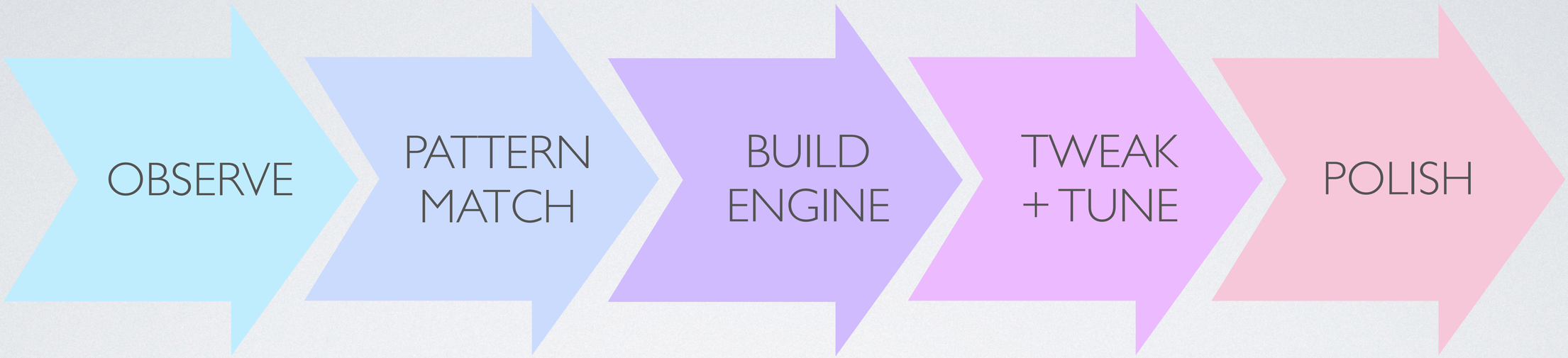
PROCEDURAL DEMO TIPS AND TRICKS

Inigo Quilez (source)

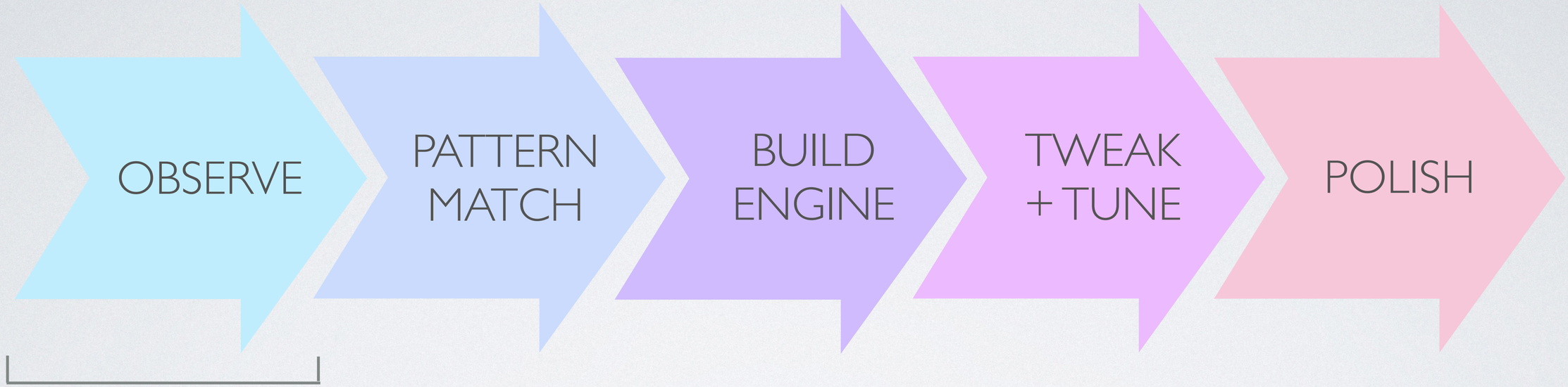
Adding professional polish to your demos

THE PROCEDURAL PIPELINE*



* if we HAD to put a formula to it...

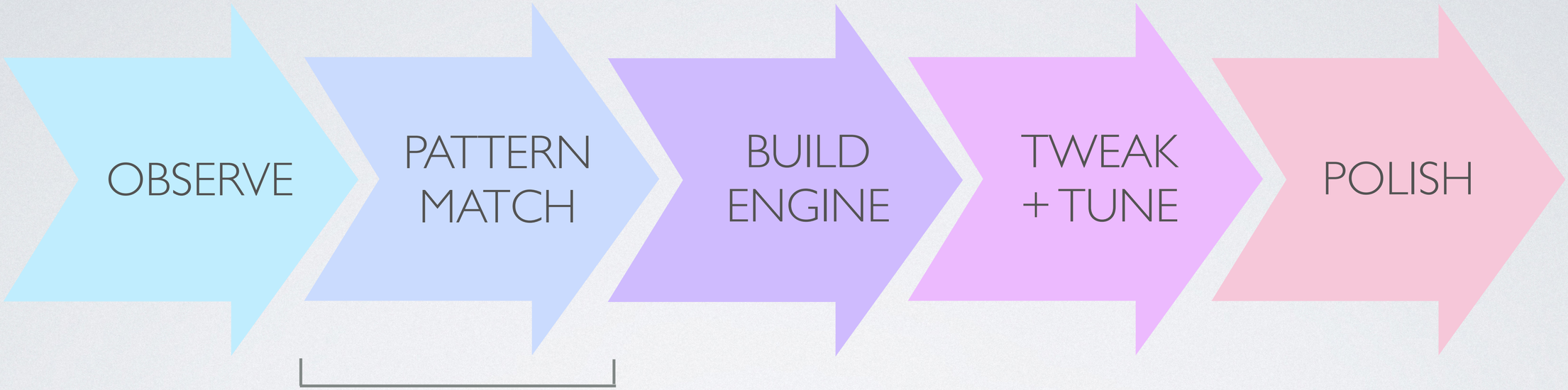
THE PROCEDURAL PIPELINE*



Identify the key characteristics of my desired output?

* if we HAD to put a formula to it...

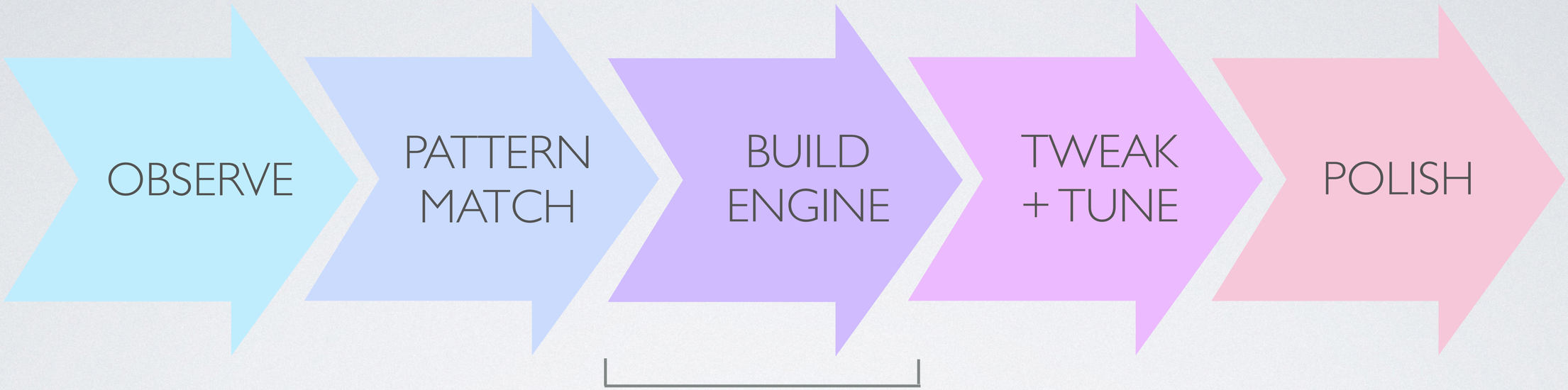
THE PROCEDURAL PIPELINE*



Identify relevant techniques or algorithms. Eg. toolbox functions

* if we HAD to put a formula to it...

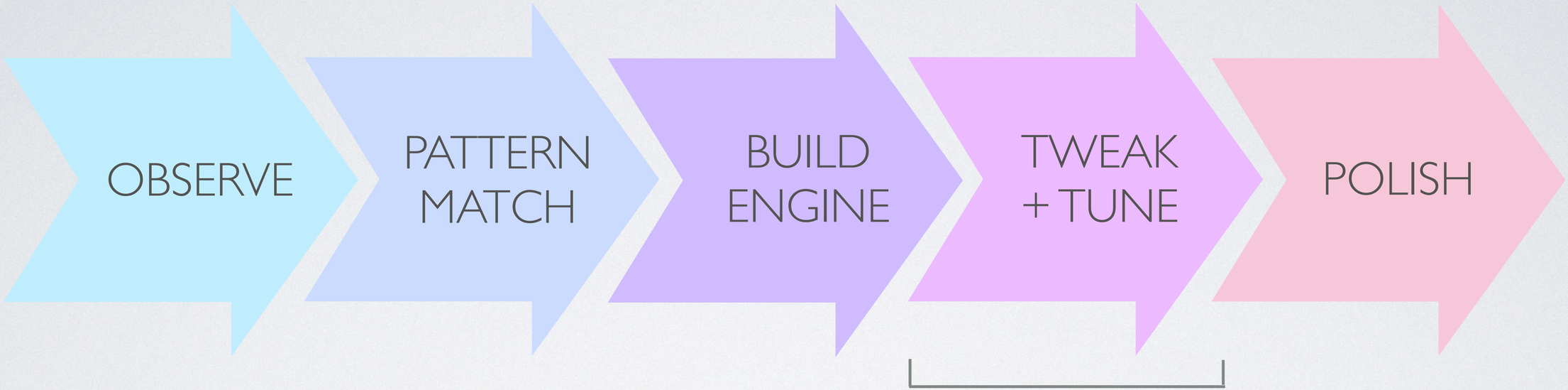
THE PROCEDURAL PIPELINE*



Implement your generator, parameterizing as you go

* if we HAD to put a formula to it...

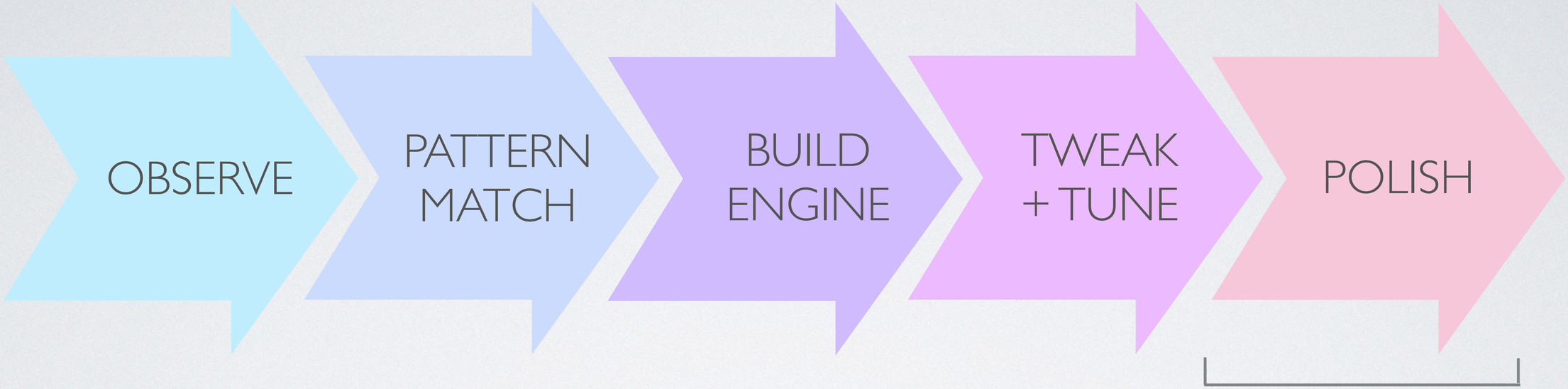
THE PROCEDURAL PIPELINE*



Using your generator, modify the default parameters and their operant ranges to tune your output

* if we HAD to put a formula to it...

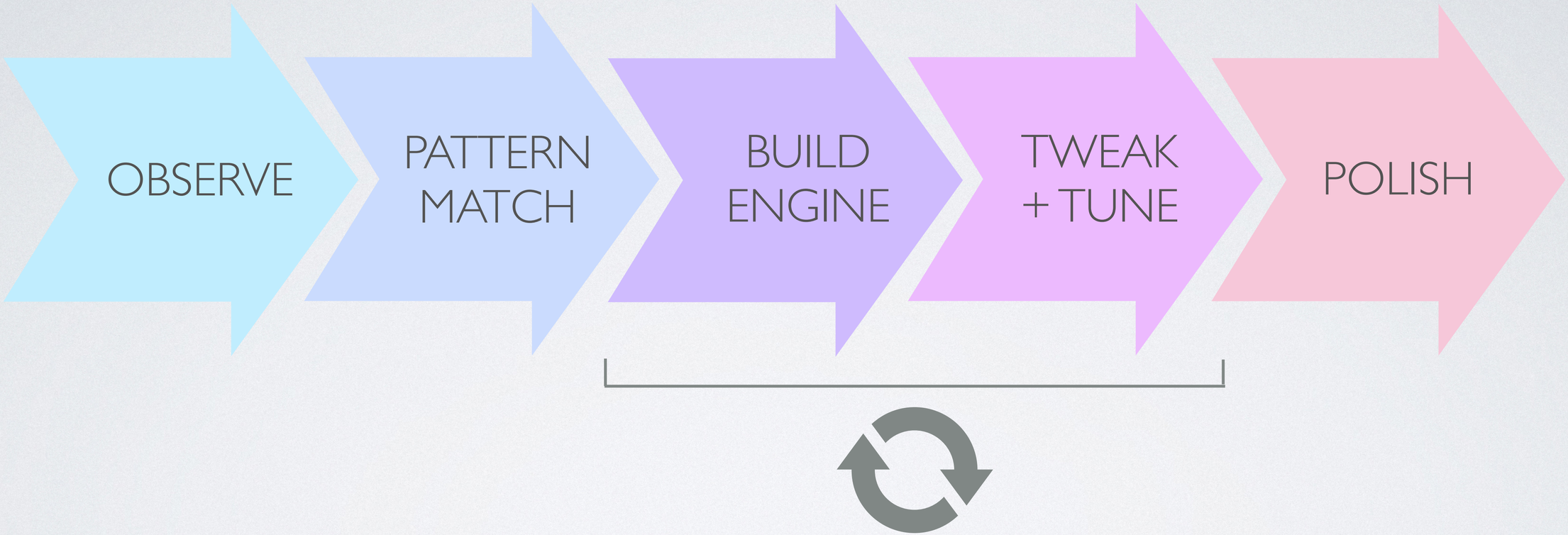
THE PROCEDURAL PIPELINE*



Bring your scene to life! Play with "extras" to polish your visuals

* if we HAD to put a formula to it...

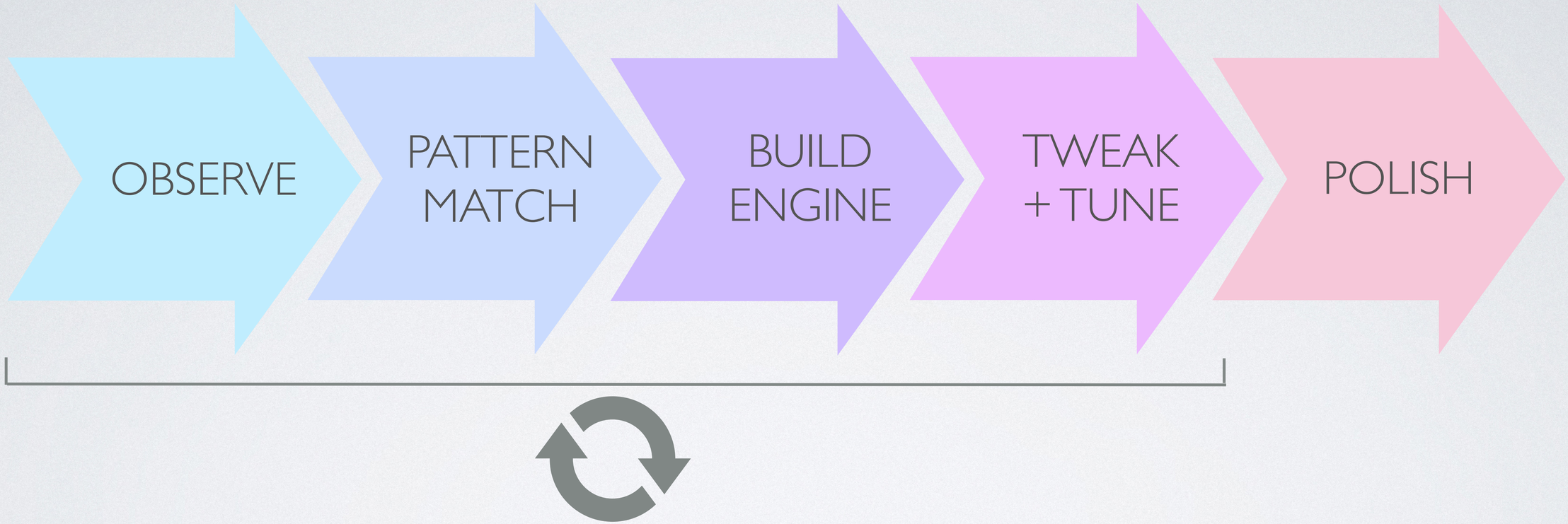
THE PROCEDURAL PIPELINE*



BTW, expect to iterate a lot here....

* if we HAD to put a formula to it...

THE PROCEDURAL PIPELINE*



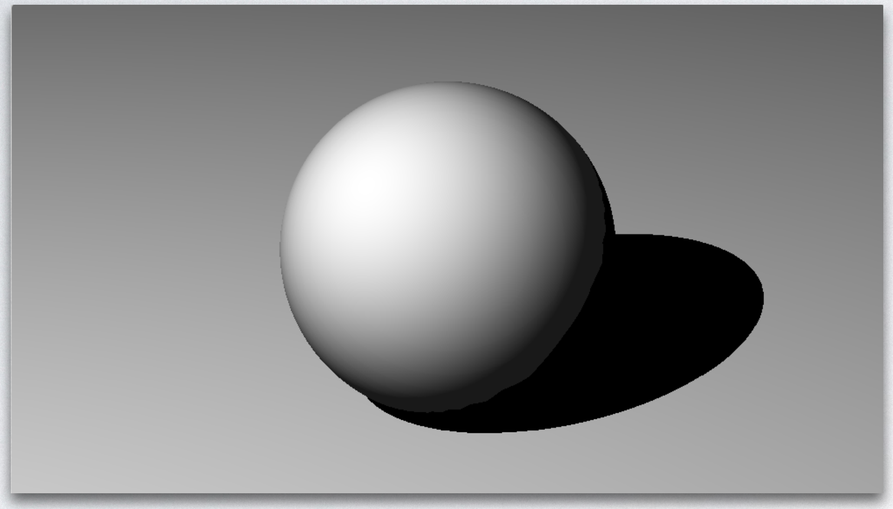
Or even here... going back to the drawing board is ok!

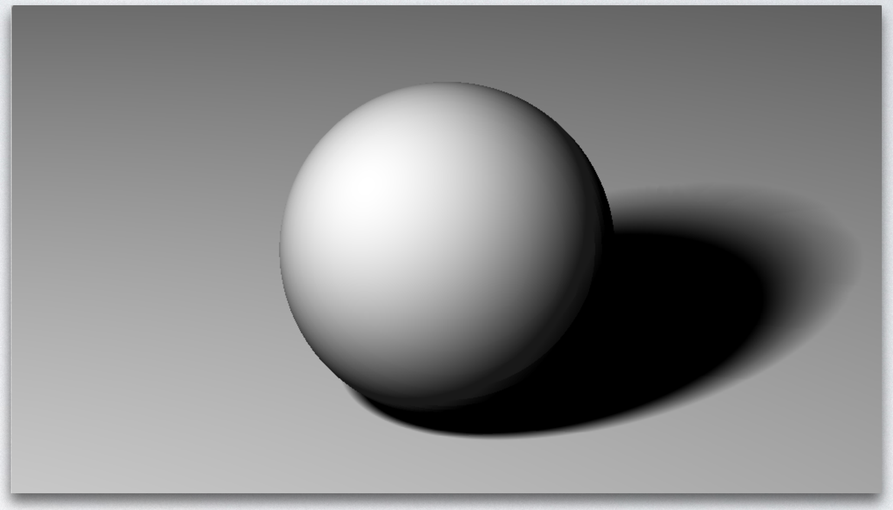
* if we HAD to put a formula to it...

KEEP YOUR GENERATOR TIDY

- Do not repeat yourself
- Lots of helper functions
- No "magic numbers"
- Constrain your input / output

SOFT SHADOWS

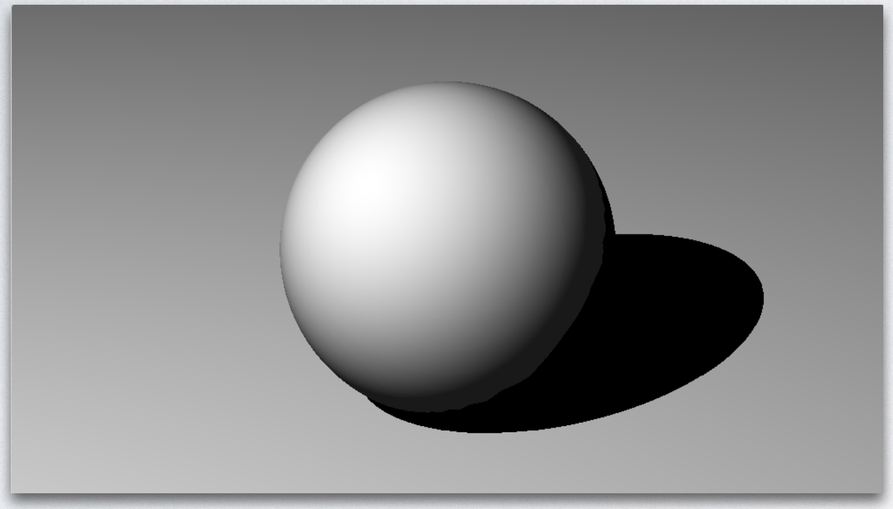




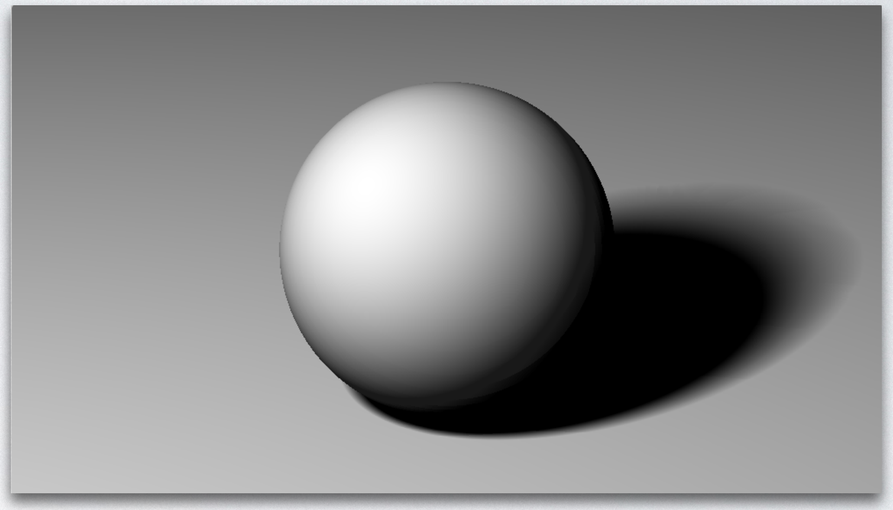
Observations:

- The closer our shadow feeler is to hitting an edge, the darker the shadow
- The closer our shadow feeler is to our feeler origin, the darker the shadow

SOFT SHADOWS

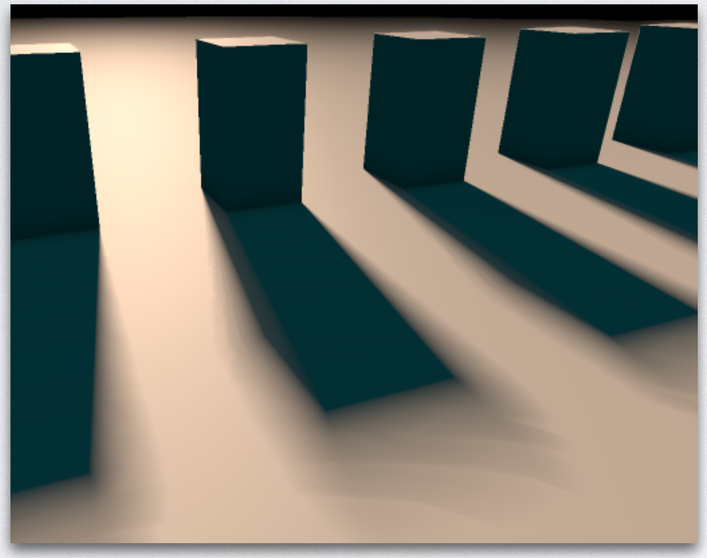


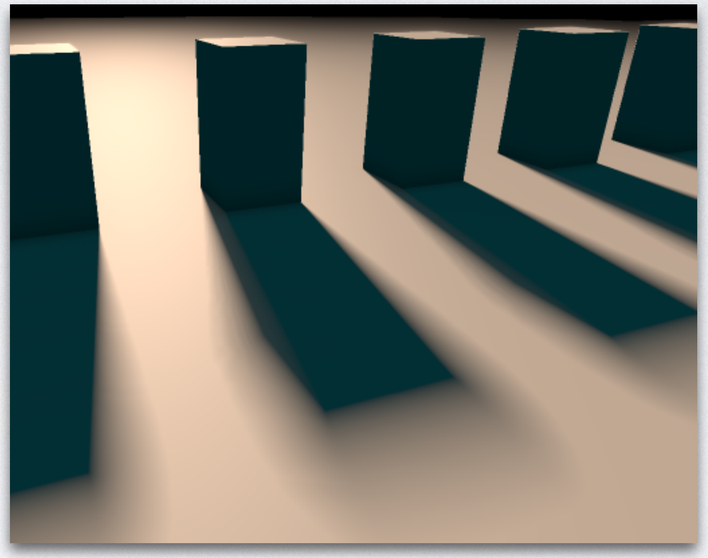
```
float shadow( in vec3 ro, in vec3 rd, float mint, float maxt )
{
    for( float t=mint; t<maxt; )
    {
        float h = sceneSDF(ro + rd*t);
        if( h<0.001 )
            return 0.0;
        t += h;
    }
    return 1.0;
}
```



```
float softshadow( in vec3 ro, in vec3 rd, float mint, float maxt, float k )
{
    float res = 1.0;
    for( float t=mint; t<maxt; )
    {
        float h = sceneSDF(ro + rd*t);
        if( h<0.001 )
            return 0.0;
        res = min( res, k*h/t );
        t += h;
    }
    return res;
}
```

SOFT SHADOWS



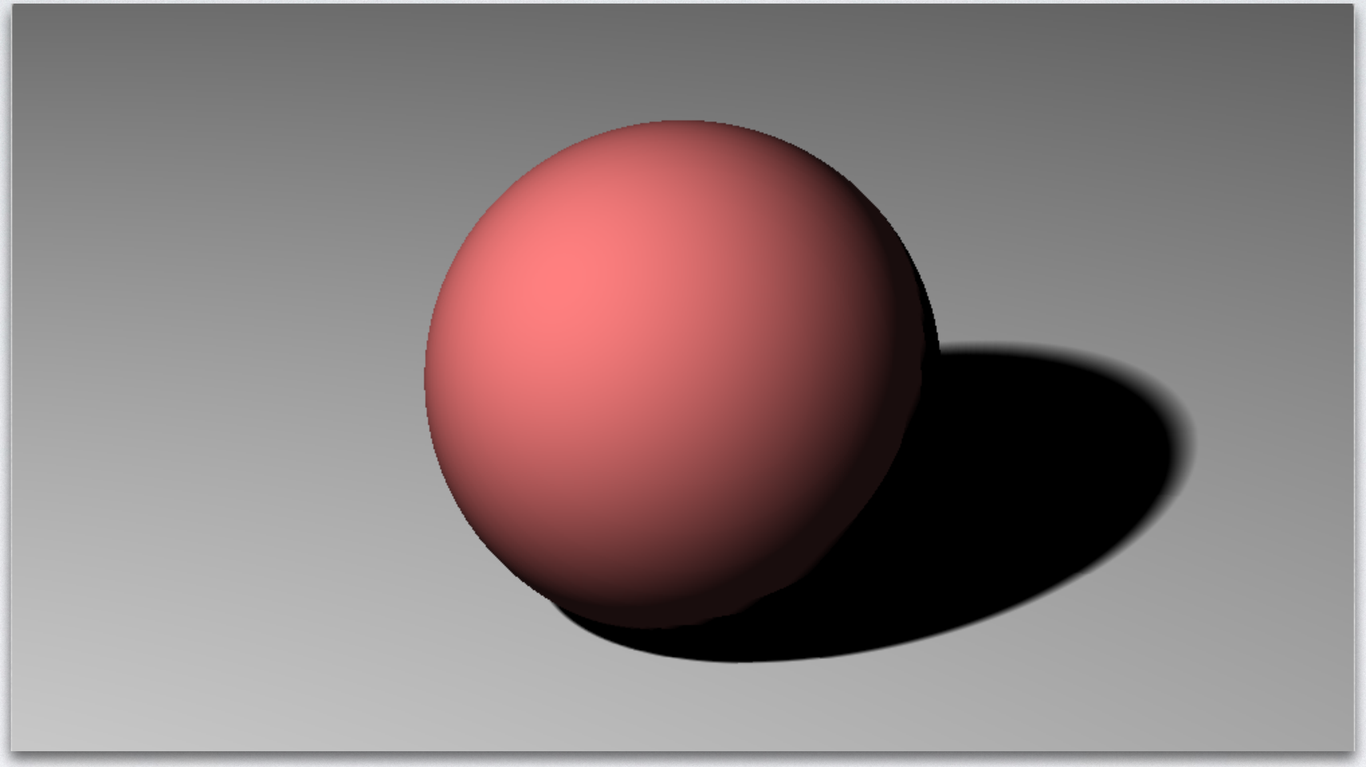


Inigo Quilez (source)

Some tricky cases can produce artifacts as the shadow feeler "misses" darkest penumbra areas

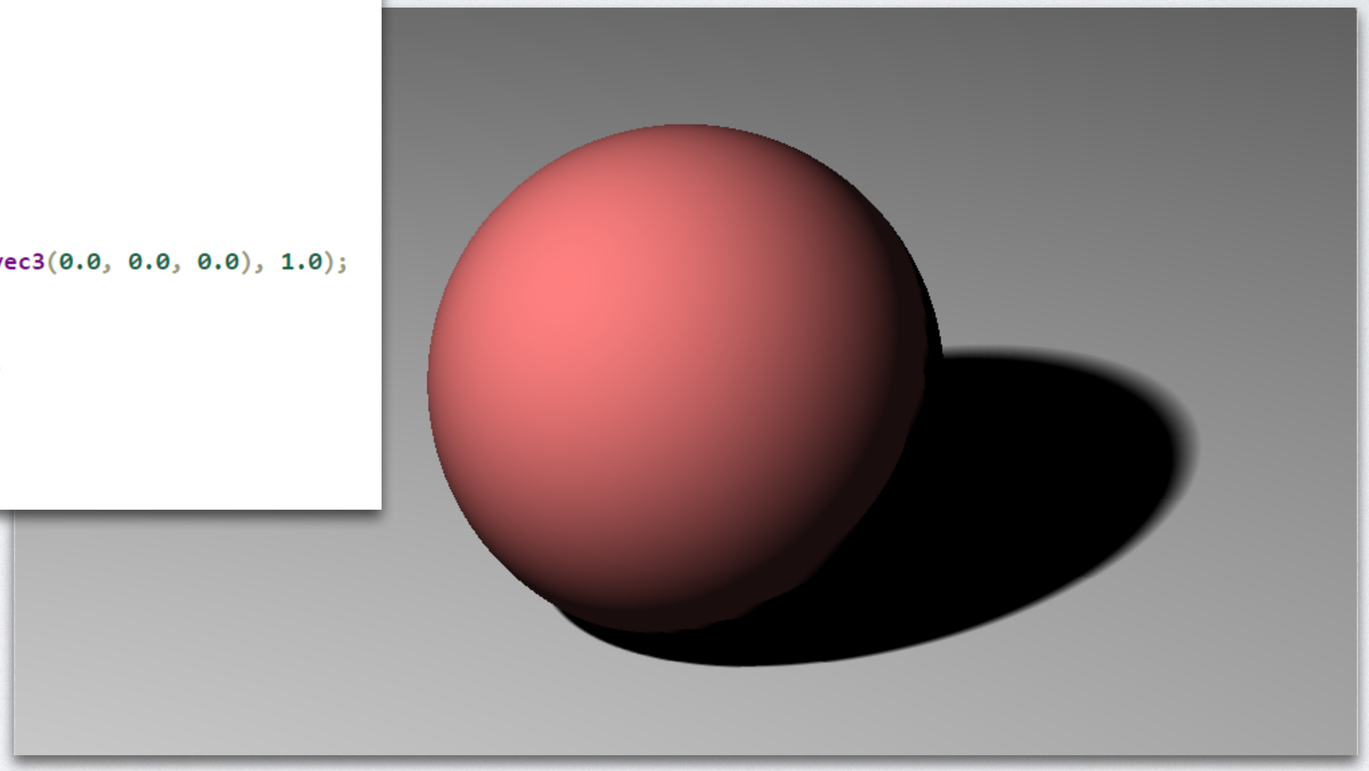
See: Sebastian Aaltonen via IQ

MULTIPLE MATERIALS



MULTIPLE MATERIALS

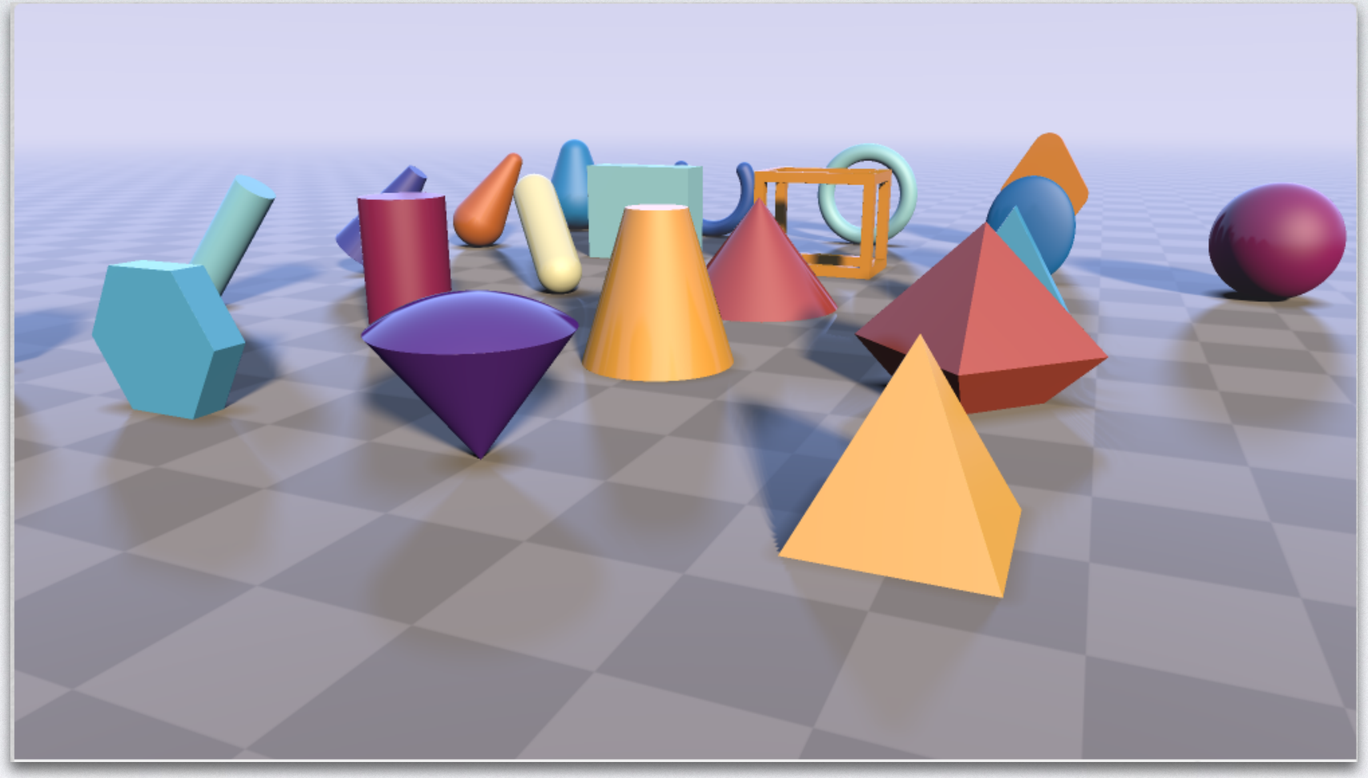
```
Geo minSDF(Geo geo1, Geo geo2)
{
    if (geo1.dist < geo2.dist)
    {
        return geo1;
    }
    return geo2;
}

Geo sceneSDF(vec3 queryPos)
{
    Geo sphere;
    sphere.dist = sdfSphere(queryPos, vec3(0.0, 0.0, 0.0), 1.0);
    sphere.material_id = 1;

    Geo plane;
    plane.dist = heightField(queryPos);
    plane.material_id = 2;

    return minSDF(sphere, plane);

}
```



```
struct Geo
{
    float dist;
    int material_id;
};
```

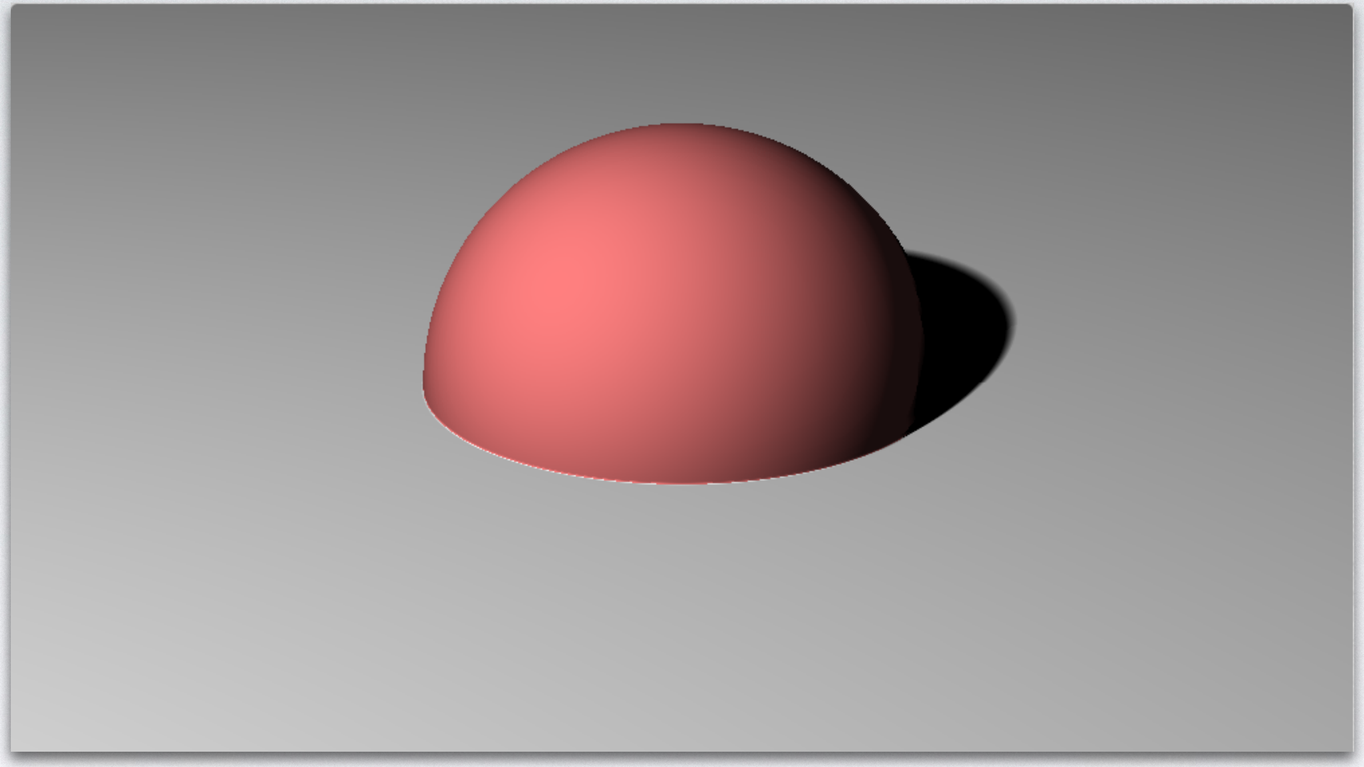
We can store material info alongside our SDF info

MULTIPLE MATERIALS



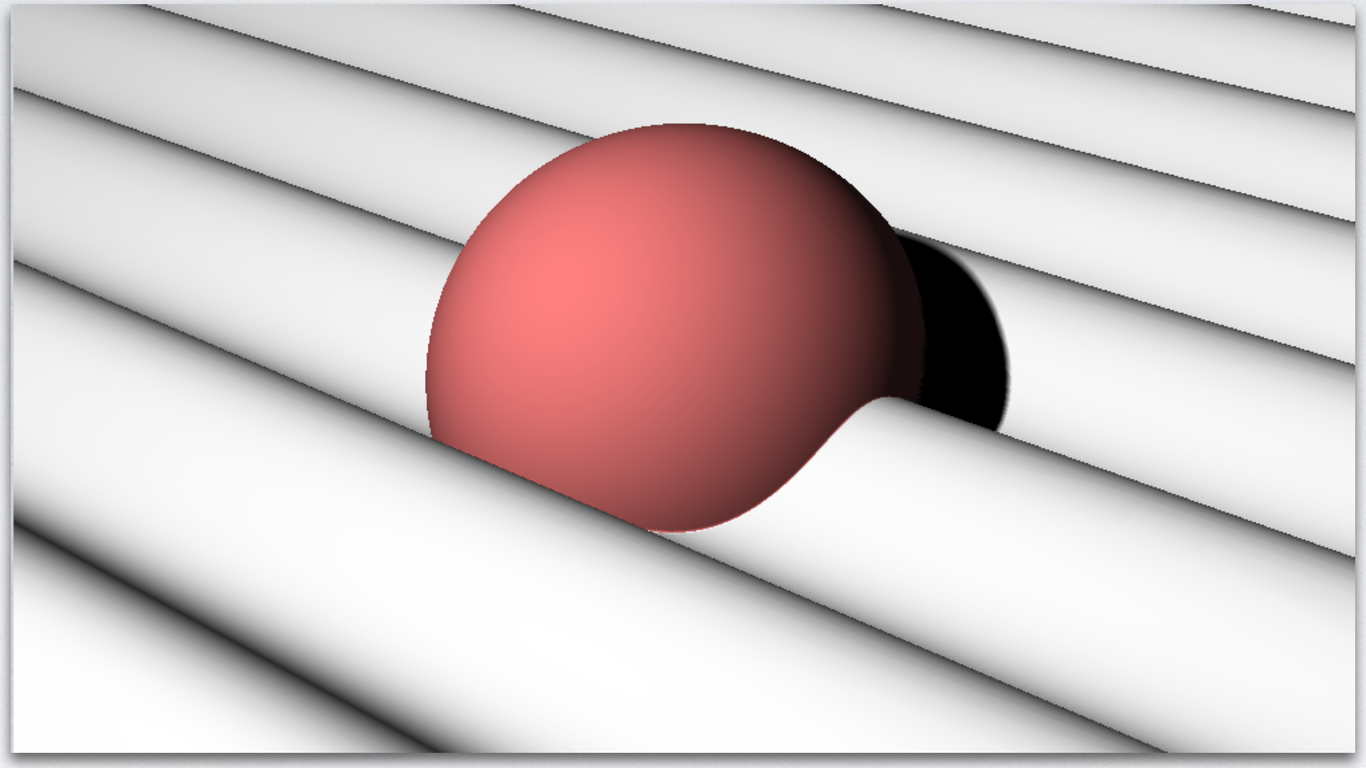
Inigo Quilez (source)

HEIGHTFIELDS



```
float heightField(vec3 queryPos, float planeHeight)
{
    return queryPos.y - planeHeight;
}
```

HEIGHTFIELDS

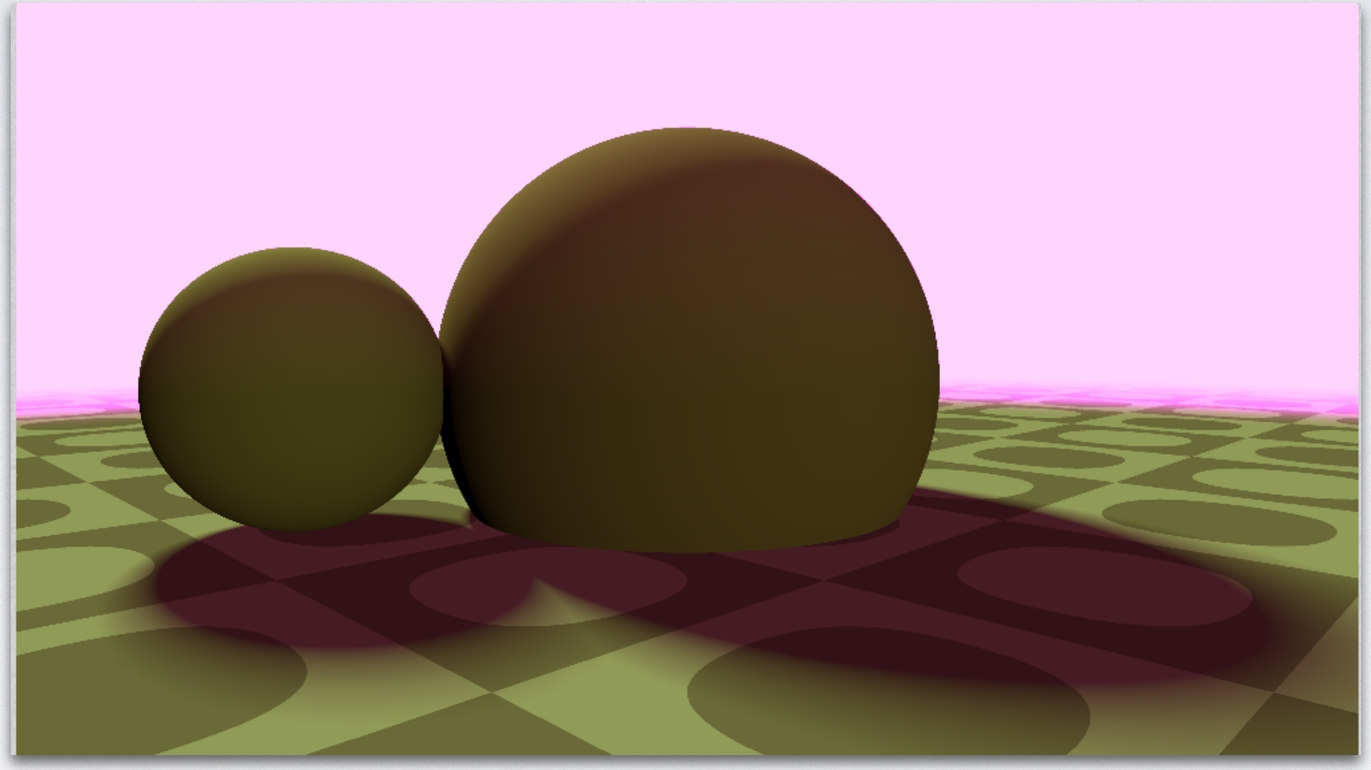


```
float heightField(vec3 queryPos)
{
    float waveHeightScale = 0.2;
    float waveFrequencyScale = 3.0f;
    return queryPos.y - cos(waveFrequencyScale * (queryPos.x - queryPos.z)) * waveHeightScale;
}
```

POLISH BREAKDOWN

Inigo Quilez (source)

LIGHTING AND COLORING IDEAS



Adam Mally (source)