Exposing Artists to OpenCL

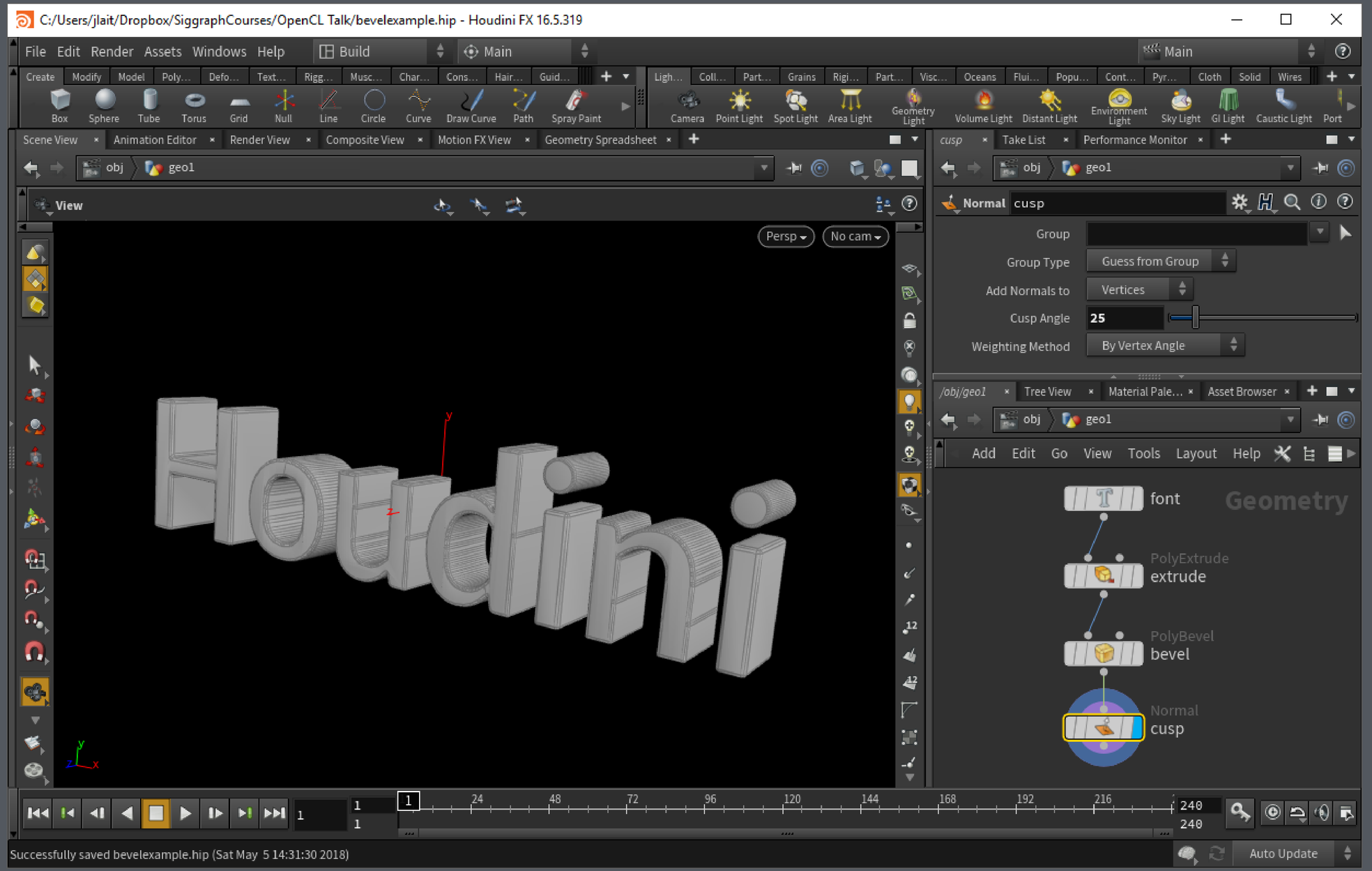
Jeff Lait | SideFX

SideFX®

Art is Technical

Houdini

What is Houdini?



Where We Came From – World of Pipes

~

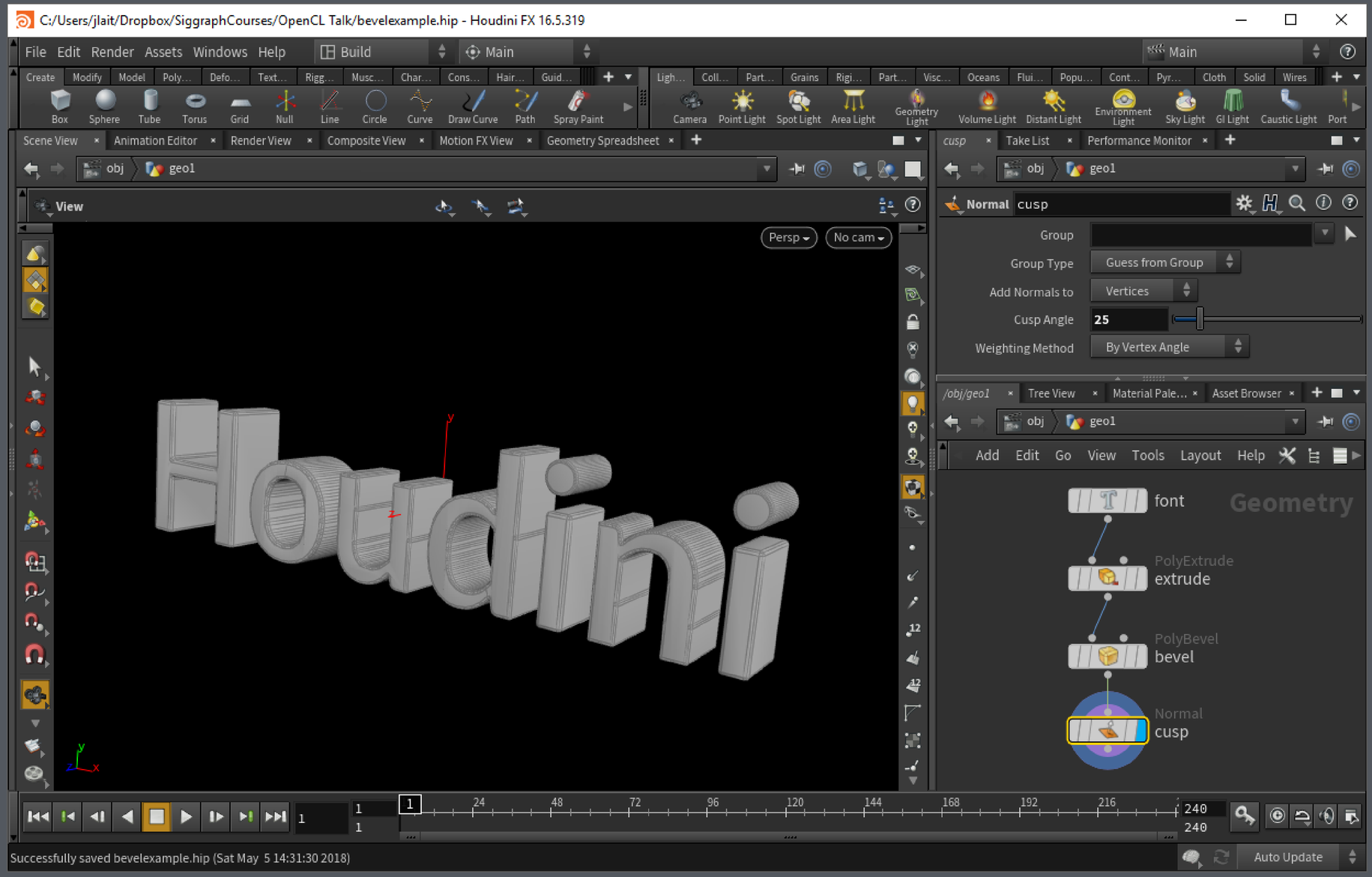
```
jlait@godel ~
$ dir
bin  bugfiles  dev_gba  houdini16.5  temp

jlait@godel ~
$ pfont -s "Houdini" -f 2 | pextrude -z -d .5 | pbevel -w .1 | pcusp -a 25 > log
o.bpoly
```

Patch Panels as an Artistic Medium

Layer-based Proceduralism

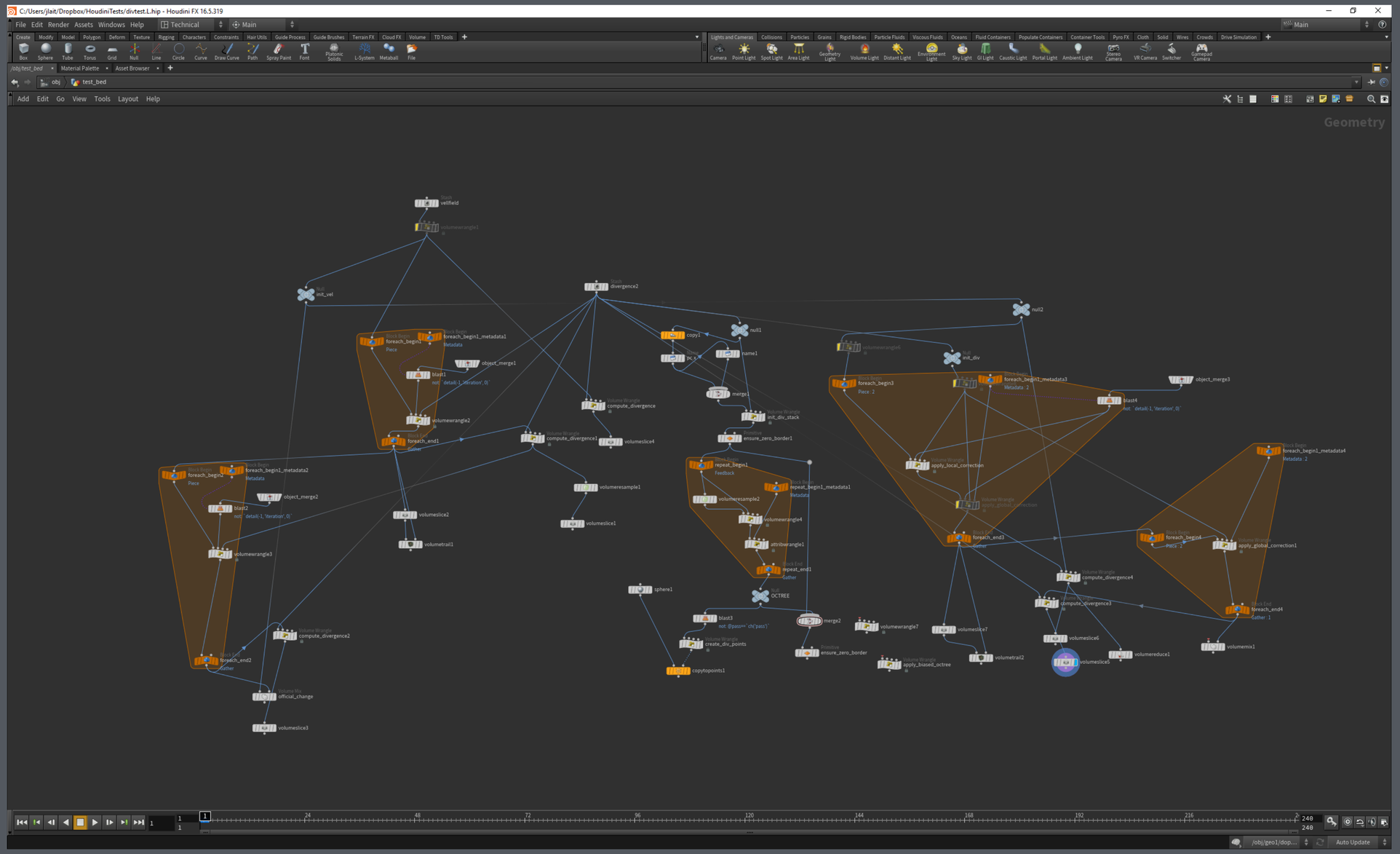
Houdini

Today: Node-Based Proceduralism



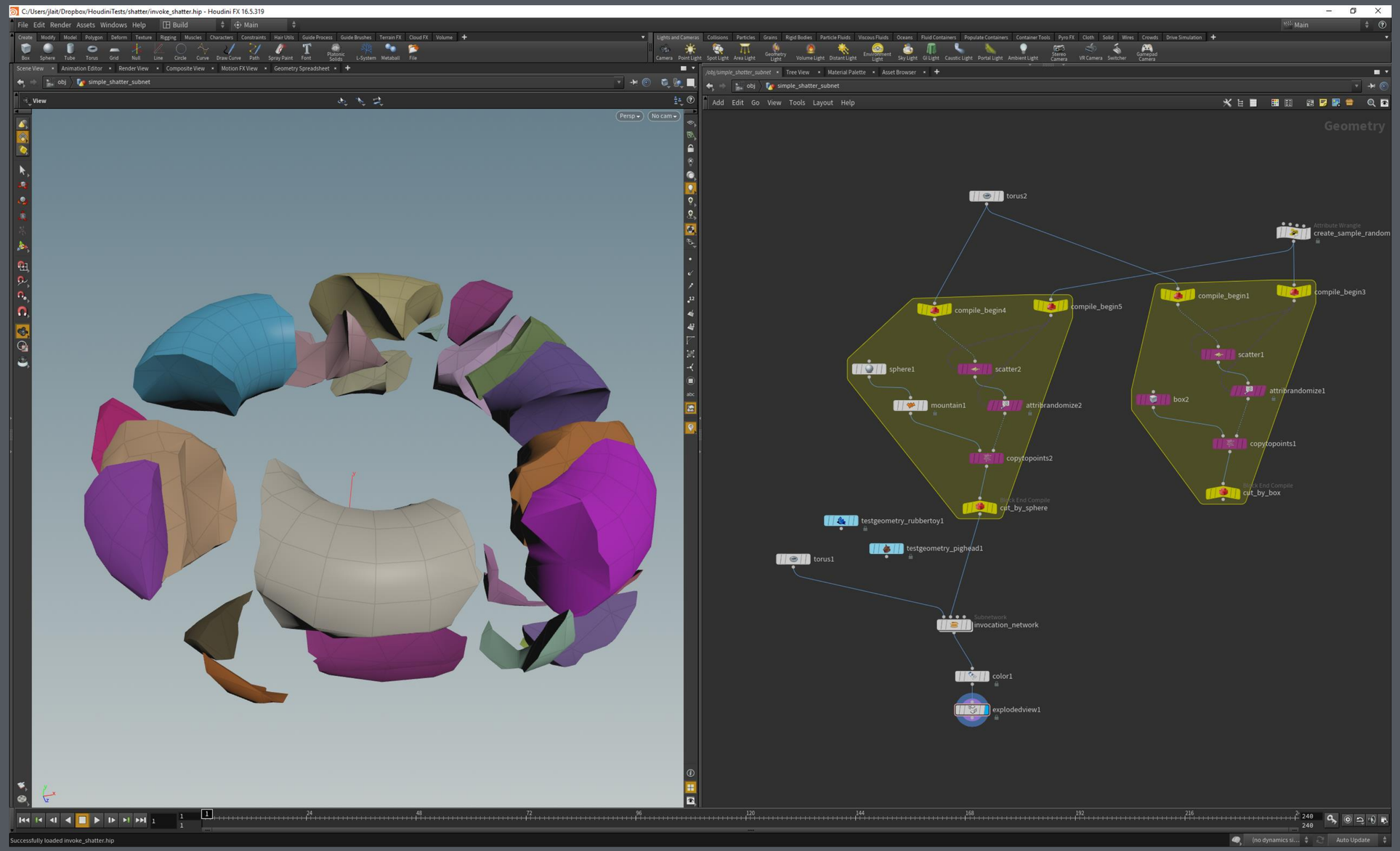
Houdini

Advantage: Dead Branches



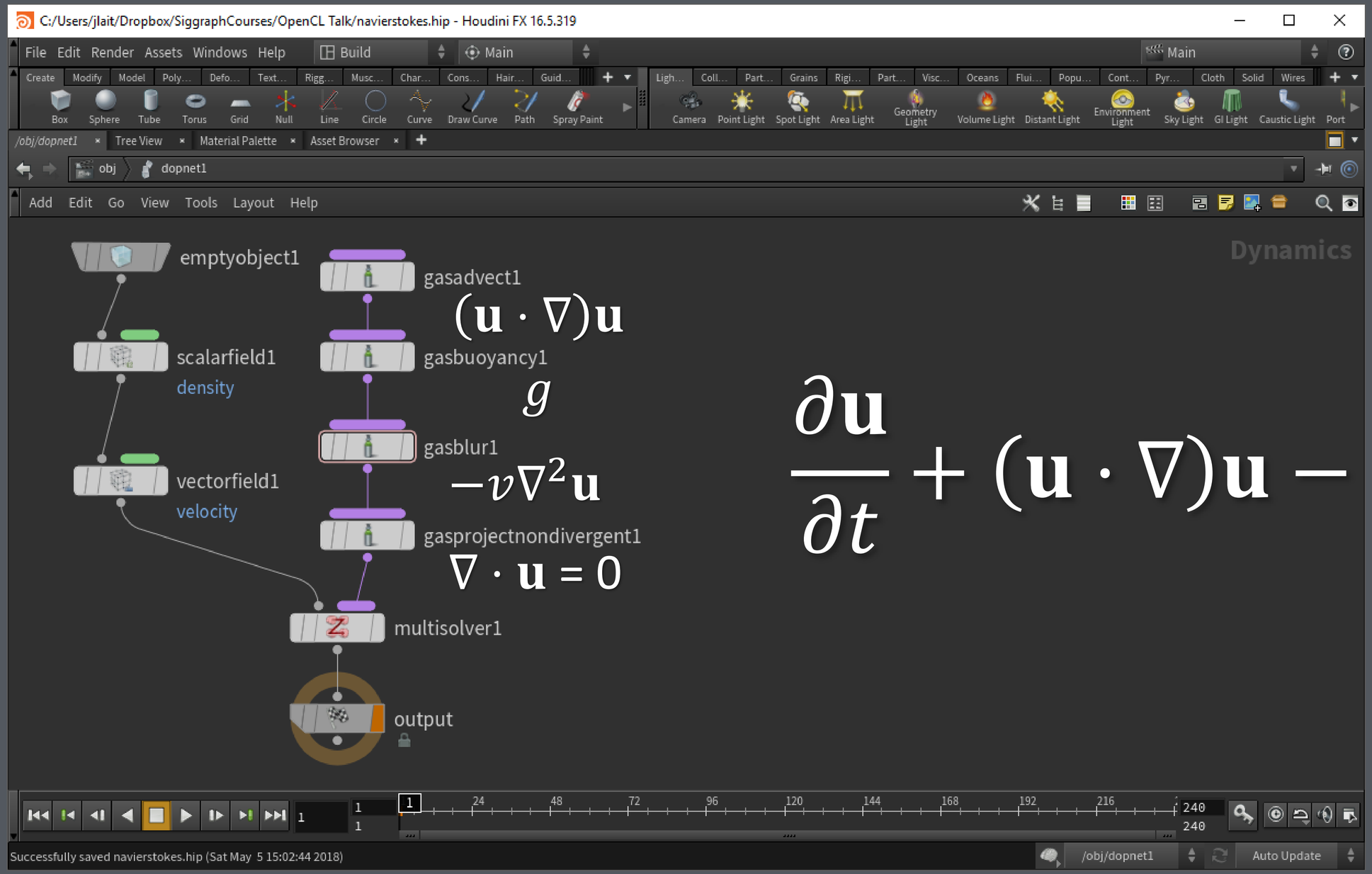
Houdini

Advantage: Multiple Sources



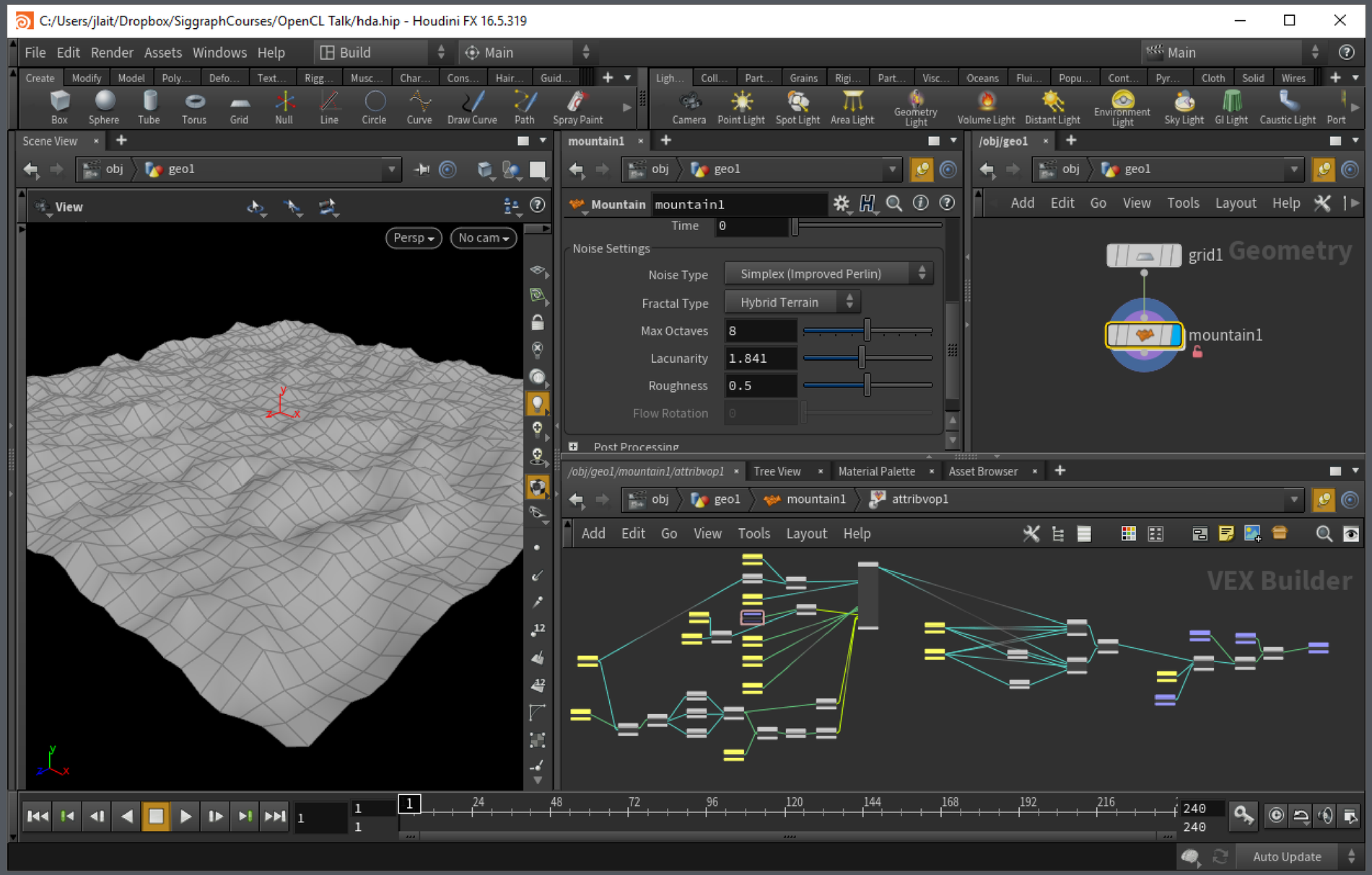
Houdini

Operator Splitting

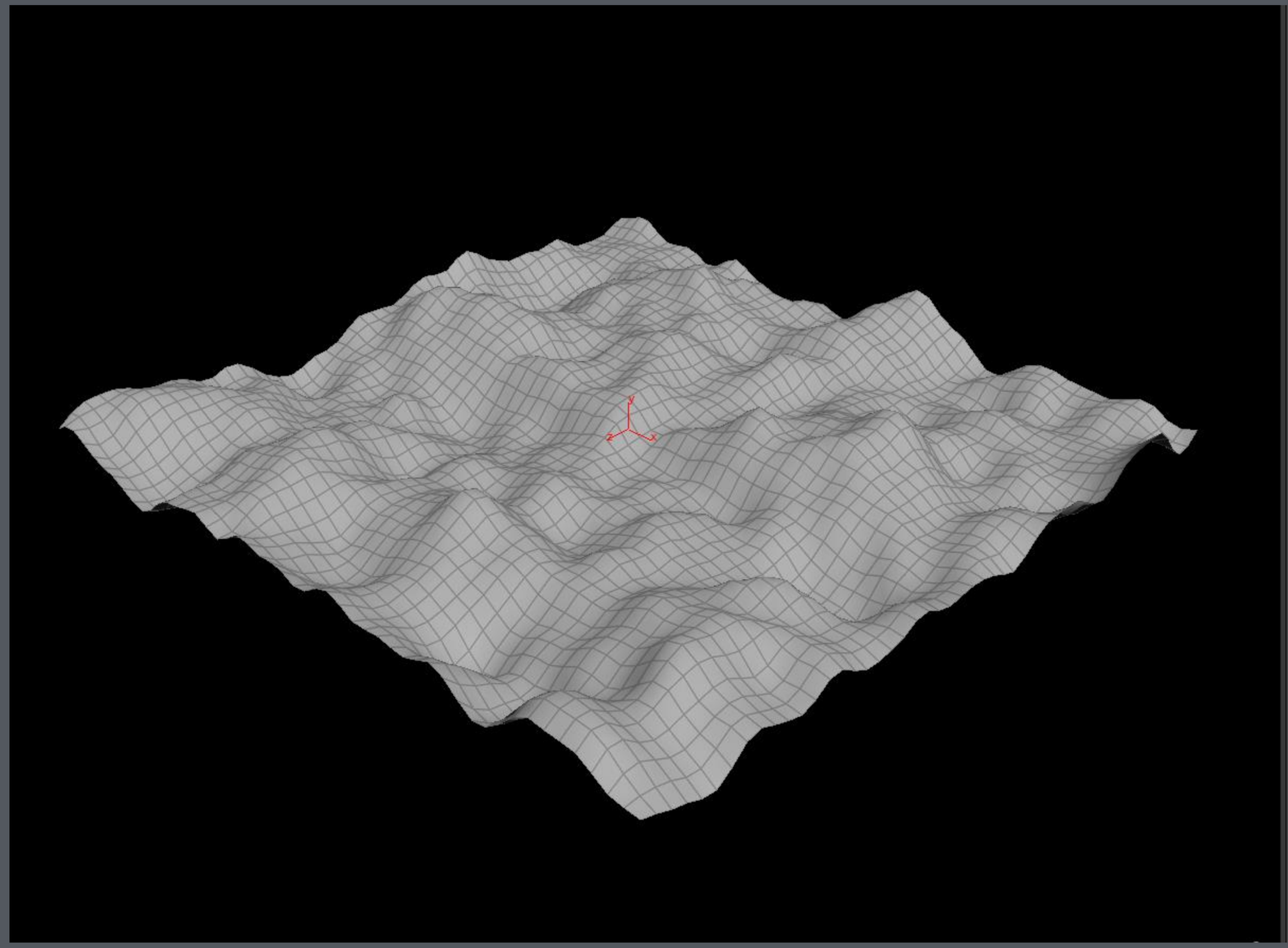


$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - v\nabla^2\mathbf{u} = g$$

Encapsulation: HDAs

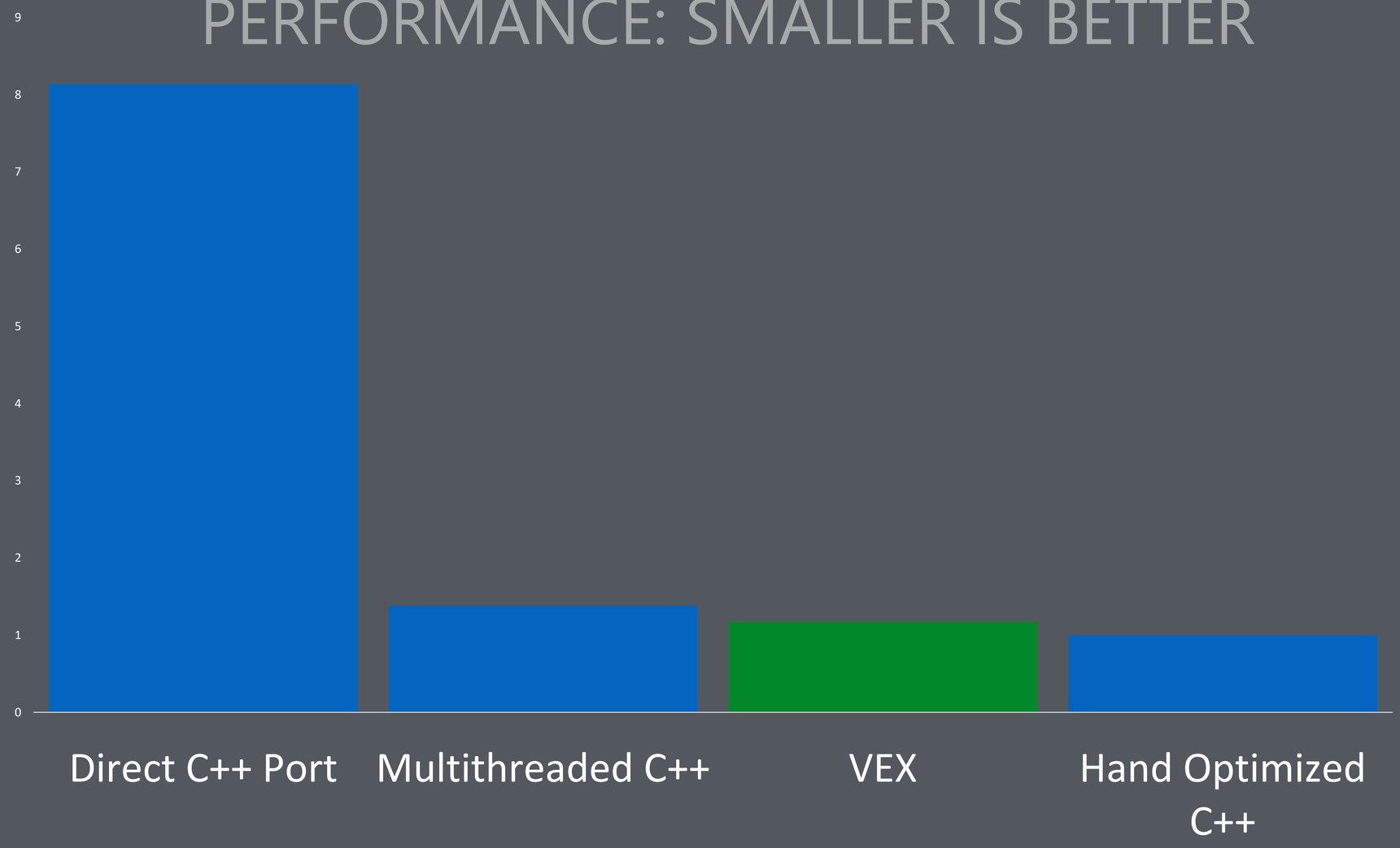


Kernel Languages: VEX



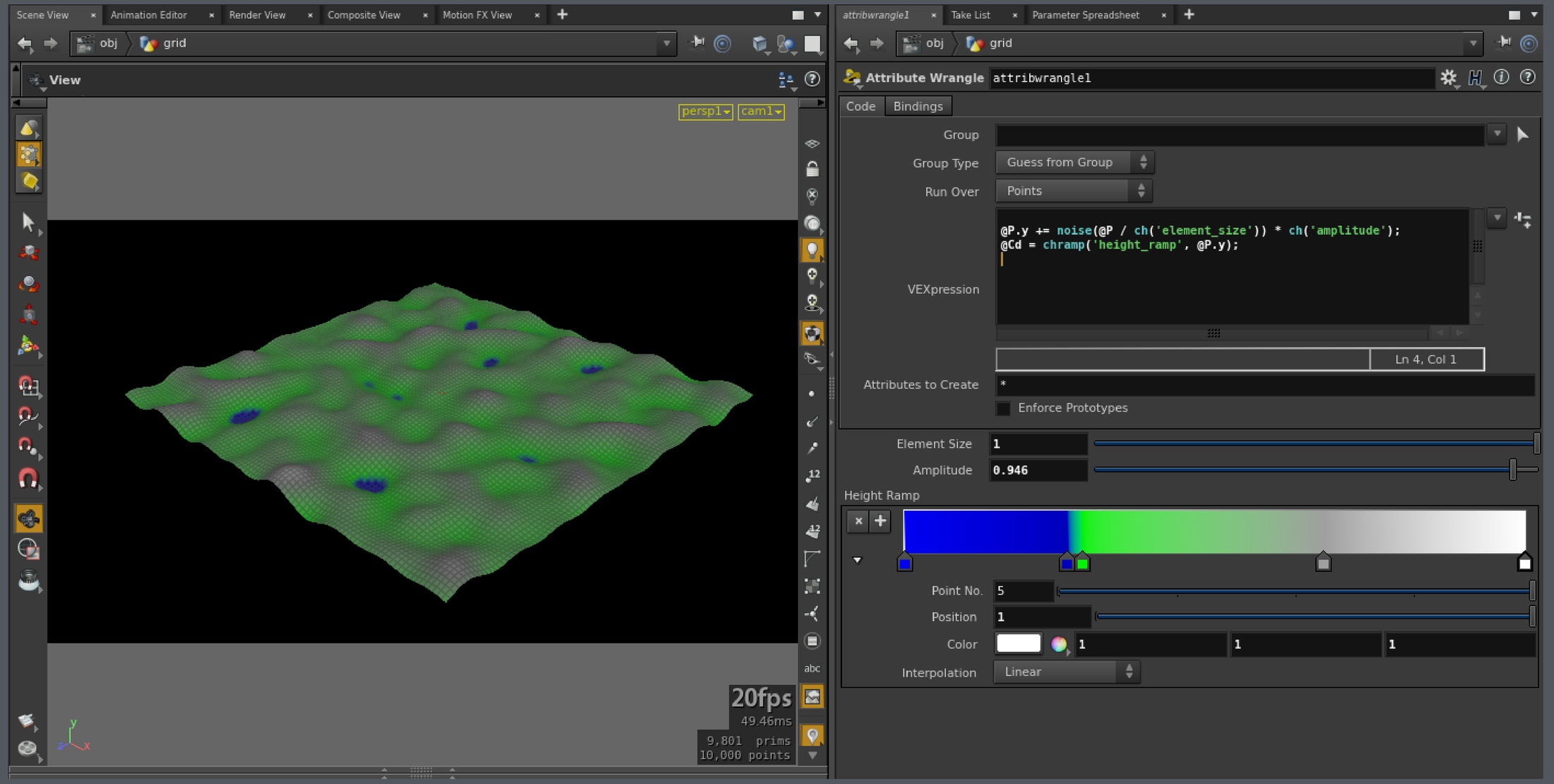
```
sop
mountain(int frac_depth=3;              // Fractal depth
         float rough=.6;                // Fractal roughness
         float height=1;                // Height amount

         vector freq=1;                 // Noise Frequency
         vector offset=0;               // Noise Offset
         string ntype="perlin";
         int do_color=0;
         float clr_off=0.4;
         float clr_scale=1;
    )
{
    int         i;
    float       scale;
    float       nval;
    vector      PP = (P + offset) * freq;

    nval = 0;
    scale = height;
    for (i = frac_depth; i-- > 0; scale *= rough)
    {
            if (ntype == "alligator")  nval += scale*anoise(PP);
        else if (ntype == "sparse")    nval += scale*snoise(PP);
        else                           nval += scale*noise0(PP);
        PP *= 2;
    }

    // Move in the direction of the surface normal
    P += normalize(N)*nval;

    if (do_color)
    {
        nval = (nval+clr_off)*clr_scale;
        if (nval < 0)
        {
            Cd = cspline(-nval,
                         { 0.1, 0.8, 1.0 },
                         { 0.1, 0.8, 1.0 },
                         { 0.1, 0.7, 1.0 },
                         { 0.1, 0.5, 1.0 },
                         { 0.1, 0.2, 1.0 },
                         { 0.1, 0.1, 1.0 },
                         { 0.1, 0.1, 0.7 },
                         { 0.1, 0.1, 0.3 },
                         { 0.1, 0.1, 0.3 });
        }
        else
        {
            Cd = cspline(nval,
                         { 0.4, 0.8, 0.2 },
                         { 0.4, 0.8, 0.2 },
                         { 0.5, 0.7, 0.2 },
                         { 0.5, 0.5, 0.2 },
                         { 0.5, 0.2, 0.5 },
                         { 0.7, 0.1, 0.6 },
                         { 0.8, 0.1, 0.7 },
                         { 0.8, 0.8, 0.8 },
                         { 0.8, 0.8, 0.8 });
        }
    }
}
```

Why Kernel Languages?

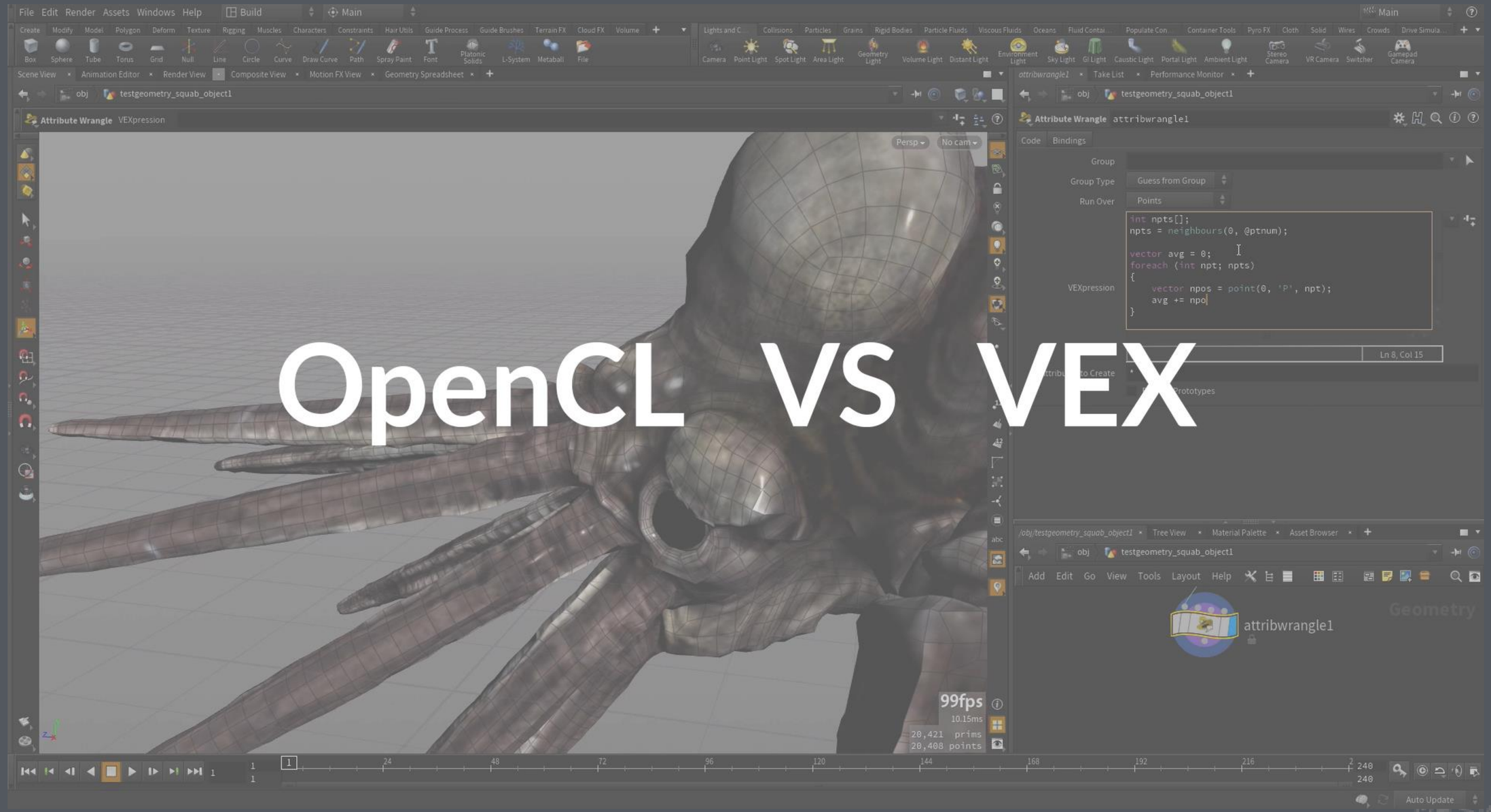


Houdini

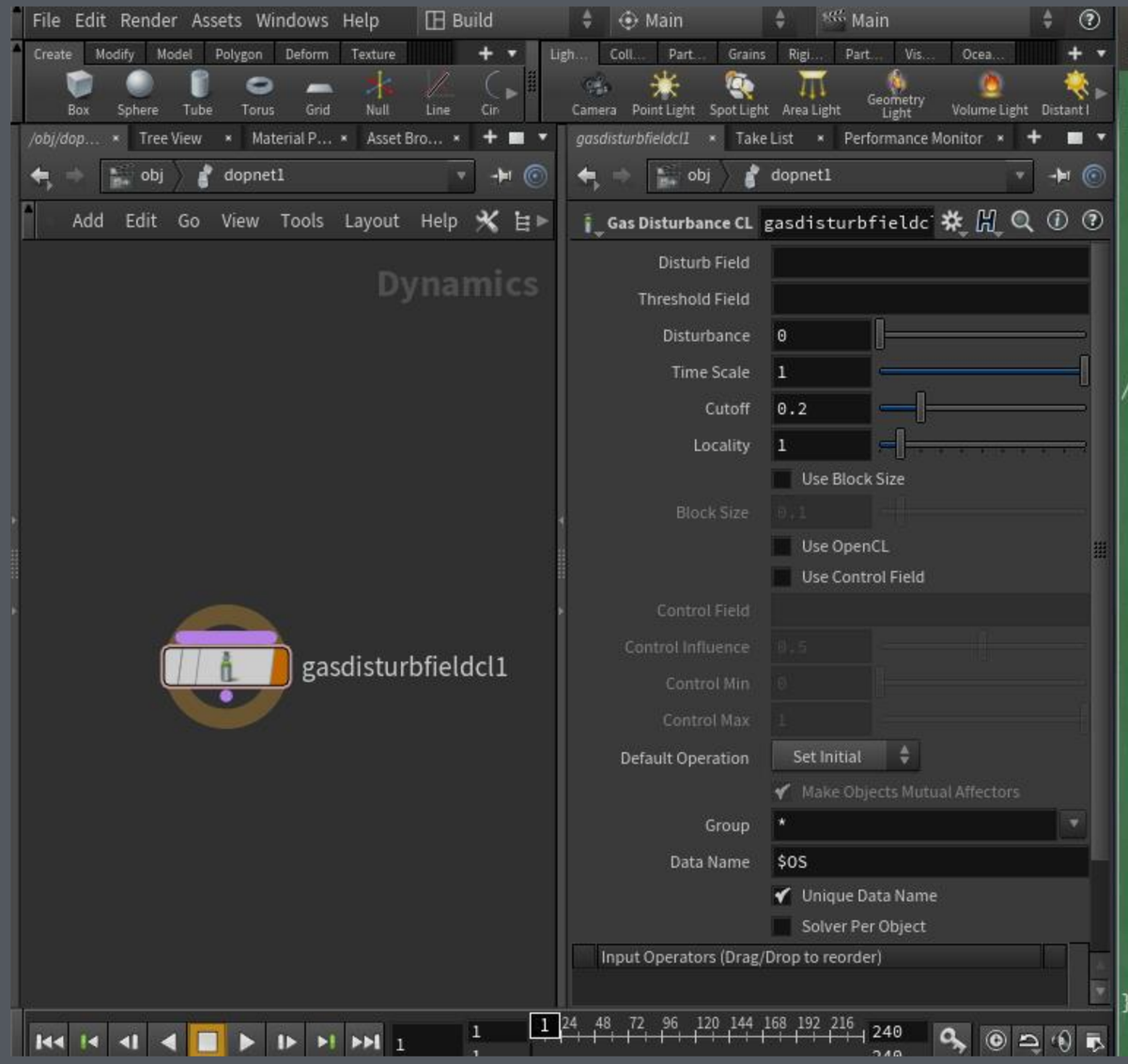
Snippet Approach

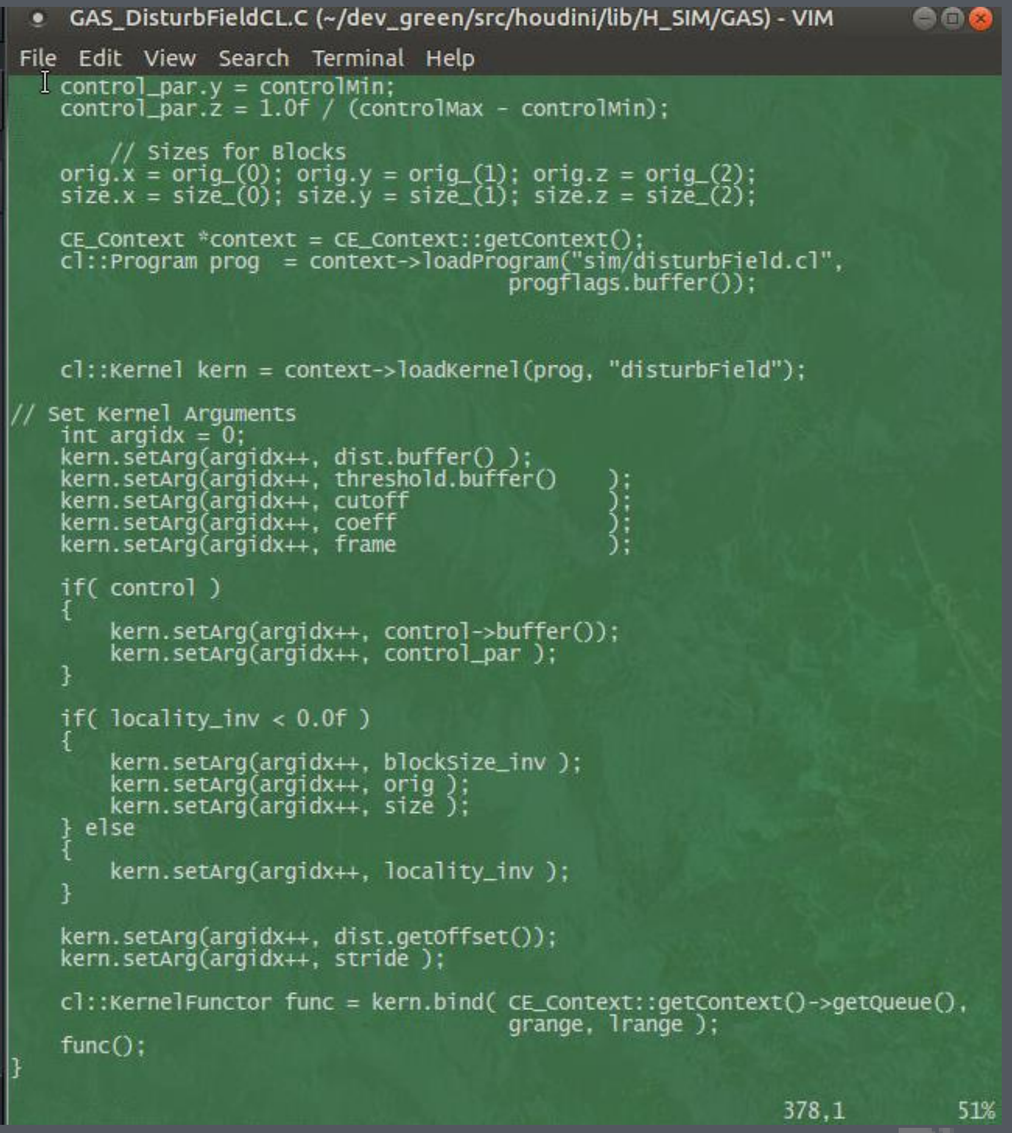


Need for OpenCL



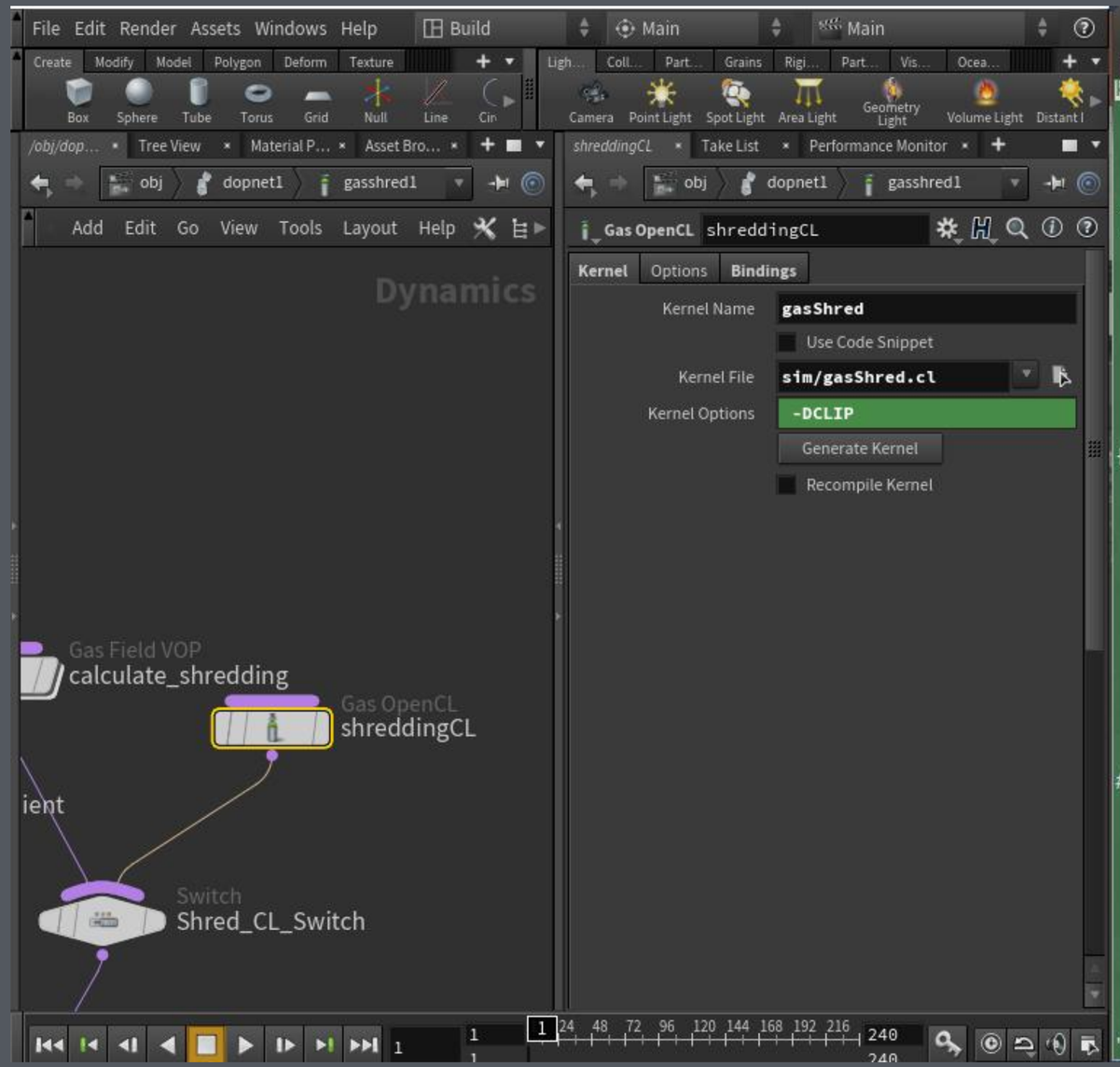
OpenCL in Houdini: Dedicated Nodes





SideFX

OpenCL in Houdini: Kernel Files



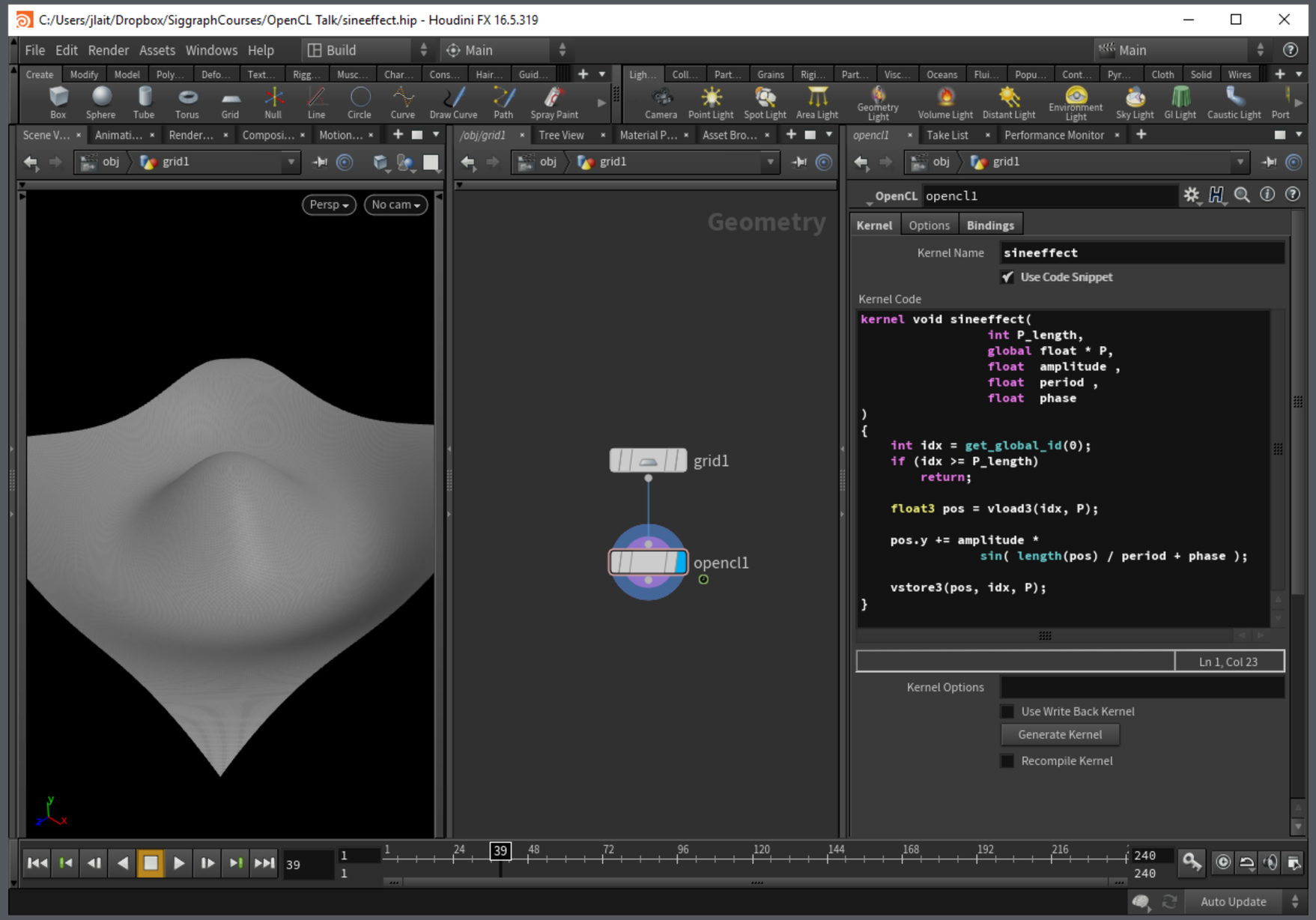
gasShred.cl (~/dev_green/src/houdini/support/ocl/sim) - VIM

File Edit View Search Terminal Help

```
kernel void gasShred(
                int stride_x,
                int stride_y,
                int stride_z,
                int stride_offset,
                float  PushPullThreshold ,
                float  PushAmount ,
                float  PullAmount ,
                float  Fade ,
                float  ClipValue ,
                float  ControlInfluence ,
                float  ControlMax ,
                float  ControlMin ,
                global float * TemperatureField ,
                global float * GradientField_1 ,
                global float * GradientField_2 ,
                global float * GradientField_3 ,
                global float * ControlField )
{
    int gidx = get_global_id(0);
    int gidy = get_global_id(1);
    int gidz = get_global_id(2);

    int idx = stride_offset + stride_x * gidx
                            + stride_y * gidy
                            + stride_z * gidz;

    float grad1         = GradientField_1[idx];
    float grad2         = GradientField_2[idx];
    float grad3         = GradientField_3[idx];
    float temperature   = TemperatureField[idx];
    float scaler;
    float sum;

#ifdef CLIP
    // normalize gradient
    float grad_norm = sqrt( grad1 * grad1 + grad2 * grad2 + grad3 * grad3 );
    grad_norm = max( 1e-6f, grad_norm );

    grad1 = grad1 / grad_norm;
    grad2 = grad2 / grad_norm;
    grad3 = grad3 / grad_norm;

    // Clamp gradient length
    grad_norm = clamp(grad_norm, 0.0f, clipvalue);

    // Restore gradient
"gasshred.cl" 92L, 2695C                                    1,1           Top
```

Houdini

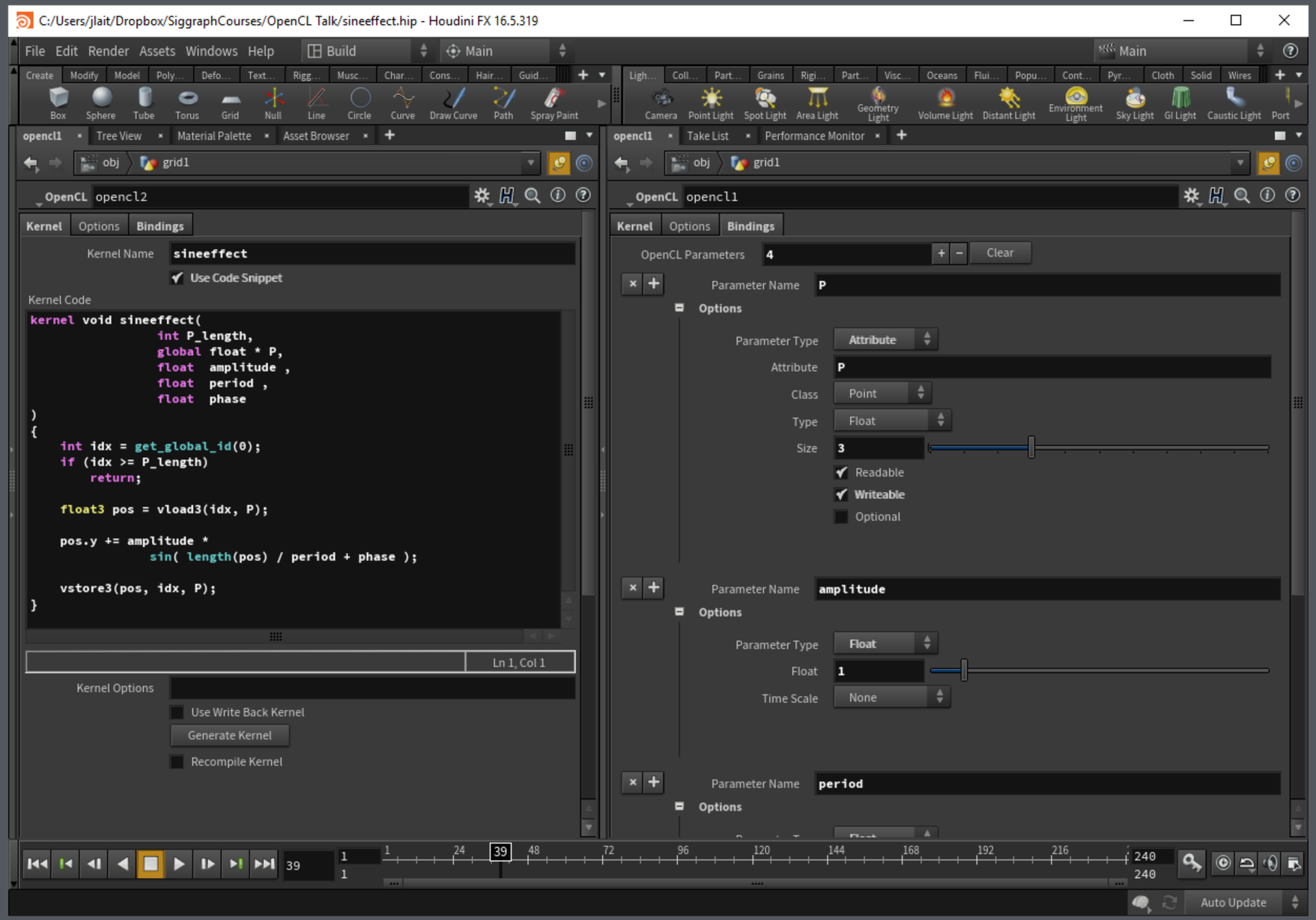
OpenCL Snippet Nodes



SideFX

Houdini

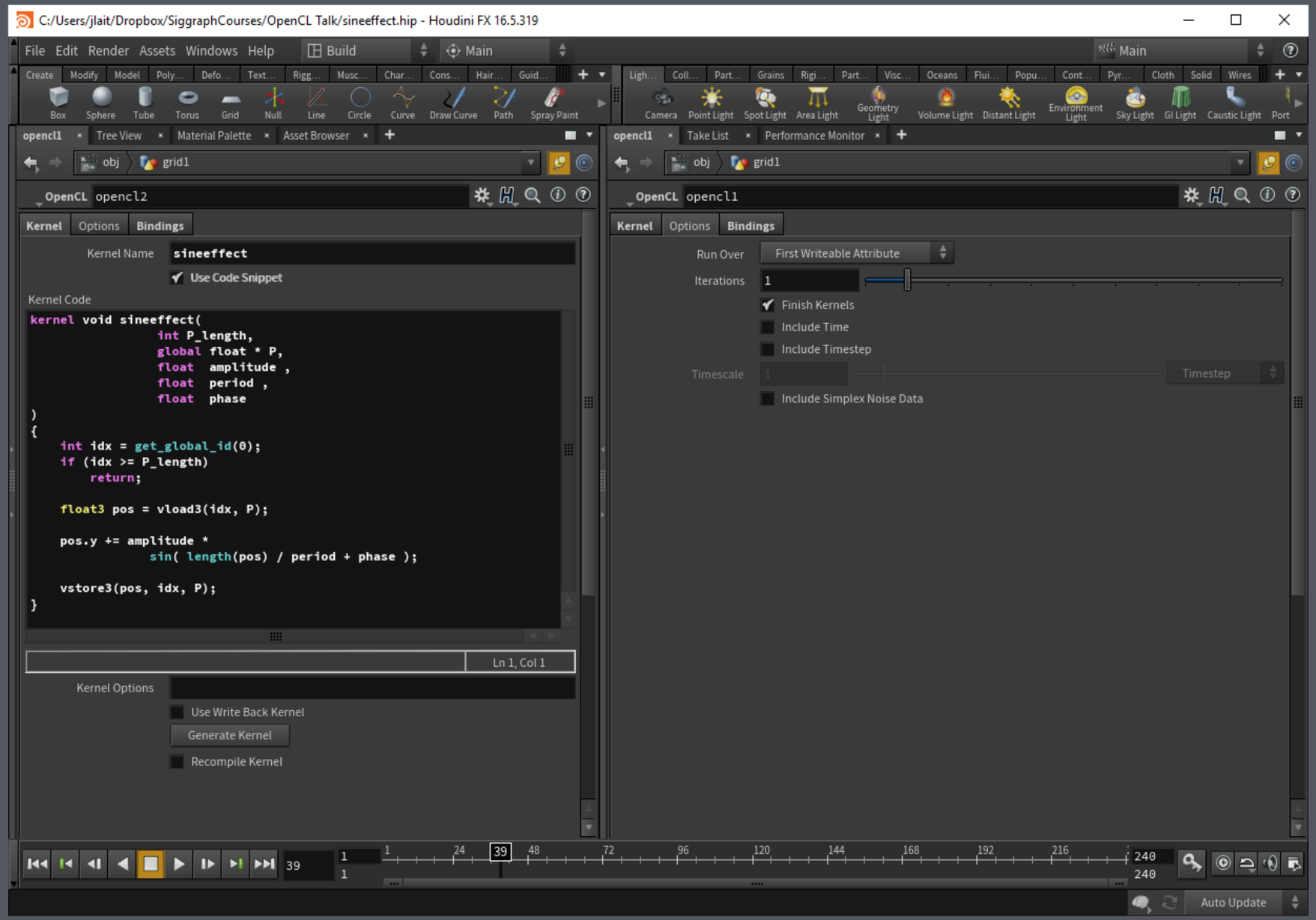
Attribute and Parameter Binding



SideFX

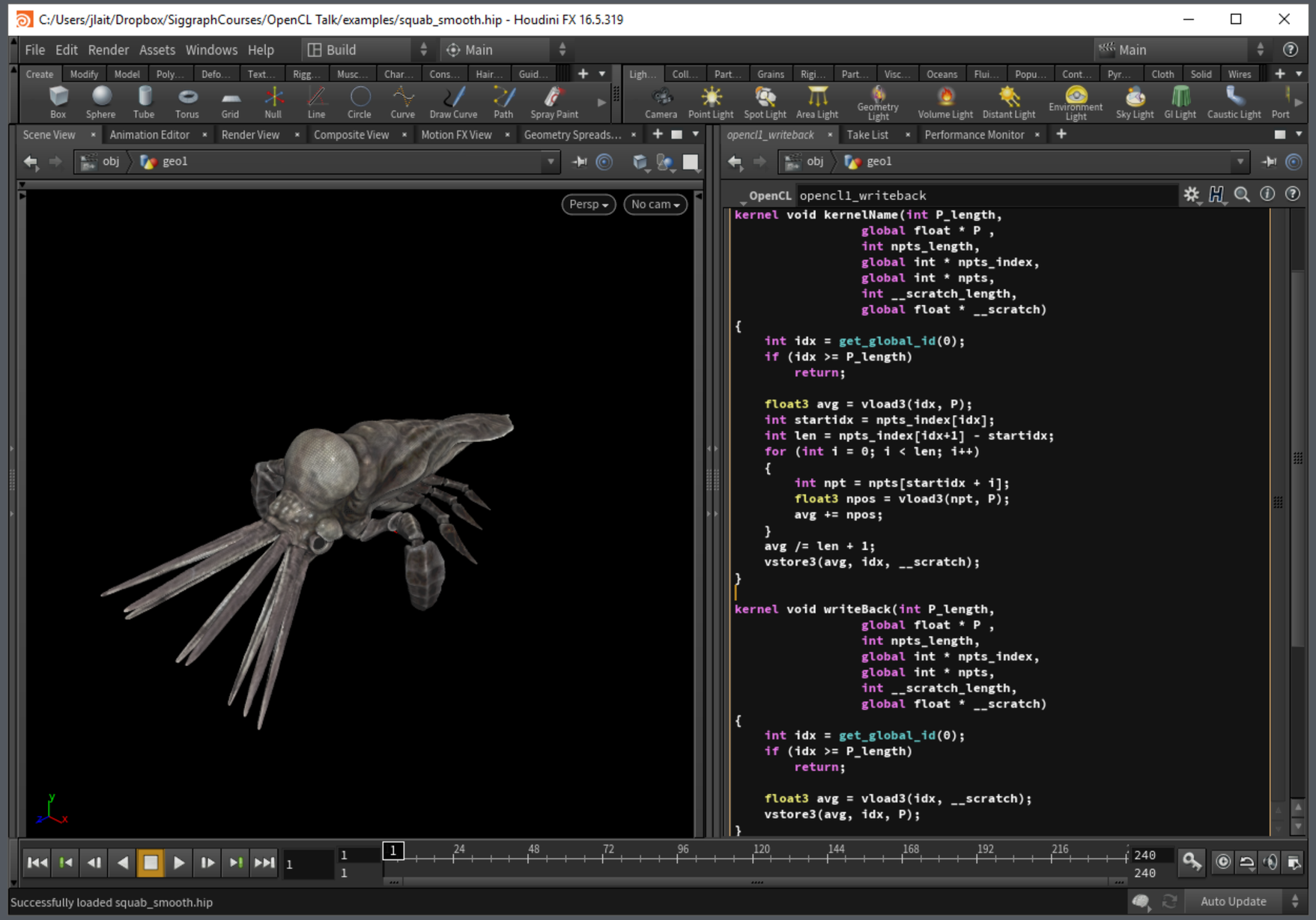
Houdini

Run-Over Selection

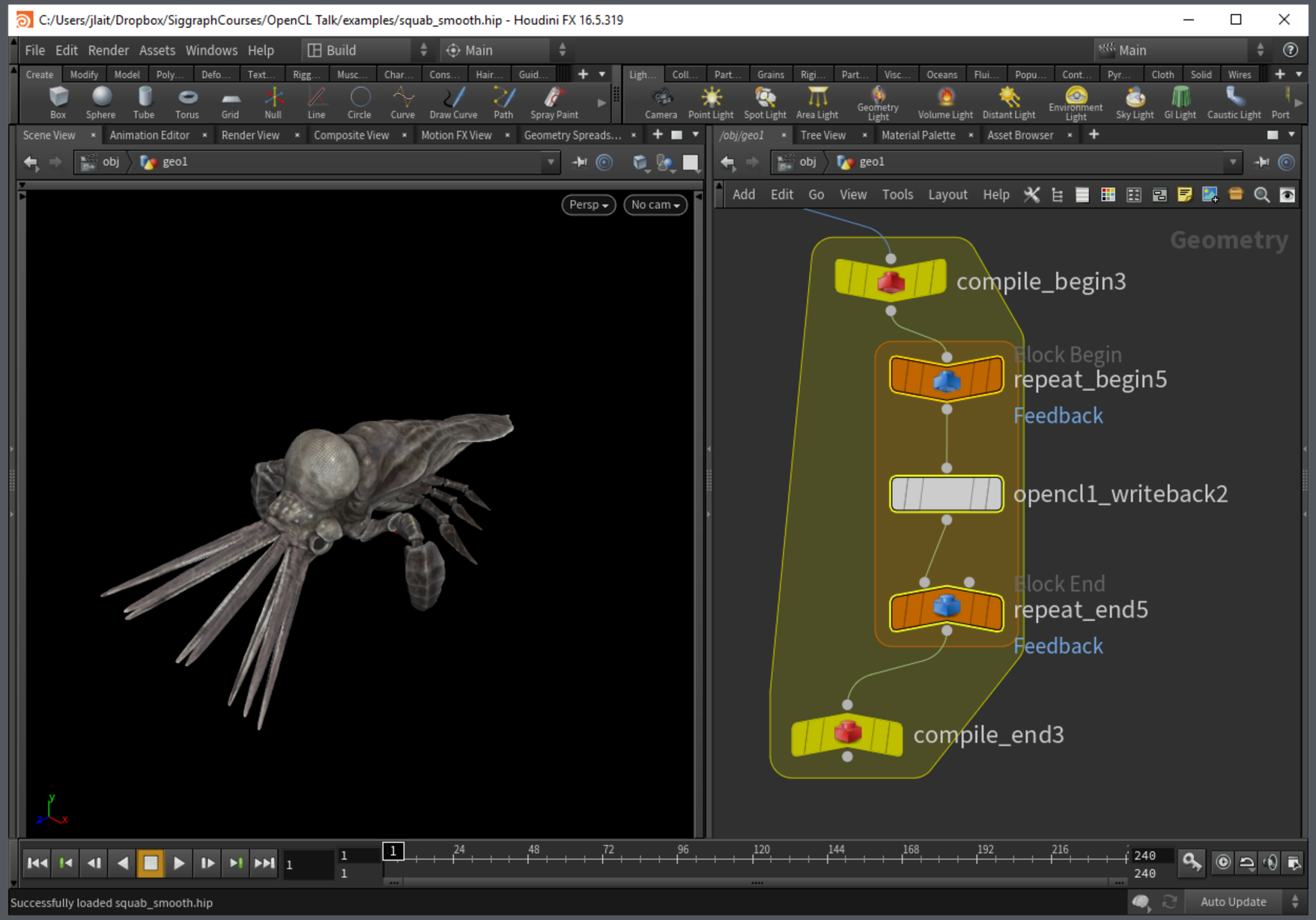


SideFX

Writeback Kernels

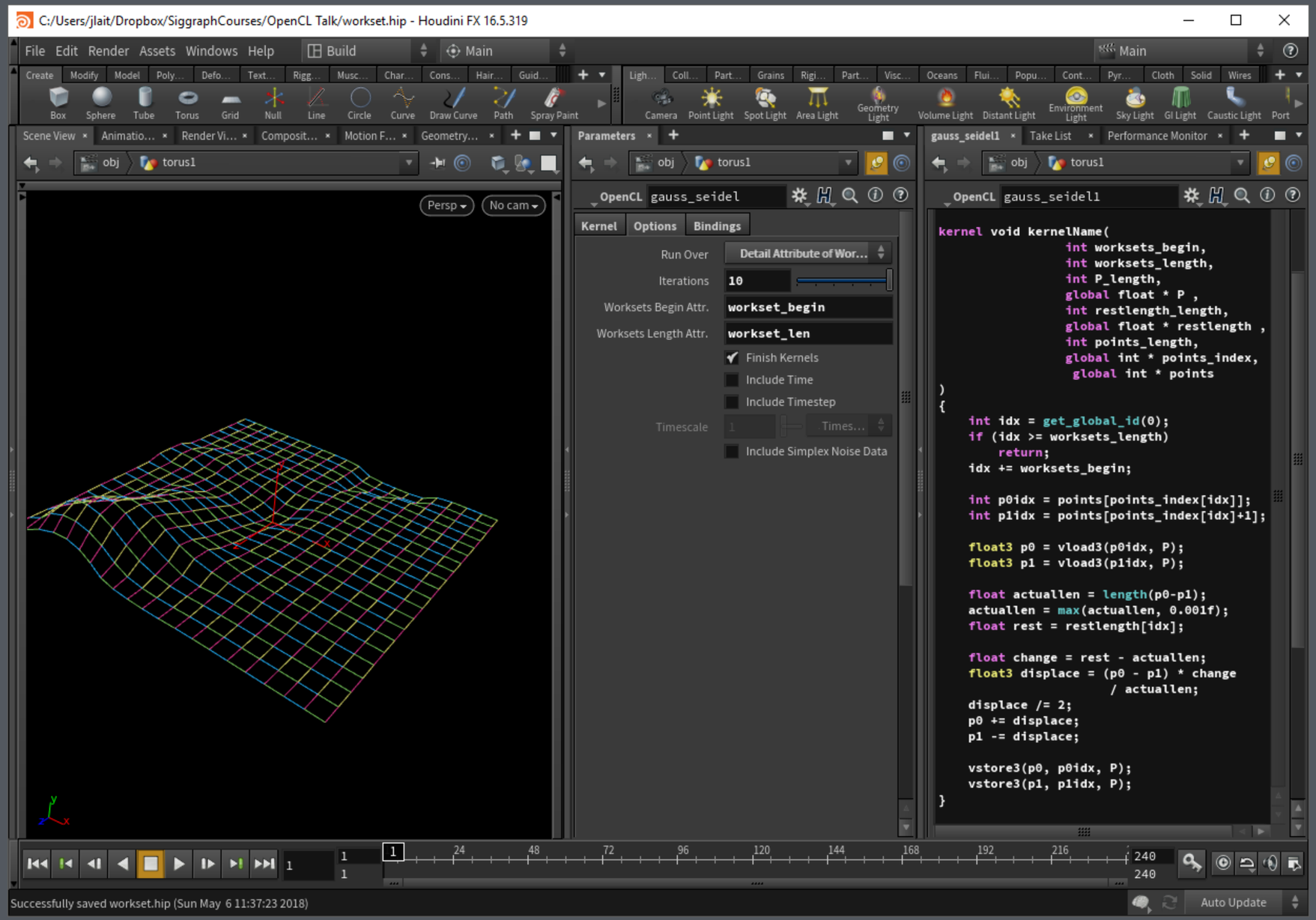


Keeping Data on GPU



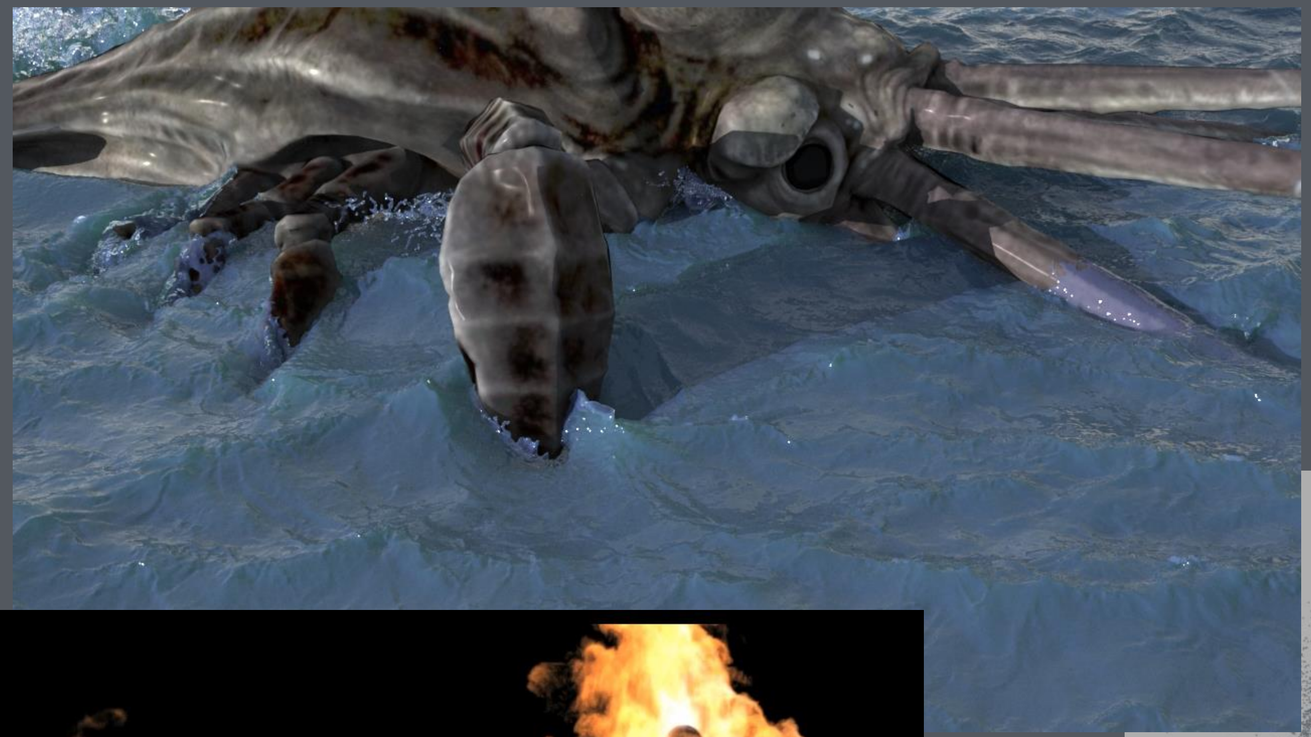
Keeping Data on GPU

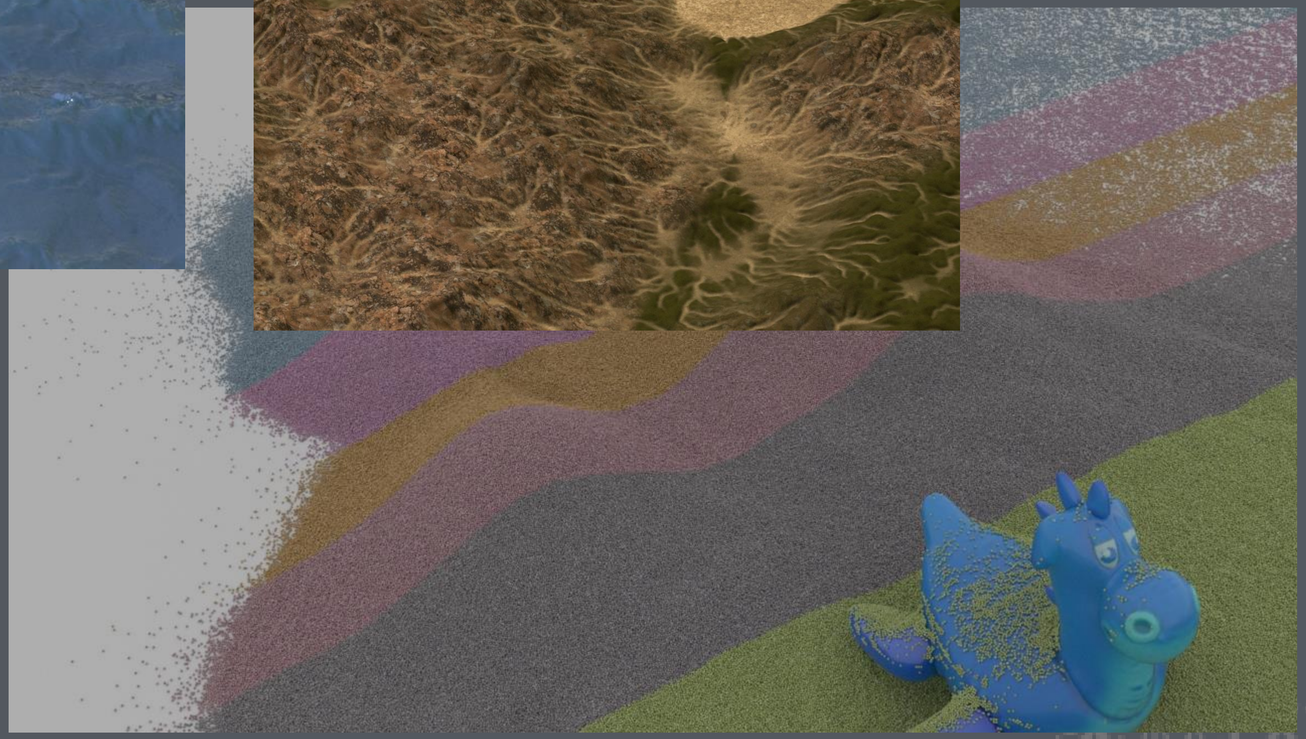
```
    /// @{
    /// The attribute data may be moved to the GPU, and hence backed
    /// by a compute buffer (GA_CEAttribute).  Flushing will delete
    /// our buffer & copy it back from the GPU if it has been marked
    /// as modified.

    /// Caches a CE Attribute.  If a cache already exists, the cached
    /// data is returned and the write flag updated.  If the cache
    /// does not exist, an attempt is made to build a buffer.  If
    /// read is true, the buffer is initialized with the geometry data.
    /// This can return NULL if a failure to build an attribute occurs
    /// (for example, unsupported type or no GPU)
    GA_CEAttribute      *getCEAttribute(GA_StorageClass storage, int &tuplesize,
bool isarray, bool read, bool write);

    /// Any CE cache which was marked as written to will be copied
    /// back to the CPU.  It will be left on the GPU, however.
    void                flushCEWriteCaches(bool clearwriteback=true);
    bool                hasPendingCEWriteBack() const;

    /// Remove all CE Caches, copying back any marked as written to.
    void                flushCECaches();

    /// @}
```


C:/Users/jlait/Dropbox/SiggraphCourses/OpenCL Talk/workset.hip - Houdini FX 16.5.319
File Edit Render Assets Windows Help
Build
Main
Box Sphere Tube Torus Grid Null Line Circle Curve Draw Curve Path Spray Paint
Camera Point Light Spot Light Area Light Geometry Light Volume Light Distant Light Environment Light Sky Light GI Light Caustic Light
Scene View
Persp
No cam
obj
torus1
Parameters
OpenCL gauss_seidel
Kernel Options Bindings
Run Over Detail Attribute of Wor...
Iterations 10
Worksets Begin Attr. workset_begin
Worksets Length Attr. workset_len
Finish Kernels
Include Time
Include Timestep
Timescale
Include Simplex Noise Data
gauss_seidel1
Take List
Performance Monitor
OpenCL gauss_seidel1
kernel void kernelName(
int worksets_begin,
int worksets_length,
int P_length,
global float * P ,
int restlength_length,
global float * restlength ,
int points_length,
global int * points_index,
global int * points
)
{
int idx = get_global_id(0);
if (idx >= worksets_length)
return;
idx += worksets_begin;
int p0idx = points[points_index[idx]];
int p1idx = points[points_index[idx]+1];
float3 p0 = vload3(p0idx, P);
float3 p1 = vload3(p1idx, P);
float actuallen = length(p0-p1);
actuallen = max(actuallen, 0.001f);
float rest = restlength[idx];
float change = rest - actuallen;
float3 displace = (p0 - p1) * change
/ actuallen;
displace /= 2;
p0 += displace;
p1 -= displace;
vstore3(p0, p0idx, P);
vstore3(p1, p1idx, P);
}
Successfully saved workset.hip (Sun May 6 11:37:23 2018)
Auto Update

Houdini

OpenCL Use in Houdini



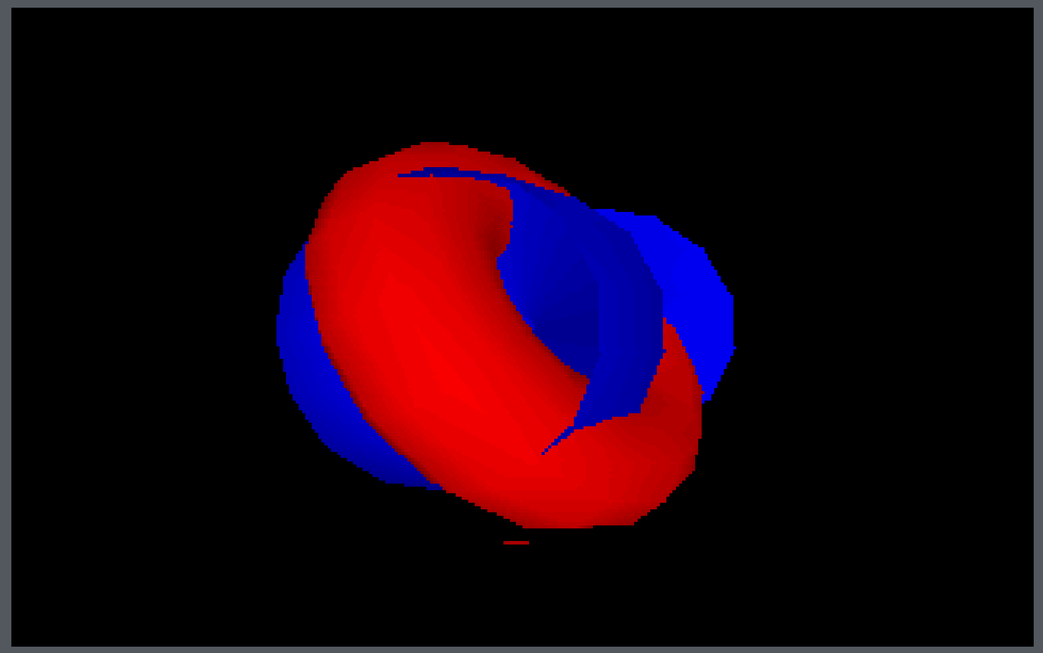


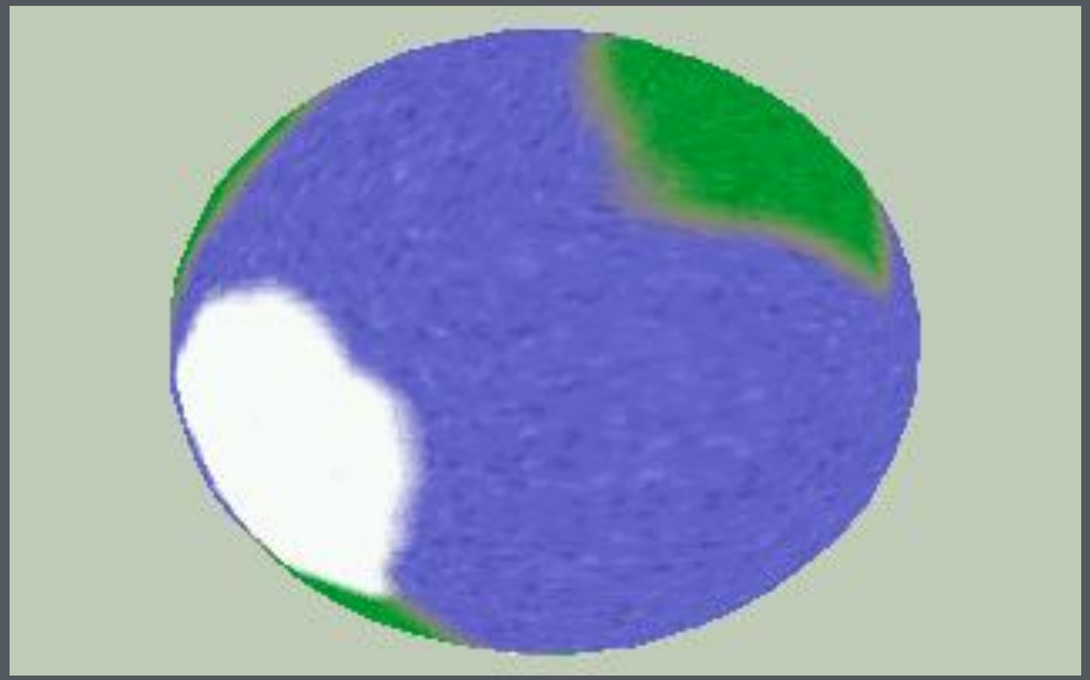
What I Want in Kernel Languages

```
kernel void smooth(int P_length,
                   global float * P ,
                   int npts_length,
                   global int * npts_index,
                   global int * npts,
                   int __scratch_length,
                   global float * __scratch)
{
    int idx = get_global_id(0);
    if (idx >= P_length)
        return;

    float3 avg = 0;
    int startidx = npts_index[idx];
    int len = npts_index[idx+1] - startidx;
    for (int i = 0; i < len; i++)
    {
        int npt = npts[startidx + i];
        float3 npos = vload3(npt, P);
        avg += npos;
    }
    avg /= len;
    vstore3(avg, idx, __scratch);
}
```

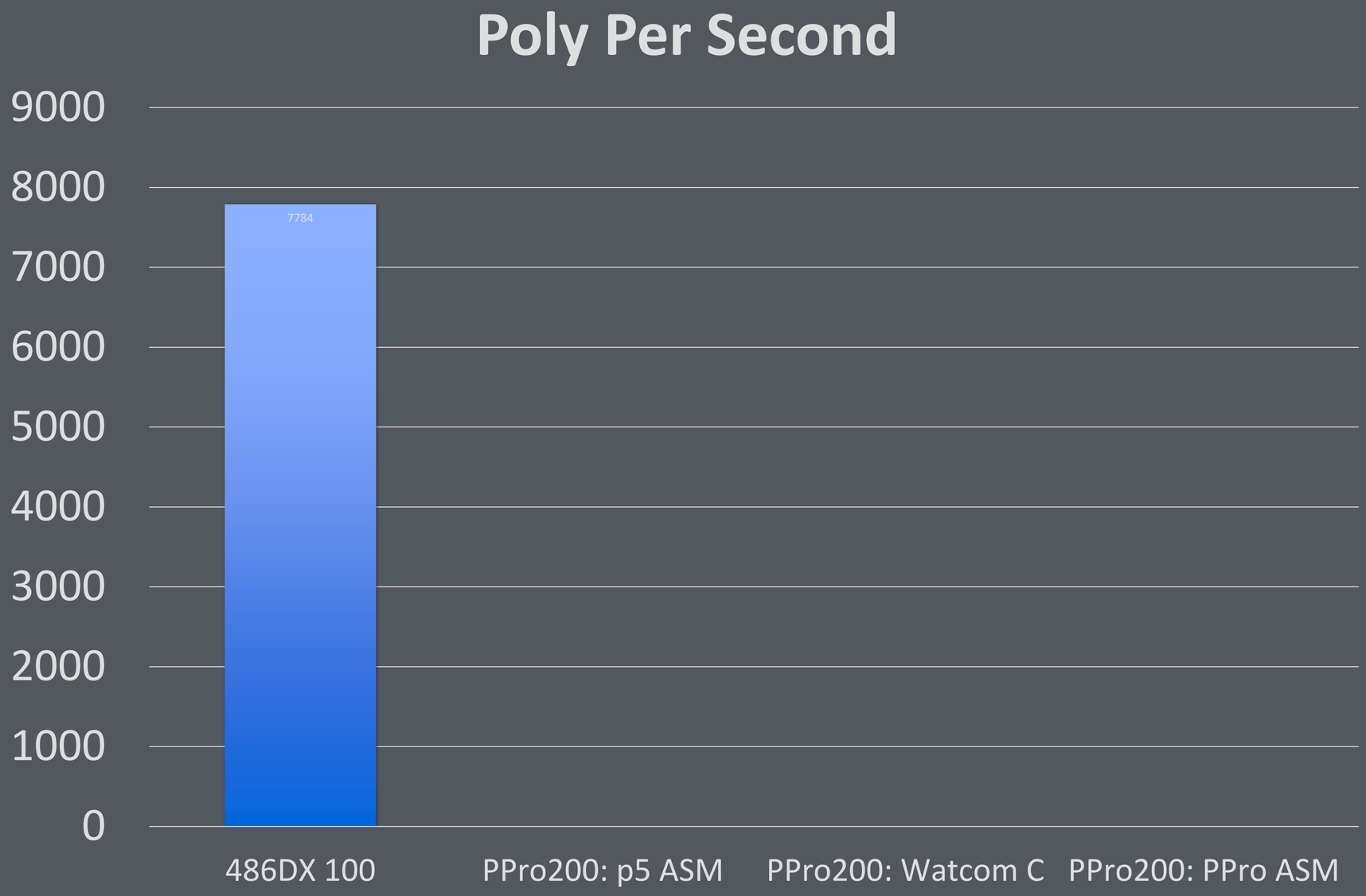
1. Performance Portable

Performance Portability: My History

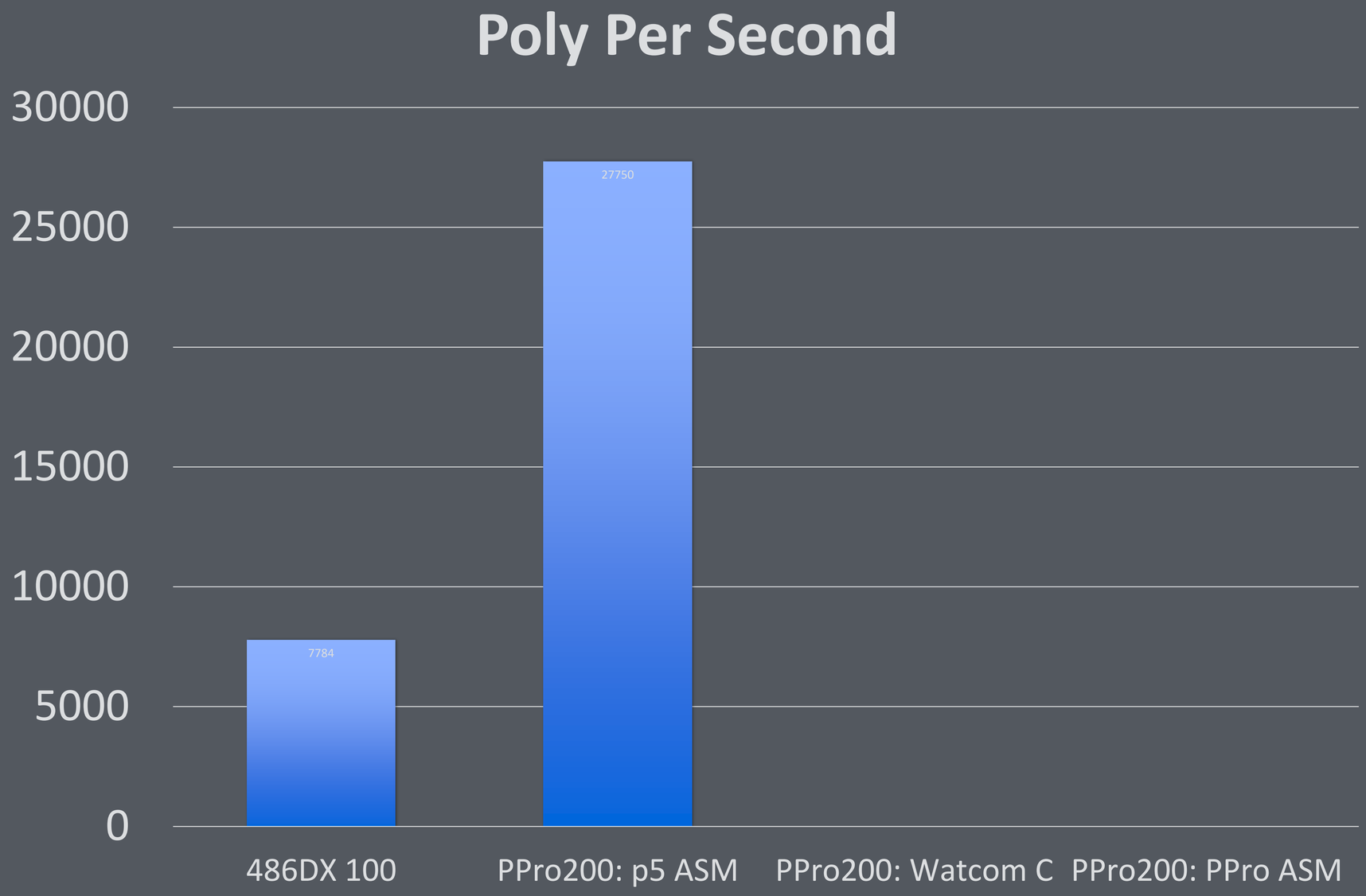




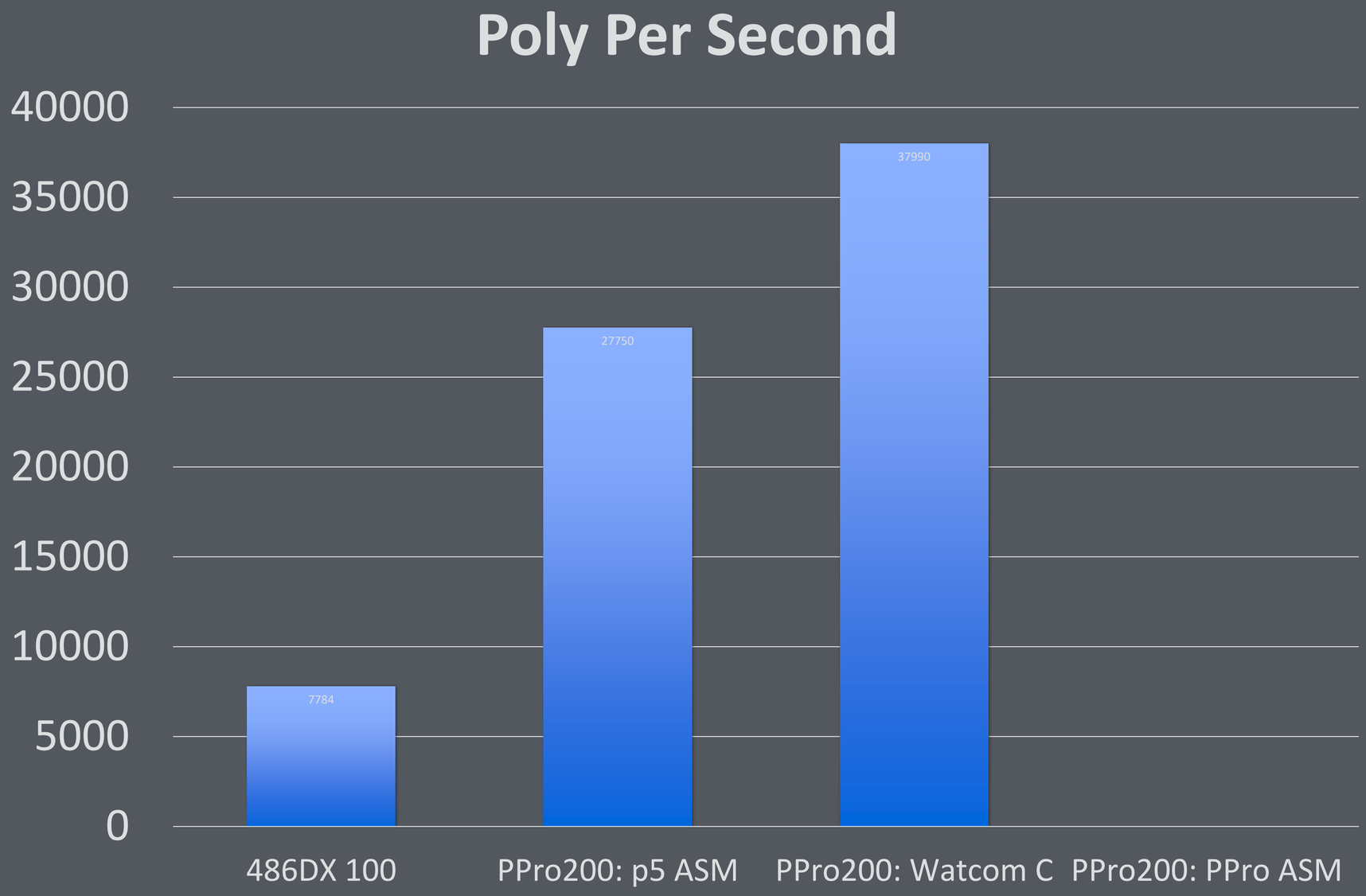
Performance Portability: My History



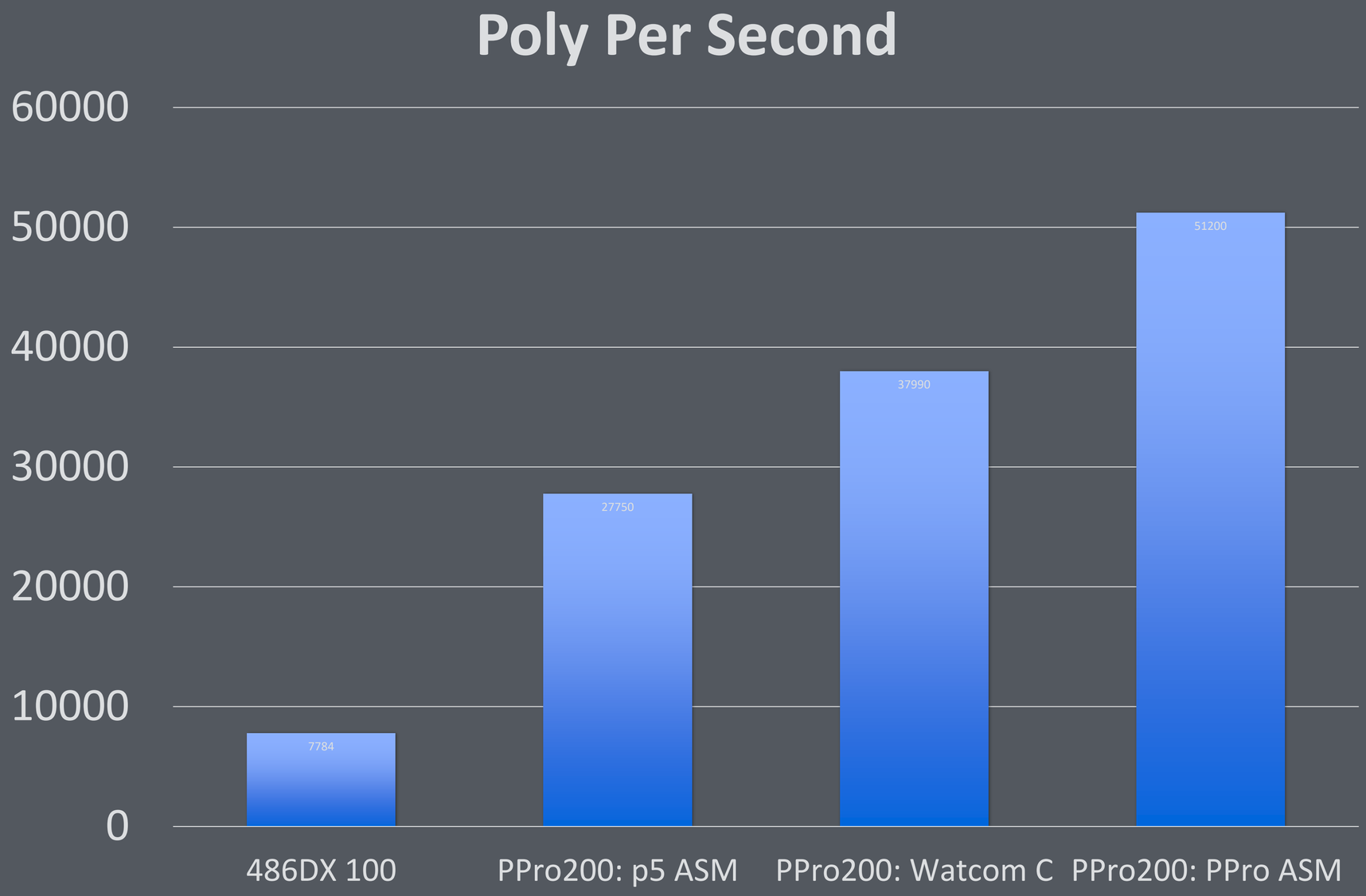
Performance Portability: My History



Performance Portability: My History



Performance Portability: My History



What I Want in Kernel Languages

```
kernel void smooth(int P_length,
                   global float * P ,
                   int npts_length,
                   global int * npts_index,
                   global int * npts,
                   int __scratch_length,
                   global float * __scratch)
{
    int idx = get_global_id(0);
    if (idx >= P_length)
        return;

    float3 avg = 0;
    int startidx = npts_index[idx];
    int len = npts_index[idx+1] - startidx;
    for (int i = 0; i < len; i++)
    {
        int npt = npts[startidx + i];
        float3 npos = vload3(npt, P);
        avg += npos;
    }
    avg /= len;
    vstore3(avg, idx, __scratch);
}
```

1. Performance Portable

What I Want in Kernel Languages

```
kernel void smooth(int P_length,
                   global float * P ,
                   int npts_length,
                   global int * npts_index,
                   global int * npts,
                   int __scratch_length,
                   global float * __scratch)
{
    int idx = get_global_id(0);
    if (idx >= P_length)
        return;

    float3 avg = 0;
    int startidx = npts_index[idx];
    int len = npts_index[idx+1] - startidx;
    for (int i = 0; i < len; i++)
    {
        int npt = npts[startidx + i];
        float3 npos = vload3(npt, P);
        avg += npos;
    }
    avg /= len;
    vstore3(avg, idx, __scratch);
}
```

1. Performance Portable
2. Default Initializers

What I Want in Kernel Languages

```
kernel void smooth(int P_length,
                   global float * P ,
                   int npts_length,
                   global int * npts_index,
                   global int * npts,
                   int __scratch_length,
                   global float * __scratch)
{
    int idx = get_global_id(0);
    if (idx >= P_length)
        return;

    float3 avg = 0;
    int startidx = npts_index[idx];
    int len = npts_index[idx+1] - startidx;
    for (int i = 0; i < len; i++)
    {
        int npt = npts[startidx + i];
        float3 npos = vload3(npt, P);
        avg += npos;
    }
    avg /= len;
    vstore3(avg, idx, __scratch);
}
```

1. Performance Portable
2. Default Initializers
3. Safe Operations

What I Want in Kernel Languages

```
kernel void tumbleMaterial(
    float flow_rate,
    float repose_angle,
    float height_factor,
    int stride_x,
    int stride_y,
    int stride_z,
    int stride_offset,
    float voxelsize_x,
    float voxelsize_y,
    float voxelsize_z,
    global float * height,
    global float * material,
#ifdef CALC_FLOW
    global float * flow_x,
    global float * flow_y,
#endif
    global float * write_back
#ifdef HAS_entrained
    , global float * entrained
#endif
    , global float * write_back_entrained
    , float entrainmentrate
    , int openborder
    )
{
    int gidx = get_global_id(0);
    int gidy = get_global_id(1);
    int gidz = get_global_id(2);

    int idx = stride_offset + stride_x * gidx
                            + stride_y * gidy
                            + stride_z * gidz;

    int bound_x = get_global_size(0);
    int bound_y = get_global_size(1);
```

1. Performance Portable
2. Default Initializers
3. Safe Operations
4. Optional Bindings

What I Want in Kernel Languages

```
kernel void tumbleMaterial(
    float flow_rate,
    float repose_angle,
    float height_factor,
    int stride_x,
    int stride_y,
    int stride_z,
    int stride_offset,
    float voxelsize_x,
    float voxelsize_y,
    float voxelsize_z,
    global float * height,
    global float * material,
#ifdef CALC_FLOW
    global float * flow_x,
    global float * flow_y,
#endif
    global float * write_back
#ifdef HAS_entrained
    , global float * entrained
#endif
    , global float * write_back_entrained
    , float entrainmentrate
    , int openborder
    )
{
    int gidx = get_global_id(0);
    int gidy = get_global_id(1);
    int gidz = get_global_id(2);

    int idx = stride_offset + stride_x * gidx
                            + stride_y * gidy
                            + stride_z * gidz;

    int bound_x = get_global_size(0);
    int bound_y = get_global_size(1);
```

1. Performance Portable
2. Default Initializers
3. Safe Operations
4. Optional Bindings
5. Default Values

What I Want in Kernel Languages

```
kernel void tumbleMaterial(
    float flow_rate,
    float repose_angle,
    float height_factor,
    int stride_x,
    int stride_y,
    int stride_z,
    int stride_offset,
    float voxelsize_x,
    float voxelsize_y,
    float voxelsize_z,
    global float * height,
    global float * material,
#ifdef CALC_FLOW
    global float * flow_x,
    global float * flow_y,
#endif
    global float * write_back
#ifdef HAS_entrained
    , global float * entrained
#endif
    , global float * write_back_entrained
    , float entrainmentrate
    , int openborder
    )
{
    int gidx = get_global_id(0);
    int gidy = get_global_id(1);
    int gidz = get_global_id(2);

    int idx = stride_offset + stride_x * gidx
                            + stride_y * gidy
                            + stride_z * gidz;

    int bound_x = get_global_size(0);
    int bound_y = get_global_size(1);
```

1. Performance Portable
2. Default Initializers
3. Safe Operations
4. Optional Bindings
5. Default Values
6. Parameter List Inspection


THANK YOU
SideFX®