Intro 4kb
sphere tracing

Maciej Matyka (maq / floppy)
Dla koła naukowego Voxel 05.2012

Co to są intra 4kb?

Demoscena

Demo

Intro

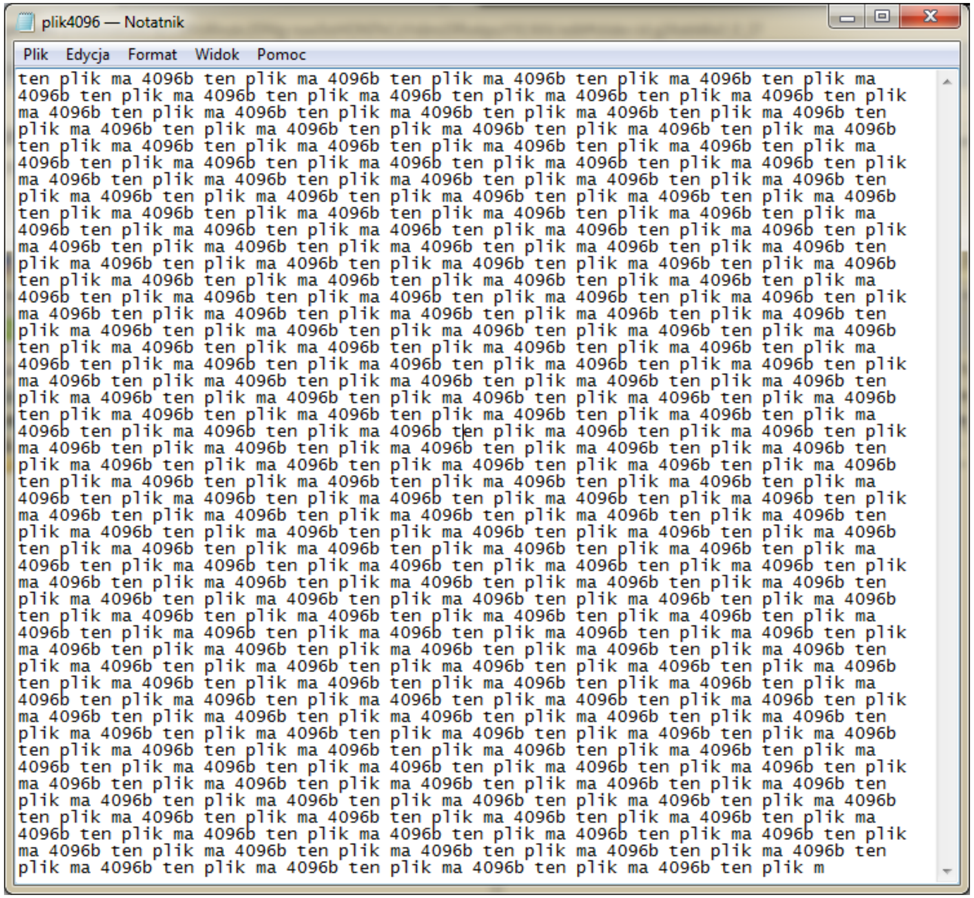
Intro 4kb (4096 bajtów)

krótko: taaaaaakie malutkie demka...

Ile to jest 4096 bajtów?

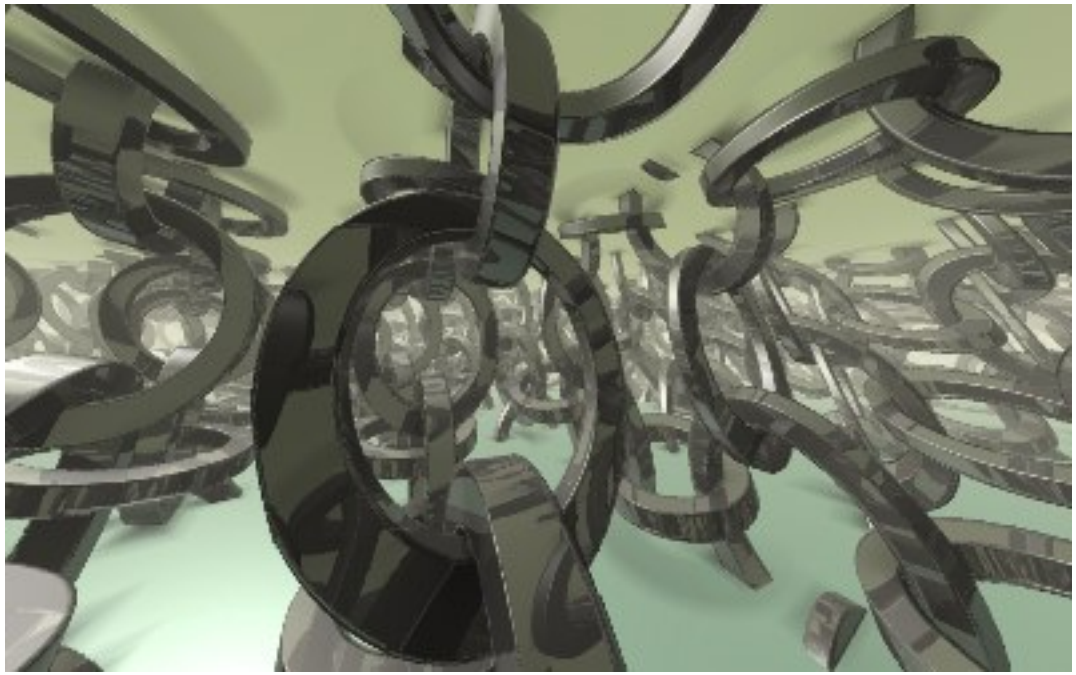
1 bajt = 8 bit

4096b = 32768 bit



Przykład: Sult / Loonies

śi



http://www.youtube.com/watch?v=foR64ZObAXA

3 miejsce
Breakpoint 2009

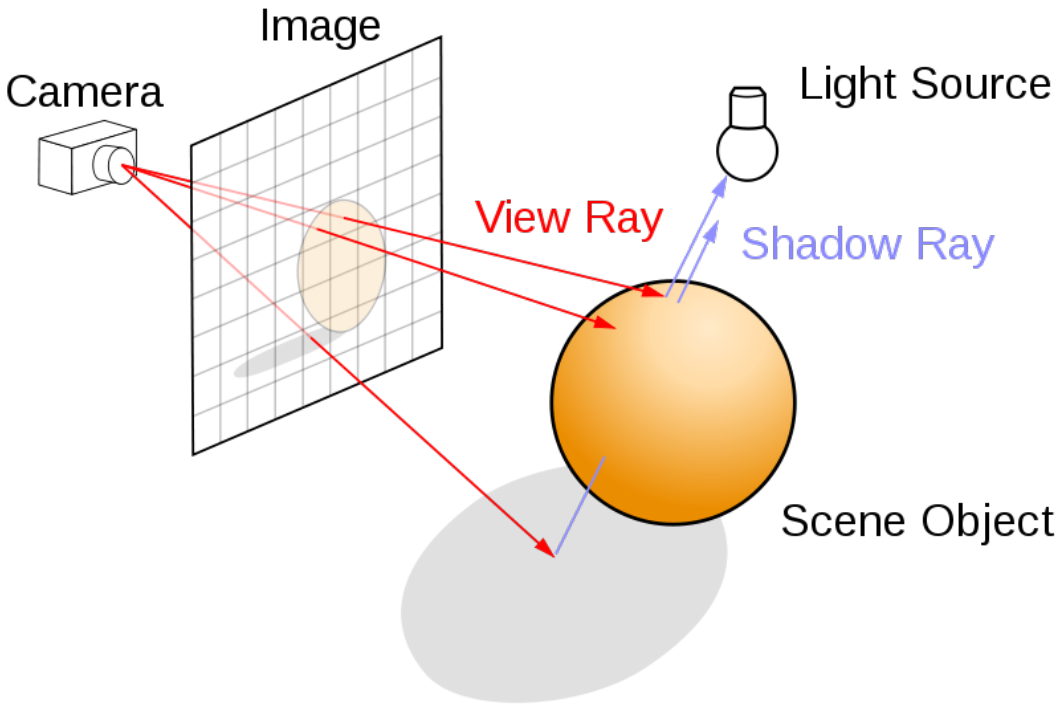
Strona techniczna

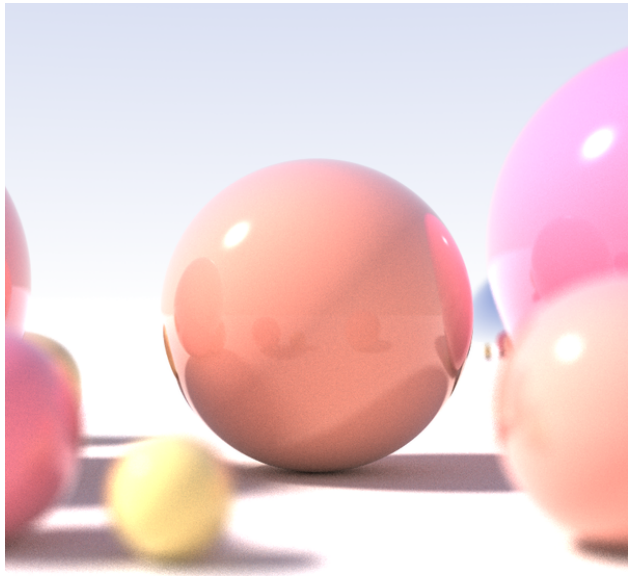
Sult w .avi sredniej jakosci > 80mb,

jak miesci sie to w 4kb?

- intro zapisane w programie cieniującym (fragment shader)
- **grafika proceduralna (ray marching)**
- muzyka proceduralna (synteza FM)
- sztuczki, optymalizacje i pakowanie (Crinkler)

Ray Tracing





(wikipedia)

Problemy: - problem z miękkimi cieniami
- dużo obiektów, fraktale 3D
- trzeba rozwiązywać równania

Ray Marching

(ważny slajd)

Zamiast liczyć punkt przecięcia promienia z obiektem w scenie (Ray Tracing)

wykonujemy spacer po promieniu szukając przecięcia z obiektami w scenie.

Równanie iteracyjne

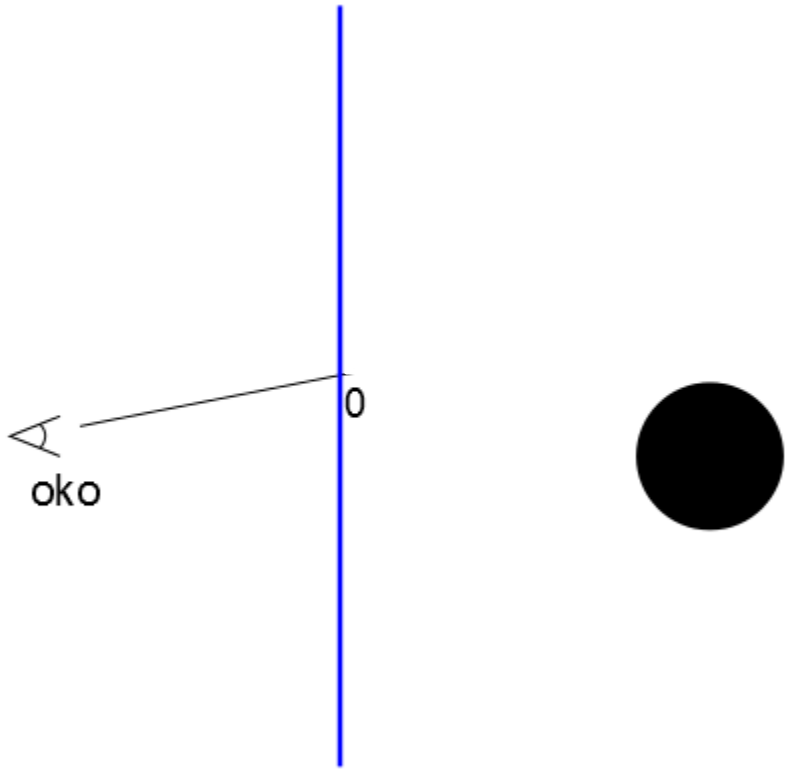
Promień:

$$\mathbf{r}(t) = \mathbf{r}_0 + t * \mathbf{dir}$$

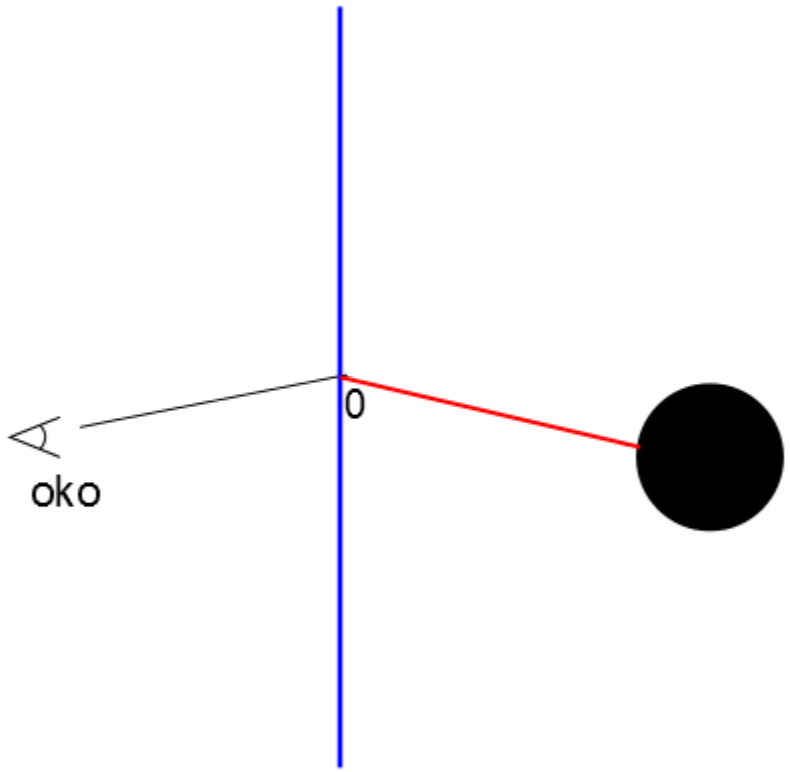
Szukamy t_i wskazującego przecięcie promienia z obiektem:

$$t_i = t_{i-1} + f(\mathbf{r}(t_{i-1}))$$

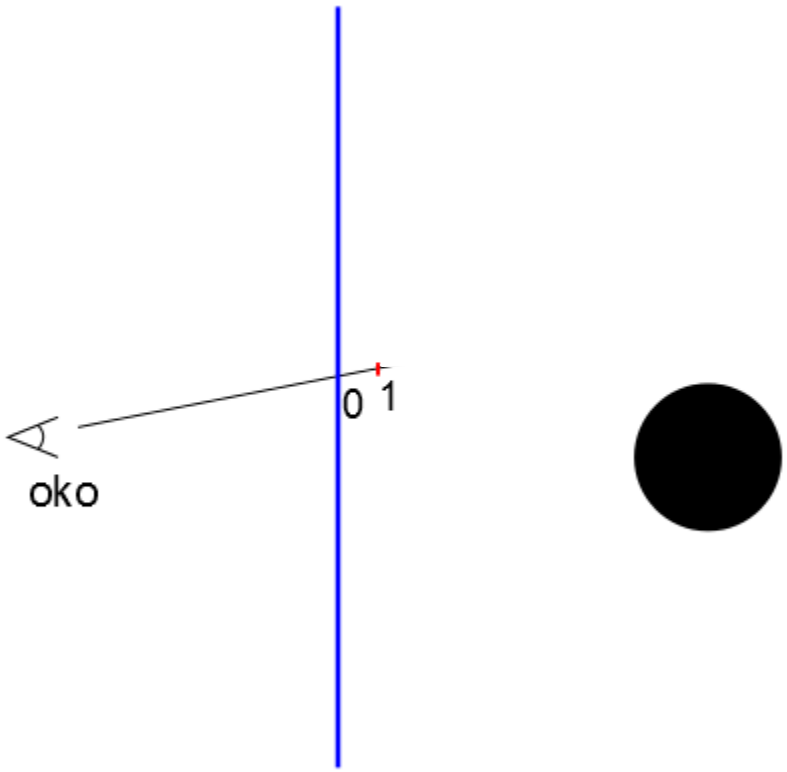
Ray Marching - Spacer



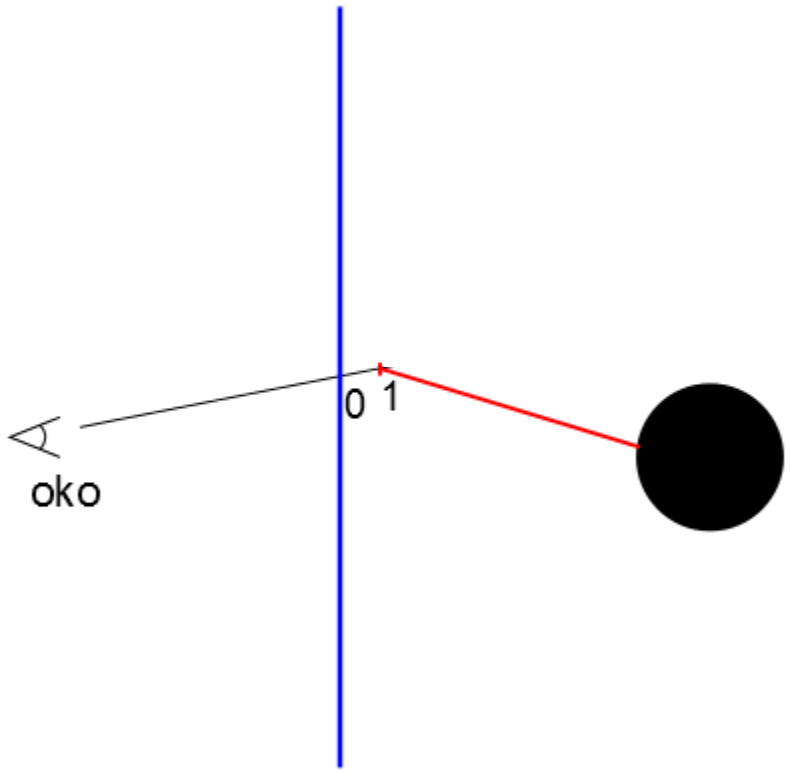
Ray Marching - Spacer



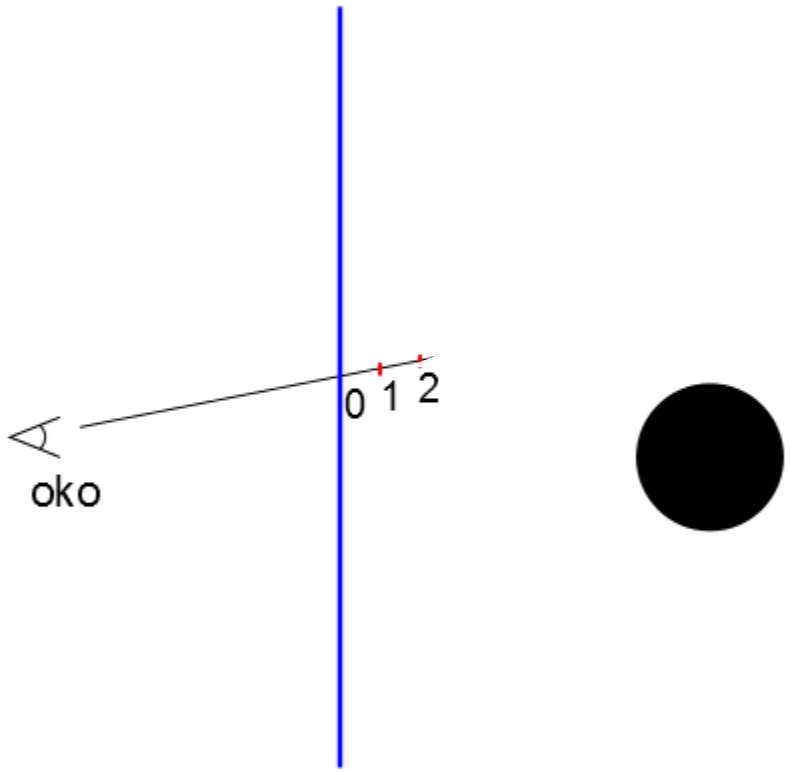
Ray Marching - Spacer



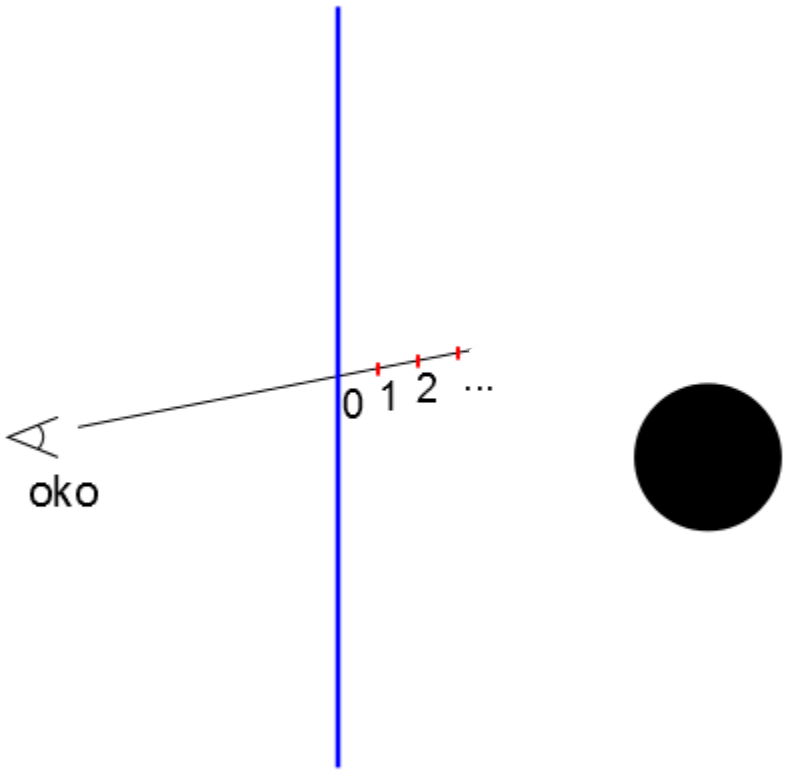
Ray Marching - Spacer



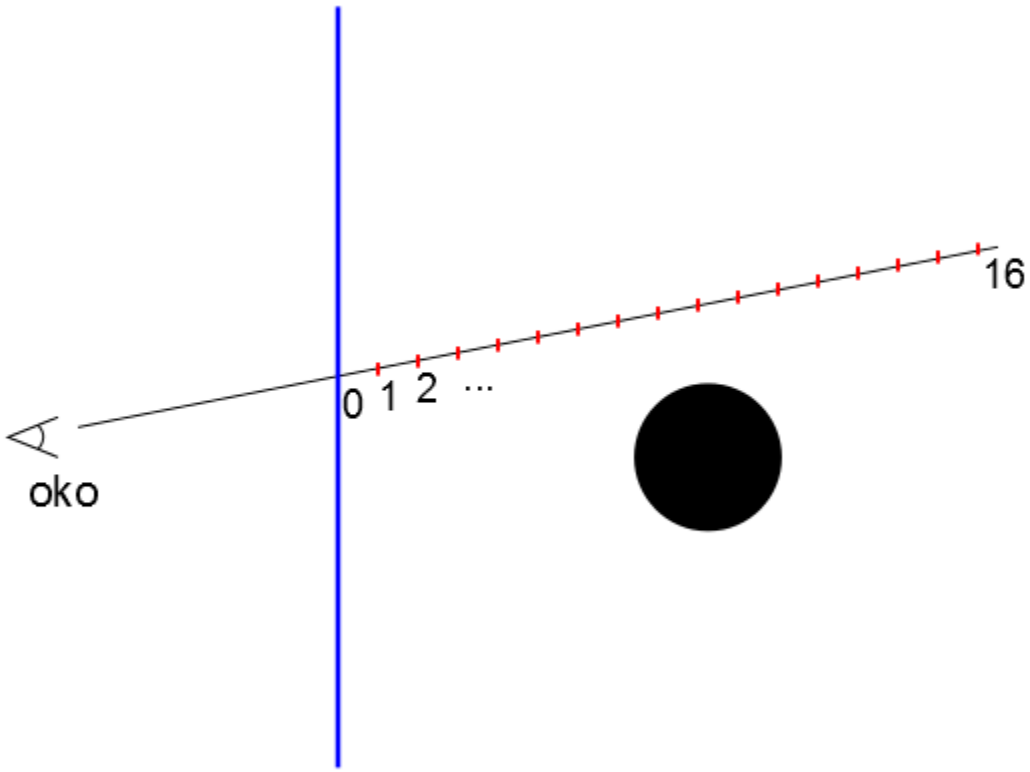
Ray Marching - Spacer



Ray Marching - Spacer



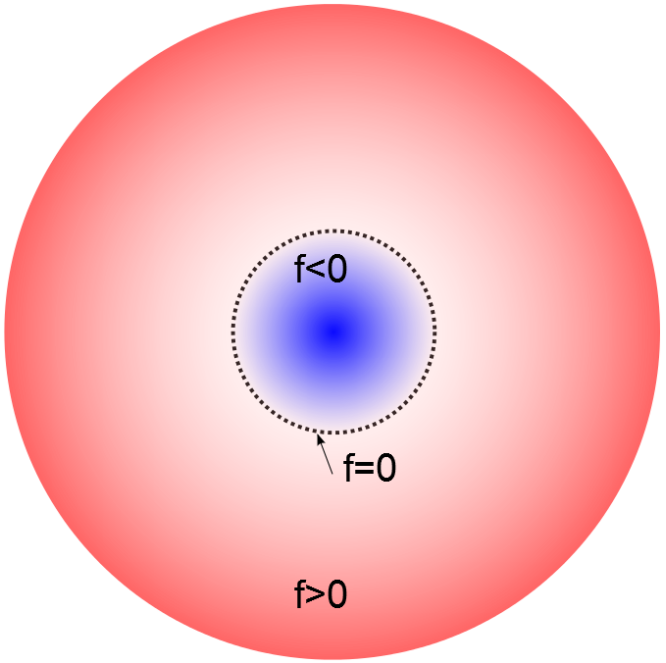
Ray Marching - Spacer



Distance Fields

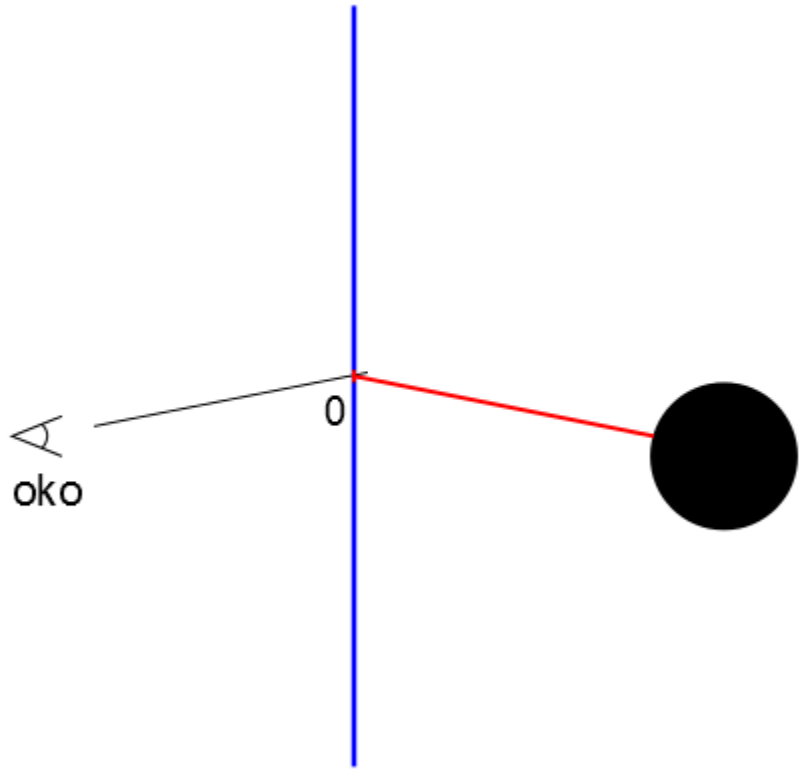
- Opis obiektów przez funkcje *implicit functions*
- dla danej pozycji **x** w scenie

f(**x**) < 0 - punkt wewnątrz
f(**x**) = 0 - punkt na powierzchni
f(**x**) > 0 - punkt na zewnątrz

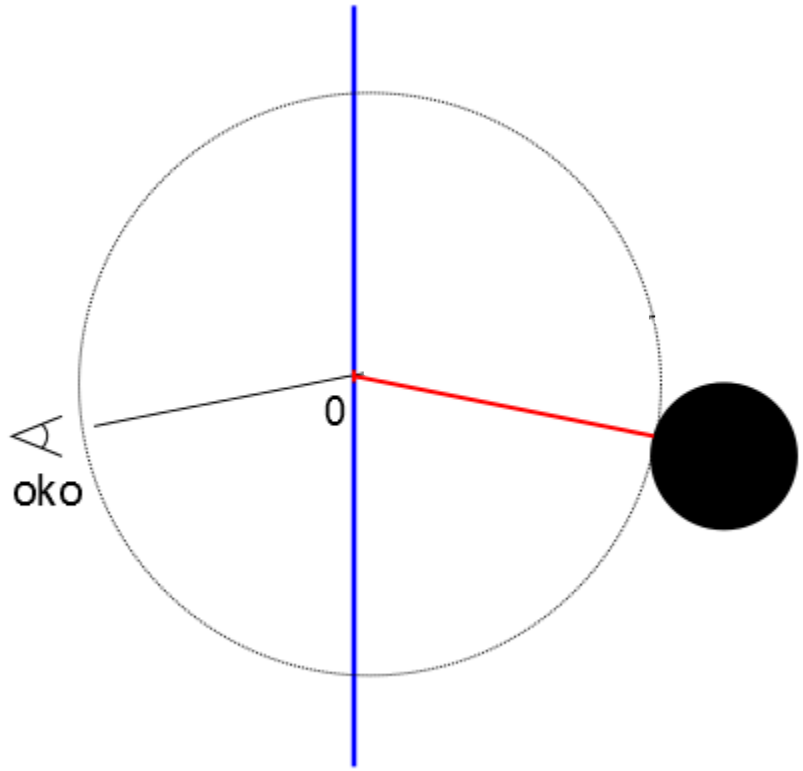


- wartosc f może niesc informację o odległosci

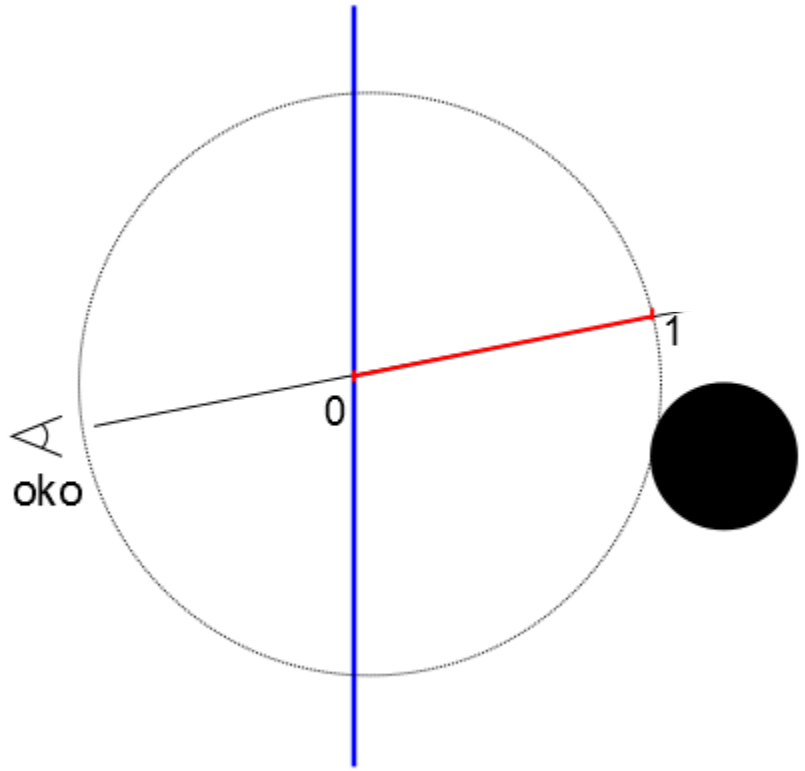
Sphere Tracing



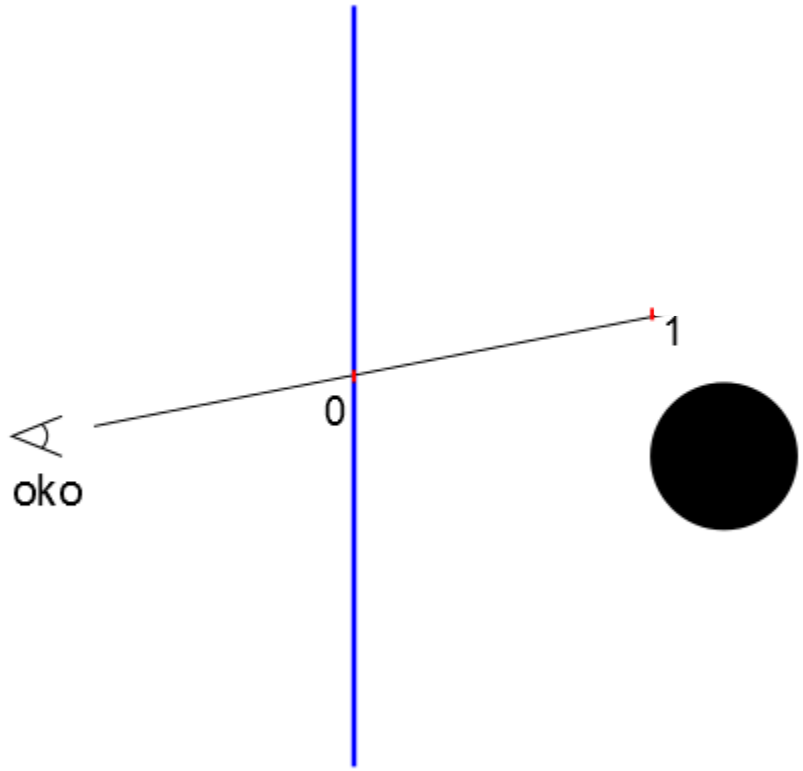
Sphere Tracing



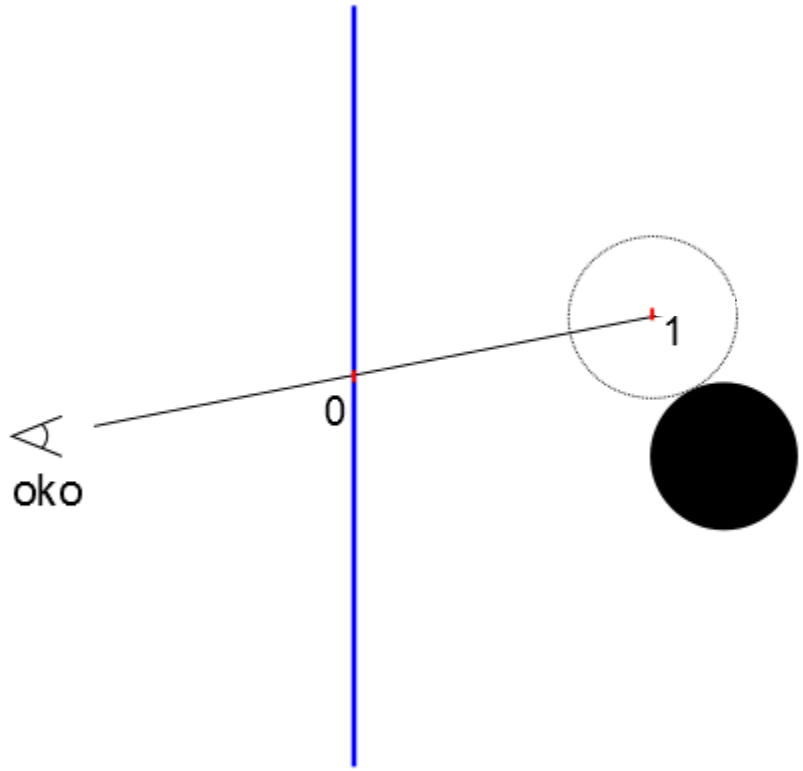
Sphere Tracing



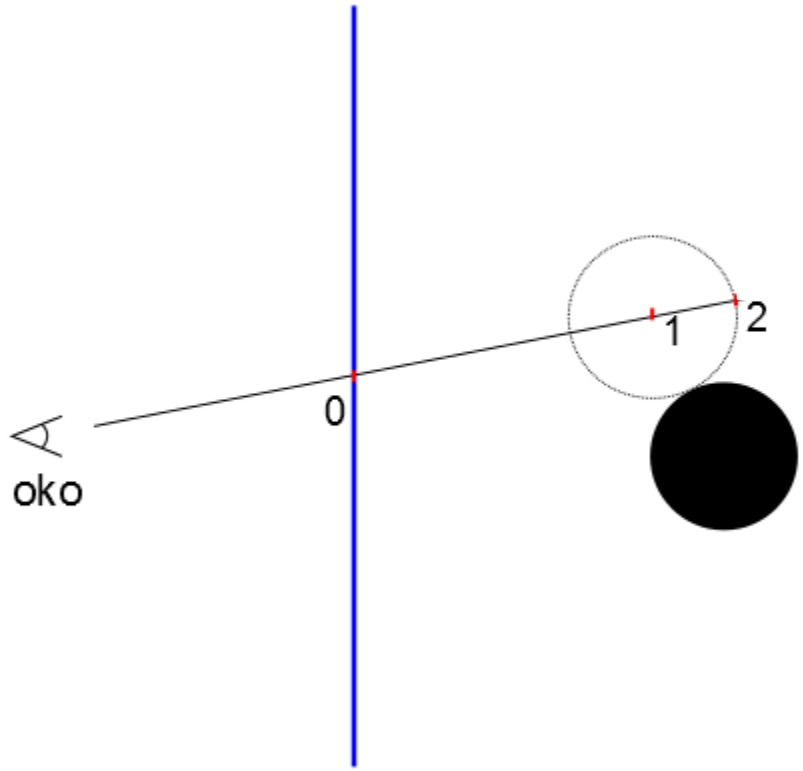
Sphere Tracing



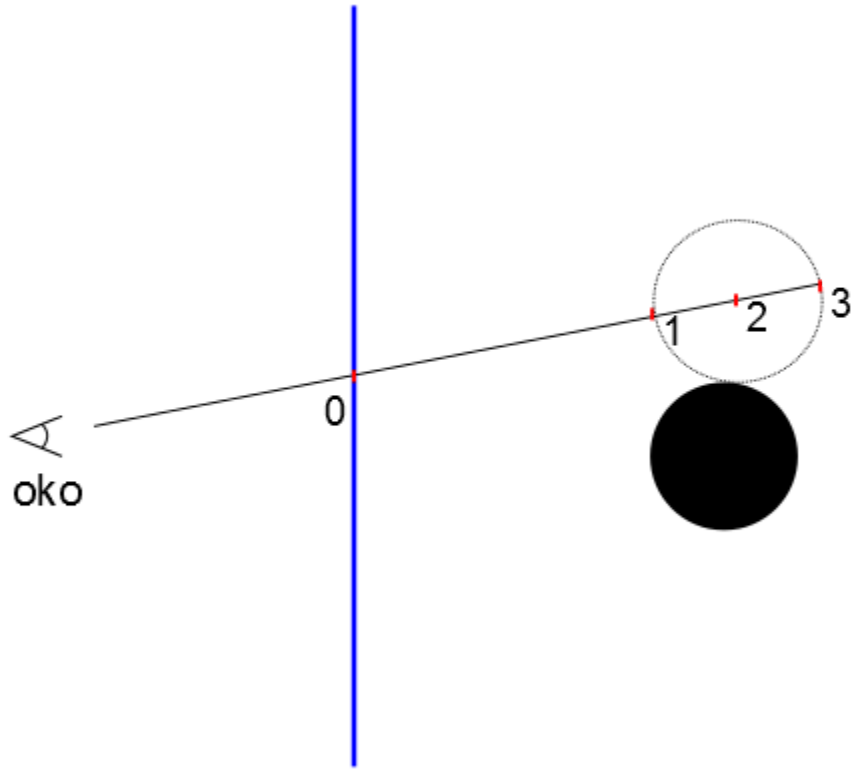
Sphere Tracing



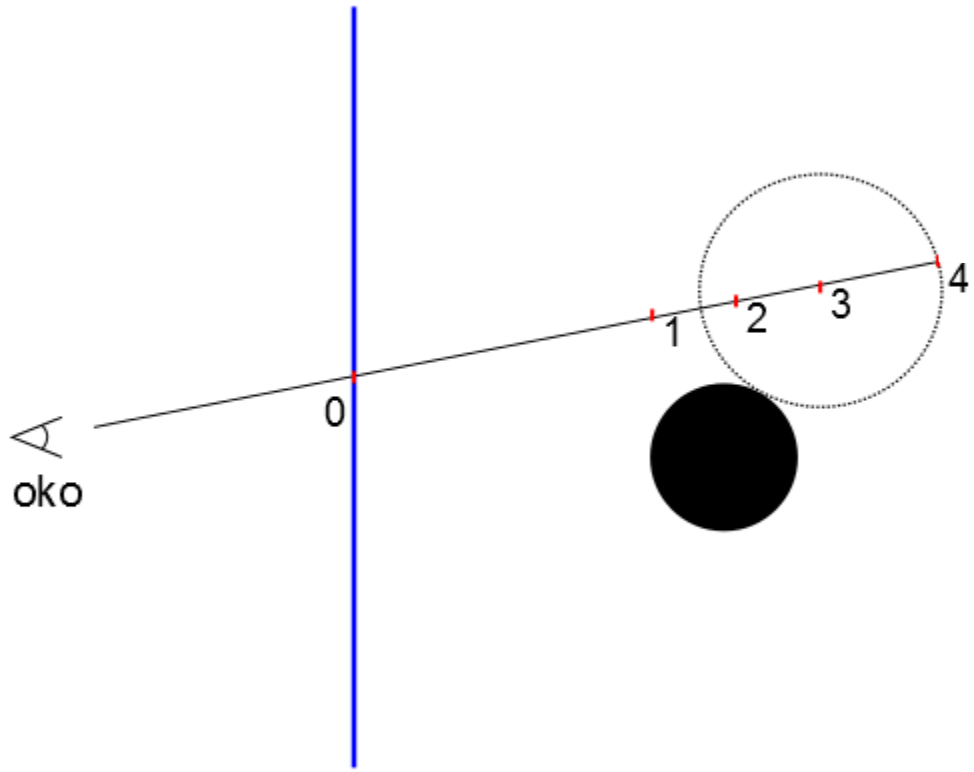
Sphere Tracing



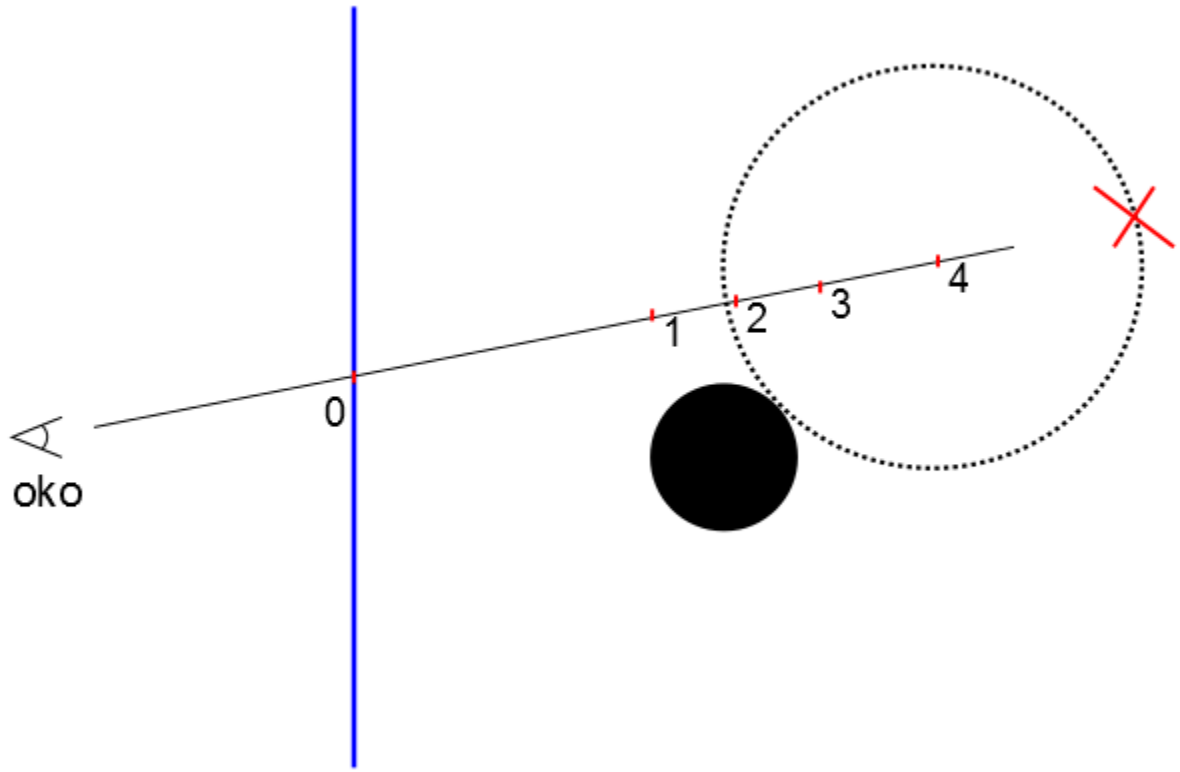
Sphere Tracing



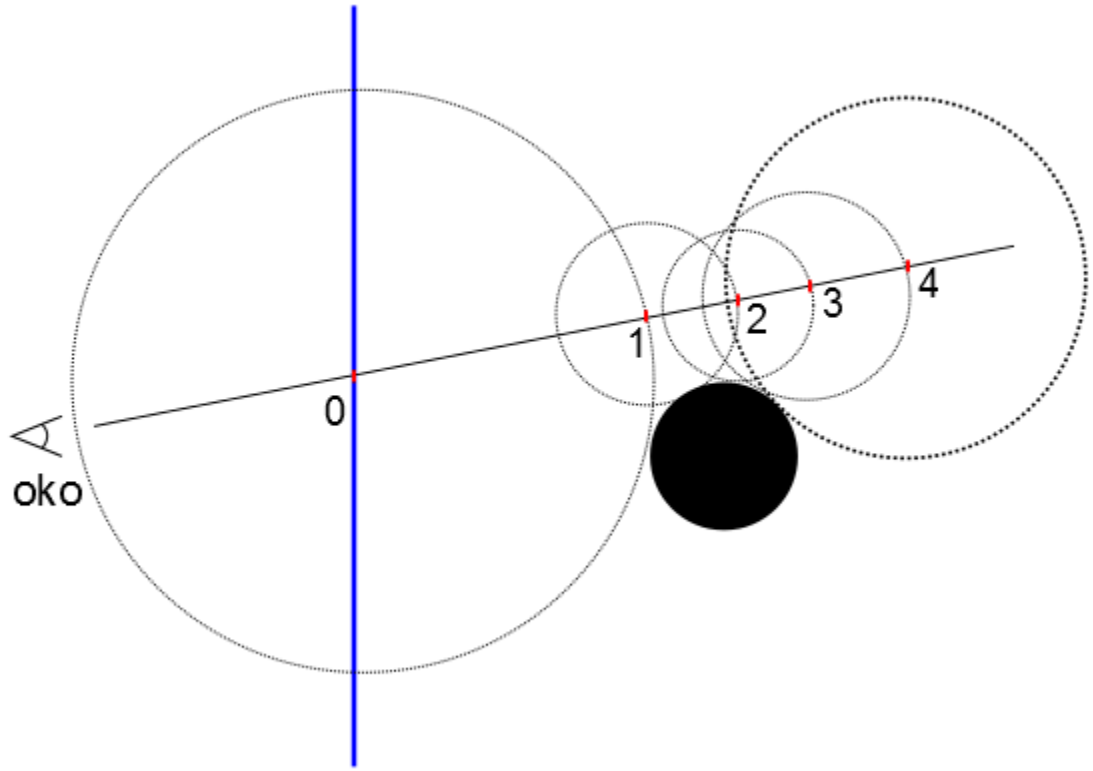
Sphere Tracing



Sphere Tracing



Sphere Tracing

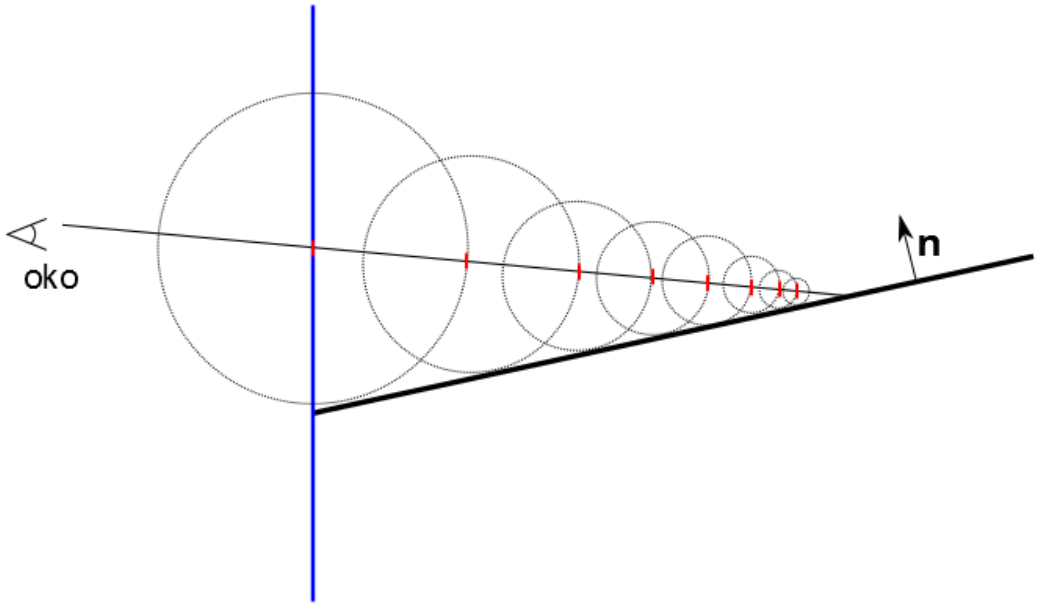


Algorytm sphere tracing

```
float d;
vec3 p = p0;
for(int i=0; i<ITER; i++)   // np. 100
{
 d = f(p);
 if(d < EPS)                // np. 0.002
    break;
 p = p + dir * d;
}
```

Przykładowe funkcje

Powierzchnia o normalnej **n** przechodząca przez punkt **n**d



$$f(\mathbf{r}) = \mathbf{r} * \mathbf{n} - d$$

J.C. Hart, *Sphere Tracing: Simple Robust Antialiased Rendering of Distance-Based Implicit Surfaces* (Course Notes, SIGGRAPH 1993)

Przykładowe funkcje

Powierzchnia o normalnej **n** przechodząca przez punkt **n**d

$$f(\mathbf{r}) = \mathbf{r} * \mathbf{n} - d$$

J.C. Hart, *Sphere Tracing: Simple Robust Antialiased Rendering of Distance-Based Implicit Surfaces* (Course Notes, SIGGRAPH 1993)

Przykładowe funkcje

Kula o promieniu R

$$f(\mathbf{r}) = |\mathbf{r}| - R$$

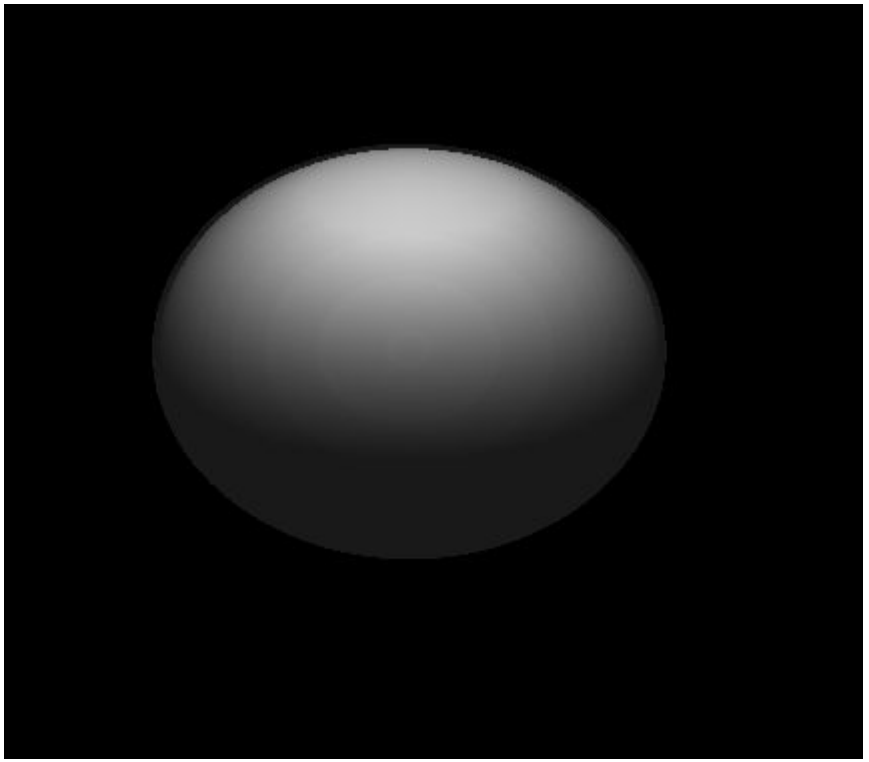
```
// przykładowy kod

float sphere(vec3 r, float R)
{
	return length(r) - R;
}
```

J.C. Hart, *Sphere Tracing: Simple Robust Antialiased Rendering of Distance-Based Implicit Surfaces* (Course Notes, SIGGRAPH 1993)

Przykładowe funkcje

Kula o promieniu R

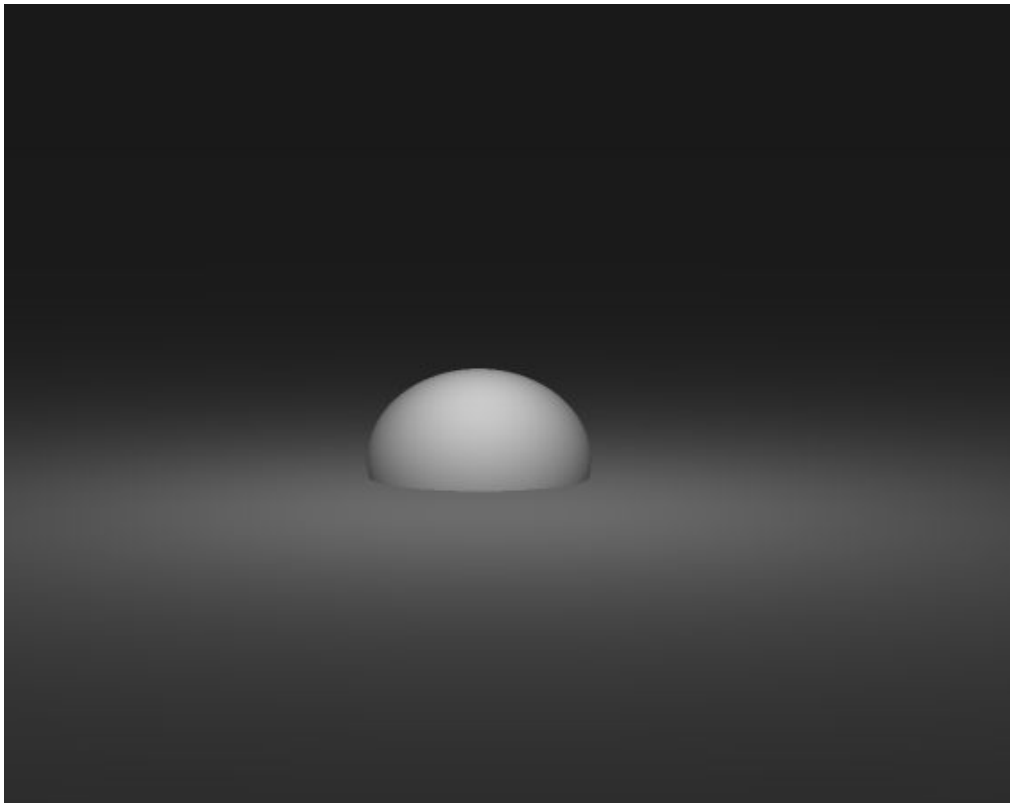
$$f(\mathbf{r}) = |\mathbf{r}| - R$$



J.C. Hart, *Sphere Tracing: Simple Robust Antialiased Rendering of Distance-Based Implicit Surfaces* (Course Notes, SIGGRAPH 1993)

Minimum dwóch funkcji

to ich suma...



Miękkie cienie

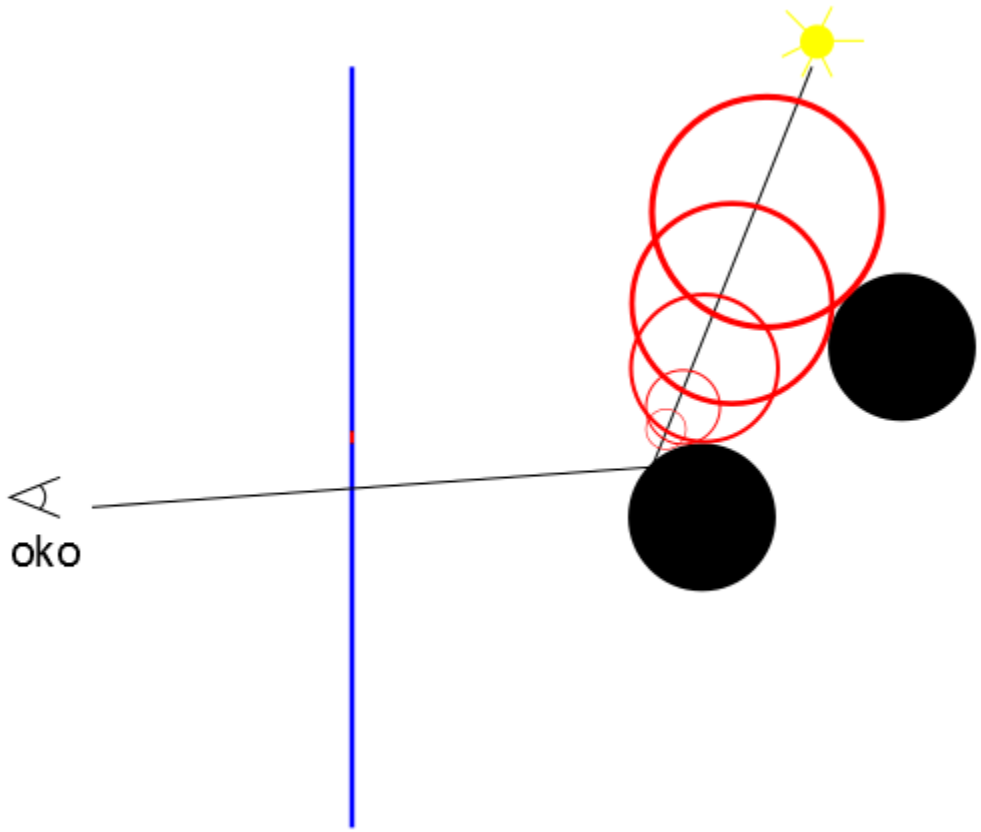
Marsz od punktu kontaktu do źródła swiatła

+

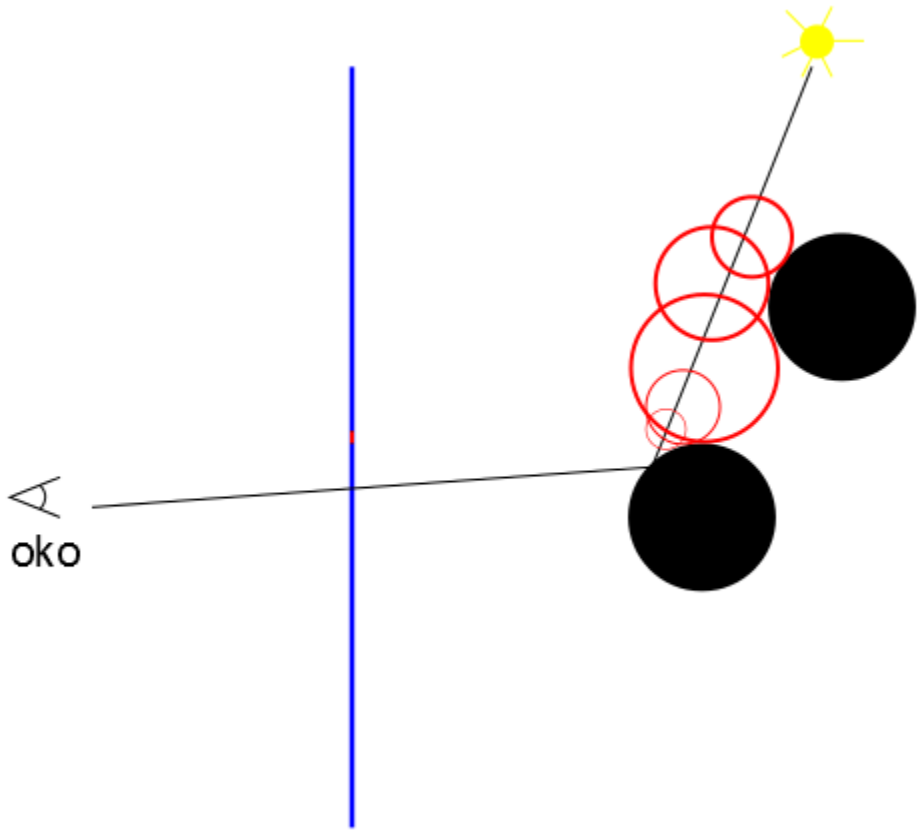
suma odległosci od innych obiektów

(b. proste w implementacji!)

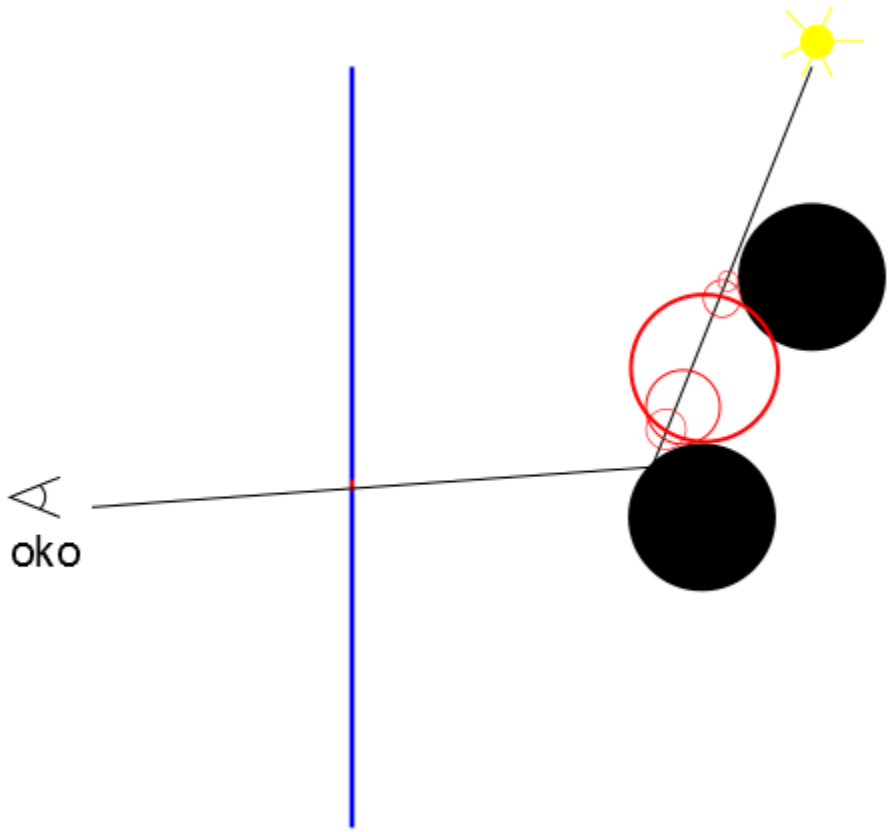
Miękkie cienie



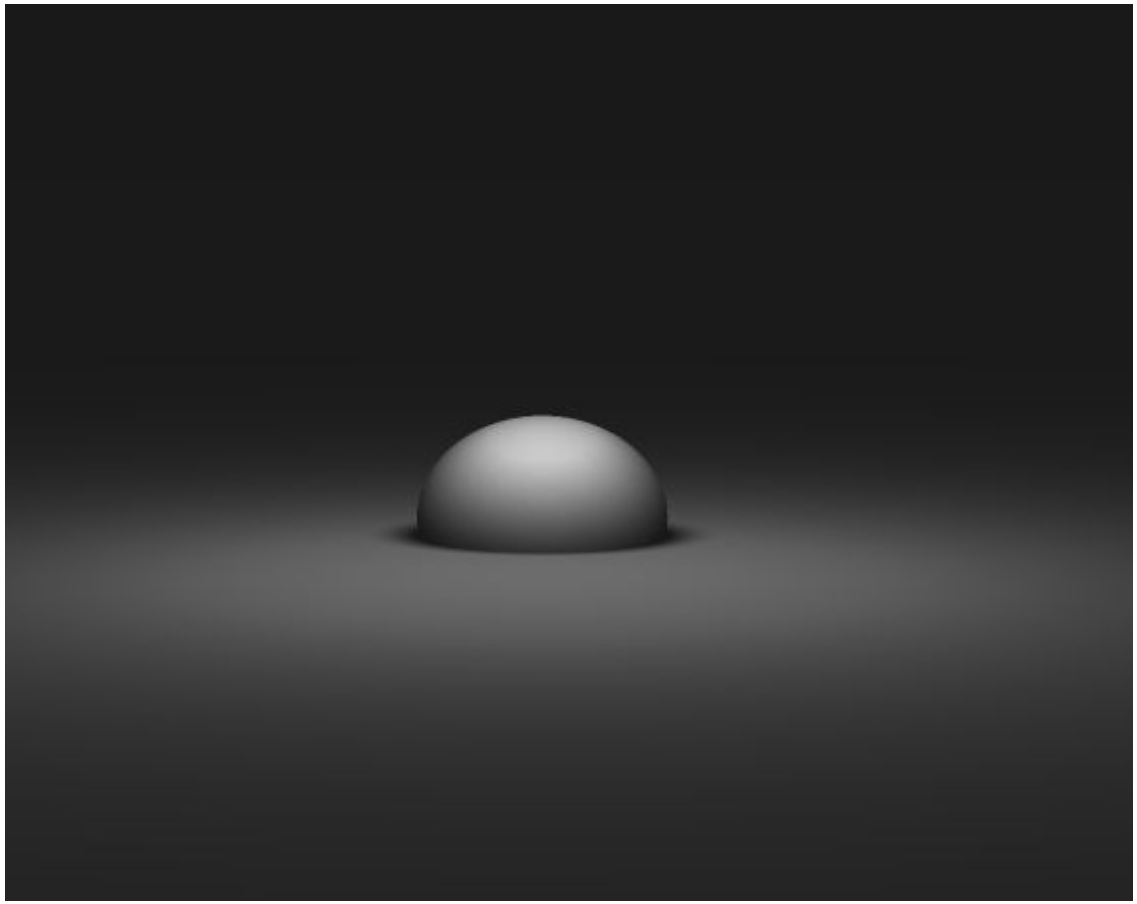
Miękkie cienie



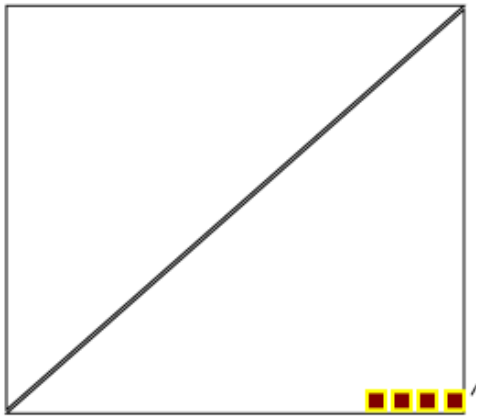
Miękkie cienie



Cienie

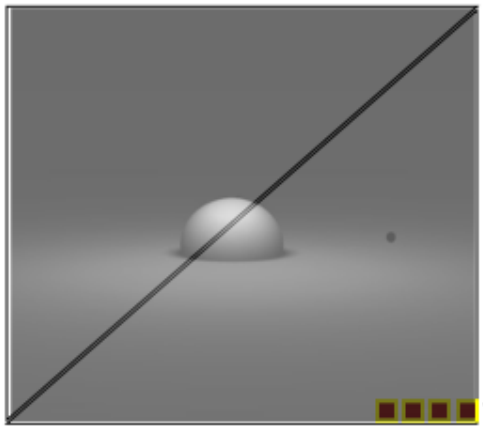


Realizacja w pixel shader



1) dwa trójkąty (fullscreen)

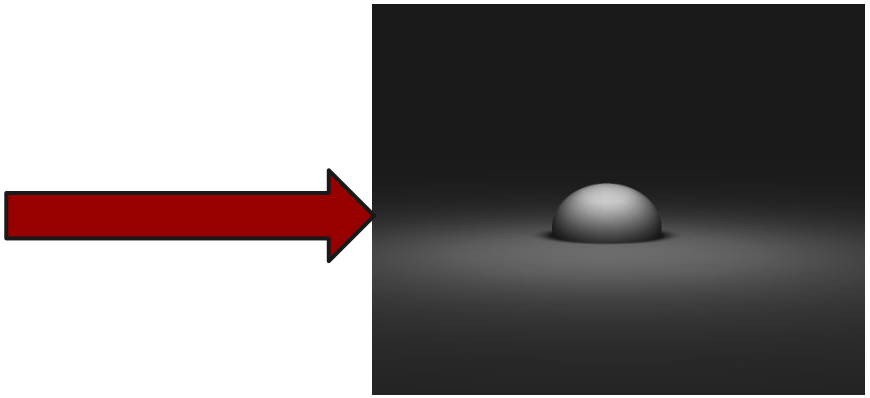
2) każdy pixel ma przypisany taki sam program cieniujący pixel shader



3) kod sphere tracingu można zapisać jako pixel shader który wykonuje się na GPU

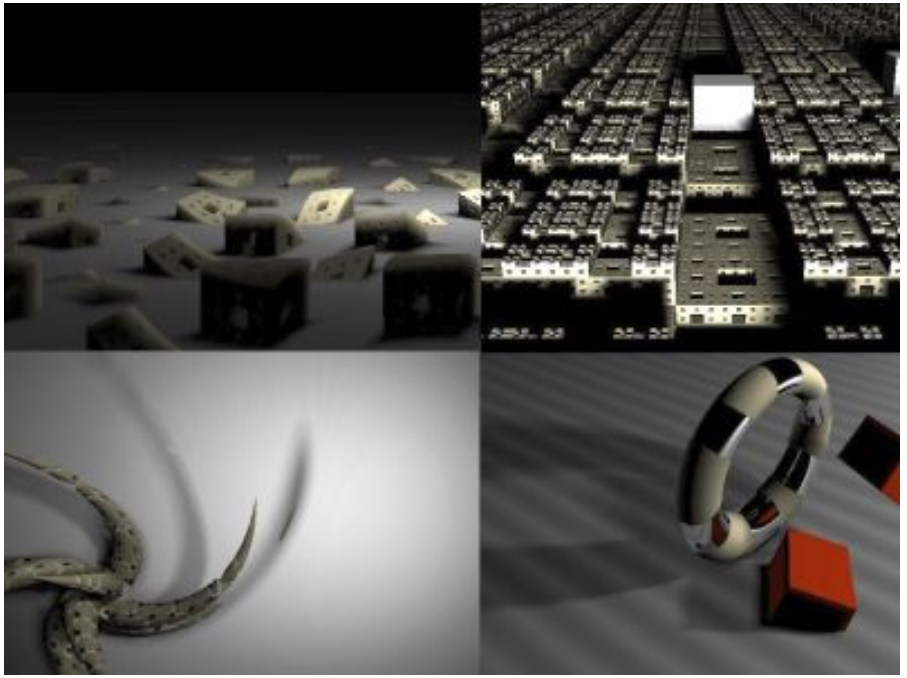
Przykładowy kod (ps)

```
varying vec3 r;
uniform float t;
uniform vec4 m;
float plane(vec3 r)
{   return r.y; }
float sphere(vec3 r, float R)
{ return length(r) - R; }

float f(vec3 r) // distance function
{
   float d=1e10;
   vec3 posr = r;
   d = sphere(r+vec3(0+m.x,-0.1+m.y,3.9), 0.5);
   d = min (d, plane(r));
   return d;
}
// credit: las / pouet thread
float shadow(vec3 p, vec3 l, float d, int i)
{
   float o;
   for (o = 0; i > 0.; i--)    o += f(p+l*i*d).x;
      return clamp(o, 0.0, 1.0);
}
// credit: http://www.pouet.net/topic.php?which=7920&page=10, rar
vec3 normal(vec3 p)
{
   #define DR 2e-4
   vec3 dr = vec3(DR,0,0);
   return ( vec3( f(p+dr.xyz).x, f(p+dr.yxz).x, f(p+dr.yzx).x ) - f(p).x) / DR;
}
#define ITER 90
#define EPS 0.002
void main()
{
   vec3 camera = vec3(-0.1,1.8,-15.0);
   vec3 p = vec3(r.x, r.y+1.2, -10.2);
   vec3 lightv = vec3(0.0,2.5,-5.0);

   vec3 dir = normalize(p-camera);
   int mat=1;
   float d;
   for(int i=0; i<ITER; i++)
   {
            d = f(p);
            if(d < EPS)
                  break;
            p = p + dir * d;
   }
   vec3 n = normal(p);
   float light = 1.0 + 0.5*(dot(n,lightv));
   light = light / pow(length(lightv-p),2);
   vec3 lightdir = normalize(lightv-p);
   light *= shadow(p, lightdir, 0.02, 10);
   vec4 color = 0.1+0.9*light * vec4(1,1,1,1);
   gl_FragColor = color;
}
```



W działaniu

AKIKO by Floppy
2 miejsce na Riverwash 2011

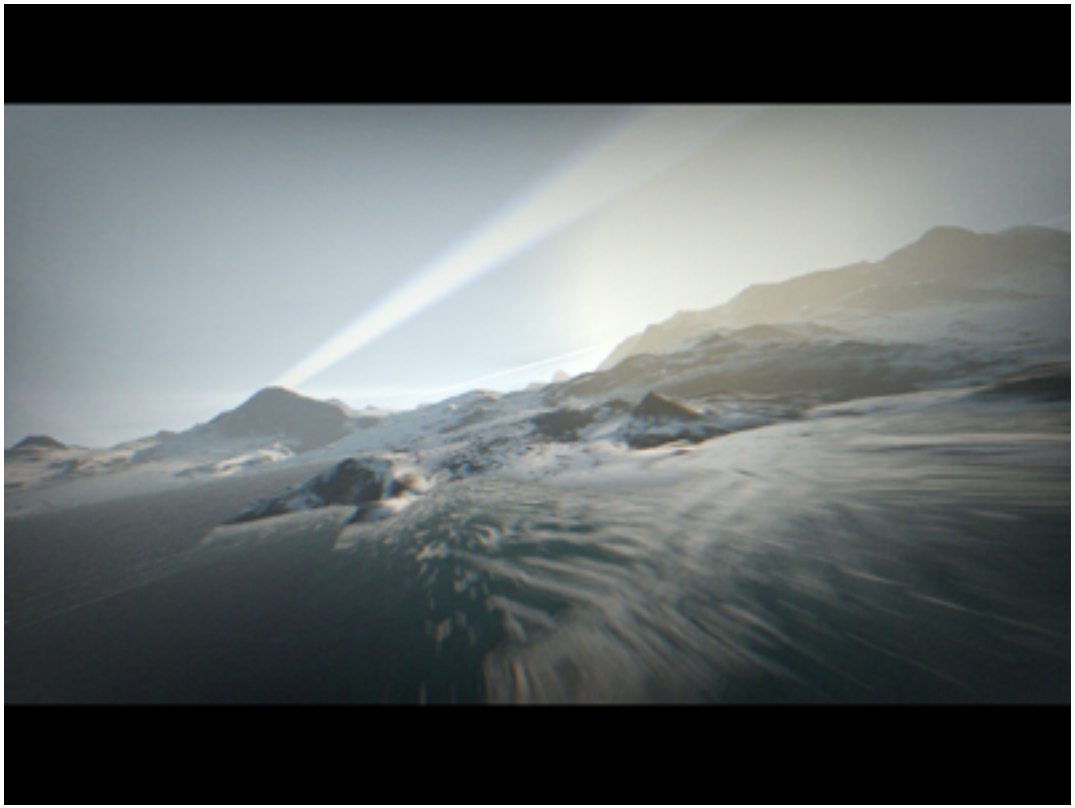


http://www.youtube.com/watch?v=mxZx5RlQsc4

Czy tylko prymitywy w 4k?

- fraktale 3d
- tereny 3d, np.
  - renderuj klasyczny objekt (np. teren)
  - wynik przenies na GPU
  - ray marching
  - efekt?
    (Rendering World with two triangles / IQ

Elevated 4kb



1 miejsce
Breakpoint 2009

http://www.youtube.com/watch?v=_YWMGuh15nE

Dobre linki

http://pouet.net/ - m.in. intra 4k

http://www.iquilezles.org/ - m.in. sphere tracing

http://www.iquilezles.org/apps/shadertoy/

Koniec

Dziękuję za uwagę

http://panoramix.ift.uni.wroc.pl/~maq/raymarching.zip

helpdesk: maciej.matyka[at] gmail.com