Mitteilung 09 | 11 Januar 2010

Eine Tages-Übung zum Raytracing

Auf der nächsten Seite seht Ihr in Pseudocode, an C angelehnt, einen rekursiven Algorithmus für das Raytracing. Ihr sollt Euch damit in Gruppenarbeit befassen.

Studiert zunächst gemeinsam den Code, um einen generellen Überblick zu gewinnen.

Identifiziert die Teile, die weitere Detaillierung verlangen. Manche davon werden einfach sein, andere werden mehr Mühe verlangen. Wir wollen mindestens einige davon behandeln. Das sollt Ihr vorbereiten, indem Ihr die Teilaufgaben noch etwas genauer fasst.

(Diese Aufgabe werdet Ihr bei einer Übungsaufgabe von Blatt 6 brauchen.)

Wir werden danach die Aufgabe behandeln, das Schnittgebilde einer Kugel mit einem Strahl zu bestimmen. Dazu sollt Ihr Vorüberlegungen anstellen.

Der Pseudocode der folgenden Abbildung ist aus Peter Comninos, *Mathematical and computer programming techniques for computer graphics*. London: Springer Verlag 2006, p. 397

Frieder Nake: OAS Gebäude Linzer Str. 9a Raum 3015 | (0421) 218-3525 | nake@informatik.uni-bremen.de
Sprechzeit: Mittwoch 12-14 | www.agis.informatik.uni-bremen.de

```
function recursive_ray_tracing_renderer(scene, image)
{
 for each pixel in the image do
  {
   level = 0;

   generate a primary_ray from the eye to the centre of the
   pixel;

   pixel_colour = trace_ray(primary_ray);
  }
} /* recursive_ray_tracing_renderer */

function trace_ray(ray)
{
 find the closest point of intersection of the ray with a
 surface of the scene;

 if there are no intersections then return(background_colour);

 compute the unit normal of the closest surface at the point
 of intersection;
 if surface is emitter
        then colour = emitter_shader(ray, surface, point);
        else colour = 0;

 for each light source do
  {
   generate a shadow_ray to the light source;

   if light source is visible then
    colour = colour
    + direct_shader(point, surface, normal, light) / n_lights;
  }

 if surface is specular then
  {
   level = level + 1;

   if level > max_level then return(colour);
    {
     generate the reflected_ray;
     colour = colour + trace_ray(reflected_ray);
    }

   if surface is transparent then
    {
     generate the transmitted_ray;
     colour = colour + trace_ray(transmitted_ray);
    }
  }
 return(colour);
} /* trace_ray */
```

Algorithm 10.1 The outline of the recursive ray-tracing rendering algorithm.