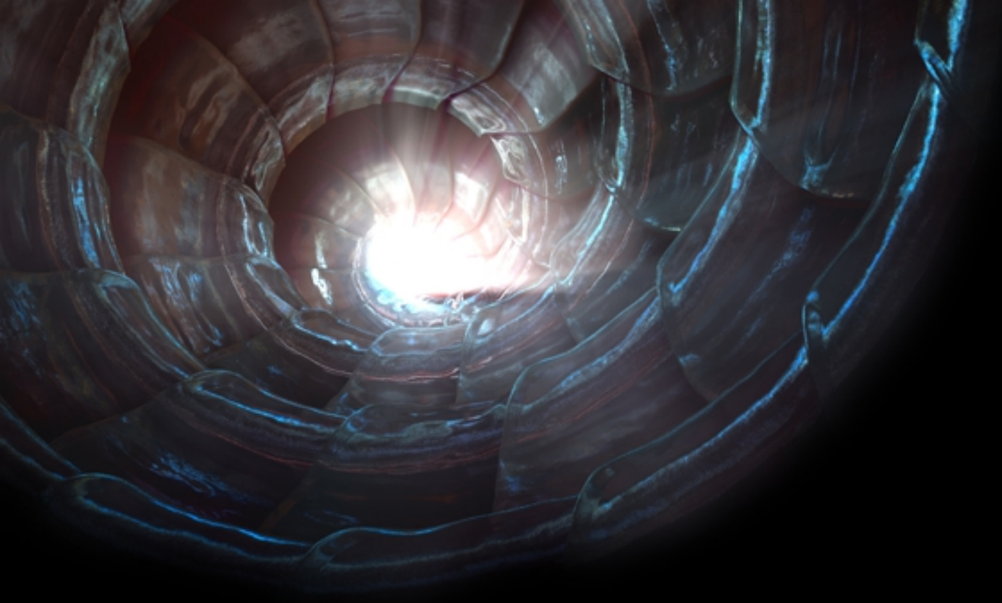


SIDE EFFECTS SOFTWARE

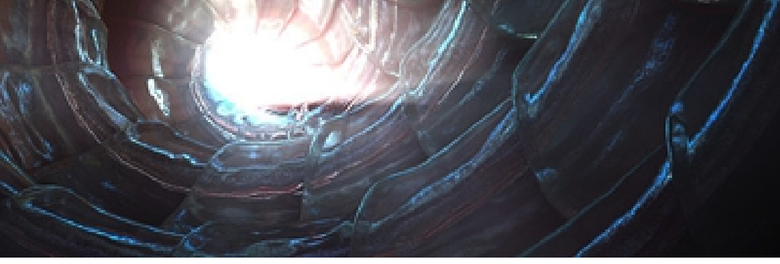
GO | PROCEDURAL

OpenVDB Adoption in Houdini

Edward Lam
Senior Software Architect

SIGGRAPH 2013

Houdini 3D ANIMATION TOOLS

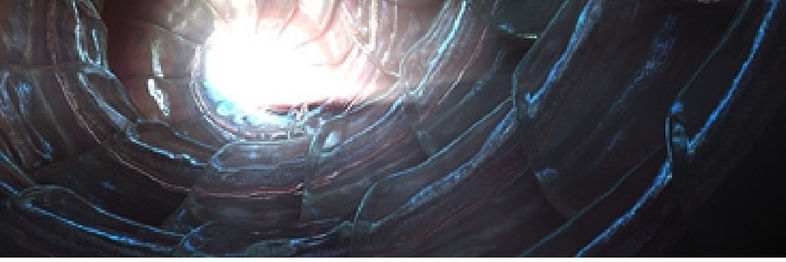


OpenVDB in Houdini

- How we added OpenVDB to Houdini
- Approach
- Integration challenges
- Coding tricks
- Volume display

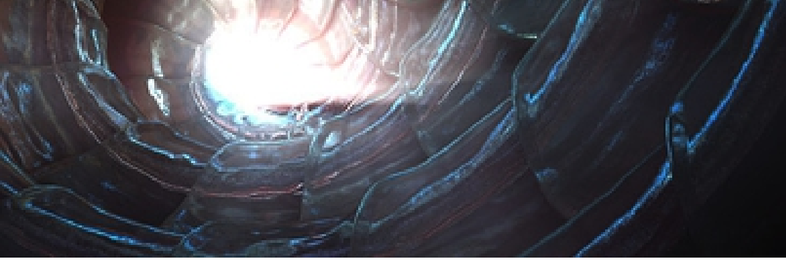
- Houdini OpenVDB Team
  - Jeff Lait, Edward Lam, Mark Alexander, Halfdan Ingvarsson, Neil Dickson, Adrian Saldanha, Adam Jeziak

SIDE EFFECTS SOFTWARE



Integrating with Existing Volumes

- Replace: Too much existing code
- Just-in-time conversion (our approach for OpenCL grids)
  - Unexpected failures due to large VDBs
  - Lossy conversion: active regions, extend conditions, etc
- Two volume types
  - Forces explicit conversions
  - Not always clear what operations need what type
- Nodes/algorithms to work with both types where appropriate
- Windows and Mac OS X port

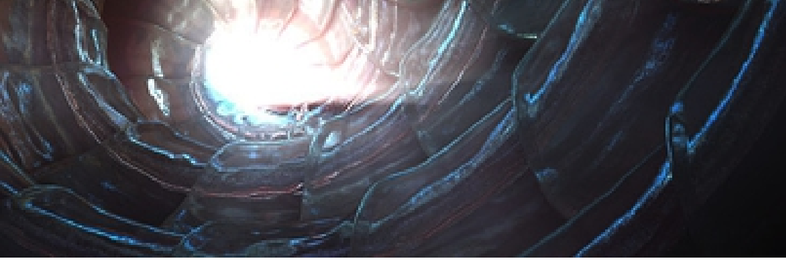


Old Volumes

- Terminology
  - Volume: the old Houdini volume type – ”Dense Volumes”
  - VDB: the VDB type – ”Sparse Volumes”
- What were they?
  - 16^3 voxel tiles, constant tile optimization
  - Similar to Field3D default configuration

Demo: BigBrain

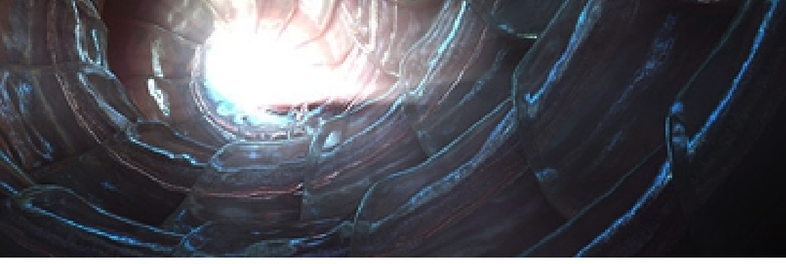
- Data courtesy of the CBRAIN project
  - https://bigbrain.loris.ca/main.php
  - http://cbrain.mcgill.ca/



Active Regions

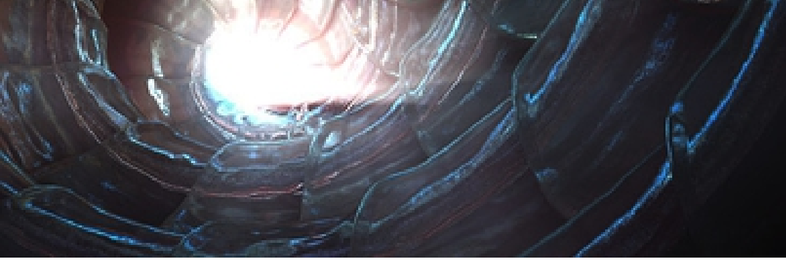
- Key difference: “active” voxels instead of “bounds”
- Dense volumes: VDB with cube of active voxels
- Operations that apply to ”all” voxels mean apply to all **active** voxels
- Need tools to adjust the active region
  - Crop/extend for Volumes
  - Union, dilate, erode, intersect for VDBs

- Demo: Conway's Game of Life in 3D



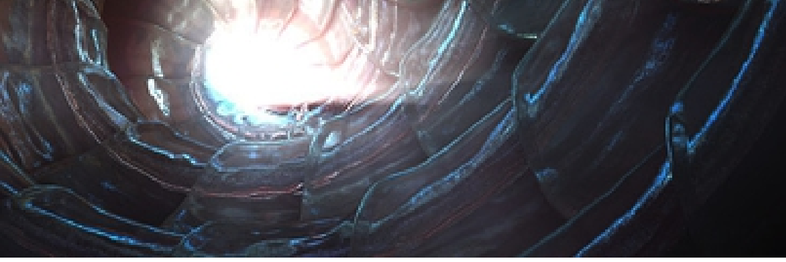
Transform Hell

- Old Volumes: Normalized $[0..1]^3$ coords over bounding box
- VDB: Poorly defined since active region is dynamic
  - No native VDB equivalent
- Can sometimes use the bbox of the active region
- Samples at center of the voxel or corner?
  - Volumes define center-sampled grids
  - VDBs define corner-sampled
- Volume samples should be the same regardless of type



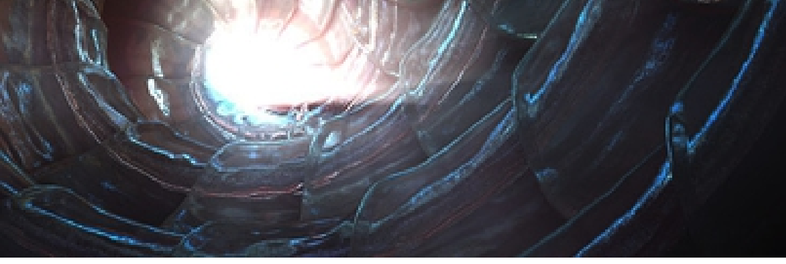
Baggage from Volumes

- Some concepts useful from Volumes
- Resolution: Extents of active voxels
  - Note: Negative voxel indices in VDBs
- Bounds: Extents of active voxels in object space
- Normalized Space: $[0..1]^3$ or $[-1..1]^3$ mapping of bounds to object space



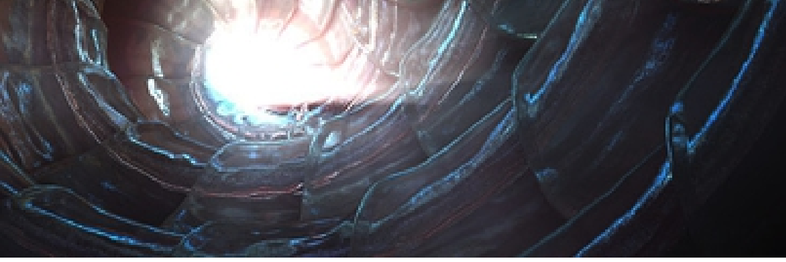
Frustum Frustration

- Same: Linear taper
  - Constant Z-steps, XY-plane shrunk along Z axis
- VDBs define only a single taper
  - Volumes have independent X & Y tapers
- VDBs define the taper at the near Z plane
  - Volumes define it at the far Z plane



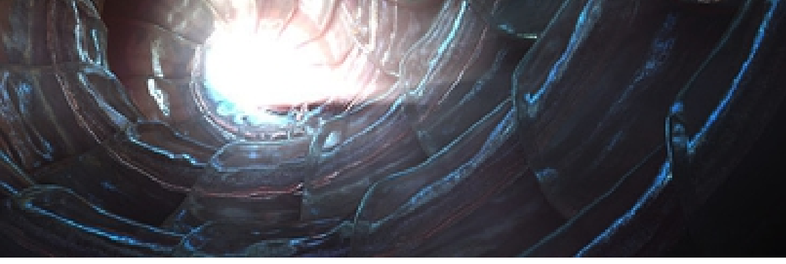
Frustum Clipping

- Normal VDBs define an infinite extent
  - Very useful and freeing for the artist!
- Taper creates a singularity
  - Nightmare of ”eyesplits” from RenderMan all over again!
  - Easy for an artist to blow up a scene by moving geometry too close to a camera
- Treat Frustum VDBs as having a finite extent
  - Writing operations clip to the defined frustum size, thus clamping at near/far planes
  - Still superior to Volumes as these can be very large



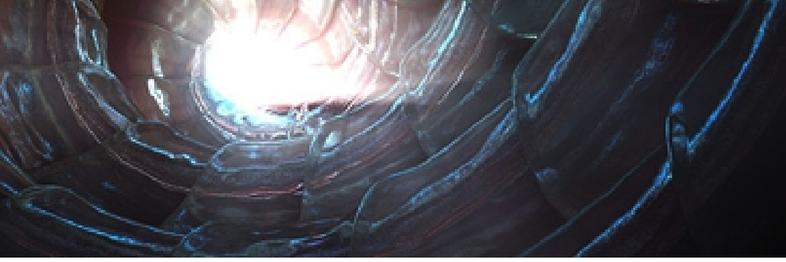
One Transform to Rule Them All

- GEO_PrimVolumeXform class
- Originally created just to factor transform out of the Volume primitive for speed
- Generalized to provide:
  - Volume to Object
  - Index to Object
- Allows sampling code to be written independent of Volume or VDB



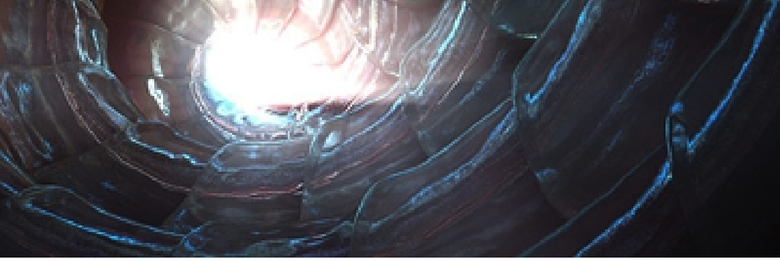
Algorithms

- Dense values → VDB ”leaf nodes”
- Constant nodes → VDB ”tiles”
- Goal: Operate over both volume and node types efficiently
- Tree visitor
- Grid per thread (thread-local)
- GridType::merge()
  - Fast since it steals data
  - Works because we ensure non-overlapping nodes
- Prune



Invalid SDFs

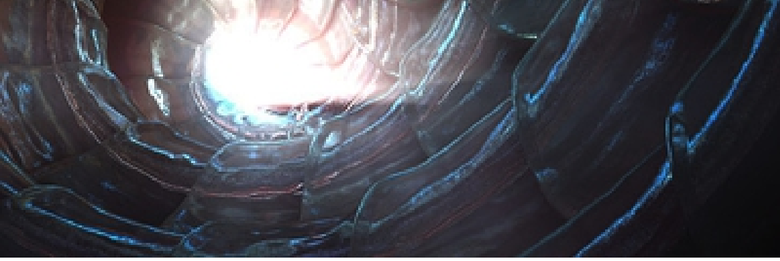
- A lot of VDB operations expect ”true” SDFs
- A lot of artists will produce ”incorrect” SDFs
- Do not assume narrow band principle is obeyed!
  - Provide mechanism to rebuild when necessary
  - Convert to Poly, Convert to SDF surprisingly effective



VDB Types

- `grid.isType<openvdb::FloatGrid>()` is a string compare
- Houdini only supports fixed set of grid types
  - `UT_VDBType UTvdbGetGridType(const GridBase &grid)`

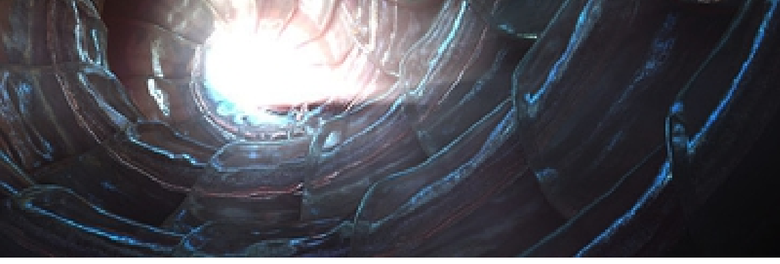
    `UT_VDB_FLOAT, UT_VDB_DOUBLE, UT_VDB_INT32, UT_VDB_INT64, etc`
- Cache type outside of loops



VDB Invocation

- Everything needs to be templated

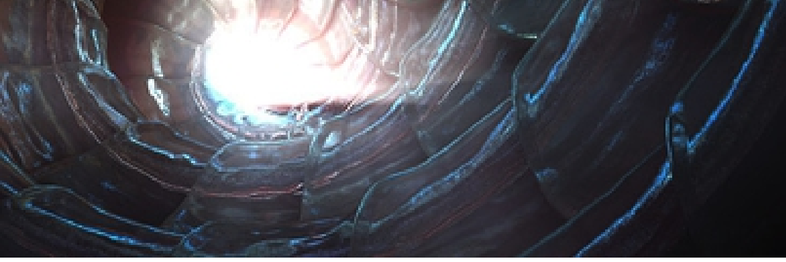
```
#define UT_VDB_CALL(GRIDT, RETURN, FNAME, GRIDBASE, ...) \
    { RETURN FNAME <GRIDT> (UTvdbGridCast<GRIDT>(GRIDBASE),__VA_ARGS__); }
        // NOTE: Visual C++ requires at least one argument in variadic
#define UTvdbCallAllType(TYPE, FNAME, GRIDBASE, …) \
          if (TYPE == UT_VDB_FLOAT) \
            UT_VDB_CALL(openvdb::FloatGrid,(void),FNAME,GRIDBASE,__VA_ARGS__) \
          else if (TYPE == UT_VDB_DOUBLE) \
            UT_VDB_CALL(openvdb::DoubleGrid,(void),FNAME,GRIDBASE,__VA_ARGS__) \
          … etc …

template <typename GridType>
static void operation(const GridType &grid, double param);

void doStuff(const GEO_PrimVDB &vdb, double param) {
    UTvdbCallAllType(vdb.getStorageType(), operation, vdb.getGrid(), param);
}
```



Math Types

- Your favorite vector library may not be openvdb::math

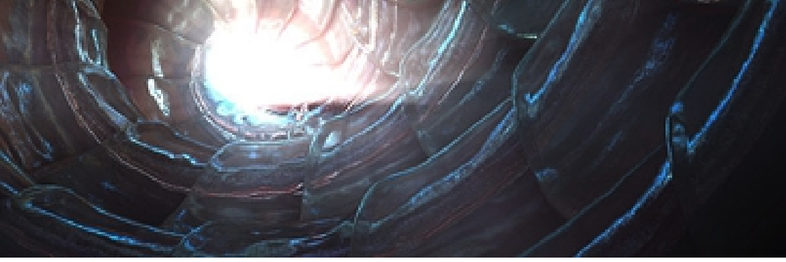
```
template <typename S>
UT_Matrix3T<S>
UTvdbConvert(const openvdb::math::Mat3<S> &src);
```



Accessors

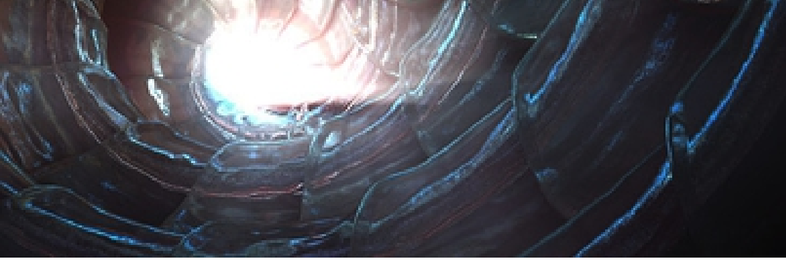
- Accessors are very important
  - One accessor for each direction reduces thrashing quite a bit

```
FloatGrid::ConstAccessor positive_acc[3] =
    { myGrid.getConstAccessor() // X
    , myGrid.getConstAccessor() // Y
    , myGrid.getConstAccessor() // Z
    };
FloatGrid::ConstAccessor negative_acc[3] =
    { myGrid.getConstAccessor() // X
    , myGrid.getConstAccessor() // Y
    , myGrid.getConstAccessor() // Z
    };
```



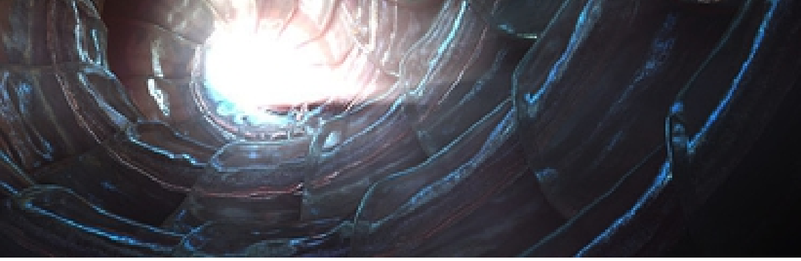
SDF Display

- For all active voxels
  - Check for neighbour crossing threshold
    - Generate point splat at neighbour crossing point
    - Set normal to gradient at the voxel
- Because of perspective distortion, care must be taken with the lighting calculation
  - Two sided lighting avoids black/white rims on a torus

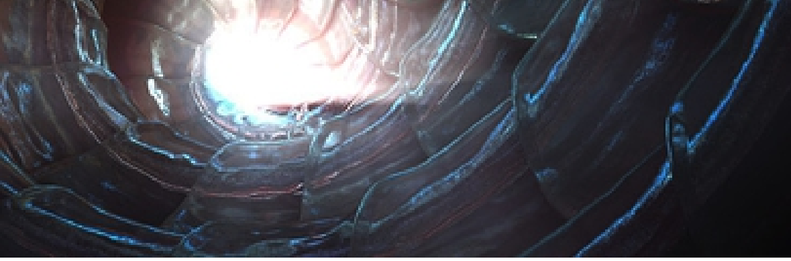


Wireframe Display

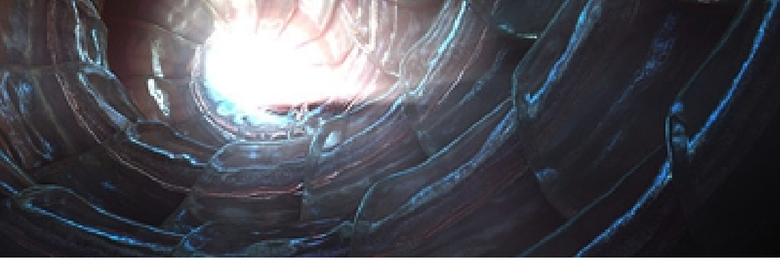
- Outline active nodes?
  - Becomes very noisy
- Outline bounding box of active nodes?
  - Looks like Volumes
- Houdini uses ”contour” of the active nodes



Active nodes vs Contours 1

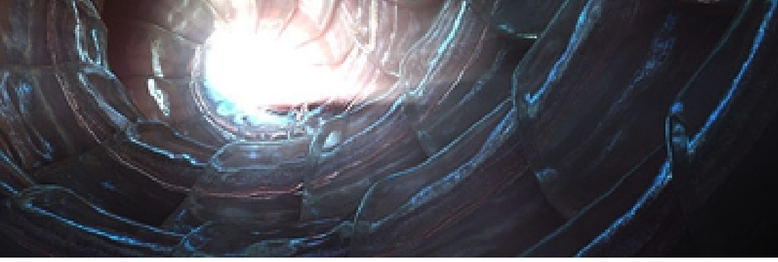


Active nodes vs Contours 2



Smoke Display

- Ideally, render tiles independently
  - But proper edge interpolation?
    - (Have to actually send 10^3 tiles to GL so it can render to the edges properly)
  - But shadows?
- Simpler: Downsample
  - Matches Volume behaviour for large volumes anyways!
  - Detached puffs of smoke decrease in res as they separate
- Demo: CloudFX



Acknowledgements

- Thanks also to Brett Miller and the OpenVDB team!