Uitleg theorie raytracing

Wat is een raytracer

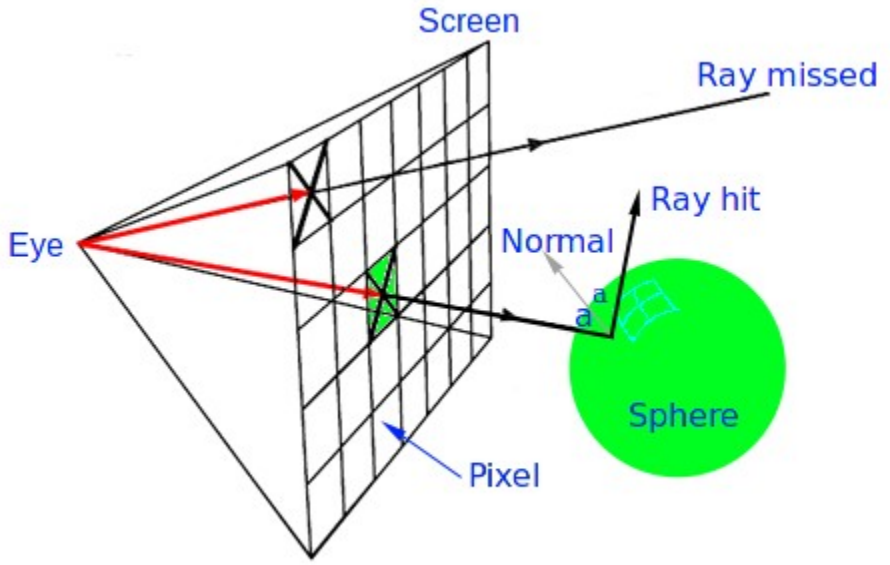
Een raytracer is een computerprogramma om foto-realistische 3D afbeeldingen te genereren, waarbij lichtbreking en weerkaatsing zeer natuurgetrouw weergegeven kunnen worden. De huiswerkopgaven bij de C++ lessen werken stap voor stap toe naar een basale raytracer.

Om installatieproblemen met libraries te vermijden en ook on-line te kunnen coderen, wordt voor uiteindelijke weergave gebruik gemaakt van beelden die bestaan uit letters, zgn. ASCII art. De toegepaste principes leiden echter bij toepassing van gekleurde pixels tot natuurgetrouwe beelden.

Voorbeeld van een afbeelding verkregen d.m.v. raytracing:

Het principe van raytracing is verbazend simpel. Vanuit het oog van de waarnemer tast een denkbeeldige straal dwars door het scherm heen pixel voor pixel de omgeving af. Als hij daar een voorwerp tegenkomt, neemt het betreffende pixel de kleur van dat voorwerp aan.

Het volgende diagram illustreert dit:



Er is nog een verfijning mogelijk. Als de straal een voorwerp raakt, reflecteert hij, waarbij hoek van inval a ten opzichte van de normaalvector (pijltje “Normal” loodrecht op boloppervlak) gelijk is aan hoek van uitval a. Het kan zijn dat de weerkaatste straal nog weer een ander voorwerp raakt. Dan pikt hij daar ook kleur van op en zo verder. Het uiteindelijke resultaat is dat het betreffende pixel een mengkleur krijgt van alle vlakken waartegen de straal is weerkaatst. Dit geeft de illusie van reflecties.

Verdere verfijningen betreffen het gedrag van de straal bij het passeren van transparante voorwerpen, omgaan met matte oppervlakken, gebruik van lichtbronnen en weergave van schaduwen.

Mis of raak

Om te kijken of een straal een bol raakt, moet je weten of hij dicht genoeg bij het middelpunt van die bol komt. Als hij dichterbij komt dan de radius van die bol, dan raakt hij de bol, anders niet.

De functie die de afstand van een lijn (de straal) tot een punt (middelpunt van de bol) berekent, is gegeven:

```
float distFromRay (Ray const &ray) {
      return ray.support.sub (center) .cross (ray.direction) .norm ();
}
```

Mocht je hiervan de wiskundige achtergrond van willen weten, kijk dan bij:

https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

De werking van de zelf te programmeren functies *sub* (verschil), *cross* (kruisproduct) en *norm* worden helder en bondig uitgelegd op:

https://nl.wikipedia.org/wiki/Vector_(wiskunde)

Het is handig om bij raytracing de stralen (lijnen) voor te stellen door middel van een steunvector en een richtingvector. Dit wordt uitgelegd op tijdstip 1:35:24 van:

http://wiztech.nl/#lineaire_algebra

Om te weten waar een straal de vloer raakt, stel je de hoogte-coordinaat gelijk aan de hoogte-coordinaat van de vloer en kijk je wat de andere twee coordinaten dan zijn. Bij een schaakbord-patroon op de vloer kun je zo bepalen of hij een witte of een zwarte tegel raakt. Een truukje is de zwarte tegels voor te stellen door gaten, zodat ze automatisch donker worden (“niks geraakt”).

Reflectie en recursie

Of je een voorwerp wel of niet raakt geeft voldoende informatie voor een plat “schaduwbeeld” van bijvoorbeeld een aantal bollen en een vloer met een schaakbord-patroon. Verder dan dat hoef je bij de opdrachten niet te gaan.

Een mooie, maar pittige, aanvulling is het weergeven van reflecties. Als een straal tegen een glimmend oppervlak botst, dan kaatst hij onder dezelfde hoek met de normaal weer weg de andere kant op (zie bovenstaand diagram). Om het raakpunt te berekenen heb je in ieder geval de volgende theorie nodig:

// https://en.wikipedia.org/wiki/Line-sphere_intersection

De betreffende straal kan na weerkaatsing verder gevolgd worden, tot hij opnieuw weerkaatst en zo verder. Bij elke weerkaatsing pikt hij kleur en/of helderheid op van het voorwerp waarin hij weerkaatst.

De makkelijkste manier om dit te programmeren is door middel van recursie. Het principe is dat een ray een *scan* functie heeft die zichzelf aanroept na elke weerkaatsing. Voorafgaand aan de nieuwe aanroep worden steunvector, richtingvector en kleur- en/of helderheids-informatie aangepast aan kleur en stand van het meest recente weerkaatsende vlak.

Om te voorkomen dat de straal zeer lang blijft heen en weer kaatsen tussen allerlei voorwerpen (en je computer zeer lang blijft rekenen) is het ook nodig het aantal doorlopen weerkaatsingen van een straal bij te houden. Als dat bijvoorbeeld meer is dan 3, dan

weerkaatst de straal niet verder.

Tips

Geef het midden van je beeldscherm de coordinaten (0, 0, 0). Werk in meters. Zet het oog van de waarnemer bijvoorbeeld op (-0.8, 0, 0), dit is 80 cm *voor* het scherm. Plaats de 3 bollen en de schaakbord-vloer zo dat ze in het echt ook op redelijke grootte te zien zouden zijn. Het beste is te werken met bollen ongeveer ter grootte van een voetbal, enkele meters *achter* je scherm, dus met positieve 1e coordinaat.

Let op: Als het midden van het scherm de coordinaten (0, 0, 0) heeft, dan **heeft de vloer een negatieve hoogte**. Hij ligt immers lager dan het midden van het scherm. Bovendien **moet de vloer zich geheel achter het scherm bevinden, niet ervoor**.

Doorloop alle pixels rij voor rij en genereer voor ieder pixel een straal. Reken de coordinaten van het pixel meteen om naar meters t.o.v. het midden van het scherm en **werk verder consequent in meters, niet in pixels!** Pixels zijn een technisch detail van jouw specifieke computer. Meters daarentegen zijn gekoppeld aan de fysieke werkelijkheid en die is stabieler dan de specificaties van beeldschermen.

De steunvector van elke straal bevindt zich in het oog van de waarnemer. De richtingvector kun je vinden door het veschil te berekenen tussen coordinaten van de desbetreffende pixel en de genoemde steunvector.

Werk met helderheden, bijv. tussen de 0 en de 1. Reken die pas op het laatste moment om naar ASCII characters die ongeveer overeenkomen met die helderheden. Gebruik maximaal 8 characters, meer is alleen maar lastig.

De opdrachten werken stap voor stap naar de eindopdracht toe. Test ook de tussenliggende opdrachten grondig. Als je dit “voorwerk” goed doet, kom je niet voor verrasingen te staan.

Het weergeven van weerkaatsing is een extra. Wel heel mooi om te zien. Indien je dit wilt progammeren is moet je er op letten dat een lijn en een bol die niet los van elkaar staan meestal twee snijpunten hebben. Daarvan is alleen het “vroegste” snijpunt interessant, het andere leidt tot “spookbeelden”.

--//--