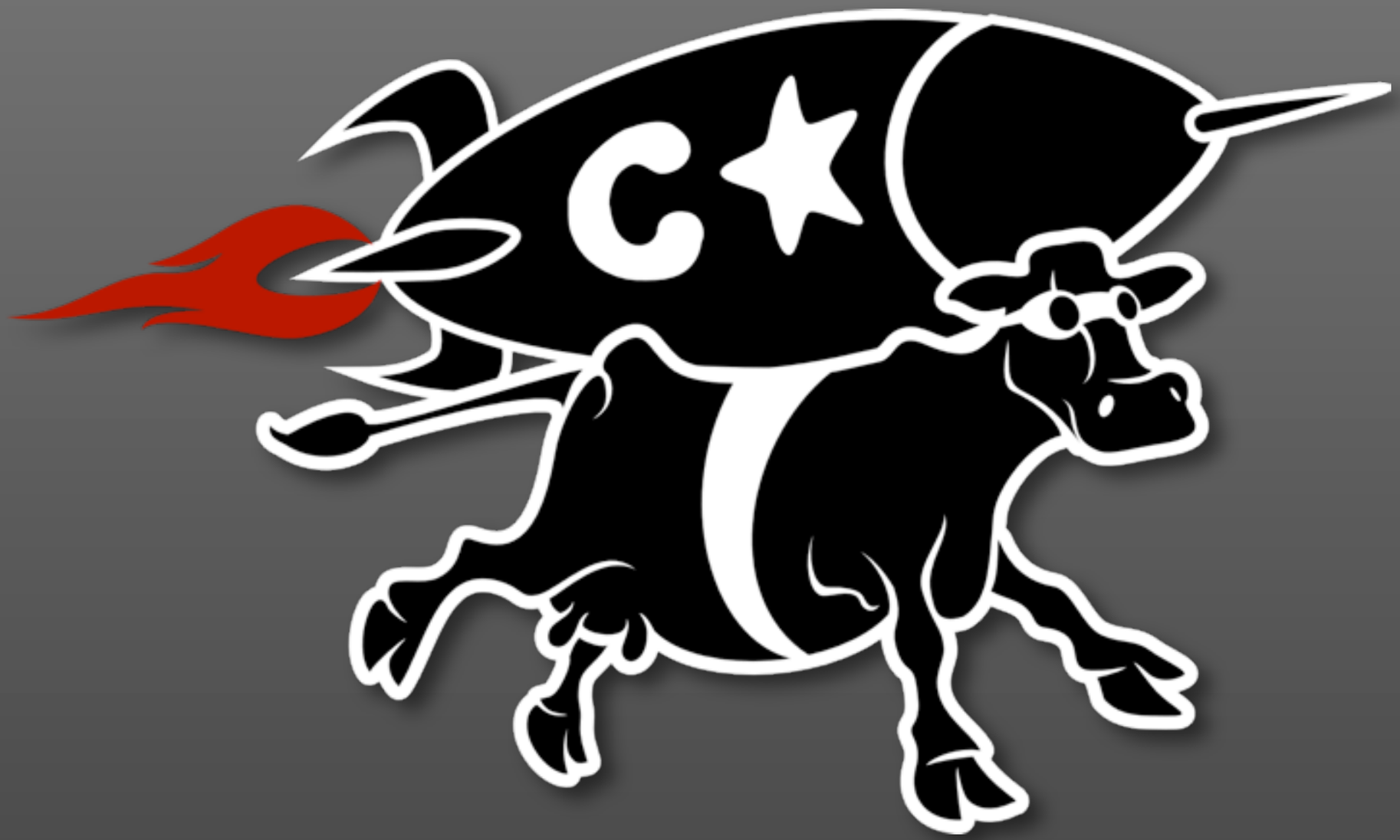

C

Cortex
Birds of a Feather

SIGGRAPH 2011

Overview

John Haddon, Image Engine

Lighting & rendering & fur

Andrew Kaufman, Image Engine

Crowd animation & rendering

Carsten Kolve, Dr. D Studios

Cool things Dan has been doing

Dan Bethell

Nuke integration

Paolo Berto, Jupiter Jazz

Oh No, Not Crowds Again

Ollie Rankin, the facility currently known as Method

Starting up with Cortex

Johannes Saam, Fuel

Future directions

John Haddon, Image Engine

Overview

John Haddon, Image Engine

What is Cortex?

- A foundation framework for VFX software development
- Cross application capable
- Well tested
- Open source
- An opportunity to collaborate

Python

Maya | Nuke | Houdini | RenderMan

Parameters, Ops, Procedurals

Data, Primitives, IO

STL, Boost, Imath, Additional Math

Parameters

- Provide input
- Strongly typed
- Perform validation
- Allow introspection
- Are mapped to and from host inputs
- Values are always native Cortex types

```
m_meshParameter = new MeshPrimitiveParameter(
    "mesh",
    "The mesh to distribute points over.",
    new MeshPrimitive()
);

m_densityParameter = new FloatParameter(
    "density",
    "The density of the distributed points.",
    100.0
);

m_offsetParameter = new V2fParameter(
    "offset",
    "A UV offset for the PointDistribution",
    Imath::V2f( 0, 0 )
);

m_densityPrimVarNameParameter = new StringParameter(
    "densityPrimVarName",
    "The primitive variable to use as a density threshold.",
    "density"
);

m_pRefPrimVarNameParameter = new StringParameter(
    "pRefPrimVarName",
    "The primitive variable that holds the reference positions.",
    "Pref"
);

m_uPrimVarNameParameter = new StringParameter(
    "uPrimVarName",
    "The primitive variable for u coordinates.",
    "s"
);
```

```
self.meshParameter = IECore.MeshPrimitiveParameter(
    "mesh",
    "The mesh to distribute points over.",
    IECore.MeshPrimitive()
)

self.densityParameter = IECore.FloatParameter(
    "density",
    "The density of the distributed points.",
    100.0
)

self.offsetParameter = IECore.V2fParameter(
    "offset",
    "A UV offset for the PointDistribution",
    IECore.V2f( 0, 0 ),
)

self.densityPrimVarNameParameter = IECore.StringParameter(
    "densityPrimVarName",
    "The primitive variable to use as a density threshold.",
    "density"
)

self.pRefPrimVarNameParameter = IECore.StringParameter(
    "pRefPrimVarName",
    "The primitive variable that holds the reference positions.",
    "Pref"
)

self.uPrimVarNameParameter = IECore.StringParameter(
    "uPrimVarName",
    "The primitive variable for u coordinates.",
    "s"
)
```

Attribute Editor

List Selected Focus Attributes Show Help

visualiser | visualiserShape | pointDistribution | initialShadingGroup

ieOpHolderNode: pointDistribution

Focus
Presets
Show Hide

Class

Parameters

Mesh **pTorusShape1.outMesh**

Density 0.1000

Offset 0.0000 0.0000

Density Prim Var Name density

P Ref Prim Var Name Pref

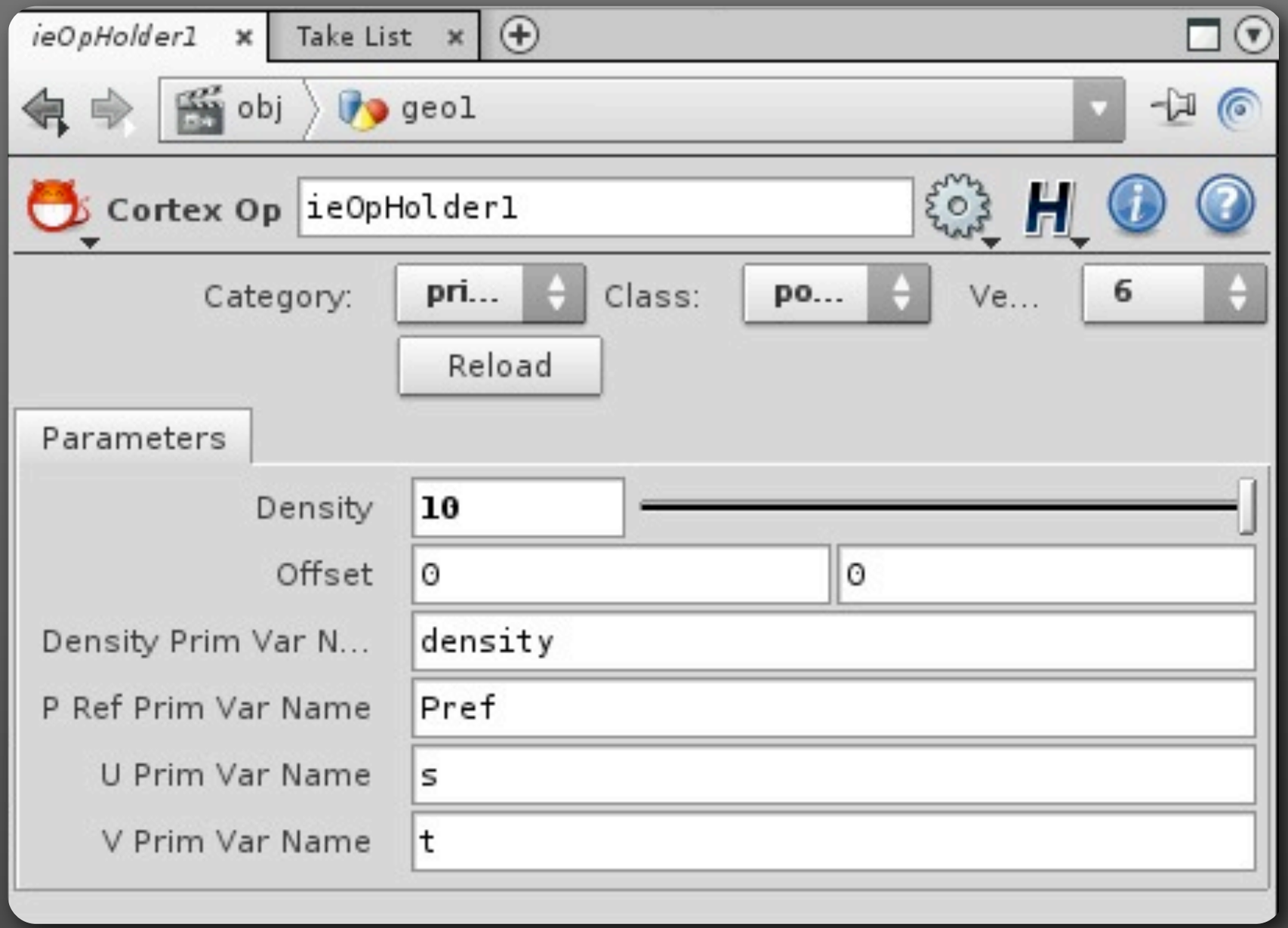
U Prim Var Name s

V Prim Var Name t

Extra Attributes

Notes: pointDistribution

Select | Load Attributes | Copy Tab | Close


ieOpHolder1
Take List
obj
geo1
Cortex Op
ieOpHolder1
Category:
pri...
Class:
po...
Ve...
6
Reload
Parameters
Density
10
Offset
0
0
Density Prim Var N...
density
P Ref Prim Var Name
Pref
U Prim Var Name
s
V Prim Var Name
t

```
-mesh read:myMesh.cob -density 10 -offset 0 0
```

Ops

- Glorified functions
- Take parameter inputs and compute a result
- Result described by output parameter
- Much Cortex functionality implemented as Ops
- Implemented in Python or C++

File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Cortex

Animation

View Shading Lighting Show Renderer Panels

Hypergraph InputOutput2

Edit View Bookmarks Graph Rendering Options Show Help

polyToru... pTorusS... pointDis... visualise... initialSh...

Traversal Depth: Unlimited

persp

2 4 6 8 10 12 14 16 18

1.00 1.00 1 24

MEL

Attribute Editor

List Selected Focus Attributes Show Help

visualiser visualiserShape pointDistribution initialShadingGroup

ieOpHolderNode: pointDistribution

Focus Presets Show Hide

Class

Parameters

Mesh pTorusShape1.outMesh

Density 0.1000

Offset 0.0000 0.0000

Density Prim Var Name density

P Ref Prim Var Name Pref

U Prim Var Name s

V Prim Var Name t

Extra Attributes

Notes: pointDistribution

Select Load Attributes Copy Tab Close

File Edit Render Windows Help IE

Auto Takes | Main

Create | Modify | Model | Poly... | Deform | Text... | Char... | Auto ... | Anim...

UVProject UVEdit UVBrush UVFuse UVPelt UVTexture UVTransfo... UVUnwrap UVQuicl

Ligh... | Crea... | Dri... | Rig... | Par... | Vol... | Pyr... | Cloth | Wires | Fur

Camera Point Light Spot Light Area Light Geometry... Distant Li... Environm... Sky Li

Scene View | Channel Editor | Render View | Composite View

obj / geo1

Select | (No Groups)

persp1 | no cam

ieOpHolder1 | Take List

obj / geo1

Cortex Op | ieOpHolder1

Category: pri... Class: po... Ve... 6

Reload

Parameters

Density	10	
Offset	0	0
Density Prim Var N...	density	
P Ref Prim Var Name	Pref	
U Prim Var Name	s	
V Prim Var Name	t	

Details View | Python Shell

obj / geo1

Node: ieOpHolder1 | View

	P[x]	P[y]	P[z]	P[w]
0	-3.99006	-0.216812	-8.73641	1
1	3.99302	-0.216812	-8.73641	1
2	3.99302	-0.216812	16.5491	1
3	-3.99006	-0.216812	16.5491	1
4	-3.99006	14.5074	-8.73641	1
5	3.99302	14.5074	-8.73641	1

/obj/geo1 | Tree View | Material Palette

obj / geo1

Geometry

1 24 48 72 96 120 144 168 192 216 240

Procedurals

- Deferred (render time) geometry creation
- “Black Box”
- Inputs described as parameters
- Output defined by calls to a renderer backend
- Backends provide an abstraction to allow support for multiple renderers
- Implemented in Python or C++

```
class SimpleProcedural( IECore.ParameterisedProcedural ) :

    ...

    def doBound( self, args ) :

        result = IECore.Box3f()
        for f in glob.glob( args["directory"].value + "/*.cob" ) :
            o = IECore.CachedReader.defaultCachedReader().read( f )
            result.extendBy( o.bound() )

        return result

    def doRender( self, renderer, args ) :

        for f in glob.glob( args["directory"].value + "/*.cob" ) :
            with IECore.AttributeBlock( renderer ) :
                renderer.setAttribute(
                    "name",
                    IECore.StringData( os.path.basename( f )
                )
                o = IECore.CachedReader.defaultCachedReader().read( f )
                o.render( renderer )
```

Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Cortex IE 3Delight Help

Shading Lighting Show Renderer Panels

At Bound Min
At Bound Max
At Bound Min And Max
At Bound Centre
At Transform Origin

persp

4.1 fps

Attribute Editor

List Selected Focus Attributes Show Help

vertCacheProceduralShape | time1 | initialShadingGroup

Focus
Presets
Show Hide

ieProceduralHolder: vertCacheProceduralShape

Class
Display
GI Preview
Draw Bound
Coordinate Systems
Transparent
Parameters
Object Directory: orkspace/testFiles/cjExample/cjModel
Animation Sequence: iple/animationCache/002/cj.######.fio
Frame: 379.0000
Render Stats
Extra Attributes

Notes: vertCacheProceduralShape

Select Load Attributes Copy Tab

340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 50

379

379.00

Lighting & Rendering

Andrew Kaufman, Image Engine

Lighting and Rendering

- Light in Maya, render in 3delight
- Use the 3delight plugin for rib generation. But...
- Geometry generated as procedurals
- Shading specified by procedurals
- Lights are custom

Geo caching

- Models stored in cob files (one per shape)
- Animation stored in fio files
  - Generally use vertex or transform caches
    - Could cache any PrimitiveVariables
- Effects exported from Houdini as cob/fio.
- Efficient loading/interpolation using InterpolatedCache

A modular procedural

- ClassParameters and ClassVectorParameters
  - Dynamically add functionality
- Geometry passed through stack of user defined modifiers
- Shader assignment
- Animation
- Attributes etc...

File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Cortex IE Help

Animation

View Shading Lighting Show Renderer Panels

FRONT

persp

4 fps

Attribute Editor

List Selected Focus Attributes Show Help

modularProcedural | modularProceduralShape | time1 | initialS

ieProceduralHolder: modularProceduralShape

Focus Presets Show Hide

Roughness 0.2500
Occlusion 1.0000
Category default
Aov aov_diffuse
Per Light AOV aov_diffuses
Color AOV aov_color
Affect Beauty
Color Texture
Class Cl
Specular
Subsurface
Master Opacity

Essentials
Procedural Text
Tbx
Utilities
Remove
Presets
Copy
Paste
Paste Linked

ler Distance
ler Grid
ler Noise
ler Stamper
ler Voronoi

Notes: modularProceduralShape

Select Load Attributes Copy Tab

110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 20

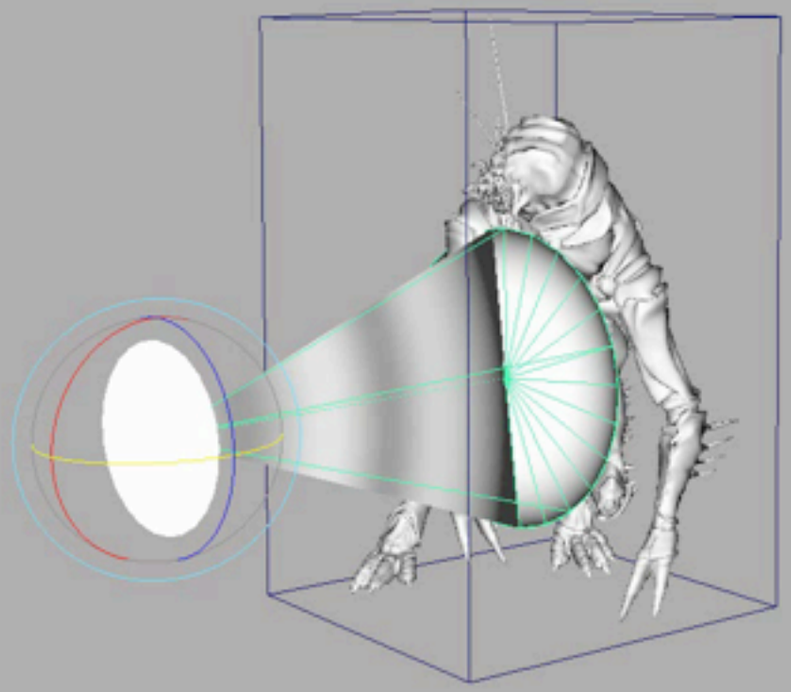
177.00

1.00 109.00 109 200 200.00 200.00 No Anim Layer No Character Set

MEL

File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Cortex IE 3Delight Help

View Shading Lighting Show Renderer Panels



persp

13.5 fps

Attribute Editor

List Selected Focus Attributes Show Help

spotLight | spotLightShape | timeToUnitConversion2

Focus

Color History

H: 0
S: 0.000
V: 1.000

HSV

Specular 1.0000
Specular Size 1.0000
Category default
Group 0
Scale 100.0000
Scale Light Disk 1.0000
Emit Photon
Shadow Mode Off

Ray Traced Shadows
Shadow Map Shadows
Decay
Radial Falloff
Gobo
Extra Attributes

Notes: spotLightShape

Select | Load Attributes | Copy Tab

340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 500

340.00

Bits and bobs

- Simple subsurface
- Socket display
- Spherical Harmonic lighting
- Median cut algorithm dome lighting
- Environment convolution
- ....

Crowd Animation & Rendering

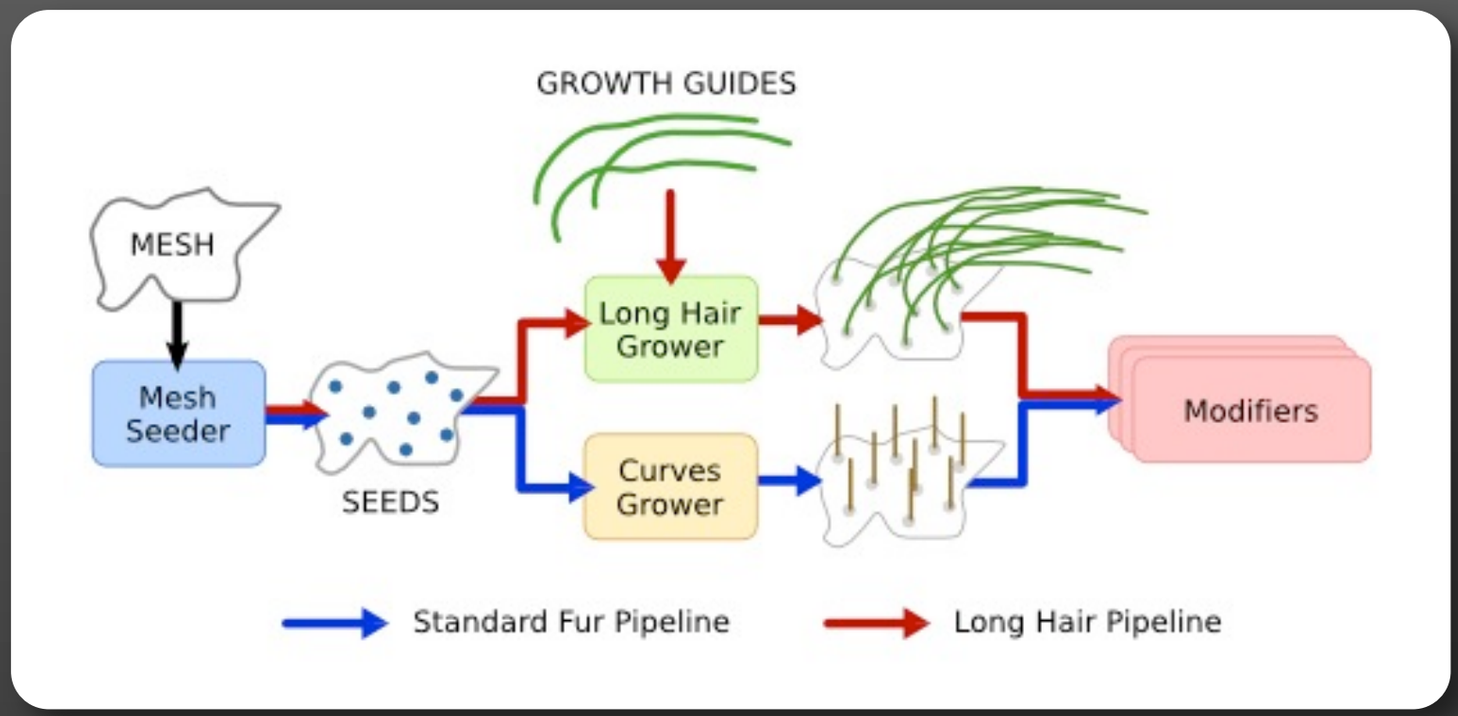
Carsten Kolve, Dr. D Studios

Fur at Image Engine

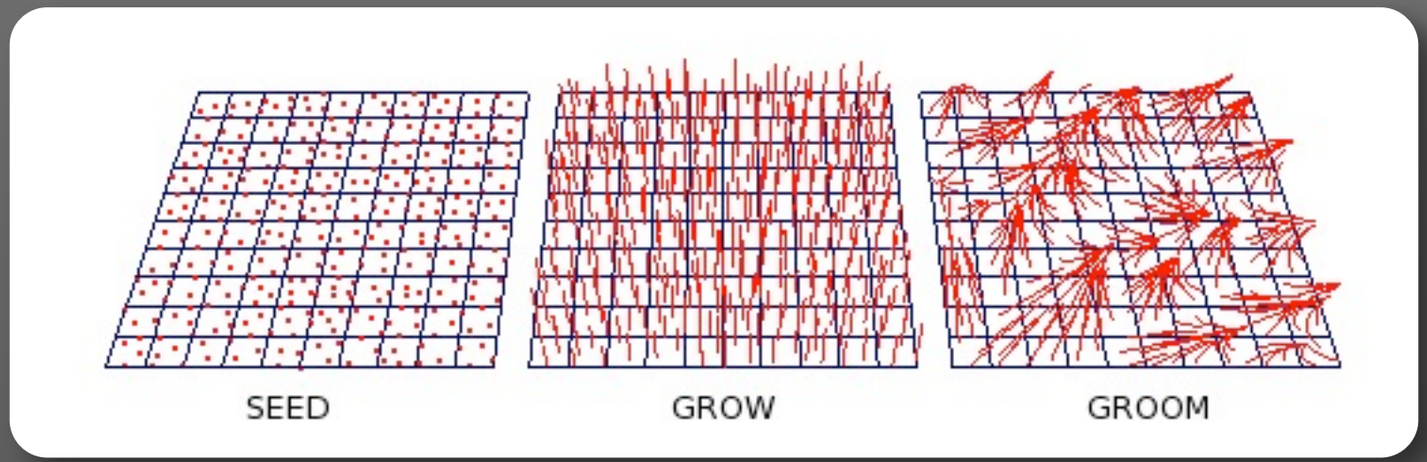
Andrew Kaufman, Image Engine

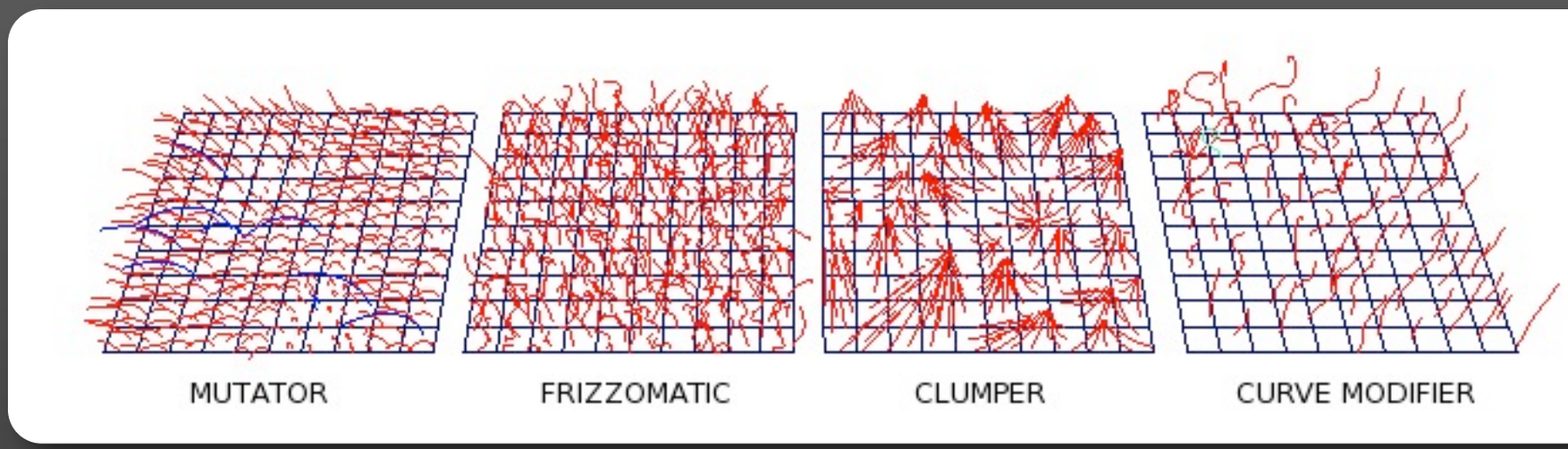
Fur System

- Component of standard procedurals
- Implemented in C++
- Multi-threaded
- Groom and light in Maya*



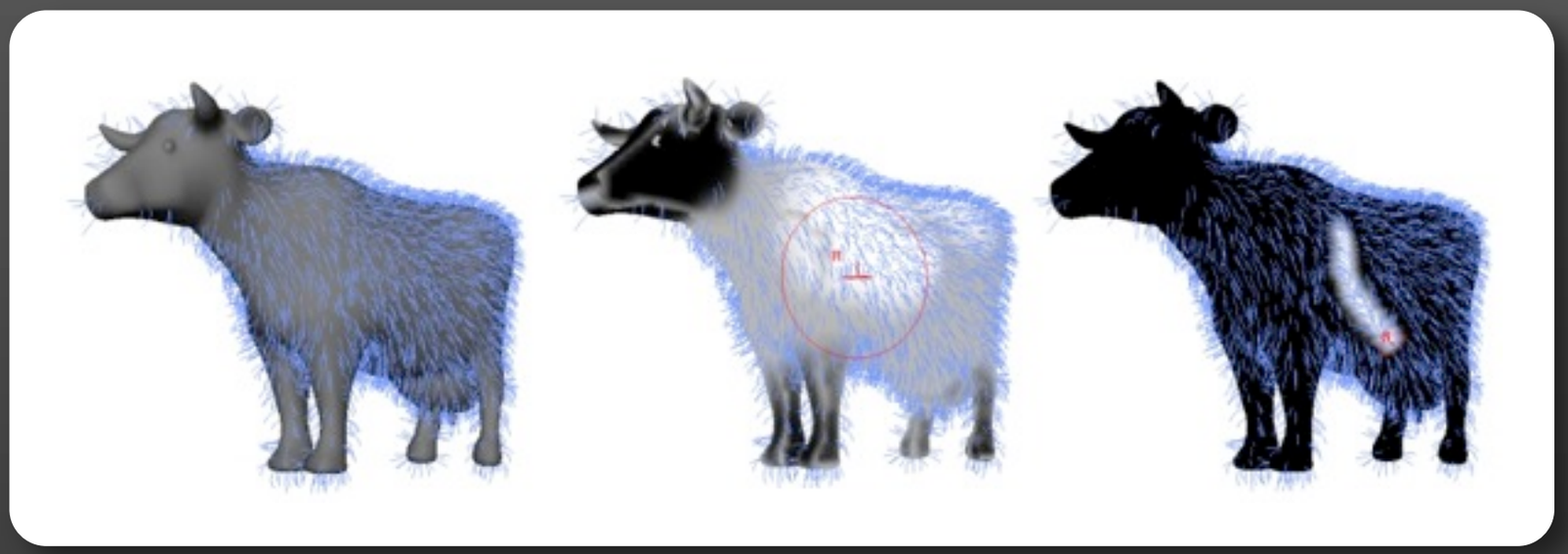
Main Components





Interactive Grooming

- Texture parameters give spatially varying control
- Vertex and image textures painted interactively
- RSL Shaders provide procedural texture inputs
- Evaluated using 3delight Sx API
- Reuse shaders from layered shading system

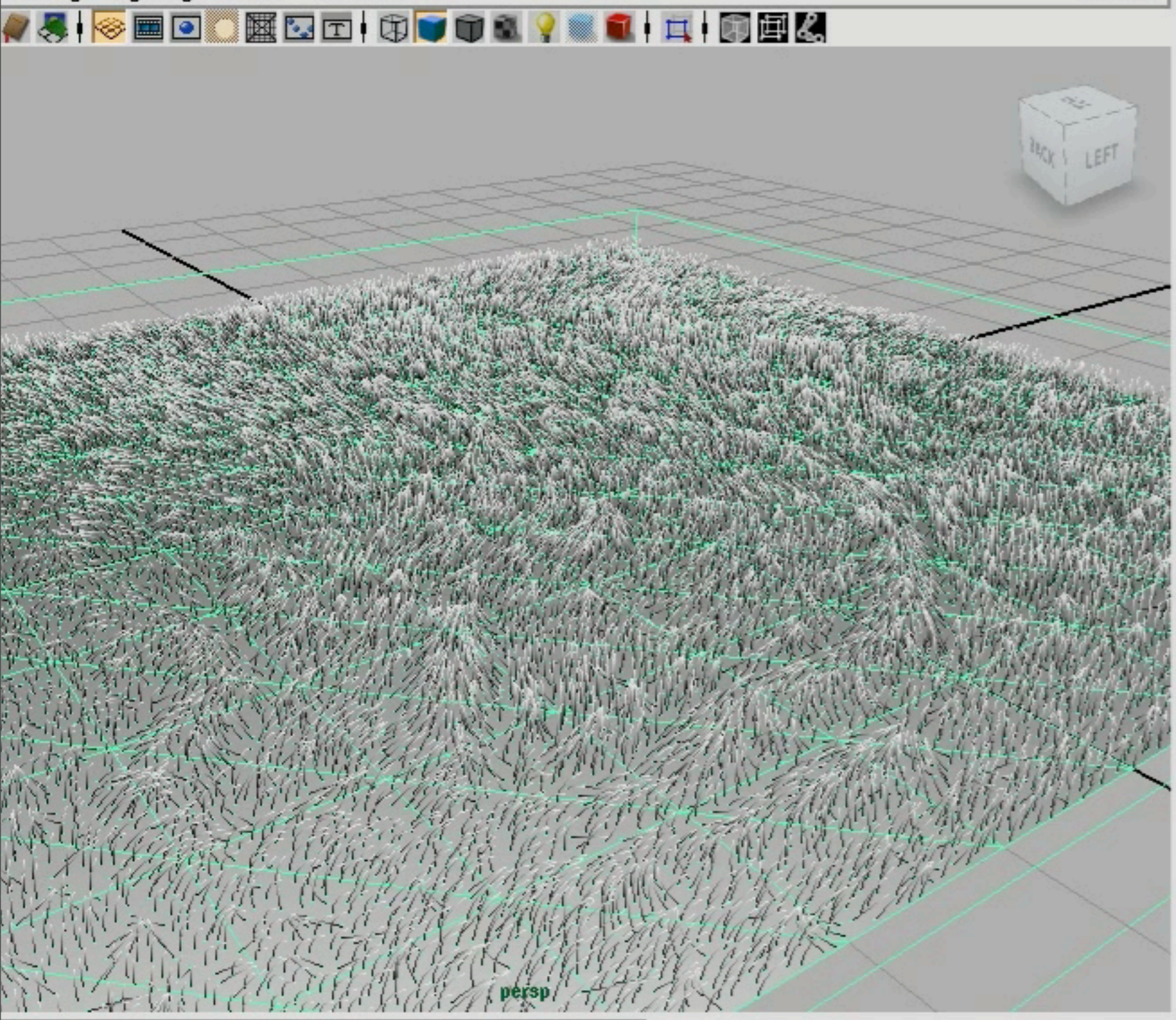


aya 2010 x64: /data/jobs/BD1/sequences/build/shots/esme/maya/scenes/rendering/furDemo.

odify Create Display Window Assets Select Mesh Edit Mesh Proxy Normals Color

dit UVs Cortex IE Help

Shading Lighting Show Renderer Panels



10 15 20

14

14.00

Attribute Editor: modularProceduralShape

List Selected Focus Attributes Show

modularProceduralShape time1 ieGeometryCombiner1

ceduralHc modularProceduralShape

Change Me

Groomer

Clumper

Ribbon Clumper

Clump Radius 0.3000

Percentage 83.3340

Attraction 0.6470

Attraction Variation 0.0000

Falloff 0.0000

Jitter 0.0000

Clump Volume 0.0000

Preserve Length

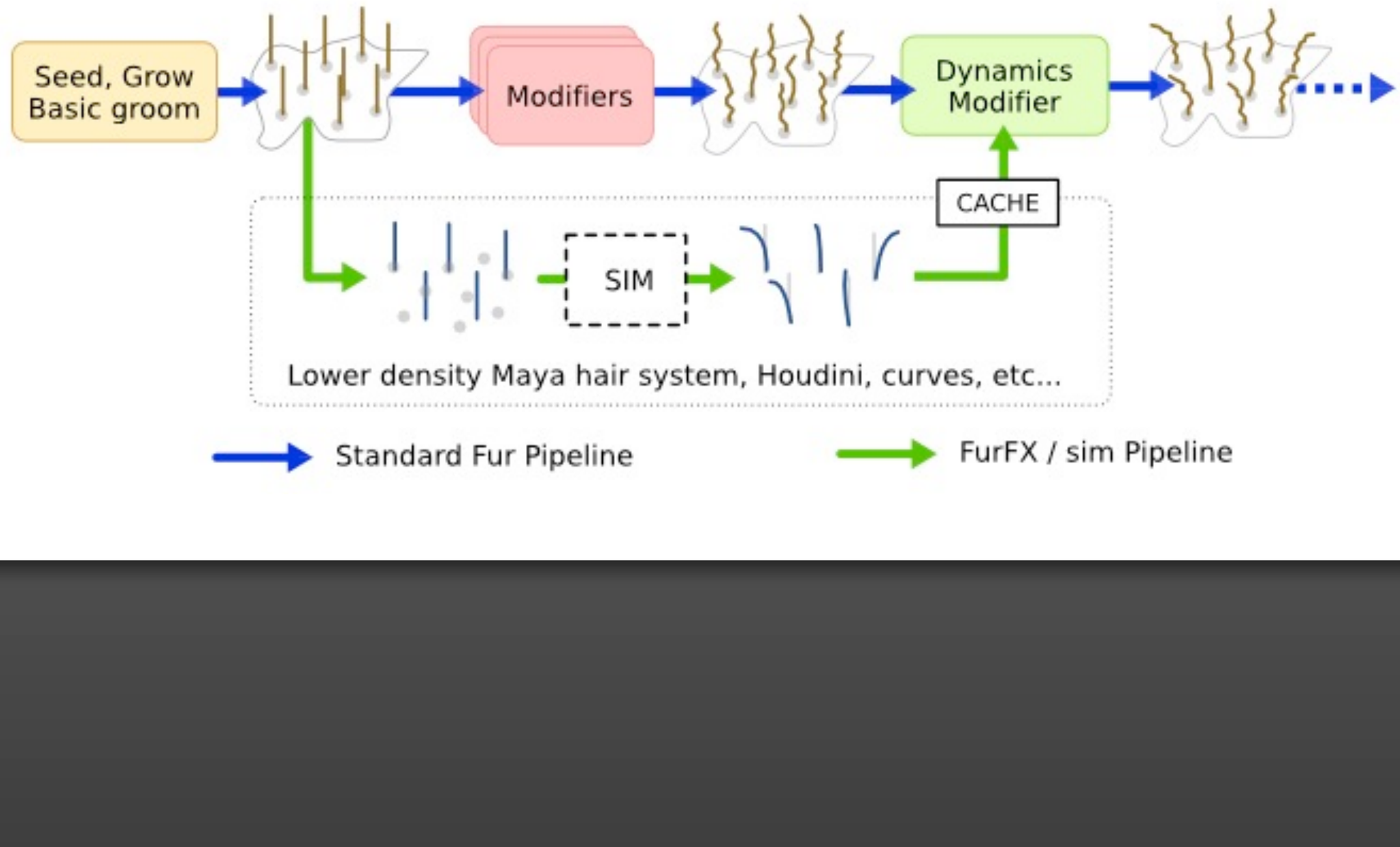
Affects Normals Modifier

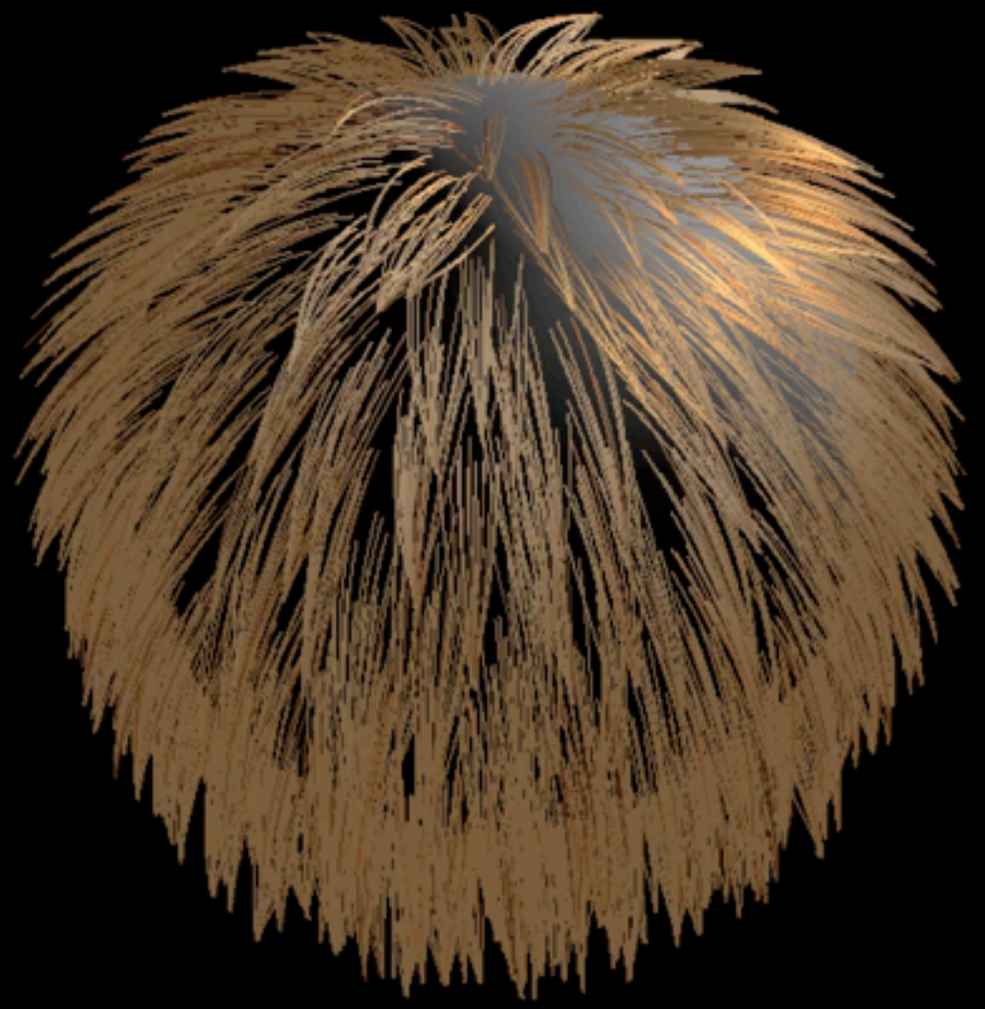
Profile

Notes: modularProceduralShape

Select Load Attributes Copy Tab Close

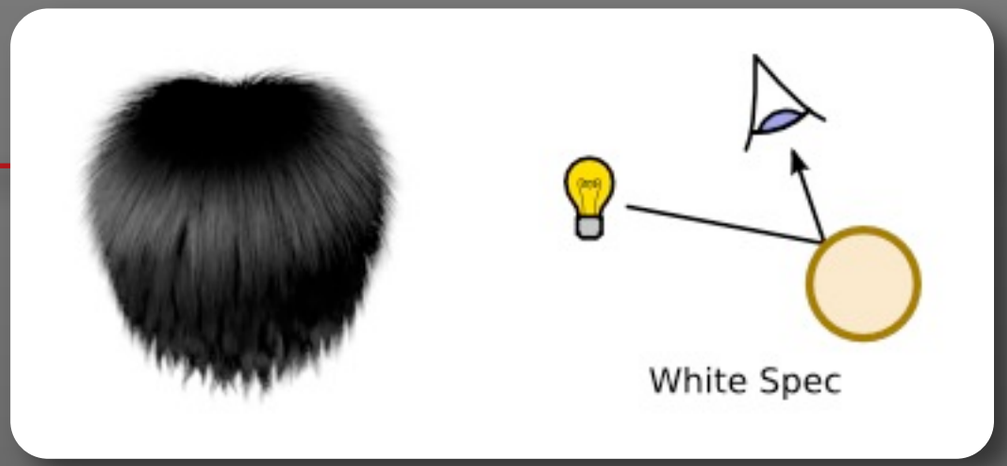
Dynamics

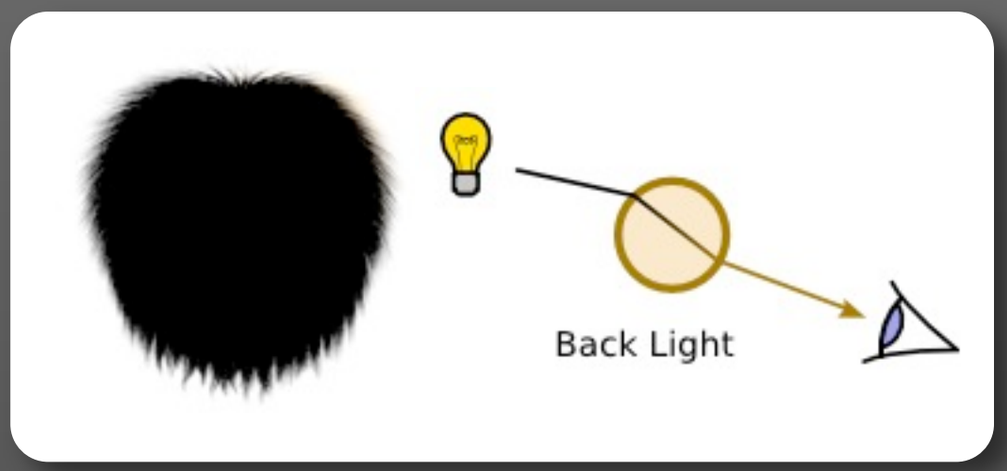

Seed, Grow
Basic groom
Modifiers
Dynamics
Modifier
CACHE
SIM
Lower density Maya hair system, Houdini, curves, etc...
Standard Fur Pipeline
FurFX / sim Pipeline

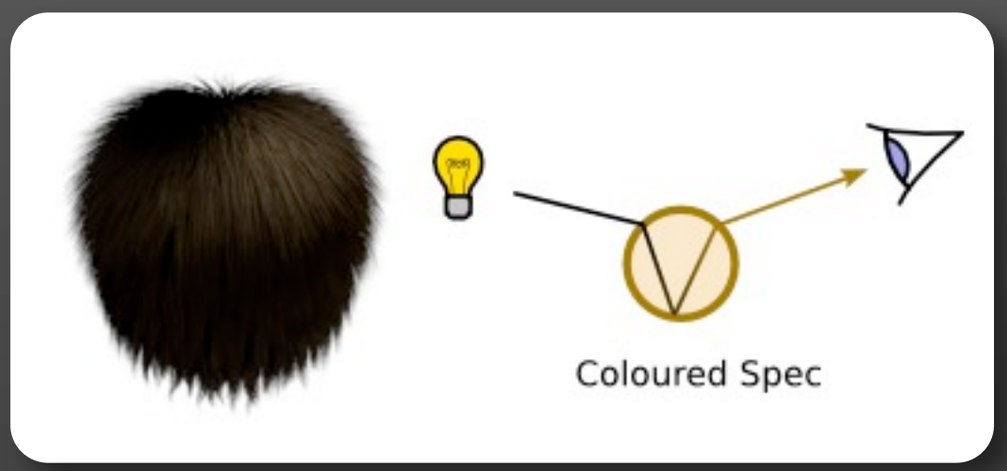



Shading

- Marschner Hair Model
- RSL for final frames
- OpenGL for preview
- Lots of deep shadows
- Experimented with SH



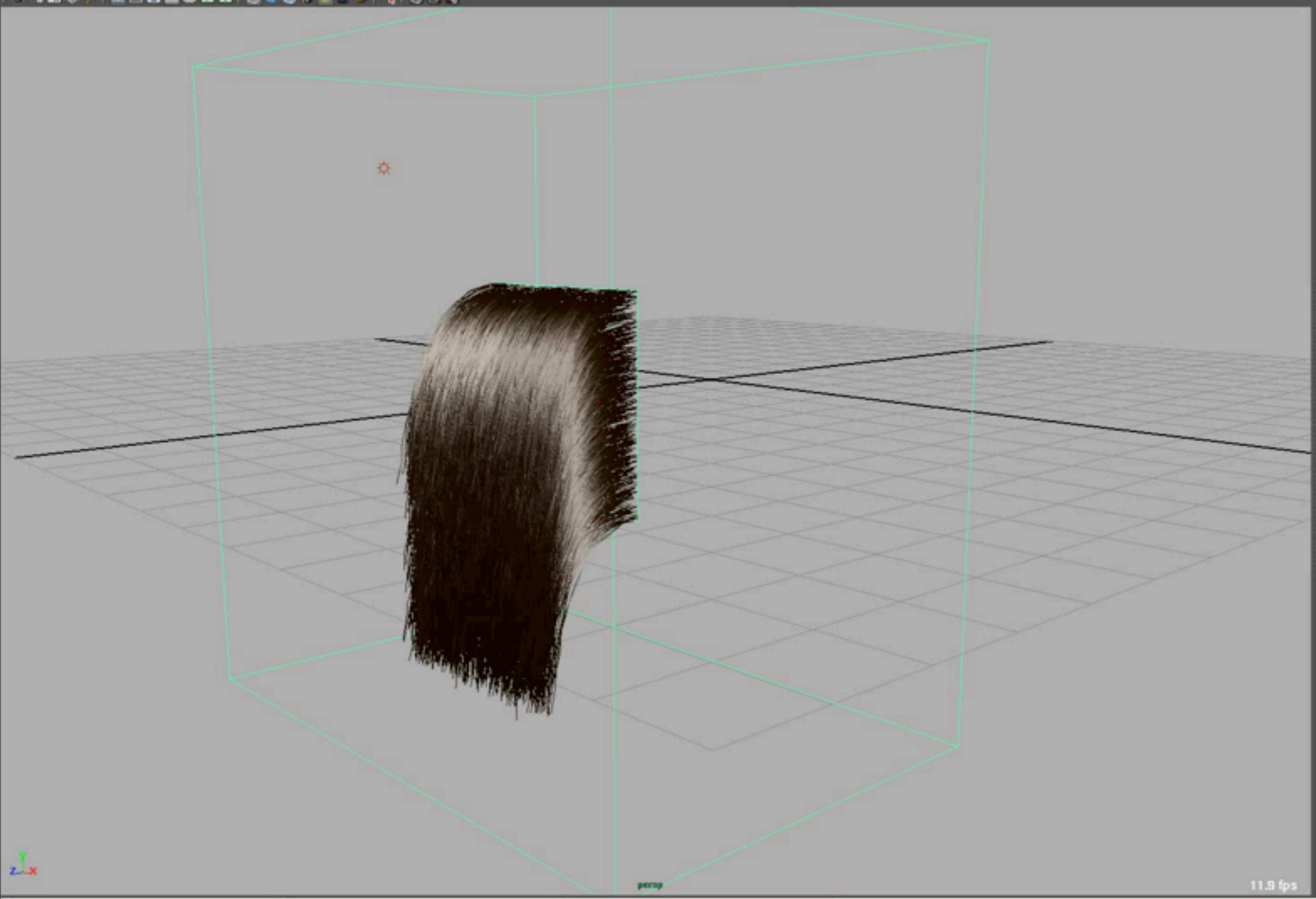




File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Cortex IE Help

Animation Curves Custom Deformation Dynamics General PaintEffects Polygons Rendering Subdivs Surfaces Toon IE_Lighting IE_Modelling

View Shading Lighting Show Renderer Panels



persp

11.9 fps

Attribute Editor

List Selected Focus Attributes Show Help

shadedFurExample | shadedFurExampleShape | initialShadingGroup

eProceduralHolder: shadedFurExampleShape

Focus Presets Show Hide

Class

Display

Parameters

Mesh Not connected

Diffuse 0.4000
White Spec 2.5000
Backlight 0.9000
Coloured Spec 1.0000

Marschner Parameters

Color
Refraction 2.1507
Eccentricity 1.0000
Shift R -8.0000
Width R 10.0000
Shift TT 4.0000
Width TT 5.0000
Shift TRT 12.0000
Width TRT 20.0000
Glint 1.0000
Caustic Width 20.0000
Caustic Fade 0.2000
Caustic Limit 0.5000

Hair

Length 4.0000
Count 5000
Frizz 0.1000
Gravity 1.0000

Render Stats

Extra Attributes

Notes: shadedFurExampleShape

Select Load Attributes Copy Tab

testShots_walkCycle
testShots_walkCycle comp v158

image engine
VISUAL EFFECTS FOR FILM

vera
2010-04-21

Wolf Test

testShots_walkCycle_comp_comp_native_v024.####.dpx

0125
0001-0240

image engine
VISUAL EFFECTS FOR FILM

"Twilight: Eclipse" Wolf Shots

Cool Things Dan Has Been Doing™

Carsten Kolve, Dr. D Studios

Making Tools with Cortex

(or How Cortex Makes It Easy To Do Cool Stuff)

danbethell@gmail.com

- Overview
  - Why use Cortex to develop Tools?
- Development
  - How do we write Tools with Cortex?
  - How do we build Tools against Cortex?
  - How do we deploy our Tools?
- Examples

Why use Cortex?

- Foundation of Core Libraries:
  - Geometry.
  - Rendering.
  - Algorithms.
  - I/O.
  - ...

- Multiple Application Support:
  - Houdini.
  - Maya.
  - Nuke.
  - Gaffer.
  - ...

- Unified plugin interface:
  - Parameters.
  - Data passing mechanisms.
  - Python Scriptability.
  - ...

Writing Plugins

- Include Cortex Headers
  - Just the ones you need!

```
#include <IECore/Op.h>
#include <IECore/CompoundParameter.h>
#include <IECore/NumericParameter.h>
#include <IECore/MeshPrimitive.h>
```

- Write Tools as Modular Components:
  - Inherit from IECore::Op

```
#include <IECore/Op.h>
#include <IECore/CompoundParameter.h>

class MyOp : public IECore::Op
{
    public:
       IE_CORE_DECLARERUNTIMETYPEDEXTENSION( MyOp, MYOP_ID, IECore::Op );
       MyOp();
       virtual ~MyOp();

    protected:
       virtual IECore::ObjectPtr doOperation(
          const IECore::CompoundObject *operands );
}
```

Building Plugins

- Build against your Cortex:
  - Same compiler!
  - Same libraries!
  - Same applications!

- Made easy by PkgConfig:
  - *$INSTALL_PREFIX/lib/pkgConfig/cortex.pc*
  - Set the *PKG_CONFIG_PATH* environment variable.

- Example using CMake:

```
pkg_check_modules( Cortex REQUIRED cortex )

include_directories( ${Cortex_INCLUDE_DIRS} )
link_directories( ${Cortex_LIBRARY_DIRS} )
target_link_libraries( MyPlugin ${Cortex_LIBRARIES} )
```

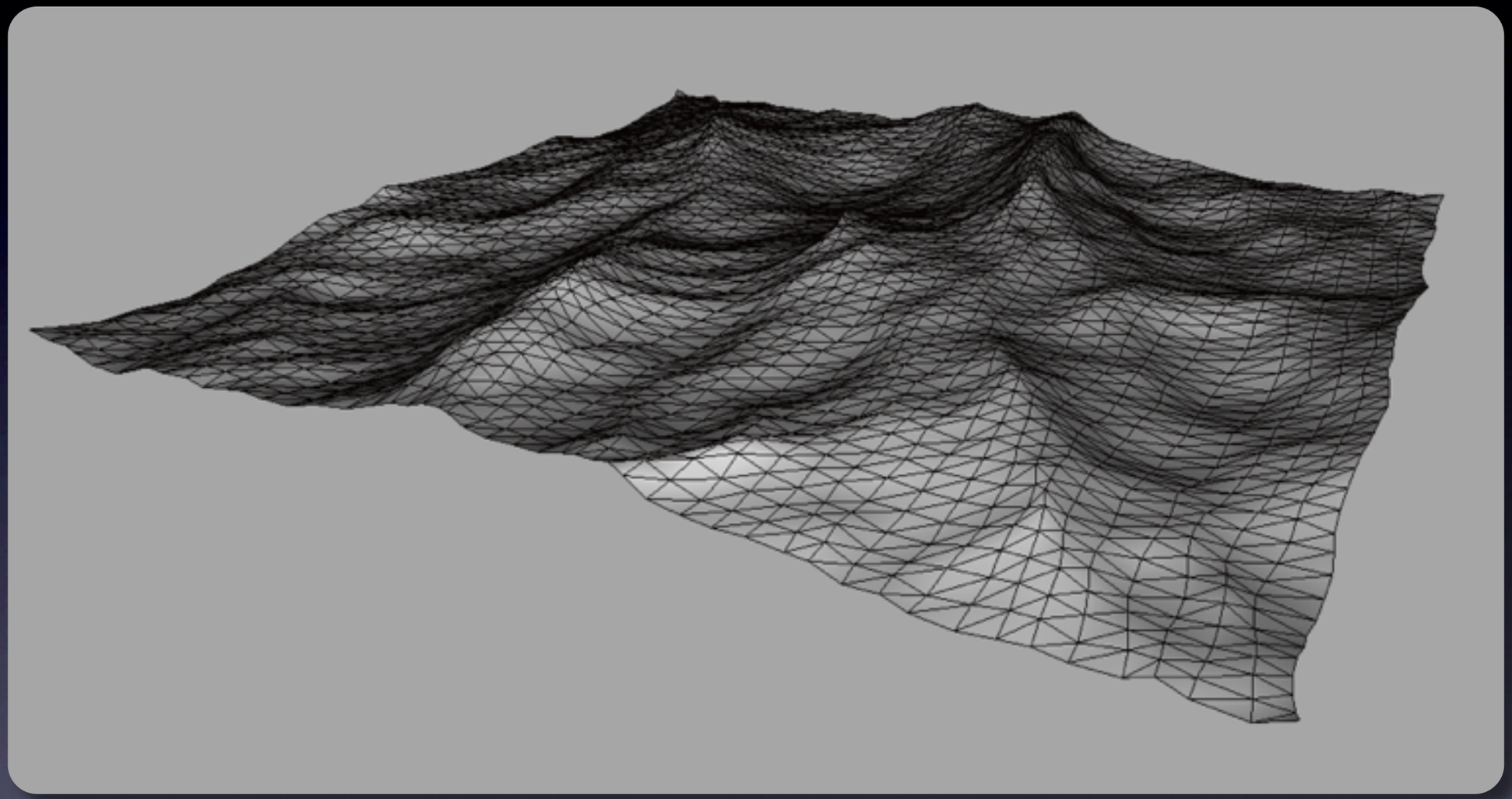
Deploying Plugins

- Cortex Environment Variables:
  - *$IECORE_OP_PATHS*
  - *$IECORE_PROCEDURAL_PATHS*
- Install an Op stub for C++ Plugins:
  - *$OP_PATH/MyPlugin/MyOp/MyOp-1.py*

```
import os, sys
sys.path.append( os.path.dirname( __file__ ) )

from MyPlugin import MyOp
```

Examples

Cortex Ocean



Port of Houdini Ocean Toolkit to Cortex

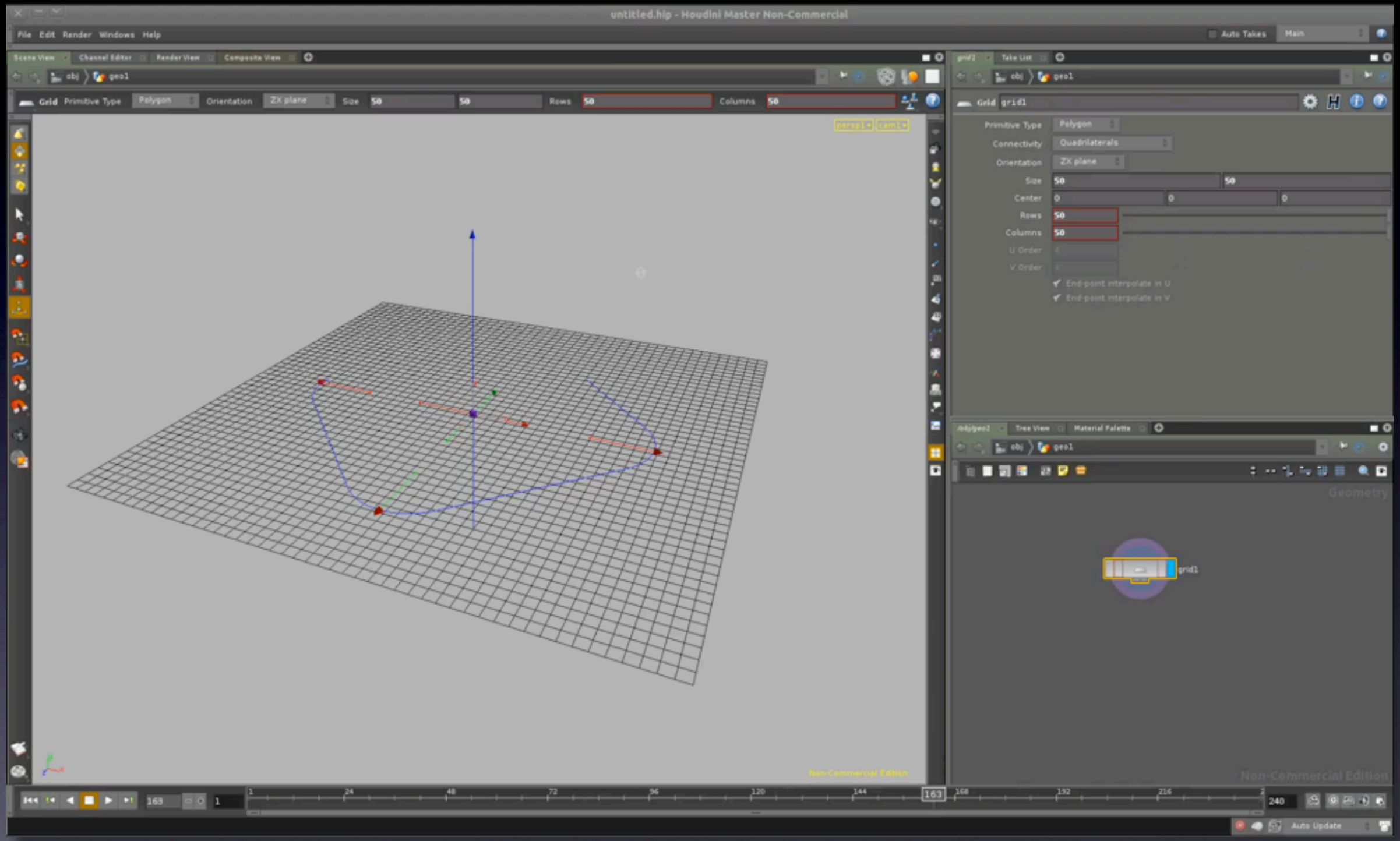
- Port of existing code to Cortex.
  - Houdini Ocean Toolkit by Drew Whitehouse
  - http://code.google.com/p/houdini-ocean-toolkit/

- Ported as a Cortex ModifyOp:
  - Just implement ::modify()

```
class OceanOp : public ModifyOp
{
    public:
        IE_CORE_DECLARERUNTIMETYPEDEXTENSION( OceanOp, 666001, ModifyOp );
        OceanOp();
        virtual ~OceanOp();

    protected:
        virtual void modify( Object *input, const CompoundObject *operands );
}
```

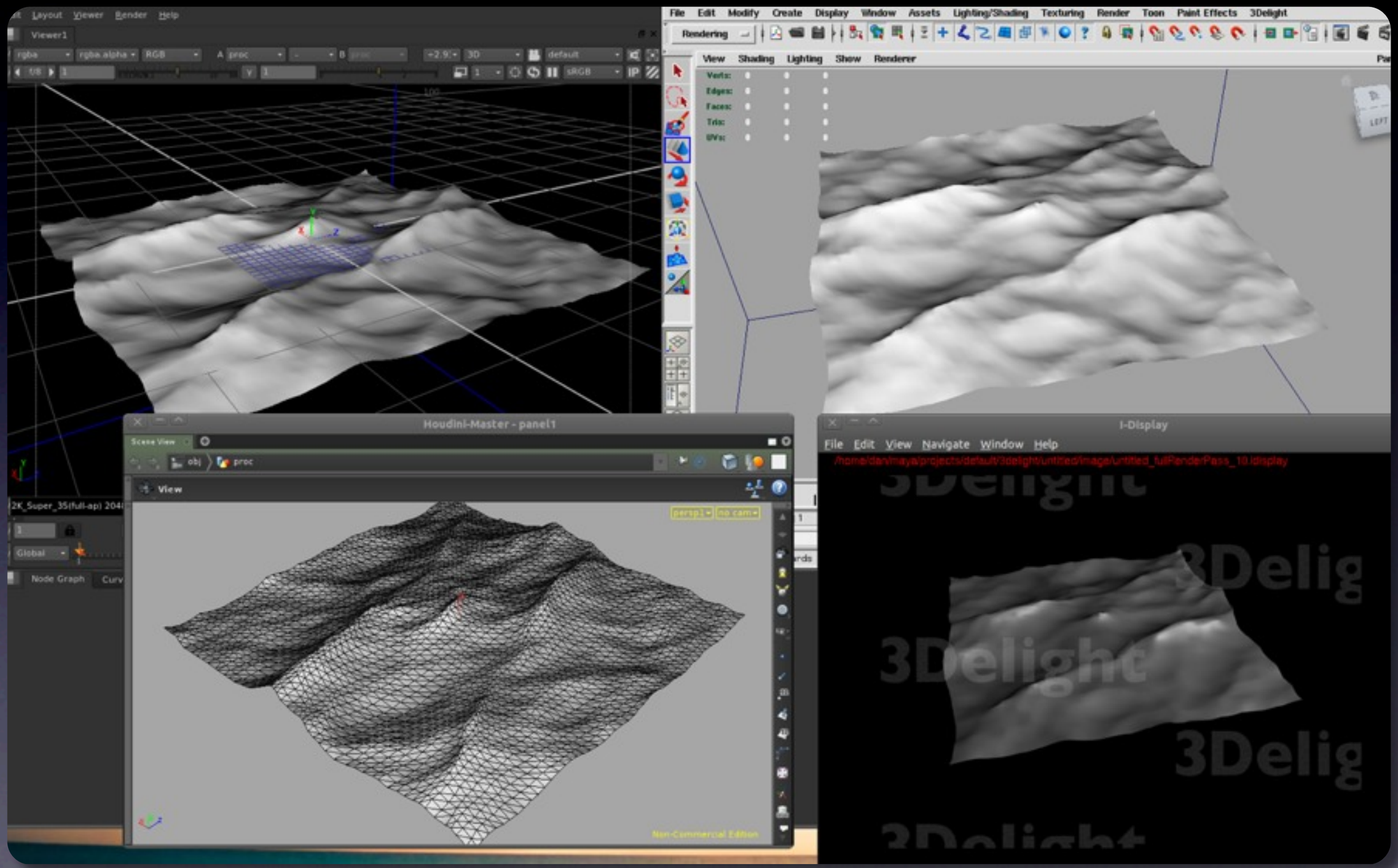
- Built & deployed with CMake:

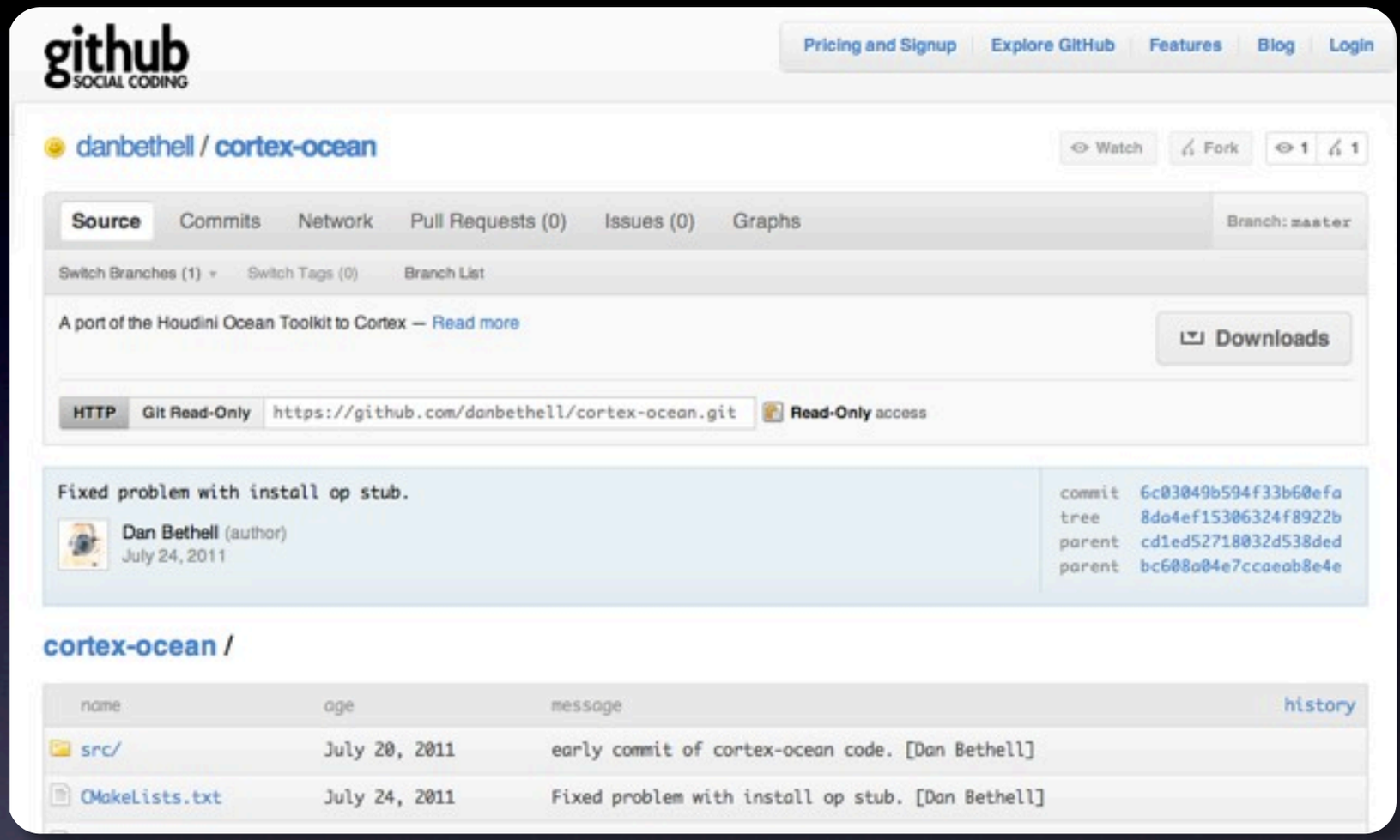
```
find_package( PkgConfig REQUIRED )
pkg_check_modules( Cortex REQUIRED cortex )

include_directories( ${Cortex_INCLUDE_DIRS} )
link_directories( ${Cortex_LIBRARY_DIRS} )

add_library( CortexOcean
  SHARED
  ${CMAKE_SOURCE_DIR}/src/OceanOp.cpp
  )

target_link_libraries( CortexOcean
  ${Cortex_LIBRARIES}
  fftw3f
  )
```



Duration: ~2 mins

- Runs unmodified in any Cortex host:





http://github.com/danbethell/cortex-ocean

Crumbs

Particle system implemented in C++/Cuda

- Passing data using Cortex:
  - Wrap custom type with IECore::TypedData<>.

```
// our custom type
namespace crumbs
{
   class Crumb
   {
      ...
   }
};
```

```
// wrapping for use with Cortex
namespace IECore
{
   typedef IECore::TypedData<crumbs::Crumb> CrumbsData;
   IE_CORE_DECLARE_PTR( CrumbsData );
}
```

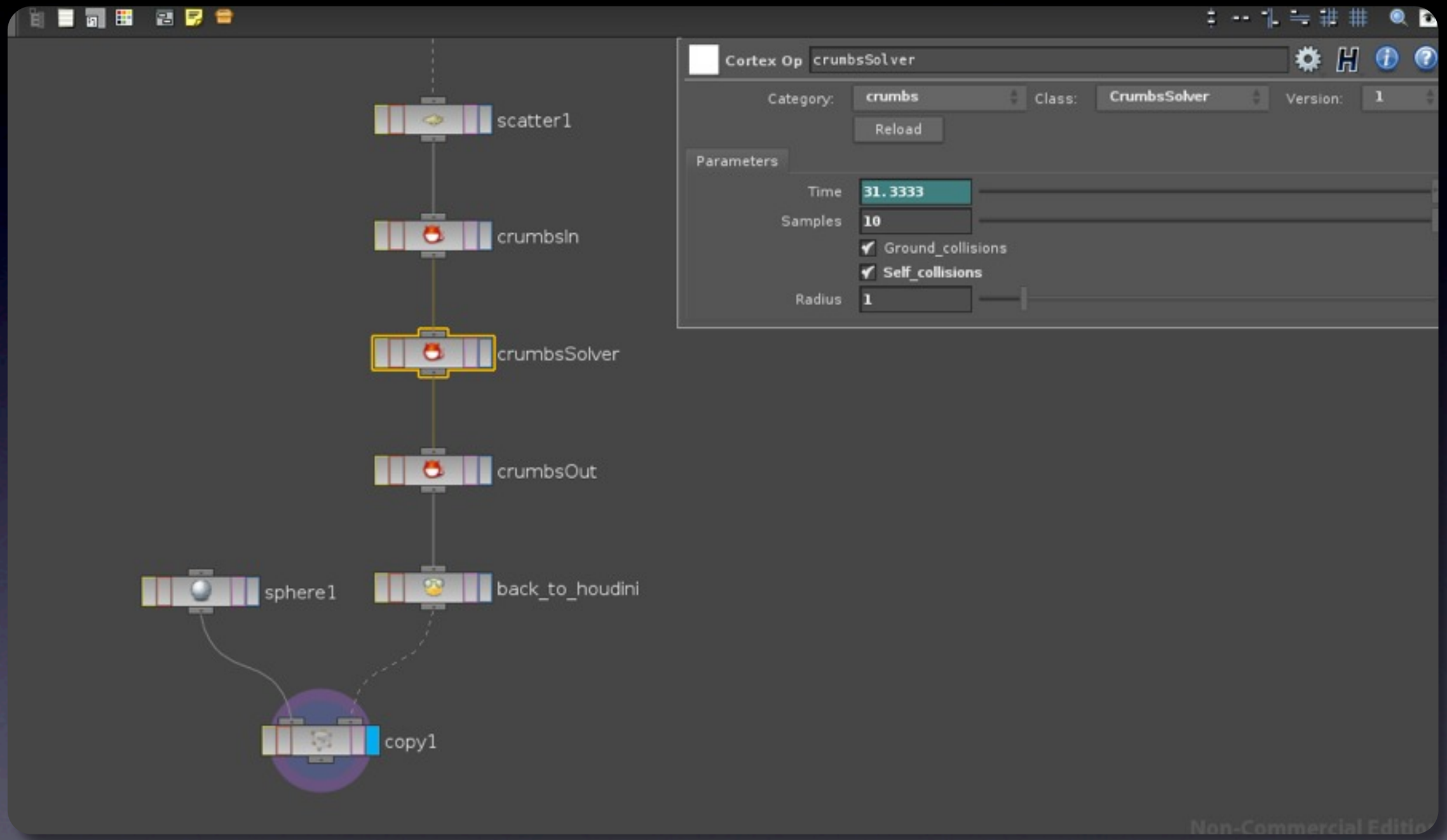
- Passing our custom data between Ops:
  - Use the IECore::ObjectParameter type.

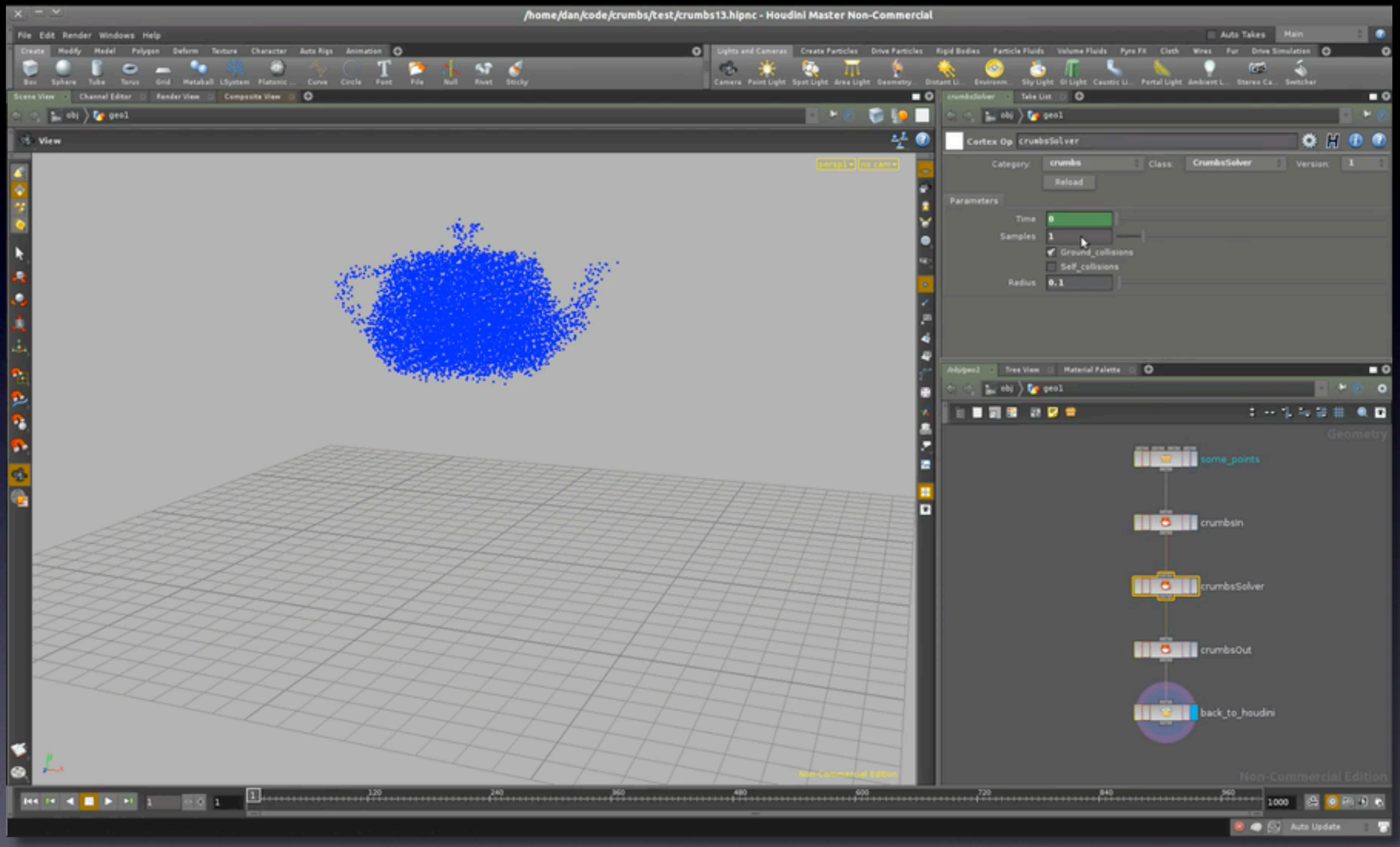
```
// define our Cortex parameter in our Op constructor
m_crumbsInput = new IECore::ObjectParameter( “input”, “Crumbs input.”,
     new crumbs::CrumbsData(), IECore::DataTypeId );

parameters()->addParameter( m_crumbsInput );
```

```
// in Op::doOperation extract the custom data
const CrumbsData *crumbs_data = operands->member<CrumbsData>( “input” );
const Crumb &crumb = crumbs_data->readable();
```

- Crumbs Types:
  - Data
    - crumbs::CrumbsData (Pass GPU info between Ops)
  - Ops
    - crumbs::CrumbsIn (Geometry Cortex->GPU)
    - crumsb::CrumbsOut (Geometry GPU->Cortex)
    - crumbs::CrumbsSolver (Simulation)

- Runs unmodified in any Cortex host:





Duration: ~1:30 mins

- Binding to Python is really quick:
  - IECorePython together with boost::python.

```
#include <boost/python.hpp>
#include <IECorePython/RunTimeTypedBinding.h>
#include “CrumbsIn.h”
#include “CrumbsSolver.h”
#include “CrumbsOut.h”

using namespace boost::python;

BOOST_PYTHON_MODULE( Crumbs )
{
   IECorePython::RunTimeTypedClass<crumbs::CrumbsIn>().def( init<>() );
   IECorePython::RunTimeTypedClass<crumbs::CrumbsSolver>().def( init<>() );
   IECorePython::RunTimeTypedClass<crumbs::CrumbsOut>().def( init<>() );
}
```

- Then... we can call our Ops from Python:

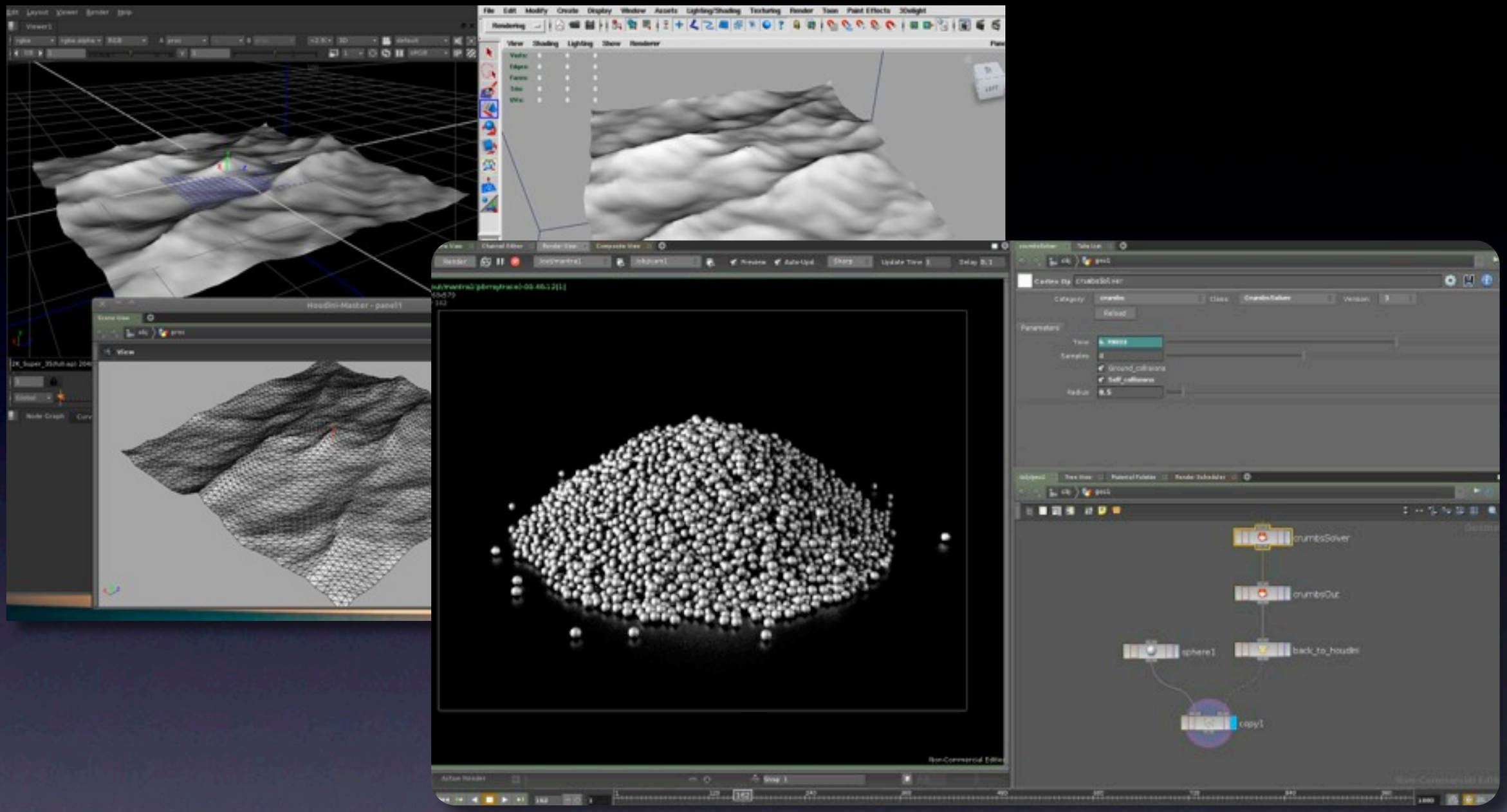
```
from IECore import *
from Crumbs import CrumbsIn, CrumbsOut, CrumbsSolver

# load points
points = Reader.create(“myPoints.bgeo”).read()

# upload, solve and download from gpu
gpu_in = CrumbsIn()( input=points )
gpu_solved = CrumbsSolver()( input=gpu_in, time=0.041 )
gpu_out = CrumbsOut()( input=gpu_solved )

# save points
Writer.create( gpu_out, “particles.0001.bgeo” ).write()
```

Released as open-source soon...

http://github.com/danbethell

So, why use Cortex?

- Develop plugins outside application constraints.
- Straightforward to build & deploy against Cortex.
- Cortex API stable across major version.



danbethell@gmail.com

Nuke Integration

Paolo Berto, Jupiter Jazz

Nuke Integration

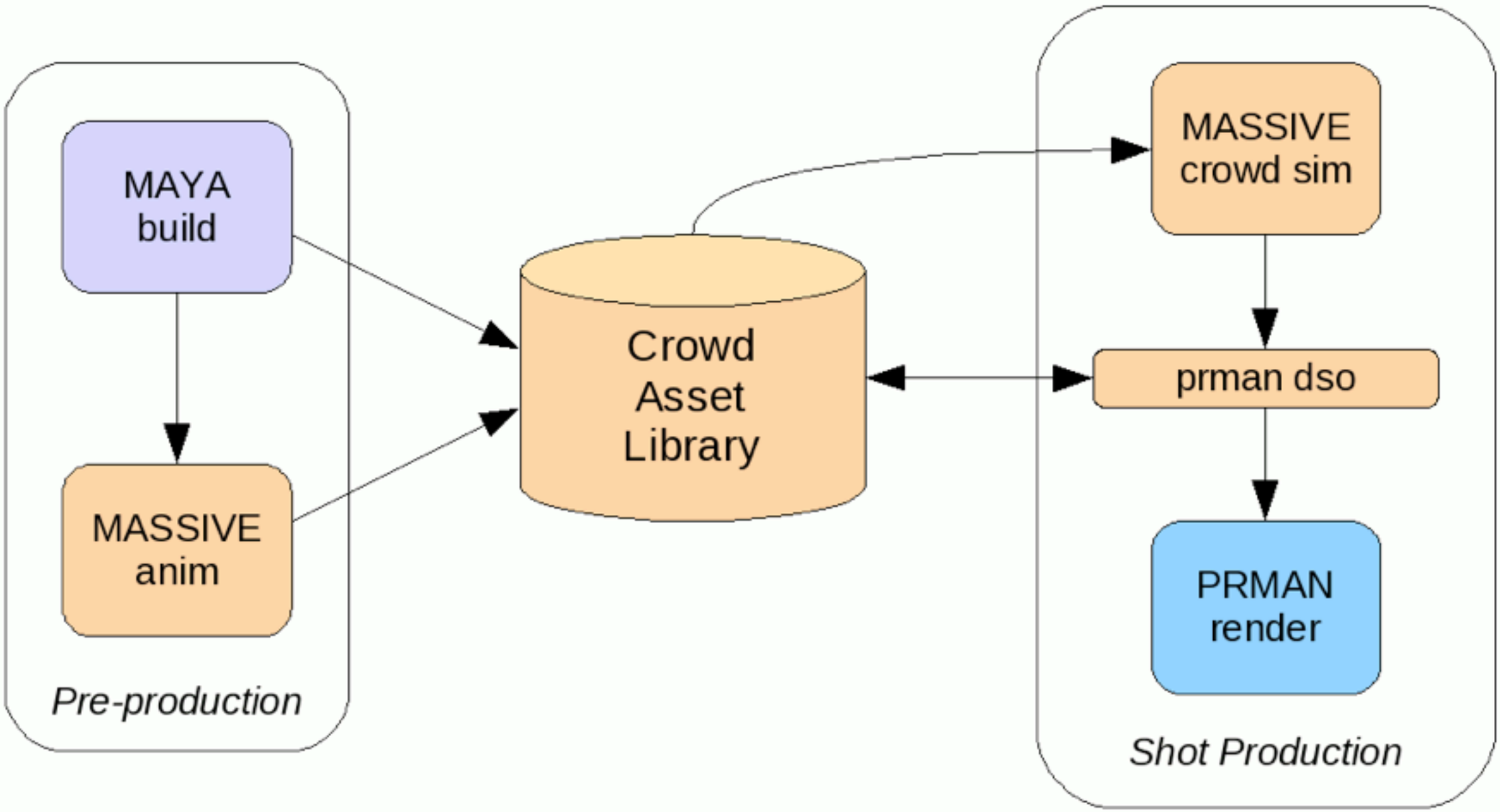
- Introduced in Cortex 6
- Hosting of procedurals and ops
- Lacking integration with built in renderer
- Wouldn’t it be nice...

Oh No, Not Crowds Again

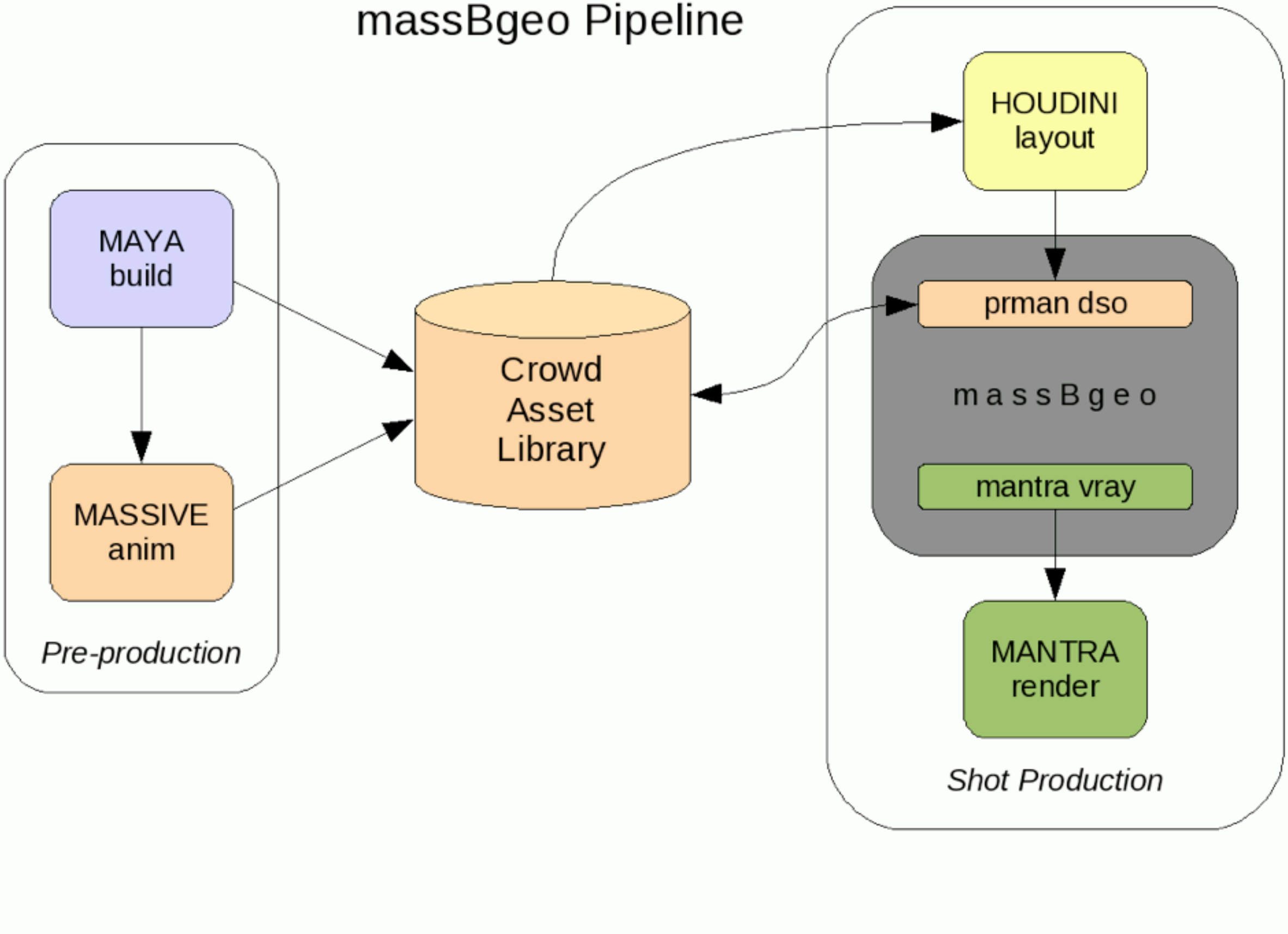
Using Cortex to Undo the Biggest Hack of All Time

Ollie Rankin, Method

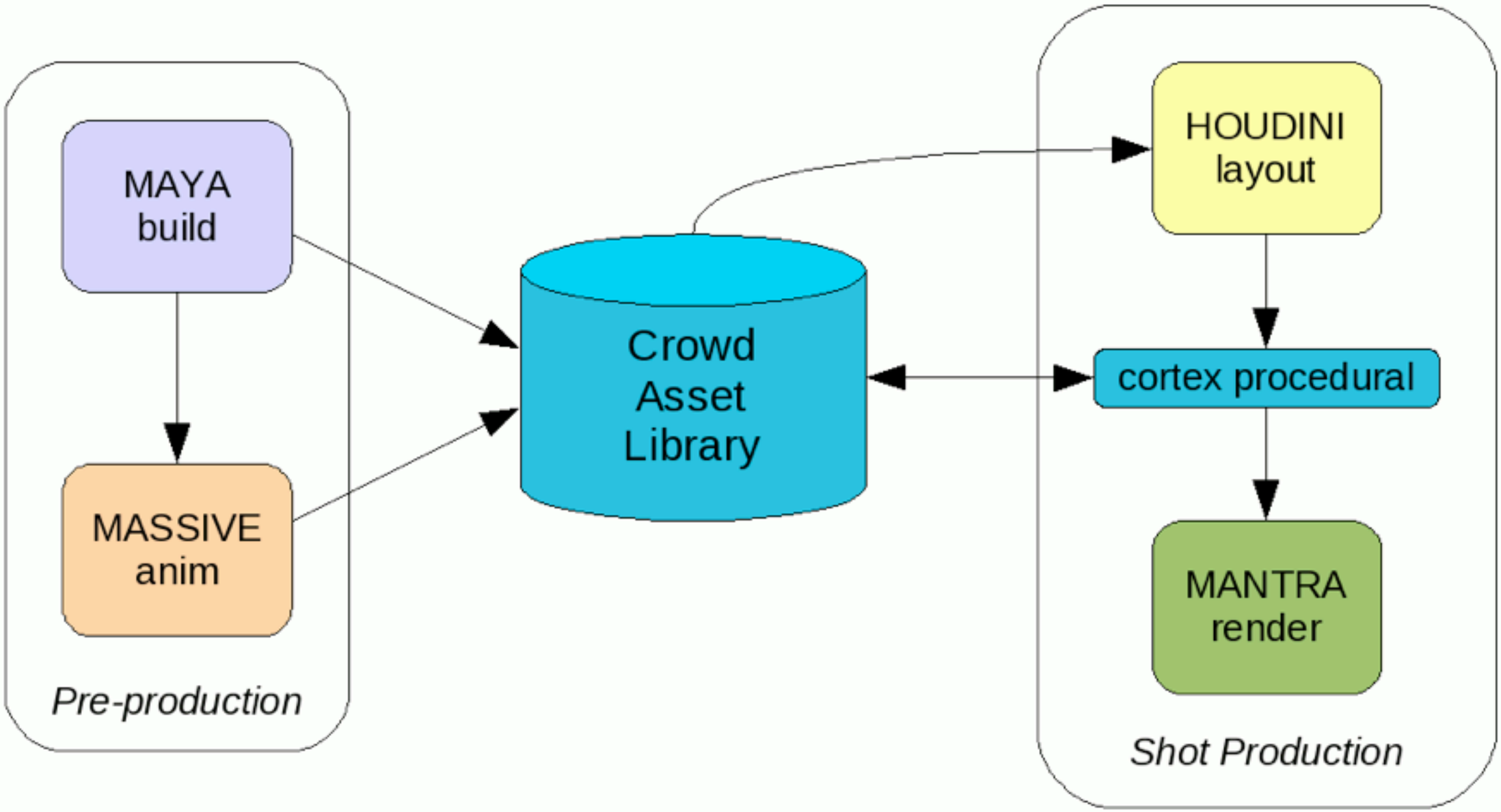
A Typical Massive Pipeline



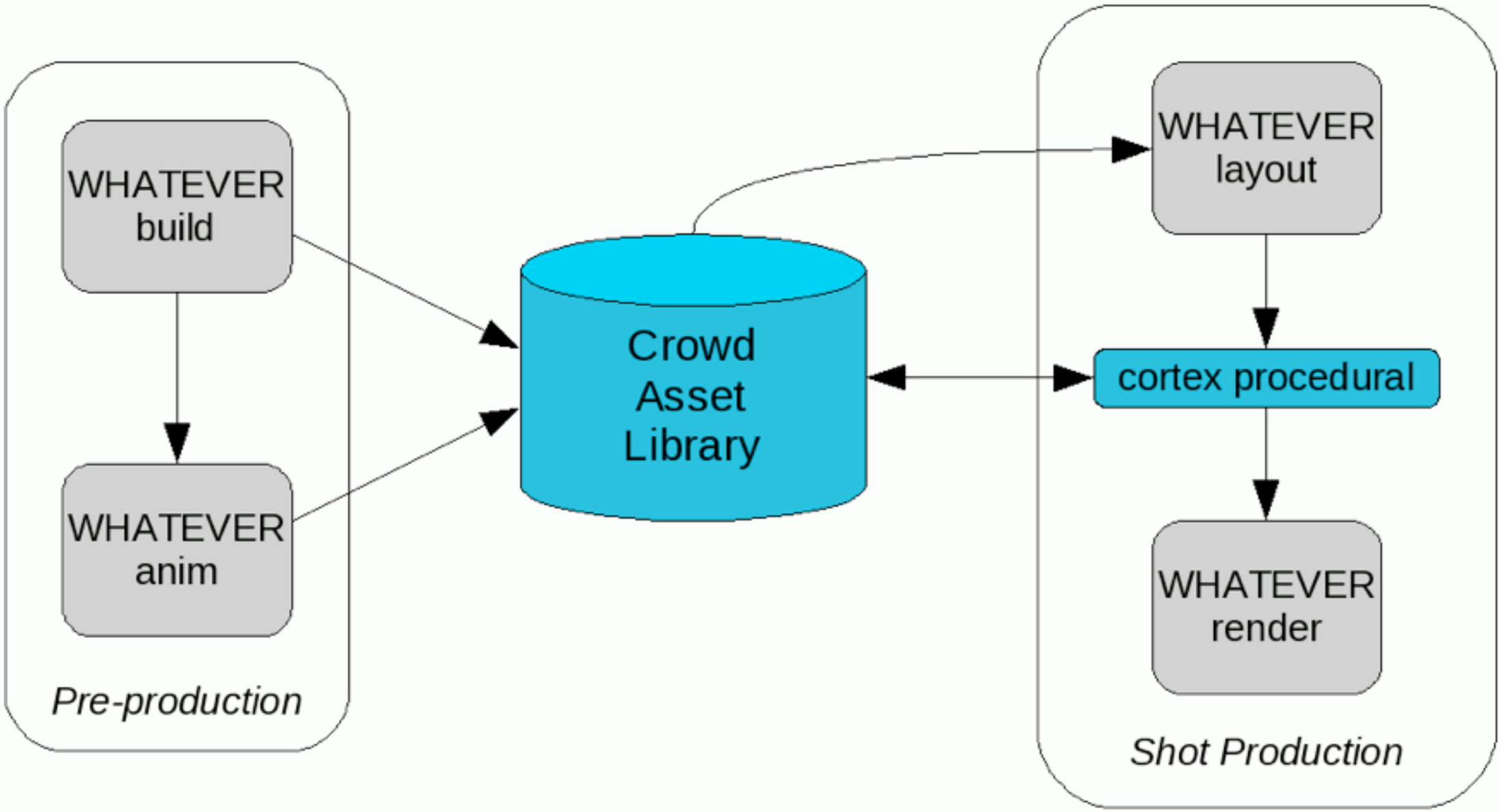
massBgeo Pipeline



First Gen Cortex Pipeline



Future Cortex Pipeline



Starting up with Cortex

Johannes Saam

fuelvfx

Situation at FUEL

- Growing from small to medium size
- Changing to a new Linux based pipeline
- Supporting as many applications as possible

Why Cortex

- Need all applications to be able to ‘talk’ to each other
- Platform for quick and flexible solutions
- Short development times and limited resources

Implications

- Compiling...
- Environment creation more complex
- Education required

Gains

- Solid base for your pipeline (find out how long it already exists)
- Support for free!
- Development for free!

First tests

- Tested mesh caching
- Cortex up to 6x faster then standard Maya caching
- Large meshes are handled really well
- File sizes on disk are a fracture
- First problem solved in only one hour of dev time

Plans

- Interchange Meshes / Particles
- Interchange Cameras
- Educate TD`s
- Contribute to the Nuke plugins
- Start Deep image support

Thanks

FUEL VFX

Pawel Olas
Max Stummer
Roy Mahli

Future Directions

John Haddon

When we have time...

- Better PRMan support
- More renderer backends
- Better documentation
- Binary downloads
- Windows builds
- Deep image support
- ....


G

Gaffer

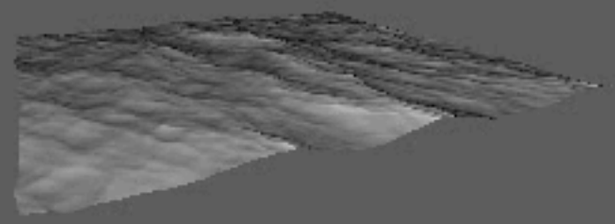
- Application framework
- Native Cortex host
- Work in progress

Gaffer continued...

- Dependency graph
- Undo/redo/cut/paste/load/save
- Files are Python scripts
- Python scripting is direct binding of C++ API
- UI toolkit wrapping Qt and OpenGL

Gaffer File Edit Layout

Viewer

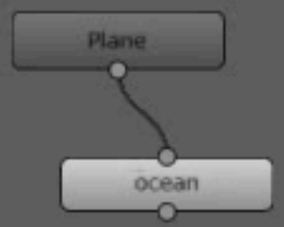


Node Editor

ocean

Waveheight 5.0
Seed 0
Chop ✓
Choppiness 2.5
Damp 1.0
Windalign 2.0
Depth 200.0
Time 0.0

Graph Editor



Script Editor

```
timePlug = getChild( "ocean" )["parameters"]["time"]
```

code, examples, documentation

code.google.com/p/cortex-vfx

discussion

cortexdev@googlegroups.com