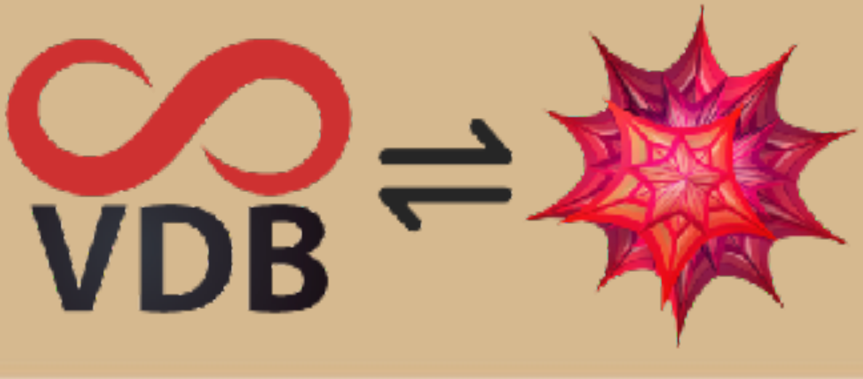
THE PREMIER CONFERENCE & EXHIBITION ON COMPUTER
GRAPHICS & INTERACTIVE TECHNIQUES

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

OPENVDBLINK

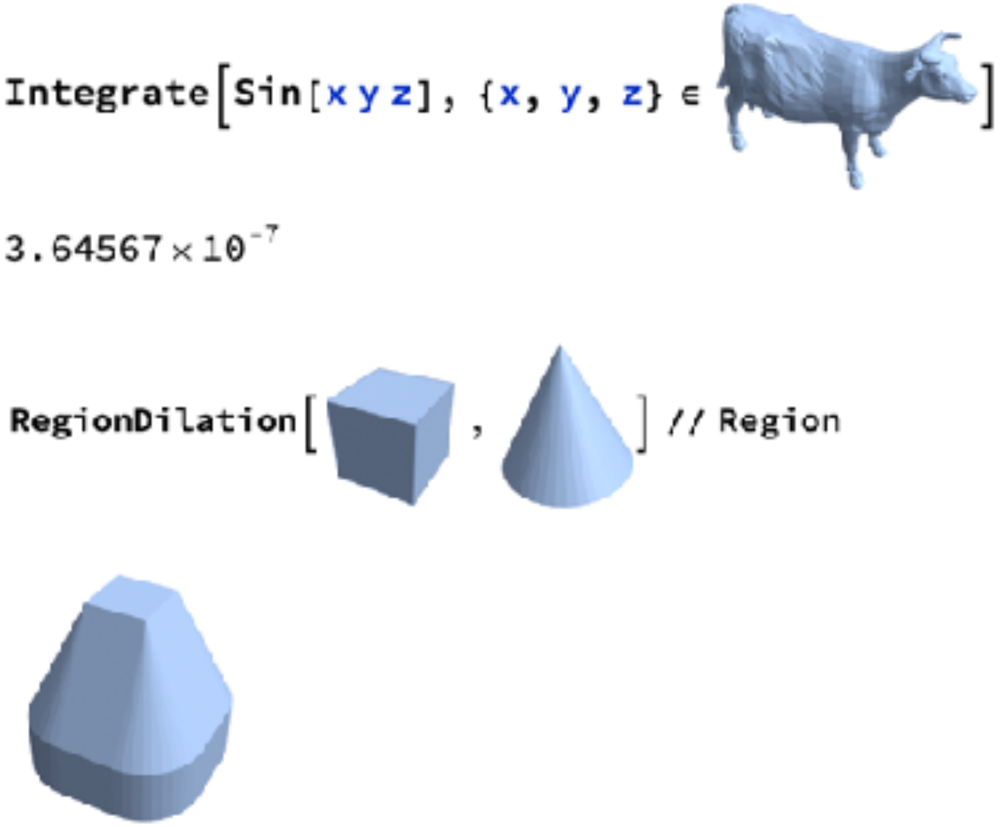
ACCESS OPENVDB IN MATHEMATICA

GREG HURST, UNITED THERAPEUTICS

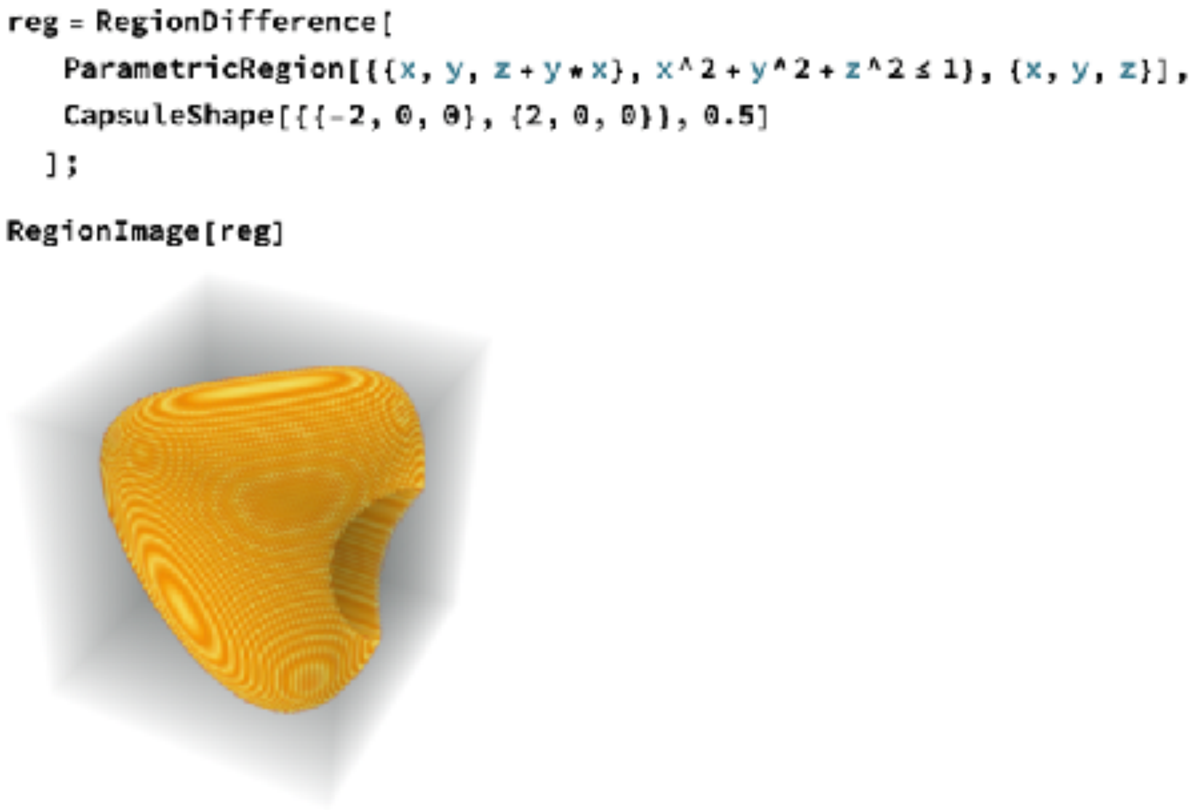


WHY MATHEMATICA?

OpenVDB fits nicely with Mathematica’s rich set of functionality regarding **geometry**, image processing, and graph theory.



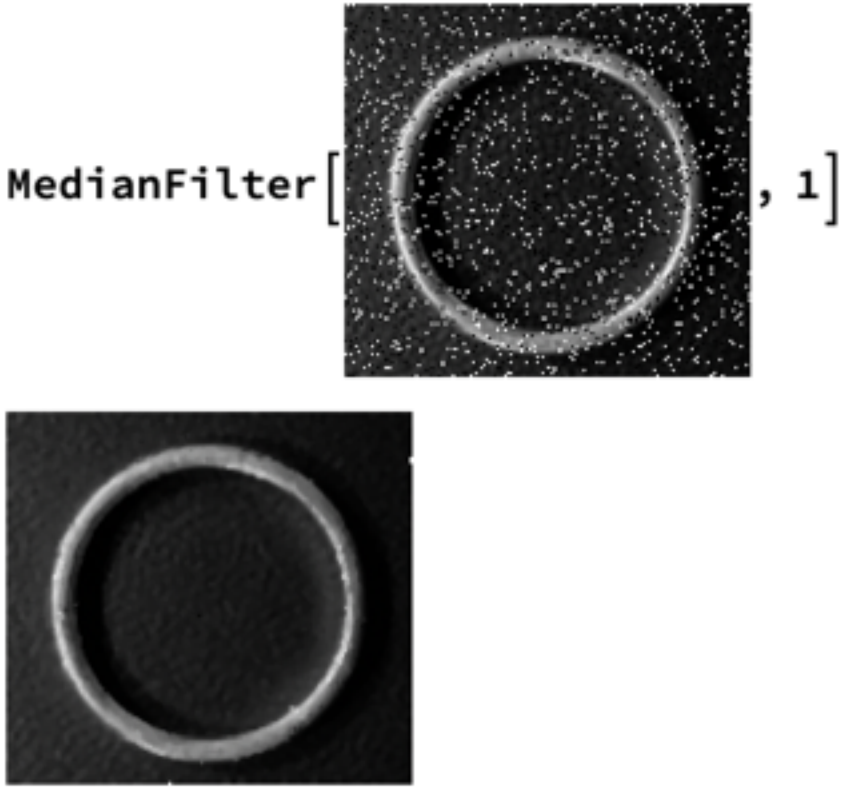
```
Integrate[Sin[x y z], {x, y, z} ∈ ]

3.64567×10^-7

RegionDilation[ , ] // Region
```



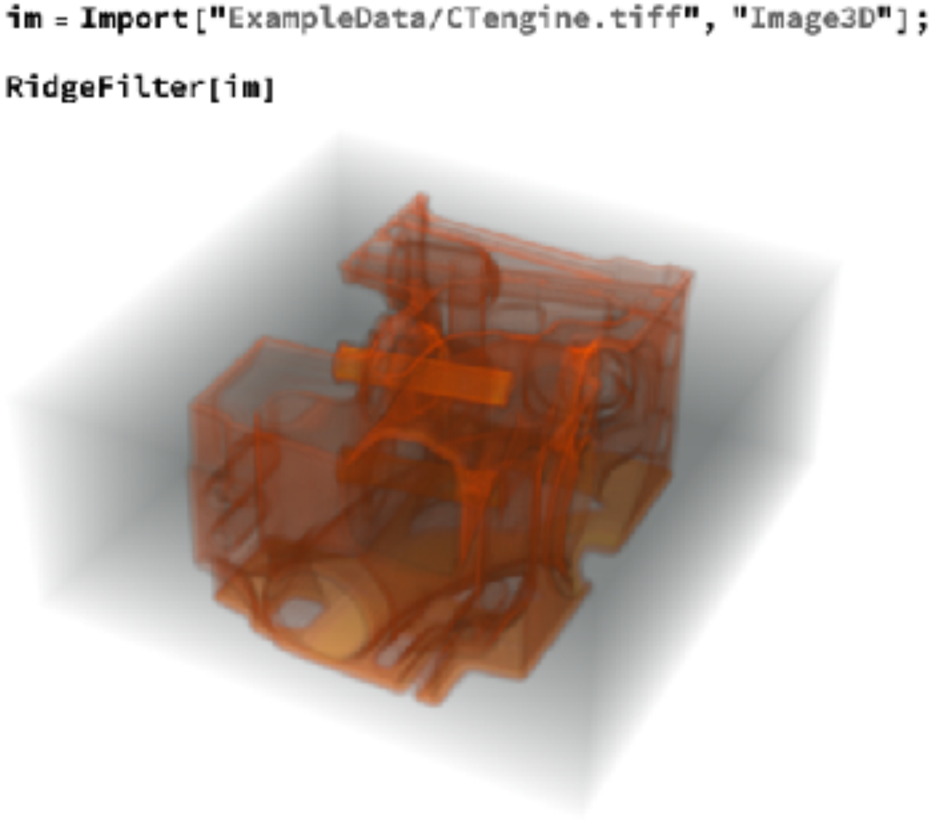
```
reg = RegionDifference[
   ParametricRegion[{{x, y, z + y * x}, x^2 + y^2 + z^2 ≤ 1}, {x, y, z}],
   CapsuleShape[{{-2, 0, 0}, {2, 0, 0}}, 0.5]
  ];

RegionImage[reg]
```

WHY MATHEMATICA?

OpenVDB fits nicely with Mathematica’s rich set of functionality regarding geometry, **image processing**, and graph theory.



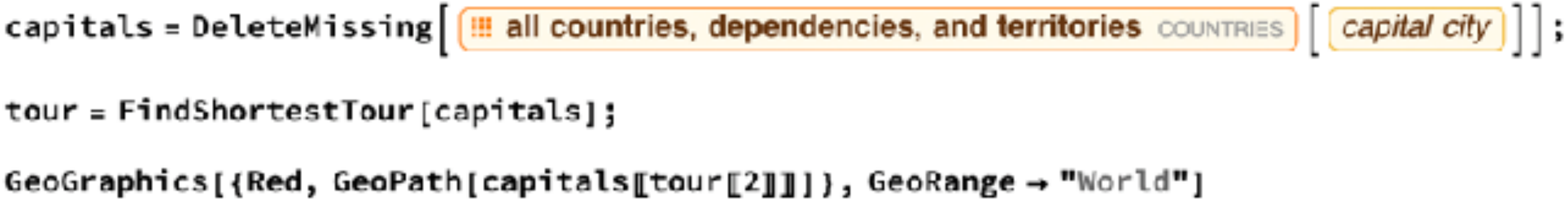
```
MedianFilter[ , 1]
```



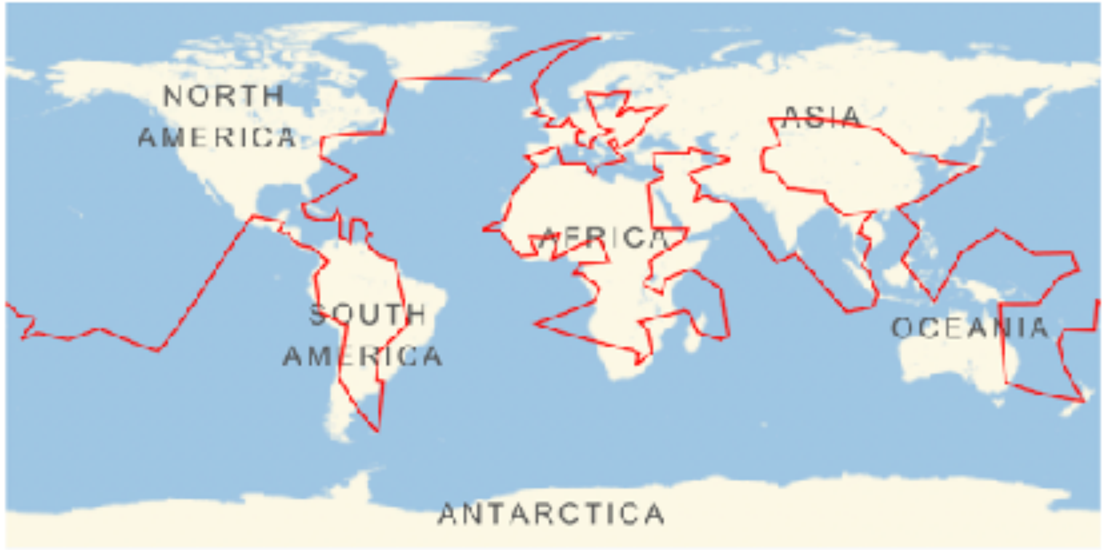
```
im = Import["ExampleData/CTengine.tiff", "Image3D"];

RidgeFilter[im]
```

WHY MATHEMATICA?

OpenVDB fits nicely with Mathematica’s rich set of functionality regarding geometry, image processing, and **graph theory**.

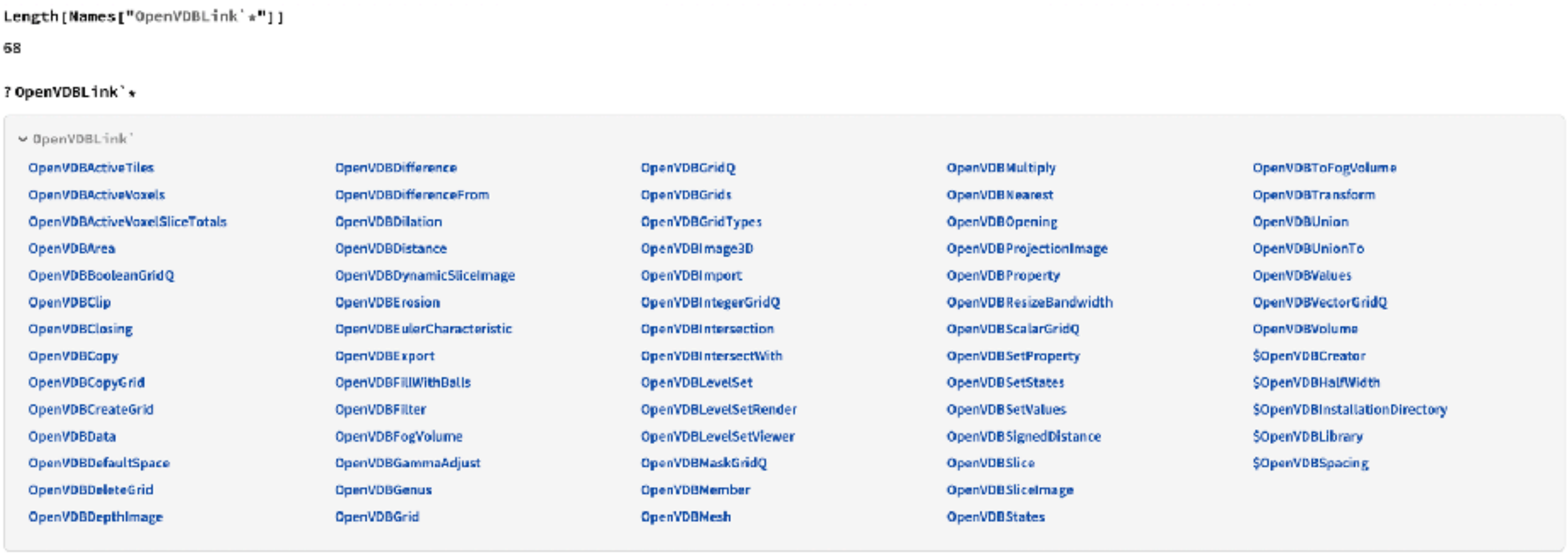


```
capitals = DeleteMissing[ all countries, dependencies, and territories COUNTRIES [ capital city ]];

tour = FindShortestTour[capitals];

GeoGraphics[{Red, GeoPath[capitals⟦tour⟦2⟧⟧]}, GeoRange → "World"]
```



OPENVDBLINK

List of functions exposed (so far)

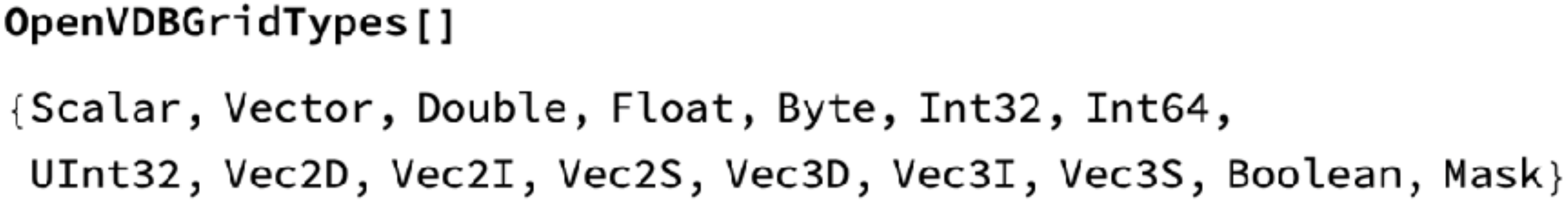


```
Length[Names["OpenVDBLink`*"]]

68

?OpenVDBLink`*
```

```
OpenVDBLink`

OpenVDBActiveTiles
OpenVDBActiveVoxels
OpenVDBActiveVoxelSliceTotals
OpenVDBArea
OpenVDBBooleanGridQ
OpenVDBClip
OpenVDBClosing
OpenVDBCopy
OpenVDBCopyGrid
OpenVDBCreateGrid
OpenVDBData
OpenVDBDefaultSpace
OpenVDBDeleteGrid
OpenVDBDepthImage
OpenVDBDifference
OpenVDBDifferenceFrom
OpenVDBDilation
OpenVDBDistance
OpenVDBDynamicSliceImage
OpenVDBErosion
OpenVDBEulerCharacteristic
OpenVDBExport
OpenVDBFillWithBalls
OpenVDBFilter
OpenVDBFogVolume
OpenVDBGammaAdjust
OpenVDBGenus
OpenVDBGrid
OpenVDBGridQ
OpenVDBGrids
OpenVDBGridTypes
OpenVDBImage3D
OpenVDBImport
OpenVDBIntegerGridQ
OpenVDBIntersection
OpenVDBIntersectWith
OpenVDBLevelSet
OpenVDBLevelSetRender
OpenVDBLevelSetViewer
OpenVDBMaskGridQ
OpenVDBMember
OpenVDBMesh
OpenVDBMultiply
OpenVDBNearest
OpenVDBOpening
OpenVDBProjectionImage
OpenVDBProperty
OpenVDBResizeBandwidth
OpenVDBScalarGridQ
OpenVDBSetProperty
OpenVDBSetStates
OpenVDBSetValues
OpenVDBSignedDistance
OpenVDBSlice
OpenVDBSliceImage
OpenVDBStates
OpenVDBToFogVolume
OpenVDBTransform
OpenVDBUnion
OpenVDBUnionTo
OpenVDBValues
OpenVDBVectorGridQ
OpenVDBVolume
$OpenVDBCreator
$OpenVDBHalfWidth
$OpenVDBInstallationDirectory
$OpenVDBLibrary
$OpenVDBSpacing
```

OPENVDBLINK

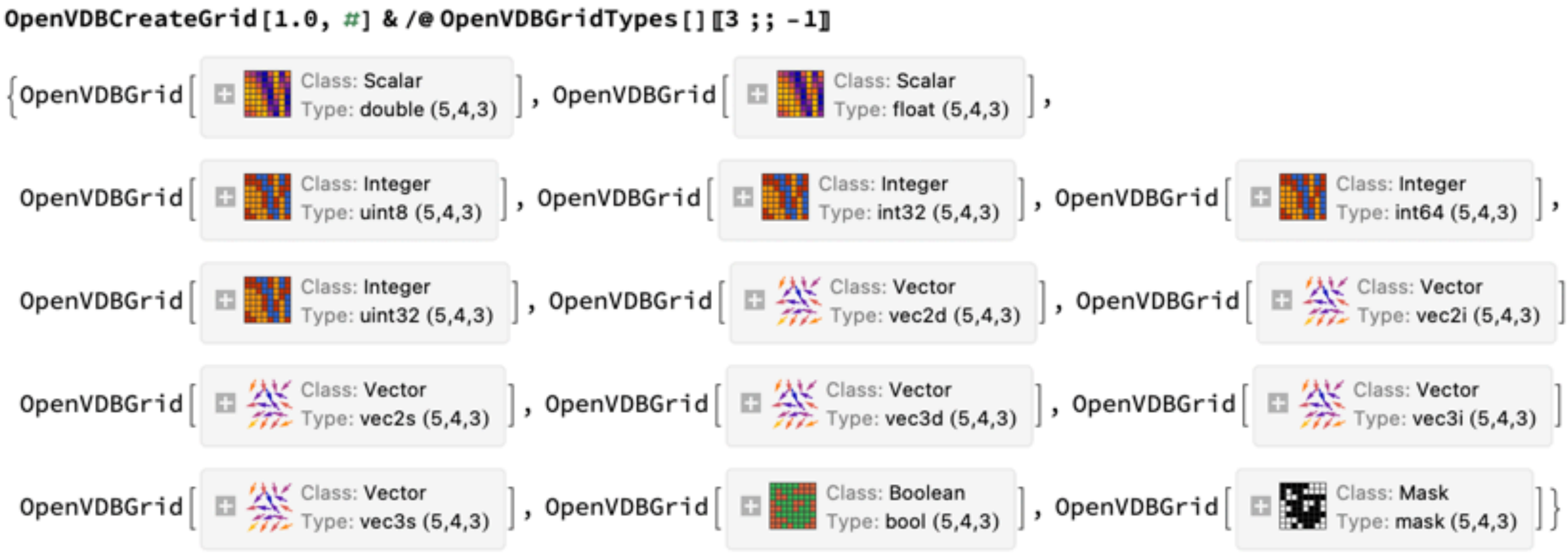
Grid types



```
OpenVDBGridTypes[]

{Scalar, Vector, Double, Float, Byte, Int32, Int64,
 UInt32, Vec2D, Vec2I, Vec2S, Vec3D, Vec3I, Vec3S, Boolean, Mask}
```

Here, "Scalar" is a synonym for "Float" and "Vector" for "Vec3S".

Grid types

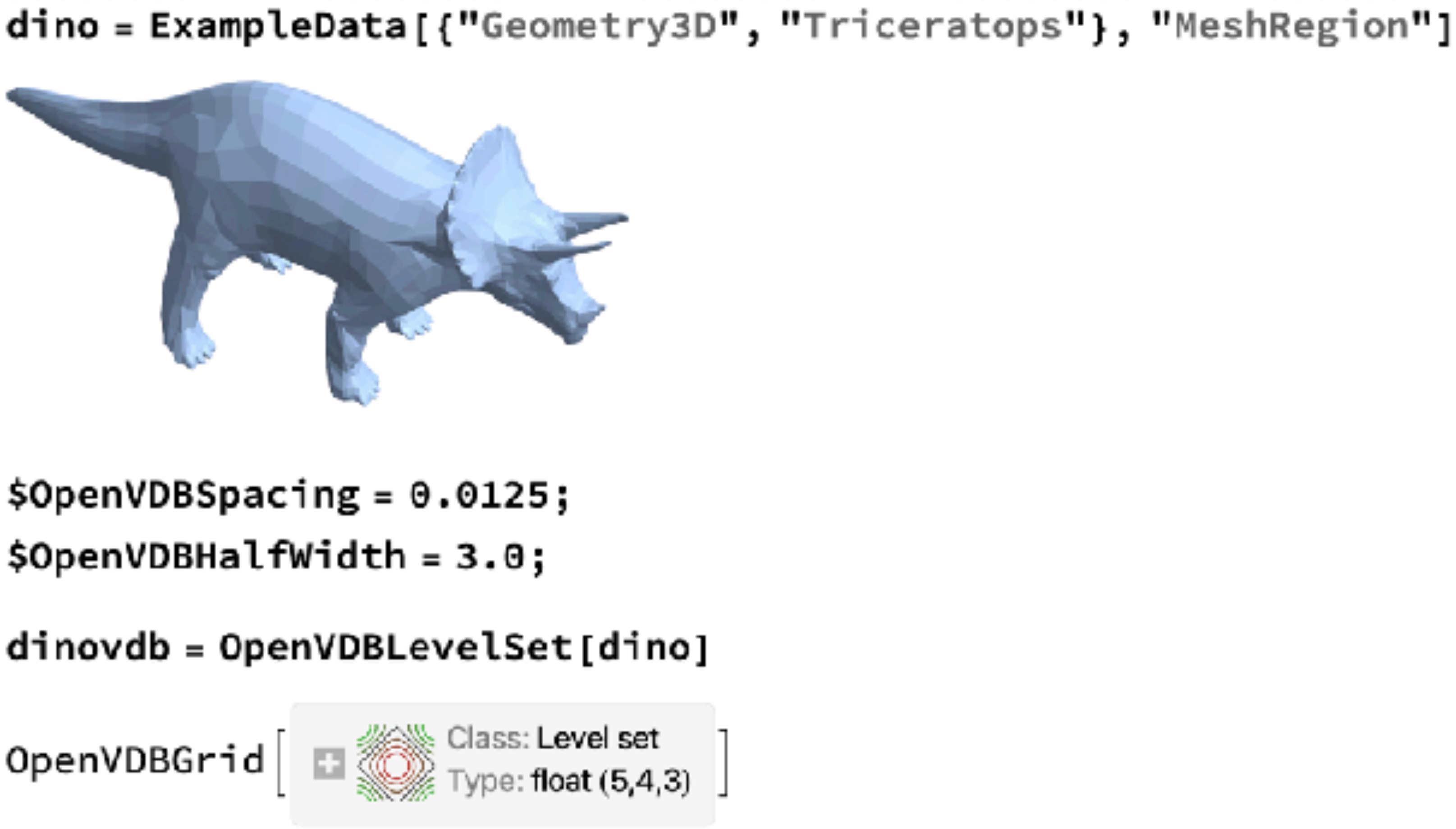


```
OpenVDBCreateGrid[1.0, #] & /@ OpenVDBGridTypes[][[3 ;; -1]]

{OpenVDBGrid[ ⊞ Class: Scalar   Type: double (5,4,3) ], OpenVDBGrid[ ⊞ Class: Scalar   Type: float (5,4,3) ],
 OpenVDBGrid[ ⊞ Class: Integer  Type: uint8 (5,4,3) ], OpenVDBGrid[ ⊞ Class: Integer  Type: int32 (5,4,3) ], OpenVDBGrid[ ⊞ Class: Integer  Type: int64 (5,4,3) ],
 OpenVDBGrid[ ⊞ Class: Integer  Type: uint32 (5,4,3) ], OpenVDBGrid[ ⊞ Class: Vector  Type: vec2d (5,4,3) ], OpenVDBGrid[ ⊞ Class: Vector  Type: vec2i (5,4,3) ],
 OpenVDBGrid[ ⊞ Class: Vector  Type: vec2s (5,4,3) ], OpenVDBGrid[ ⊞ Class: Vector  Type: vec3d (5,4,3) ], OpenVDBGrid[ ⊞ Class: Vector  Type: vec3i (5,4,3) ],
 OpenVDBGrid[ ⊞ Class: Vector  Type: vec3s (5,4,3) ], OpenVDBGrid[ ⊞ Class: Boolean  Type: bool (5,4,3) ], OpenVDBGrid[ ⊞ Class: Mask  Type: mask (5,4,3) ]}
```

Grids are displayed in elided form in a notebook.

OPENVDBLINK

Level sets



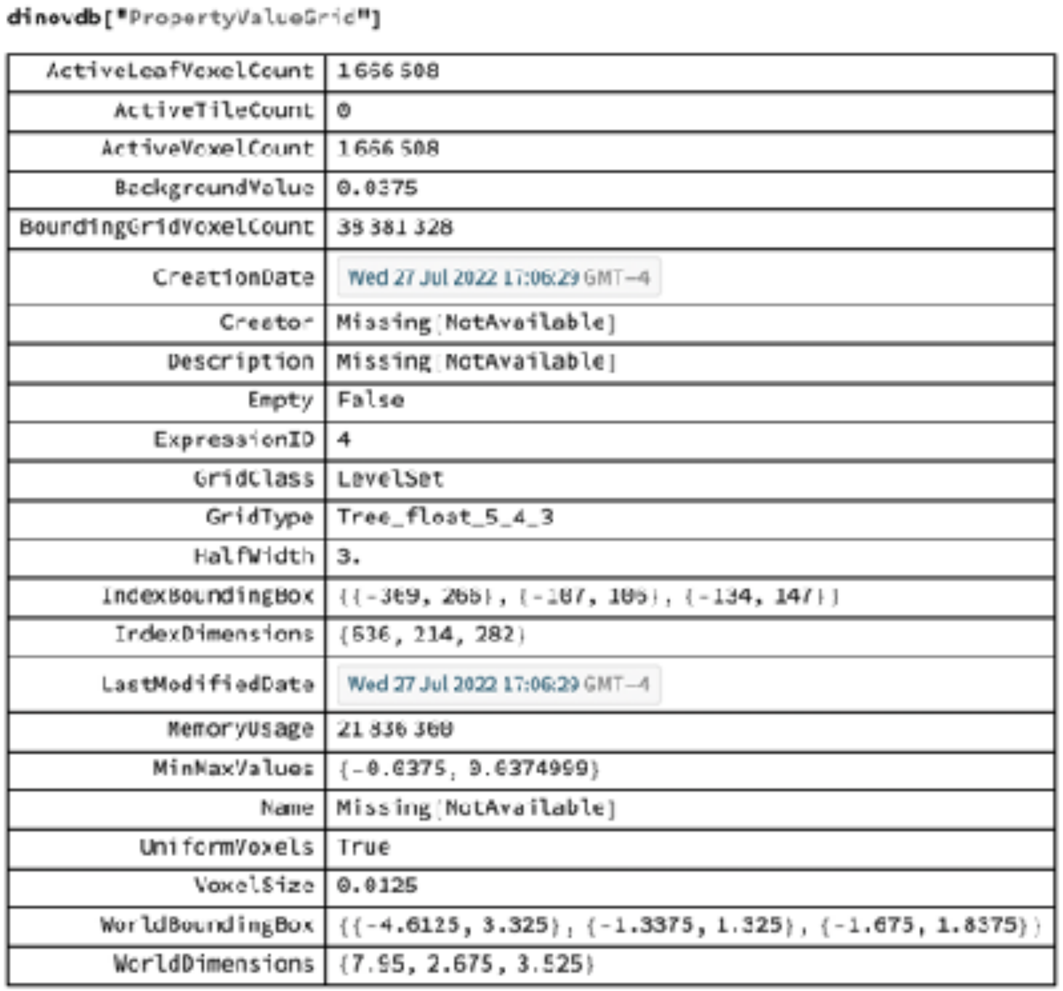
```
dino = ExampleData[{"Geometry3D", "Triceratops"}, "MeshRegion"]
```

```
$OpenVDBSpacing = 0.0125;
$OpenVDBHalfWidth = 3.0;

dinovdb = OpenVDBLevelSet[dino]

OpenVDBGrid[ Class: Level set
             Type: float (5,4,3) ]
```

$OpenVDBSpacing and $OpenVDBHalfWidth are optional global settings.

OPENVDBLINK

Grid properties



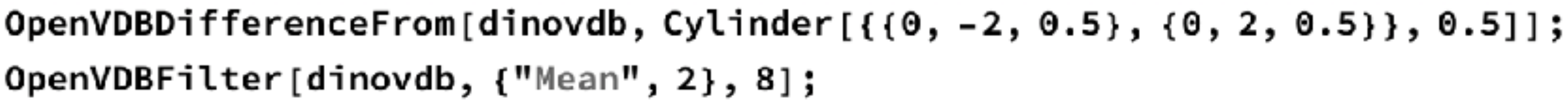
```
dinovdb["PropertyValueGrid"]
```

ActiveLeafVoxelCount	1 656 508
ActiveTileCount	0
ActiveVoxelCount	1 656 508
BackgroundValue	0.0375
BoundingGridVoxelCount	38 381 328
CreationDate	Wed 27 Jul 2022 17:06:29 GMT−4
Creator	Missing[NotAvailable]
Description	Missing[NotAvailable]
Empty	False
ExpressionID	4
GridClass	LevelSet
GridType	Tree_float_5_4_3
HalfWidth	3.
IndexBoundingBox	{{-369, 266}, {-107, 106}, {-134, 147}}
IndexDimensions	{636, 214, 282}
LastModifiedDate	Wed 27 Jul 2022 17:06:29 GMT−4
MemoryUsage	21 836 368
MinMaxValues	{-0.0375, 0.0374999}
Name	Missing[NotAvailable]
UniformVoxels	True
VoxelSize	0.0125
WorldBoundingBox	{{-4.6125, 3.325}, {-1.3375, 1.325}, {-1.675, 1.8375}}
WorldDimensions	{7.95, 2.675, 3.525}

OPENVDBLINK

Modifications

Let's drill a hole in our dino friend and apply 8 applications of a mean filter with window 2.



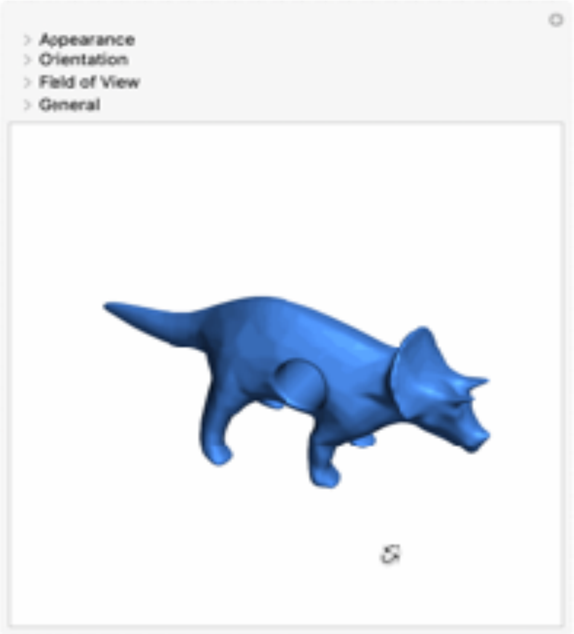
```
OpenVDBDifferenceFrom[dinovdb, Cylinder[{{0, -2, 0.5}, {0, 2, 0.5}}, 0.5]];
OpenVDBFilter[dinovdb, {"Mean", 2}, 8];
```

OPENVDBLINK

Interactive level set viewer

Let’s inspect our creation. The viewer in the notebook lets you choose custom shading, pan, zoom, rotate, clip, etc.

Under the hood, it uses the LevelSetRayIntersector and LevelSetRayTracer classes with custom PBR shader. No conversion to a mesh for rendering.

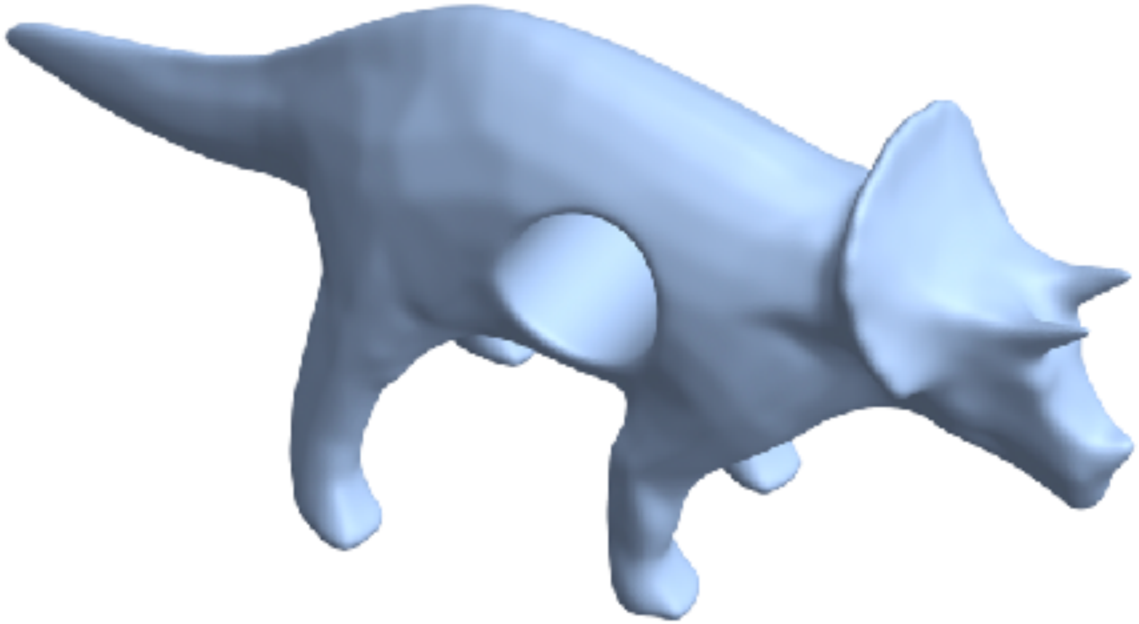


OPENVDBLINK

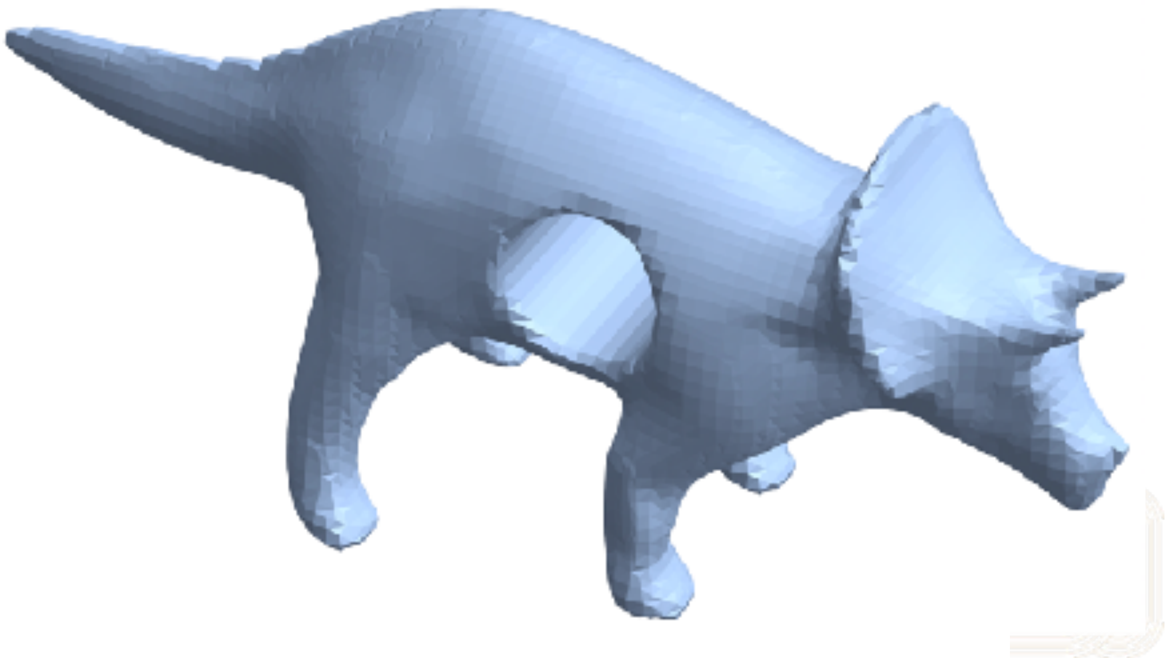
Level set to mesh

Vary the adaptivity:

```
OpenVDBMesh[dinovdb]
```



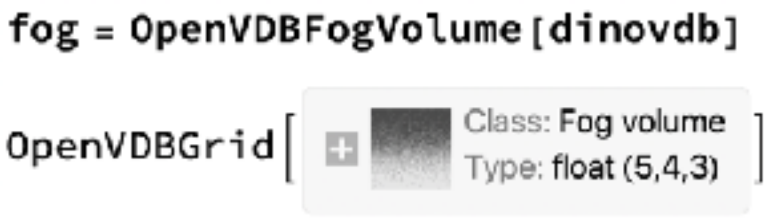
```
OpenVDBMesh[dinovdb, "Adaptivity" → 1]
```



OPENVDBLINK

Fog Volumes

Convert a level set into a fog volume:

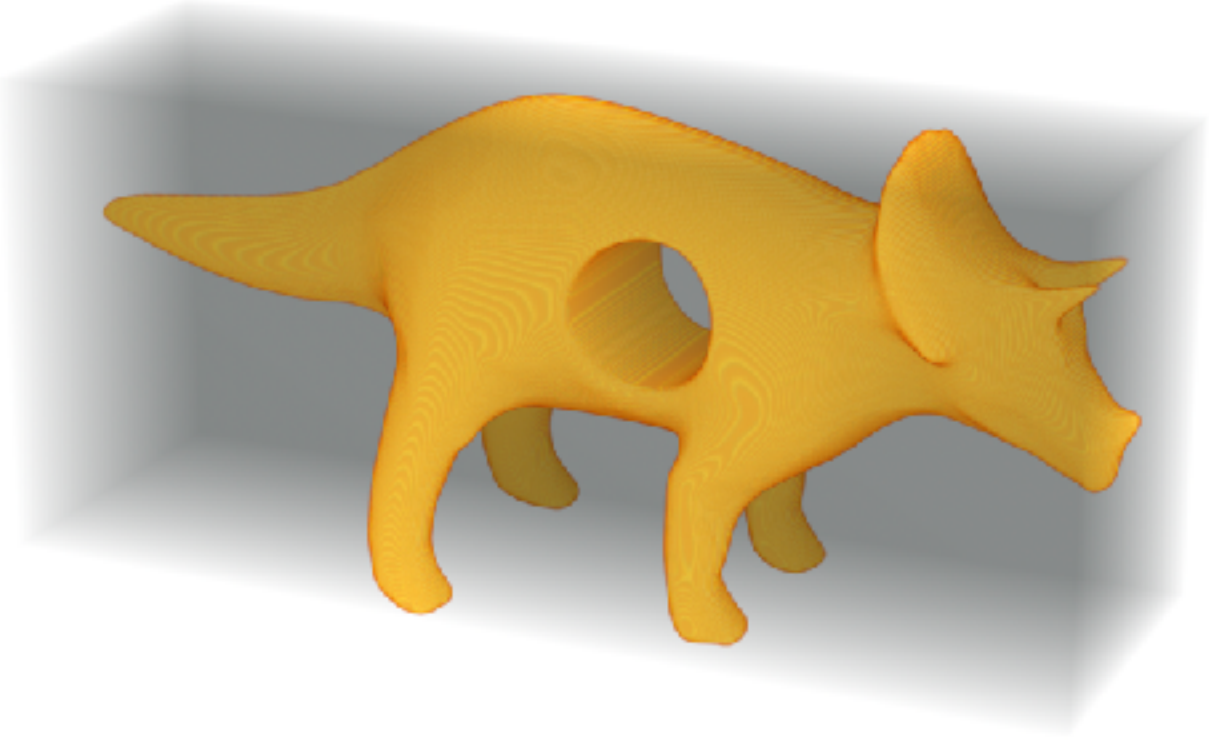


```
fog = OpenVDBFogVolume[dinovdb]

OpenVDBGrid[ Class: Fog volume
             Type: float (5,4,3) ]
```

OPENVDBLINK

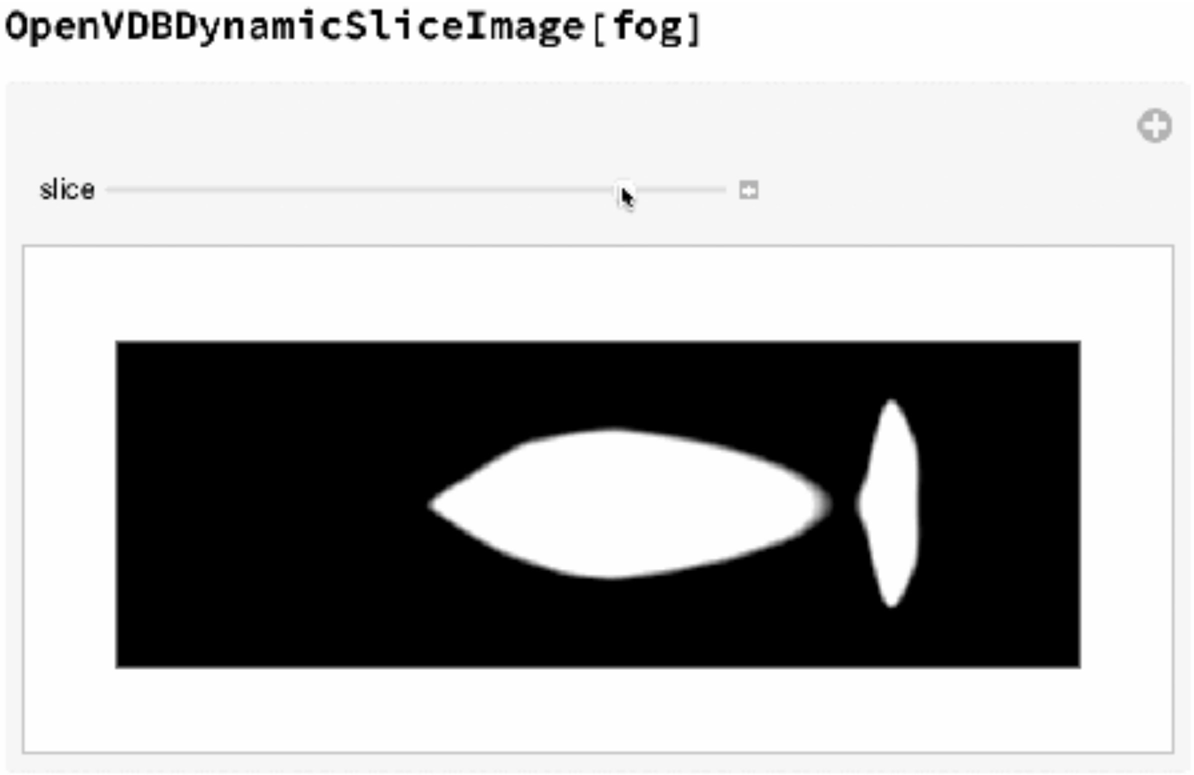
Fog Volumes

Visualize as a (dense) Image3D object: `OpenVDBImage3D[fog]`



Fog Volumes

Slice through the object:

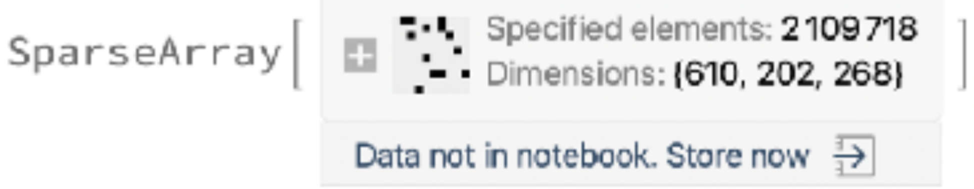


```
OpenVDBDynamicSliceImage[fog]
```

OPENVDBLINK

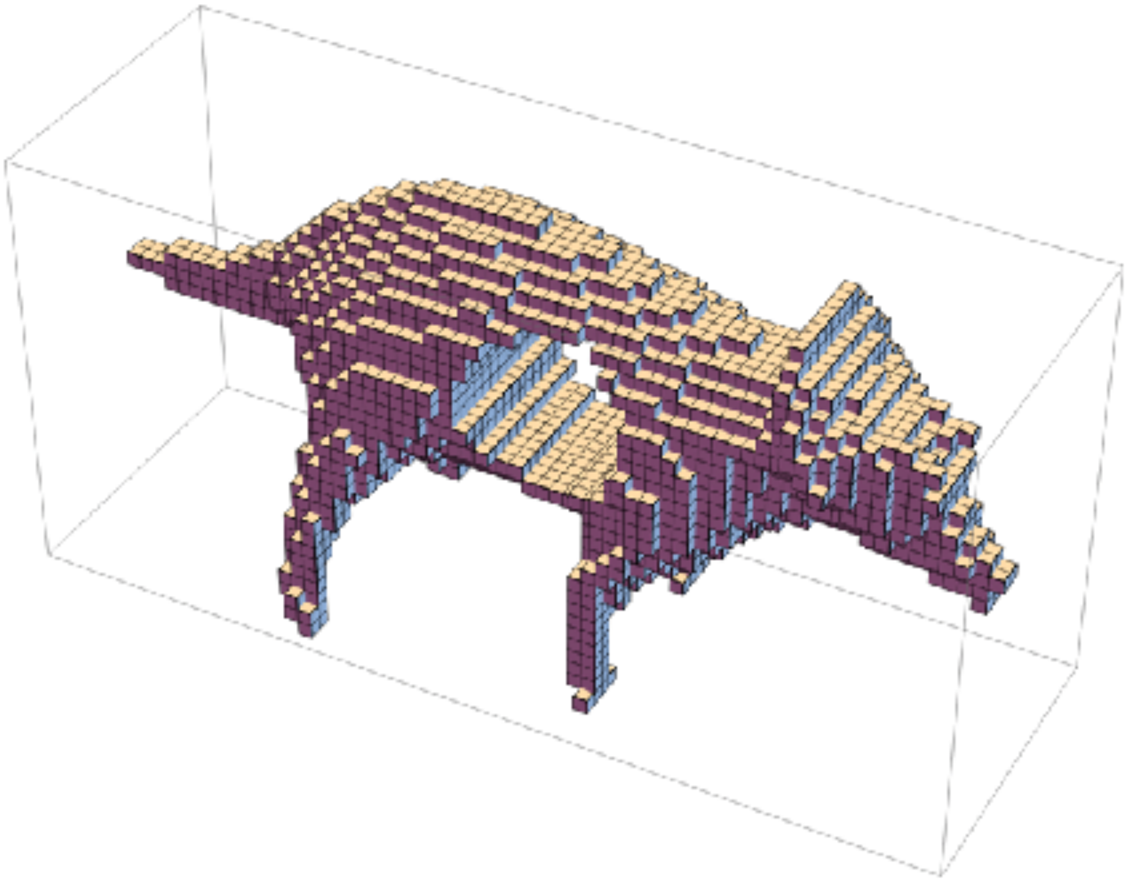
Voxel and tile data

```
OpenVDBActiveVoxels[fog]
```



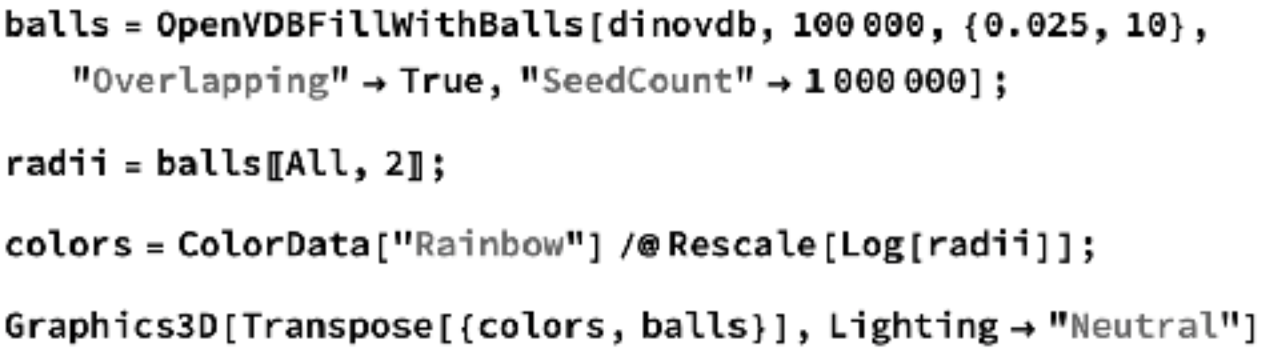
```
SparseArray[ Specified elements: 2109718
             Dimensions: {610, 202, 268} ]
```

```
Graphics3D[Cuboid @@@ OpenVDBActiveTiles[fog]]
```

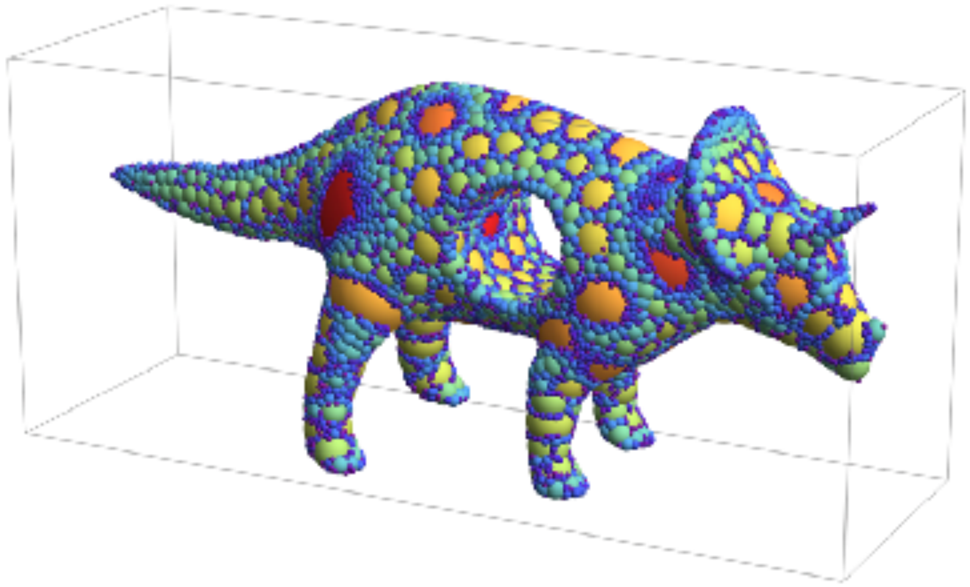


OPENVDBLINK

Neat example

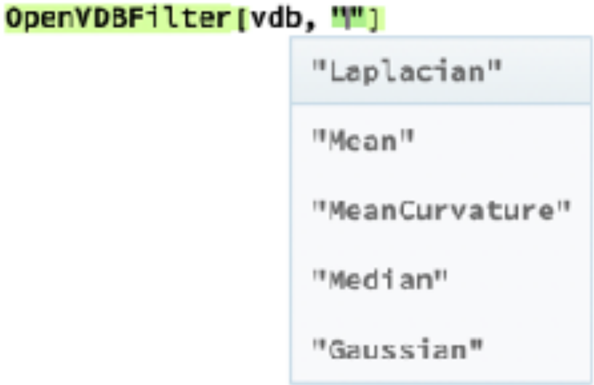


```
balls = OpenVDBFillWithBalls[dinovdb, 100 000, {0.025, 10},
   "Overlapping" → True, "SeedCount" → 1 000 000];

radii = balls⟦All, 2⟧;

colors = ColorData["Rainbow"] /@ Rescale[Log[radii]];

Graphics3D[Transpose[{colors, balls}], Lighting → "Neutral"]
```



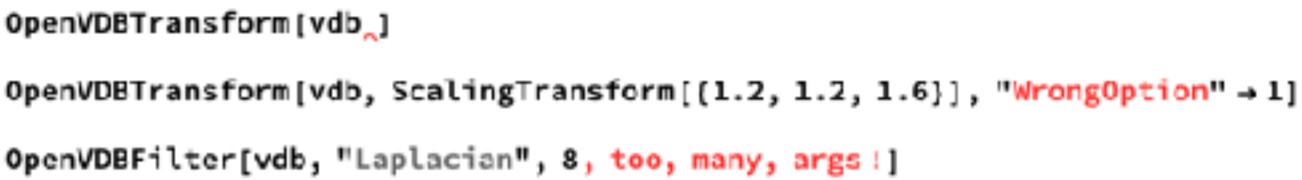
Notebook interfacing

- Mathematica's interrupter works with OpenVDB and is enabled with cmd-.
- OpenVDB can throw exceptions to Mathematica.
- A ate:



```
OpenVDBFilter[vdb, ""]
"Laplacian"
"Mean"
"MeanCurvature"
"Median"
"Gaussian"
```

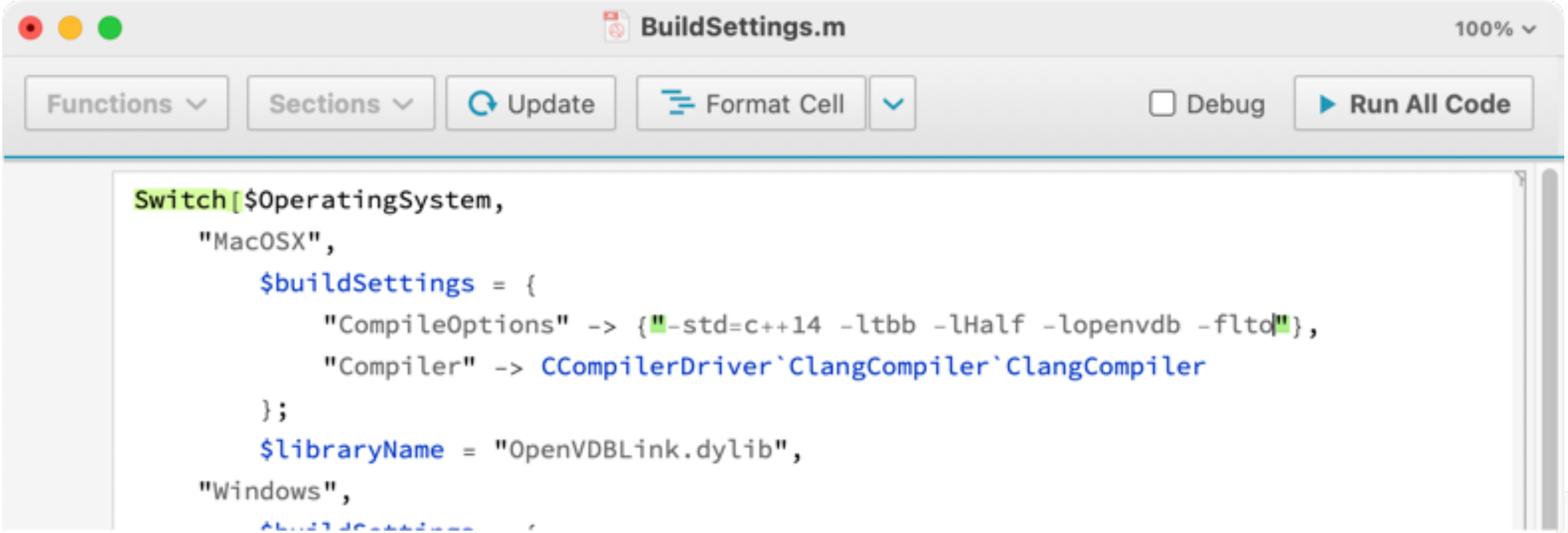
- Highlighting to indicate missing / extra / invalid arguments:



```
OpenVDBTransform[vdb,]
OpenVDBTransform[vdb, ScalingTransform[{1.2, 1.2, 1.6}], "WrongOption" → 1]
OpenVDBFilter[vdb, "Laplacian", 8, too, many, args !]
```

BUILD SETTINGS

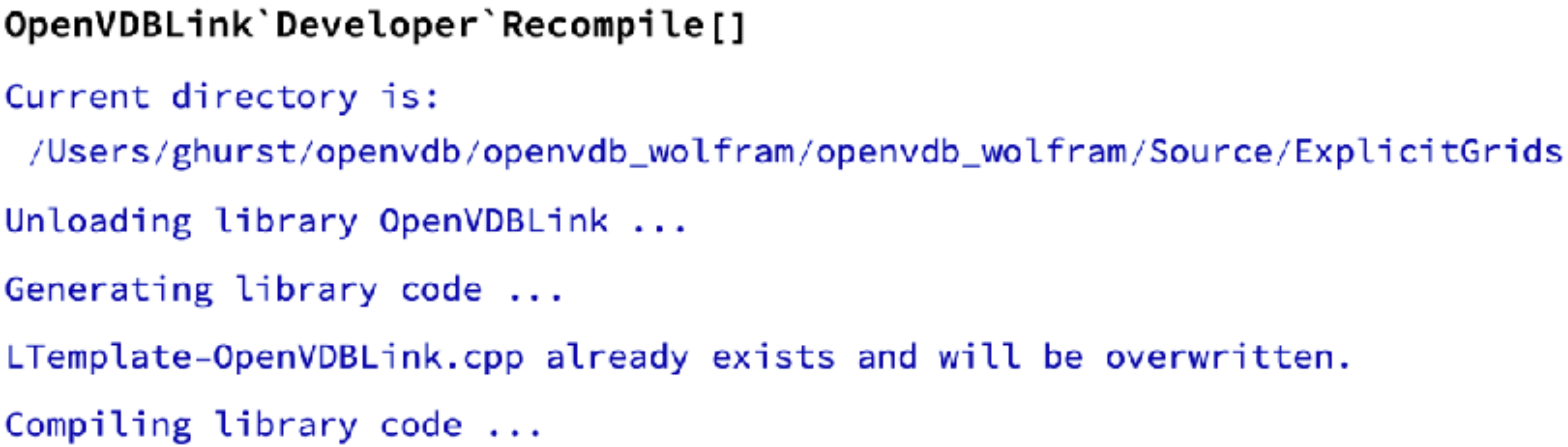
Adjust any build settings as necessary:



```
Switch[$OperatingSystem,
    "MacOSX",
        $buildSettings = {
            "CompileOptions" -> {"-std=c++14 -ltbb -lHalf -lopenvdb -flto"},
            "Compiler" -> CCompilerDriver`ClangCompiler`ClangCompiler
        };
        $libraryName = "OpenVDBLink.dylib",
    "Windows",
        $buildSettings = {
```

COMPILATION

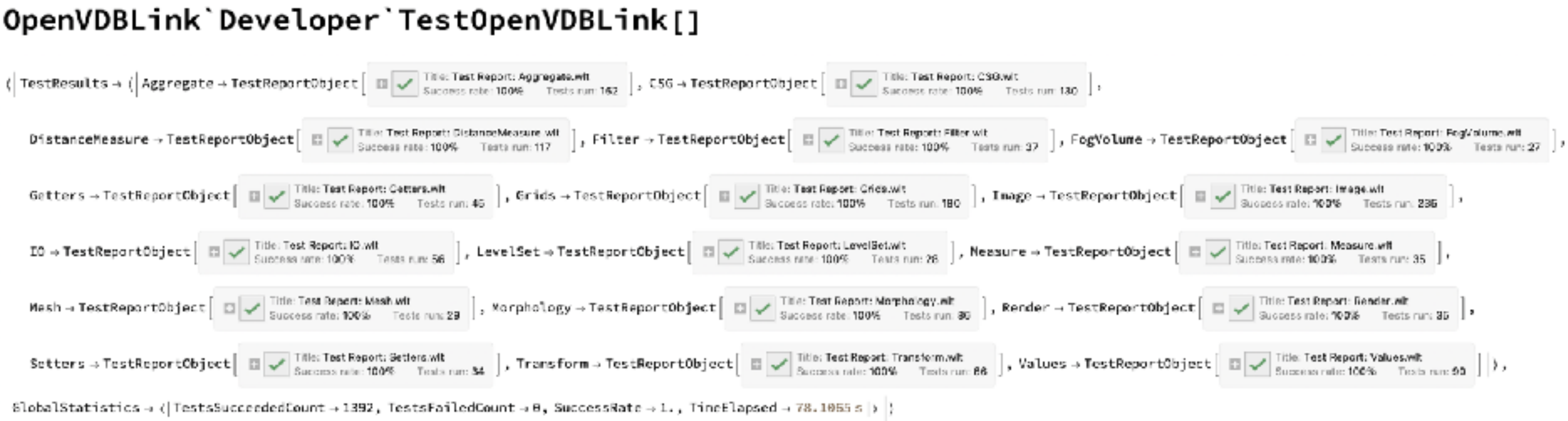
Run this in a Mathematica notebook. Only necessary the first time or if changes are made to the code.



```
OpenVDBLink`Developer`Recompile[]

Current directory is:
 /Users/ghurst/openvdb/openvdb_wolfram/openvdb_wolfram/Source/ExplicitGrids

Unloading library OpenVDBLink ...

Generating library code ...

LTemplate-OpenVDBLink.cpp already exists and will be overwritten.

Compiling library code ...
```

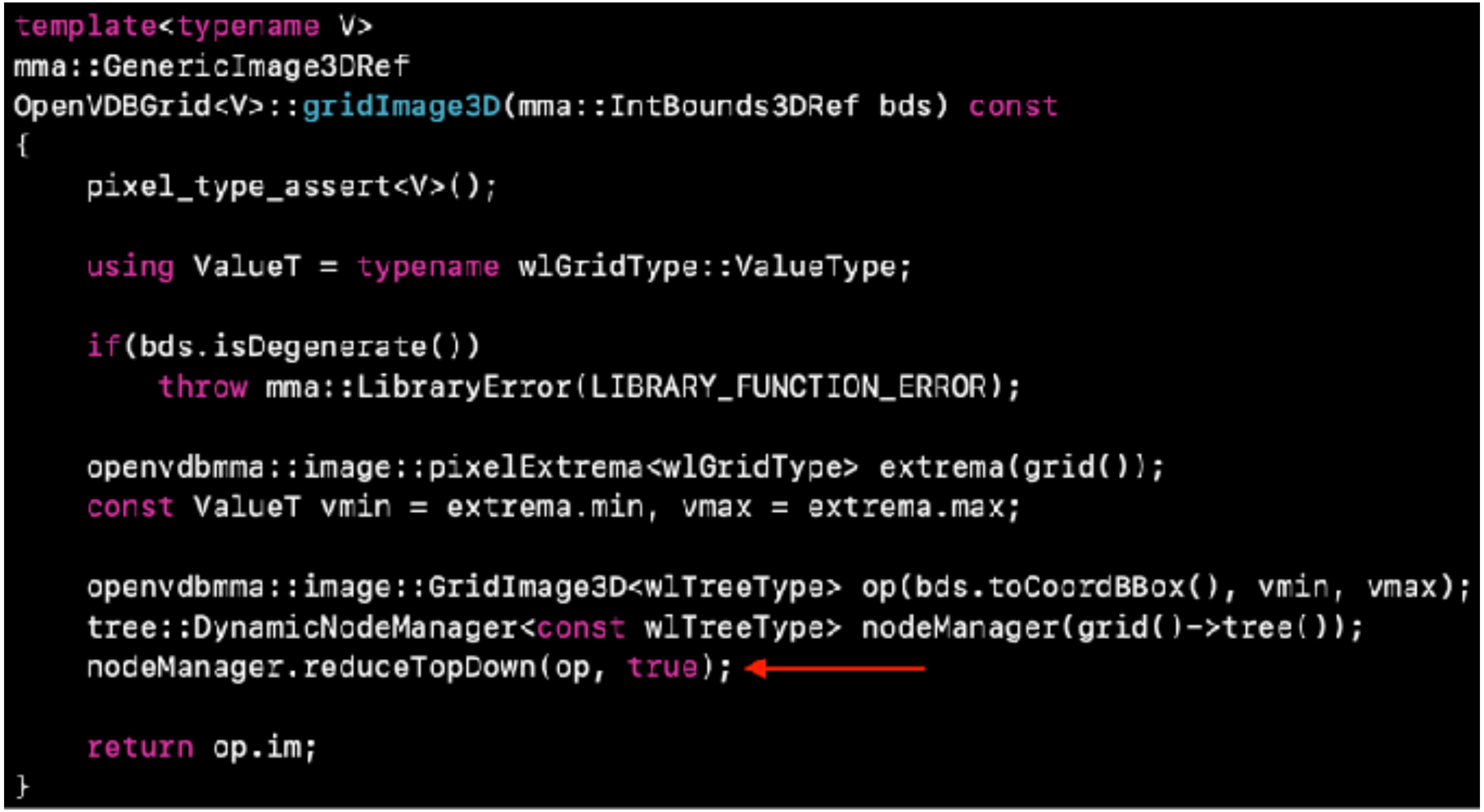
UNIT TESTS

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG



```
OpenVDBLink`Developer`TestOpenVDBLink[]

<|TestResults → <|Aggregate → TestReportObject[ ✓ Title: Test Report: Aggregate.wlt Success rate: 100% Tests run: 162 ], CSG → TestReportObject[ ✓ Title: Test Report: CSG.wlt Success rate: 100% Tests run: 130 ],
   DistanceMeasure → TestReportObject[ ✓ Title: Test Report: DistanceMeasure.wlt Success rate: 100% Tests run: 117 ], Filter → TestReportObject[ ✓ Title: Test Report: Filter.wlt Success rate: 100% Tests run: 37 ], FogVolume → TestReportObject[ ✓ Title: Test Report: FogVolume.wlt Success rate: 100% Tests run: 27 ],
   Getters → TestReportObject[ ✓ Title: Test Report: Getters.wlt Success rate: 100% Tests run: 46 ], Grids → TestReportObject[ ✓ Title: Test Report: Grids.wlt Success rate: 100% Tests run: 180 ], Image → TestReportObject[ ✓ Title: Test Report: Image.wlt Success rate: 100% Tests run: 236 ],
   IO → TestReportObject[ ✓ Title: Test Report: IO.wlt Success rate: 100% Tests run: 56 ], LevelSet → TestReportObject[ ✓ Title: Test Report: LevelSet.wlt Success rate: 100% Tests run: 26 ], Measure → TestReportObject[ ✓ Title: Test Report: Measure.wlt Success rate: 100% Tests run: 35 ],
   Mesh → TestReportObject[ ✓ Title: Test Report: Mesh.wlt Success rate: 100% Tests run: 29 ], Morphology → TestReportObject[ ✓ Title: Test Report: Morphology.wlt Success rate: 100% Tests run: 80 ], Render → TestReportObject[ ✓ Title: Test Report: Render.wlt Success rate: 100% Tests run: 35 ],
   Setters → TestReportObject[ ✓ Title: Test Report: Setters.wlt Success rate: 100% Tests run: 34 ], Transform → TestReportObject[ ✓ Title: Test Report: Transform.wlt Success rate: 100% Tests run: 88 ], Values → TestReportObject[ ✓ Title: Test Report: Values.wlt Success rate: 100% Tests run: 55 ]|>,
 GlobalStatistics → <|TestsSucceededCount → 1392, TestsFailedCount → 0, SuccessRate → 1., TimeElapsed → 78.1055 s|>|>
```

UNDER THE HOOD

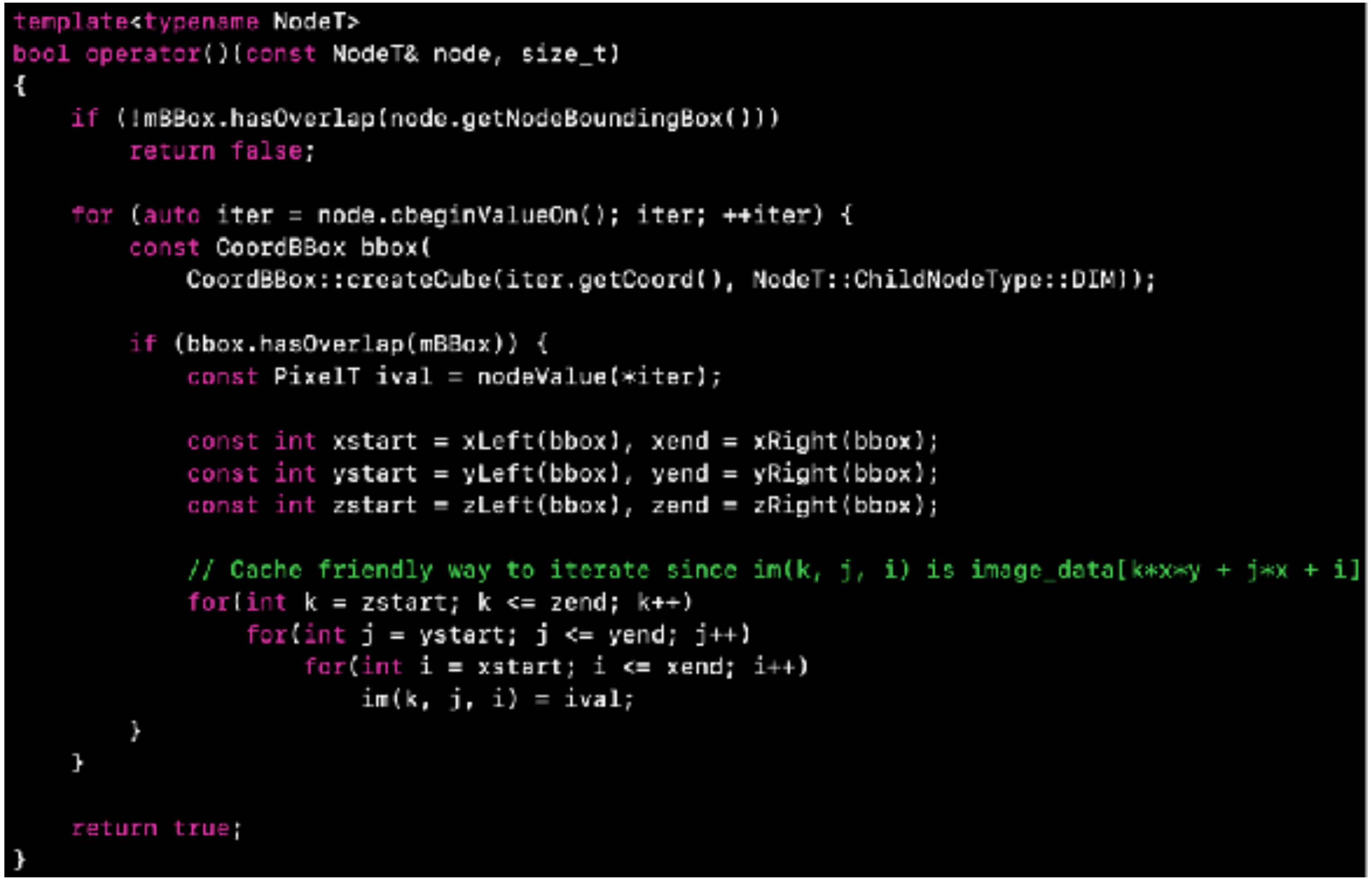


```
template<typename V>
mma::GenericImage3DRef
OpenVDBGrid<V>::gridImage3D(mma::IntBounds3DRef bds) const
{
    pixel_type_assert<V>();

    using ValueT = typename wlGridType::ValueType;

    if(bds.isDegenerate())
        throw mma::LibraryError(LIBRARY_FUNCTION_ERROR);

    openvdbmma::image::pixelExtrema<wlGridType> extrema(grid());
    const ValueT vmin = extrema.min, vmax = extrema.max;

    openvdbmma::image::GridImage3D<wlTreeType> op(bds.toCoordBBox(), vmin, vmax);
    tree::DynamicNodeManager<const wlTreeType> nodeManager(grid()->tree());
    nodeManager.reduceTopDown(op, true);

    return op.im;
}
```

Functions make use of efficient schemes such as DynamicNodeManager

UNDER THE HOOD

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG



```
template<typename NodeT>
bool operator()(const NodeT& node, size_t)
{
    if (!mBBox.hasOverlap(node.getNodeBoundingBox()))
        return false;

    for (auto iter = node.cbeginValueOn(); iter; ++iter) {
        const CoordBBox bbox(
            CoordBBox::createCube(iter.getCoord(), NodeT::ChildNodeType::DIM));

        if (bbox.hasOverlap(mBBox)) {
            const PixelT ival = nodeValue(*iter);

            const int xstart = xLeft(bbox), xend = xRight(bbox);
            const int ystart = yLeft(bbox), yend = yRight(bbox);
            const int zstart = zLeft(bbox), zend = zRight(bbox);

            // Cache friendly way to iterate since im(k, j, i) is image_data[k*x*y + j*x + i]
            for(int k = zstart; k <= zend; k++)
                for(int j = ystart; j <= yend; j++)
                    for(int i = xstart; i <= xend; i++)
                        im(k, j, i) = ival;
        }
    }

    return true;
}
```

A routine DynamicNodeManager uses to populate an active tile in an Image3D

THANK YOU!

Greg Hurst

ghurst588@gmail.com