Raytracing

Ann-Kathrin Kapfenstein, Daniel Koch, Anja Schwenk

18. Mai 2017

Was ist Raytracing?

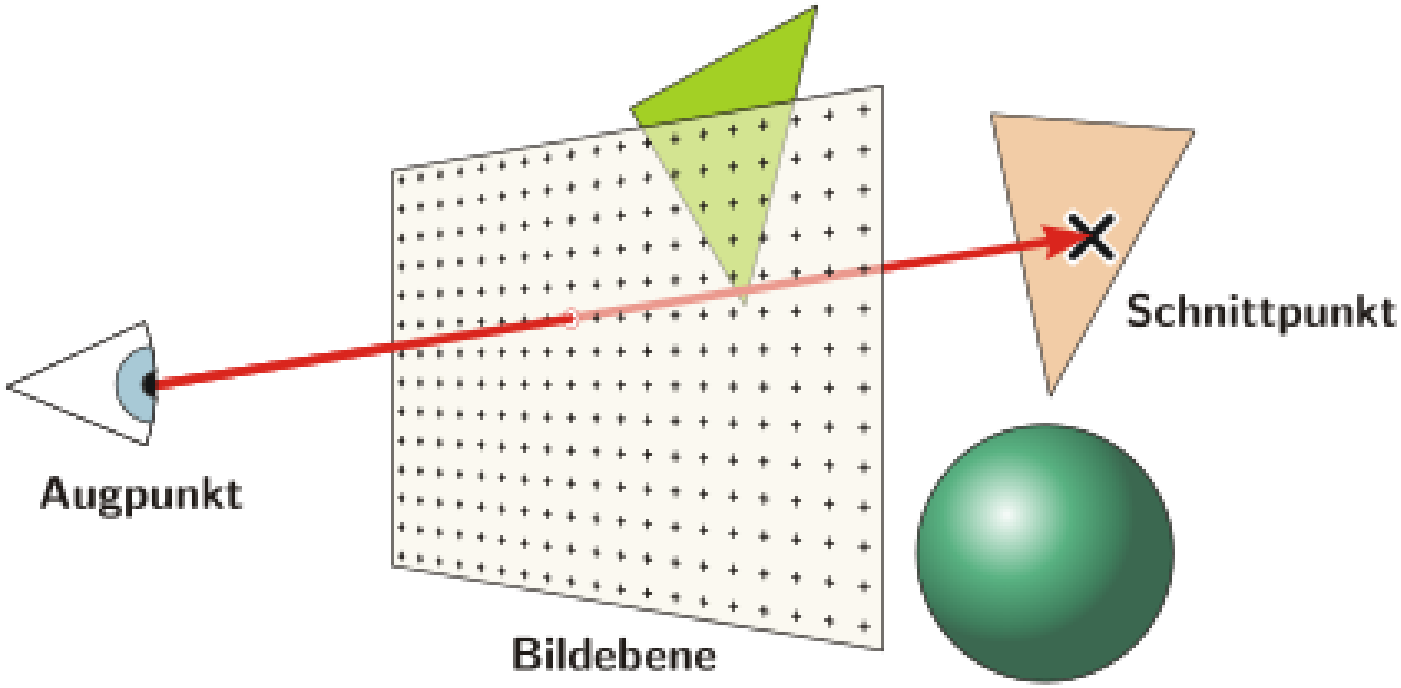


Abbildung: https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Raytracing.svg/440px-Raytracing.svg.png

Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

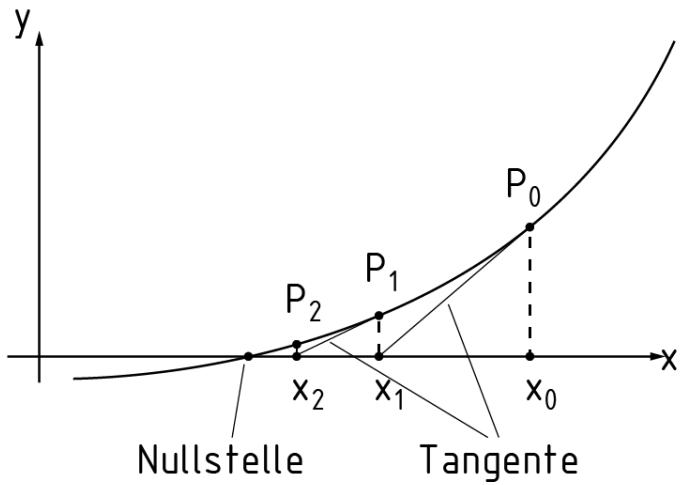
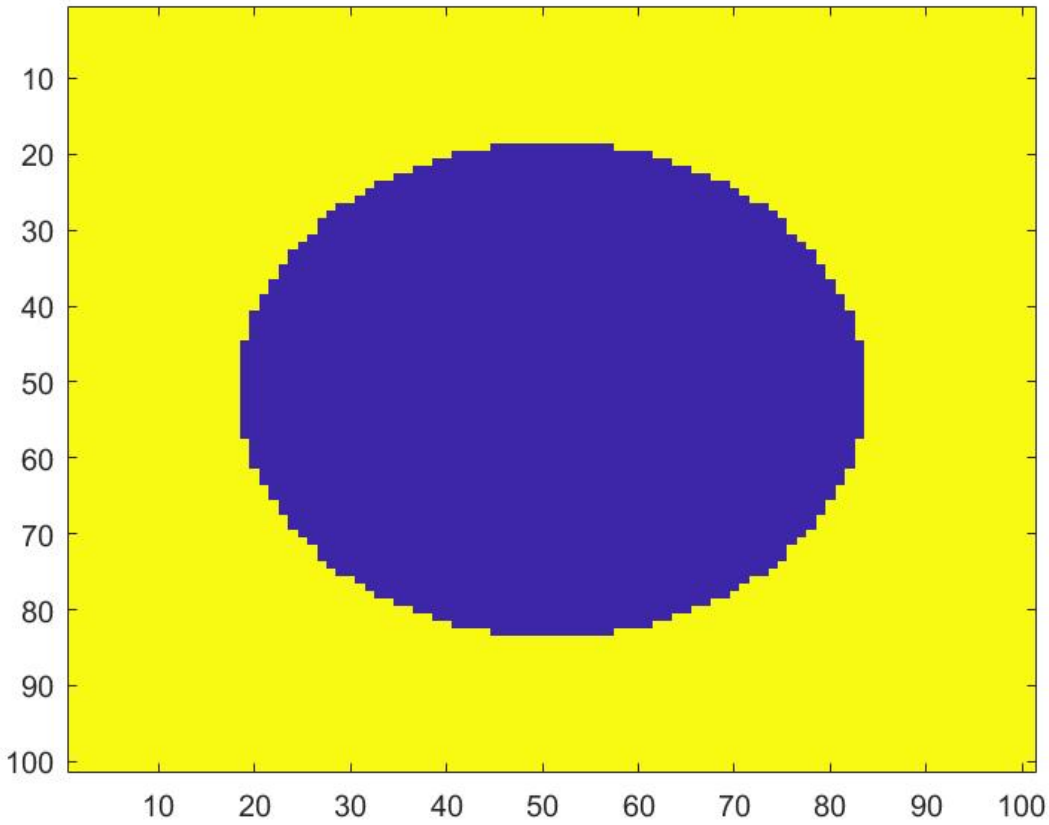


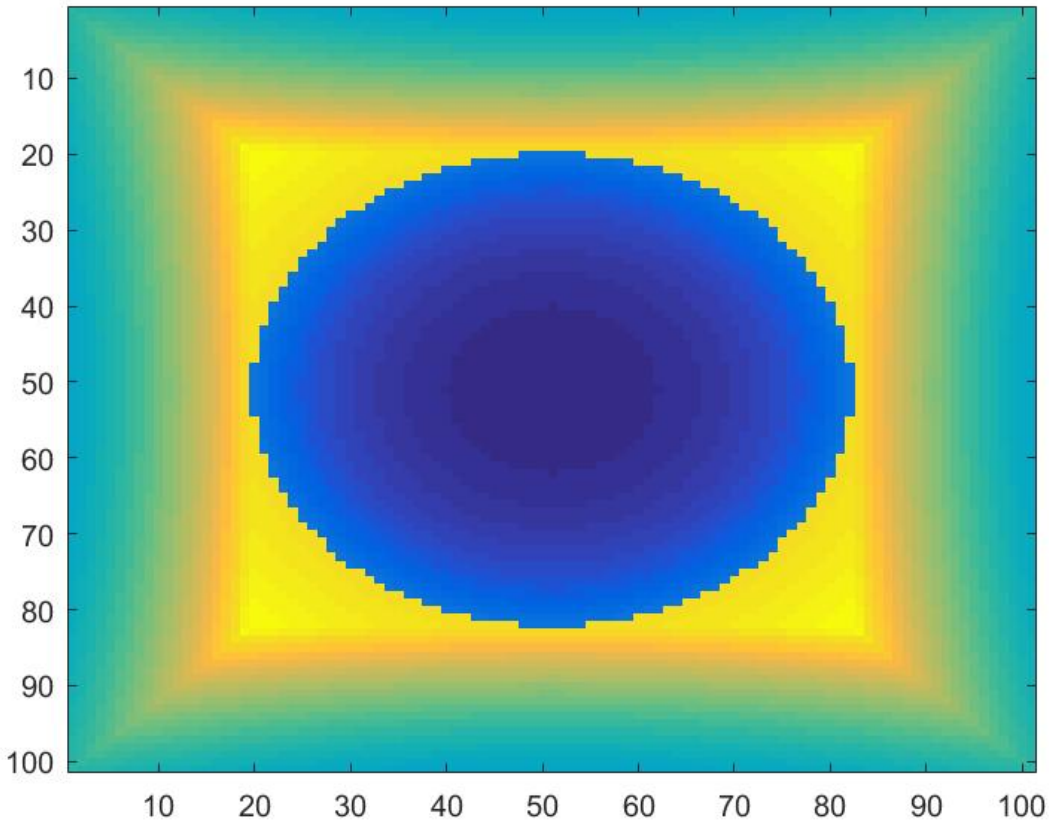
Abbildung: http://glossar.item24.com/fileadmin/sync/data/images/A0003_ DE.png

```
1  function [bool,N] = Newton(grid, eye, rays, f, iter)
2  % grid = Gitter
3  % eye = Auge
4  % rays = Strahlen
5  % f Koerperfunktion von R3 nach R
6  % iter = Anzahl der Iterationen des Newtonverfahrens
7  % bool = Wahrheitswert ob NS existiert, falls ja, dann 1, sonst 0
8  % N = Nullstellenmatrix
9
10 % x0 Werte berechnen
11 N= zeros(size(grid,1),size(grid,2));
12 % schrittweite
13 h=0.000001;
14 % Maximaler Abstand zwischen vorletzter und letzter Ieration, dass letzte
15 % Iteration als Nullstelle anerkannt wird
16 eps = 0.01;
17
18 %Newtonverfahren mit numerischer Ableitung
19 for k=1:iter
20     M=N;
21     N = N-comp(f,N,rays,eye,h);
22 end
23
24 % befuellen der Bool-Matrix
25 bool = abs(M-N);
26 bool(bool<eps) = 0;
27 bool(bool>=eps) = 1;
28 bool = ones(size(grid,1),size(grid,2)) - bool;
29 bool(N<0) = 0;
30 end
```

Erstes Bild



Zweites Bild



Diffuse Beleuchtung

$$I_{diff} = \begin{cases} 0 & , falls\ n(x) \cdot r_{licht} \geq 0 \\ I_{dir} \cdot \rho(x) \cdot \frac{|n(x) \cdot r_{licht}|}{|n(x)| \cdot |r_{licht}|} & , sonst \end{cases}$$

```
function [Normal]= normalvector(f,N,eye,rays)
% f = Koerperfunktion
% N = Nullstellenmatrix
% eye = Auge
% rays = Strahlen
% Normal = Matrix der Normalenvektoren

% Vorbelegung
Normal = zeros(size(rays));
% Schrittweite
h=0.0000001;

% Schleife zur Berechnung der Oberflaechenpunke
for i = 1:3
    rays(:,:,i) = eye(i) + N(:,:).*rays(:,:,i);
end

% Berechnung der Normalenvektoren mit numerischer Ableitung
Normal(:,:,1)=(f(rays(:,:,1)+h,rays(:,:,2),rays(:,:,3))-f(rays(:,:,1)-h,rays(:,:,2),rays(:,:,3)))./(2*h);
Normal(:,:,2)=(f(rays(:,:,1),rays(:,:,2)+h,rays(:,:,3))-f(rays(:,:,1),rays(:,:,2)-h,rays(:,:,3)))./(2*h);
Normal(:,:,3)=(f(rays(:,:,1),rays(:,:,2),rays(:,:,3)+h)-f(rays(:,:,1),rays(:,:,2),rays(:,:,3)-h))./(2*h);
end
```

Glanzpunkte

$$H(v) = I - 2\frac{vv^T}{||v||^2}$$

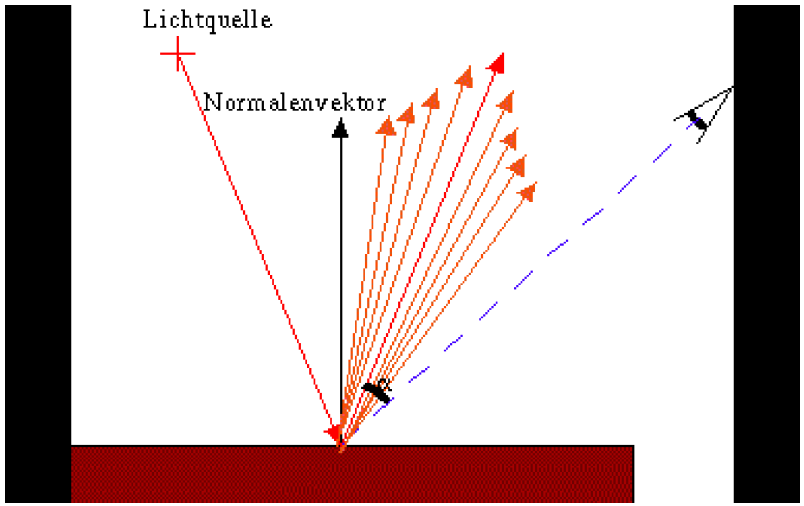


Abbildung: http://www.raytracer.de/raytracing/spiegelnd.gif

Winkelberechnung:

$$\cos(\alpha) = \frac{\langle \vec{u}, \vec{v} \rangle}{|\vec{u}| \cdot |\vec{v}|}$$

$$I_{spec} = f(\cos(\alpha)) = f\left(\frac{\langle \vec{u}, \vec{v} \rangle}{|\vec{u}| \cdot |\vec{v}|}\right)$$

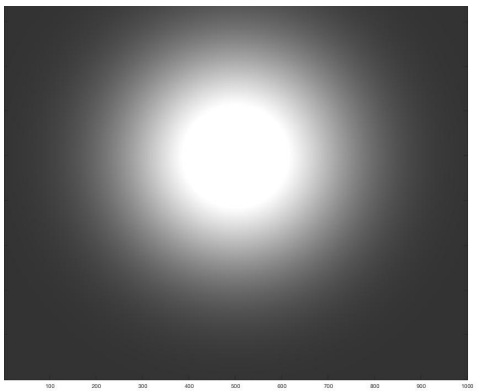


Abbildung: $\cos(x)^3$

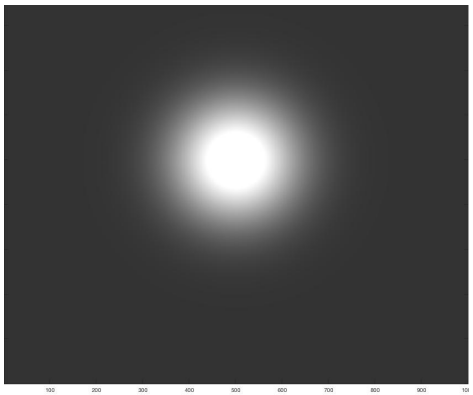
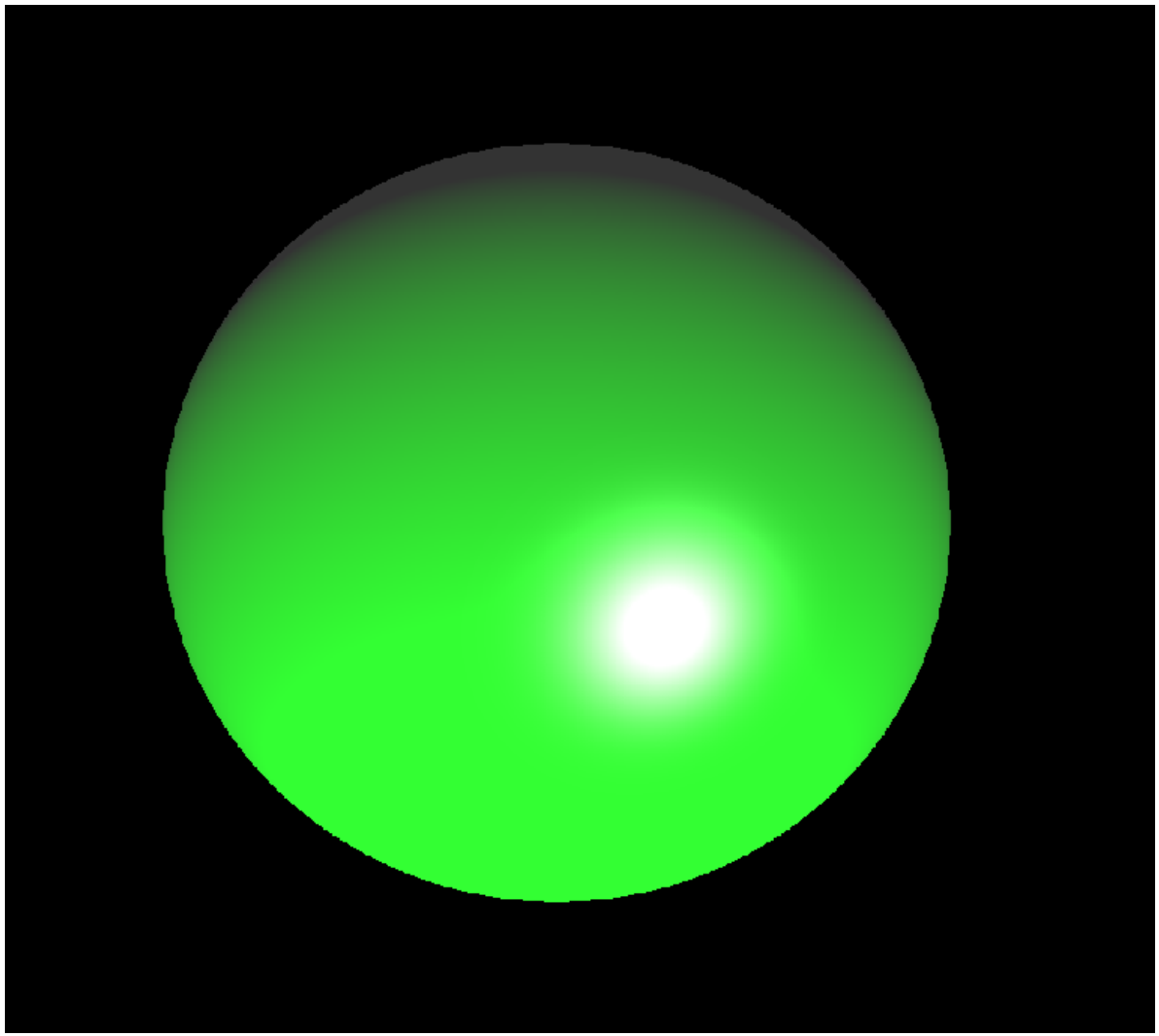


Abbildung: $\cos(x)^{10}$

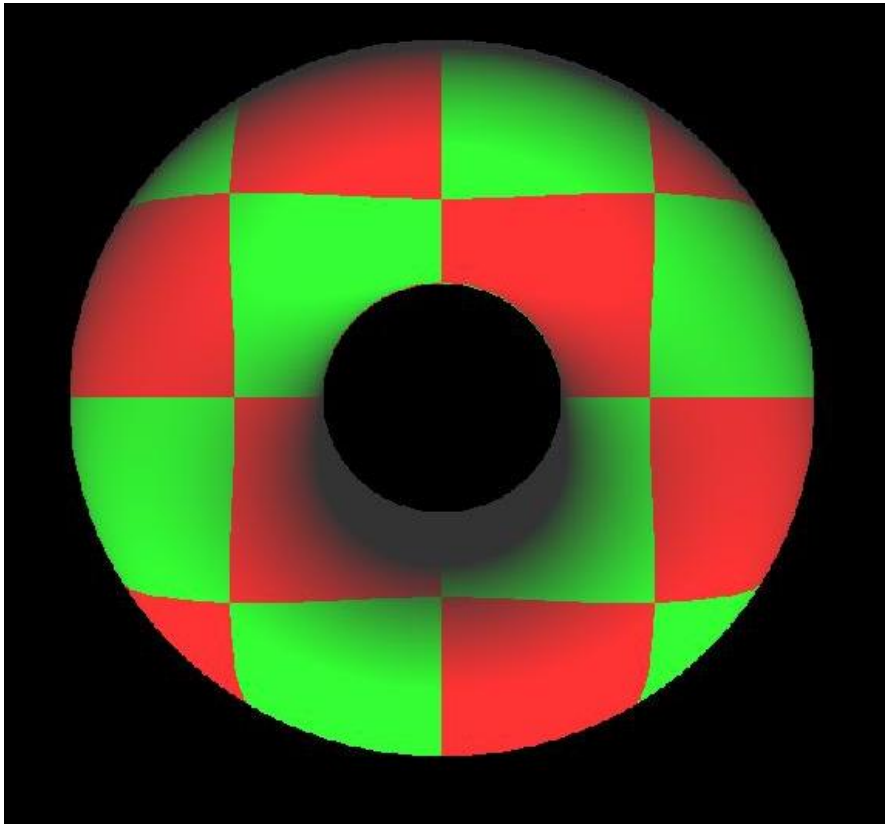
Ambiente Beleuchtung

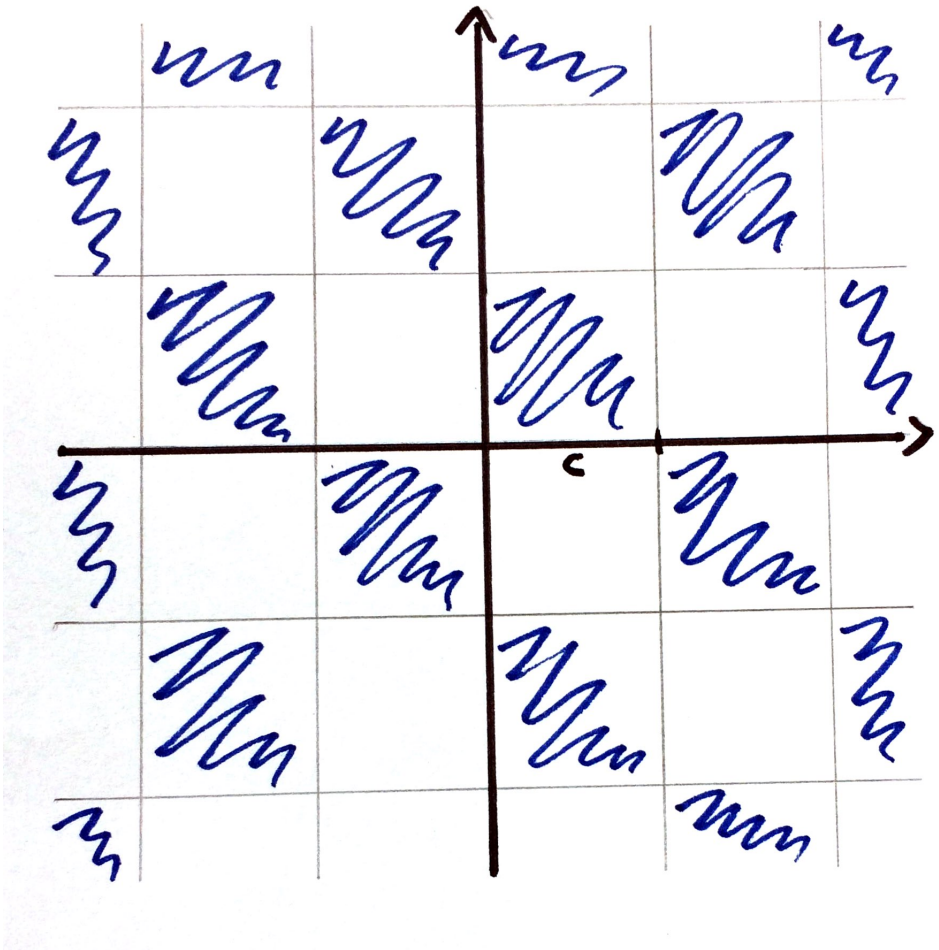
```
%befuellen der Matirx mit Beleuchtungswerten

    for i =1:size(rays,1)
        for j=1:size(rays,2)
            %prueft ob Schnittpunkt positiv ist
            if bool(i,j)>0
                a = dot(squeeze(Normal(i,j,:)),rlight);
                if a< 0
                    %diffuse, d.h. blickwinkelunabhaengige Helligkeit
                    I_diff(i,j,1) = dir .* rho_color(1) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                    I_diff(i,j,2) = dir .* rho_color(2) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                    I_diff(i,j,3) = dir .* rho_color(3) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                end
            end
            %ambiente, d.h. richtungsunabhaengige Helligkeit
            if bool(i,j)==1
                I_amb(i,j,1)= amb(1);
                I_amb(i,j,2)= amb(2);
                I_amb(i,j,3)= amb(3);
            end
        end
    end

if spec == 1
    mirror = zeros(size(rays));
    for i = 1:size(rays,1)
        for j = 1:size(rays,2)
            if bool(i,j)>0
                H = eye(3) - 2* (squeeze(Normal(i,j,:)) * squeeze(Normal(i,j,:))')./(norm(squeeze(Normal(i,j,:)))^2);
                mirror(i,j,:) = H*(squeeze(Surface(i,j,:)) - lamp);
                a = dot(-squeeze(rays(i,j,:)),squeeze(mirror(i,j,:)));
                if a > 0
                    I_spec(i,j,:) = [1;1;1] .* (a./(norm(squeeze(rays(i,j,:)))*norm(squeeze(mirror(i,j,:))))).^10;
                end
            end
        end
    end

end
```

Ambiente Beleuchtung



Schachbrettmuster




c

```
1   function y = chessboard(x,a,b)
2 -  c=3;
3 -  if mod(floor(x(1)/c),2) == 1 && mod(floor(x(2)/c),2) == 0 && mod(floor(x(3)/c),2) == 0
4 -      y=a;
5 -  elseif mod(floor(x(1)/c),2) == 1 && mod(floor(x(2)/c),2) == 1 && mod(floor(x(3)/c),2) == 0
6 -      y=b;
7 -  elseif mod(floor(x(1)/c),2) == 1 && mod(floor(x(2)/c),2) == 0 && mod(floor(x(3)/c),2) == 1
8 -      y=b;
9 -  elseif mod(floor(x(1)/c),2) == 1 && mod(floor(x(2)/c),2) == 1 && mod(floor(x(3)/c),2) == 1
10 -     y=a;
11 -  elseif mod(floor(x(1)/c),2) == 0 && mod(floor(x(2)/c),2) == 0 && mod(floor(x(3)/c),2) == 0
12 -     y=b;
13 -  elseif mod(floor(x(1)/c),2) == 0 && mod(floor(x(2)/c),2) == 1 && mod(floor(x(3)/c),2) == 0
14 -     y=a;
15 -  elseif mod(floor(x(1)/c),2) == 0 && mod(floor(x(2)/c),2) == 0 && mod(floor(x(3)/c),2) == 1
16 -     y=a;
17 -  elseif mod(floor(x(1)/c),2) == 0 && mod(floor(x(2)/c),2) == 1 && mod(floor(x(3)/c),2) == 1
18 -     y=b;
19 -  end
20 -  end
```

Mehrere Objekte

```
33    % Berechnung von Wahrheitswerten, Nullstellen und Beleuchtung fuer jedes
34    % Objekt
35 -  for i = 1:amount_objects
36 -      [Bool(:,:,i),NS(:,:,i)] = Newton(grid,eye,rays,str2func(equations{i}), newton);
37 -      ABig(:,:,:,i) = lighting2(rlight,amb,dir,lamp,str2func(equations{i}),NS(:,:,i),eye,rays,Bool(:,:,i),rho(i,:),chess(i),spec);
38 -  end
39
40    % Bestimmung des naehesten Objektes
41 -  for i = 1:height+1
42 -      for j = 1:width+1
43            % Index naehstes Objekt
44 -          k = 0;
45 -          for l = 1:amount_objects
46 -              if (Bool(i,j,l) == 1 && k == 0) || (Bool(i,j,l) == 1 && NS(i,j,l) < NS(i,j,k))
47 -                  k = l;
48 -              end
49 -          end
50
51 -          if k == 0
52 -              lightings(i,j,:) = [0;0;0];
53 -          else
54 -              for l = 1:3
55 -                  lightings(i,j,l) = ABig(i,j,l,k);
56 -              end
57 -          end
58 -      end
59 -  end
```

Laufzeitoptimierung

```
%befuellen der Matirx mit Beleuchtungswerten

    for i =1:size(rays,1)
        for j=1:size(rays,2)
            %prueft ob Schnittpunkt positiv ist
            if bool(i,j)>0
                a = dot(squeeze(Normal(i,j,:)),rlight);
                if a< 0
                    %diffuse, d.h. blickwinkelunabhaengige Helligkeit
                    I_diff(i,j,1) = dir .* rho_color(1) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                    I_diff(i,j,2) = dir .* rho_color(2) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                    I_diff(i,j,3) = dir .* rho_color(3) .* norm(a)./(norm(squeeze(Normal(i,j,:))).*norm(rlight));
                end
            end
            %ambiente, d.h. richtungsunabhaengige Helligkeit
            if bool(i,j)==1
                I_amb(i,j,1)= amb(1);
                I_amb(i,j,2)= amb(2);
                I_amb(i,j,3)= amb(3);
            end
        end
    end
```

```
    % Indices wo Objekt ist
    indices = bool>0;
    % Skalarprodukte an diesen Stellen mit Normalenvektor und Lichtrichtung
    dots(indices) = Normal1(indices).*rlight(1) + Normal2(indices).*rlight(2) + Normal3(indices).*rlight(3);
    % Indices da wo Skalarprodukte kleiner 0
    newindices = dots < 0;
    % Befuellen dieser Matrixindices
    I_diff1(newindices) = dir .* rho_color(1) .* abs(dots(newindices))./(sqrt(Normal1(newindices).^2 + Normal2(newindices).^2 + Normal3(newindices).^2).*norm(rlight));
    I_diff2(newindices) = dir .* rho_color(2) .* abs(dots(newindices))./(sqrt(Normal1(newindices).^2 + Normal2(newindices).^2 + Normal3(newindices).^2).*norm(rlight));
    I_diff3(newindices) = dir .* rho_color(3) .* abs(dots(newindices))./(sqrt(Normal1(newindices).^2 + Normal2(newindices).^2 + Normal3(newindices).^2).*norm(rlight));
    % Zusammensetzen der Matrix
    I_diff(:,:,1) = I_diff1;
    I_diff(:,:,2) = I_diff2;
    I_diff(:,:,3) = I_diff3;
```

Ausblick

- Schatten
- Spiegelung
- Glaskugel
- Bounding Box
- Anpassung an Objekte