SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

OPENVDB

CODE CASE EXAMPLES

THE PREMIER CONFERENCE & EXHIBITION ON
COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

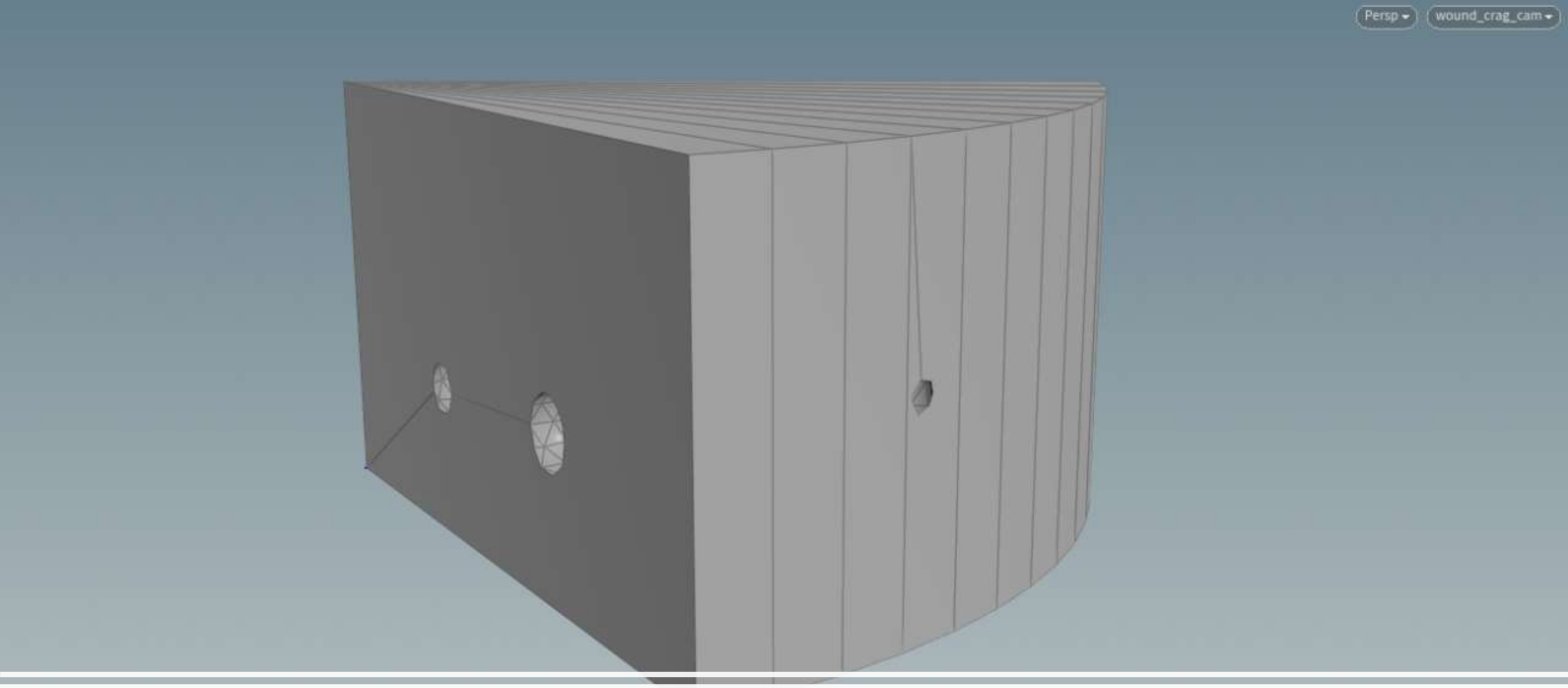


SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

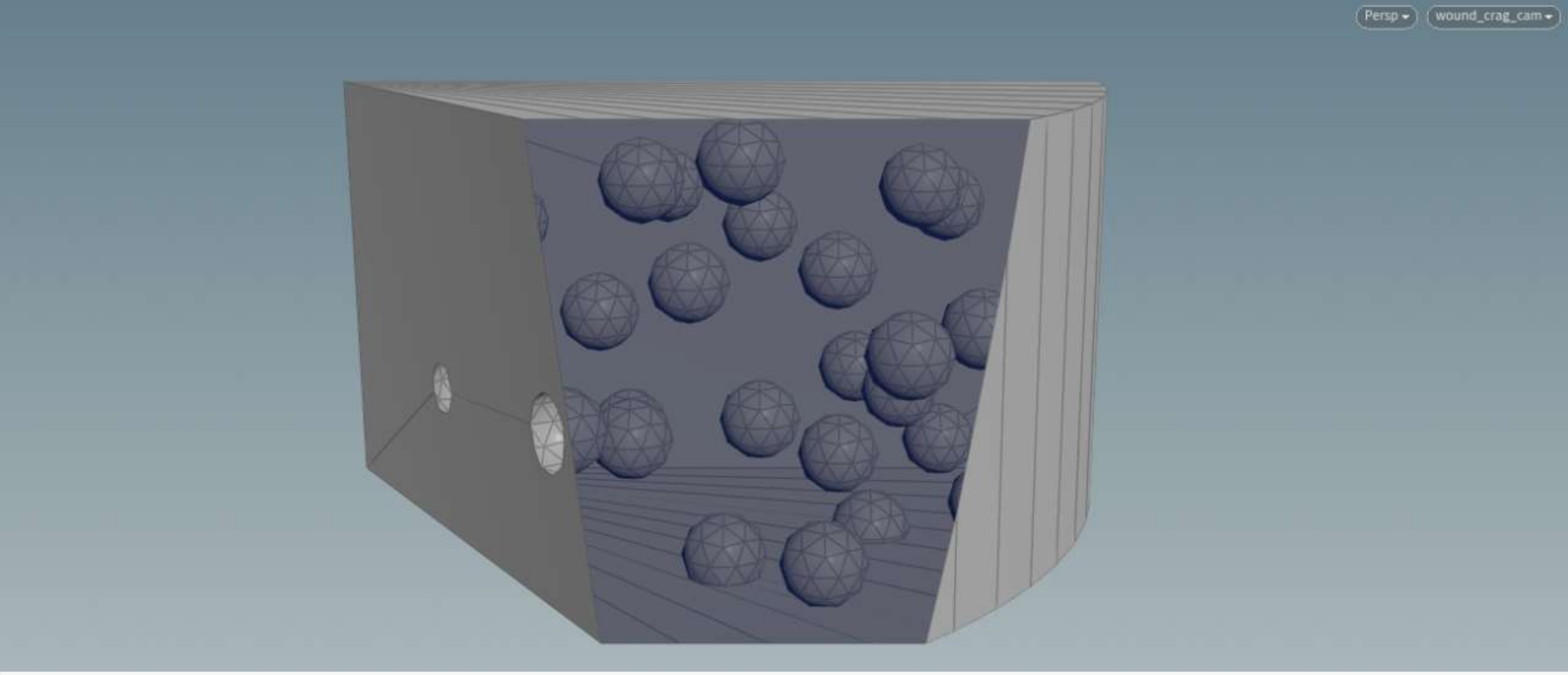
KEEPING HOLES

PUTTING THE SIGN IN SIGNED DISTANCE FIELDS

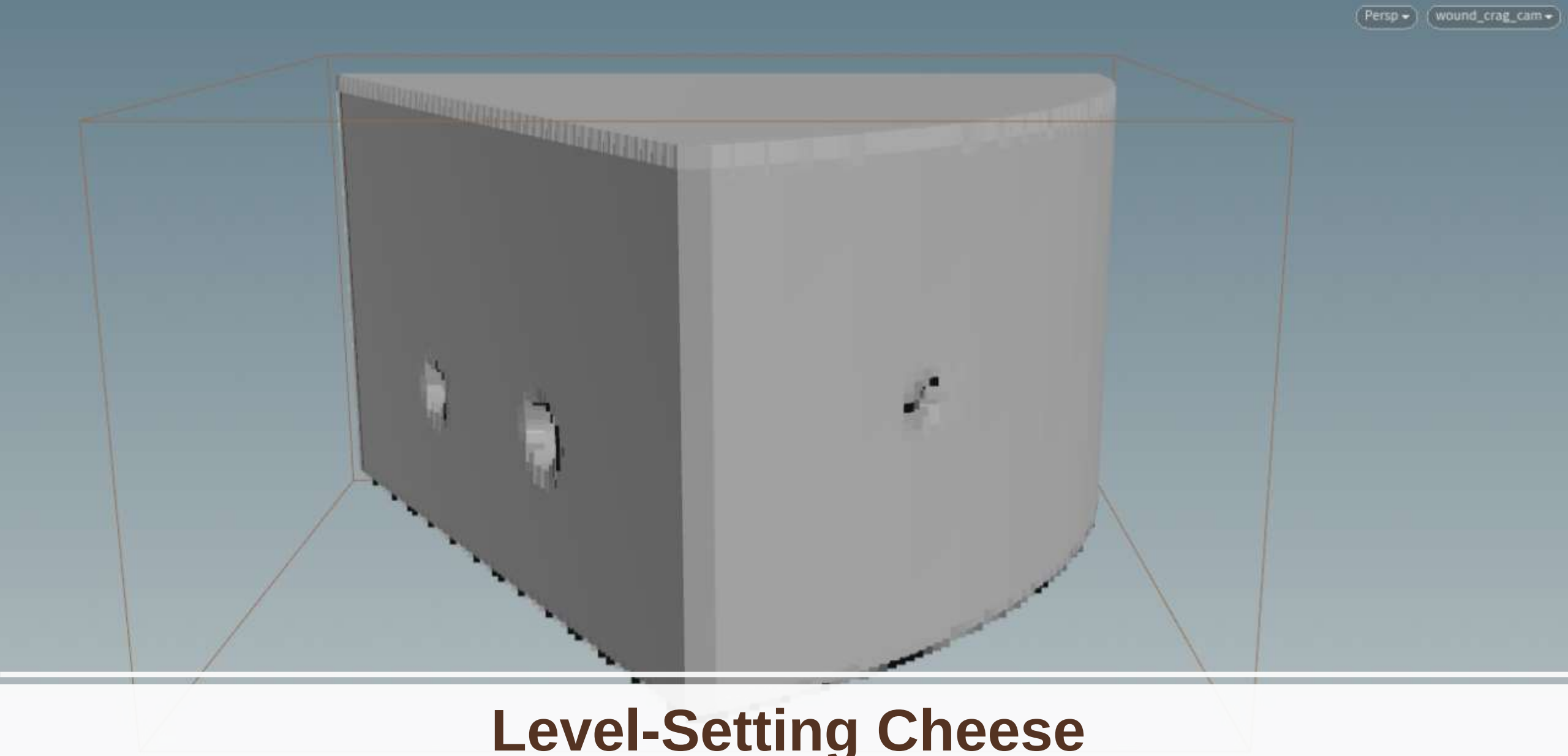


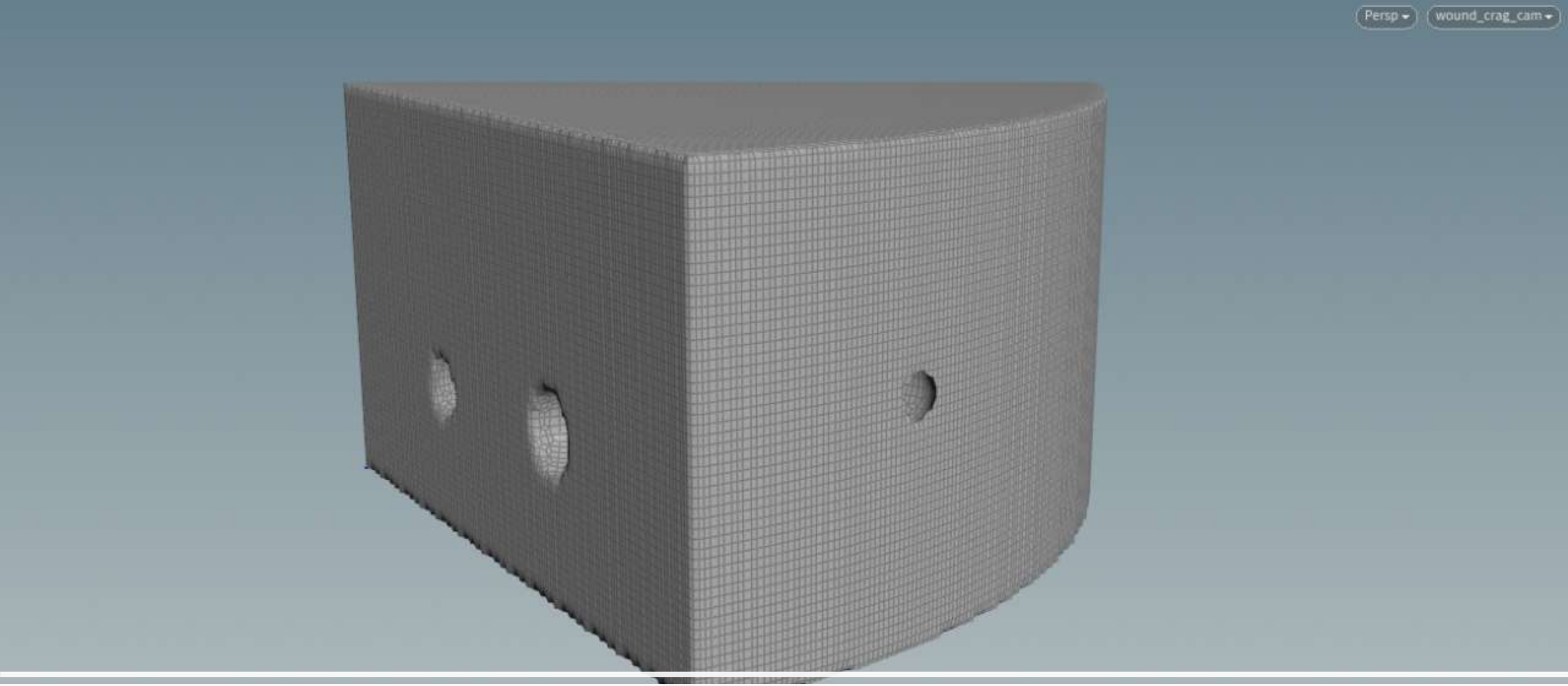
A Simple Block of Cheese



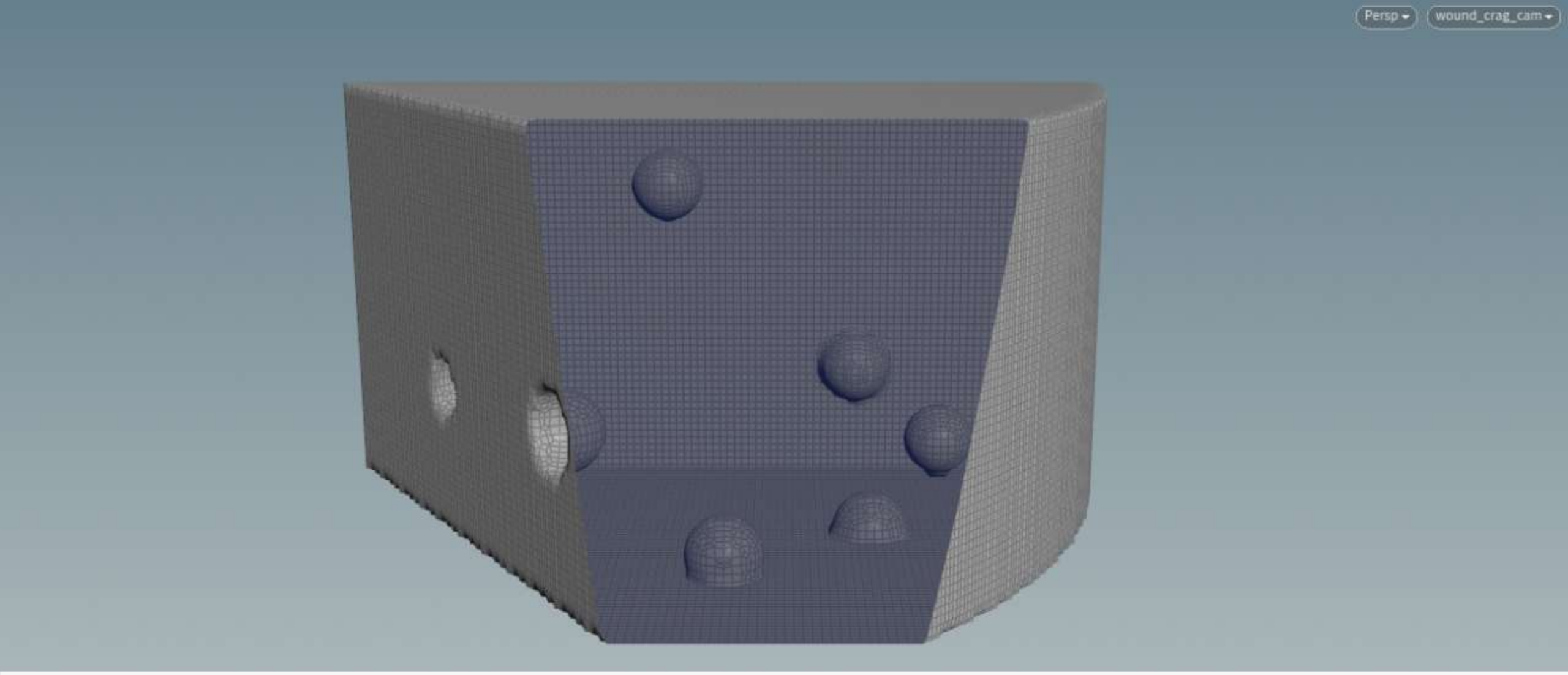
A Simple Block of Cheese with Holes



Level-Setting Cheese



Cheese Resurfaced



Unholy Cheese

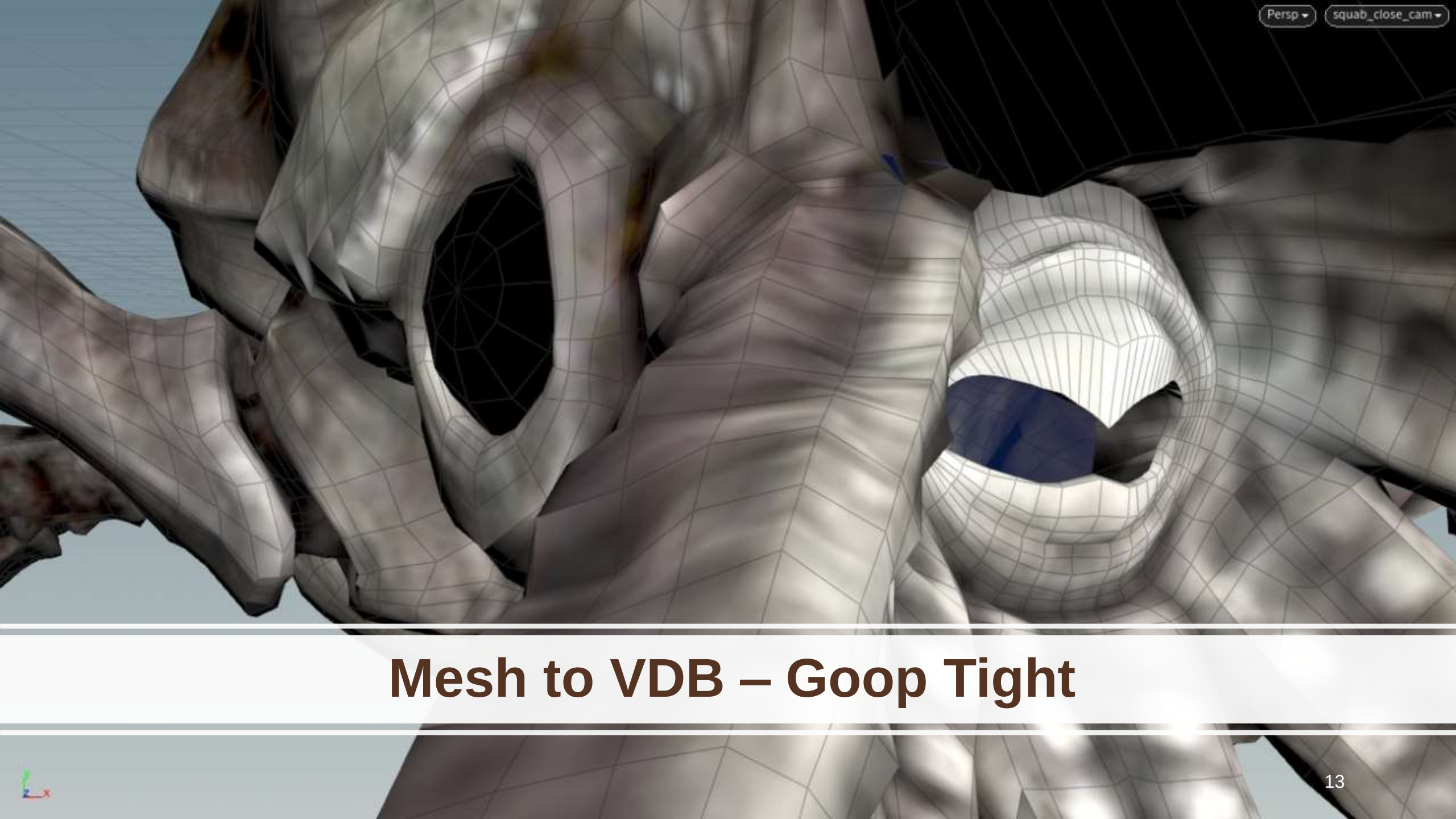
Mesh to VDB – Self Intersection

Mesh to VDB – Self Intersection

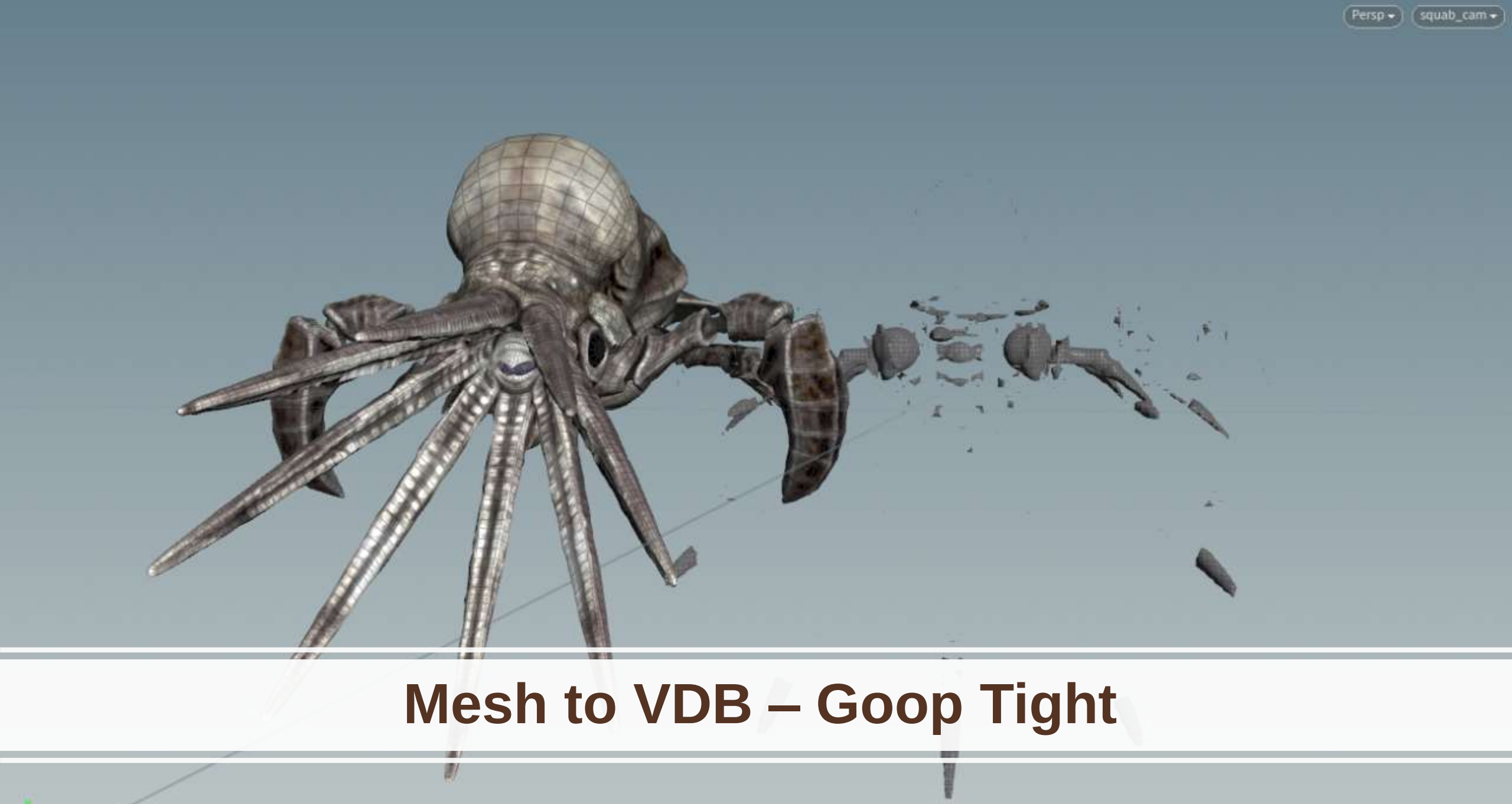
Mesh to VDB – Kit Bashed

Mesh to VDB – Kit Bashed

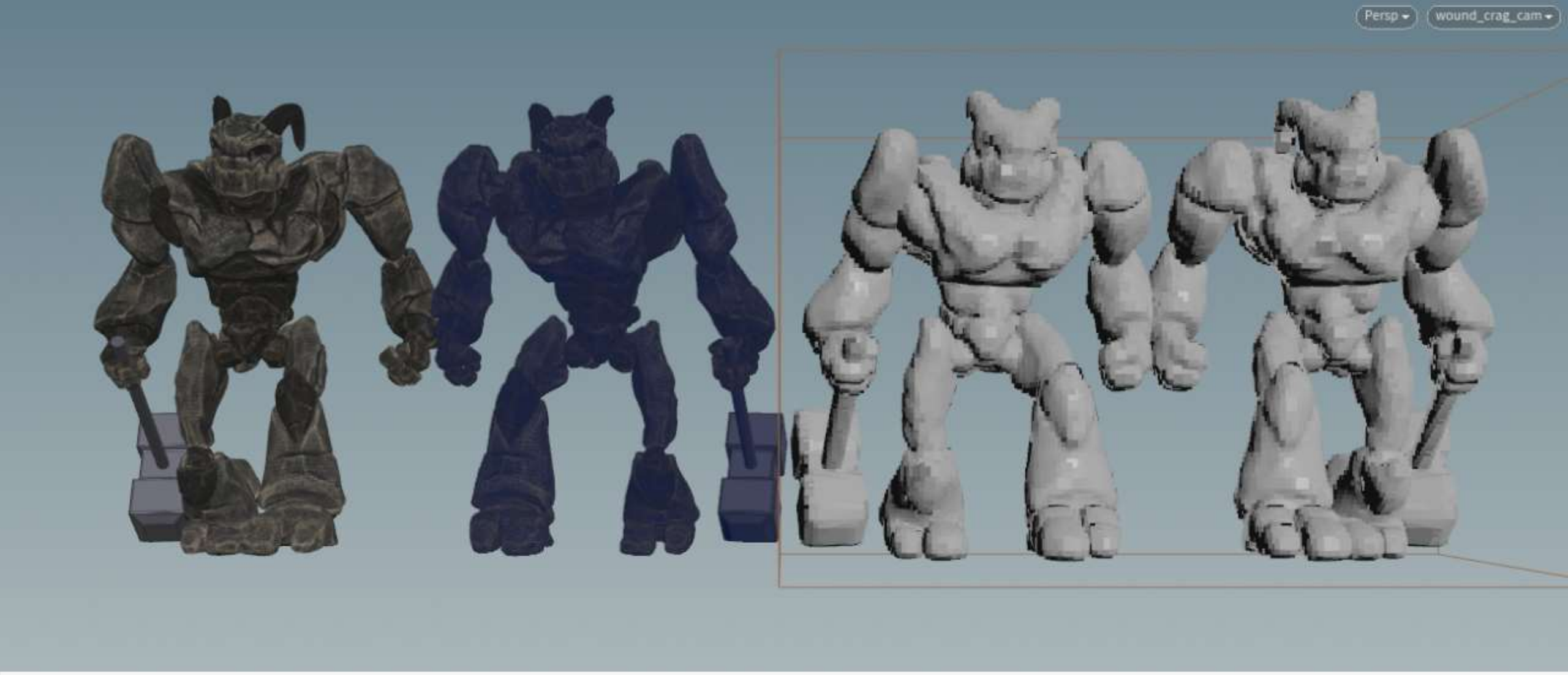
Mesh to VDB – Goop Tight



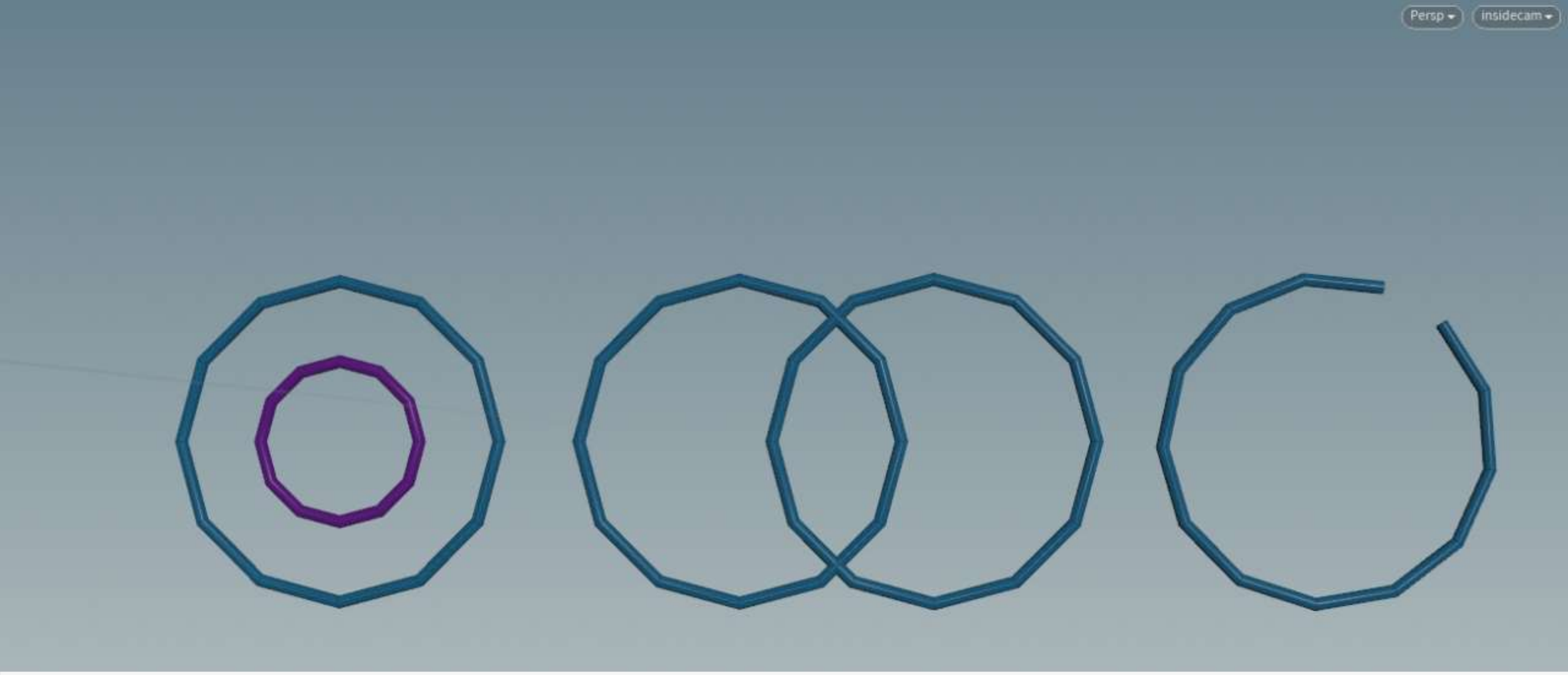
Mesh to VDB – Goop Tight



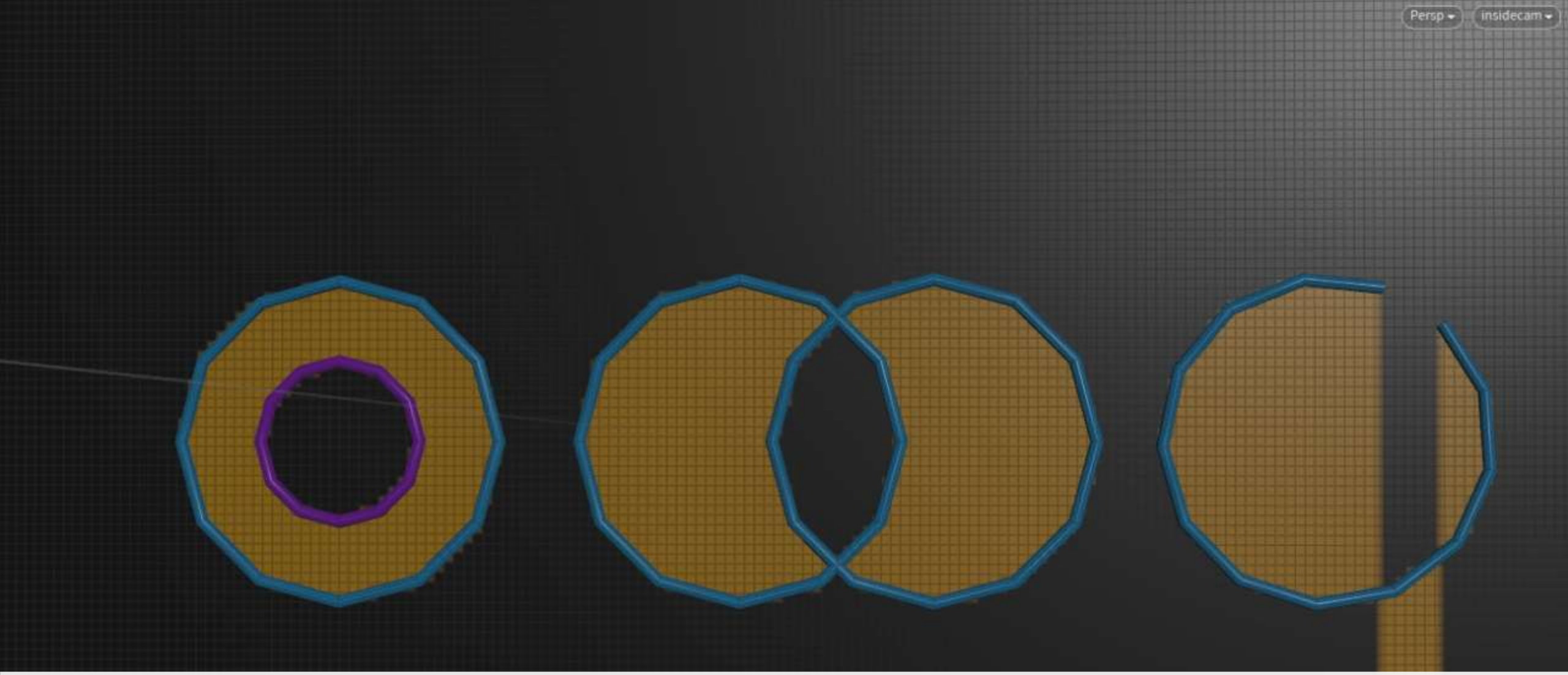
Mesh to VDB – Goop Tight



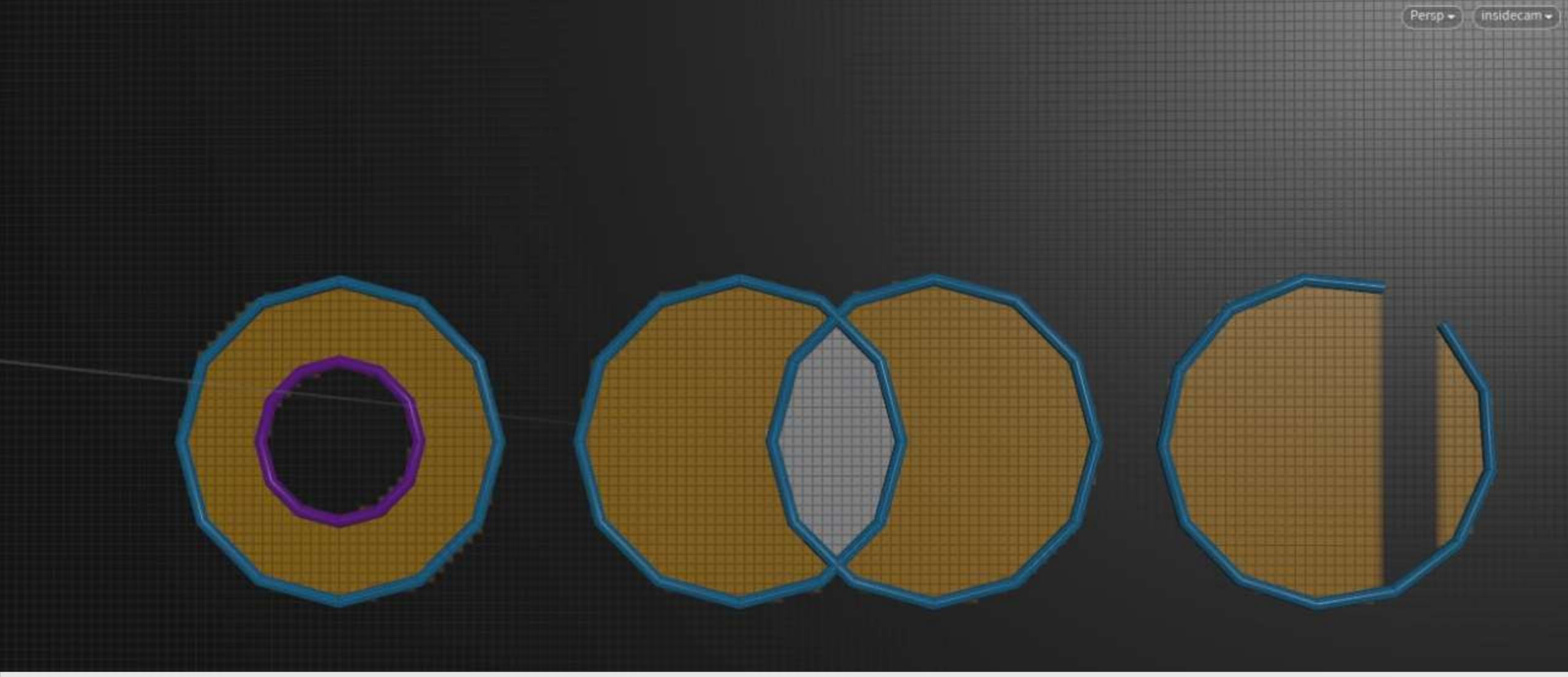
Mesh to VDB – Bad Winding



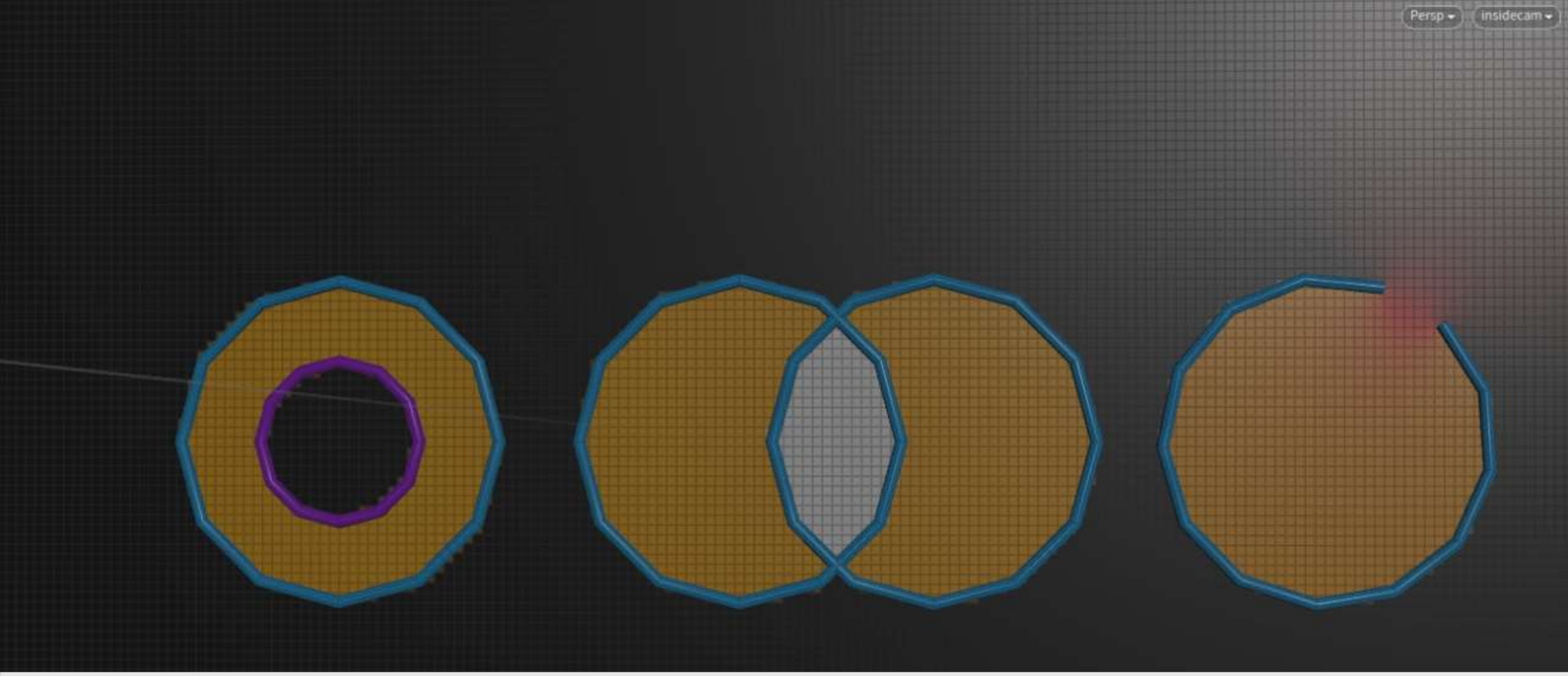
Inside Tests



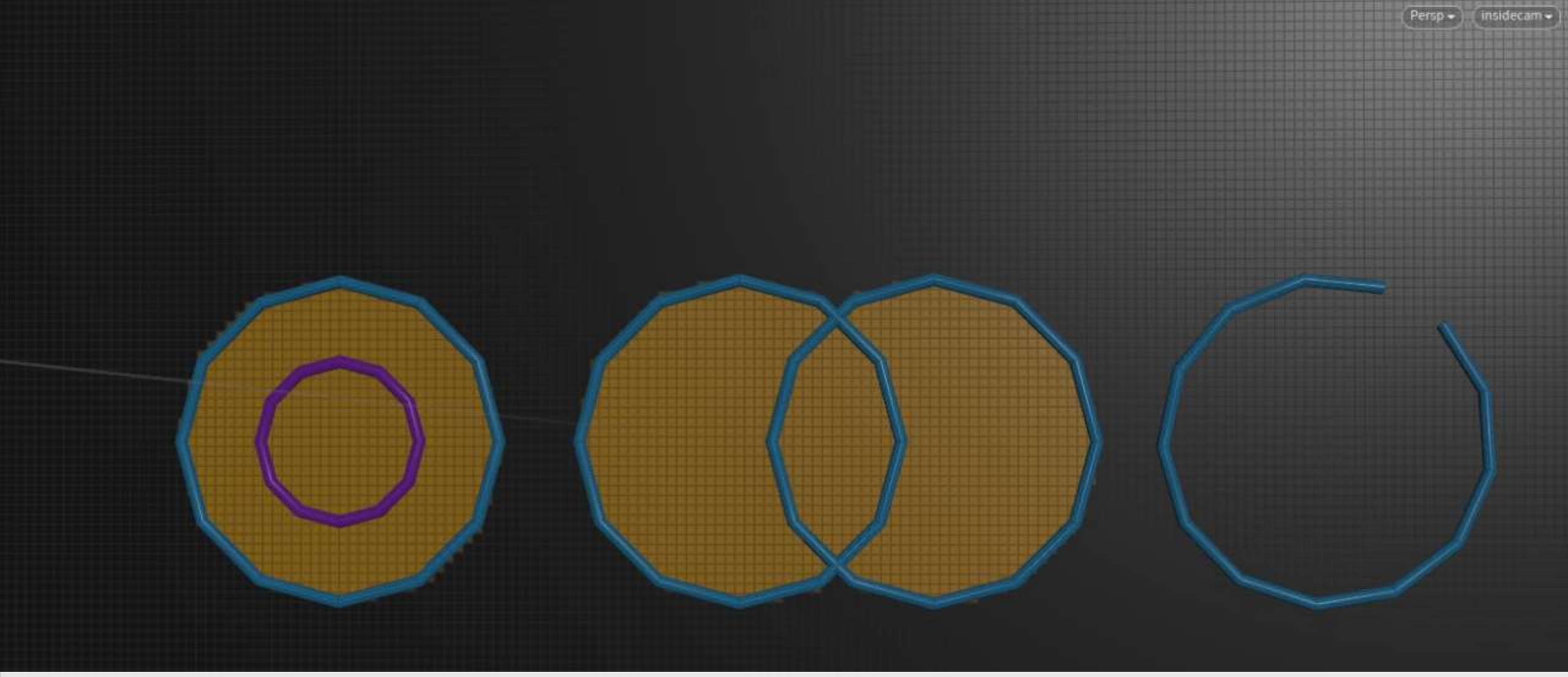
Inside Tests - Alternating



Inside Tests - Winding



Inside Tests – Generalized Winding



Inside Tests – Exterior Fill

Thanks to Tomáš Skřivan!

MESHTOVOLUME


```
template <typename GridType, typename MeshDataAdapter, typename Interrupter,
          typename InteriorTest = std::nullptr_t>
typename GridType::Ptr
meshToVolume(
    Interrupter& interrupter,
    const MeshDataAdapter& mesh,
    const math::Transform& transform,
    float exteriorBandWidth = 3.0f,
    float interiorBandWidth = 3.0f,
    int flags = 0,
    typename GridType::template ValueConverter<Int32>::Type* polygonIndexGrid = nullptr,
    InteriorTest interiorTest = nullptr,
    InteriorTestStrategy interiorTestStrategy = EVAL_EVERY_VOXEL);
```

MESHTOVOLUME


```
/// @brief Different strategies how to determine sign of an SDF when using
/// interior test.
enum InteriorTestStrategy {

    /// Evaluates interior test at every voxel. This is useful when we rebuild already
    /// existing SDF where evaluating previous grid is cheap
    EVAL_EVERY_VOXEL = 0,

    /// Evaluates interior test at least once per tile and flood fills within the tile.
    EVAL_EVERY_TILE = 1,
};
```

MESHTOVOLUME: EXPLICIT INSTANTIATION


```
#define _FUNCTION(TreeT) \
    Grid<TreeT>::Ptr meshToVolume<Grid<TreeT>>(util::NullInterrupter&, \
        const QuadAndTriangleDataAdapter<Vec3s, Vec3I>&, const
openvdb::math::Transform&, \
        float, float, int, Grid<TreeT>::ValueConverter<Int32>::Type*,
        std::nullptr_t, InteriorTestStrategy)
OPENVDB_REAL_TREE_INSTANTIATE(_FUNCTION)
#undef _FUNCTION
```

SIMPLE ORACLE


```
auto backToOldGrid = [&xform, &grid](const Coord& coord) -> openvdb::math::Vec3d {
    return grid.transform().worldToIndex(xform->indexToWorld(coord));
};

auto interiorTest =
    [acc = grid.getConstAccessor(), &backToOldGrid, &xform]
    (const Coord& coord) -> bool
{
    if (xform == nullptr) {
        return acc.getValue(coord) <= 0 ? true : false;
    }
    else {
        float value = openvdb::tools::BoxSampler::sample(acc, backToOldGrid(coord));
        return value <= 0 ? true : false;
    }
};
```

SIMPLE ORACLE


```
auto backToOldGrid = [&xform, &grid](const Coord& coord) -> openvdb::math::Vec3d {
    return grid.transform().worldToIndex(xform->indexToWorld(coord));
};

auto interiorTest =
    [acc = grid.getConstAccessor(), &backToOldGrid, &xform]
    (const Coord& coord) -> bool
{
    if (xform == nullptr) {
        return acc.getValue(coord) <= 0 ? true : false;
    }
    else {
        float value = openvdb::tools::BoxSampler::sample(acc, backToOldGrid(coord));
        return value <= 0 ? true : false;
    }
};
```

SIMPLE ORACLE


```
auto backToOldGrid = [&xform, &grid](const Coord& coord) -> openvdb::math::Vec3d {
    return grid.transform().worldToIndex(xform->indexToWorld(coord));
};

auto interiorTest =
    [acc = grid.getConstAccessor(), &backToOldGrid, &xform]
    (const Coord& coord) -> bool
{
    if (xform == nullptr) {
        return acc.getValue(coord) <= 0 ? true : false;
    }
    else {
        float value = openvdb::tools::BoxSampler::sample(acc, backToOldGrid(coord));
        return value <= 0 ? true : false;
    }
};
```

SIMPLE ORACLE


```
auto backToOldGrid = [&xform, &grid](const Coord& coord) -> openvdb::math::Vec3d {
    return grid.transform().worldToIndex(xform->indexToWorld(coord));
};

auto interiorTest =
    [acc = grid.getConstAccessor(), &backToOldGrid, &xform]
    (const Coord& coord) -> bool
{
    if (xform == nullptr) {
        return acc.getValue(coord) <= 0 ? true : false;
    }
    else {
        float value = openvdb::tools::BoxSampler::sample(acc, backToOldGrid(coord));
        return value <= 0 ? true : false;
    }
};
```

SIMPLE ORACLE


```
auto backToOldGrid = [&xform, &grid](const Coord& coord) -> openvdb::math::Vec3d {
    return grid.transform().worldToIndex(xform->indexToWorld(coord));
};

auto interiorTest =
    [acc = grid.getConstAccessor(), &backToOldGrid, &xform]
    (const Coord& coord) -> bool
{
    if (xform == nullptr) {
        return acc.getValue(coord) <= 0 ? true : false;
    }
    else {
        float value = openvdb::tools::BoxSampler::sample(acc, backToOldGrid(coord));
        return value <= 0 ? true : false;
    }
};
```

SIMPLE ORACLE


```
return meshToVolume<GridType>(*interrupter, mesh, *transform, exBandWidth, inBandWidth,
    DISABLE_RENORMALIZATION, nullptr
#if OPENVDB_USE_ORACLE_IN_REBUILD
    , interiorTest, EVAL_EVERY_VOXEL
#endif
    );
```

SIMPLE ORACLE


```
return meshToVolume<GridType>(*interrupter, mesh, *transform, exBandWidth, inBandWidth,
    DISABLE_RENORMALIZATION, nullptr
#if OPENVDB_USE_ORACLE_IN_REBUILD
    , interiorTest, EVAL_EVERY_VOXEL
#endif
    );
```

COMPLEX ORACLE


```
GU_WindingNumber3DApprox windingNumber;
auto interiorTest =
    [transform, &windingNumber]
    (const openvdb::Coord& coord) -> bool
{
    auto pt = UTvdbConvert(transform->indexToWorld(coord));
    auto wn = windingNumber.eval(pt, 2.0);
    return fabs(wn) >= 0.5 ? true : false;
};
windingNumber.init(*inputGdp, nullptr, 2);
```

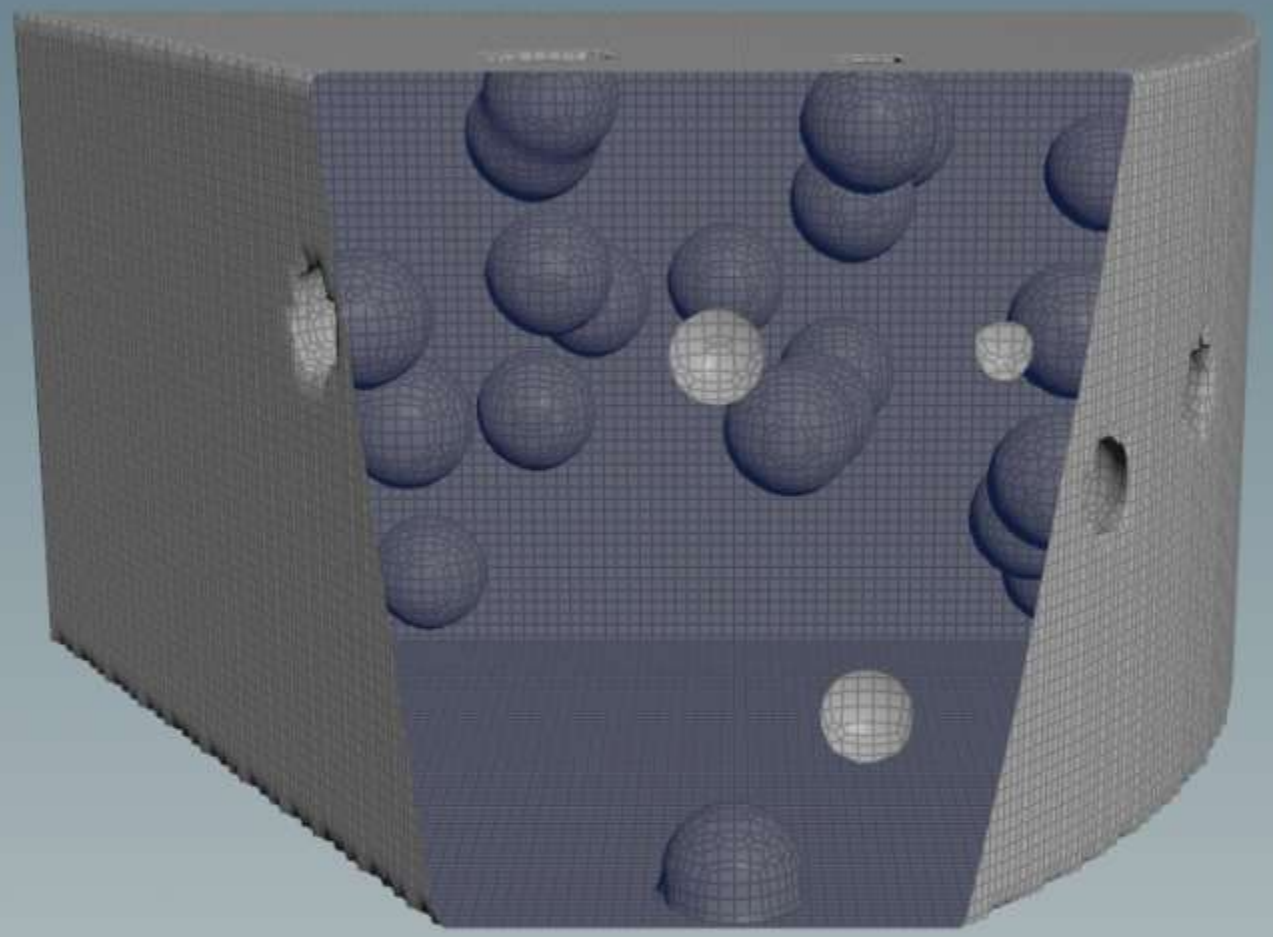
COMPLEX ORACLE


```
grid = openvdb::tools::meshToVolume<openvdb::FloatGrid>
    (
     boss.interrupter(),
     mesh, *transform, exBand, inBand, conversionFlags,
     primitiveIndexGrid.get(),
     interiorTest,
     openvdb::tools::EVAL_EVERY_TILE
    );
```

Fast winding numbers for soups and clouds.

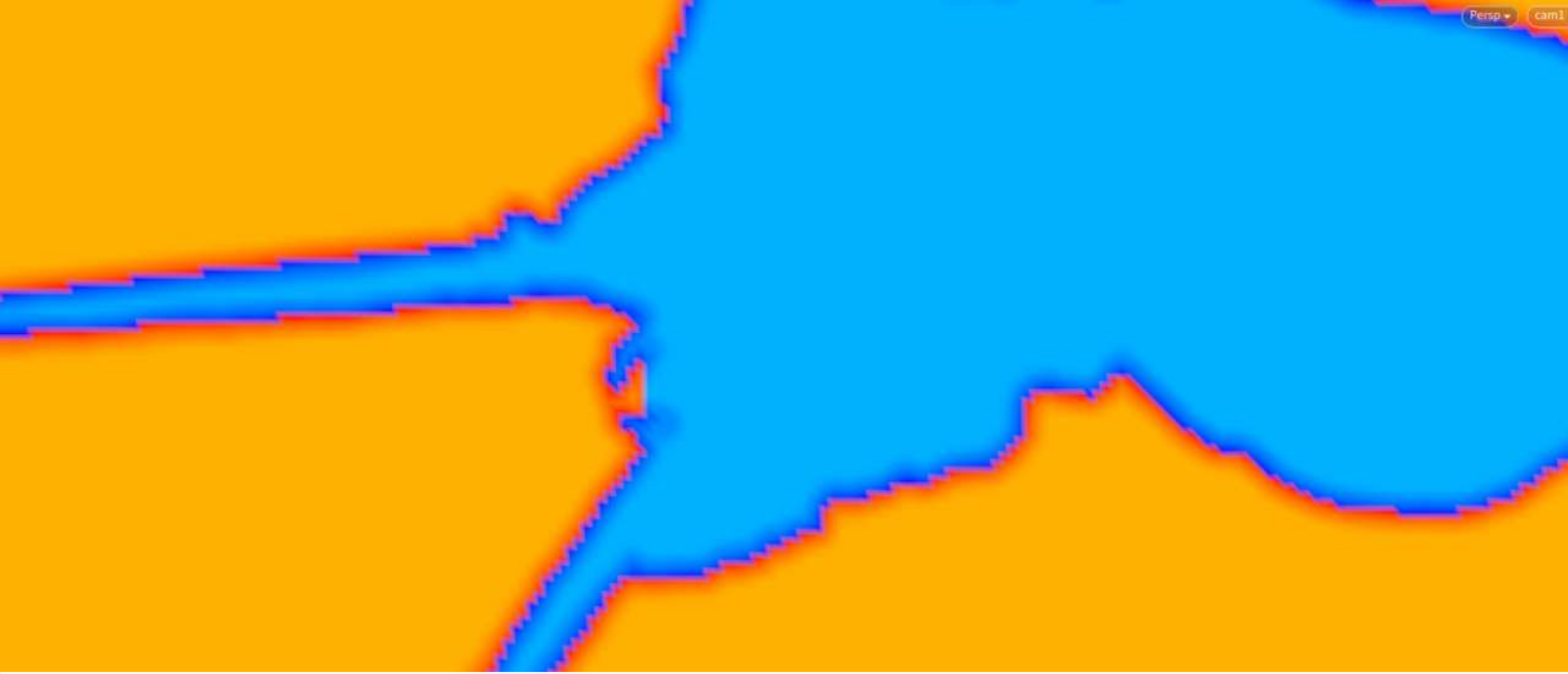
Barill, G., **Dickson, N**. G., Schmidt, R., Levin, D. I. W., & Jacobson, A. (2018)

https://doi.org/10.1145/3197517.3201337



Cheese Holes

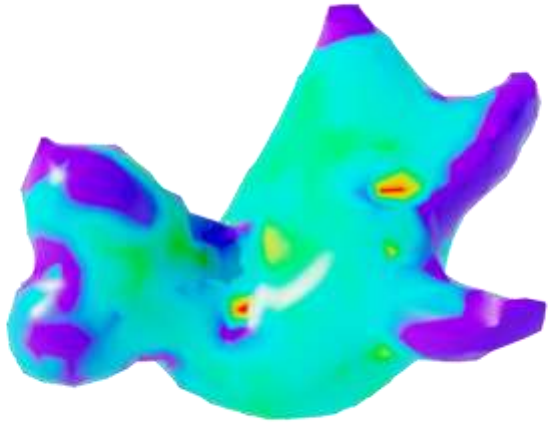
Restores the Squab



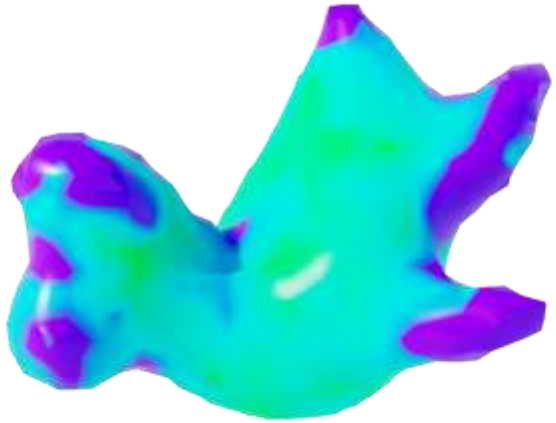
Gap Filling

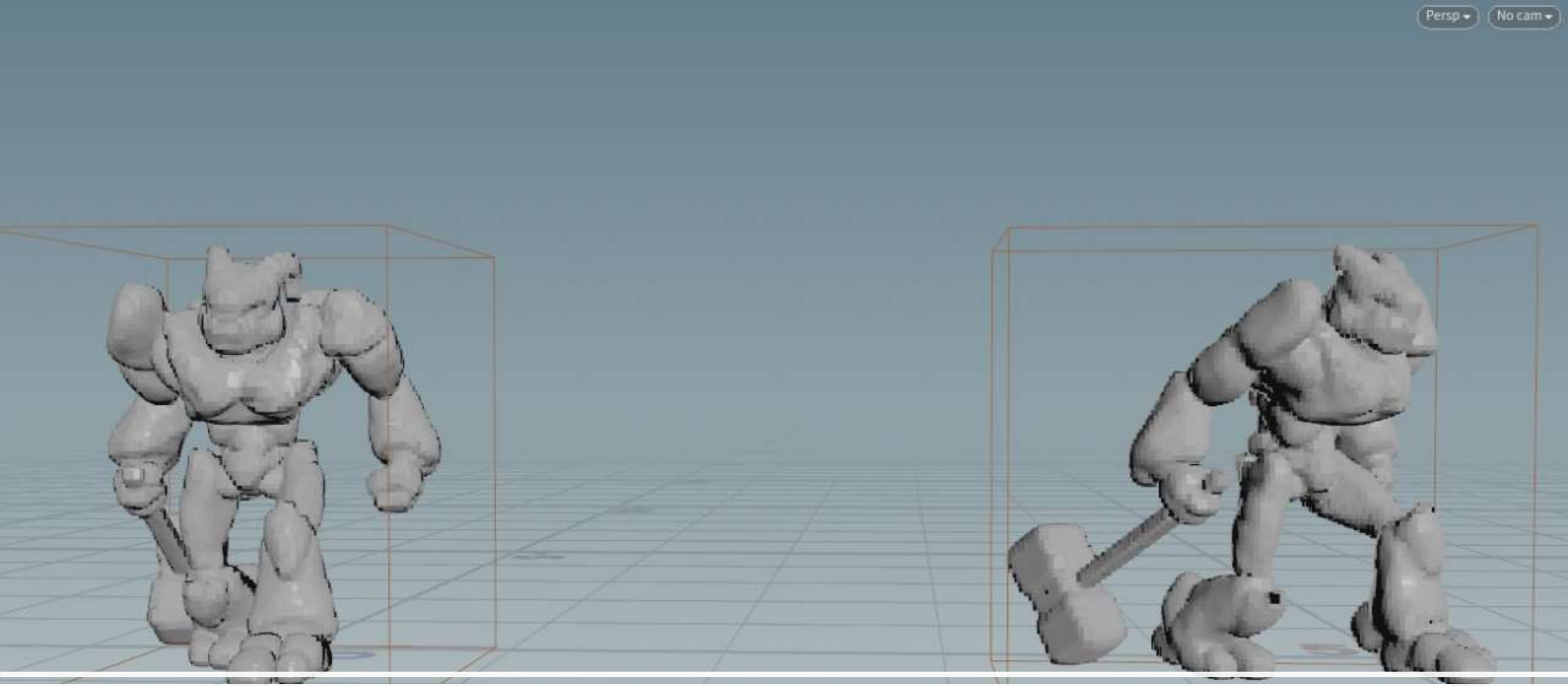
LEN(GRAD(SDF))-1

VOTING

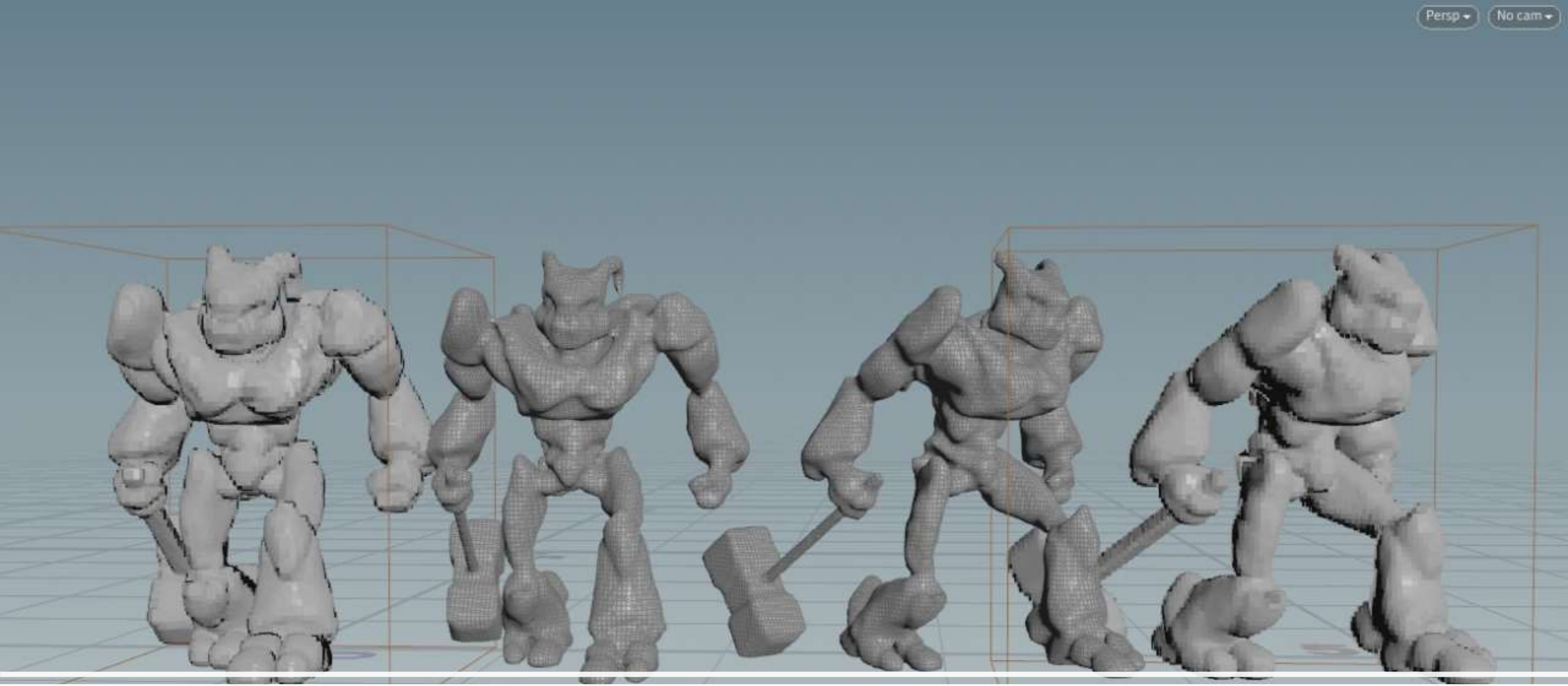


SIMPLE ORACLE

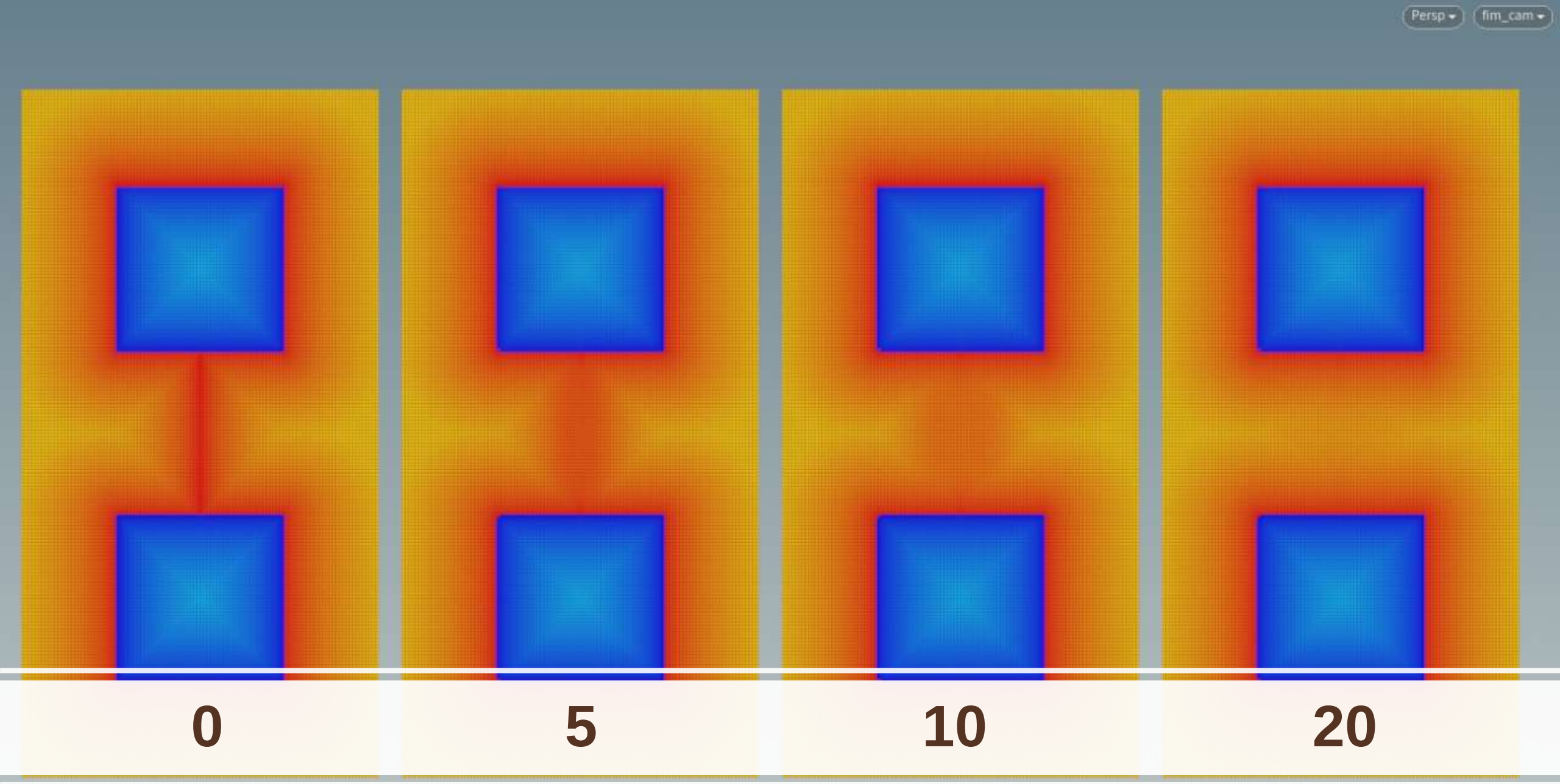


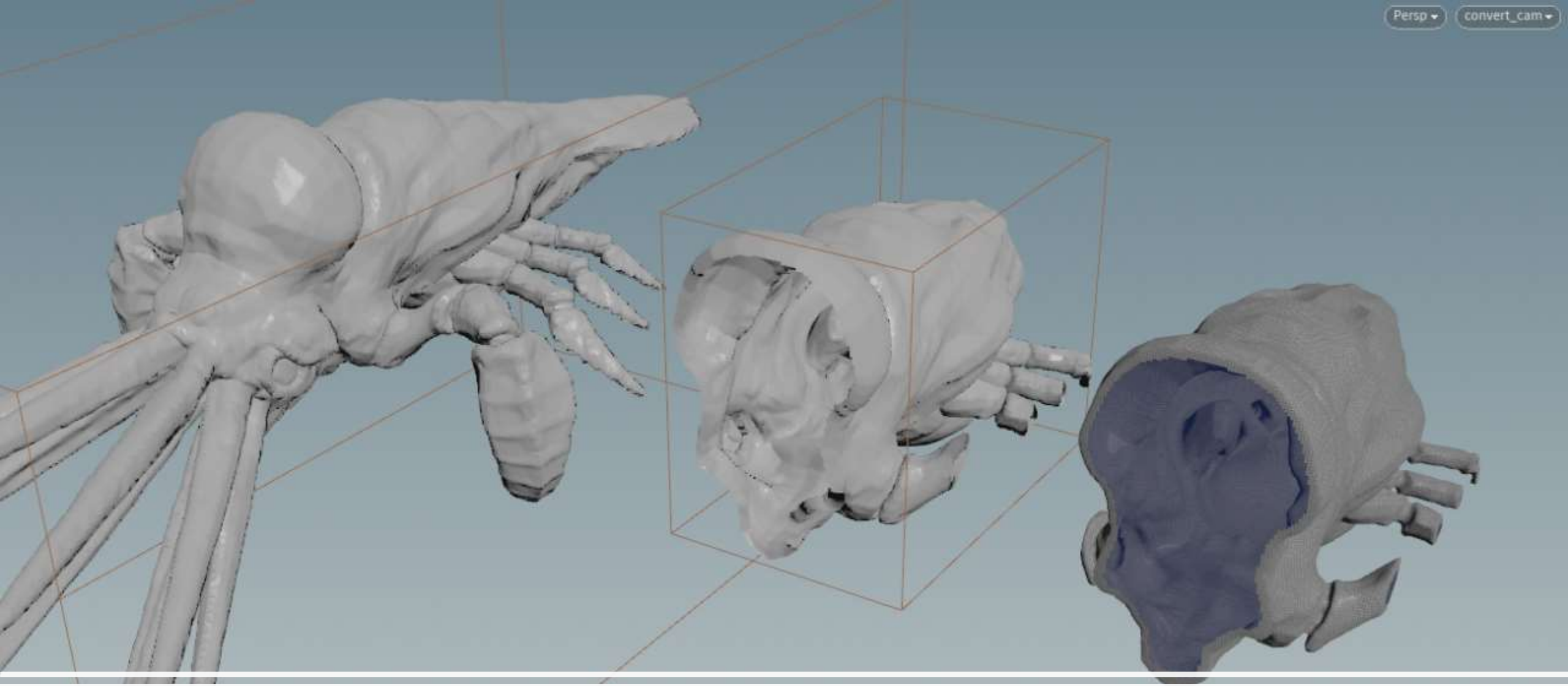


Levelset Resample

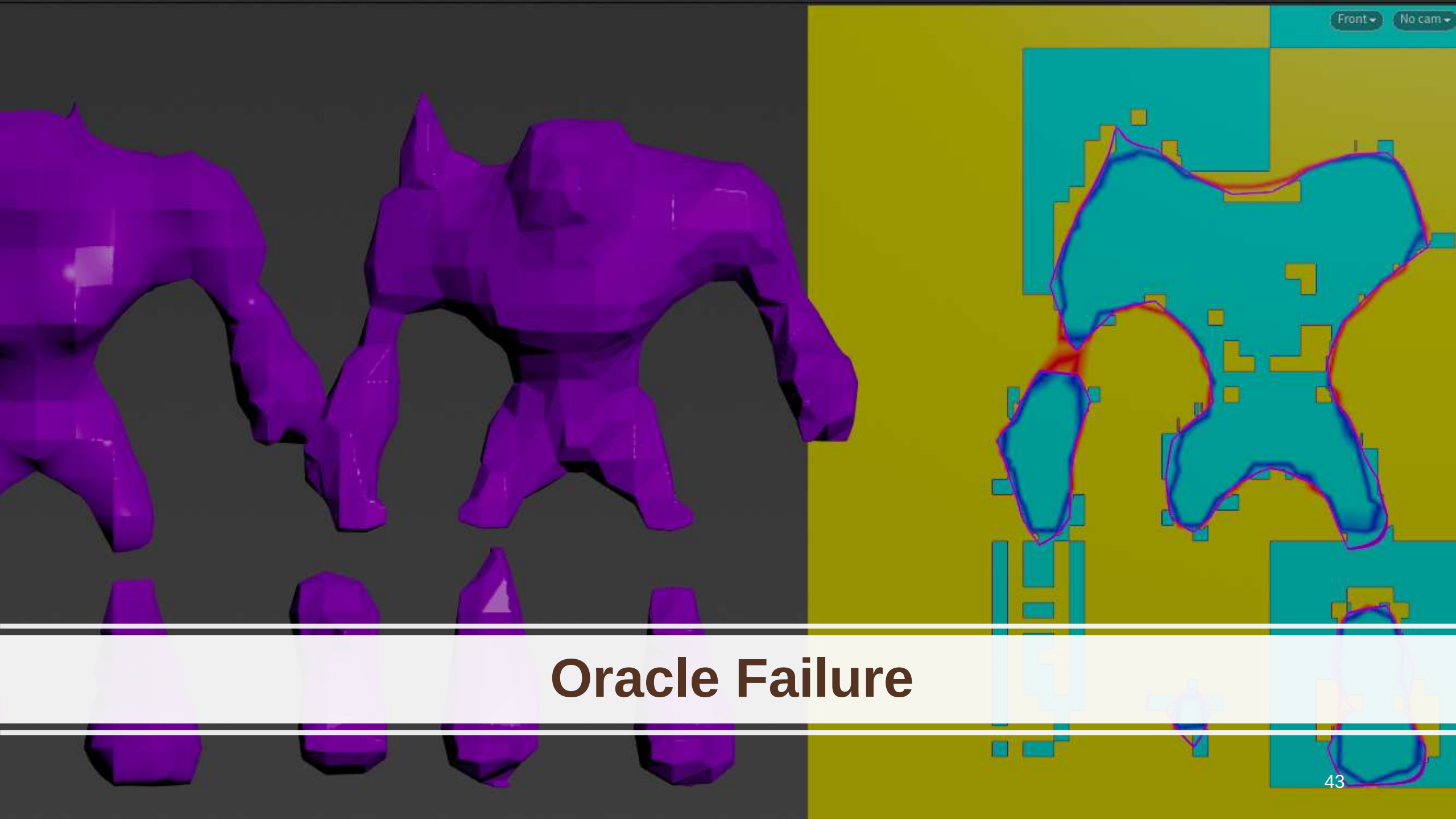


Levelset Resample via Rebuild

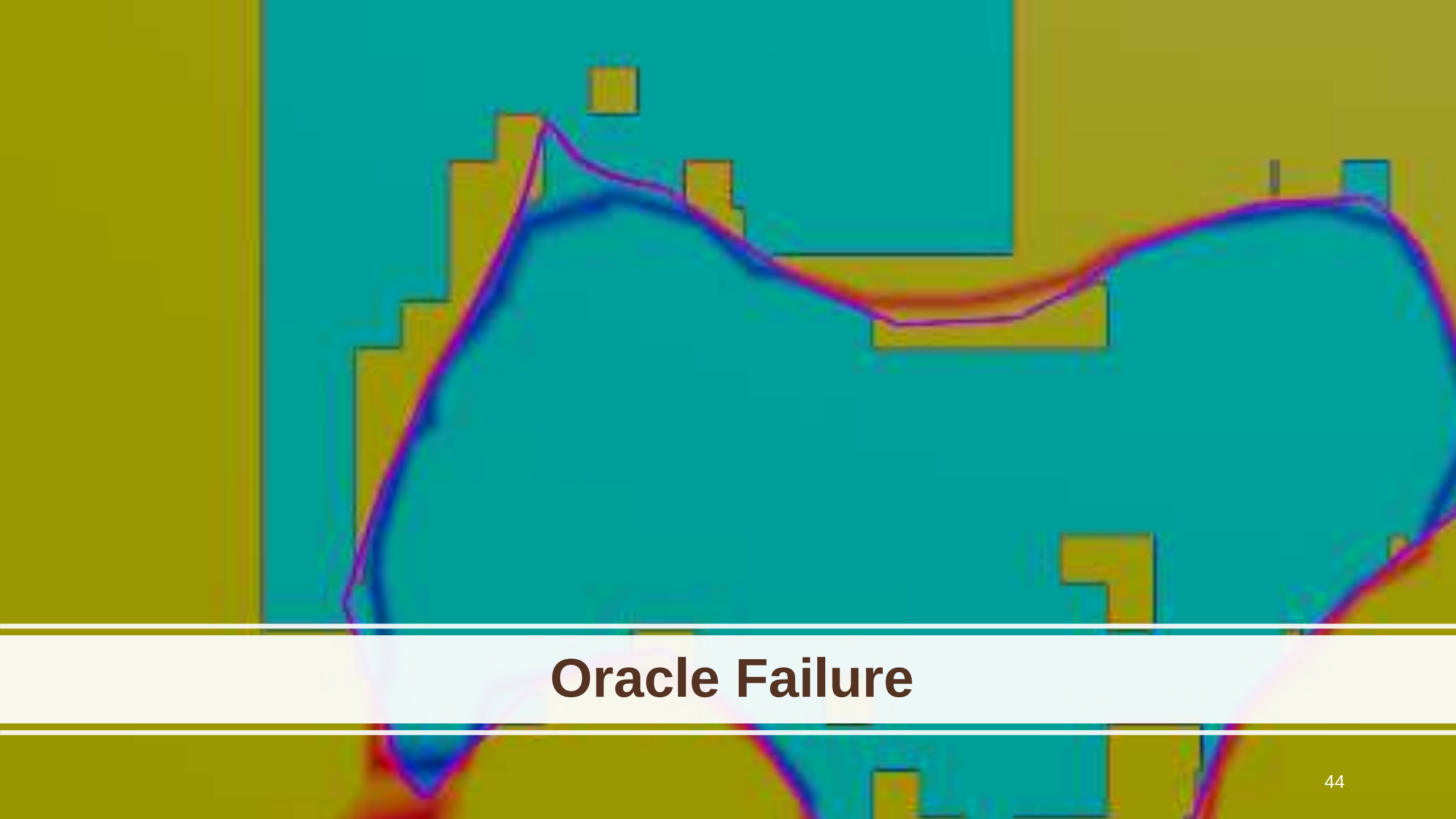

Persp
fim_cam
0
5
10
20



Signed Tile Boundaries



Oracle Failure



Oracle Failure

FUTURE WORK

- Fix tiled-sign problems in VDB to Polygons
- If dilation is detected (transforms always uniform for SDF)
  - Dynamic Node Manager to activate narrow band
  - Directly sample input
  - Polish in-place