

# VEX

- Camera Stuff
- Points
- If then statements
- Transforms and Junk
- Orientation

# 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

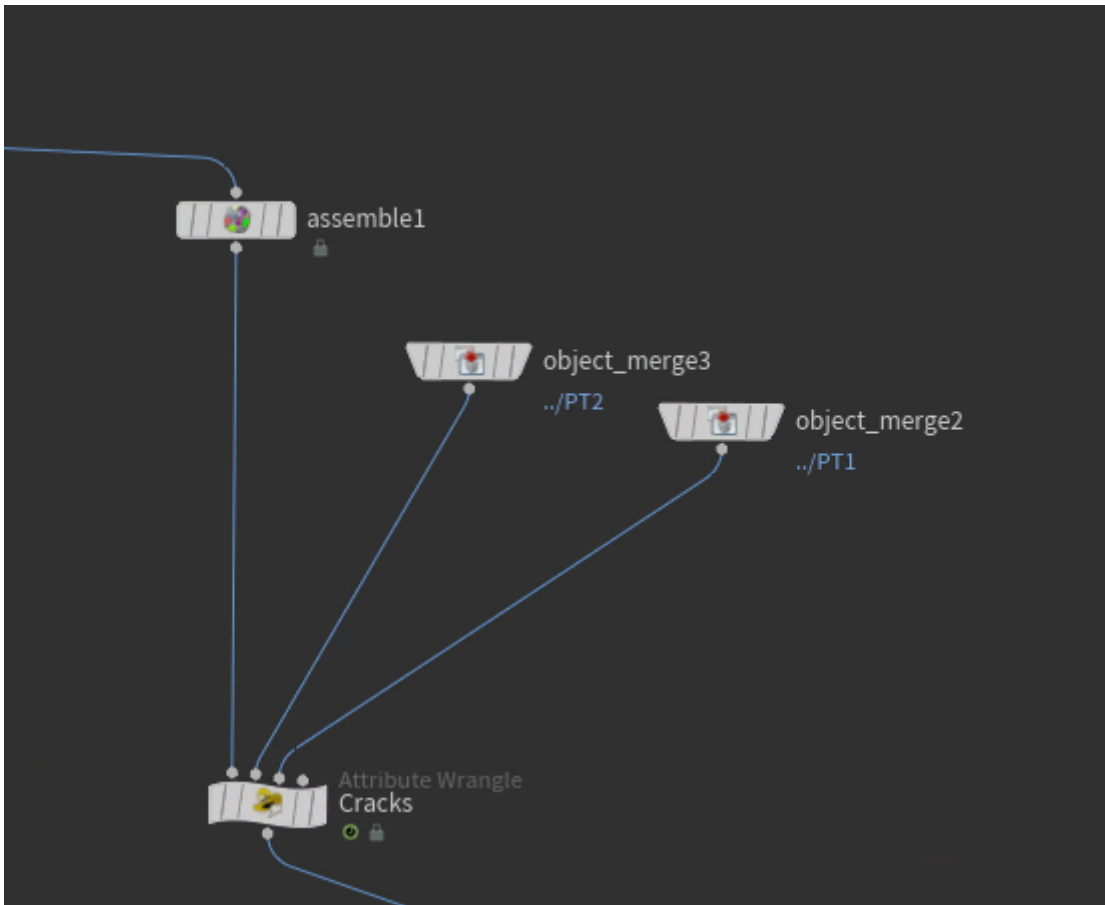
```
@pscale = @pscale>.4?.2:@pscale
```

# 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);
```

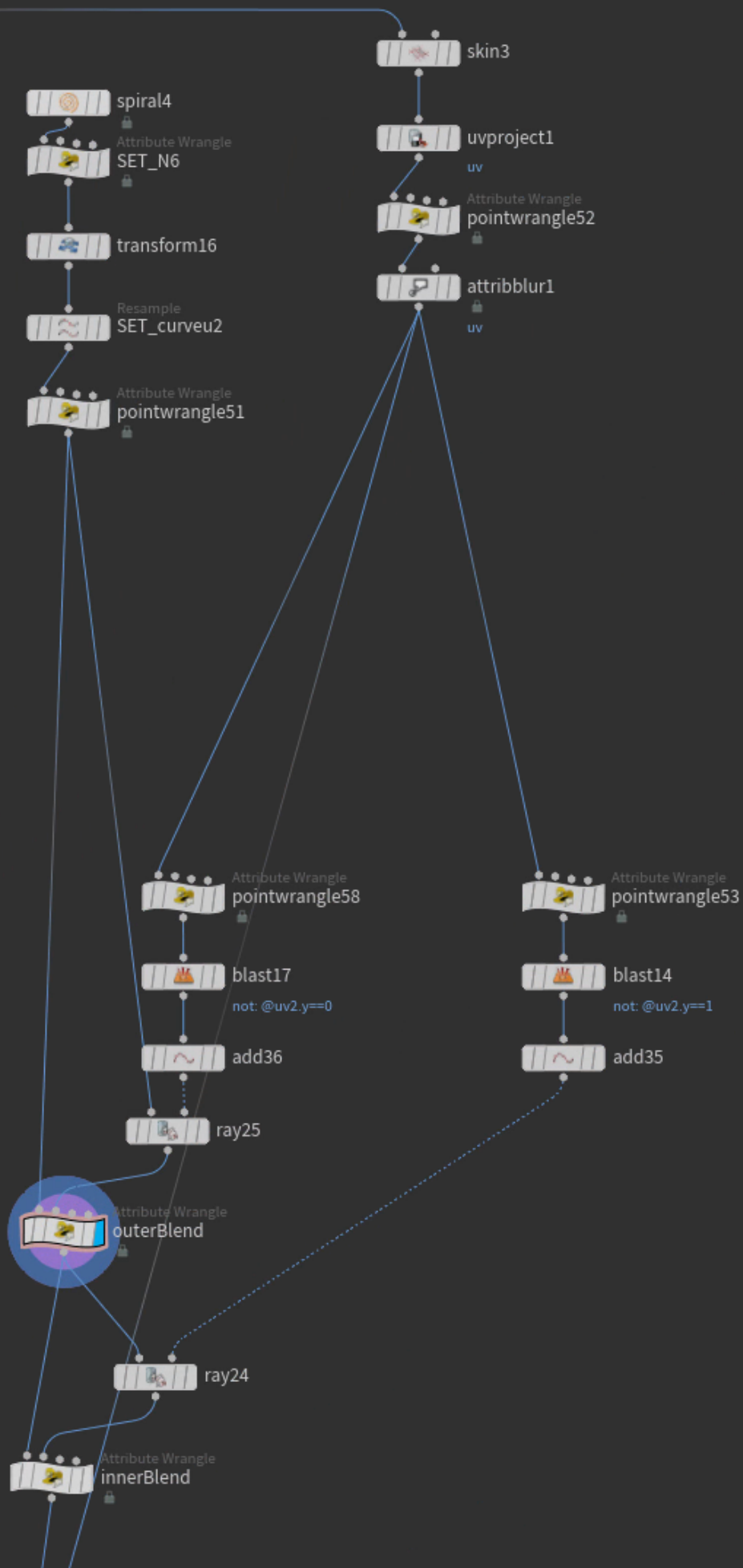
## 2. rotate packed fracture based on point + distance:



```
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 );
```

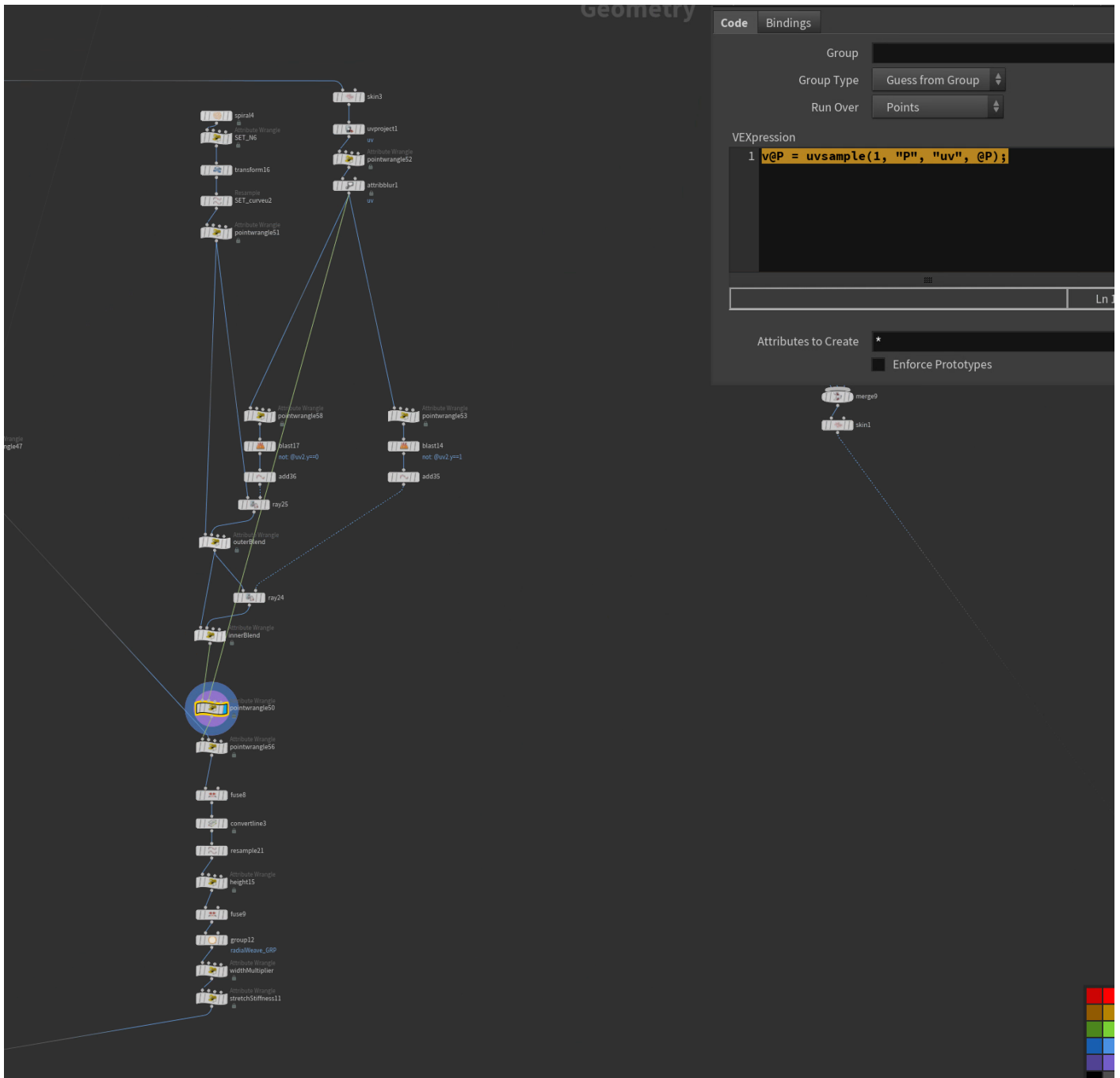
### 3. Blending spiral (end beg):



```
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});
```