Houdini FX

POPs

Random Spin:

```
// The following makes it random:
axis = rand(@id) - set(0.5, 0.5, 0.5);
spinspeed *= rand(@id+0.1);
```

Python

One off shelf scripts

Version up sops that have a "version" parm - I use this to version up a lot of exports in tops or rops:

```
import hou
nodes = hou.selectedNodes()
for x in nodes:
    ver = x.parm("version").eval() + 1
    x.parm("version").set(ver)
```

Send a ROP fetch to TOPs name "topnet1" for convenient cooking.:

```
import hou
nodes = hou.selectedNodes()
topnet = hou.node("/obj/topnet1")
for x in nodes:
    name = x.name()
    topcache = topnet.createNode("ropfetch")
    p = hou.IntParmTemplate("version", "Version", 1)
    g = topcache.parmTemplateGroup()
    g.append(p)
    topcache.setParmTemplateGroup(g)
    topcache.parm("version").set(ver)
    topcache.parm("roppath").set(path)
    topcache.setName(name)
```

Send SOP to a ROP net named "ropnet1", adds version parm to update the original SOP version cache:

```python
import hou
nodes = hou.selectedNodes()
ropnet = hou.node("/obj/ropnet1")
for x in nodes:
    name = x.name()+"_"+str(x.parent())
    path= x.path()
    if x.type().name() == "rop_geometry":
        ver = x.parm("vp_version")
    elif x.type().name() == "rop_alembic":
        ver = x.parm("vp_version")
    else:
        ver = x.parm("version")
    topcache = ropnet.createNode("fetch")
    topcache.parm("source").set(path)
    p = hou.IntParmTemplate("version", "Version", 1)
    g = topcache.parmTemplateGroup()
    g.append(p)
    topcache.setParmTemplateGroup(g)
    topcache.parm("version").set(ver)
    topcache.setName(name)
```

Make a file node from a rop export node:

```python
import hou, re
nodes = hou.selectedNodes()
parent = nodes[0].parent()
nodeList = []
for x in nodes:
    name = x.name()+"_load"

    version = x.parm("vp_version").eval()
    path= x.parm("sopoutput").eval()
    path = str(path)
    newpath1 = re.sub("_v\d+", "_v{0}", path).format("`padzero(3,ch('version'))`")
    newpath1 = re.sub("\.\d+", ".$F4", newpath1)
```

```python
        file = parent.createNode("file")
        try:
            file.setName(name)
        except:
            continue
        file.setColor(hou.Color(0,0,1))
        file.parm("file").set(newpath1)
        p = hou.IntParmTemplate("version", "Version", 1)
        g = file.parmTemplateGroup()
        g.append(p)
        file.setParmTemplateGroup(g)
        file.parm("version").set(version)
        nodeList.append(file)

parent.layoutChildren(items=nodeList)
```

PythonModule Scripts

Get houdini version:

```
hver = hou.applicationVersionString().rpartition('.')[0]
```

Move Files

```python
import os,hou, shutil
selected = hou.selectedNodes()
for x in selected:
    if x.parm("env_map"):
        file = x.parm("env_map").eval()
        parm1 = x.parm("env_map")
    elif x.parm("fileName"):
        file = x.parm("fileName").eval()
        parm1 = x.parm("fileName")
    else:
        pass
    filename = os.path.basename(file)
    destdir = os.path.join(hou.expandString("$HIP"), "assets")
    destpath = os.path.join(destdir, filename)
    if os.path.exists(file):
        if os.path.exists(destdir):
            pass
        else:
            os.makedirs(destdir)

        if os.path.exists(destpath):
            dpath = os.path.join(os.path.join("$HIP", "assets"), filename)
            parm1.set(dpath)
        else:
            shutil.copy(file, destdir)
            dpath = os.path.join(os.path.join("$HIP", "assets"), filename)
            parm1.set(dpath)

    else:import os,hou, shutil
selected = hou.selectedNodes()
for x in selected:
    if x.parm("env_map"):
        file = x.parm("env_map").eval()
```

```
        parm1 = x.parm("env_map")
elif x.parm("fileName"):
        file = x.parm("fileName").eval()
        parm1 = x.parm("fileName")
else:
        pass
filename = os.path.basename(file)
destdir = os.path.join(hou.expandString("$HIP"), "assets")
destpath = os.path.join(destdir, filename)
if os.path.exists(file):
        if os.path.exists(destdir):
                pass
        else:
                os.makedirs(destdir)


        if os.path.exists(destpath):
                dpath = os.path.join(os.path.join("$HIP", "assets"), filename)
                parm1.set(dpath)
        else:
                shutil.copy(file, destdir)
                dpath = os.path.join(os.path.join("$HIP", "assets"), filename)
                parm1.set(dpath)


else:
        pass
        pass
```

split by group

```python
import hou
selected = hou.selectedNodes()[0]
groups = [g.name() for g in selected.geometry().primGroups()]
for i, name in enumerate(groups, 1):
    #make a split node
    split = selected.createOutputNode("blast")
    split.setName(name)
    split.parm("group").set(name)
    split.parm("negate").set(1)
    split.moveToGoodPosition()



for node in selected:
    out = node.createOutputNode("null")
    out.setName("OUT_"+node)
    out.moveToGoodPosition()
```

Python

Update frame range from file

```python
def grabFrames(self):
    import os
    plate = self.parm("backplate").eval()
    if not plate:
        framex = self.parm("frange1").eval()
        framey = self.parm("frange2").eval()
    else:
        flist = [x.split(".")[-2] for x in os.listdir(os.path.dirname(plate))]
        fmin = min(flist)
        fmax = max(flist)
        self.parm("frange1").set(int(fmin))
        self.parm("frange2").set(fmax)
        framex = self.parm("frange1").eval()
        framey = self.parm("frange2").eval()

    hou.playbar.setFrameRange(framex, framey)
    hou.playbar.setPlaybackRange(framex, framey)
```

OnCreated scripts

OnCreated Script to set name, color, shape of node:

```
cachename = hou.ui.readInput("Enter cache name")
node = kwargs["node"]
node.setName(cachename[1])
node.setUserData('nodeshape', "tilted")
node.setColor(hou.Color(1,0,0))
```

Bake Camera from Alembic

```python
import hou
obj = hou.node("/obj")

ocam = hou.node(hou.ui.selectNode(node_type_filter=hou.nodeTypeFilter.ObjCamera))

reslist = ["1920x1080", "3840x2160", "3072x2109", "2224x1548", "Keep Original", "Custom"]
sl = hou.ui.selectFromList(reslist, message="choose resolution", sort=True, exclusive=True)
stopval = sl[0]
resolution = reslist[stopval]
if resolution is "Keep Original":
pass
elif resolution is "Custom":
custom = hou.ui.readInput("Enter resolution as #x# format, ie: 1920x1080. The x is needed.", title="Enter
Resolution")
resolution = custom[1]
ocam.parm("resx").set(resolution.split("x")[0])
ocam.parm("resy").set(resolution.split("x")[1])

else:
ocam.parm("resx").set(resolution.split("x")[0])
ocam.parm("resy").set(resolution.split("x")[1])

try:
shotinfo = hou.node("/obj/SHOTINFO")
backplate = shotinfo.parm("backplate")
except:
backplate = " "
pass

camName = ocam.name() + "_baked"
#setback = ocam.parm("vm_background").set(backplate)
tcam = obj.createNode("cam", camName)
```

```python
tcam.moveToGoodPosition()


#copy keyframes
cam_xform=["tx", "ty", "tz", "rx", "ry", "rz"]
cam_parms=["resx", "resy", "aspect", "focal", "aperture", "orthowidth", "shutter", "focus", "fstop"]
tcam.parm("vm_background").set(backplate)


parms_bake = list()
parms_bake.extend(cam_xform)
parms_bake.extend(cam_parms)
start = hou.playbar.playbackRange()[0]
end = hou.playbar.playbackRange()[1]


with hou.undos.group("bake cam"):
for x in range(int(start), int(end +1)):
hou.setFrame(x)
tcam.setWorldTransform(ocam.worldTransform())
for p in parms_bake:
parm = tcam.parm(p)
if parm.name() in cam_xform:
parm.setKeyframe(hou.Keyframe(parm.eval()))
else:
parm.setKeyframe(hou.Keyframe(ocam.parm(p).eval()))
```

Get frame range from alembic

```python
import _alembic_hom_extensions as ahe; import hou; anode = hou.pwd(); cam =
anode.parm("camera").eval(); fpath = hou.node(cam).parm("fileName"); ver =
fpath.eval().split("_")[-1].split("/")[0]; print(ver); name = hou.node(cam).children() ; print(name[0]);
anode.parm("cameraVer").set(ver); name = str(name[0]); anode.parm("cameraName").set(name);
alembicpath= fpath.eval(); timerange = ahe.alembicTimeRange(alembicpath);
start_time=timerange[0]*hou.fps(); end_time = timerange[1]*hou.fps();
anode.setParms({"framemin":start_time, "framemax":end_time})
```

Example On Node Buttons

This is how a button on a non-adminned node needs to be laid out:

import hou ; do something ;

This grabs frame range from supplied alembic camera, updates names, etc:

```
import _alembic_hom_extensions as ahe; import hou; anode = hou.pwd(); cam = anode.parm("camera").eval();
fpath = hou.node(cam).parm("fileName"); ver = fpath.eval().split("_")[-1].split("/")[0]; print(ver); name =
hou.node(cam).children() ; print(name[0]); anode.parm("cameraVer").set(ver); name = str(name[0]);
anode.parm("cameraName").set(name); alembicpath= fpath.eval(); timerange =
ahe.alembicTimeRange(alembicpath); start_time=timerange[0]*hou.fps(); end_time = timerange[1]*hou.fps();
anode.setParms({"framemin":start_time, "framemax":end_time})
```

This sets frame range from ranges on node:

```
import hou; anode = hou.pwd(); start = anode.parm("framemin").eval(); end = anode.parm("framemax").eval();
hou.playbar.setFrameRange(start, end); hou.playbar.setPlaybackRange(start, end)
```

This grabs info from animation alembic:

```
import hou; anode = hou.pwd(); anim = anode.parm("anim").eval(); fpath = hou.node(anim).parm("fileName");
ver = fpath.eval().split("_")[-1].split("/")[0]; name = hou.node(anim) ; anode.parm("animVer").set(ver); name =
```

```
str(name); anode.parm("animName").set(name);
```

Send to Tops

```python
import hou
nodes = hou.selectedNodes()
topnet = hou.node("/obj/topnet1")
for x in nodes:
    name = x.name()
    fstart = x.parm("f1").eval()
    fend = x.parm("f2").eval()
    ver = x.parm("version")
    path = x.path()

    topcache = topnet.createNode("ropfetch")
    if x.parm("cachesim").eval() is 1:
        topcache.parm("batchall").set(1)
        topcache.setColor(hou.Color(.5,0,0))
    else:
        topcache.parm("framesperbatch").set(15)
        topcache.setColor(hou.Color(0,.5,0))
    p = hou.IntParmTemplate("version", "Version", 1)
    g = topcache.parmTemplateGroup()
    g.append(p)
    topcache.setParmTemplateGroup(g)
    topcache.parm("version").set(ver)
    topcache.parm("roppath").set(path)
    topcache.setName(name)
```

Set Frame Range without Script

```
import hou; anode = hou.pwd(); start = anode.parm("framemin").eval(); end = anode.parm("framemax").eval();
hou.playbar.setFrameRange(start, end); hou.playbar.setPlaybackRange(start, end)
```

ROP

PostRender open MPLAY:

```
import os; img_path = "`chs("picture")`".replace("$F", "\*"); os.system("mplay %s" % img_path)
```

Pull version from hipname

Can be used in a parameter. Returns a integer.

```python
import hou, re
version = re.findall('_v\d+', hou.hipFile.basename())[0]
version = int(re.findall('\d+', version)[0])
return version
```

Get Frame Size of Image File

```
node= hou.pwd(); bg=node.parm("image").eval(); res=hou.imageResolution(bg);
node.parm("framesizex").set(res[0]); node.parm("framesizey").set(res[1]);
```

Python

Create Agent from built in mocap rig - Simple

This will take a test mocap rig 3 built in to houdini, create an agent, add agent clips from the built in library (must be in your scene already), and lay everything out.

```python
import hou
obj = hou.node("/obj")
agentname = hou.ui.readInput("Agent Name", title="Name")
agentname =  agentname[1]


agentpNode = obj.createNode("geo")
agentNode = agentpNode.createNode("agent")


agentpNode.setName(agentname)
agentNode.setName(agentname)
agentnet = agentpNode


inputMoc = hou.ui.selectNode(title="Select Character Rig to base agent on", multiple_select=False,
node_type_filter=hou.nodeTypeFilter.ObjSubnet)



agentNode.parm("agentname").set(agentname)
agentNode.parm("objsubnet").set(inputMoc)



#clips = hou.ui.selectNode(multiple_select=True, node_type_filter=hou.nodeTypeFilter.Obj)
clips = hou.ui.selectNode(title="Select all clips - mocap biped 3 rigs", multiple_select=True,
node_type_filter=hou.nodeTypeFilter.ObjSubnet)
agentclip = agentnet.createNode("agentclip")
clipprops = agentnet.createNode("agentclipproperties")


clipnum = len(clips)
```

```
clipcount = 1


agentclip.parm("locomotionnode").set("Hips")



#print(clipnum)
for x in clips:
    node = hou.node(x)
    name = node.name()
    path = str(x)
    nframes = node.parm("nFrames")
    fs =hou.playbar.timelineRange()[0]
    fe = int(fs) + int(nframes.eval())
    agentclip.parm("framerange" + str(clipcount) +"_1").deleteAllKeyframes()
    agentclip.parm("framerange" + str(clipcount) +"_2").deleteAllKeyframes()
    agentclip.parm("clips").set(clipnum)
    agentclip.parm("name" + str(clipcount)).set(name)
    agentclip.parm("objsubnet" + str(clipcount)).set(path)
    agentclip.parm("framerange" + str(clipcount) +"_1").set(fs)
    agentclip.parm("framerange" + str(clipcount) +"_2").set(fe)
    agentclip.parm("converttoinplace" + str(clipcount)).set(True)

    clipprops.parm("numclips").set(clipnum)
    clipprops.parm("clipname_" + str(clipcount)).set(name)
    clipprops.parm("enableblending_" + str(clipcount)).set(True)
    clipprops.parm("framesbefore_" + str(clipcount)).set("5")
    clipprops.parm("framesafter_" + str(clipcount)).set("5")




    clipcount = clipcount+1

agentclip.setFirstInput(agentNode)
clipprops.setFirstInput(agentclip)
agentpNode.layoutChildren()
```

Send selected nodes to new object (fancy way to create object merges)

```python
import hou
nodes = hou.selectedNodes()
destnode = hou.node(hou.ui.selectNode(title="select destination node"))
for x in nodes:
    name = x.name()+"_"+str(x.parent())
    path= x.path()
    objmergenode = destnode.createNode("object_merge")
    objmergenode.parm("objpath1").set(path)

    objmergenode.setName(name)
    objmergenode.setColor(hou.Color(0,1,0))
```

Send nodes to new alembic export

```python
import hou
nodes = hou.selectedNodes()
destnode = hou.node("/obj/EXPORT")
if not destnode:
    destnode = hou.node("/obj").createNode("geo")
    destnode.setName("EXPORTS")

for x in nodes:
    name = x.name()+"_"+str(x.parent())
    path= x.path()
    objmergenode = destnode.createNode("object_merge")
    objmergenode.parm("objpath1").set(path)

    objmergenode.setName(name)
    objmergenode.setColor(hou.Color(0,1,0))

    alembicrop = destnode.createNode("rop_alembic")
    alembicrop.setName(str(x.parent()))
    alembicrop.parm("trange").set(1)
    alembicrop.parm("f1").deleteAllKeyframes()
    alembicrop.parm("f1").set(int("1001"))
    alembicrop.parm("build_from_path").set(1)
    alembicrop.setInput(0, objmergenode)
    destnode.layoutChildren()
```

TOPs - symlink output file of parent

```python
exportFile = str(work_item.expectedInputFiles[0])

dir = os.path.dirname(exportFile)

newFile = os.path.join(dir, "cache.abc")


os.symlink(exportFile, newFile)
```

VEX

Camera Stuff

auto focus, get distance from object and camera:

`vlength(vtorigin("/obj/geo1", "/obj/cam1"))`

Points

divide points into 3 equal parts:

```
i@part = floor(fit(rand(@ptnum+.258), 0, 1, 0, 2.9));
```

point normal to center:

If then statements

If the pscale is greater than .4 then set it to .2, if not set it to its current pscale
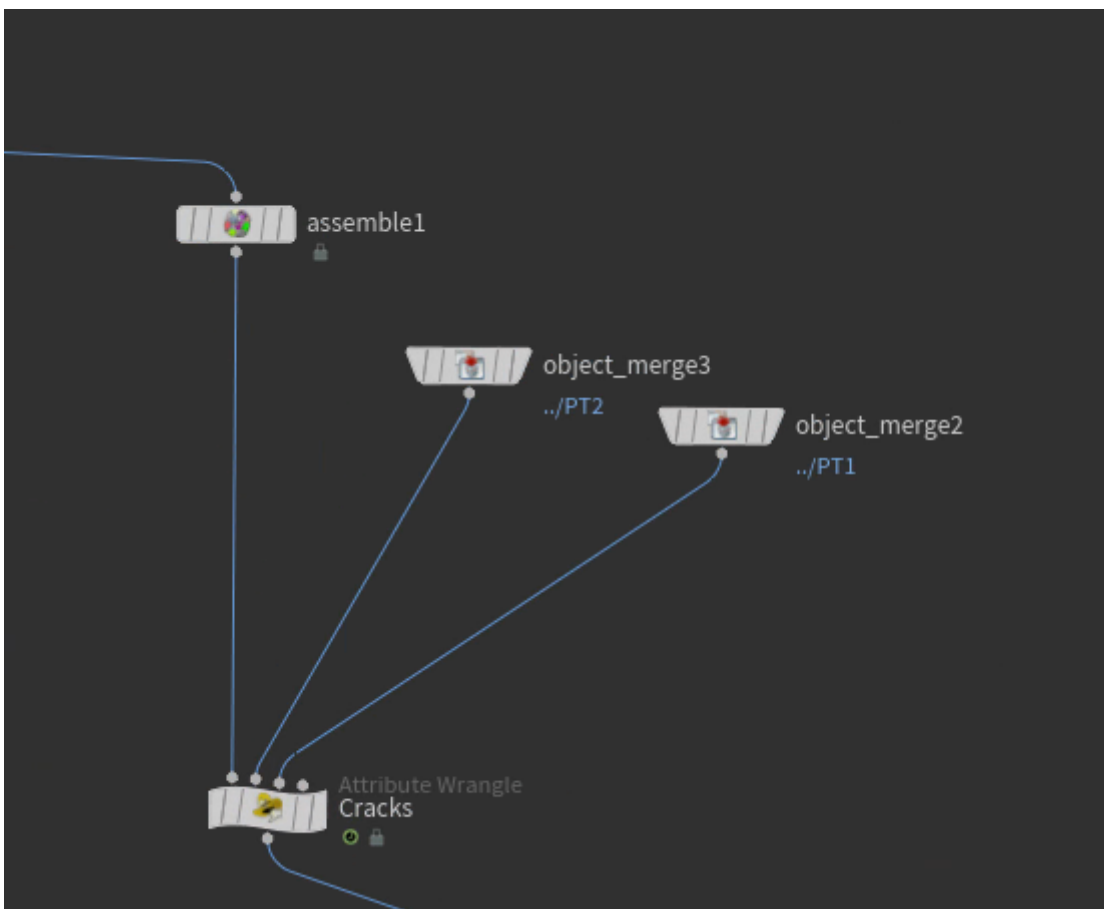
`@pscale = @pscale>.4?.2:@pscale`

Transforms and Junk

1. transforms to attribute matrix:

```
p@orient = quaternion(3@transform);
v@scale = cracktransform(0,0,2,set(0.0.0). 3@transform);
```

## 2. rotate packed fracture based on point + distance:


```
vector p1= set(@P.x, @P.y, @P.z);

vector crack1 = point(1, "P", 0);
vector crack2 = point(2, "P", 0);
vector p2 = crack1-p1;
vector p3 = crack2-p1;
float n = fit ( length ( p2 ), 0, ch("maxdist"), ch('mult'), 0 );
float n2 =  fit ( length ( p3 ), 0, ch("maxdist2"), ch('mult2'), 0 );
```

```
vector4 q0 = quaternion ( 0 );

vector4 q1 = sample_orientation_uniform ( rand ( @ptnum ) );

vector4 q2 = slerp ( q0, q1, n+n2 );

matrix3 xform = qconvert ( q2 );


setprimintrinsic ( 0, "transform", @ptnum, xform );
```
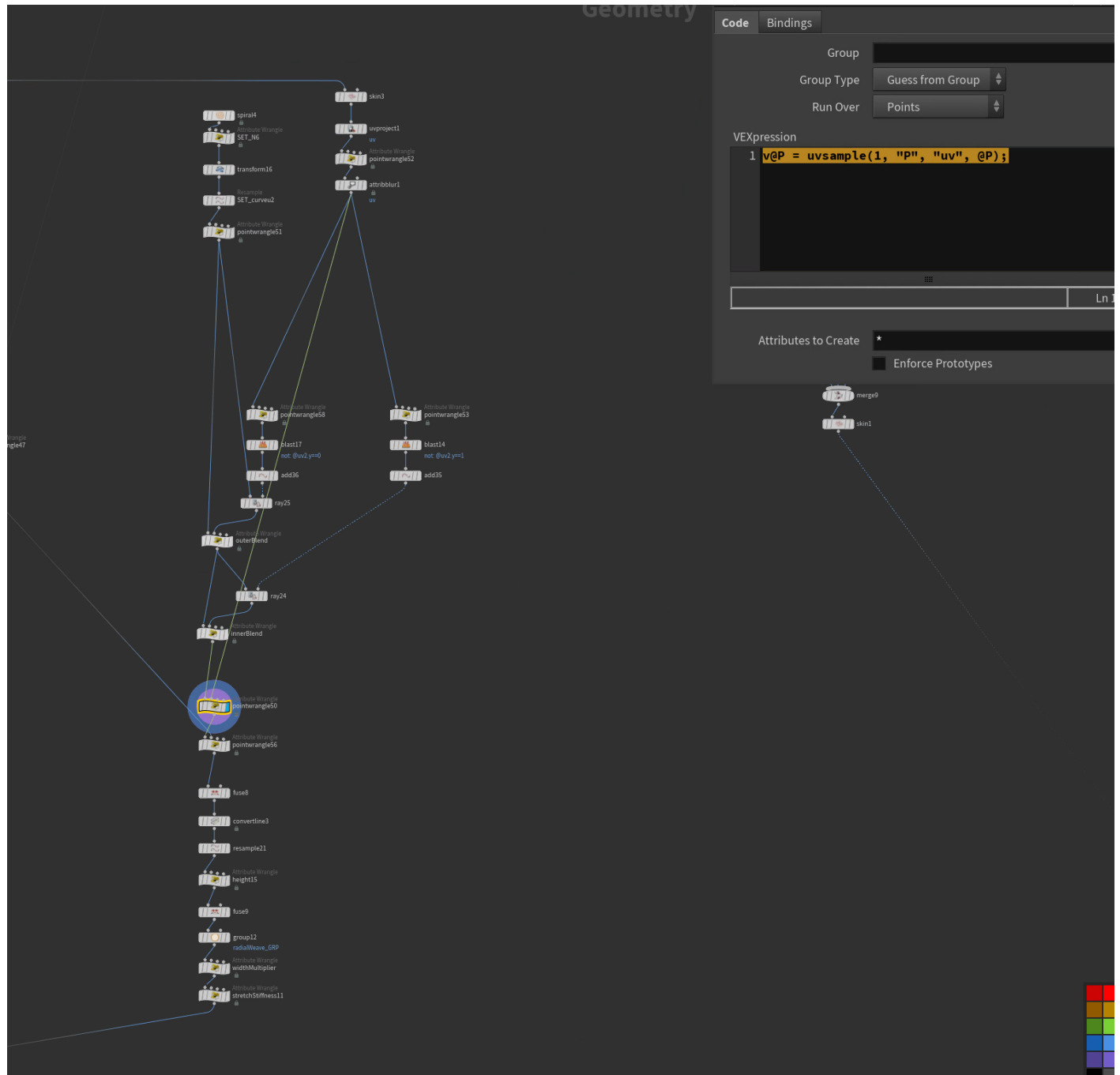
3. Blending spiral (end beg):
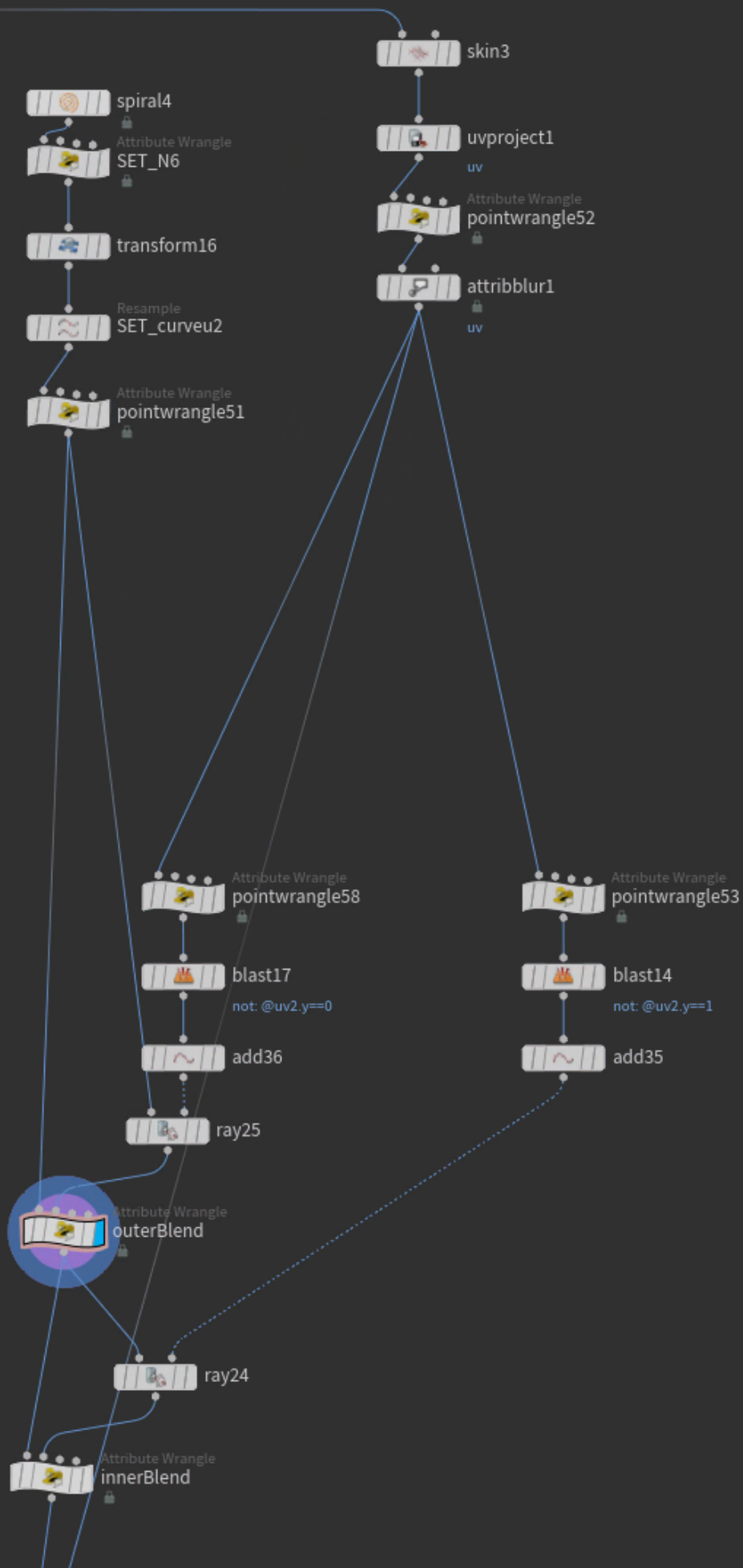
```
vector target = point(1, "P", @ptnum);

float blend = chramp("blendAlongSpiral", @curveu)*chf("multiplier");


@P = lerp(@P, target, blend);
```

4. Position copy via uv:


```
v@P = uvsample(1, "P", "uv", @P);
```

5. move near points together:

```
int near = nearpoint(1, @P);
vector target = point(1, "P", near);
@P = target;
```

Orientation

Get transform and orientation from camera:

```
string camera = "/obj/alembicarchive1/Camera2/CameraShape2"; // path to your camera
@P = ptransform(camera, "space:current", {0,0,0});
@N = ntransform(camera, "space:current", {0,0,-1});
```