Bounding Box at T - 0.5
BVH Structure

Root node (unbounded)

Internal Node 1

Internal Node 2

Leaf Node

Sub Leaf Node

Clarisse BVH Tree
Primitive = Leaf

Custom BVH Tree
Primitive = 2x2x2 Voxels
Interactive Speed Raytracing 1 Billion Points in Clarisse!
Memory Footprint:
- 10.4GB (VDB Points)
- 20% (VDB Grid)
- 6.4MB (BVH Tree)

Position (16-bit)
Velocity (32-bit + 16-bit)

720p with 6 anti-alias samples
Renders in 200s
Optimum Voxel Size

Voxel Size: 0.25
Leaves: 583
Memory: 956 MB

Voxel Size: 0.05
Leaves: 17,200
Memory: 999 MB

Voxel Size: 0.01
Leaves: 789,000
Memory: 2,829 MB

Slow Performance
Low Memory, Fast Performance
High Memory
Open-Source

OpenVDB API

- AttributeArray
- AttributeSet
- PointDataLeaf
- PointConversion
- Serialisation

openvdb::points::initialize()

OpenVDB Houdini

- OpenVDB Points SOP
- Viewport Visualisation

Houdini 13+

Due to be announced on OpenVDB mailing list very soon!
Email me for more information (dan@dneg.com)