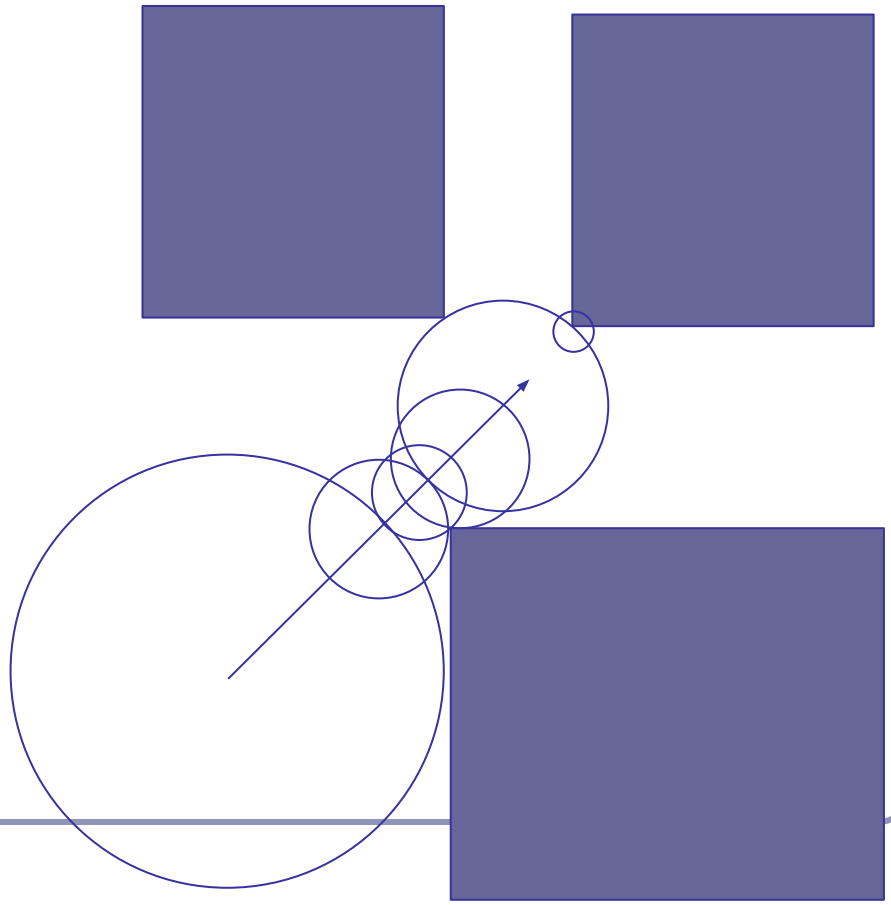
# Raymarching Signed Distance Fields

To raytrace or raycast implicit functions, consider *signed distance fields*.

1. Fire ray into scene
2. At each step, measure distance field function: $d$ = [distance to nearest object in scene]
3. Advance ray along ray heading by distance $d$

Early paper:

http://graphics.cs.illinois.edu/sites/default/files/rtqjs.pdf



# Raymarching Signed Distance Fields

Sample distance functions

```
float sdSphere(
  vec3 p, float s) {
   return length(p)-s;
}
```

```
float sdBox(
  vec3 p, vec3 b) {
   vec3 d = abs(p) - b;
   return
  min(max(d.x,max(d.y,d.z)),
  0.0) + length(max(d,0.0));
}
```

Sample distance functions

```
float sdCylinder(
  vec3 p, vec3 c) {
   return length(p.xz-c.xy)-
   c.z;
}
```

```
float sdTorus(
  vec3 p, vec2 t) {
  vec2 q = vec2(length(p.xz)-
t.x,p.y);
  return length(q)-t.y;
}
```

Source: http://iquilezles.org/www/articles/distfunctions/distfunctions.htm

# Raymarching Signed Distance Fields

```
vec3 raymarch(vec3 pos, vec3 raydir) {
  int step = 0;
  float d = getSdf(pos);

  while (abs(d) > 0.001 && step < 50) {
    pos = pos + raydir * d;
    d = getSdf(pos);
    step++;
  }

  return
      (step < 50) ? illuminate(pos, rayorig)) : background;
}
```

# Raymarching Signed Distance Fields

Finding the normal: compute the local gradient

```
float d = getSdf(hitpoint);
vec3 normal = normalize(vec3(
    getSdf(vec3(pt.x + 0.0001, pt.y, pt.z)) - d,
    getSdf(vec3(pt.x, pt.y + 0.0001, pt.z)) - d,
    getSdf(vec3(pt.x, pt.y, pt.z + 0.0001)) - d));
```