SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

OPENVDB COURSE

SIMD & PARTICLE SURFACING

NICHOLAS AVRAMOUSSIS
Rendering Software Engineer - WETA

THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

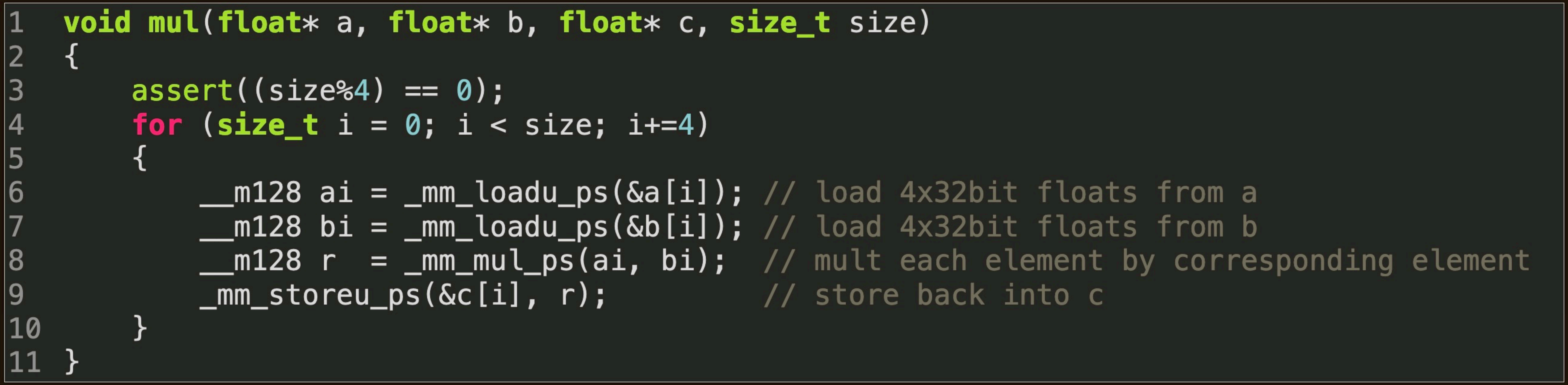
SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

SIMD IN OPENVDB

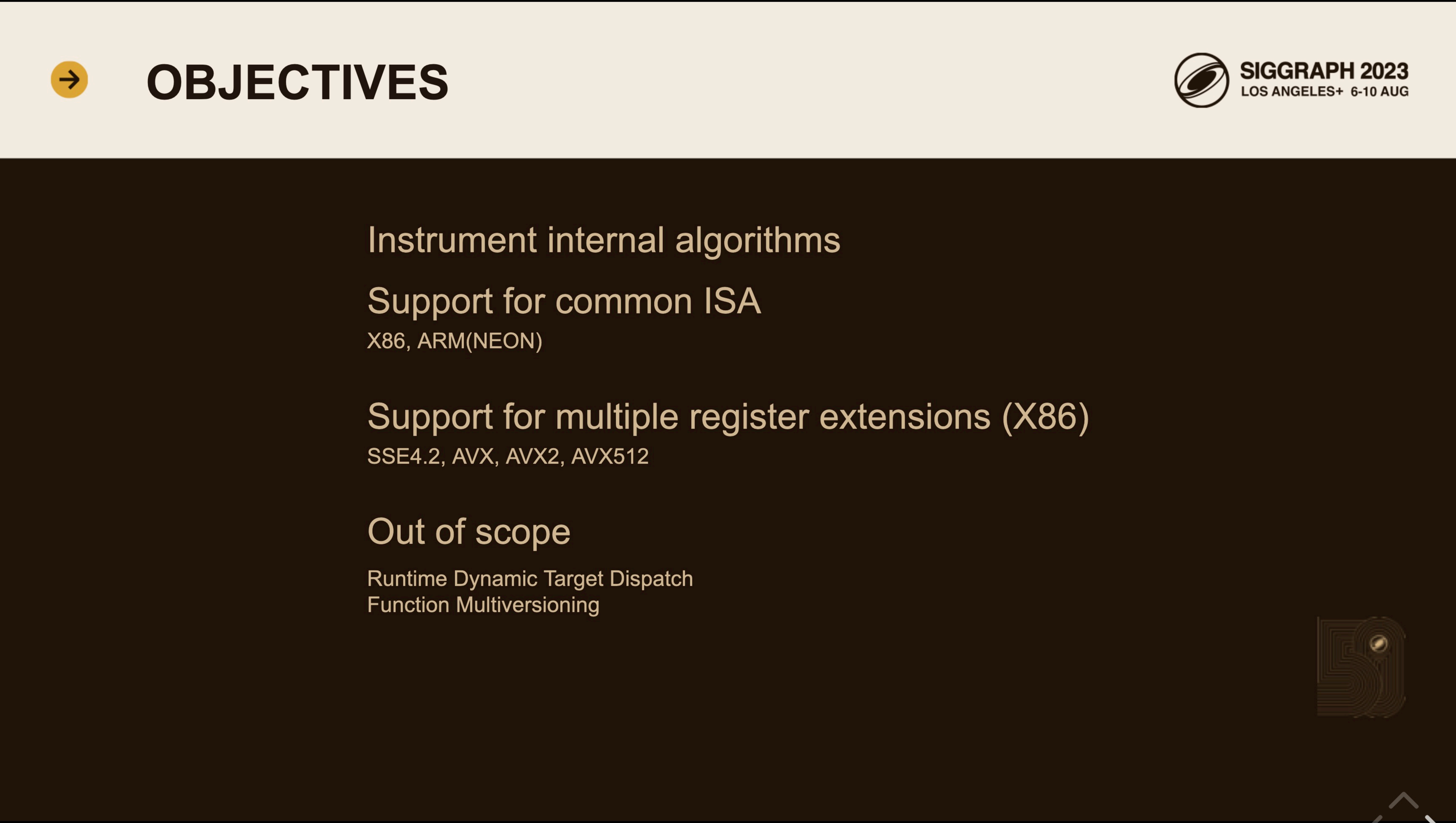
WHAT DO WE MEAN BY SIMD?

```
void mul(float* a, float* b, float* c, size_t size)
{
    for (size_t i = 0; i < size; ++i)
    {
        c[i] = a[i] * b[i];
    }
}
```

WHAT DO WE MEAN BY SIMD?



```
void mul(float* a, float* b, float* c, size_t size)
{
    assert((size%4) == 0);
    for (size_t i = 0; i < size; i+=4)
    {
        __m128 ai = _mm_loadu_ps(&a[i]); // load 4x32bit floats from a
        __m128 bi = _mm_loadu_ps(&b[i]); // load 4x32bit floats from b
        __m128 r  = _mm_mul_ps(ai, bi);  // mult each element by corresponding element
        _mm_storeu_ps(&c[i], r);         // store back into c
    }
}
```



OBJECTIVES

Instrument internal algorithms

Support for common ISA

X86, ARM(NEON)

Support for multiple register extensions (X86)

SSE4.2, AVX, AVX2, AVX512

Out of scope

Runtime Dynamic Target Dispatch
Function Multiversioning

AUTO VECTORIZATION

```cpp
const Mat4<float> &operator*=(const Mat4<float>& m1)
{
    Mat4<float> m0(*this); // copy of this
    for (int i = 0; i < 4; ++i) {
        this->mm[i] = ... // do the thing
    }
    return *this;
}
```

m_1 may overlap with $*\underline{this}$! e.g.

```cpp
Mat4f a, b; b = a*a; // b = a.operator*(a);
```

Compiler may decide not to vectorize :(

With Mat4f&

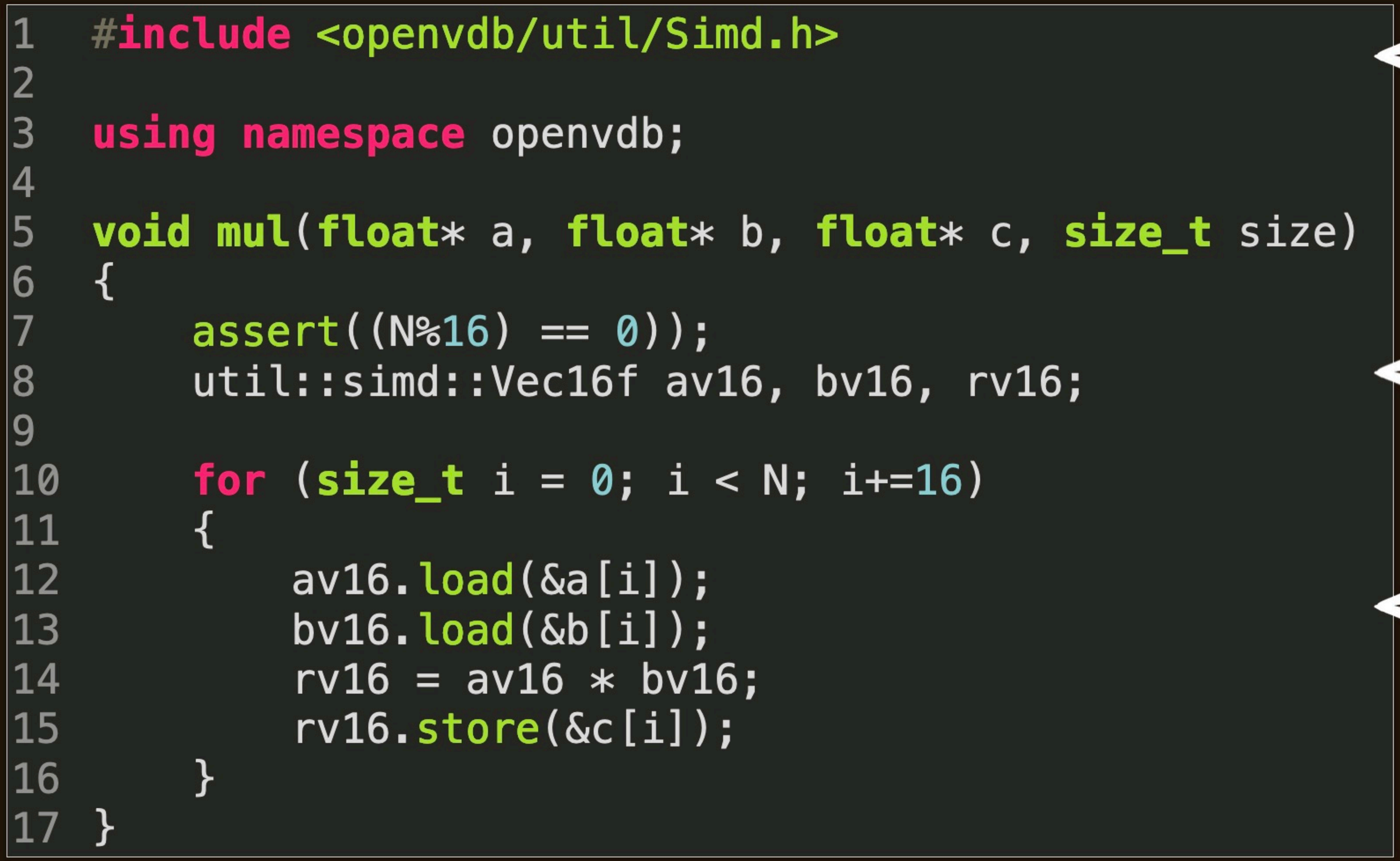
```
...
%0 = load float, float* %arraydecay3, align 4
%1 = load float, float* %arrayidx9, align 4
%mul10 = fmul float %m0.sroa.4.0.copyload, %1
%2 = tail call float @llvm.fmuladd.f32(float %m0.sro
%3 = load float, float* %arrayidx14, align 4
%4 = tail call float @llvm.fmuladd.f32(float %m0.sro
%5 = load float, float* %arrayidx19, align 4
%6 = tail call float @llvm.fmuladd.f32(float %m0.sro
store float %6, float* %m0.sroa.0.0..sroa_idx, align
...
```

With Mat4f

```
...
%33 = fmul <4 x float> %3, %32
%34 = shufflevector <4 x float> %7, <4 x float> unde
%35 = call <4 x float> @llvm.fmuladd.v4f32(<4 x floa
%36 = shufflevector <4 x float> %7, <4 x float> unde
%37 = call <4 x float> @llvm.fmuladd.v4f32(<4 x floa
%38 = shufflevector <4 x float> %7, <4 x float> unde
%39 = call <4 x float> @llvm.fmuladd.v4f32(<4 x floa
%40 = bitcast float* %agg.tmp.sroa.0.sroa.13.0.agg.t
store <4 x float> %39, <4 x float>* %40, align 4
...
```

https://godbolt.org/z/9Pj73Y4cz

WRAPPER LIBRARIES



```cpp
#include <openvdb/util/Simd.h>

using namespace openvdb;

void mul(float* a, float* b, float* c, size_t size)
{
    assert((N%16) == 0));
    util::simd::Vec16f av16, bv16, rv16;

    for (size_t i = 0; i < N; i+=16)
    {
        av16.load(&a[i]);
        bv16.load(&b[i]);
        rv16 = av16 * bv16;
        rv16.store(&c[i]);
    }
}
```

Import SIMD Wrappers

Namespaced/aliased SIMD containers

Compile time instruction selection

EMULATION

	SSE4.2 (128)	AVX (256)	AVX512
`simd::Vec4f // (32 x 4 = 128)`	`__m128 xmm;`	`__m128 xmm;`	`__m128 xmm;`

`simd::Vec8f // (32 x 8 = 256)`	`simd::Vec4f low;`
	`simd::Vec4f high;`
		`__m256 ymm;`	`__m256 ymm;`

`simd::Vec16f // (32 x 16 = 512)`	`// 2x simd::Vec4f;`
	`using Vec8f = simd::Vec8f_e;`
	
	`simd::Vec8f low;`
	`simd::Vec8f high;`
		`simd::Vec8f low;`
		`simd::Vec8f high;`
			`__m512 zmm;`

Compatible API type for all vector extensions

https://github.com/vectorclass/version2

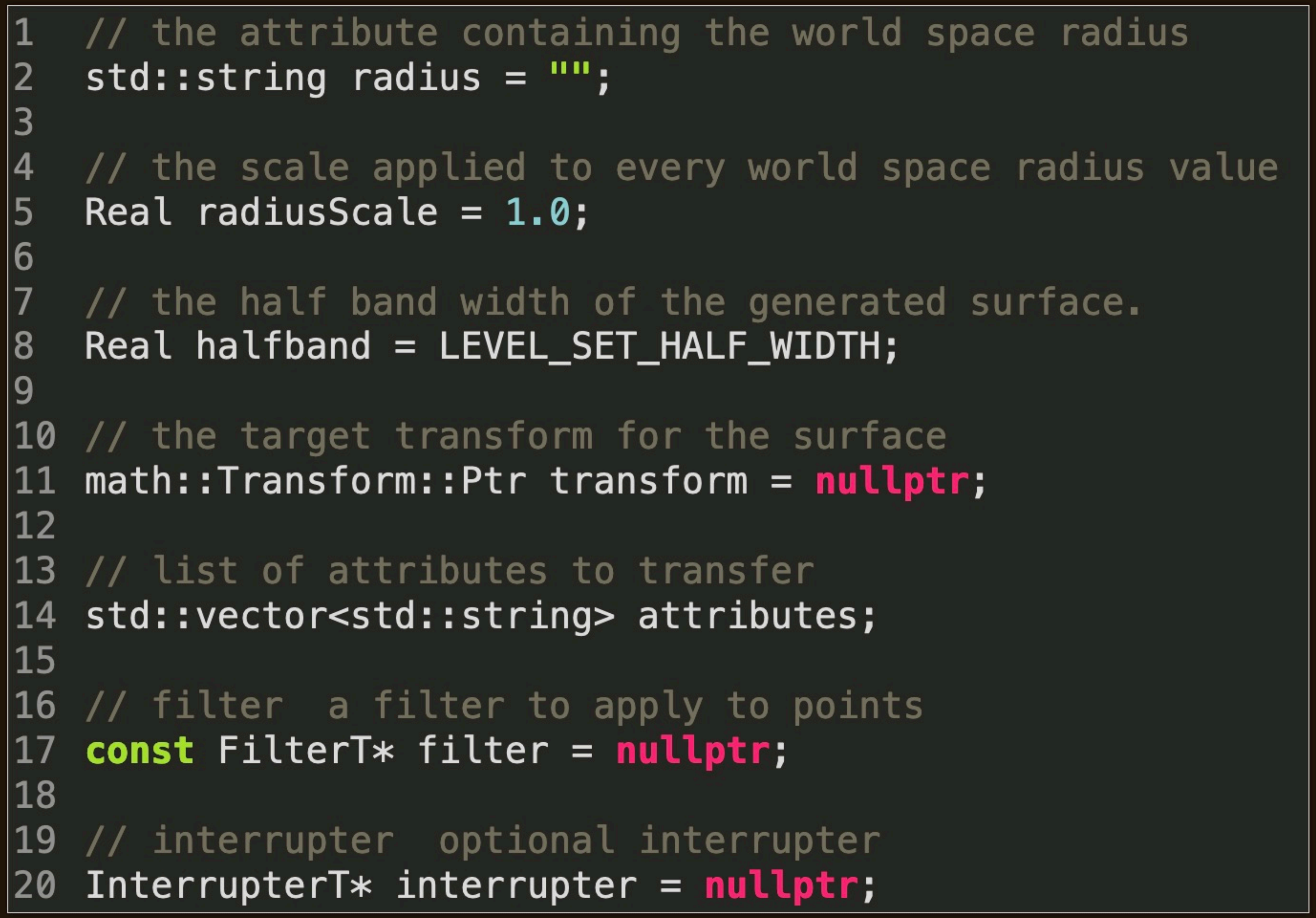
PRE V9.1.X TOOLS

```cpp
#include <openvdb/tools/ParticlesToLevelSet.h>

/// @brief Populate a scalar, floating-point grid with fixed-size, CSG-unioned
/// level set spheres described by the given particle positions and the specified radius.
template<typename GridT, typename ParticleListT, typename InterrupterT = util::NullInterrupter>
void particlesToSdf(const ParticleListT&, GridT&, Real radius, InterrupterT* = nullptr);

/// @brief Activate a boolean grid wherever it intersects the fixed-size spheres
/// described by the given particle positions and the specified radius.
template<typename GridT, typename ParticleListT, typename InterrupterT = util::NullInterrupter>
void particlesToMask(const ParticleListT&, GridT&, Real radius, InterrupterT* = nullptr);

template<typename SdfGridT,
         typename AttributeT = void,
         typename InterrupterT = util::NullInterrupter>
class ParticlesToLevelSet;
```

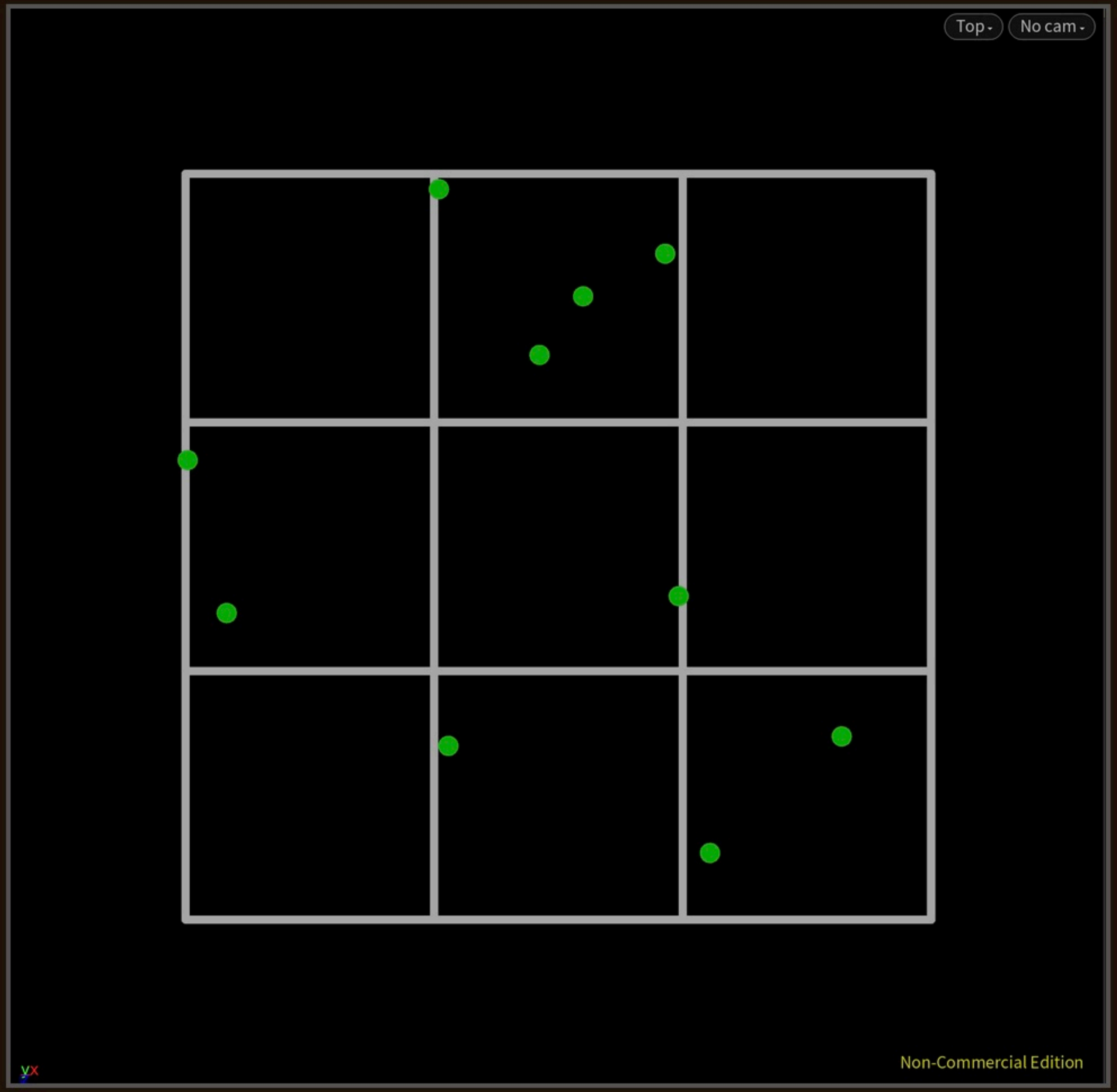
EXISTING TOOLS: SPHERES

signed distance field of an isolated particle x_0 with radius r_0:

$$\phi(x) = |x - x_0| - r_0$$



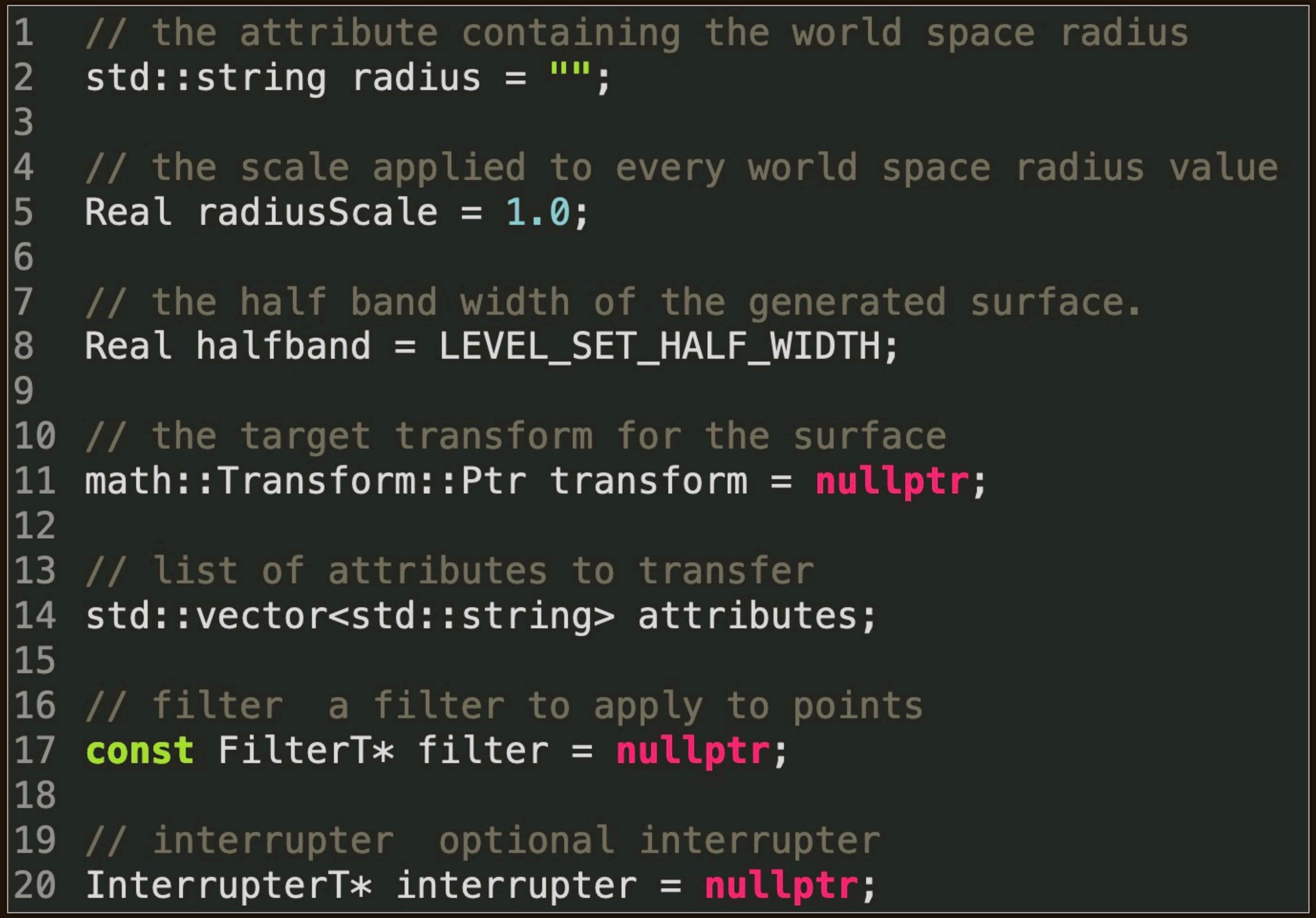
```
// the attribute containing the world space radius
std::string radius = "";

// the scale applied to every world space radius value
Real radiusScale = 1.0;

// the half band width of the generated surface.
Real halfband = LEVEL_SET_HALF_WIDTH;

// the target transform for the surface
math::Transform::Ptr transform = nullptr;

// list of attributes to transfer
std::vector<std::string> attributes;

// filter  a filter to apply to points
const FilterT* filter = nullptr;

// interrupter  optional interrupter
InterrupterT* interrupter = nullptr;
```



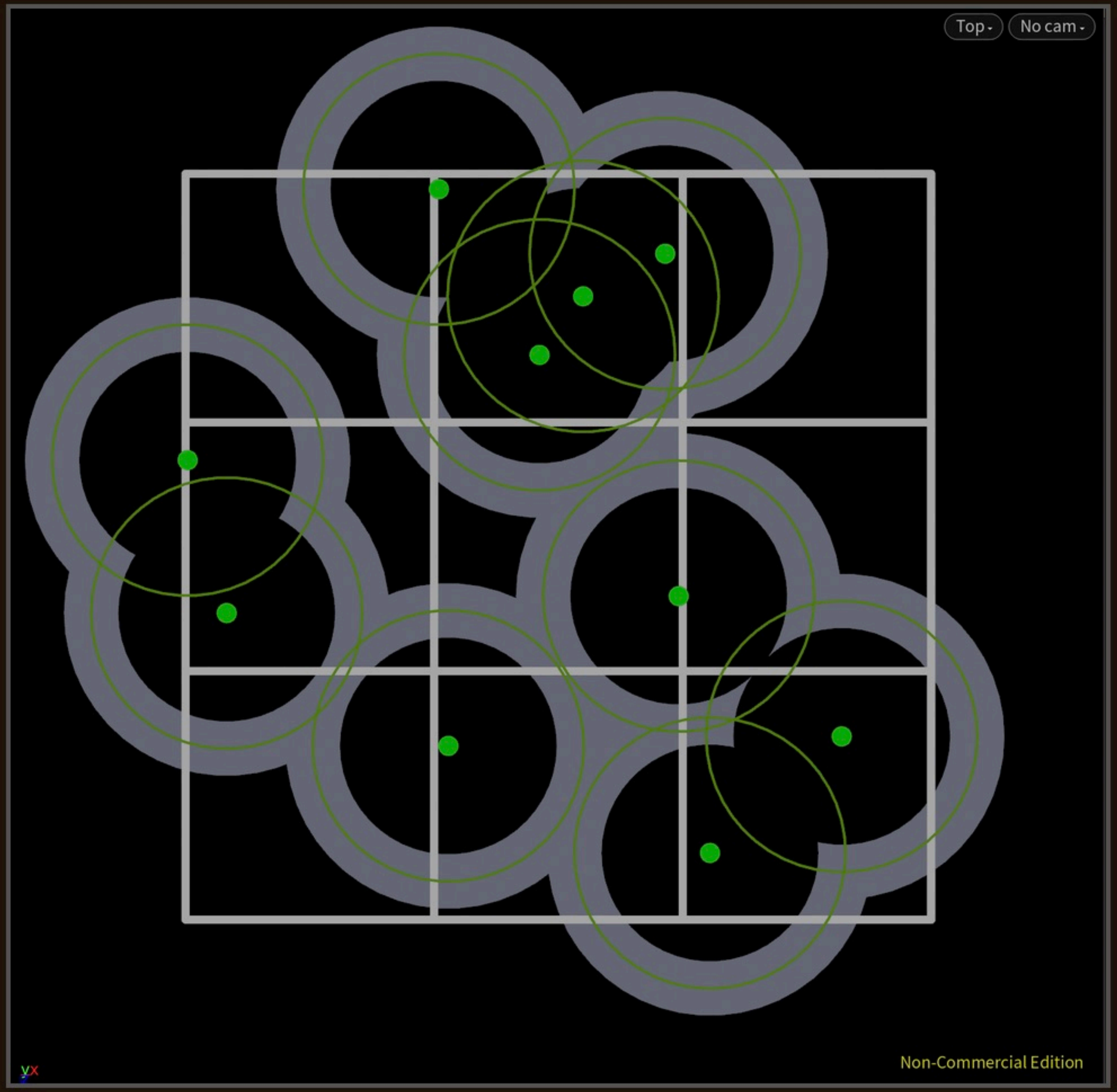
EXISTING TOOLS: SPHERES

signed distance field of an isolated particle x_0 with radius r_0:

$$\phi(x) = |x - x_0| - r_0$$



```
// the attribute containing the world space radius
std::string radius = "";

// the scale applied to every world space radius value
Real radiusScale = 1.0;

// the half band width of the generated surface.
Real halfband = LEVEL_SET_HALF_WIDTH;

// the target transform for the surface
math::Transform::Ptr transform = nullptr;

// list of attributes to transfer
std::vector<std::string> attributes;

// filter  a filter to apply to points
const FilterT* filter = nullptr;

// interrupter  optional interrupter
InterrupterT* interrupter = nullptr;
```



EXISTING TOOLS: SMOOTH SPHERES

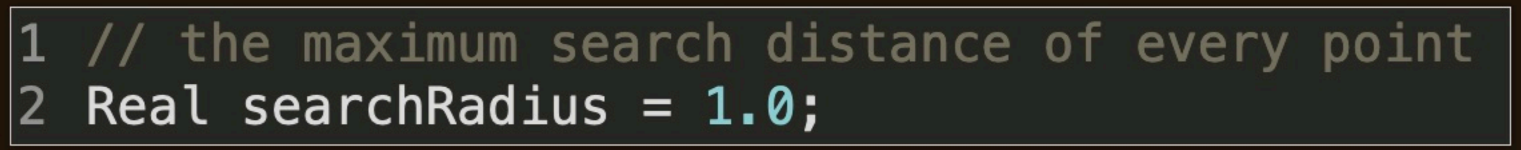
x_0 replaced by a weighted average of the nearby particle positions and r_0 replaced by a weighted average of their radii:

$$\phi(x) = |x - \bar{x}| - \bar{r} \tag{7}$$

$$\bar{x} = \sum_i w_i x_i \tag{8}$$

$$\bar{r} = \sum_i w_i r_i \tag{9}$$

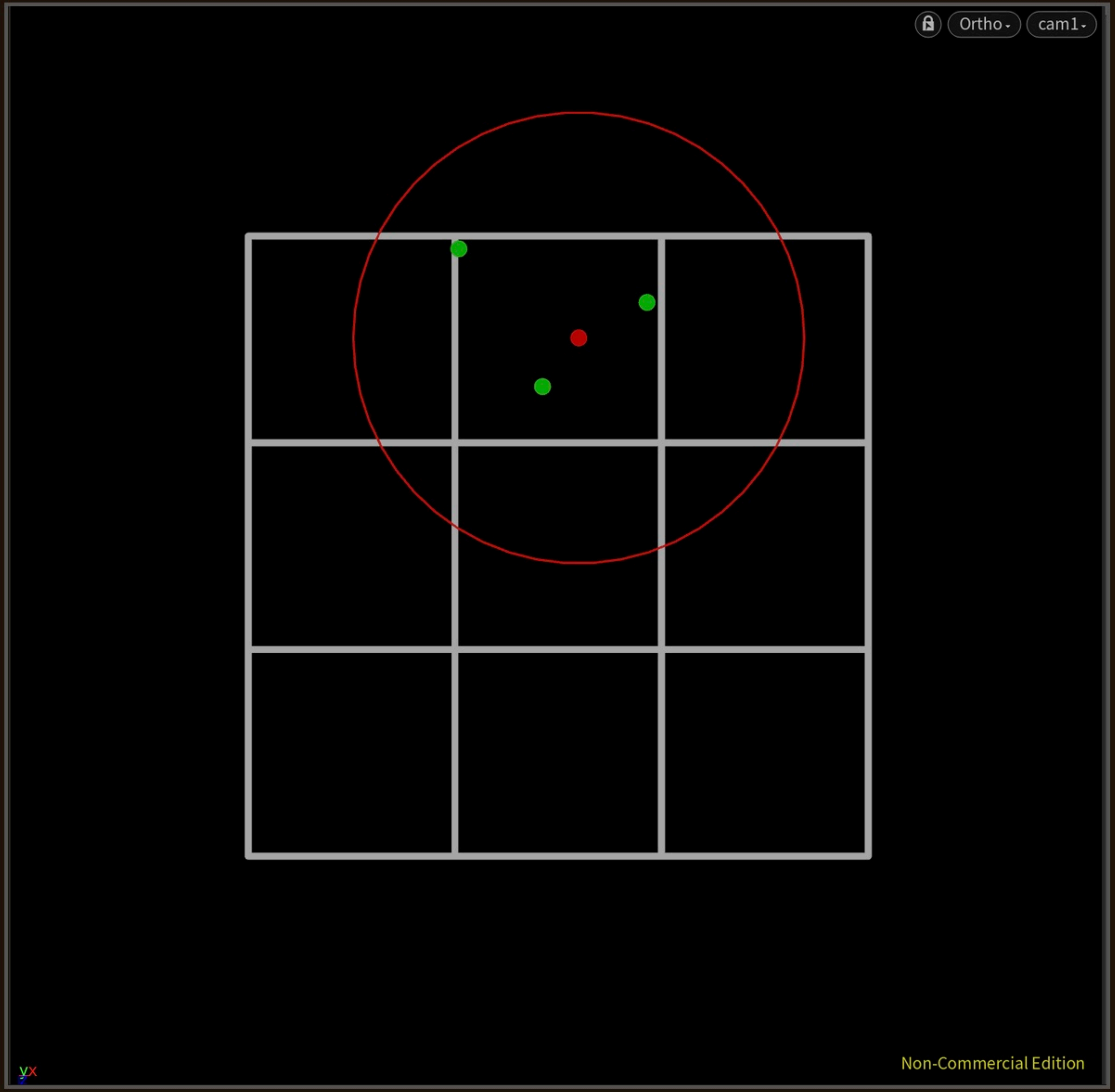
$$w_i = \frac{k(|x - x_i|/R)}{\sum_j k(|x - x_j|/R)} \tag{10}$$



```
// the maximum search distance of every point
Real searchRadius = 1.0;
```

$$k(s) = max(0, (1 - s^2)^3)$$

[Animating Sand as a Fluid - Zhu Bridson 05]



EXISTING TOOLS: SMOOTH SPHERES

x_0 replaced by a weighted average of the nearby particle positions and r_0 replaced by a weighted average of their radii:

$$\phi(x) = |x - \bar{x}| - \bar{r} \tag{7}$$

$$\bar{x} = \sum_i w_i x_i \tag{8}$$

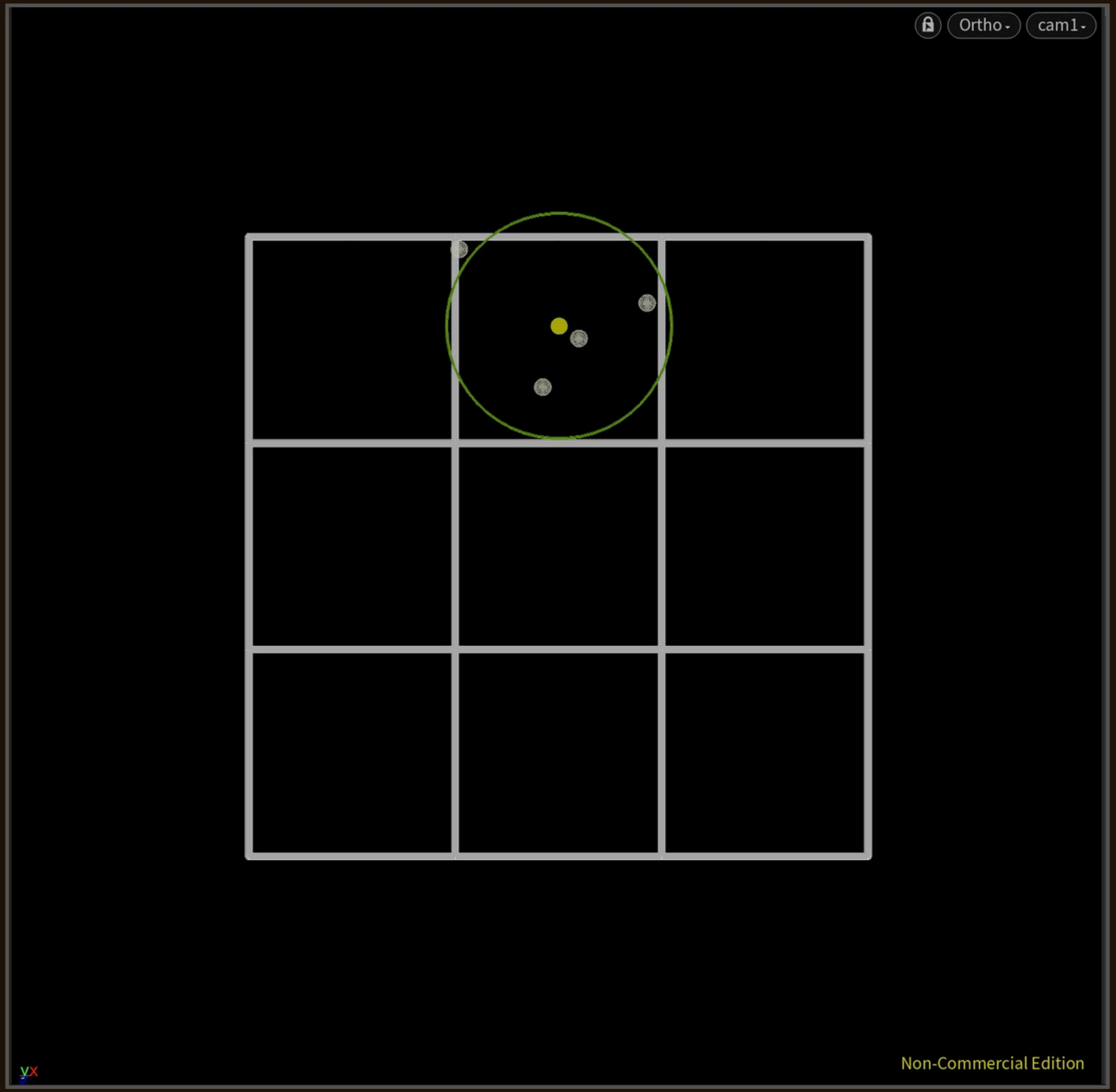
$$\bar{r} = \sum_i w_i r_i \tag{9}$$

$$w_i = \frac{k(|x - x_i|/R)}{\sum_j k(|x - x_j|/R)} \tag{10}$$

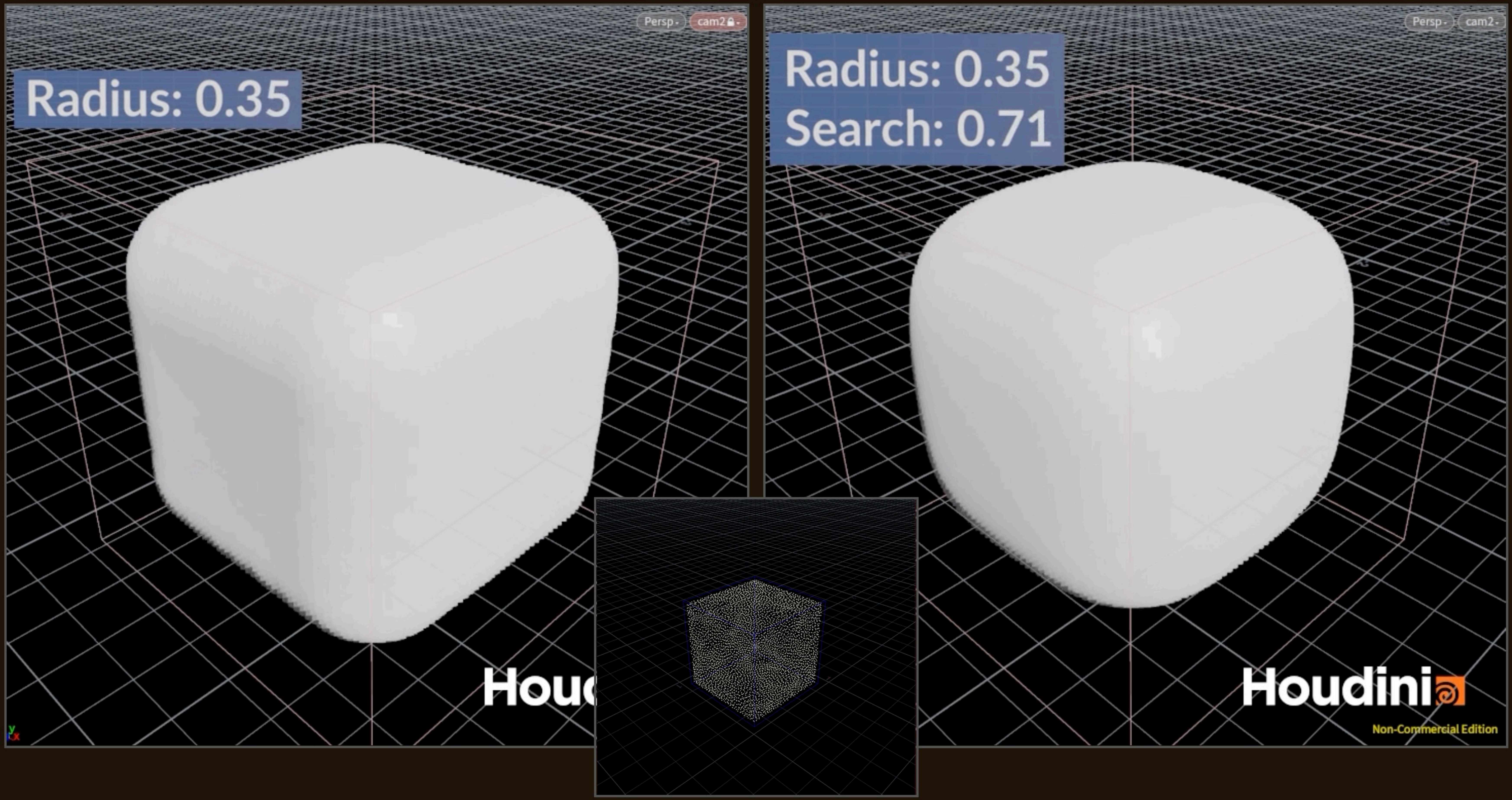
```
// the maximum search distance of every point
Real searchRadius = 1.0;
```

$$k(s) = max(0, (1 - s^2)^3)$$

[Animating Sand as a Fluid - Zhu Bridson 05]



EXISTING TOOLS: SMOOTH SPHERES



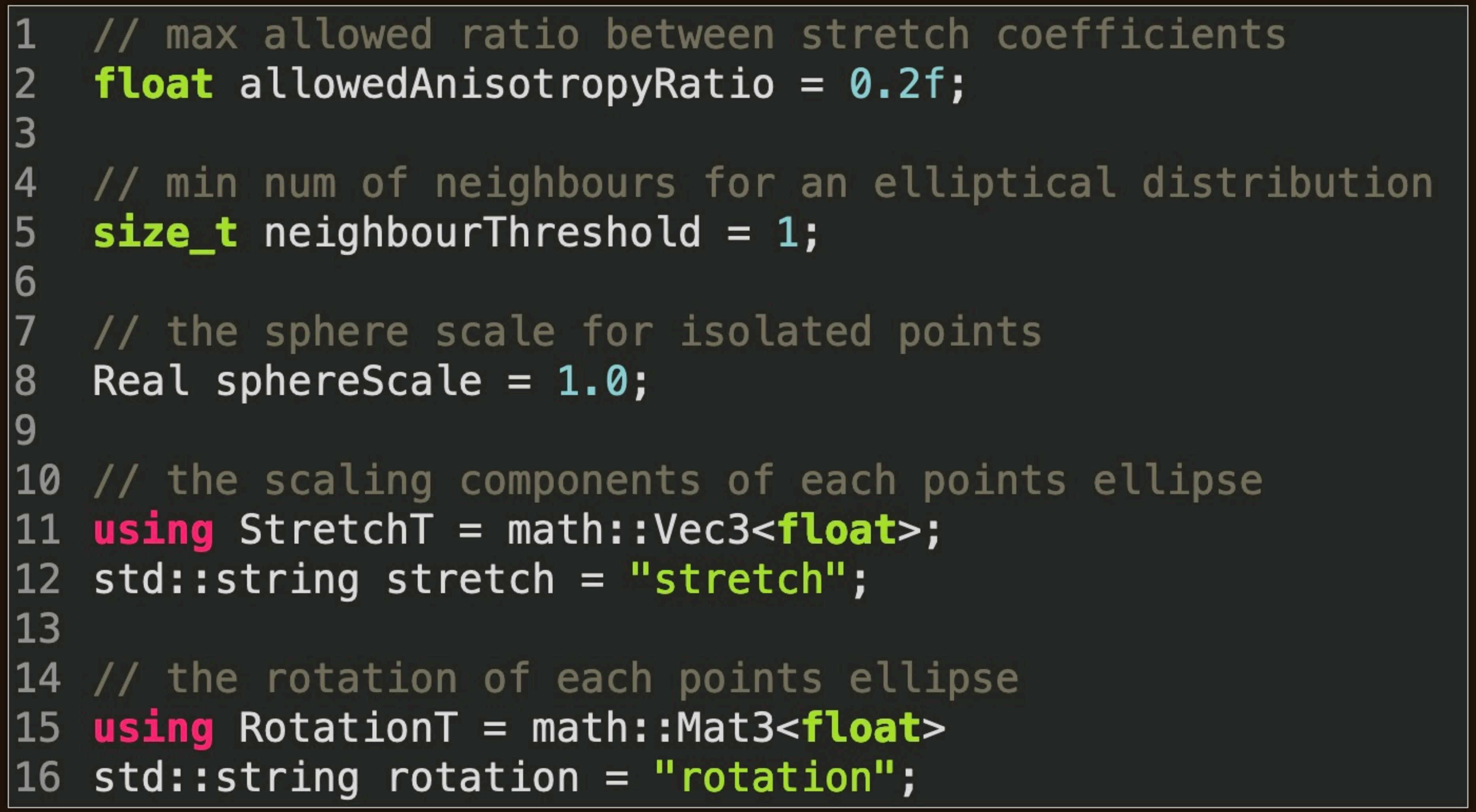
UPCOMING TOOLS: ELLIPSOIDS

determine an anisotropy matrix for each particle using weighted principal component analysis

$$C = \mathbf{R}\Sigma\mathbf{R}^T \tag{12}$$

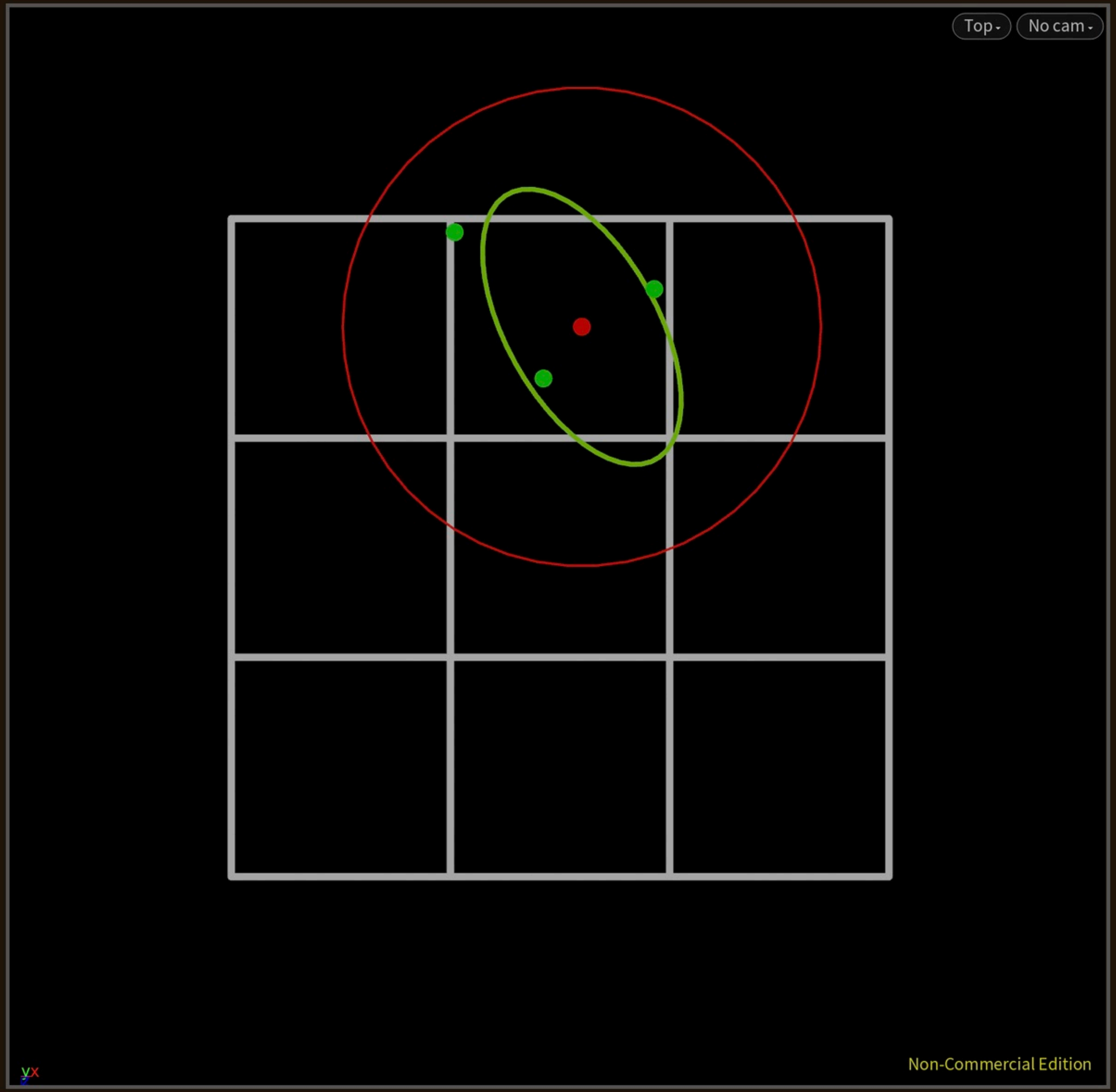
$$\Sigma = diag(\sigma_1, ..., \sigma_d) \tag{13}$$

$$\widetilde{\Sigma} = \begin{cases} diag(\sigma_1, \widetilde{\sigma}_2, ..., \widetilde{\sigma}_d), & \text{if } N > N_\epsilon, \\ \mathbf{I}, & \text{otherwise} \end{cases} \tag{15}$$



```
// max allowed ratio between stretch coefficients
float allowedAnisotropyRatio = 0.2f;

// min num of neighbours for an elliptical distribution
size_t neighbourThreshold = 1;

// the sphere scale for isolated points
Real sphereScale = 1.0;

// the scaling components of each points ellipse
using StretchT = math::Vec3<float>;
std::string stretch = "stretch";

// the rotation of each points ellipse
using RotationT = math::Mat3<float>
std::string rotation = "rotation";
```

[Reconstructing Fluid Surfaces with Anisotropic Kernels - Yu Turk 13]



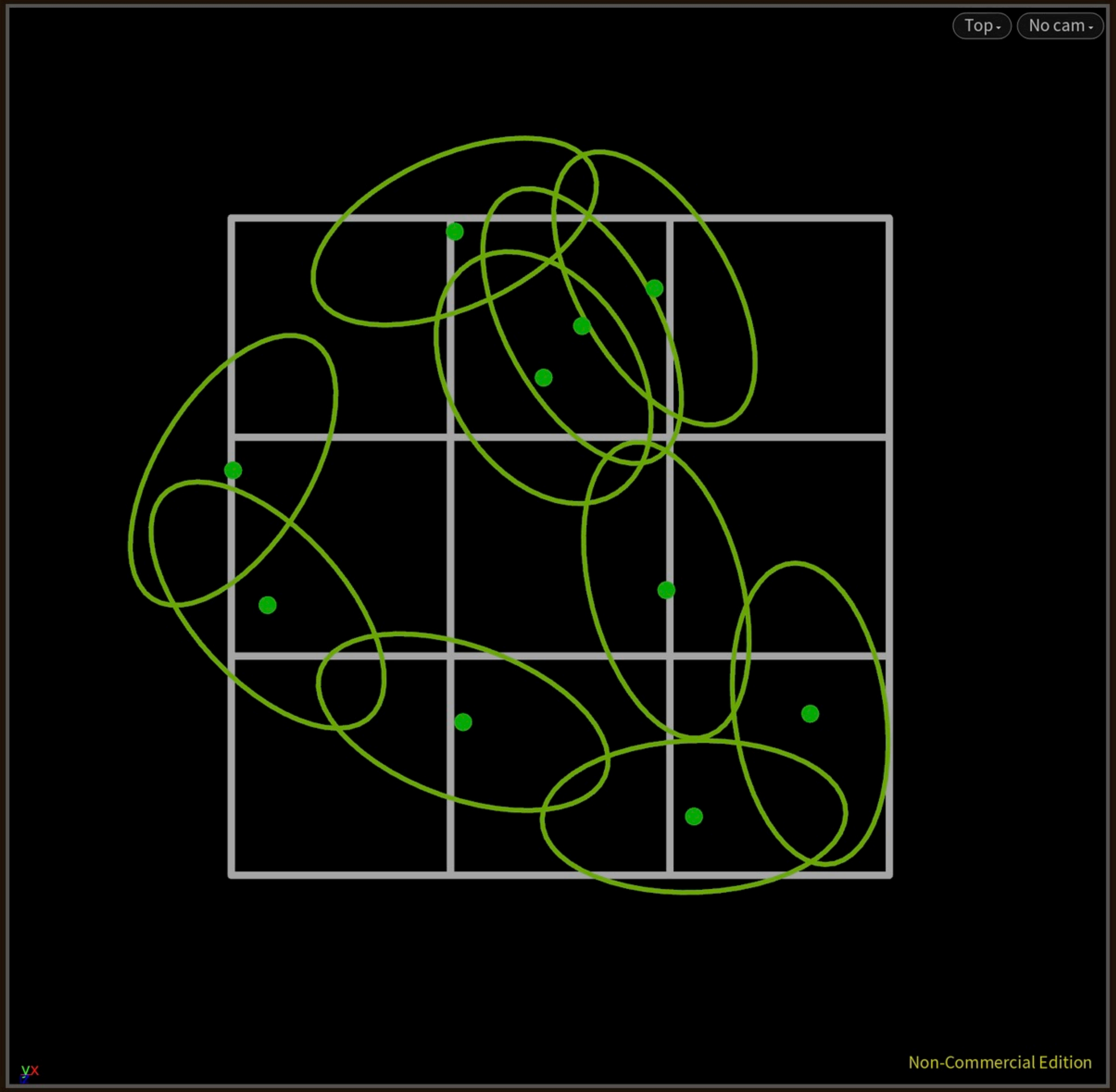
$N_\epsilon = 1$ and $\sigma_1 \geq k_r\sigma_d$ where $k_r = 5$

UPCOMING TOOLS: ELLIPSOIDS

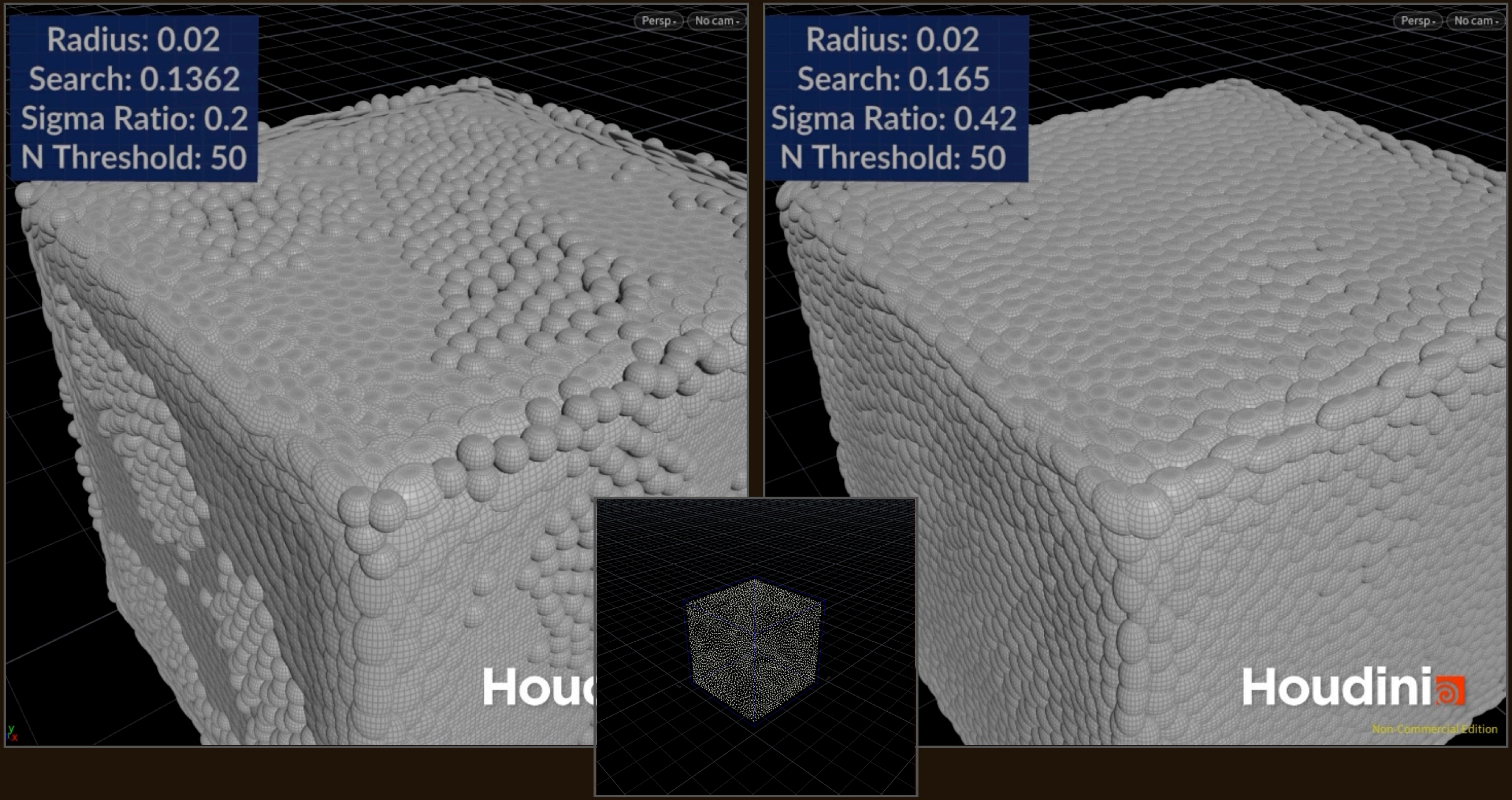
$$\mathbf{G}_i = \frac{1}{h_i} \mathbf{R} \widetilde{\Sigma}^{-1} \mathbf{R}^T \qquad (16)$$

Houdini VEX to deform spheres to ellipses

```
matrix3 rotation = 3@rotation; // R
vector inv_stretch = 1.0 / v@stretch; // Σ^-1

// center of the sphere the point belongs to
vector center = v@center;

// compute RΣ^-1 and store in "rotscale"
float data[] = set(rotation);
for (int i = 0; i < 9; ++i) {
    data[i] *= inv_stretch[i%3];
}
matrix3 rotscale = set(data);

// compute G = RΣ^-1R^T
matrix3 xform = (rotscale*transpose(rotation));

// to origin, xform by inverse G, xform back
v@P = ((v@P - v@center) * invert(xform)) + v@center;
```



UPCOMING TOOLS: ELLIPSOIDS



Radius: 0.02
Search: 0.1362
Sigma Ratio: 0.2
N Threshold: 50

Radius: 0.02
Search: 0.165
Sigma Ratio: 0.42
N Threshold: 50

SPHERES, SMOOTH, ELIPSOIDS

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

```
openvdb::points::rasterizeSdf(points, SETTINGS);
```

- Trivial narrow band sphere stamping

```
SETTINGS = openvdb::points::SphereSettings<>();
```

  - Extremely fast and efficient
  - Blobby, symmetrical narrow band level set

- Averaged position of influence

```
SETTINGS = openvdb::points::SmoothSphereSettings<>();
```

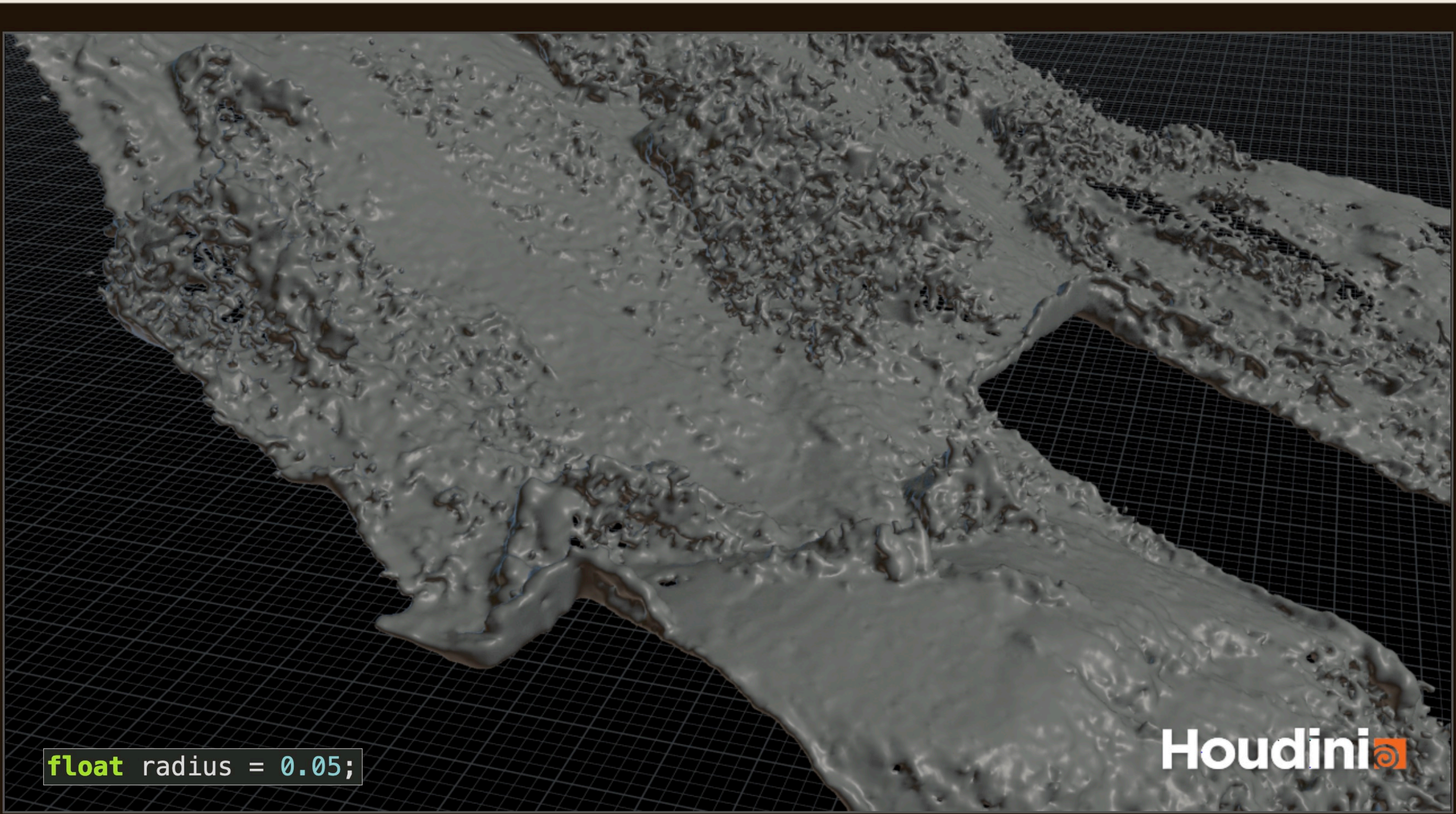
  - Smoother, more blended connections
  - More artistically pleasant surface
  - [Animating Sand as a Fluid - Zhu Bridson 05]

- Anisotropic surfacing

```
SETTINGS = openvdb::points::EllipsoidSettings<>();
```

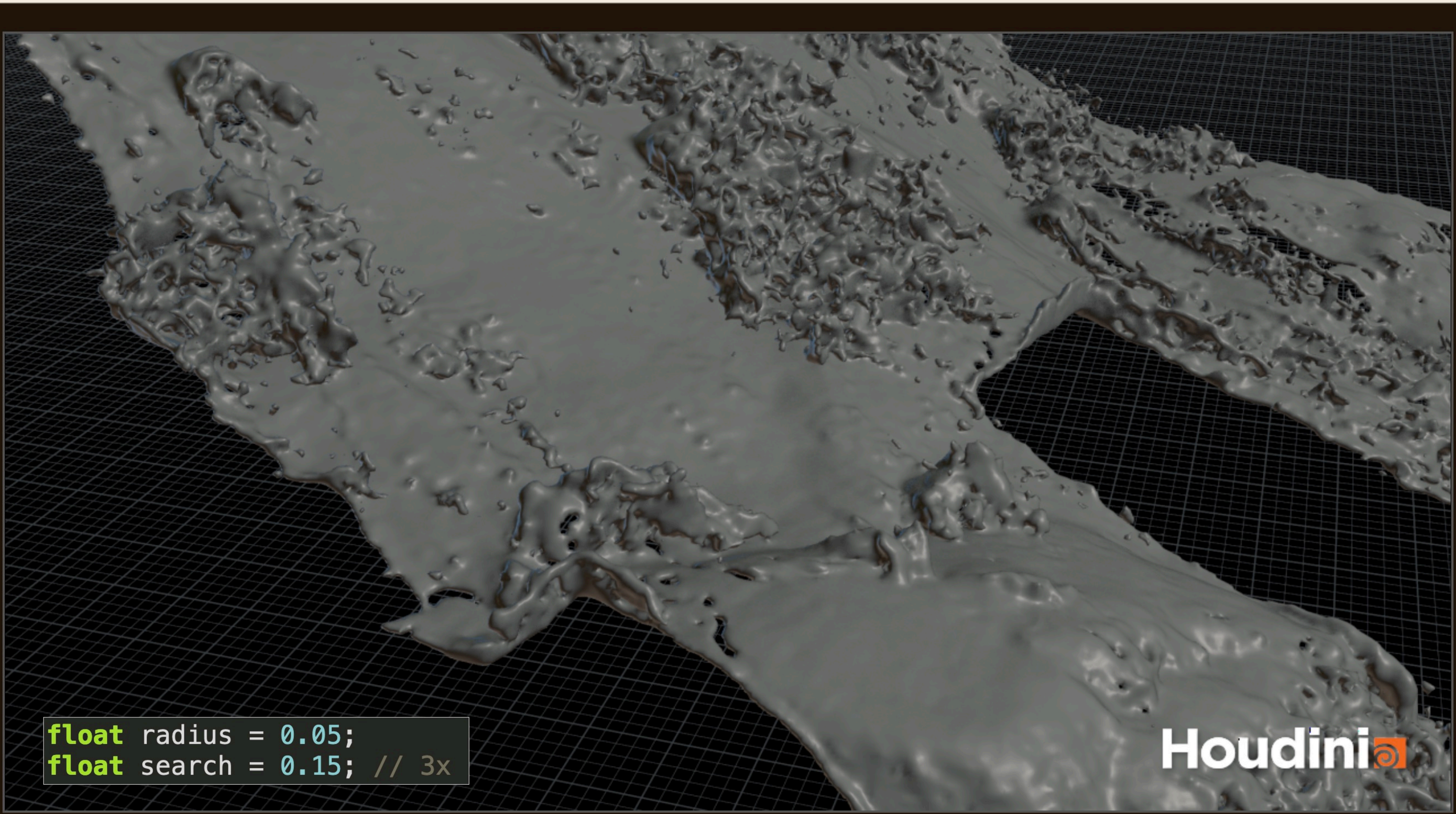
  - Ellipsoid transformations from local point distribution (PCA)
  - Captures smooth surfaces, thin streams and sharp features
  - May not necessarily produce a **symmetrical** narrow band level set
  - [Reconstructing Fluid Surfaces with Anisotropic Kernels - Yu Turk 13]

POINT RASTERIZE SDF: RESULTS



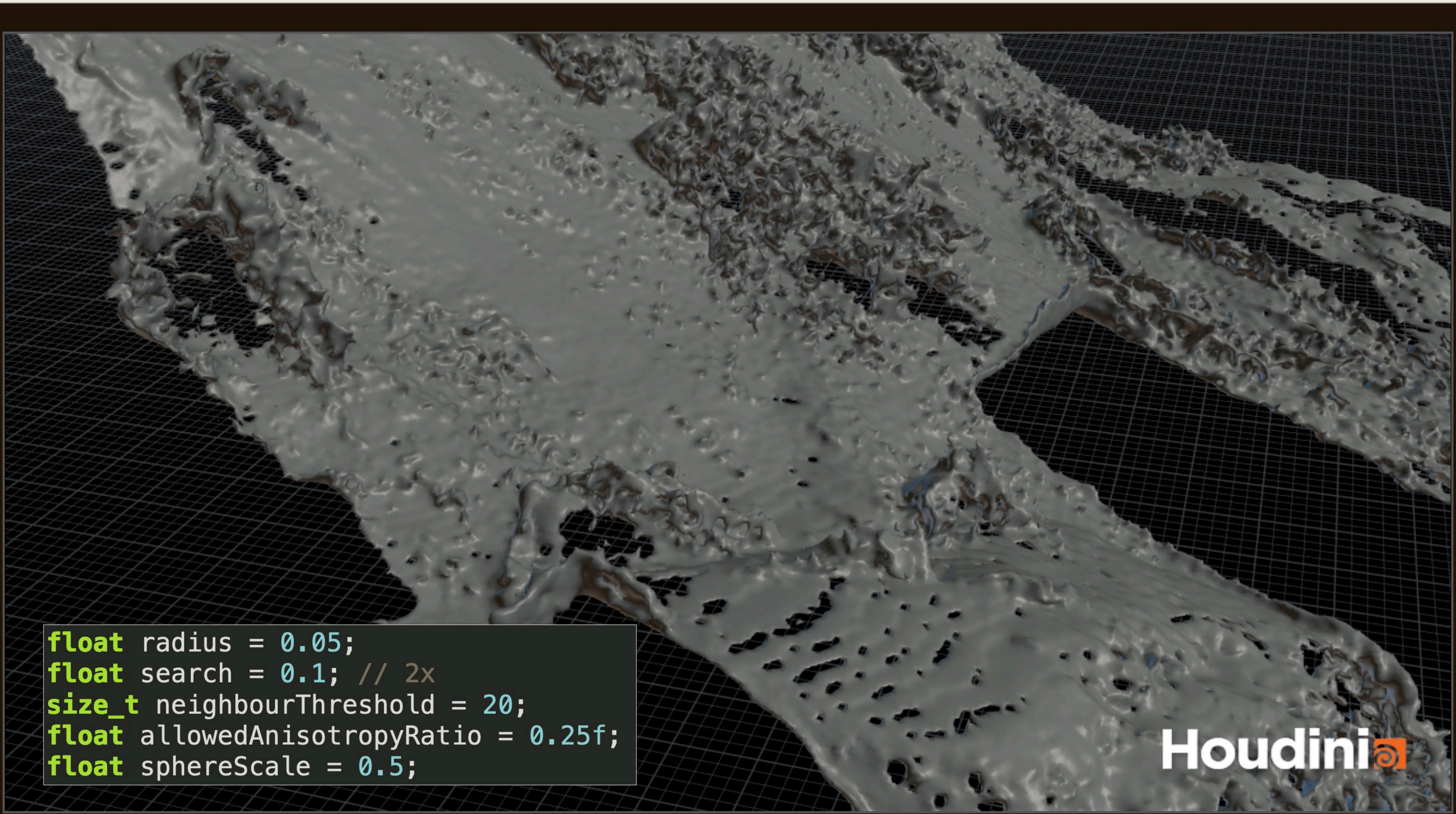
```
float radius = 0.05;
```

POINT RASTERIZE SDF: RESULTS



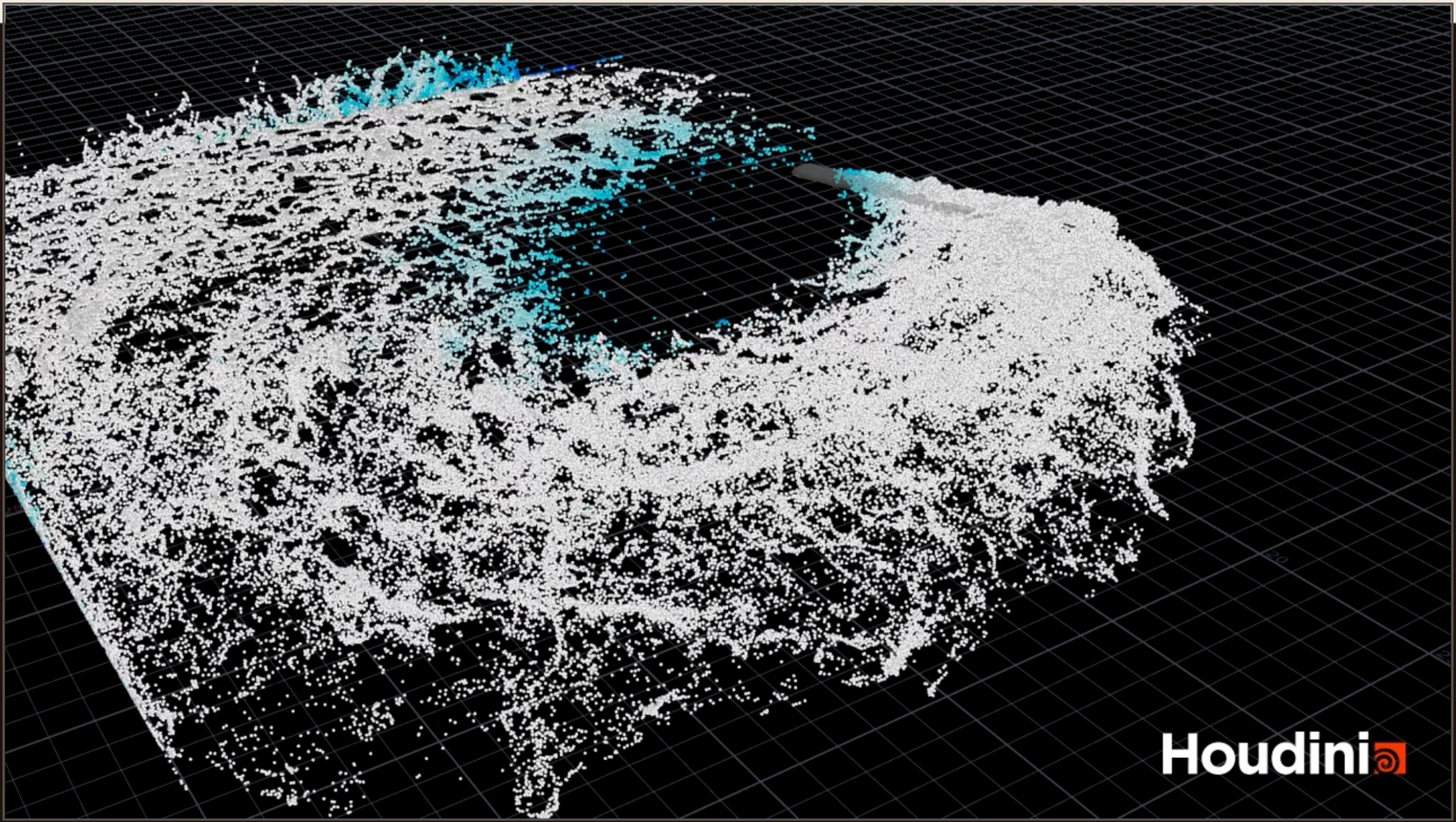
```
float radius = 0.05;
float search = 0.15; // 3x
```

POINT RASTERIZE SDF: RESULTS

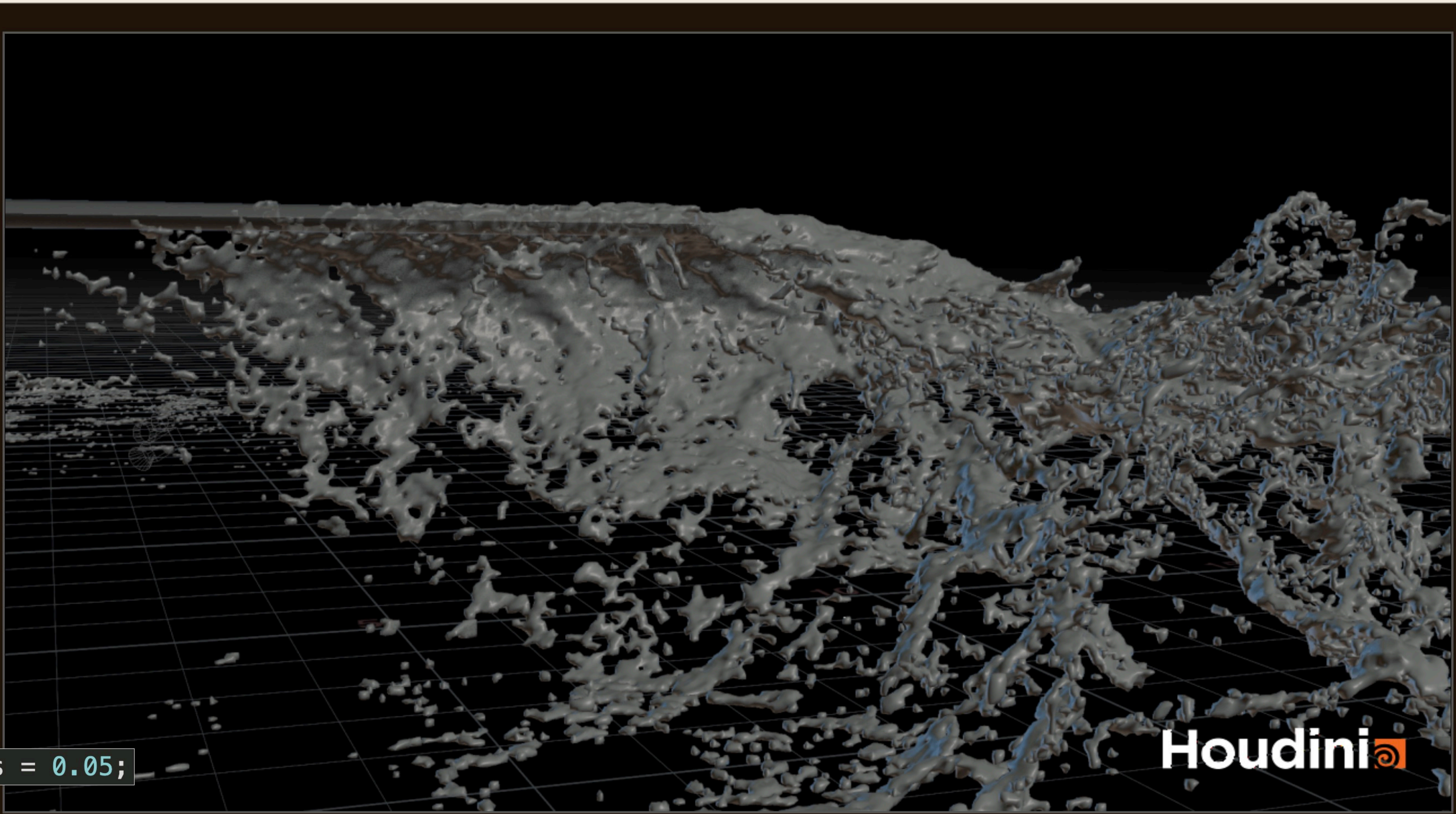


```
float radius = 0.05;
float search = 0.1; // 2x
size_t neighbourThreshold = 20;
float allowedAnisotropyRatio = 0.25f;
float sphereScale = 0.5;
```

POINT RASTERIZE SDF: RESULTS

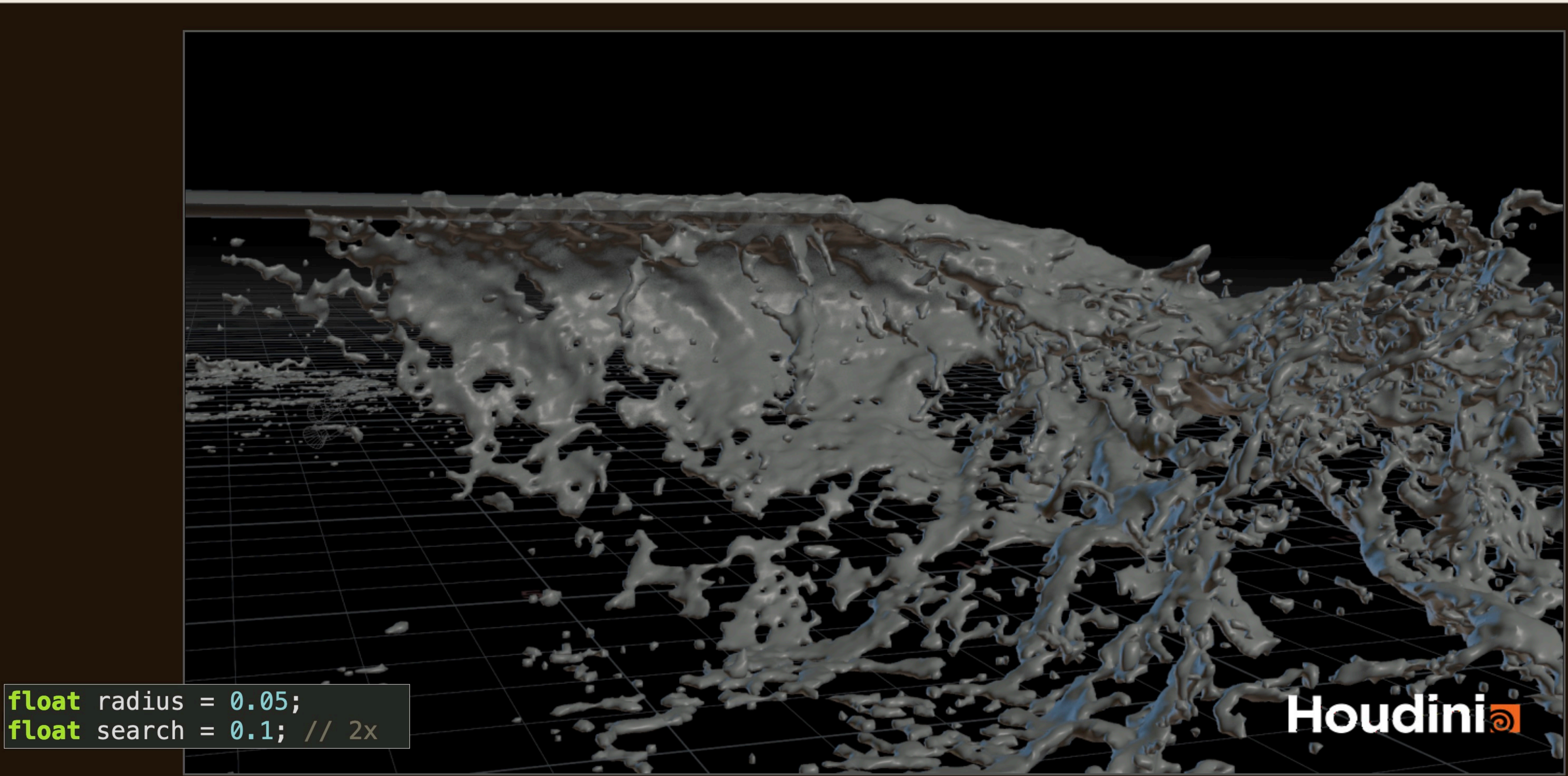


POINT RASTERIZE SDF: RESULTS



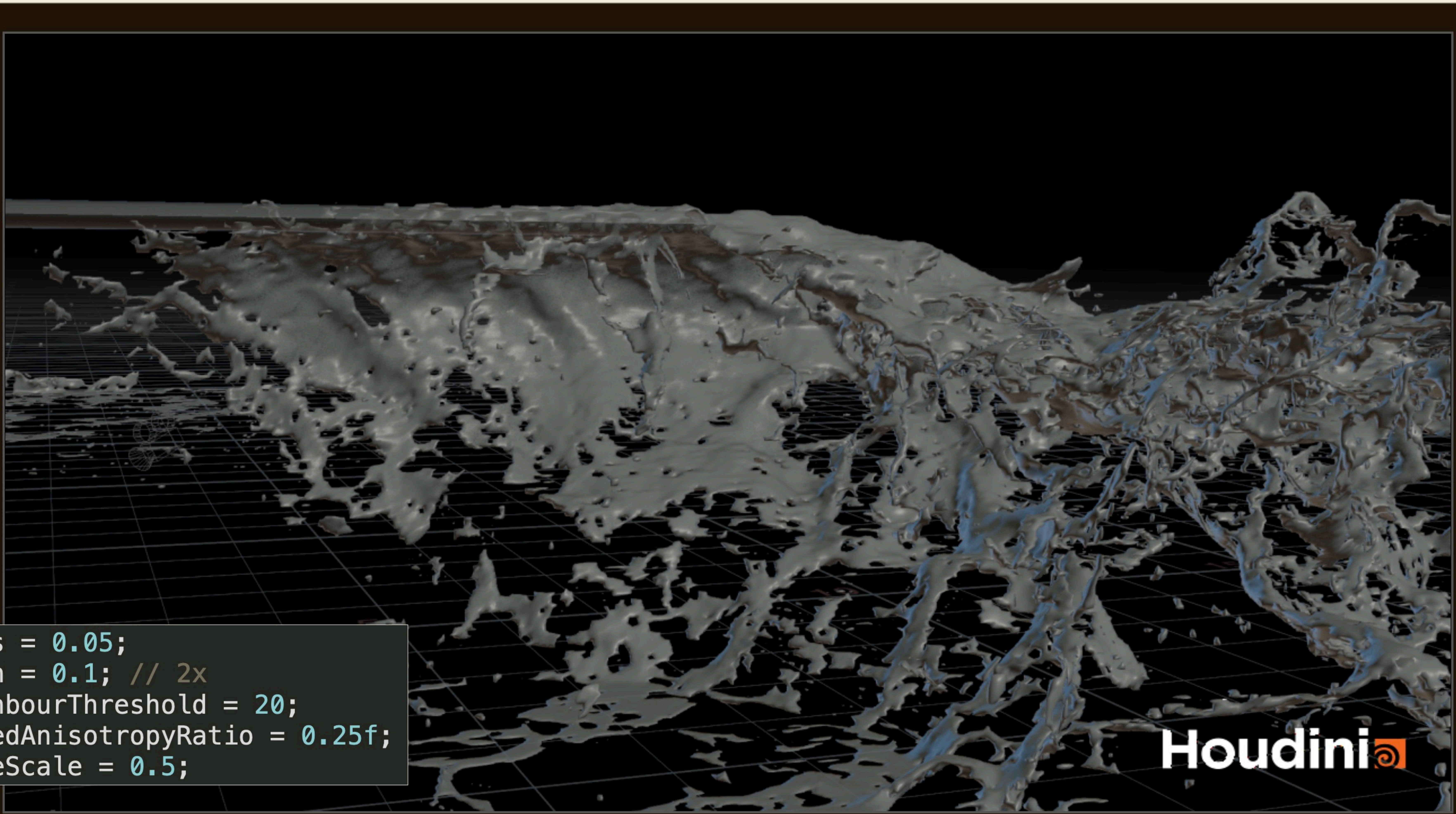
```
float radius = 0.05;
```

POINT RASTERIZE SDF: RESULTS



```
float radius = 0.05;
float search = 0.1; // 2x
```

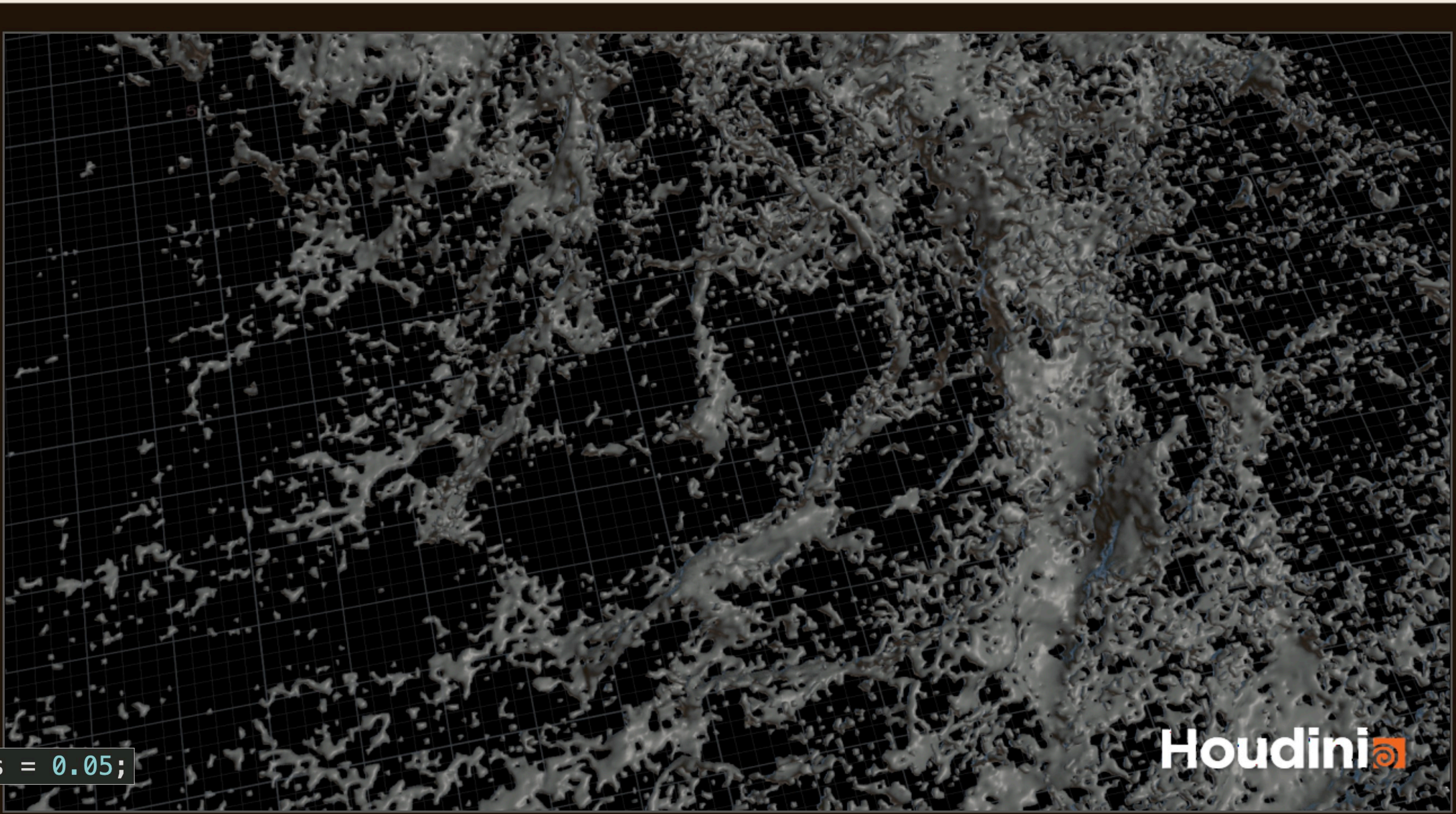
POINT RASTERIZE SDF: RESULTS



```
float radius = 0.05;
float search = 0.1; // 2x
size_t neighbourThreshold = 20;
float allowedAnisotropyRatio = 0.25f;
float sphereScale = 0.5;
```

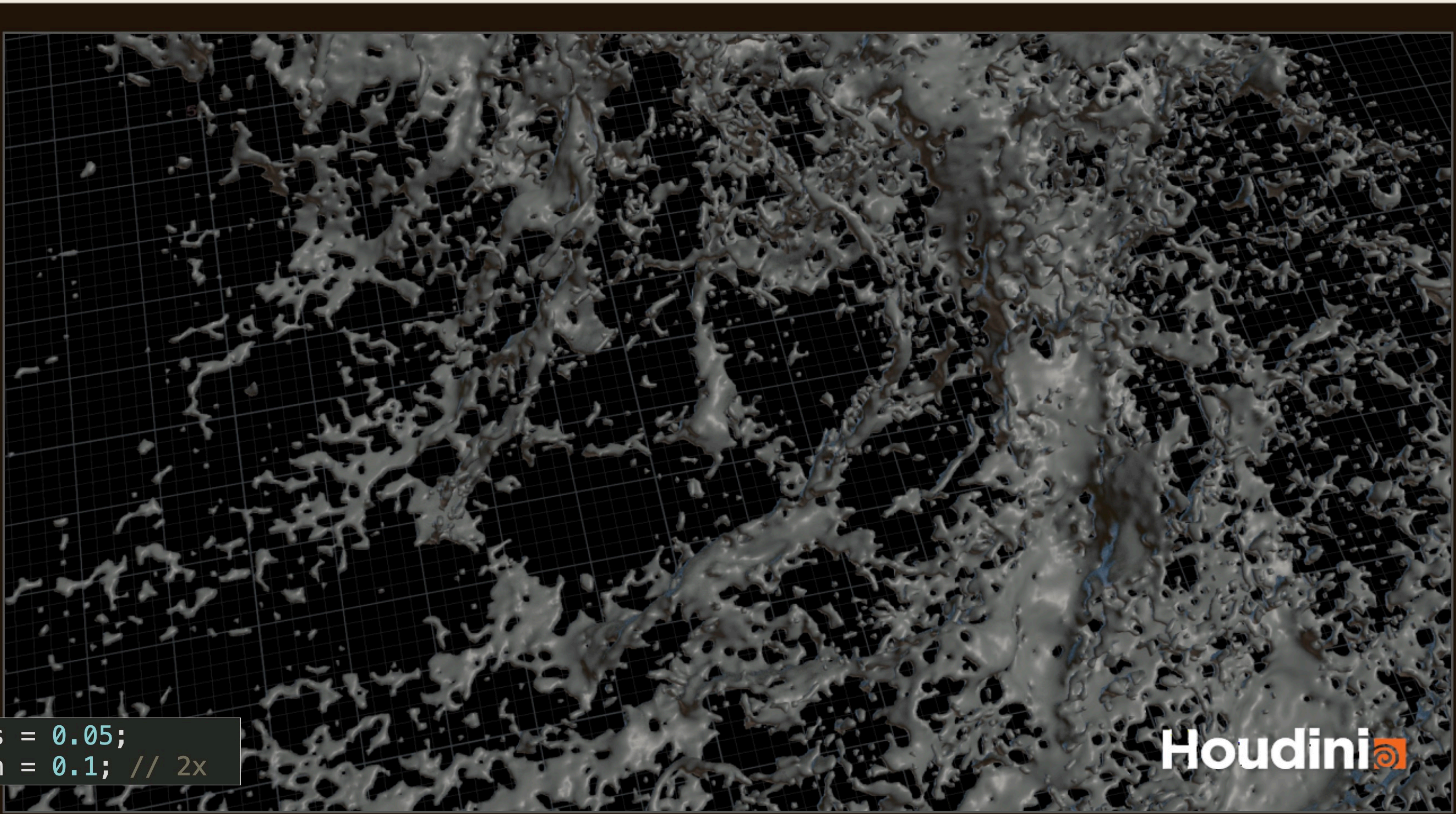
POINT RASTERIZE SDF: RESULTS

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG



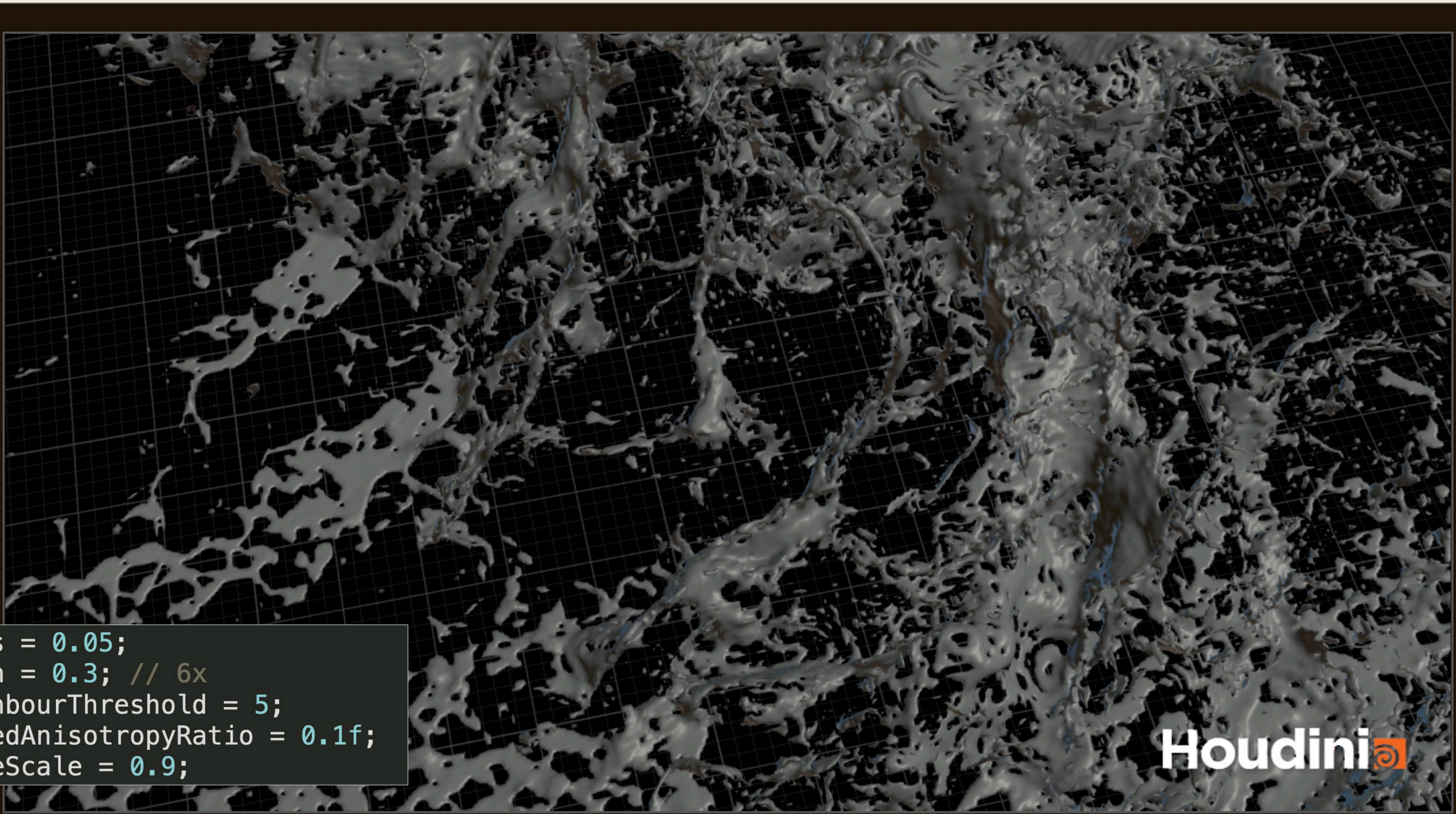
```
float radius = 0.05;
```

POINT RASTERIZE SDF: RESULTS



```
float radius = 0.05;
float search = 0.1; // 2x
```

POINT RASTERIZE SDF: RESULTS



```
float radius = 0.05;
float search = 0.3; // 6x
size_t neighbourThreshold = 5;
float allowedAnisotropyRatio = 0.1f;
float sphereScale = 0.9;
```

POINT RASTERIZE SDF: RESULTS

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

	Old	New			Anisotropic	
	v9.X P2LS	**Spheres**	**Average**	**Anisotropic**	**WPCA**	**Rasterization***
Performance	502ms	239ms	453ms	496ms	185ms	+ 311ms
Memory	266MB	254MB	265MB	342MB	121MB	→ 342MB

WPCA Search By Voxel Width

	1x	2x	4x	8x
Average Neighbour Count	11	43	176	757
Performance	96ms	185ms (1.9x)	363ms (3.8x)	1124ms (11.7x)

- 1mil points, $r = 0.05$, $search = 0.1$, $dx = 0.05$
- Apple M1 Max 10-Core
- *Not yet vectorized

POINT RASTERIZE SDF: RESULTS

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

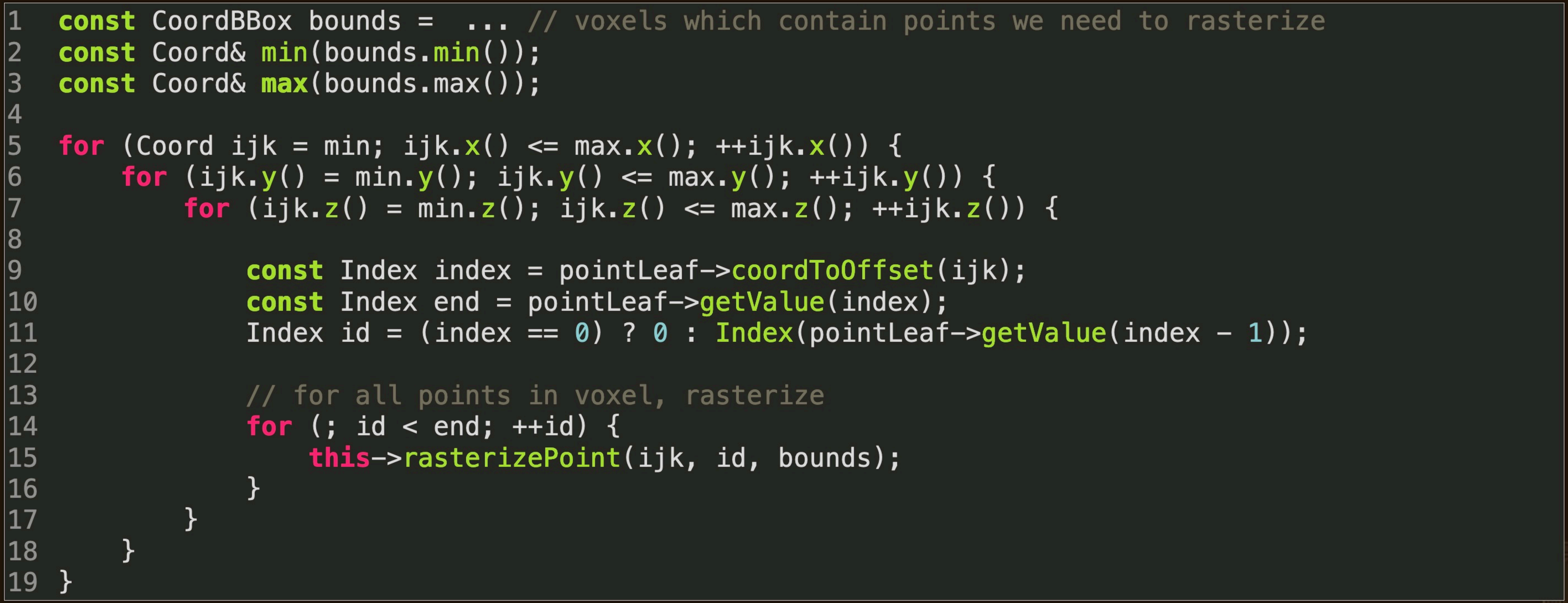
	Old	New			Anisotropic	
	v9.X P2LS	Spheres	Average	Anisotropic	WPCA	Rasterization*
Performance	502ms	239ms	453ms	496ms	185ms	+ 311ms
Memory	266MB	254MB	265MB	342MB	121MB	→ 342MB

WPCA Search By Voxel Width

	1x	2x	4x	8x
Average Neighbour Count	11	43	176	757
Performance	96ms	185ms (1.9x)	363ms (3.8x)	1124ms (11.7x)

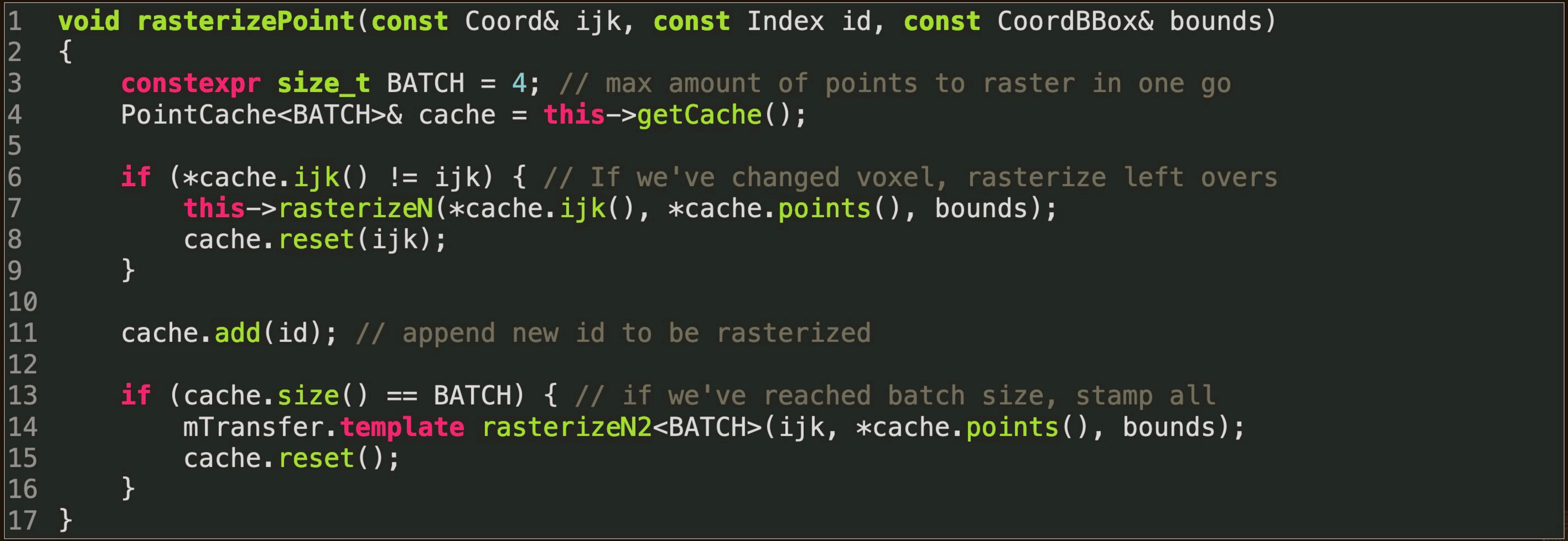
- 1mil points, $r = 0.05$, $search = 0.1$, $dx = 0.05$
- Apple M1 Max 10-Core
- *Not yet vectorized

POINT RASTERIZE SDF: SIMD

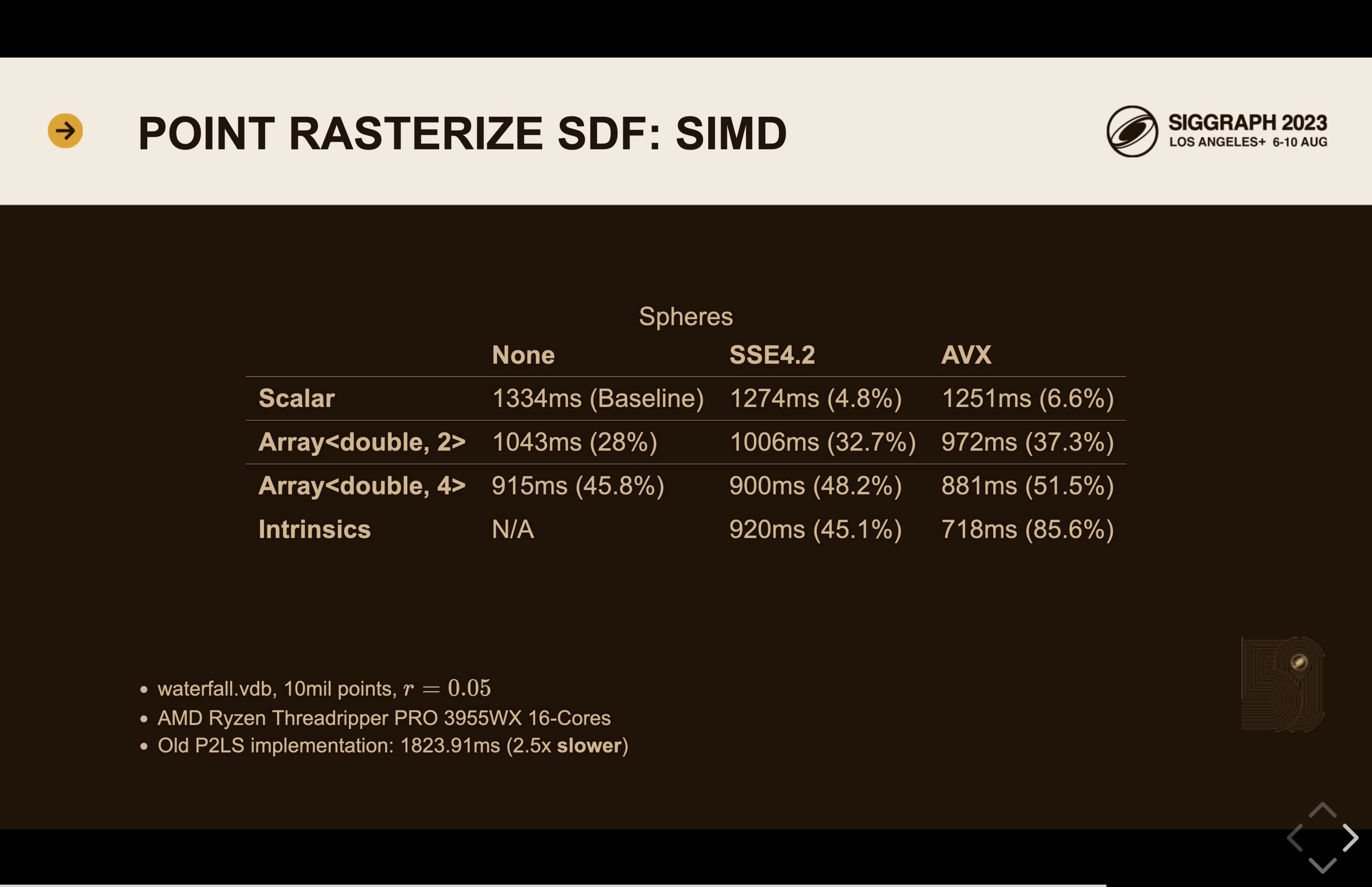


```cpp
const CoordBBox bounds =  ... // voxels which contain points we need to rasterize
const Coord& min(bounds.min());
const Coord& max(bounds.max());

for (Coord ijk = min; ijk.x() <= max.x(); ++ijk.x()) {
    for (ijk.y() = min.y(); ijk.y() <= max.y(); ++ijk.y()) {
        for (ijk.z() = min.z(); ijk.z() <= max.z(); ++ijk.z()) {

            const Index index = pointLeaf->coordToOffset(ijk);
            const Index end = pointLeaf->getValue(index);
            Index id = (index == 0) ? 0 : Index(pointLeaf->getValue(index - 1));

            // for all points in voxel, rasterize
            for (; id < end; ++id) {
                this->rasterizePoint(ijk, id, bounds);
            }
        }
    }
}
```

POINT RASTERIZE SDF: SIMD



```cpp
void rasterizePoint(const Coord& ijk, const Index id, const CoordBBox& bounds)
{
    constexpr size_t BATCH = 4; // max amount of points to raster in one go
    PointCache<BATCH>& cache = this->getCache();

    if (*cache.ijk() != ijk) { // If we've changed voxel, rasterize left overs
        this->rasterizeN(*cache.ijk(), *cache.points(), bounds);
        cache.reset(ijk);
    }

    cache.add(id); // append new id to be rasterized

    if (cache.size() == BATCH) { // if we've reached batch size, stamp all
        mTransfer.template rasterizeN2<BATCH>(ijk, *cache.points(), bounds);
        cache.reset();
    }
}
```



POINT RASTERIZE SDF: SIMD

Spheres

	None	SSE4.2	AVX
Scalar	1334ms (Baseline)	1274ms (4.8%)	1251ms (6.6%)
Array<double, 2>	1043ms (28%)	1006ms (32.7%)	972ms (37.3%)
Array<double, 4>	915ms (45.8%)	900ms (48.2%)	881ms (51.5%)
Intrinsics	N/A	920ms (45.1%)	718ms (85.6%)

- waterfall.vdb, 10mil points, $r = 0.05$
- AMD Ryzen Threadripper PRO 3955WX 16-Cores
- Old P2LS implementation: 1823.91ms (2.5x **slower**)

POINT RASTERIZE SDF: SIMD

Spheres

	None	SSE4.2	AVX
Scalar	1334ms (Baseline)	1274ms (4.8%)	1251ms (6.6%)
Array<double, 2>	1043ms (28%)	1006ms (32.7%)	972ms (37.3%)
Array<double, 4>	915ms (45.8%)	900ms (48.2%)	881ms (51.5%)
Intrinsics	N/A	920ms (45.1%)	718ms (85.6%)

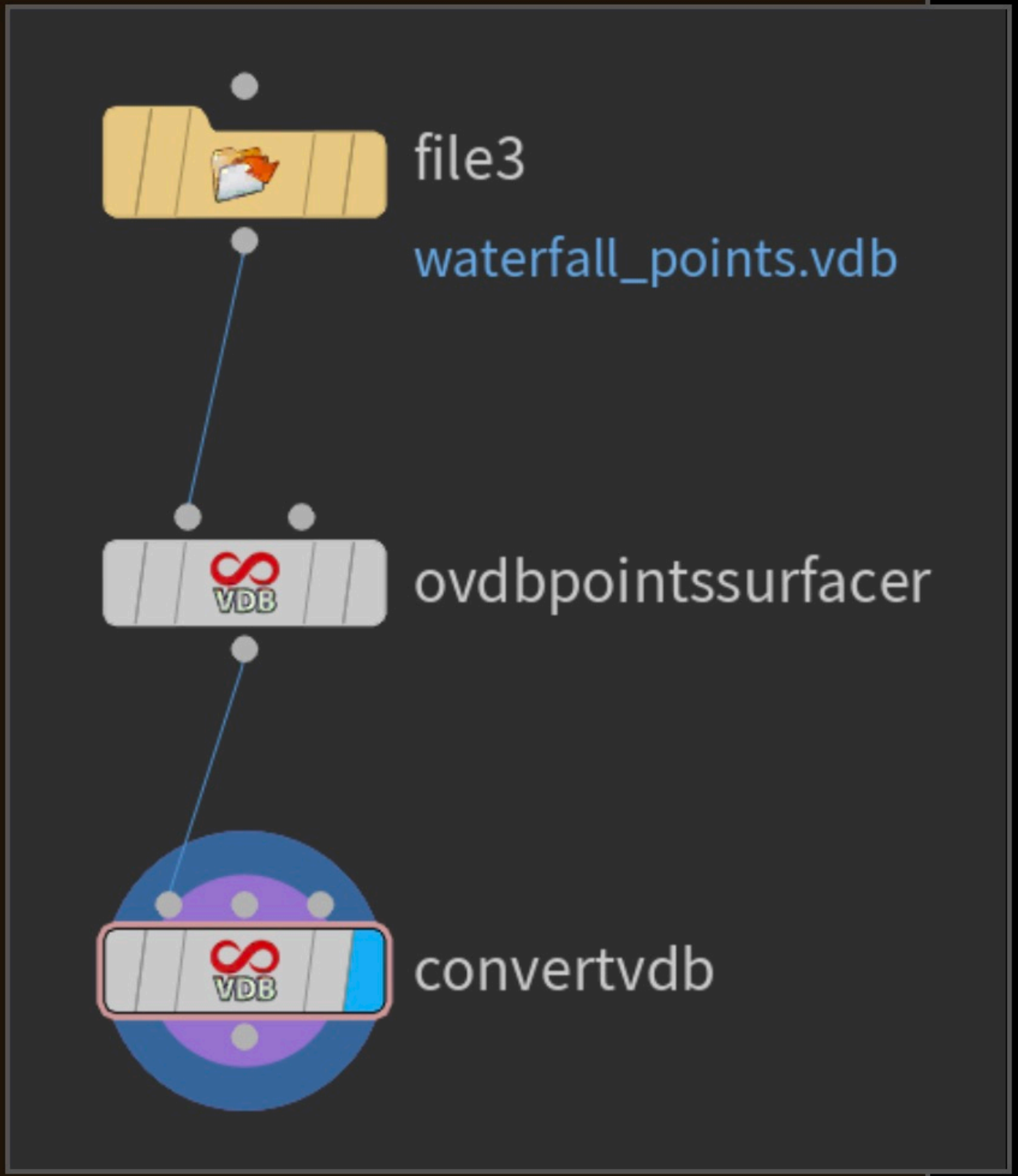
- waterfall.vdb, 10mil points, $r = 0.05$
- AMD Ryzen Threadripper PRO 3955WX 16-Cores
- Old P2LS implementation: 1823.91ms (2.5x **slower**)

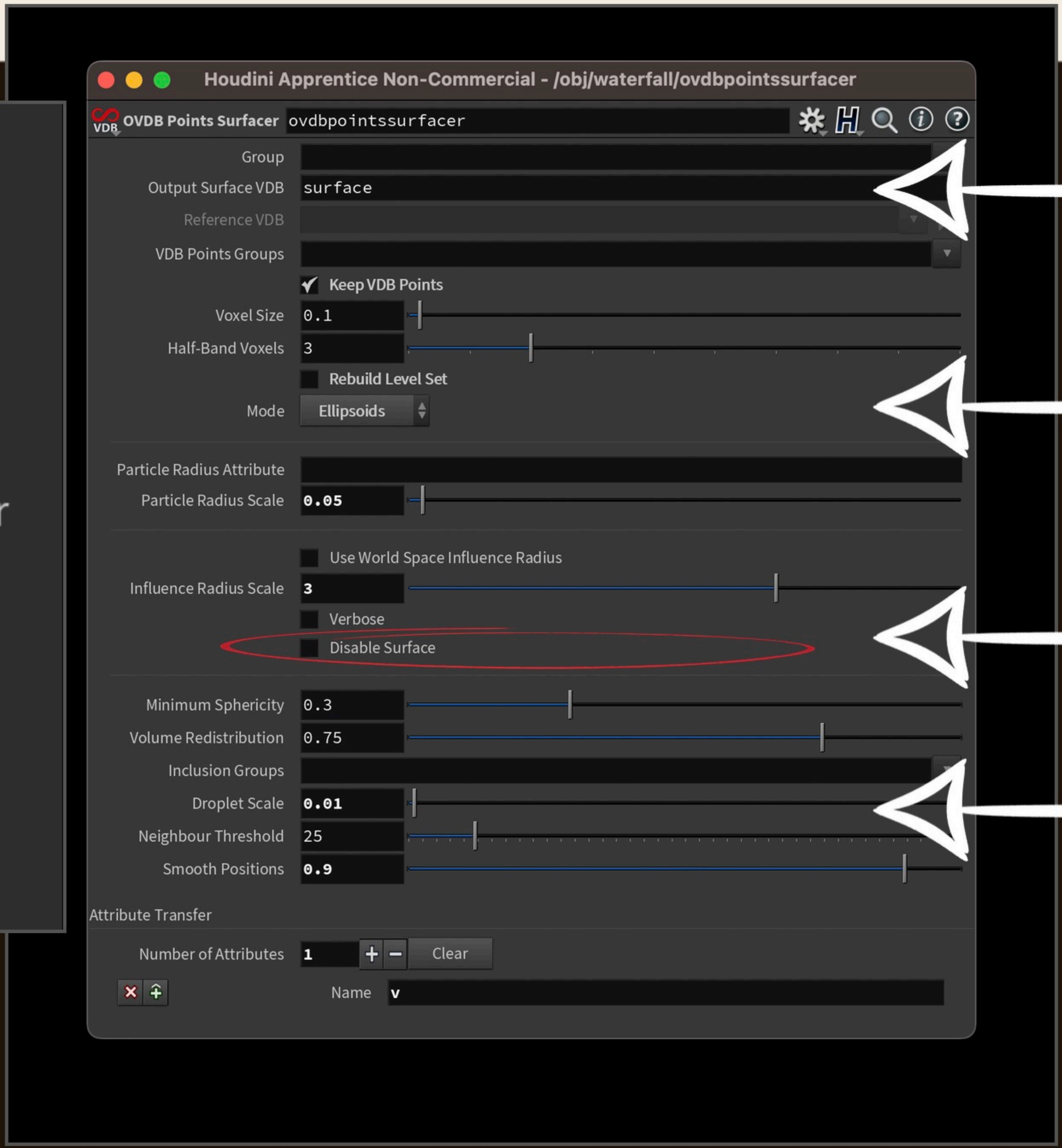
THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

OPENVDB AX
FUTURE WORKFLOWS

OPENVDB AX: FUTURE WORKFLOWS





Houdini Apprentice Non-Commercial - /obj/waterfall/ovdbpointssurfacer

OVDB Points Surfacer ovdbpointssurfacer

Group
Output Surface VDB surface
Reference VDB
VDB Points Groups
☑ Keep VDB Points
Voxel Size 0.1
Half-Band Voxels 3
□ Rebuild Level Set
Mode Ellipsoids

Particle Radius Attribute
Particle Radius Scale 0.05

□ Use World Space Influence Radius
Influence Radius Scale 3
□ Verbose
□ Disable Surface

Minimum Sphericity 0.3
Volume Redistribution 0.75
Inclusion Groups
Droplet Scale 0.01
Neighbour Threshold 25
Smooth Positions 0.9

Attribute Transfer

Number of Attributes 1 + − Clear
Name v

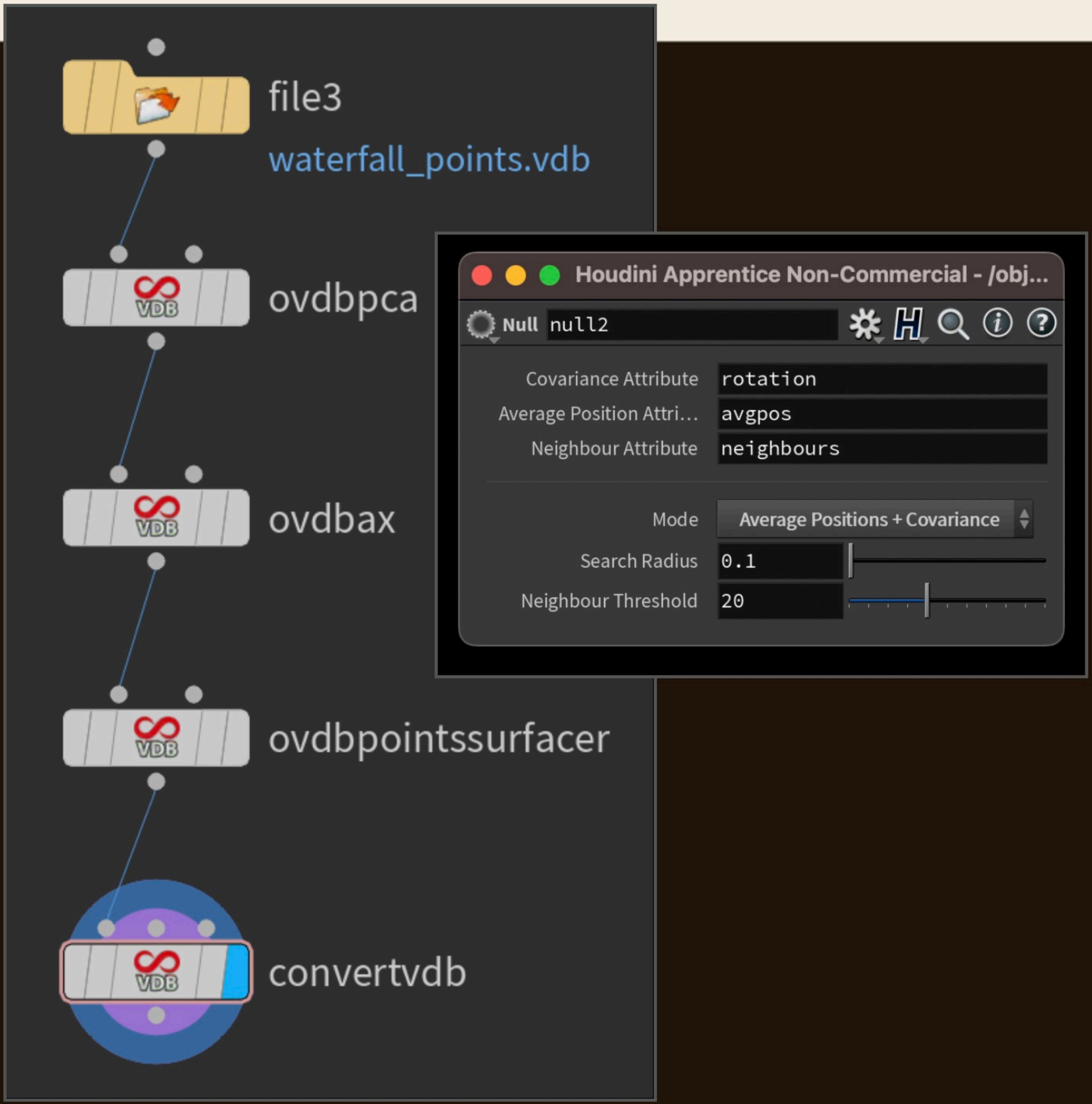
Input Primitive Names/Groups

Mode

"Disable Surface"?

PCA/Ellips Settings

OPENVDB AX: FUTURE WORKFLOWS



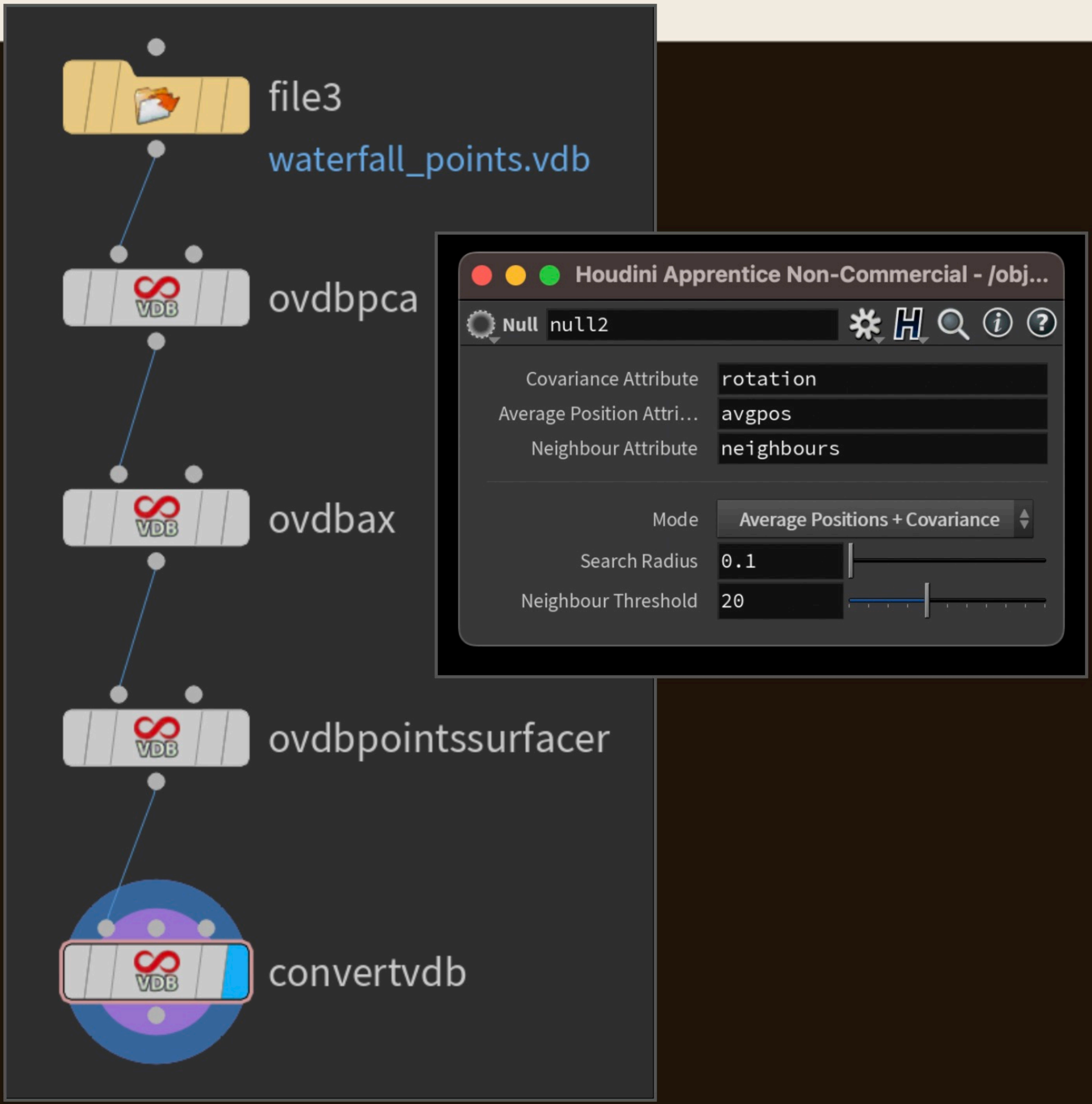
null2

Covariance Attribute: rotation
Average Position Attri...: avgpos
Neighbour Attribute: neighbours
Mode: Average Positions + Covariance
Search Radius: 0.1
Neighbour Threshold: 20

Compute transformations and blend isolated droplets

```
mat3f rot = identity3();
vec3f stretch;

if (!ingroup("ellipses"))
{
    // not marked as having valid elliptical distribution
    stretch = ch("droplet_scale");
    int max_neighbours = ch("max_neighbours");
    // scxale droplet size by neighbour count
    stretch *= (float(i@neighbours) / float(max_neighbours));
}
else
{
    // singular value decomposition of covariance
    decomposeSymmetricMatrix(mat3f@covariance, rot, stretch);
    stretch = reverse(sort(stretch)); // descending order

    // clamp anisotropy
    float ms = ch("min_sphericity");
    float max_allowed_stretch = stretch[0] * ms;
    stretch[1] = max(stretch[1], max_allowed_stretch);
    stretch[2] = max(stretch[2], max_allowed_stretch);

    // normalise the principal lengths
    stretch *= 1.0f / cbrt(product(stretch));
}

// store results
vec3f@stretch = stretch;
mat3f@rotation = rot;
```

OPENVDB AX: FUTURE WORKFLOWS



Null null2

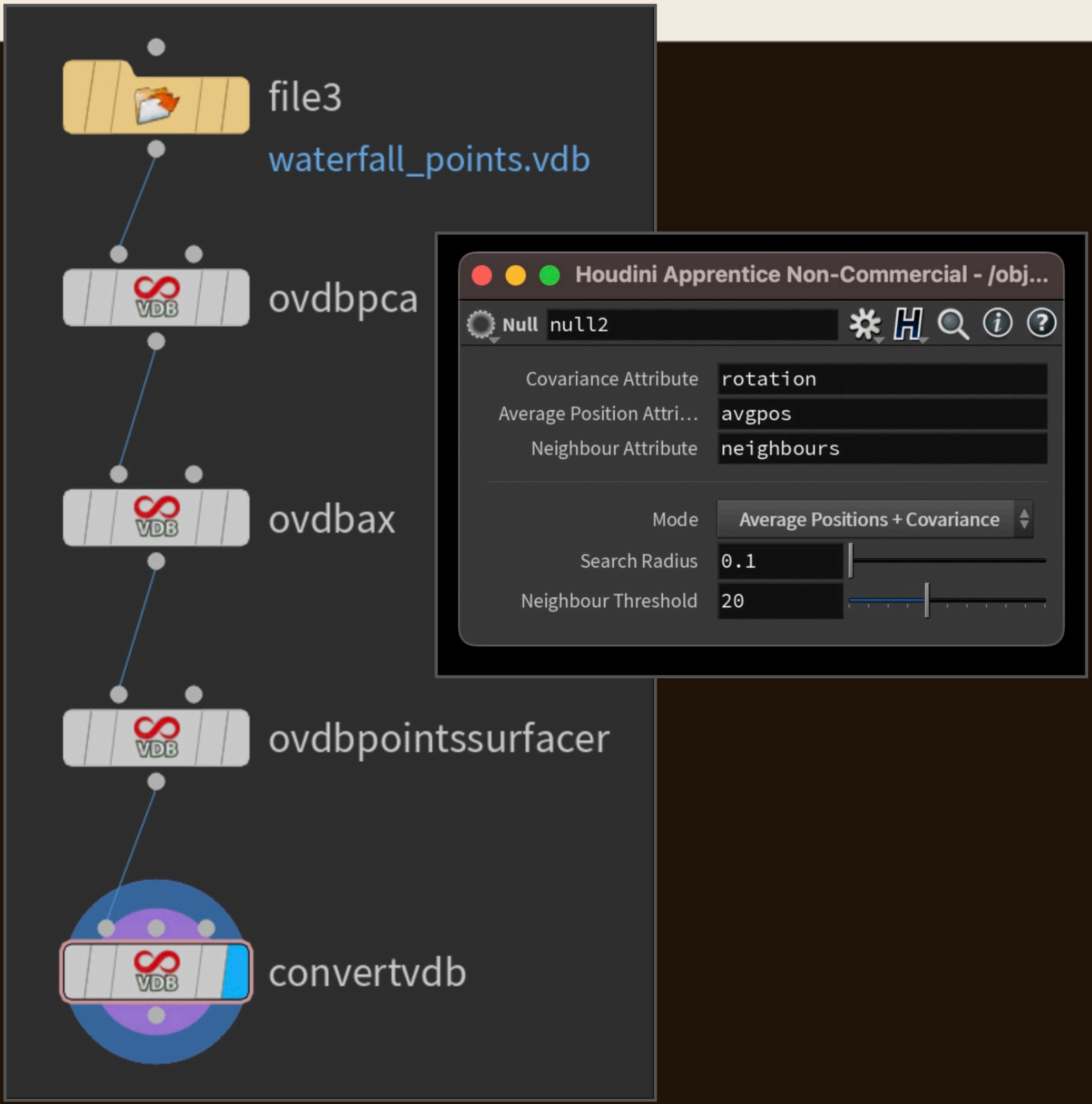
Covariance Attribute rotation
Average Position Attri... avgpos
Neighbour Attribute neighbours

Mode Average Positions + Covariance
Search Radius 0.1
Neighbour Threshold 20

Weighted blend local/global volume preservation

```
float weight = ch("average_pos_weight");
float scale = 1.0f / cbrt(product(v@stretch));
v@stretch *= (1.0f - weight) * scale + weight * ch("global_scale");
```

OPENVDB AX: FUTURE WORKFLOWS



Null null2

Covariance Attribute: rotation
Average Position Attri...: avgpos
Neighbour Attribute: neighbours
Mode: Average Positions + Covariance
Search Radius: 0.1
Neighbour Threshold: 20

Smooth positions with average weight

```
float weight = ch("avg_volume_weight");
v@P = (1.0f - weight) * v@P + weight * f@avgpos;
```

SUMMARY

- Framework for X86 SIMD instrumentation for VDB 11.X
  - https://github.com/AcademySoftwareFoundation/openvdb/pull/1648
- New surfacing methods for advanced SDF rasterization in 11.0.0
  - https://github.com/AcademySoftwareFoundation/openvdb/pull/1634
- More tools to be augmented with SIMD in the future
- Advancements to surfacing pipeline with AX post VDB 11

THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

OPENVDB COURSE

THANKS FOR LISTENING

NICHOLAS AVRAMOUSSIS
RENDERING SOFTWARE ENGINEER - WETA

NAVRAMOUSSIS@WETAFX.CO.NZ
OPENVDB-DEV@LISTS.ASWF.IO