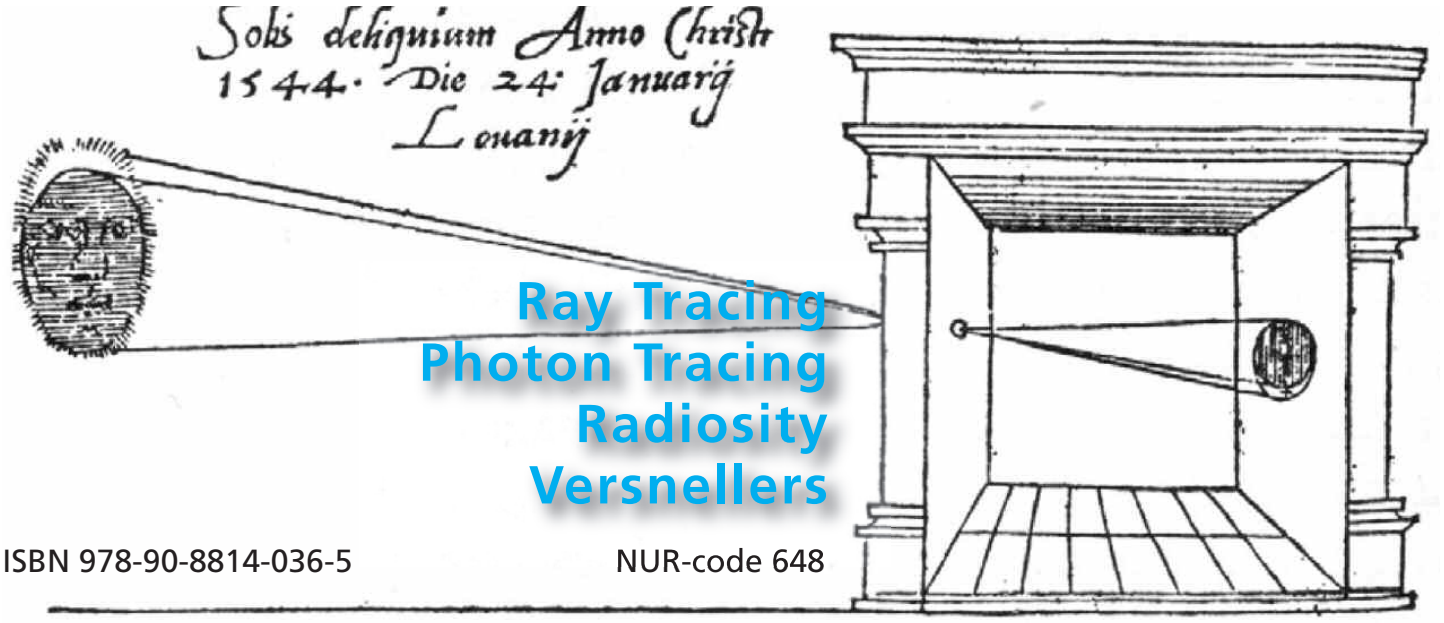
Principe Renderen

Render principes verklaard
Oude en nieuwe school
CPU versus GPU

Ray Tracing
Photon Mapping
Wiskundige onderbouwing

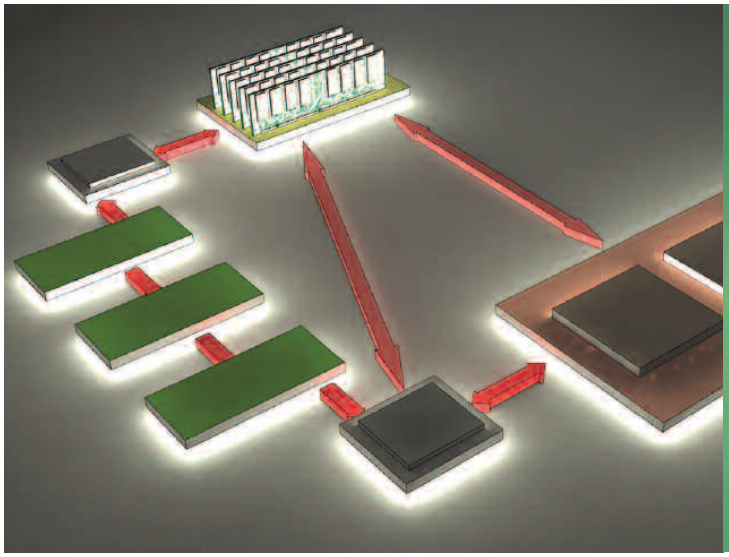
Render programma codes
Werkstations - NVIDIA
Literatuur - geschiedenis

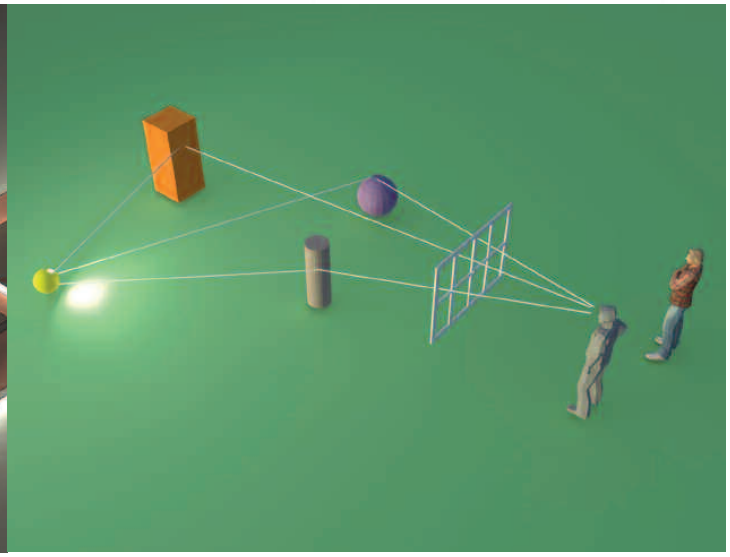


Ray Tracing
Photon Tracing
Radiosity
Versnellers

ISBN 978-90-8814-036-5

NUR-code 648

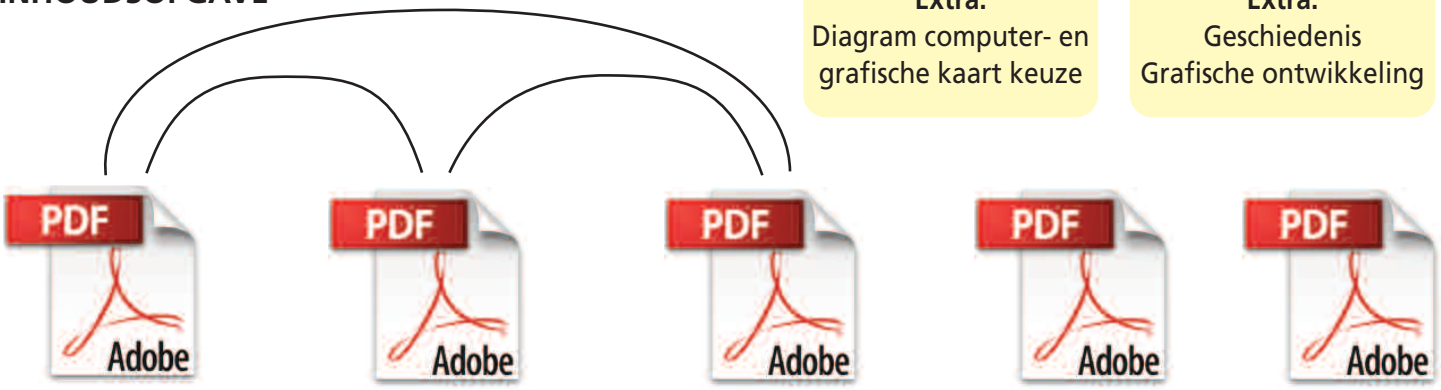




INHOUDSOPGAVE

Extra:
Diagram computer- en grafische kaart keuze

Extra:
Geschiedenis Grafische ontwikkeling



render principes | programma codes | literatuur met originele pdf's | NVIDIA LIJN | Werkstations

Inleiding

Renderen is een soort religie aan het worden, waarbij er een strenge leer is en sterke voor- maar ook tegenstanders te vinden zijn. Eén van de vele arena's waarin zich dat voor iedereen afspeelt zijn de locale Forums van de diverse programma's.

De religie breidt zich ook uit richting hardware, processor en/of grafische kaart, super werkstations (Boxx Extreme) en dito parallele kaart oplossingen. Maar daar blijft het niet bij, marketingafdelingen van render firma's doen er nog een schepje bovenop door allerlei claims naar voren te brengen, meestal zonder enige technische onderbouwing. Het waarheidsgehalte van veel beweringen is nauwelijks vast te stellen.

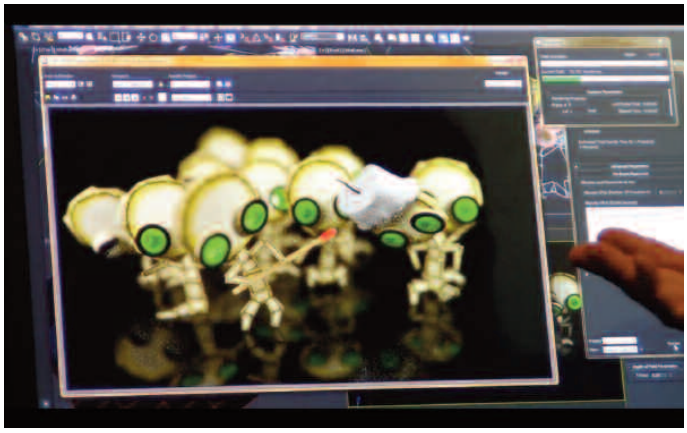
Eén van de claims is "*Physic correct renderings*", of "*Het snelste render programma dat er te koop is*", "*Dit is het enige renderprogramma* ". In principe zou er maar één programma aan deze beweringen kunnen voldoen, maar er zijn er meer dan tien die allemaal claimen "*Wij zijn de snelste*". '*Fysisch correct*' is geen vastgelegde term, waardoor de marketing ook daarmee aan de haal is gegaan. Wat 'fysisch correct' echt is, komen we wellicht achter door het lezen van meer dan 160 PDF's met onderzoeken, dissertaties en proefschriften. Literatuur PDF.

Daaruit blijkt dat zelfs voor wiskundigen en render programmeurs het nauwelijks mogelijk is om van buitenaf (het uiteindelijk product) zelfs maar een deel van zo'n claim te bewijzen of uit te leggen. *"And Maxwell Render Engine (within Artlantis) is considered to be the most physically accurate rendering engine on the market."*

De bedoelding van de term "fysisch correct" zou kunnen zijn dat het programma zo dicht mogelijk bij de natuur (= werkelijkheid) komt, waarbij een lichtbron photonen uitstraalt en tevens elektromagnetische lichtgolven. De werkelijkheid (de render afbeelding) is echter virtueel met vectoren en gevangen in data en algoritmes.

En wat blijkt uit de literatuur van specialisten? Algoritmes kunnen zó complex zijn, dat de term op losse schroeven komt te staan. Daar komt nog bij dat de verschillen tussen de diverse renderingstijden, factoren van meer dan 100 x uiteen kunnen lopen. Er zijn bijzonder weinig deskundigen die zelfs met de broncode op schoot (kennis die nooit beschikbaar zal worden gesteld) de claim kunnen aantonen of bewijzen.

Verder is het een trend om allerlei toepassingen

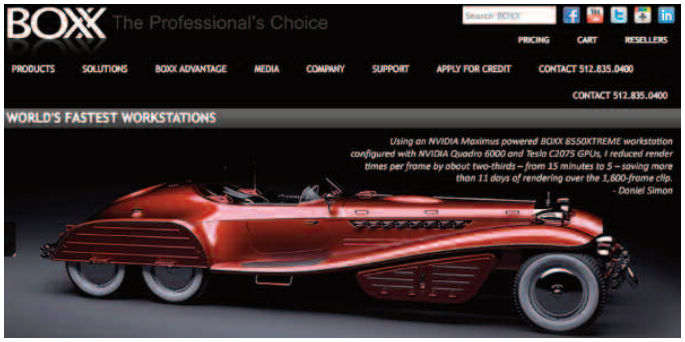


Film Siggraph 2012
http://youtu.be/ouOjzuvcC5w
BOXX@SIGGRAPH 2012: CPU vs. GPU-Based Rendering in 3ds Max

"Abvent has announced the release of Artlantis 4.1, powered by the Maxwell Render Engine from Next Limit Technologies. This latest release combines the speed and ease-of-use Artlantis users have come to rely on with the power and physical accuracy of Maxwell Render. As the fastest stand-alone 3D rendering application developed especially for architects and designers, Artlantis 4.1 takes architectural visualization to new heights."

Render principes in twee uitvoeringen:

PAARS met alle beschikbare literatuur bestanden (ruim 600 MB) en BLAUW met ruim 180 MB interessante literatuur bestanden bijgevoegd.
Er is ook een Engelstalige "Render principles" verschenen bij uitgeverij Ontmoeting.



ROW van Boxx
Alleen ethernet is nodig en netspanning.
Render Farm on Wheels, the First Turn-Key Rendering Solution. A self contained render farm solution. Met tien render BOXX modules keurig naast elkaar in één 24U of 42U rack compleet met display, keyboard, master node, storage, network switch etc.

uit te brengen, die gekoppeld zijn aan de **fotografie**, waarbij de lens, inclusief zijn fouten in het lensstelsel als uitgangspunt voor de render instelling wordt gebruikt. Daarbij wordt ook sluitertijd, scherptediepte, onscherpte, diafragma, interne lensspiegelingen, soort opzetlens, openingshoek, lensvertekening en verkleuring meegenomen.

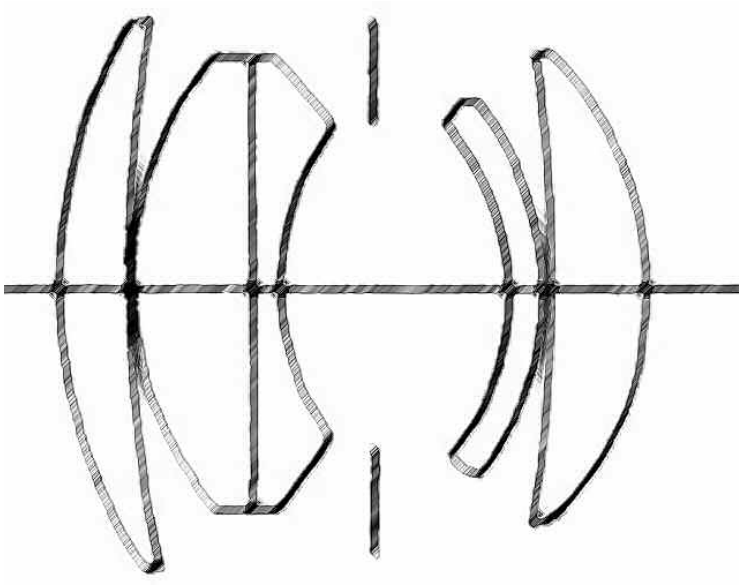
De 'sport' om te renderen met **NVIDIA CUDA** kaarten is van de laatste jaren en zal zich verder ontwikkelen tot een volwaardige manier waarop renders kunnen worden uitgevoerd, die niet mogelijk zijn met dehuidige programma's van de vorige generatie die alleen met de CPU werken. De snelheidswinst met gebruikmaking van goede NVIDIA GPU kaarten (of parallel kaarten à la Intel of anderen in de toekomst wellicht) en dito moederborden (voorlopig voorbehouden aan Windows computer gebruikers, met uitzondering van Mac Pro) is zo aanzienlijk dat professionele renderbedrijven zich niet meer kunnen veroorloven om deze kant van de ontwikkeling links te laten liggen. De meeste hullen zich nu nog in de ontkenningsfase.
De ontwikkelingen gaan steeds verder. Er is helaas stagnatie opgetreden bij NVIDIA met de nieuwe generatie Kepler kaarten die Fermi zouden gaan opvolgen. Maar wellicht wordt dat binnenkort wel weer wordt ingehaald. **Intel** is eind 2012 ook nog niet op de markt verschenen met z'n parallele kaart met Xeon Phi-coprocessoren.

*Zowel **Octane Render** als **Bunkspeed** geven aan dat Kepler nog niet die snelheid en verbetering laat zien die werd beoogd, soms zelfs het tegendeel. Vandaar dat deze kaarten nog niet worden ondersteund of aanmoedigd.*

De mogelijkheden van GPU-renderers zijn verschillend en divers, renderen binnen een aantal seconden levert een aanzienlijke tijdwinst op en maakt andere toepassingen mogelijk. Er kunnen meer renderingen worden afgeleverd met hogere resolutie met betere kwaliteit.

Het is ook mogelijk om binnen de deadline animatie films te maken, die altijd voor een prop in de planning zorgden.

WebGL / IOS gebaseerde 3D bestanden. Of 360° panorama's voor tablets en smartphones. Dat kan commercieel bijzonder goed worden benut ten gunste van de klant, de kwaliteit van het product en de productie capaciteit van het renderbedrijf. Ook augmented reality is met iPad en iPhone al mogelijk.



"De totale render CPU-tijd kost ca. $ 0,10 per uur, de professionele ontwerper ligt dichter bij de $ 40,00 per uur, waardoor interactiviteit ook bijzonder belangrijk is".

"Renderen is de kunst om compromissen zoveel mogelijk te verdoezelen."

"wij praten er liever niet over"

Uit Forum geknipt

Q: The wish is an 'Displacement function' . . .
Answer:
Displacement is already on the wishlist for a while.... Thanks anyway.

Answer:
Sorry to be sceptic, but I think this answer is worth very little.
It is easy to add things to a wishlist, but the wishlist has to become reality one day! If not... Delete it from the wishlist and or give us a timetable of the goals Abvent set for some new future stuff.
In example GPU rendering is on the wishlist

as well for a long time but my guess is that there will never be a ATL GPU render engine, but it looks like a policy of having a wishlist of things that user would like, even you know they probably-never-can-do-stuff from the wishlist.

I tried to get some answers about GPU being on the wishlist (see links below) but never got a satisfying answer. It looks like to me this is also happenng to the displacement function ...

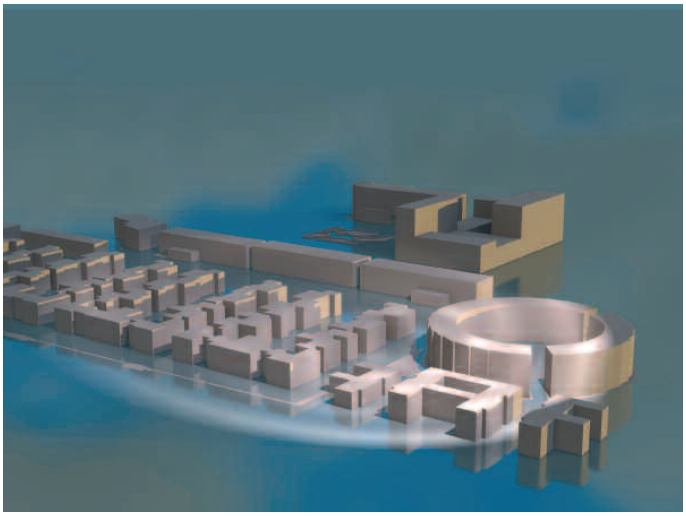
Q. For how long is that on the wishlist?
Please be more specific and give a goal or timeline of realizing stuff from the wishlist. I'm sad about it, cause the current possibilities are immense and it is a great program ... but a program is becoming obsolete ... and see that everything I said now, does not change the structure of the program! Get new animation options, BG, clouds, lighting, editing, rendering ... everything that already exists but with new features, new options.

Zomaar een reactie van een teleurgestelde Forum inzender. Indien de beantwoording van een Forum bij de fabrikant(en) er 'bijhangt', dan haken **professionele klanten** op een gegeven moment af. De beantwoording van Forum vragen zou altijd door technisch onderlegde vakmensen moeten worden gedaan. Iemand die dicht bij de bron (programmeurs) staat en niet bang is om ook kritische vragen professioneel en on-commercieel te beantwoorden. Beantwoording van vrijwel alle vragen is belangrijk, omdat vragenstellers met geheel verschillende niveau's óók antwoord willen hebben op hún vragen.

NIET BEANTWOORDEN WEKT WREVEL OP BIJ VRAGENSTELLERS.

Besteed daar als fabrikant tijd, moeite en geld aan en zie het komt vanzelf via een betere profilering plus extra verkoop terug.

Zie een Forum als *onkostenpost* en de Forum vragenstellers raken steeds meer verwijderd van het merk en diens producten. Toekomstige klanten lezen de reacties en gaan twijfelen om zo'n product aan te schaffen.



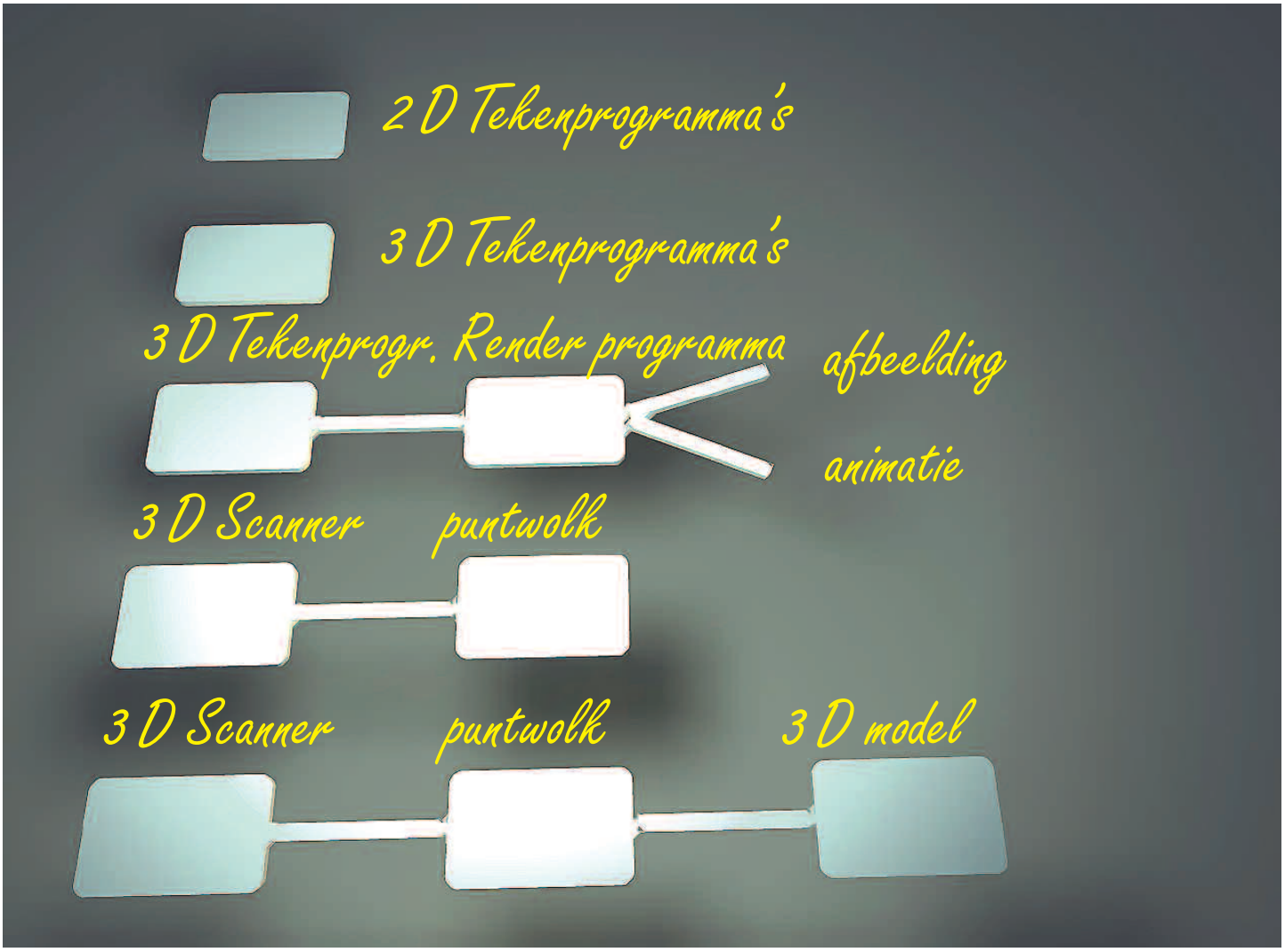
Rendering Artlantis Studio

Wegens privacy redenen worden geen namen van vragenstellers genoemd. Origineel in Forum te vinden.

Webinars over Render programma's
Indien u deze volgt kunt u zelf constateren dat er regelmatig te weinig voorbereidingstijd wordt uitgetrokken om het de kijkers optimaal naar de zin te maken. Het vraagt veel tijd en moeite vóóraf om het goed te doen. Een professionele webinar is als een boek, met een titel, korte omschrijving van de onderwerpen en duidelijke vermelding met welke hardware wordt gedemonstreerd (als tekstpagina's te lezen in de webinar).

Opmerkingen
Webinars gaan vaak over gemakkelijk toegankelijke onderdelen van een programma. Maar dat zou met een aantal praktijk tips en trucs beter kunnen.

Vragen tijdens een Webinar worden vaak terzijde geschoven of naar E-mail doorverwezen. Kritische opmerkingen van kijkers worden soms genegeerd.
Goede webinars zijn er ook en vormen een verrijking van de kennis van de kijkers. Bij de mindere goede komen alleen de positieve onderwerpen aan bod, telkens opnieuw om de tijd vol te maken. Daardoor blijft er geen tijd over om ingewikkelde vragen te beantwoorden en dat komt goed uit. Vragen over 'kunt u mij ook nog wat laten zien van . . .' staan niet in de planning en komen niet in aanmerking om aan het slot te tonen.
Life renderingen maken gaat helaas vaak niet vanwege de tijd. Bij GPU render programma's zou dat wel kunnen.



1. 2 D Tekenprogramma's zoals we die al tientallen jaren kennen verdwijnen langzaam van het toneel.
2. 3 D en 2,5 D programma's nemen de plaats in.
3. 3 D programma's zijn zo algemeen en standaard, dat de volgende stap voor de hand ligt: een renderingsprogramma. Om de 3D- of 2D tekening aan te kleden met lichtbronnen, texturen en nog veel meer.
4. Hoe komen we aan een optimaal 3 D model? Vroeger kwam daar het meetlint aan te pas, diverse kladtekeningen en vervolgens een heus digitaal 3 D model op maat. Tegenwoordig zijn er apparaatjes om afstanden in kamers en woonhuizen nauwkeurig te meten. Een stap verder is het werken met 3 D scanners die middels lasers de ruimte of het object aftastten en de afstand en kleur van elke reflectie in een puntwolk (in 3 D) vastlegt. Zo'n puntwolk is een dikke brei van losstaande punten in een 3 D vlak.
5. Aan de hand van een 3D puntwolk is het mogelijk om redelijk snel en accuraat een compleet nieuwe 3D tekening te maken.

http://www.pointools.com/pointools-plug-in-for-sketchup.php

Voordelen van het gebruik van puntwolken:

a. Maak de scan -> model workflow korter en sneller
b. Sla de kostbare conversies over
c. Behoud de visuele kwaliteit en nauwkeurigheid
d. Vergroot de productiviteit van het maken van 3D modellen.

PDF:
Pointools-Plug-in-for-SketchUp.pdf
nummer [160] in de literatuur.

OUDE SCHOOL

- **Ray tracen blijft altijd bestaan, daar kunnen we (de fabrikanten) nog jaren mee door.**
- **De gemiddelde klant heeft toch geen tijd om zich in nieuwe ontwikkelingen te verdiepen.**

NIEUWE SCHOOL

- **Photon gebaseerde programma's zijn altijd langzaam geweest, maar met versnellers (SPPM en DACRT) komt daar verandering in. De technieken en mogelijkheden zijn vrijwel gelijk aan Ray Tracers, maar dan met meer mogelijkheden, fraaier en sneller.**

Foto van De Zaan in Zaandam, Nederland opgedeeld in kubussen met één kleur.

In 3D ontwikkeld met hulp van Open Source **Processing**.

Elk stukje van de foto krijgt zijn unieke kleur Deze manier van opdelen heeft overeenkomsten met de manier waarop bij een renderprogramma een afbeelding opgebouwd wordt uit unieke pixels, die uiteindelijk de visualisatie vormen van het gehele 3D model.

Zie ook **Visualisatie** met Processing:
http://ontmoeting.nl/visualisatie-2.html

Siggraph 2012

Siggraph is één van de belangrijkste instellingen waar bedrijven en universiteiten samenkomen om naar nieuwe ontwikkelingen op het gebied van renderen, games, film en grafische toepassingen te kijken.

OUDE SCHOOL

- **Het verhogen van de CPU klok frequentie is de meest toegepaste methode om de snelheid bij renderen te vergroten**

- **Let niet op nieuwe ontwikkelingen, er zitten veel haken en ogen aan, waaronder een beperkt GPU geheugen waardoor grote 3D bestanden niet met GPU, maar wel met CPU gerenderd kunnen worden. Voeg gewoon een groot aantal CPU Cores toe en de renderwereld staat voor u open. En de fabrikant kan mooi gebruik maken van z'n oude codes.**

- **Afhankelijk van het gekozen systeem van renderen zullen de verschillen in renderingstijd bij diverse fabrikanten niet veel meer kunnen worden verbeterd. Het lineaire eind van de curve is in zicht. De markten zijn verdeeld en de processor fabrikanten doen er alles aan om de CPU interessant te houden, maar wellicht dat ze de tijd niet mee hebben en de boot al gedeeltelijk hebben gemist.**

NIEUWE SCHOOL

- **Bij de GPU wordt steeds meer gewerkt met parallele systemen, zowel voor de processoren als de bijkomende geheugens. CUDA van NVIDIA lijkt voorlopig de oplossing om met veel Cores een bijzondere prestatie te leveren. Maar Intel komt er binnenkort aan.**

- **Niemand werkt meer met CPU processoren alleen. De nieuwe school gaat uit van een C#, Fortran of C++ programma om parallele processoren zo goed mogelijk hun werk te laten doen. Daarbij zijn gigantische snelheidsverbeteringen mogelijk binnen de nieuwe opgerekte beperkingen van de hardware. De geheugen beperking van GPU is inmiddels software matig opgelost.**

- **Als we dan toch de overstap maken (eerst de klant & dan de fabrikant? . . .) naar parallele hardware, dan zullen zelfs kleine verbeteringen in de hardware, maar ook in die van de software, behoorlijke verschillen in de prestaties en overeenkomstig in de marktverhoudingen opleveren. De markten zullen drastisch door elkaar worden geschud.**

Nieuw en oud
GPU en CPU

In (meer dan 10.000) publicaties zoemt het van nieuwe ontwikkelingen op het gebied van GPU als zaligmakend voor de NIEUWE TOEKOMST VAN HET RENDEREN.
De Forums puilen uit van berichten van gebruikers die de fabrikanten aansporen om 'even snel over te schakelen naar GPU'.

In deze uitgave gaan we daar graag in mee, omdat de curves van parallele processen veel sneller omhoog lopen, dan bij welke toekomstige CPU ontwikkeling dan ook. Maar, wie had verwacht dat **Intel** in z'n goedkope processoren een minimale grafische kaart zou integreren typen HD 3000 en HD 4000?
Een soort grafische kaartfunctie die de concurrende kaarten van NVIDIA en ATI (NB. Intel's eigen merk) overbodig zou moeten maken voor de onderkant van dat specifieke marktsegment?

Dat **Intel** daarmee de plank volledig misslaat is aan hen, ze hebben onvoldoende specificaties, waardoor deze alleen voor simpele E-mail berichten en internetten zijn te gebruiken. Voor serieuze Games, 3D- of Render toepassingen is het ten ene male ONGESCHIKT. Nergens zijn de spec's te vinden dat bv. OpenGL, OpenCL of DirectX zou worden ondersteund. Staat het er niet bij (Intel site), dan is het kennelijk niet ondersteund.

Een miskleun aan de **onderkant van de computer markt** om de prijs van een Notebook, Laptop en Desktop zo laag mogelijk te maken.

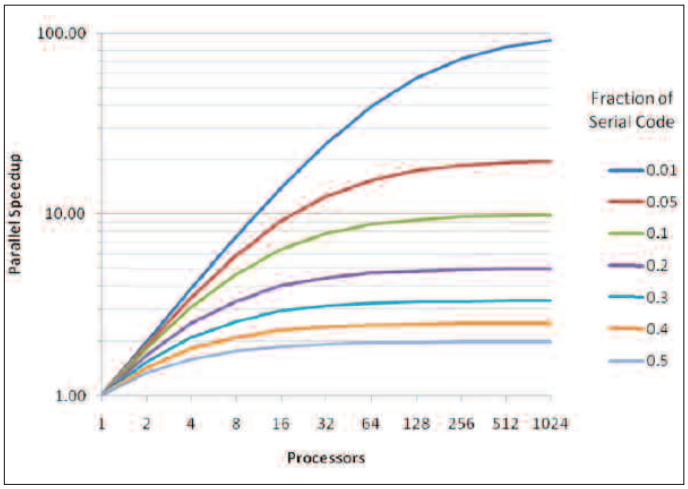
Helaas zien we dat **Apple** met een serie ook gebruik maakt van de Intel HD HD 3000 en HD 4000 chip's en ook de goedkopere Mac Mini met HD 3000 en gedeeld RAM geheugen voor de grafische functies. Geen goede ontwikkeling voor een kwaliteitsproduct (Apple onwaardig) met een hogere prijs dan de standaard Windows laptopbakken. Toch functioneren deze HD 3000 en HD 4000's bij Apple wèl incl. de in het operating systeem aanwezige OpenGl. Bij Windows wordt dat wellicht opgevangen door DirectX, alleen diverse laptops bij Windows met de HD's geven geen thuis als een 3D programma moet worden opgestart.

Is de keuze voor GPU renderen t.o.v. CPU dan zo eenvoudig?
In het geheel niet, het probleem is ingewikkeld. De fabrikant van een renderingsprogramma moet bijna alles wat tot nu toe werd ontwikkeld aan de kant schuiven en geheel OPNIEUW beginnen. En dat in een tijd dat de verkopen van bestaande producten zijn afgenomen ten opzichte van een aantal jaren daarvoor. De fabrikant zal nieuwe programmeurs moeten aantrekken, die net van de universiteit komen en daar de ontwikkelingen van GPU renderen hebben bestudeerd. Daar komt bij dat de fabrikant binnen enkele jaren werd geconfronteerd met meer dan twintig nieuwe serieuze concurrenten op ZIJN renderingsmarkt en het eind is nog lang niet in zicht.

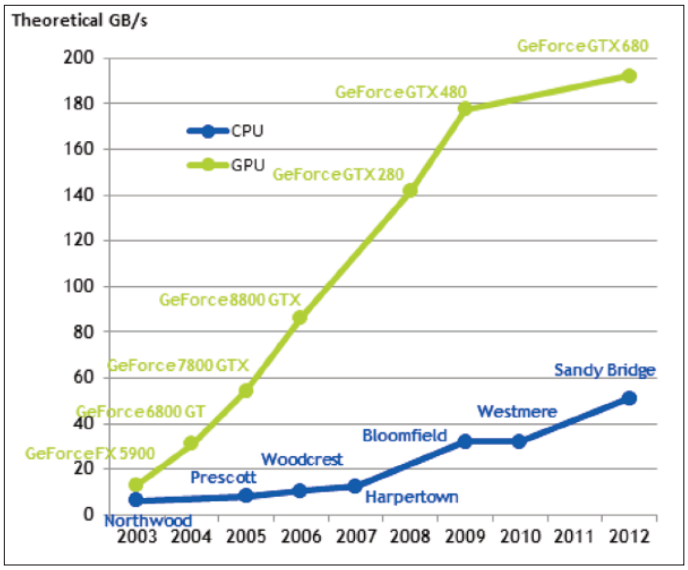
Op de vorige pagina een overzicht in gekleurde blokken, dat enerzijds het bestaande beeld van renderingsprogramma's goed weergeeft, anderzijds het probleem van omschakeling naar voren brengt. Welke fabrikant is in staat (heeft zoveel geld achter de hand) om een dergelijke kostbare overstap voor te financiëren?
En welke fabrikanten proberen nog geld los te krijgen om de ontwikkeling in hoog tempo om te buigen?

Eén van de nieuwe trends wordt door deze 2 fabrikanten (er zijn er meer) zichtbaar:

V-RAY met z'n nieuw RT serie renderprogramma's (2012 geïntroduceerd), waarbij een



Een niet alledaagse manier van presenteren.
De grafiek behoeft dan ook enige uitleg.
Horizontaal: aantal denkbeeldige processoren, links vertikaal: de relatieve snelheidswinst bij parallele processen.
Rechts vertikaal: de diverse kleuren van de hoeveelheid (percentages) seriële (old school) codes in een programma.
Kiezen we de bovenste curve de donker blauwe, dan bestaat het programma uit slechts 1 % aan seriële code en 99 % uit parallele codes voor aansturing van de GPU hardware. Hiermee wordt dan direct de grootste snelheidswinist geboekt.



Links vertikaal: de theoretische GB/s snelheid. De groene grafiek de diverse GPU kaarten met typenummers. Onderaan in het blauw de moederbord architectuur / processor architectuur van CPU's. Bijgewerkt tot ca. 2011. Het verloop en de mogelijkheden van parallel processoren zal overduidelijk zijn.

Arion rendering van interieur

keuze mogelijk is tussen CPU, OpenCL en CUDA renderen. De klant kan dan z'n eigen hardwarekeuze maken. Geschikt voor een brede markt van **beginners tot aan professionals**. Maar wel drie afzonderlijke renderingsprogramma onderdelen om te ontwikkelen! Ook **Thea Render** is bezig om een GPU CUDA gebaseerd programma te ontwikkelen.

Arion met een "*hybrid acceleration unbiased rendering*" (GPU + CPU).

Daarmee kunnen alle GPU's in het systeem worden gebruikt, alle CPU's cores en uiteraard ook tegelijk, zowel GPU's als CPU's.

Alle beschikbare rekenkracht wordt daarbij optimaal benut. **Arian** heeft een oplossing gevonden voor de beperking van het VRAM geheugen van de GPU. Indien dat tekort schiet kan worden overgeschakeld naar CPU rendering (RAM) of een combinatie daarvan. In **Arion** is de opschaalbaarheid vrijwel lineair met de beschikbare hardware van de gebruiker. Hoe meer grafische kaarten worden bijgezet des te sneller gaat het renderen. Indien u bv. vier stuks GTX 480 GPU's gebruikt, dan kunt u tot 3 - 4 x zo snel renderen dan met één enkele GTX 480!

NB Hou er rekening mee dat in het algemeen de grootste 3D scene die u kunt renderen afhankelijk is, bij het gebruik van meerdere GPU kaarten, van de hoeveelheid geheugen van het *kleinste* geheugen van één van de kaarten. Maar later zien we (out-of-core) dat ook deze beperking ook is opgelost.

Verder de al jaren geleden gestarte Nieuw Zeelandse firma **Octanerender** die zich geheel heeft gestort op het CUDA NVIDIA GPU renderen. Maar ook **Arion**, **Lightworks** en **OptiX Ray Tracing Engine** hebben forse snelheidsverbeteringen en innovaties laten zien.

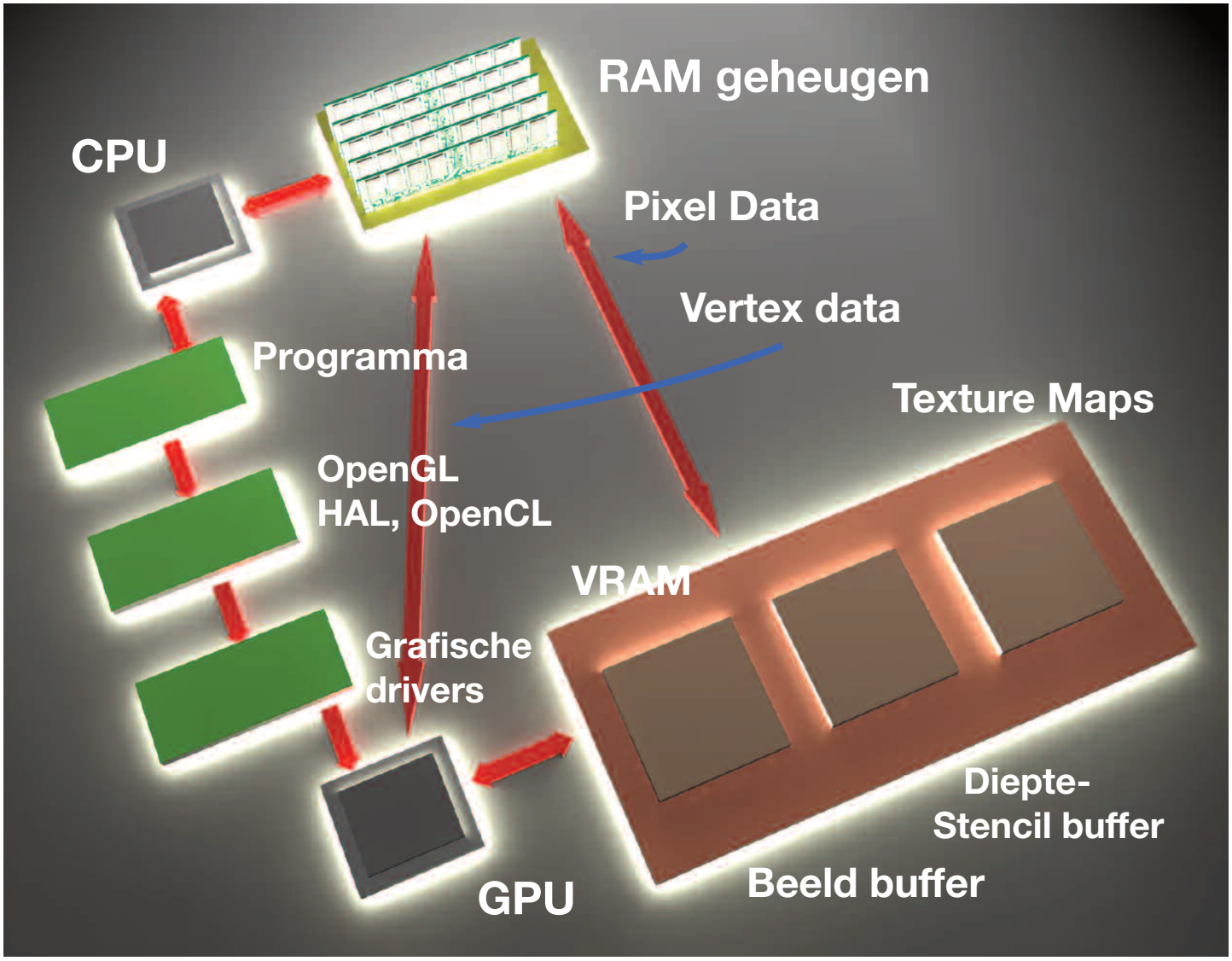
Codes zelf ontwikkelen of aanschaffen?

IRAY is het renderprogramma dat NVIDIA aanbiedt aan diverse software leveranciers waaronder Autodesk, Bunkspeed, Dassault Systems, Cinema 4D e.a.

Mental Ray wordt gebruikt door ondermeer Autodesk, Dassualt Systems, FreeForm, PTC, Matica, VISoft en Cinema 4D.

Ook **Maxwell** doet daar aan mee en verkoopt **Maxwell Engine** als plug-in voor LightWave, SolidWorks, Modo en geïntegreerd in Artlantis 4.1. Met het verschil dat Maxwell begin 2013 nog gebruik maakt van oude techniek, waardoor de renderingstijd in schril contrast komt te staan met de aanstormende GPU nieuwkomers. De rendertijd samen met de kwaliteit, UI en de geboden mogelijkheden zullen de kernpunten vormen om de toekomstige klant te lokken.

OptiX van NVIDIA wordt **zonder licentie** aangeboden met een compleet renderdeel, out-of-core systeem en nog veel meer.



Communicatie tussen CPU en GPU

Het RAM geheugen lijkt een sleutelrol te spelen in het proces en dat klopt gedeeltelijk. De snelheid van de bus is van belang, hoe hoger des te beter. Die van het GPU geheugen (VRAM van video Random Access Memory) is vele malen sneller. Communiceren wordt daardoor afgeremd door het RAM hoofdgeheugen en de computerbus.

Via het programma worden de instructies uitgevoerd in de CPU in samenwerking met het RAM geheugen. Afzonderlijke opdrachten lopen via OpenGL, HAL, OpenCL e.a. via de Grafische drivers naar de GPU. Deze kan gebruik maken van o.m. snel VRAM geheugen. In VRAM zijn ook de front- en back image buffers te vinden. In de front is de exacte pixel data te vinden die zichtbaar is via de Viewport (genormaliseerde coördinaten in een rechthoekig vlak voor het scherm, waar de data verschijnt.
De Diepte- & Stencil buffer is een extra buffer als aanvulling op het Color buffer, ze maken vaak gebruik van dezelfde VRAM geheugenplekken in de GPU hardware.

In moderne parallel georiënteerde programma's wordt OpenGL en OpenCL overgeslagen en gaat de informatie rechtstreeks naar de GPU. Die door z'n parallele capaciteit vele malen sneller werkt dan welke CPU dan ook. Met de CUDA in moderne NVIDIA kaarten zijn meerdere cores aan te sturen die afzonderlijk van elkaar en tegelijk diverse intensieve taken kunnen afhandelen. Met name intensieve vector berekeningen voor renderingsprogramma's kunnen zo aanzienlijk sneller worden uitgevoerd.

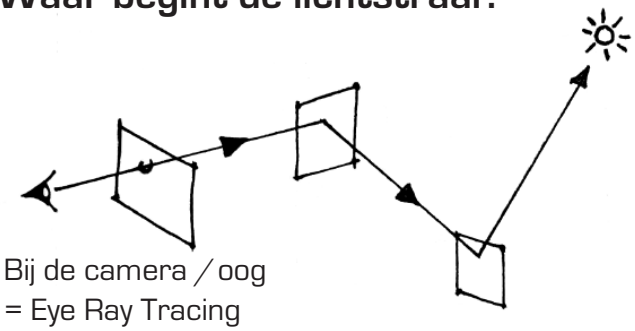
Geschiedenis
Ray Tracing

Het renderen van 3D modellen heeft zijn weg gevonden naar miljoenen gebruikers wereldwijd. Niet alleen voor architecten en interieurontwerpers, maar ook in films, Games, medici, natuurkundigen, wetenschappelijk onderzoek etc.
Rendering is altijd een uitdaging tussen hoeveel computerkracht en soort programma (~ prijs) u wilt uitgeven om het resultaat in zo kort mogelijke tijd te bereiken. S. Metzger (CG supervisor) grapt er op los door te zeggen dat er niet over renderingen kan worden gesproken zonder dat iemand of een groep zich daar over opwint: "Renderingen maken is een soort religie aan het worden met een enorm aanbod van renderingsprogramma's en heel veel nieuwe ontwikkelingen".

De toekomst wordt vaak bepaald door in het verleden ontwikkelde gedachten, ideeën en uitvindingen. Voor renderen geldt dat in hoge mate, aangezien de huidige renderingsprogramma's allemaal stuk voor stuk te danken zijn aan o.m. Appel en Turner Whitted, Cook, Jensen, Glassner en vele anderen. We zullen er een aantal van de revue laten passeren met naam en toenaam.

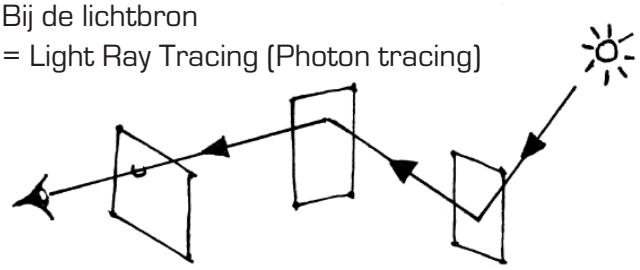
Render programma's zijn in wezen *'redelijk eenvoudig'* om te schrijven en goed als universiteits afstuudeeropdracht te doen. Maar in een sterk concurrerende markt en commerciële omgeving is het extreem complex en ingewikkeld. Er worden heel andere eisen aan de interface gesteld, dan voor universitaire deelprojecten met één vastgesteld doel. Voor **Arnold Render** met Solid Angle als object waren 200.000 regels hoogwaardige C++ codes nodig om een goede balans te vinden tussen 'niet te veel toeters en bellen en trucs'. Marcos Fajardo (Arnold oprichter) vertelde tijdens Siggraph 2010: "De totale render CPU tijd kost ca. $ 0,10 per uur, maar de professionele ontwerper ligt dichter bij de $ 40,00 per uur, waardoor interactiviteit ook bijzonder belangrijk is".

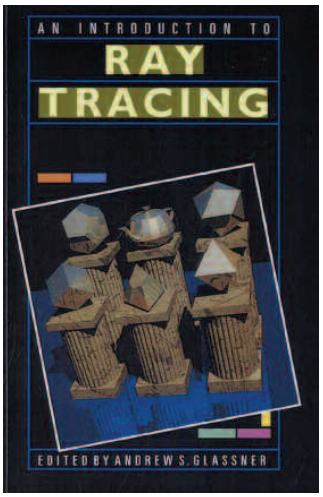
Waar begint de lichtstraal?



Bij de camera / oog
= Eye Ray Tracing

Bij de lichtbron
= Light Ray Tracing (Photon tracing)



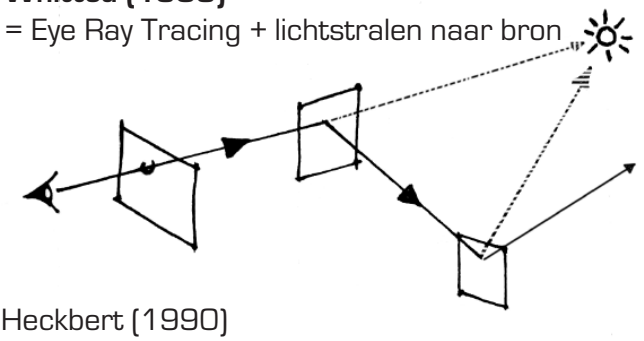


Ray Tracing (1989) van Andrew S. Glassner, een van z'n boeken.

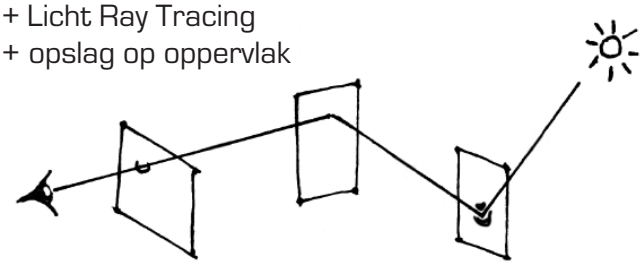
An Introduction to Ray Tracing develops from fundamental principles to advanced applications, providing "how-to" procedures as well as a detailed understanding of the scientific foundations of ray tracing. It is also richly illustrated with four-color and black-and-white plates.

Hybride oplossingen

Whitted (1980)
= Eye Ray Tracing + lichtstralen naar bron



Heckbert (1990)
= Eye Ray rracing
+ Licht Ray Tracing
+ opslag op oppervlak



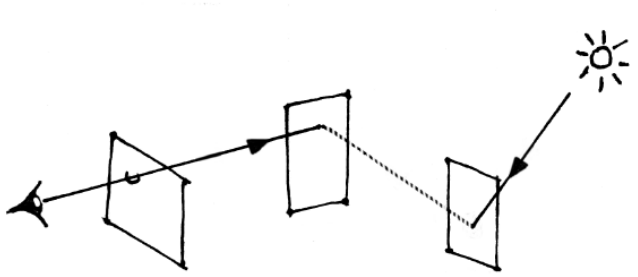
Renderen is vele keuzes maken die tot de relatief en subjectief 'beste' resultaten leiden. Binnen de tijd die u daarvoor ter beschikking hebt. Bij de te maken keuzes komt ook de complexe schaduwtechniek, bewegingsonscherpte, scherptediepte, licht effecten, achtergrondlicht en afbeeldingen en met welke objecten en even zo belangrijk met welke Shaders / materialen er wordt gewerkt.

'Renderen is de kunst om compromissen zoveel mogelijk te verdoezelen'

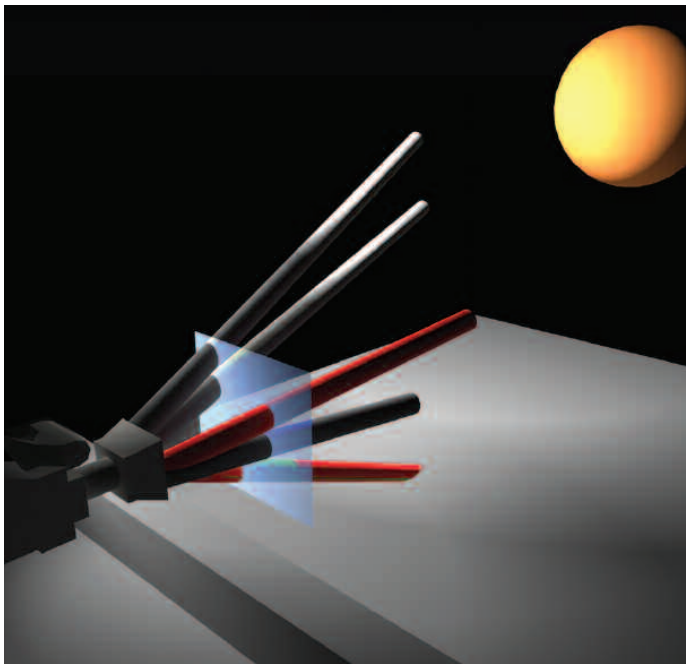
Arthure Appel beschreef in 1968 als eerste het principe van Ray Tracing. Daarmee het verborgen oppervlakte probleem oplossend. En om schaduwen met 3D polygonen weer te geven. Goldstein en Nagel toonden later in 1971 aan hoe Ray Tracing kon worden toegepast om rechthoekige vormen te gebruiken. Kay en Greenberg maakten in 1979 bekend hoe transparantie kan worden opgenomen. In **Whitted's** CACM artikel beschreef hij het algemene Recursive Ray Tracing Algoritme (1980) met reflectie en refractie van spiegelende oppervlakken en schaduw van puntlicht bronnen. Whitted is één van de vele die regelmatig worden aangehaald en z'n ideeën worden nog dagelijks toegepast in render programma's. In 1987 voegde Heckbert er realistisch renderen aan toe. Maar daarmee verstomde de ontwikkelingen zeker niet. Meer en meer worden de nieuwe onderzoeksresultaten verborgen achter betaalde internetsites die deze in de vorm van PDF's aan de man brengen.
Of in universitaire studies die mede worden gefinancieerd door bedrijven waardoor het onderzoeksresultaat te vaak achter gesloten deuren blijft.

De eerste boeken over renderen gebaseerd op natuurkundige principes zijn die van Cohen en Wallace (Radiosity and Realistic Image Synthesis 1993), Sillion en Puech (Radiosity and Global Illumination 1994. In beide professionele werken worden de Radiosity methoden beschreven.
In 1995 kwam daar **Glassner** z'n encyclopedie bij, die het theoretische fundament legde voor realistische renderingen. Illumination and Color in Computer Generated Imagery

Veach (1995)

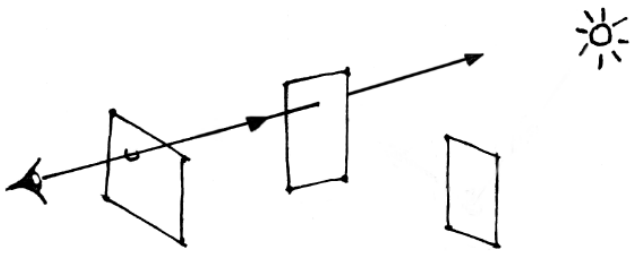


Eye Ray Tracing
+ Licht Ray Tracing
+ verbindingslijn tussen reflectiepunten

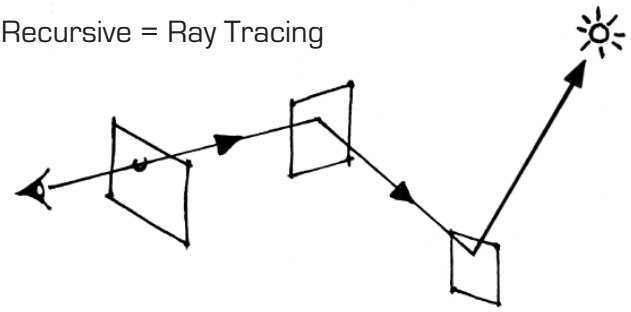


Casting versus Tracing
Werpen t.o.v. trekken

1ste pad = Ray Casting

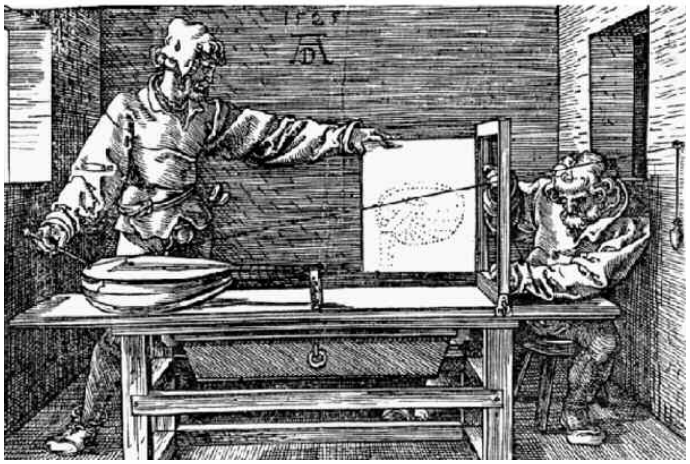


Recursive = Ray Tracing



van Haal in 1989 is één van de eerste uitgaven die renderen in een natuurkundige gebaseerd framewerk plaatste.
Detré, Bala en Bekaert zette in 2006 met hun *Advanced Global Illumination* het geheel in de huidige tijd.

Greenberg e.a. maakte zich sterk voor een natuurkundige precise renderingen (1997) die gebaseerd waren op de maten van materialen uit de reële wereld. Tegelijk werd het menselijk visuele systeem onder de loep genomen.
Al deze ideeën hebben, samen met die van Cornell, gedurende meer dan tien jaar bijgedragen aan de basis voor het huidige digitale renderen van 3D modellen.



Principe van Renderen
Albrecht Dürer. Het touwtje geeft een lichtstraal weer. Om het strak te houden is op de achterwand een gewicht eraan gebonden.

Ray Tracing principe

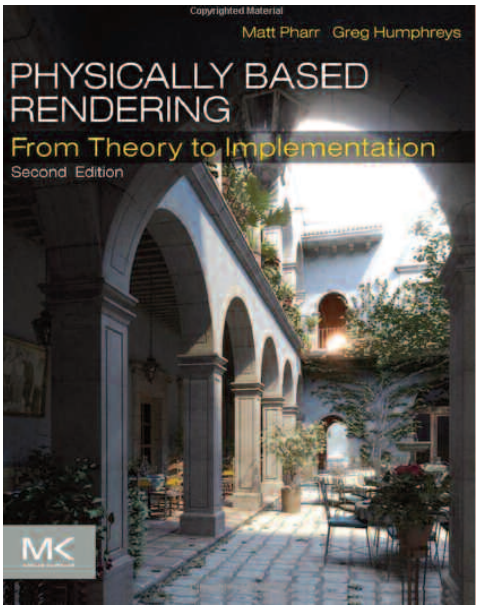
Bepaald de zichtbaarheid van oppervlakken door deze met virtuele lichtstralen af te tasten, vanaf het oog (camera / oog) van de toeschouwer naar de objecten in de 3D scene (Backward tracing). Daarmee voorziet dit principe in een eenvoudige en krachtige manier om te renderen incl. Global reflectie en transmissie effecten. Het verschil tussen Forward ray-tracing of light tracing en Backward tracing of Eye-tracing komt later aan de bod.

De basis Ray Tracing techniek omvat:

1. Eén brandpunt (het oog van de kijker / camera) in het 3D vlak en één venster waardoor heen gekeken wordt op een bepaalde afstand. Het venster kan worden gezien als een raster van vertikale en horizontale lijnen waarbinnen zich de individuele pixels bevinden. Dat zijn uiteindelijk de pixels van de te renderen afbeelding. Een lage resolutie scan gaat met een 'grof' raster van slechts enkele honderden lijntjes, een hoge resolutie levert vele duizenden lijnen op en kwadratisch aantal pixels.

2. Voor elk pixel in het venster worden een aantal lichtstralen geschoten vanaf het gekozen brandpunt (1). In principe houden we aan, dat ons oog (camera) een brandpunt heeft, terwijl zowel een digitale- als analoge camera en ons oog een klein oppervlak heeft waar de lichtstralen worden verwerkt tot foto of resp. tot 'zichtbaar beeld'. Een te beperkt aantal lichtstralen om tot de kleur en helderheid van een pixel te komen, is te zien aan de **ruis** tijdens de opbouw en soms ook in het eindproduct. Pas na behoorlijke aantallen lijkt de ruis te verminderen. Zelfs met tien duizenden lichtstralen is het niet altijd gezegd dat een object / voorwerp van het 3D model ècht helemaal is afgetast.

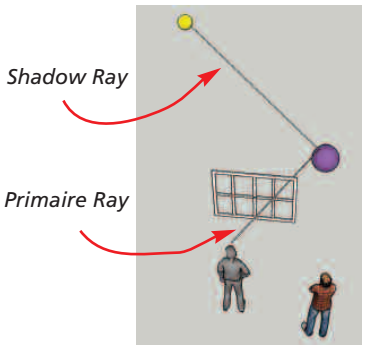
3. Daarna volgt het berekenen / beoordelen van de lichtstralen, die een object raken (treffen), eigenlijk het hart van elke Ray Tracer. Het berekenen van deze punten is eenvoudig voor vlakken en bollen, maar ingewikkelder voor andere objecten. Het uitzoeken van welk deel van het 3D model wordt geraakt met een lichtstraal, is het meest intensieve onderdeel van het renderingsprogramma. Hier gaat veel tijd in zitten om dat vast te stellen.



4. Bij elk trefpunt van een lichtstraal dient de **Normaal** te worden berekend. De Normaal staat loodrecht op het oppervlak, in het punt van aanraking. Later kan de in- en uitvalshoek in drie dimensies van de lichtstraal worden berekend, die ofwel door een oppervlak wordt weerkaatst (spiegelend), geabsorbeerd of door het oppervlak heen gaat (glas). In deze berekening wordt het oppervlak doorgegeven aan het pixelvenster. Zodra een lichtstraal in het 3D model is getrokken en het trefpunt is bepaald, dient te worden bekeken of het voorwerp in de schaduw, of juist in het licht van de aanwezige lichtbronnen ligt. Indien zo'n lichtstraal een object in de schaduw treft dan noemen we het schaduwstraal (Shadow Ray). De eerste straal die we vanuit het oog door het venster heen sturen wordt de Primary Ray genoemd of de Visibility Ray of de Camera of Eye Ray. Het gehele proces van lichtstralen zowel vanuit het oog of vanuit de lichtbron wordt Path Tracing genoemd. Het komt aardig in de buurt van de werkelijkheid waarbij Photonen vanuit een lichtbron één of meerdere voorwerpen treffen en daar opnieuw photonen los laten om de reis in een andere richting en met een lagere energie te vervolgen. Het enige verschil is dat we vanwege de efficiëncy niet vanuit de lichtbron, maar vanuit het oog starten.

Albrecht Dürer. Een brancpunt (oog), een doorkijk venster en de lichtstralen en object, alle ingrediënten van renderen.

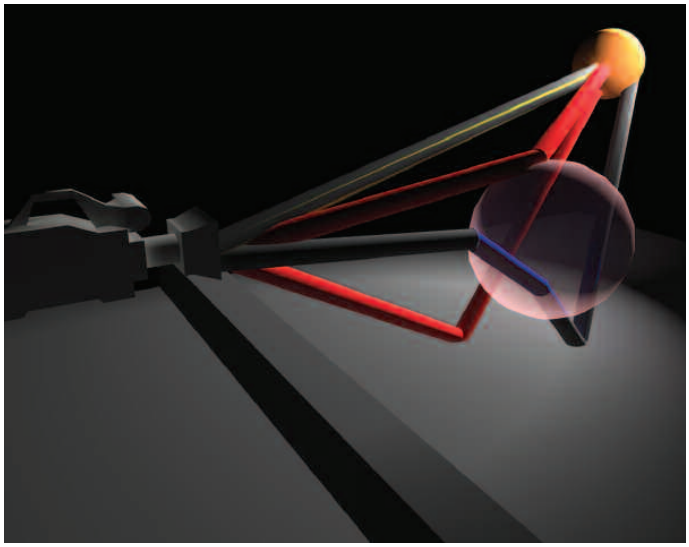
Na de eerste Primary Ray en trefpunt volgt een nieuwe straal, de Shadow Ray om te bepalen in hoeverre het trefpunt in de schaduw ligt.



Aan de hand van het aanrakingstrefpunt op de objecten wordt de helderheid en kleur bepaald. Dat betekent dat de weergave van de objecten alleen wordt vastgelegd door de manier waarop het licht het object raakt. Daarmee wordt een duidelijke scheiding gemaakt tussen wèl en niet verlichte delen van het object, helder of geheel donker. Maar er zijn veel soorten oppervlakken, gepolijste die op een spiegel lijken en oppervlakken waarbij de lichtstralen door glas heen gaan. En oppervlakken waarbij de lichtstraal uitdoofd en alle schakeringen daartussen in.

Bij renderingsprogramma's worden al deze verschillen vaak aangegeven bij de diverse corresponderende texturen of kleuren of oppervlakken die aan het oppervlak zijn toegekend. Bij Ray Tracing kunnen zowel reflecterende als absorberende materialen worden weergegeven.

Een lichtbron is een zender van photonen (lichtdeeltjes) die een 3D model kunnen beschijnen. Het is een scala van verschillende lichtstralen die bij de lichtbron(nen) begint en zo het 3D model ingaan. Om tot stilstand te komen, te worden gereflecteerd of te worden geabsorbeerd als ze een object tegenkomen. 3D objecten zien we omdat het licht wordt verstrooid op dat deel van het



oppervlak, er worden photonen terug gestuurd naar ons oog (camera) waardoor we de objecten kunnen waarnemen.

Aangezien de photonen bij elk oppervlak een nieuw spectrum van nieuwe photonen los maakt is het een complex geheel van lichtstralen die op elk oppervlak een andere werking hebben. Meerdere reflecties zorgen ervoor dat de structuur en het oppervlak van een object beter kan worden beoordeeld. Door de complexe meervoudige reflecties is het ook mogelijk om een indruk te krijgen van het schaduwgedeelte van een object. Schaduwgedeelten zorgen voor diepte in 3D objecten.
Om dit principe voor een renderingsprogramma toe te passen is helaas niet erg efficiënt. Er zouden dan miljoenen lichtstralen moeten worden berekend, die uiteindelijk niet in het zichtveld van de kijker (camera) zouden uitkomen. Dus heel veel voor niets uitgerekend, omdat ze niet binnen het genoemde venster vallen en dus geen bijdrage aan de uiteindelijke rendering opleveren. Vandaar dat het volgen (tracen) van een lichtstraal op deze manier bij renderen niet wordt gebruikt.

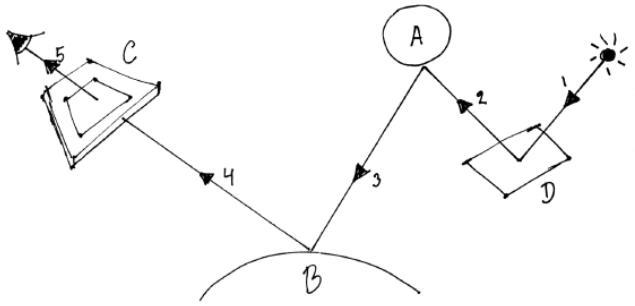
Een oplossing is gevonden door de lichtstralen op de *omgekeerde manier* door het 3D model te sturen. Niet vanuit de lichtbron(nen), maar vanuit de camera (oog). Daarbij wordt nog onderscheid gemaakt in lichtstralen die worden weerkaatst om uiteindelijk bij een lichtbron uit te komen.
En systemen waarbij van twee kanten lichtstralen worden berekend (één vanuit het brandpunt en één willekeurig vanuit de lichtbron(en)). Er wordt dan later bekeken welke lichtstralen elkaar binnen een bepaalde grensgebied kruisen. Ook wel *Bidirectionele Monte Carlo Ray Tracing* methode genoemd. Dit blijkt een delicaat proces te zijn dat zeer goed moet worden afgestemd. Zie ondermeer *Realistic Image Synthesis Using Photon Mapping* door Henrik **Wann Jensen** (2001).

We kiezen hier voor het eerste model. Door het venster (= renderpixels) naar het 3D model. De lichtstralen worden daarbij gevolgd of ze een oppervlak raken en zo ja, hoe dat in zijn werk gaat (met welk effect).

Het idee van **Ray Tracing** werd voor het eerst door **Arthur Appel** 1969 voorgesteld. Hij realiseerde zich dat z'n eerste gedachten van deze techniek (*Some Techniques for Shading Machine Renderings of Solids*) met een beperkt aantal programmaregels een belangrijk nadeel had:

> "This method is very time consuming, usually requiring for useful results several thousands times as much calculation time as a wire frame drawing. About one half of of this time is devoted to determining the point to point correspondence of the projection and the scene."

Met andere woorden de techniek is ronduit *langzaam*, trefpunten zoeken kost veel tijd. Dat was dan ook direct het grootste nadeel, maar met de komst van steeds snellere computers werd het mogelijk om deze techniek toe te passen. Toch blijft Ray Tracing t.o.v. bv. Z-buffer algoritmen relatief traag.



De lichtraal vanuit de lichtbron (1) wordt door een oppervlak gereflecteerd (D) en via nog een object A naar B gestuurd. Daar gaat de derde gereflecteerde lichtstraal door het venster heen naar ons oog / camera. De kleur in het venster C is afhankelijk van de kleur van Object B, maar op de achtergrond ook A en D via kleurmening in de lichtralen van 1, 2 naar 3. De hele loop van lichtstraal 1 - 2- 3 -4 - 5 kan worden voorgesteld door een boomstructuur (Ray Tree). Waarbij meerdere reflecties die in beeld komen de takken van een boom vormen. Bedenk dat in werkelijkheid in het programma niet vanuit de lichtbron, maar vanuit de camera wordt uitgegaan.
Daarbij zullen er zeker vele lichtstralen onderweg niet meer bijdragen aan het resultaat. Moeten ze dan wel eindeloos worden uitgerekend? Nee, het proces wordt beeindigd a) als de scene wordt verlaten (bounding box of sphere) of b) als de bijdrage te klein wordt.

De techniek werd later door ondermeer **Whitted** (1979 "*Improved Illumination Model for Shaded Display*") en vele anderen verder gepopulariseerd en versneld. Met extra Specular reflecties en refracties voor het actuele te verlichten model.
Een voordeel van Ray-Tracing en de equivalente Photonen in een life omgeving is dat effecten zoals reflectie en refractie kunnen worden toegepast. Om glas en spiegels na te bootsen. De manier waarop de lichtstralen worden gereflecteerd en breking plaats vindt wordt later besproken.

Ray Tracing is na al die jaren nog steeds een aantrekkelijke manier van renderen gebleken. Zie ondermeer het boek uit 2010 Elsevier van Matt Pharr en Greg Humphreys met als titel: *Physically based rendering from Theory to Implementation, second editon*. ISBN 978-0-12-375079-2

Eén van de voordelen van Ray Tracing is dat het algemene algoritme ook perspectief en gebogen oppervlakken meeneemt. De lichtstralen die worden gevolgd vanuit de camera (oog) naar de pixel locatie in het venster zijn allemaal binnen het zichtbeeld van het model. De dichtstbijzijnde objecten worden het eerste bereikt, waar direct ook de grootste aandacht naar toe gaat.

Ray tracing bestaat uit twee rekenintensieve afzonderlijke processen:

a) is het 3D punt zichtbaar voor een bepaald pixel?
b) pas schaduw toe.

Radiosity

De eerste methoden om te renderen middels Ray Tracing gaan uit van de locale verlichting door lichtbronnen in het 3D model. Highlights ontstaan door directe verlichting van objecten. In de beelden die worden geproduceerd gingen we uit van een mat oppervlak. Daarna volgde 'Global Illumination' met Ray Tracing vanuit het oog van de camera met meerdere lichtstralen vanuit de objecten (reflectie) zelf. Met Ray Tracing zijn spiegels, transparantie en abosorberende objecten mogelijk. Maar zelfs met deze geavanceerde technieken blijven er nog andere effecten over die niet meegenomen kunnen worden. Eén daarvan is de reflectie van licht tussen twee difuuse objecten of te wel *color bleeding*. Een effect dat met twee kubussen kan worden verklaard. Het komt het beste naar voren in lichte omstandigheden met één kleur en een wit object of witte onder- of achtergrond. De gebieden in de omgeving van de kleur krijgen een zweem van die kleur mee, waarbij de delen die het dichtst bij de kleur liggen het meeste kleur meekrijgen. Verderweg verwatert dit effect. Hoe feller het licht des te meer kleur bloeding kan worden waargenomen. Verklaarbaar omdat dan meer photonen in allerlei richtingen worden afgebogen waarbij de meeste energie in de buurt van het gekleurde model terecht komt.

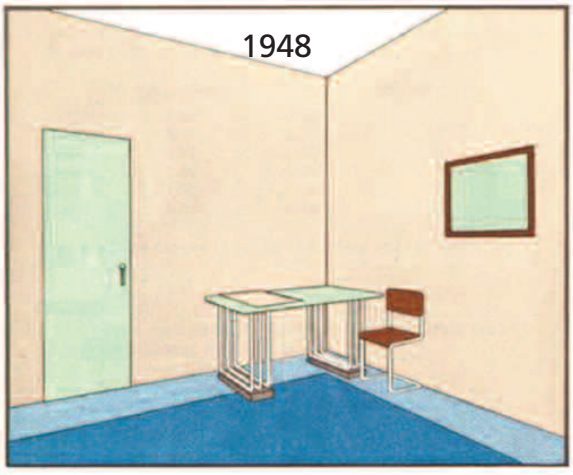
Conventionele Radiosity is een poging om indirect licht tussen oppervlakken met difuse reflectie hun resp. oppervlakte kleuren mee te geven. Met de techniek van **Radiosity** wordt licht als energie gezien en verlicht de 3D scene door een soort energie uitwisseling. Radiosity bepaald in welke mate elk object in de scene energie (licht) ontvangt en verder verspreid òf absorbeerd. De uiteindelijke Radiosity verlichting bestaat dus uit een energie uitwisseling in de 3D scene.

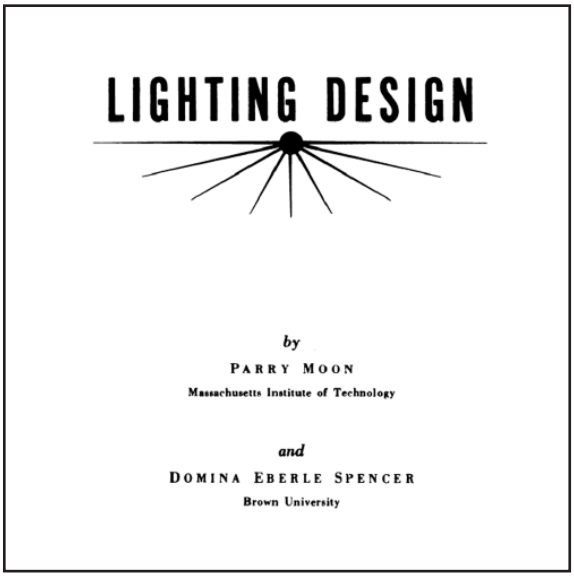
Radiosity is meestal veel minder helder en nauwelijks waarneembaar, dan de grote lichtbronnen in de scene. Vandaar dat ze extra omhoog worden geschaald (Tone Reproduction problem). Er wordt een scheiding aangebracht tussen gereflecteerde Radiosity en uitgestraalde Radiosity. In een aantal uitgebreide professionele renderingsprogramma's is dat dan ook door de gebruiker afzonderlijk in te stellen. Een aantal onderzoekers hebben zich hiermee bezig gehouden (Ashdown, Tumblin en Rushmeier komen met een berekening die de maximale helderheid en contrast bereik van de monitor er in betrekken. Er wordt ook gevoeligheid en contrast compressie constantes ingevoerd die de natuur zo dicht mogelijk benaderen.

Na het uitvoeren van deze toch wel belangrijke *Tone Reproduction* worden de Radiosity waarden gebruikt om de scene te renderen.

Cohen en Greenberg hebben een routine voorgesteld waarbij onderscheid kan worden gemaakt tussen interieur (binnen) scenes of gedeeltes en daarbuiten. Ook de hoeken van een object worden daarin betrokken. Cohen en Wallace pasten een gemiddelde toe van twee eerder geïntroduceerde patches.
Al deze extra bewerkingen vallen onder het 'fine tunen' van een renderingsprogramma. En maken een renderingsprogramma simpel en goedkoop of uitgebreid en gemaakt om door professionals te worden gebruikt met een corresponderend prijskaartje.

Verbeteringen
Telkens komt de tijdfactor om de hoek kijken, een rendering kan nog zo mooi zijn, maar als er geen tijd is om deze nogmaals uit te voeren, dan heeft het geen zin gehad. **Rendertijd** is bijzonder belangrijk ook bij professionele gebruikers. Deze maken weliswaar van geavanceerde hardware en soms zelfs meerdere computers tegelijk (renderfarm aan huis) gebruik, ze hebben vaak ook heel andere eisen omtrent resolutie, scherpte en detailweergave. Een simpele methode om de detailweergave te verbeteren is die van het opdelen van de Polygoon oppervlakken (Mesh) naar een fijnere maat.
Daarmee wordt ook het aantal af te tasten lichtstralen hoger die allemaal moeten worden berekend. Er kan ook een hogere resolutie worden gekozen met een fijnere maas waar doorheen gekeken wordt. Ook hier neemt het aantal lichtstralen (lees de rendertijd) sterk toe.





Lichtstralen kunnen in vier groepen worden verdeeld:
1. pixels rays of oogstralen
2. illumination rays of schaduw stralen
3. reflection rays (van objecten)
4. Transparency rays (stralen die door een object heen gaan)

2 t/m 4 zijn de bekende manieren waarop Photons worden opgewekt door het contact met oppervlakken, zowel er naar toe als er vandaan (schaduw, reflectie en transparantie).

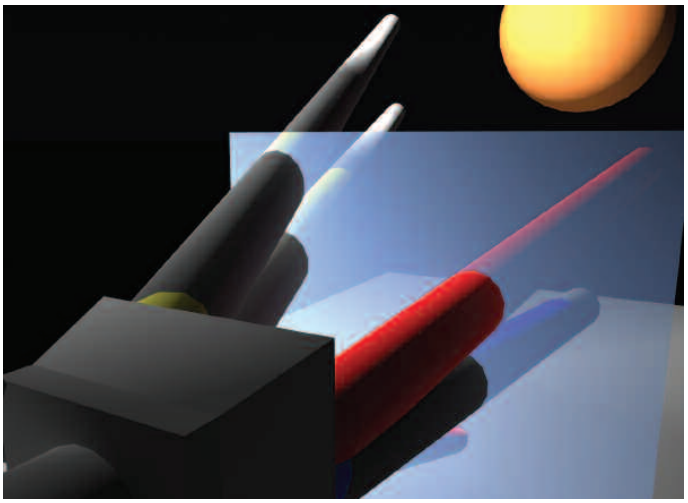
De schaduwstralen komen rechtstreeks van de lichtbron en worden dan weer opnieuw uitgestraald het 3D model in.

Het testen van bv. een schaduwstraal wordt gedaan door eenvoudig de vraag te beantwoorden van:
Is een straal van een lichtbron(en) direct in beeld te zien?
Indien de lichtbron direct waar te nemen is, dan is er een directe verbinding tussen oog en lichtbron. En er zullen een aantal photons (afhankelijk van de helderheid van de lichtbron) rechtstreeks ons oog (of virtueel venster voor het opbouwen van de resolutie pixels) bereiken.
Als de lichtbron niet rechtstreeks te zien is omdat er één of meerdere objecten tussen zitten, dan kunnen geen photonen ons oog bereiken. Het is dus het schaduwgedeelte in

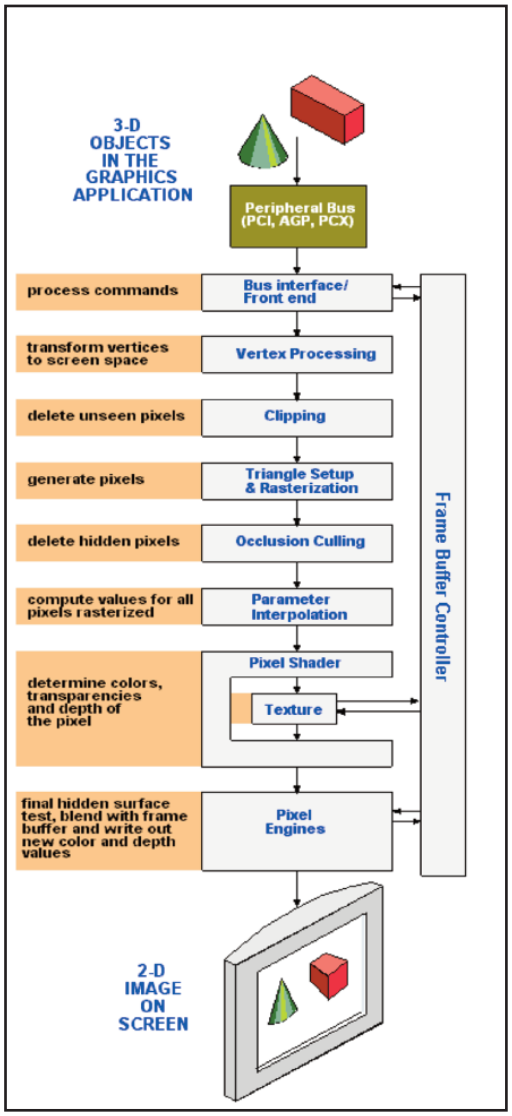
de scene voor die lichtbron. In dat geval wordt over een Shadow Ray (schaduw lichtstraal) gesproken.
In wezen is een ***Shadow Ray*** gelijk aan alle andere lichtstralen, behalve dat het wordt gebruikt om de scene als het ware af te tasten op mogelijke schaduwplekken.
Wat gebeurt er met een schaduwstraal als deze wel rechtstreeks naar een lichtbron kan kijken? De schaduwvoeler wordt dan omgedoopt tot *Illumination Ray*, die licht vanaf de lichtbron stuurd. En volgens de omgekeerde methode van het oog naar de lichtbron gaat. In al deze gevallen wordt met een mat oppervlak gewerkt, maar wat gebeurt er met een reflecterend oppervlak of een transparant oppervlak?

In de loop van de jaren werden daar eindeloos veel patches op los gelaten, om dat in redelijke banen te leiden. Inmiddels is dat wel uitgekristaliseerd en wordt een scheiding aangelegd in kwaliteit (prijs) van een renderingsprogramma om dit wèl of niet goed, of beter uit te voeren.

Radiosity wordt niet alleen bij grafische renderingen gebruikt. Ook bij Elektromagnetische golven (ionosfeer onderzoek) of bij geluidsweergave verbetering door onderzoek naar akoestische eigenschappen van een virtuele ruimte.



Ook bij thermische modellen wordt Radiosity toegepast. Het gaat terug tot 1920 toen stralingsvergelijkingen werden ontwikkeld voor het uitwisselen van energie tussen ideale diffuse oppervlakken. Later komen we opnieuw op Radiosity terug.



Een Renderings 'pipeline' volgens de Computer Language Co. Inc., 2004.

Moon en Spencer brachten in 1946 hun eerste 'Radiosity' renderings afbeelding naar buiten, waarbij ze de energie uitwisseling van een lege kamer (zie vorige pagina) berekenden. Die vervolgens aanschouwelijk werden gemaakt door stukken rechthoekig gekleurd papier uit te knippen en op te plakken volgens het **Munsell Color model**.

In 1950 werd het mogelijk om met computers verlichting en energie te berekenen.
In de jaren 1980 werd Computer Graphics toegepast (radiative transfer theory).
Zie de kamer met zwart/wit 'rendering' op de vorige pagina.

Ray Tracing en de natuur

Raytracing is in wezen niets meer of minder dan een poging om de natuur na te bootsen. De op de volgende pagina getekende ge-

kleurde bundels zijn in wezen oneindig klein (photons) en dicht op elkaar gepakt. Vaak worden ze met dunne gekleurde lijnen aangegeven, maar op papier blijft daar te weinig van over.

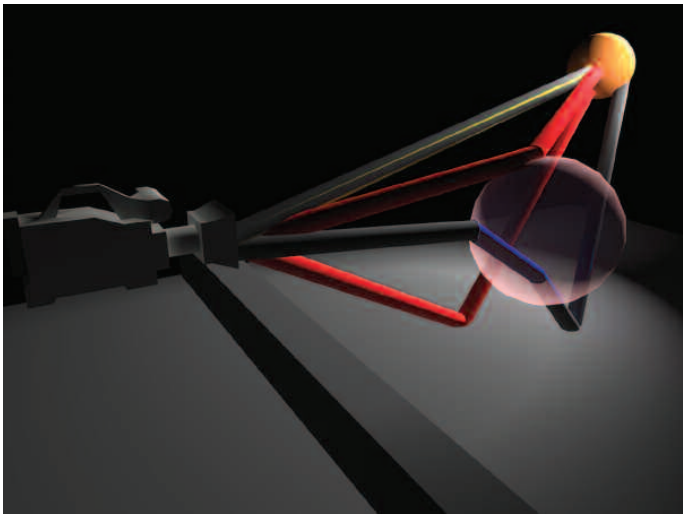
In de natuur worden de stralenbundels van de lichtbron richting object en camera gestuurd.
Daarbij worden oppervlakten geraakt, waarbij er vier mogelijkheden zijn:

a) het licht wordt in z'n geheel geabsorbeerd en dooft uit
b) het licht wordt iets verzwakt in energie gereflecteerd en de 3D ruimte ingestuurd
c) het licht wordt volledig gereflecteerd (spiegel)
d) het licht wordt in het materiaal afgebogen en daarna opnieuw gereflecteerd.

Indien we met renderen uitgaan van het 'natuurlijke' modellen dan worden miljoenen photonen (lichtstralen) van de lichtbron het 3D model ingestuurd. Daarvan zullen er ook heel veel nooit aankomen bij de kijker (in dit geval het oog van de camera). Ze worden in zijn geheel uitgerekend, maar dragen in het niets bij aan de uiteindelijk rendering.
Het kost veel rekenwerk en tijd, terwijl in het uiteindelijke resultaat er niets van te zien zal zijn. Met als nadeel de extra renderingstijd die de maker van de rendering voor lief zal moeten nemen.

Vandaar dat men bij het maken van een rendering het principe heeft omgedraaid, niet de lichtbron straalt licht uit en zendt photonen het 3D model in, maar de camera (het oog van de kijker). Meestal wordt hiervoor een fotocamera of filmcamera afgebeeld, maar een menselijk oog zien we ook in tekeningen. Bij renderingsprogramma's wordt ook vaak een **camera** als icoon voor het maken van een rendering gebruikt.

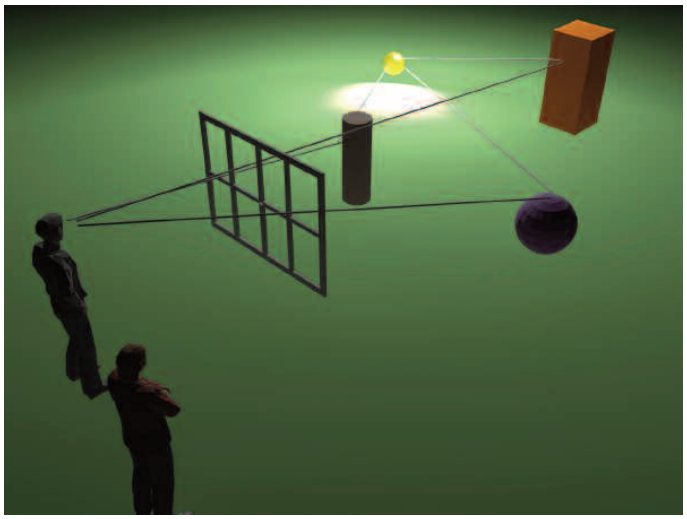
Vanuit de camera worden lichtstralen het 3D model ingeschoten, waarbij in wezen het rendement hoog is, elke lichtstraal gaat door een virtueel raster (de gekozen resolutie) heen en draagt meer of minder bij in de opbouw van de uiteindelijke rendering.



Tijdens dat licht bombardement vanuit de camera worden wèl of geen 3D objecten geraakt of bereikt. En dat geldt ook voor de lichtbron(en).

In een renderingsprogramma heeft een lichtstraal vanuit de camera een brandpunt, beginpunt en een richting (3D). In wezen is er een punt waar de lichtstralen vandaan komen en op een bepaalde afstand (door de gebruiker in te stellen) wordt een 2D vlak met een virtueel raster (de uiteindelijke rendering) gepaseerd. Hier zal het resultaat van de rendering zichtbaar worden.

Hoe ziet zo'n virtueel raster eruit?
In wezen is het de rendering zelf, indien u instelt op 800 x 600 pixels dan krijgt u een rendering van 800 x 600 = 480.000 pixels. Met vlakken die één pixel groot zijn. Door elk raster/pixel vlak kunnen meerdere lichtstralen worden gestuurd, de resultante van het ge-



hele proces wordt weergegeven in de helderheid en kleur van dat betreffende pixel. Indien het rastervlak van de camera wordt verwijderd, dan wordt de stralenbundel smaller, waardoor de 3D modellen op het scherm groter worden afgebeeld. En als het rastervlak naar de camera toe wordt ingesteld zal het 3D model juist verder weg worden weergegeven in de rendering.

Indien het 2D rastervlak wordt gedraaid, met wijzigingen van het brandpunt wordt een andere 3D wereld weergegeven.

Bij **Ray Tracing** zien we dus dat de ingestelde resolutie een belangrijk onderdeel uitmaakt van de render**tijd**. Een instelling van 1024 x 768 levert 786.432 pixels op waar meerdere lichtstralen doorkomen, gemiddeld zo'n 3 tot 5 of meer hetgeen al snel tot meer dan een miljoen berekeningen met zich mee brengt bij een relatief lage resolutie.

1024 x 768 px -> 786.432 pixels totaal
1280 x 960 px -> 1.228.800 pixels totaal
2560 x 1920 px -> 4.915.200 pixels totaal
5120 x 3840 px -> 19.660.800 pixels totaal
10240 x 7680 px -> 78.643.200 pixels totaal

Allerlei verfijningen bij het renderen worden tot stand gebracht door extra lichtstralen op het model af te schieten. Zoals bijvoorbeeld bij het bepalen van de schaduwzijden. Anti-Alias, spiegelingen, schaduwgedeelten gedetaileerd weergeven etc. Het gevolg is dat de renderingstijd nog verder toeneemt.

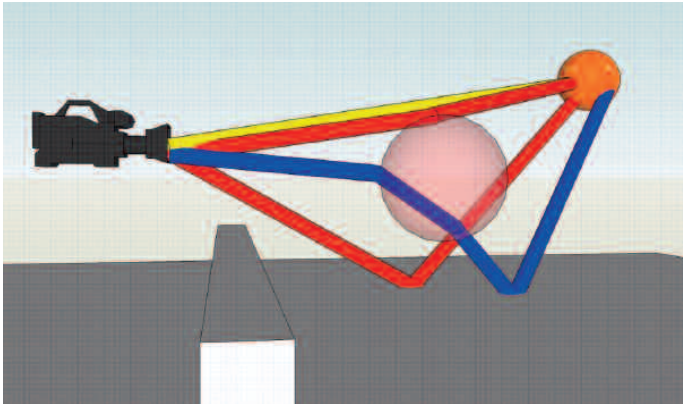
Probeer zelf maar eens Anti-alias aan- of uit te zetten en meet het tijdsverschil.

De gele bundel gaat rechtstreeks van de camera naar de lichtbron / zon. De bovenste rode bundels worden op de drie dimensionale bol gereflecteerd. Ze worden dus door het 3D model (scenery) afgebogen en opnieuw richting lichtbron gestuurd.

De andere rode bundel gaat naar de grondoppervlak, wordt daar gereflecteerd en gaat vervolgens ook naar de lichtbron. De blauwe bundel wordt in de bol afgebogen en vervolgens via de grond ook weer gereflecteerd naar de lichtbron.

We kijken door het raster heen naar het 3D model. Het raster moet de uiteindelijke rendering voorstellen met pixels.

De omgekeerde richting van de kijker naar de objecten en naar de zon toe, houdt in dat de afzwakking van de energie van de Photonen juist wel vanuit de lichtbron moet worden berekend. En dat maakt de toe te passen routines er zeker niet eenvoudiger op.

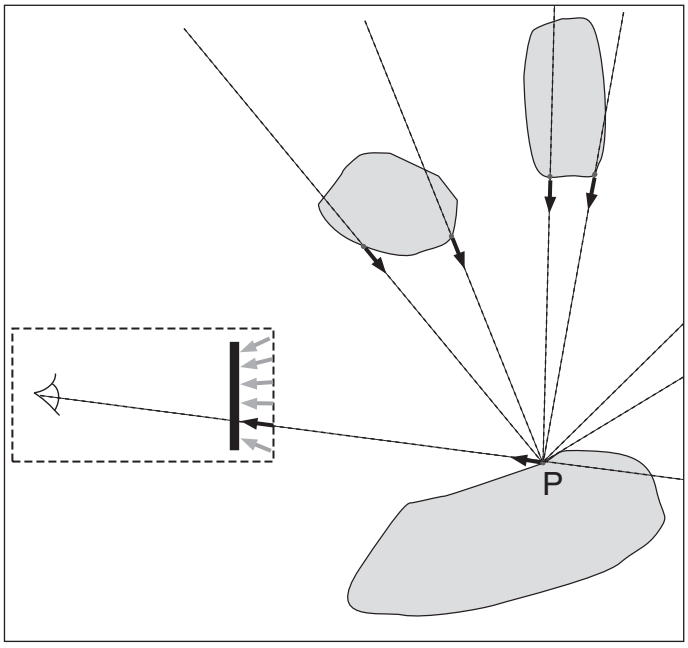


Global Illumination

"GI" is de algemene naam van een groep algoritmes die bij 3D computer graphics worden gebruikt. Ze dienen er voor om een rendering realistischer te maken. Niet alleen het licht van de directe verlichting (rechtstreeks vanaf de lichtbron), maar ook de lichtstralen die van dezelfde lichtbron komen nadat ze gereflecteerd zijn door allerlei oppervlakken in de 3D scene.

Reflecties, refracties en schaduwen zijn allemaal voorbeelden van Global Illumination. Een gebruikte methode is om de Global Illumination te berekenen en tijdelijk op te slaan, samen met de geometrie. Met de opgeslagen informatie kunnen afbeeldingen van verschillende standpunten worden gemaakt, zonder dat bij elk standpunt er opnieuw een lichtberekening moet plaatsvinden. De algemene rendervergelijking [Kajiya 1986] verklaart dat het licht vanuit elk punt op de oppervlakte in een scene afhangt van alle andere punten in dezelfde scene. Dit leidt tot een oneindige lus voor Global Illumination, hetgeen met andere algoritmes in goede (lees kortere) banen moet worden geleid. Met behulp van een simpel 2D model heeft [jarosz12theory-1.pdf,] dat probleem proberen te vereenvoudigen.

Global Illumination maakt gebruik òf kan gebruik maken van allerlei bekende algoritmes zoals Radiosity, Ray Tracing, Beam Tracing, Cone



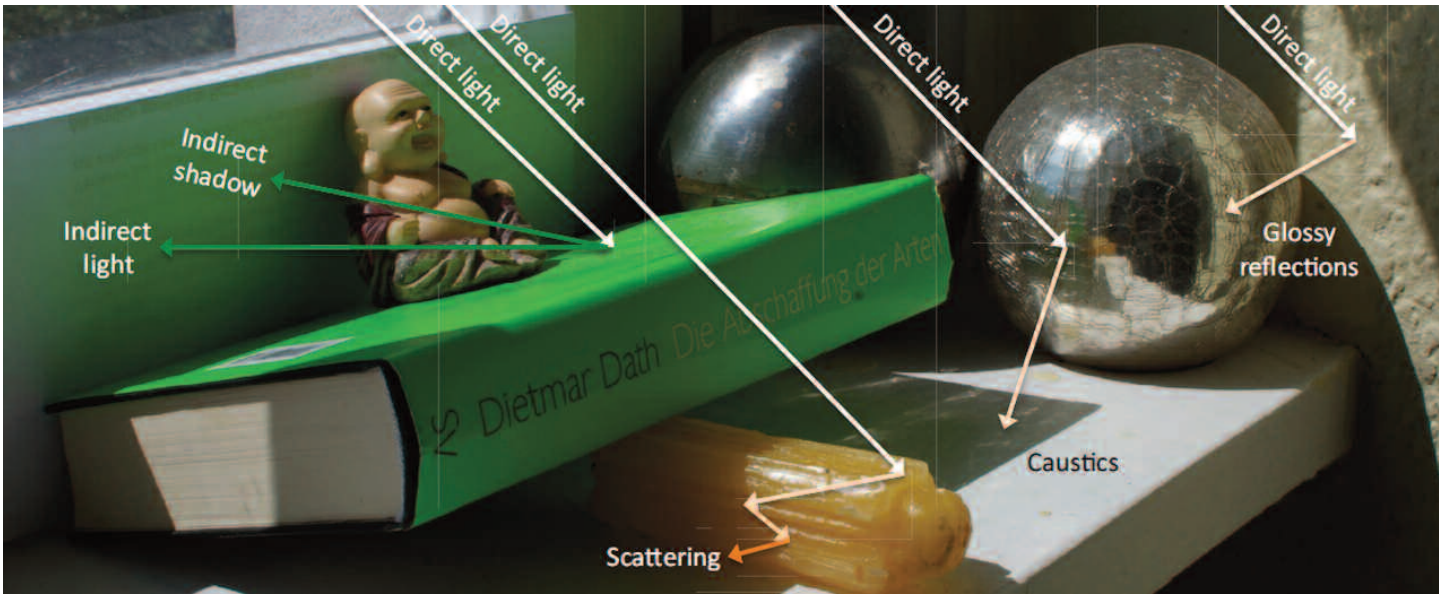
Volgens de algemene render vergelijking is de straling van een punt P afhankelijk van de straling van alle andere punten die zichtbaar in beeld zijn voor dat punt. Hun stralingswaardes op hun beurt, hangen ook weer af van alle punten die zichtbaar zijn voor deze punten, enzovoort. Het herhalende principe leidt al gauw tot bijzonder lange rendertijden. Dit is één van de oorzaken dat 'Global Illumination' niet alleen globaal maar ook ingewikkeld is.

Ontwikkelaars en marketing mensen van renderprogramma's zijn dan ook altijd zeer verguld als ze 'GI' in hùn programma is ondergebracht.

Vrij naar: Figure 2.12 [98-1700.pdf]

Onderstaand de algemene universele renderformule:

$$L_\lambda(\mathbf{x},\omega',\lambda) = L_{\text{emit}}(\mathbf{x},\omega',\lambda) + \int_{\Omega_\odot} f_s(\mathbf{x},\omega,\omega',\lambda)\, L_\lambda(G(\mathbf{x},\omega),-\omega,\lambda)\, |\omega\cdot\hat{\mathbf{n}}|\, d\omega \qquad (2.28)$$



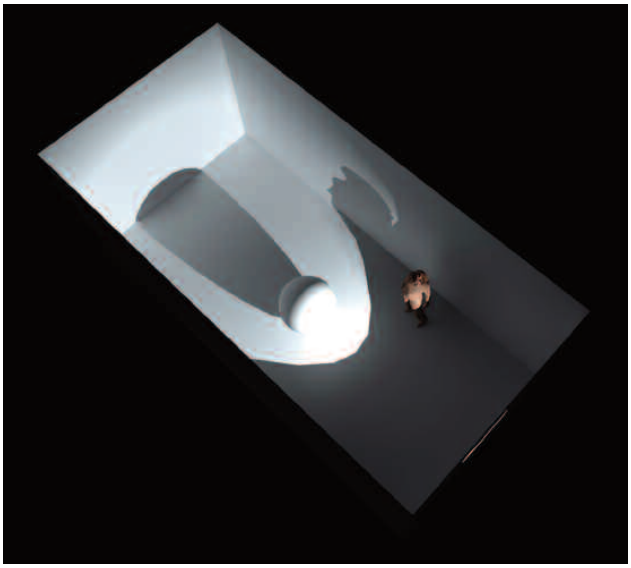
Meerdere difuuse en spiegelende reflecties, caustic en verstrooïend met Global Illumination
Literatuur: **GISTAR_CGF12.pdf**
The State of Art in Interactive Global Illumination (1981)

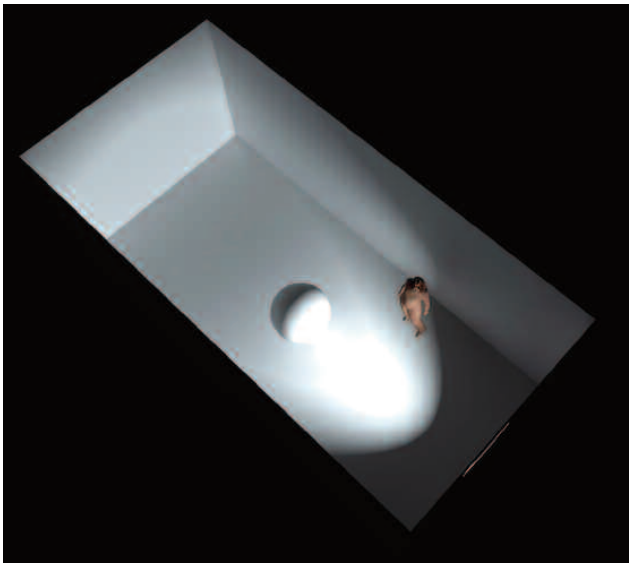
Tracing, Path Tracing, Metropolis Light Transport, Ambient Occlusion, Photon Mapping plus Image based Lighting.

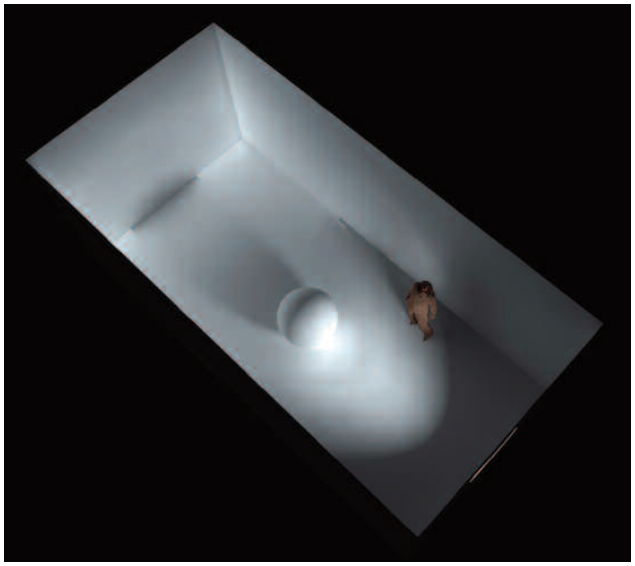
In de natuur, die we maar al te graag willen nabootsen is het zo, dat licht vanaf elk oppervlak kan worden gereflecteerd. Hoe helderder een oppervalk (hoe witter het oppervlak) des te meer licht er zal worden gereflecteerd.

In de werkelijkheid en dus ook bij het maken van een rendering kunnen we er dus van uitgaan dat elk oppervlak en elke lichtbron bijdraagt aan het totale plaatje. Maar oneindig lang durende algoritmes met alle goede bedoelingen van dien ("*fysisch correct renderen*") is niet waar we op zitten te wachten, *het moet er goed en perfect uitzien met een aantal kleine beperkingen die zo min mogelijk zichtbaar moeten zijn*. Dat is het praktische doel.

Het principe van **Global Illumination** wordt ook door fotografen gebruikt door grote witte schermen in de fotoscene op te stellen. Bij mode foto's zien we het gebruik van deze schermen veel terugkomen. Of bij het maken van films waarbij de film crew grote gekleurde reflectoren buiten beeld vasthouden om de verlichting van de acteurs te verbeteren.





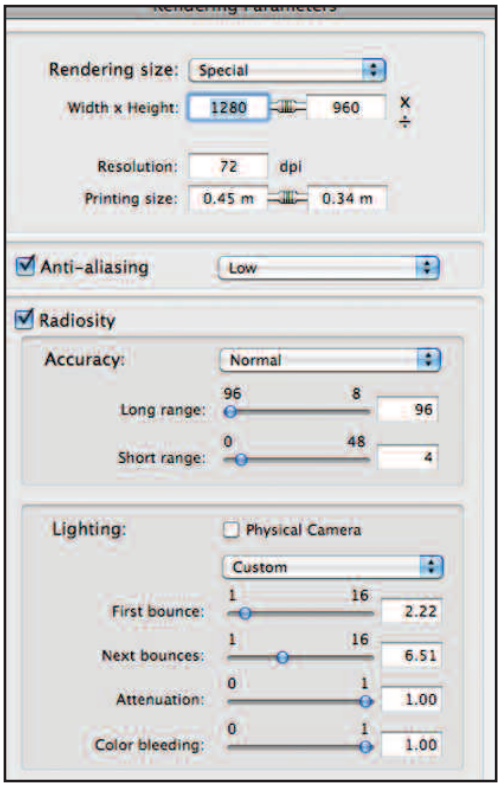


Radiosity

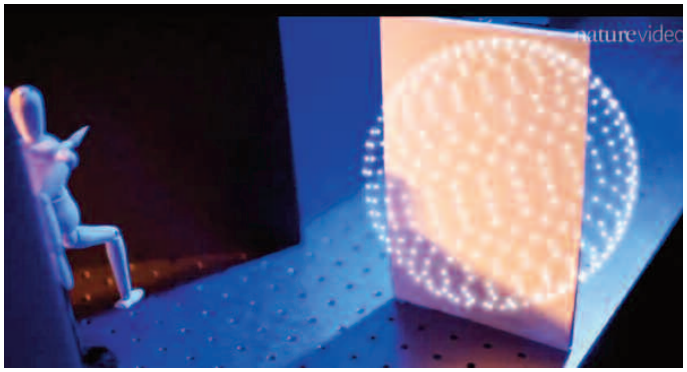
Naast Ray Tracing is er nog een belangrijk thema bij renderingsprogramma's: Radiosity (1984) met *"Modeling the Interaction of Light Between Diffuse Surfaces"*. Een aantal Renderingsprogramma's hebben deze optie niet en sommigen zeggen dat ze de functie hebben, maar tonen dat niet in de rendering. Radiosity is een immitatie van indirecte verlichting van 3D objecten. Het is een algoritme om 'Global Illumination' (difuus licht) te laten inwerken op objecten en de omgeving.

Bij 'gewone' Ray tracing wordt het licht bekeken dat van de lichtbron(nen) gaat naar de objecten. Delen die niet belicht worden en dus in de schaduw liggen, worden daarbij niet gedetailleerd weergegeven. Ze worden in werkelijkheid wel degelijk belicht door de omliggende omgeving en daar komt Radiosity om de hoek kijken.

Lichtbronnen geven Photonen af, die van object naar object worden gereflecteerd. Indien ze op een gekleurd object komen, reflecteren ze een gedeelte van die kleur op de ondergrond terug. Maar tijdens dat proces wordt elke keer energie verloren, hetgeen aan de verminderde helderheid ten opzichte van de afgelegde afstand is te zien. De energie wordt gedeeltelijk door het oppervlak opgenomen, maar de energie wordt ook verdeeld door verstrooiing van de Photonen op het oppervlak, zie afbeelding rechts boven.



Expert menu Artlantis met Radiosity aangevinkt.



Fantastische video hoe Photons na reflectie met een oppervlak in allerlei richtingen worden afgebogen (soort Radiosity). Alsof het een nieuwe puntlichtbron betreft. Een aantal lichtstralen zal nooit meer in beeld (rendering) komen en voorgoed verdwijnen. Bron: ***Femto fotografie MIT Media Lab met Prof. Ramesh Raskar*** *e.a..*

Schaal is belangrijk bij Radiosity
Een renderingsprogramma dient dit een fysisch model zo goed mogelijk na te bootsen, waarbij de 1:1 schaal van het 3D model wel degelijk uitmaakt in de weergave.

Radiosity in de praktijk bij Renderingsprogramma Artlantis

Een voorbeeld van Radiosity instellingen in renderingsprogramma Artlantis (R + S).

Klik bij *Preferences* om het Expert Display zichtbaar te maken.
Expert Mode Viewpoint Rendering Parameters in the "Photorealistic" Engine. De rendertijd hangt af van de gekozen instellingen die hieronder zijn gekozen.

Instellingen menu

Radiosity: aangevinkt
Accuracy: Normal - Average - High - Custom
Lighting: Physical Camera wel of niet aangevinkt, indien aangevinkt dan Physical Camera (uitgebeten te heldere rendering) en niet aangevinkt dan volgt Auto Power verlichting.
De instellingen hebben een behoorlijke invloed op de renderingstijd.
Lighting: Interior / Exterior / Custom

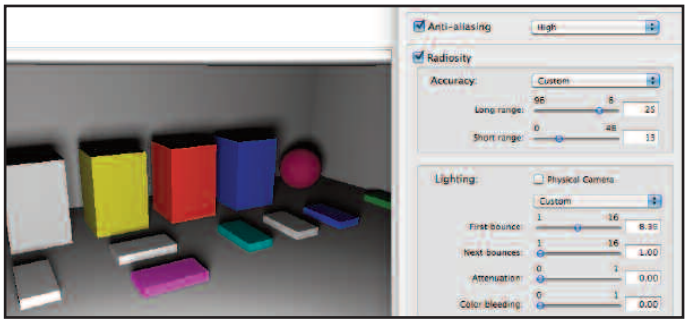
Long range
Waardes tussen de 96 en 8 pixels.
De Radiosity wordt over een klein aantal pixels berekend, de andere worden geextrapoleerd. De afstand wordt bepaald door de gemiddelde afstand tussen twee punten waar Radiosity wordt berekend. Hoe kleiner deze afstand, des te meer de densiteit van de pixels toeneemt.

Short range
Waardes tussen de 0 en 48 pixels.
Dit is de Radiosity dichtbij oppervlakken (bv. een hoek tussen 2 wanden). Deze hebben afzonderlijke bewerkingen nodig, waardoor de schaduw kwaliteit kan worden verbeterd.
Indien 0 wordt ingesteld, dan wordt geen berekening uitgevoerd. Hoe hoger de ingestelde waarde des te effectiever de schaduwen zullen worden berekend die met Radiosity verband houden. **Afbeeldingen met een aantal praktijk voorbeelden.**

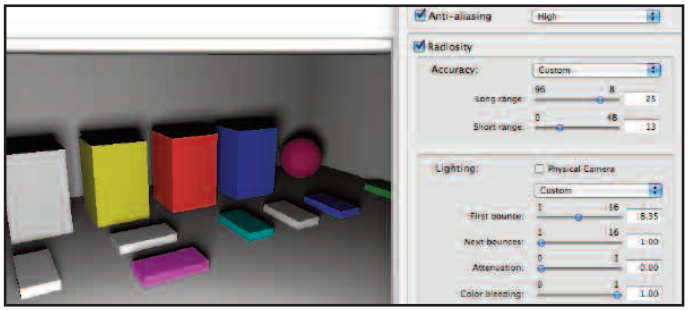
Lighting
Interior / Exterior / Custom
Hiermede wordt de algemene difuuse verlichting ingesteld. Bij de Interior en Exterior instellingen wordt standaard instellingen gekozen. Custom wordt gebruikt als de gebruiker de waarden die er stonden zelf wijzigt. Of als een bestand van een oudere Artlantis 3 versie wordt gebruikt.

Lighting Power
First Bounce
Waardes van 1 tot 16
Hiermede wordt de kracht van de eerste reflectie ingesteld. Alle oppervlakken die direct licht van de Heliodon of van de lichtbronnen ontvangen worden hierdoor beïnvloed.
Het is de toonwaarde van de reflecties van de oppervlakken die licht ontvangen. Een hogere waarde (max. 16) levert een groter contrast op tussen licht en schaduw. De lichte partijen en de donkere partijen zijn min of meer zonder detail. Door het instellen van de tweede "Next Bounces" kan daar verandering in worden gebracht.

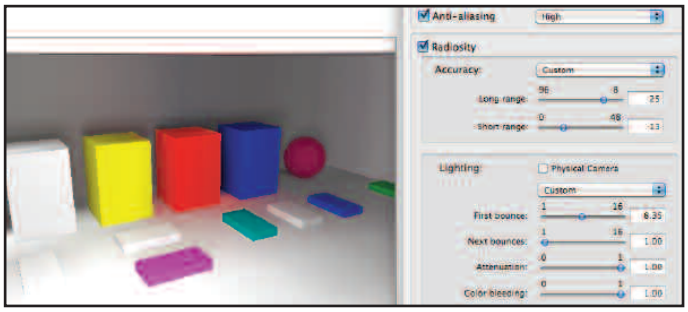
Next bounces
Waardes van 1 tot 16
Hiermede wordt de Radiosity kracht van alle meerdere reflecties na de eerste geregeld. De oppervlakken die indirect licht ontvangen worden hier door beïnvloed. Door het verhogen van de



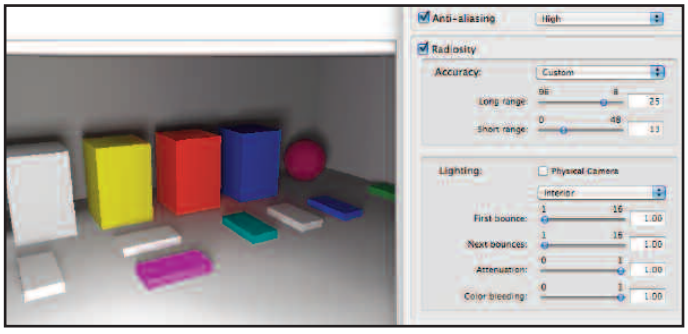
Accuracy Custom, Long Range = 25, short = 13.
First bounce = 8 en next bounces =1. Attenuation = 0 en Color Bleeding = 0.



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 8 en next bounces =1. Attenuation = 0 en Color Bleeding = 1.



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 8 en next bounces =1. Attenuation = 1 en Color Bleeding = 1.



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 1 en next bounces =1. Attenuation = 1 en Color Bleeding = 1.

waarde wordt het beeld helderder, speciaal bij interieur scenes.
Nadat de eerste reflectie geweest is volgen de andere indien dat wordt ingesteld. Een hogere instelling (max. 16) levert meer energie op bij de oppervlakken, waardoor ze meer licht weergeven van de lichtbron. Ook de reflectie op de oppervlakken wordt daardoor beter zichtbaar.
Betere detailering van donkere partijen.

Attenuation
Waardes tussen 0 en 1.
Hiermee wordt de absorptie van het licht na een of meerdere oppervlakken reflecties geregeld. Een lage waarde verhoogt het contrast van de scene, waardoor relatief sterkere schaduwen zichtbaar worden in bv. een interieur scene die met indirecte verlichting wordt verlicht.

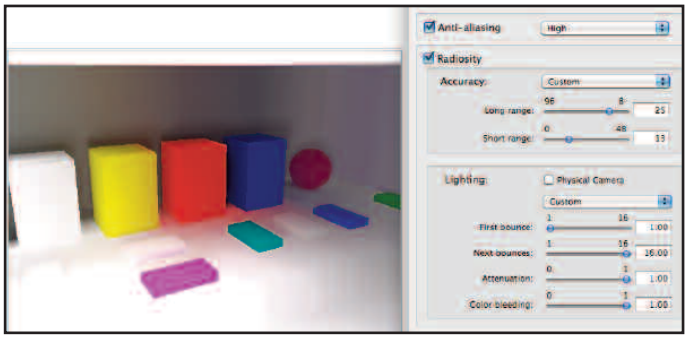
Een getal van 1.00 (maximaal) betekent dat de energie wordt behouden (niet verzwakt op de weg er naar toe) terwijl de lichtstraal naar de oppervlakte gaat. Een getal van 0.50 betekent een halvering van de energie voor elke reflectie. Daarmee wordt ook het contrast van de Radiosity schaduwpartijen beïnvloed. Ook de normaal verlichte oppervlakken wijzigen.

Color Bleeding
Waardes van 0 tot 1.
Stel hier de uitwisseling in van kleuren tussen de oppervlakken onderling. Met kleine waardes worden de kleuren vlakker (minder verzadigd).

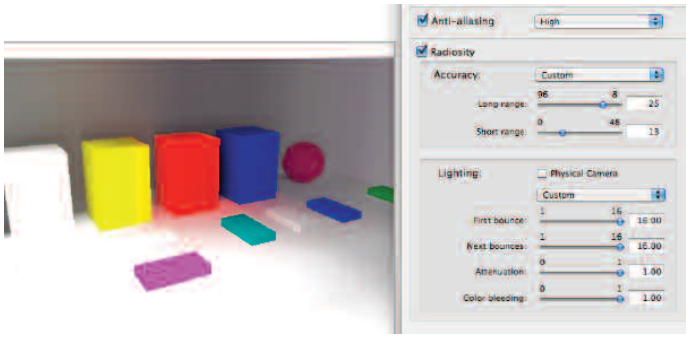
Regelt hoeveel kleur van oppervlak naar een oppervlak wordt doorgegeven. Een lage waarde levert een lage verzadiging (vervlakking) op voor de Radiosity berekening.

Meer reflecties instellen vraagt uiteraard meer lichtstralen om uit te rekenen, vandaar dat de renderingstijd toeneemt. De gereedschappen zijn er in dit renderingsprogramma, waarbij kennis van de renderingsprincipes bijdraagt aan de optimale keuze van de instelling en het resultaat.

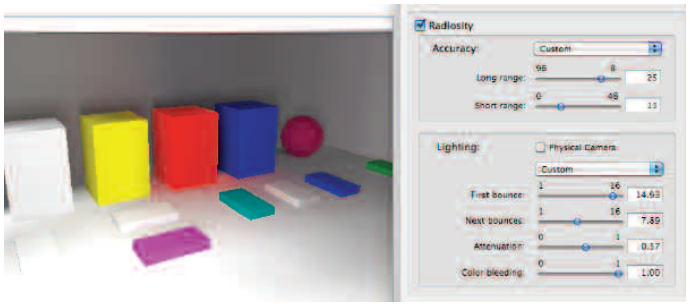
Bij **Maxwell Render** wordt op de website verteld: "Cameras in Maxwell Render™ operate completely different from those in other render engines. Traditionally, most render engines use a pinhole camera. This type of camera simulates a tiny hole



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 1 en next bounces =16. Attenuation = 1 en Color Bleeding = 1.



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 16 en next bounces =16. Attenuation = 1 en Color Bleeding = 1.



Accuracy Custom, Long Range = 25, short = 13.
First bounce = 14 en next bounces = 8. Attenuation = 0.60 en Color Bleeding = 1.00.

that allows light rays coming from the scene to reach the viewing surface. Instead, Maxwell Render™ simulates a real camera with the associated lens set, diaphragm aperture, diaphragm blades and various other settings. By using this type of camera model Maxwell Render™ can automatically simulate depth of field or aperture diffraction."

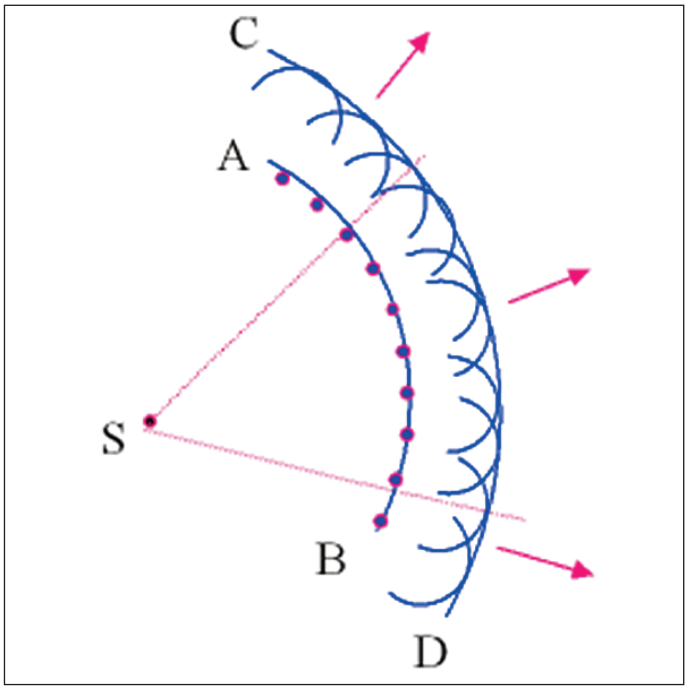
Alhoewel een groot deel daarvan op waarheid berust, dient bovenstaand citaat toch als een Publiciteitsinstrument te worden beschouwd. Waarbij men uit de weg gaat dat medio januari 2013 er nog steeds geen GPU / OpenCL uitvoering (hybride) van Maxwell is en alles met de CPU moet worden uitgerekend.
Waarbij de renderingstijden fors langer zijn, dan van alle andere renderprogramma's bijelkaar.

Mind teaser

Indien er een lichtdeeltje Photon bestaat en wetenschappers twijfelen daar niet aan, hoe is het dan mogelijk dat deze zo snel gaat?

Of te wel we hebben een lichtbron, bv. een LED die we met een schakelaar aandoen.
De LED straalt vrijwel direct Photonen uit die van de ene op de andere miljoenste van een seconde van 0 naar 300.000 km/sec snelheid worden gebracht. Deeltjes in de natuur in een zo'n korte tijd zoveel versnellen is in wezen niet mogelijk. Maar bij Photonen gaat het wel. Hoe zit dat?
Een Photon kan dus geen restmassa hebben anders zou deze eerst 'op gang moeten komen'.
E = h . nu En een Photon behoeft ook niet eerst te acceleren om op deze snelheid te komen.
EM golven behoeven ook niet op snelheid te komen en de Photon valt mede onder deze groep.

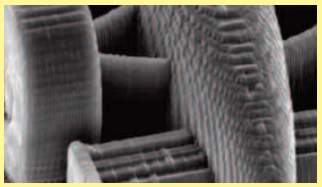
De naam Photon
Stamt uit 1926, bedacht door Gilbert Lewis. Een Photon heeft zowel de eigenschappen van een golfvorm als van een deeltje. Ondanks dat het mede een deeltje betreft (wave-particle duality) kan toch over frequentie, golflengte, amplitude en andere eigenschappen worden gesproken.

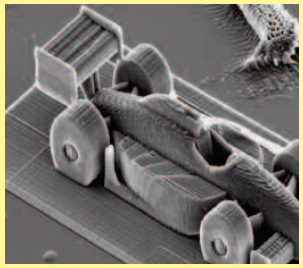


Huygen's Principe (1690)
Elk golffront is een samenstel van golven.
Elk punt op het golffront werkt als een onafhankelijke nieuwe bron voor de volgende. AB en CD zijn twee golffronten.
Huygens beschrijft de golfvoortplanting en de lijn loodrecht op het golffront geeft de voortplantingsrichting aan. Als een puntlicht oneindig ver weg staat is het golffront vlak.

Java Applets over diverse optische principes:
http://www.phy.ntnu.edu.tw/ntnujava/index.php?board=6.0

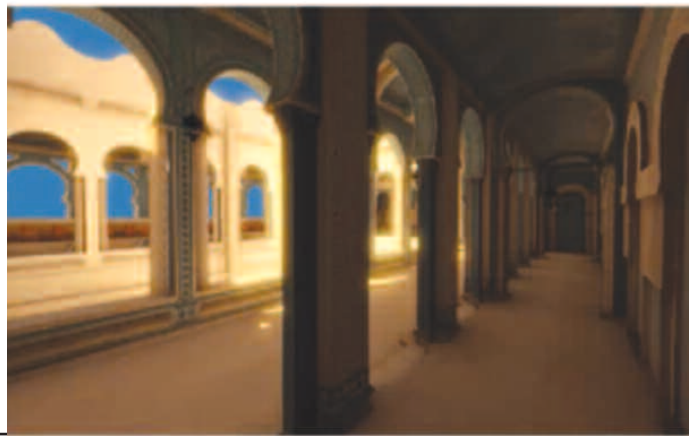
Photon's zijn elektrisch neutraal en ze zijn gelijk aan hun antiphoton. Gepolariseerd licht komt van de manier waarop er ofwel een linkerhand of rechterhand photon is (parallel aan de uitzendrichting). De Photonen die de zon uitzendt doen er ongeveer 8 minuten door het vacuum (299.792.458 m/s) over, om de aarde te bereiken.
Maart 2012. Onderzoekers werken overal in de wereld met 3D printers. De Universiteit van Wenen heeft een doorbraak voor 3D printing techniek bereikt. Het is nu mogelijk om 3 dimensionele objecten te printen die extreem fijn zijn (Two-photon lithography).
De auto is slechts 285 micron lang.
0,000285 mm
285 10^{-6} meter

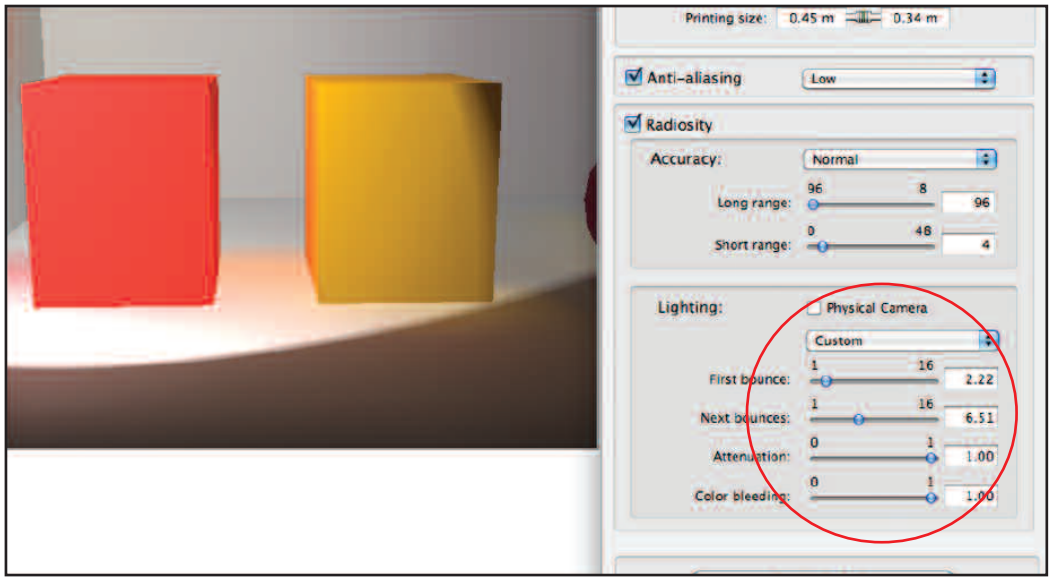




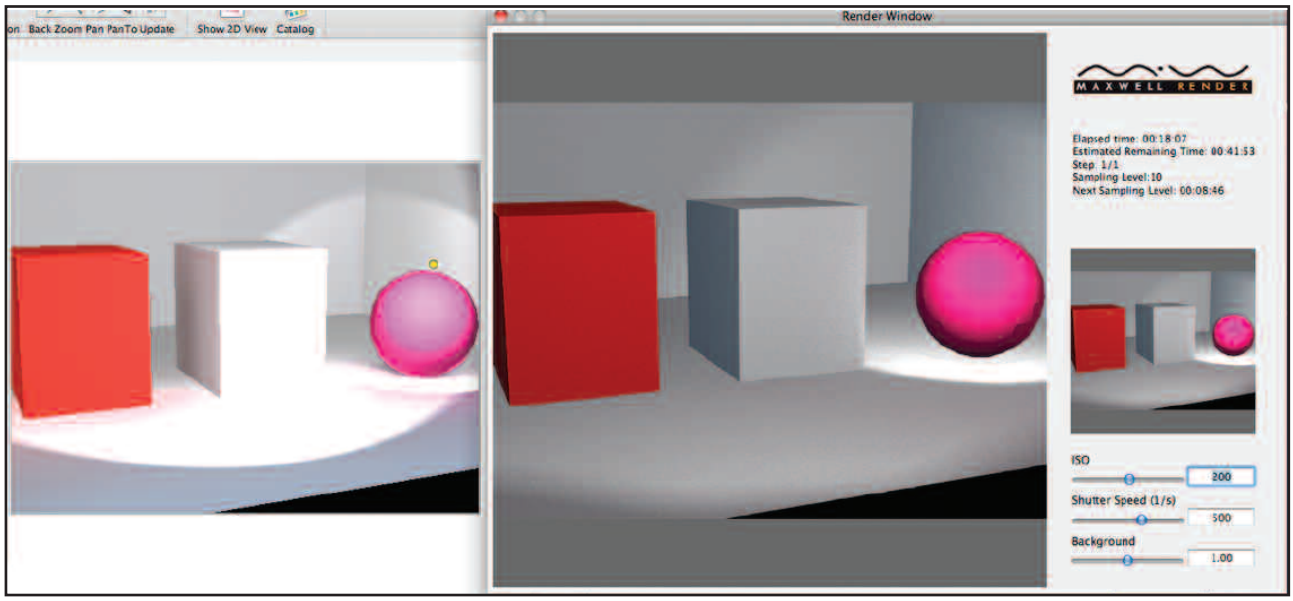
Zonder Radiosity
Directe verlichting + globaal omgevingslicht

Radiosity
Meer diepte
Schaduwen ingevuld en gekleurd
Betere verloop in schaduwen
Model wordt met 2 lichtbronnen verlicht: zon en kunstlicht





Simpel 3D model in kamer. First Bounce en Next bounces te regelen in Artlantis Render programma.
Let op de kleuring van de witte oppervlakken (Radiosity).
Onder: zelfde model nu in Maxwell Engine (SL = 10) weergegeven, zonder enige Radiosity in beeld, maar wel met een beter tintverloop.



Radiosity

Met het onderzoeksrapport over Radiosity (1984) *"Modeling the Interaction of Light Between Diffuse Surfaces"* werd het tweede belangrijke algoritme van die tijd uitgevonden. Het zou één van de meest gebruikte methoden worden om 3d scenes te verlichten. Door de vele aanvullende en overlappende systemen, die daarna werden ontwikkeld vormt het een complex terrein. Radiosity is een algemeen verlichtings algoritme dat een nauwkeuriger onderlinge relatie van licht met de aanwezige objecten mogelijk maakt.

Links directe verlichting, rechts Radiosity. Voorbeeld van Wikepedia.

Radiosity zorgt er voor dat een 3D scene opgesplitst wordt in geometrie met een aantal sub patches. Met behulp van lineaire vergelijkingen worden de lichtstralen berekend vanaf de lichtbron naar deze opgeslagen patches. Radiosity lost slechts een deel op van de GI (Global Illumination) vergelijking, netzoals Ray Tracing ook slechts voor een deel doet.

LightWave vanaf versie 9.6 (inmiddels vers. 11) is zo'n renderprograma dat goed gebruik maakt van Radiosity. Het is op CPU gebaseerd met een modeller die in 4 aanzichten zichtbaar is te maken. De computereisen zijn bijzonder laag. Ze passen twee en een half verschillende methoden voor Radiosity berekeningen toe. De eerste is de bekende Monte Carlo, de tweede Final Gather en de halve is de 'Background Radiosity'. Die laatste is een speciale nieuwe variant op de bestaande Monte Carlo. Elk van deze methoden kan worden aangevinkt en worden ingesteld. Enkele menunamen van Radiosity opties: interpolated, blur background, use transparency, volumetric Radiosity, Ambient Occlusion, Directional Rays, use Gradients, use Behind Test en Use Bumps. Ook Indirect Bounces en Secondary Bounce Rays en nog een aantal instellingen maken met mogelijk precies die sfeer aan de rendering mee te geven die gewenst wordt.

https://www.lightwave3d.com/

Nieuw bij LightWave is dat Octane Render zal worden ingepast in het programma. Met een compleet nieuwe set textures en materialen. Aangezien de bestaande materialen van LW niet kunnen worden gebruikt met GPU.

Renderingen met Radiosity gemaakt (als het render programma deze optie bezit) zijn natuurlijker dan bv. met Raytracing of Scanline renderen. Een behoorlijk nadeel van Radiosity is dat het renderen aanzienlijk langer duurt.

Radiosity als warmtebron
Stelt u zich voor een straalkachel die rondom in een ruimte met enkele objecten warmte uitstraalt.
De warmtestraling is in dit geval een vervanging van de lichtstralen. Overal meten we (ook achter een object of er vlak naast, of op de grond) met een thermometer de locale temperatuur. Voorwerpen die de warmtestraling reflecteren of juist vasthouden maken het complex.

Om de berekeningen en het render resultaat niet te statisch te maken is de **Monte-Carlo** methode ontwikkeld, deze wordt doorlopen met telkens andere startcondities, waardoor er een betere verdeling ontstaat voor het eindresultaat de rendering. Oneerbiedig een soort dobbelmethode genoemd. De Monte-Carlo methode is één van de langzaamste, maar wel nauwkeurigste methodes. FG (Final Gather) gaat sneller, maar is minder nauwkeurig. De richting van de stralen wordt niet opgenomen in de berekening, alle oppervlakken stralen op een difuse manier licht uit.

Met vensters waar daglicht doorschijnt wordt het een moeilijk te beheersen systeem. Vandaar dat veel renderprogramma's een instelknop hebben voor binnenshuis / buitenshuis renderingen. Backdrop Only zorgt voor de verlichting van oppervlakken voor snelle buitenshuis renderingen. De strakke afbaking van de schaduwpartijen wor-

den bij Radiosity zacht en aangenaam. Het is alsof met een lichtstraal de diameter van het puntlicht telkens van plaats wisselt, waardoor een meer geleidelijke overgang van verlicht naar schaduw wordt bereikt. Iets dat de zon, door zijn relatief grote diameter ook standaard al doet.

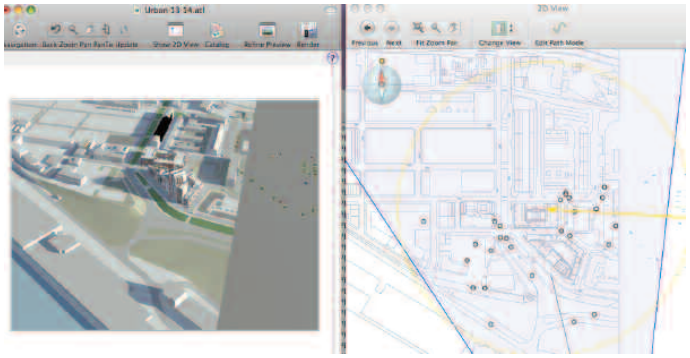
Het licht dat in een ruimte reflecteert op de aanwezige objecten, zorgen voor een soort kleurbloeding bij de aanrakingsoppervlakken. Een deel van de kleur van bv. de wanden worden door de aanwezige objecten als het ware verzwakt teruggegeven. Het rekenproces dient meerdere keren te worden herhaald. Ook objecten onderling kunnen dit effect geven.

Door de relatief lange rekentijden zijn er in de loop van de tijd veel oplossingen bedacht, die zo dicht mogelijk bij het oorspronkelijke idee moeten blijven, maar met kortere renderingstijd.

Daar begint het dilema voor de programmeur / wiskundige. De klanten van een programma vragen om een perfecte rendering in zo kort mogelijke tijd. Perfect en korte tijd staan daarbij vaak haaks op elkaar. Het is de taak van de programmeur / wiskundige om deze duidelijke tegenstrijdigheid zo goed mogelijk op te lossen. Daarbij zal automatisch aan het 'fysisch correct render resultaat' worden gesleuteld. Percentages goed of minder goed zijn daarbij niet te geven.

Dat wil echter niet zeggen dat het renderprogramma dat er het langst over doet ook meteen kan worden uitgeroepen tot het meest 'eerlijke' programma dat er is. Die conclusie wordt al gauw gemaakt en ligt ook voor de hand.

De mogelijkheden van nieuwe intelligente algoritmes zijn zo interessant met individuele curves, om de werkelijkheid zo goed mogelijk te benaderen, dat de recht-toe-recht-aan systemen in de schaduw worden gesteld. Bij Radiosity (zie '*A Global Illumination Implementation.pdf*' blz. 6) is het zo dat gemiddeld 90% van de kwaliteit in de eerste 10% van de Radiosity berekeningen wordt bereikt. Het zou dus zeer aantrekkelijk zijn om na die 90% de berekening op een nette manier te beëindigen. Alleen de manier waarop dat zou moeten gebeuren zal met allerlei handmatige instellingen door de gebruiker zelf moeten worden geregeld. Dat blijkt niet altijd even vriendelijk



Preview en 2D view van Artlantis Render. Een renderprogramma met Radiosity.

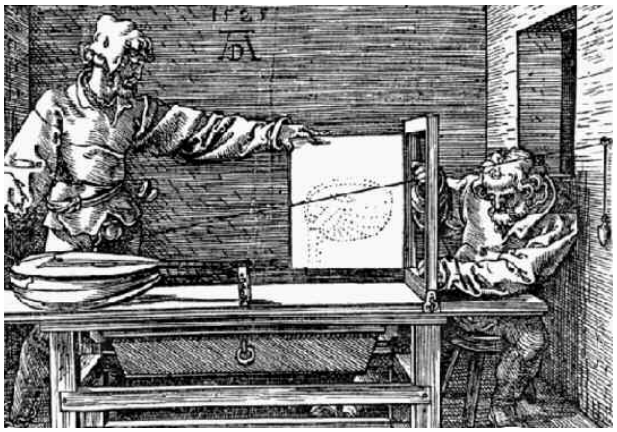
voor de gebruiker van het renderingsprogramma, een automatische regeling is vanwege het complexe karakter niet goed mogelijk.

Zelfs het "*Bi-Directional Path Tracing as a new Monte Carlo Algortihm for physically-based rendering*" kan niet hard worden gemaakt door de ontwikkelaars aangezien Monte Carlo juist uitgaat van het willekeurig 'vergeten te berekenen' aantal uit te voeren lichtstralen / vectoren. Wanneer wordt er iets vergeten en onder welke omstandigheid wordt dat gedaan? Met een universele Bel curve als richtsnoer? Is het dan nog fyisch correct?

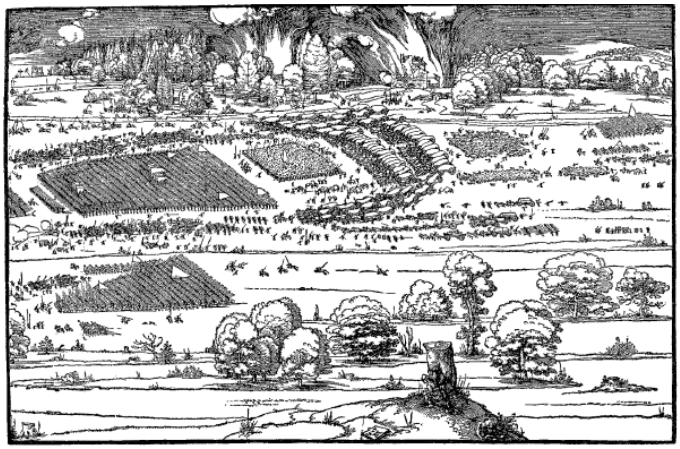
Real Time rendering aangewakkerd door de Games- en film industrie zal in de toekomst voor nieuwe algoritmes gaan zorgen die zeker niet fysisch correct zijn of in de buurt komen, maar die wel in een renderprogramma voor architecten kan worden aangepast om bijzonder fraaie renderingen in korte tijd te maken.
Of te wel "*What are the next big step forward for content production?*
** Quick iteration times = quality*" uit de presentatie van Johan Andersson, Siggraph 2012. Daarbij wordt het accent verlegd naar schaalbaarheid vanaf iPhone met 1 W tot 300 Watt (300 x) voor computers met resoluties van 320 x 480 en 5760 x 1200 (45 x schaal) of hoger.
Of met "Assessment of five radiosity acceleration techniques" die te koop is als PDF.
"*A comparison of four visibility acceleration techniques for radiosity*" uit de 'The Visual Computer'. En vele andere publicaties.

Een aantal van Glassner's boeken
Richtprijs Bol.com sept. 2012 voor BTW verhoging

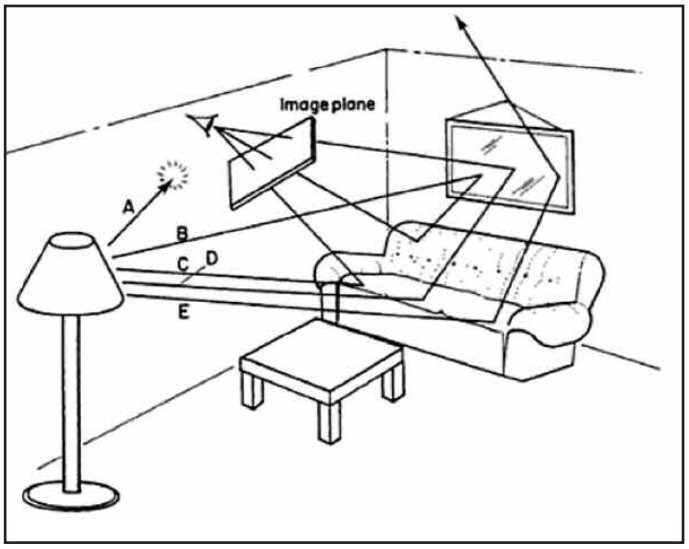
Morphs, Mallards and Montages	52,-
3D Computer Graphics	22,-
An Introduction To Ray Tracing	100,-
Interactive Storytelling	45,-
Processing for Visual Artists	61,-
Andrew Glassner's other Notebook	43,-



Albrecht Dürer, 1525, [1471 - 1528], kunstschilder die houtsneden en kopergravures maakte. Mechanische werking van het maken van een perspectief tekening.
Vervang eenvoudig het touw met contragewicht en het tekenvel door een imaginair venster en het is een renderingsopzet geworden.

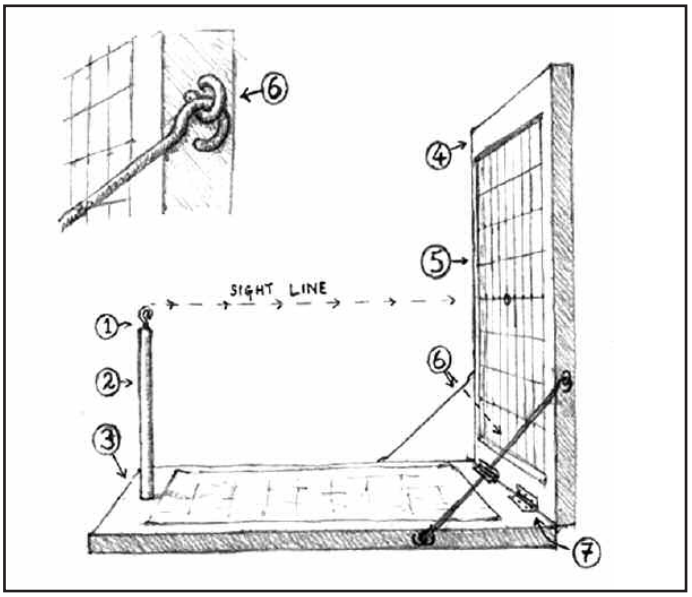


Dürers houtsnede van een landschap met legereenheden.



Afbeelding uit **Ray Tracing** 1989 van Andrew S. Glassner. Enige lichtstralen zoals A en E bereiken het uiteindelijk gewenste venster voor het maken van de rendering nooit. Andere lichtstralen volgen eenvoudige of gecompliceerde routes om wèl op het zichtvenster (image plane) uit te komen.

Het maken van een portret door een kunstenaar die gebruik maakt van een perspectief apparaat om op glas te tekenen. Uit 'Course in the Art of Drawing' by Albrecht Durer, published Nuremberg 1525.



Programmeren

Ray Tracing

“Het renderen van 3D modellen heeft zijn weg gevonden naar miljoenen bruikers”

De kracht van **Ray Tracing** is dat er slechts een beperkt aantal programmaregels nodig zijn om de basis principes te laten werken. Samen met een simpele interface en enkele 3D modellen ingebakken in het programma komen we dan aan ca. 200 programma regels. Met de uitbreiding van de mogelijkheden en een aantal versnellers plus een uitgebreide luxe interface en import filters wordt dat aantal gemakkelijk vertienvoudigd.
Het kunnen ingeven van meerdere lichtbronnen, transparantie, afzwakking van de lichtkegel, instellen van de openingshoek etc.

Ray Tracing versus Ray Casting

Ten opzichte van Ray Casting wordt met Ray Tracing het natuurlijke en fysisch correcte schijnsel van licht als het wegschieten van lichtdeeltjes (Photonen) bedoeld. De meest gebruikte techniek is die van Path Tracing, Bi-directional Path Tracing of Metropolis light transport. Naar de uitvinder Whitted genoemd is de *Whitted Style Ray Tracing* methode en allerlei afgeleiden daarvan. Bij de meeste algoritmes wordt licht als een vector gezien met een beginpunt, een lengte en richting in het 3D model.
Helaas is het niet mogelijk om met één straal te volstaan. Er worden meerdere berekeningen van stralen op het 3d model los gelaten om uiteindelijk zo goed mogelijk een pixel van de uiteindelijke totale renderingsafbeelding te bepalen.

Het principe van de hoek van inval is de hoek van uitval op een oppervlak, wordt meegenomen plus die van meerdere reflecties van verschillende objecten in het model. Reflectie is redelijk eenvoudig, het wordt pas echt lastig als een object uit een glazen voorwerp bestaat waarbij de lichtstralen in het voorwerp zelf dringen en daar met een andere snelheid (hoeken) hun weg vervolgen naar buiten. Indien de weg van de straal kort is kan ook een lichtbron als object worden bereikt. Maar het kan ook afhankelijk van de gekozen instellingen op een gegeven moment worden gestopt (beperking van het aantal weerkaatsingen in het model) of door een ingestelde tijdslimiet. Is zo'n grens bereikt, dan wordt het benodigde pixel in de renderingsafbeelding berekend.

Pseudocode voor het eerste Ray Tracing principe
U kunt deze code omwerken voor bv. C++ of het op Java gebaseerde Processing dat een Open Source programma is, onafhankelijk van het platform.

```
for (int j = 0;
     j < beeldHoogte; ++j) {
   for (int i = 0;
        i < beeldBreedte; ++i) {
        // bereken de richting van de eerste primary lichtstraal
        berekenPrimStraal (i, j, &primStraal);
        // richt de eerste lichtstraal naar de 3D scene en ga op zoek naar het trefpunt van
een oppervlak
Point pTref;
Normal nTref;
float minAfstand = oneindig;
Object object = NULL;

for (int k = 0; k < objects.grootte(); ++k) {
     if (Trefpunt(objects[k], primStraal, &pTref, &nTref)) {
        float afstand = Afstand (oogPositie, pTref);
        if (afstand < minAfstand) { object = objects[i];
           minAfstand = afstand;  // update min distance {
         }
        }
    }
// als object 0 is dan is het een schaduw straal;
if (object != NULL) {
   schaduwStraal.richting = lichtPositie - pTref;
    bool Schaduw = false;
   for (int k = 0; k < objects.size(); ++k) {
        if (Intersect (objects[k], schaduwStraal)) {
           ligtInSchaduw = true; break;
         }
      }
   }
   if   (! ligtInSchaduw) // niet in de schaduw dan pixel inkleuren
        pixels[i] [j] = object -> color * licht.helderheid;
     else
        pixels[i] [j] = 0;
   }
 }
```

De kracht en eenvoud van een Ray Tracing programma zoals hierboven blijkt duidelijk uit het gering aantal programmaregels. Samen met een bescheiden interface en een extra 3D bibliotheek voor de scene komen we aan enkele honderden code-regels om de meest simpele 3D modellen zelf te kunnen renderen.

In de **Programma Render PDF** uitgave met codes treft u meer programma voorbeelden aan.

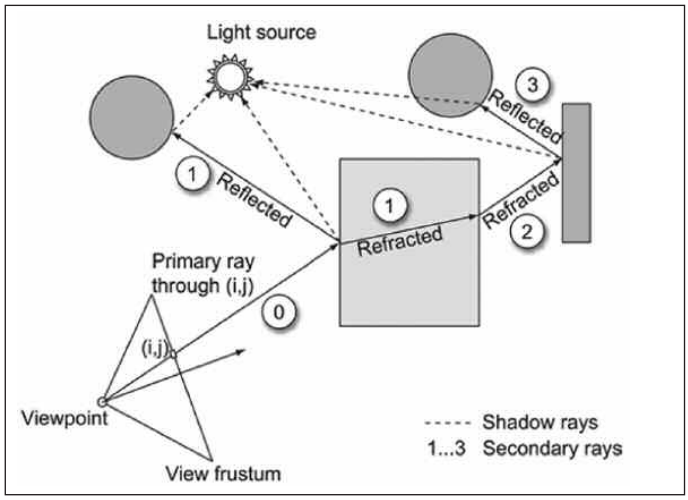
Distribution Ray Tracing
Met deze afgeleide techniek worden vanaf elk trefpunt meerdere stralen afgevuurd.

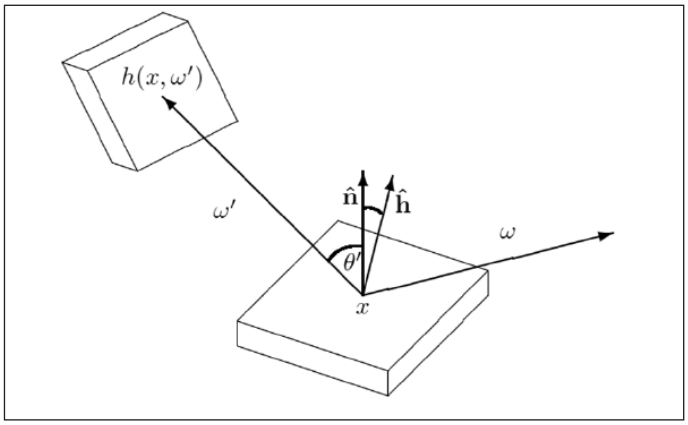
Path Tracing
Hierbij wordt een enkele straal vanaf elk trefpunt losgelaten waarbij vaak met een keuze van middeling gewerkt volgens de Monte Carlo methode. De verlichting resp. inkleuring van de pixels is afhankelijk van de berekening van de integraal van de benadering van de richtingen rond de gekozen Normaal.

Bij Games of animatie films is Ray Tracing vaak te langzaam om met een hoge Frame snelheid te kunnen werken. Maar ook hier komen nieuwe ontwikkelingen aan, die een groter gebruik van Ray Tracing mogelijk maken zowel wat kwaliteit, maar ook wat kortere renderingstijd betreft. Blue Moon Rendering Tools bezit een Ray Tracer die voor meerdere films is ingezet: A bug's Life, Stuart Little, Hollow Man, Swordfish . . .

Path tracing wordt ook wel 'unbiased' genoemd omdat het dicht bij de fysische- en wiskundige renderingsvergelijking staat. Andere renderingsmethodes zijn niet of in mindere mate instaat om alle soorten van lichttransport (glas, niet diffuuse of glimmende oppervlakken en spiegels) goed weer te geven (Radiosity / Ray Tracing).

Maar omdat het berekenen van de integraal, betekent dat er veel berekeningen voor nodig zijn met veel verschillende lichtstralen is het bereiken van de ultieme kwaliteit ook hier een kwestie van geduld en rendertijd.





Figuur 1 van de literatuur Alexander Keller
Geometrie van de stralingsvergelijking. Lit.: quasi_monte_carlo_methods_in_computer_gr_92604.pdf

Figuur 2 van de literatuur Alexander Keller
De gebruikte 3D modellen als test voor de verbetering en aanpassing van de bestaande originele Monte Carlo methode. De nieuwe methode komt aanzienlijk sneller tot een goed eindresultaat dan de originele uitvoering.
quasi_monte_carlo_methods_in_computer_gr_92604.pdf Op pagina 13 van deze PDF de tabel met de versnellingswaardes per 3D scene.

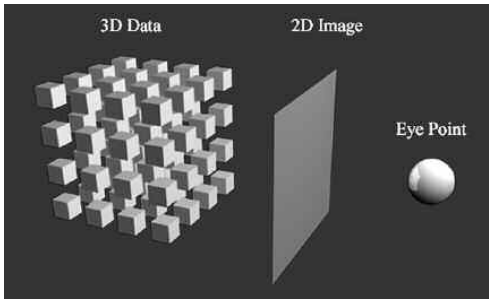
Ook bij "Ray_Tracing_Acceleration_using_Displacement_Fields_on_GPU.pdf" lezen we dat er een methode is ontwikkeld om Ray Tracing aanzienlijk te versnellen door een andere manier van vastleggen en combineren van de vectorpunten die bij elkaar in de buurt komen. Afbeelding: Ray Tracing Rendering Algorithme.

Ray Casting

Gegeven een straal, zoek een object die wordt aangedaan (geraakt) door de straal. Dat is de meest simpele manier om Ray Casting (1968 Arthur Appel en 1982 Scott Roth) uit te leggen. Test gewoon elke straal ten opzichte van alle objecten, maar dat is bijzonder tijd intensief, vandaar dat er een aantal algoritmes zijn bedacht om dat te versnellen.
Ray Casting maakt tegenwoordig deel uit van veel render systemen in het bijzonder op het gebied van Global Illumination.

Een aantal auteurs maken onderscheid tussen *Ray Casting* en *Ray Tracing*. De eerste is een simpele zichtbaarheidstest en de tweede is wat ingewikkelder door een lichtstraal te volgen gedurende verschillende reflecties met diverse oppervlakken van de objecten in de 3D scene.

Bij Ray Casting worden de stralen ook tegengesteld (vanuit het zichtpunt) het 3D model ingestuurd. Er kan ook terrein (bv. bergen en hoogte modellen) worden gebruikt. Ray Casting is snel door de beperkingen die opgelegd worden aan het model. Zo staan de muren vrijwel altijd 90° t.o.v. de vloer. Vandaar dat in een Ray Casting Game het zichtpunt niet langs de Z-as kan worden gedraaid.



In Ray Casting wordt het 3D model pixel voor pixel afgetast, regel voor regel vanaf het camera (oog) standpunt. Zodra een denkbeeldige straal een object treft wordt de helderheid en kleur van het trefpunt berekenend aan de hand van verschillende technieken.
In de meest eenvoudige vorm krijgt het pixel bijvoorbeeld de kleur van groen mee, indien het 3D model op dat punt groen is gekleurd. De juiste kleur wordt middels een texture map overgenomen van het 3D model.

Een stap verder gaat het, als het pixel niet alleen de kleur groen krijgt, maar ook de verlichting van dat trefpunt wordt meegenomen in de bepaling van de 'werkelijke' kleur en helderheid van dat punt. Daarbij worden nog geen kunstmatige lichtbronnen betrokken. Om een redelijke nauwkeurigheid te verkrijgen is het nodig dat diverse stralen worden uitgezonden onder kleine verandering van de hoek. Het pixel (trefpunt) krijgt dan een gemiddelde waarde mee van al die stralen. Maar het proces is daarmee nog niet afgerond. Indien één of meerdere lichtbronnen in de berekening worden meegenomen, dan zullen er nog wat stralen en vervolgberekeningen moeten volgen. Vanuit het trefpunt wordt een eenvoudige berekening naar de camera gemaakt en tevens de invalshoek wordt berekend van de lichtbron of bronnen op het trefpunt. Samen met een middeling van de sterkte van de lichtbronnen, de invalshoek, de afstanden tot de lichtbronnen wordt een optelsom gemaakt van de helderheid en kleur van dat pixel.
Een andere simulatie kan plaats vinden door het gebruik van Radiosity en oms kan dat meer of minder in combinatie met de voorgaande technieken worden toegepast.

Ray casting wordt voornamelijk gebruikt voor snelle simulaties zoals bv. bij Games en animaties. Door de snelheid van het renderen is de kwaliteit iets minder belangrijk. Er worden nog allerlei nieuwe technieken toegepast om details, die er minder toe doen, eenvoudiger uit te rekenen om tijd te winnen. Door deze technieken kan een Game er soms bij stilstand ‘vlak’ uit komen te zien waarbij het lijkt alsof het met een beperkt aantal verfkleuren is ingekleurd. Maar bij Games wordt niet stil gestaan bij de bestaande principes, nieuwe technieken maken het mogelijk om ‘echte’ renderingen te maken die er goed uitzien met hoge Frame snelheden. Zelfs spiegelingen van oppervlakken worden tegenwoordig meegenomen in auto race games. Aangezien de Game markt groot is kan de conventionele architectuur markt van deze nieuwe ontwikkelingen leren en ze toepassen voor de 'High End' renderingen, die wat langer mogen en vaak ook moeten duren. Maar ook daar zal snel verandering in optreden.
[literatuur nr. 156 - 1997]

Volume Rendering
Deze term wordt gebruikt om de technieken te beschrijven die visualisatie van drie-dimensionele datasets mogelijk maken. Het gaat daarbij om 'bemonsterde functies' van 3D ruimtelijke afmetingen door het berekenen van 2D projecties van een gekleurd semi-transparant volume.

Volume Rendering komen we bij architectuur niet snel tegen, maar wel bij medische toepassingen, waarbij de data bijvoorbeeld afkomstig is van een X-ray Tomagraphy, CT-scanner of PET (Positron Emission Tomagraphy).

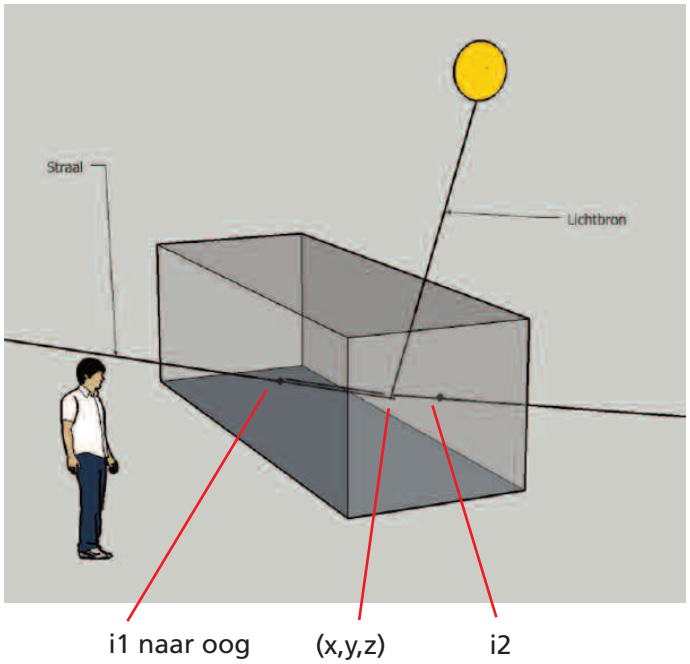
De meeste **Ray Casting** algoritmes zijn gebaseerd op het Blinn/Kajiya model, waarbij het volume een dichtheid D bezit die door de 'straal' R vanuit het oog wordt benaderd.

Langs de straal is er sprake van een verlichting I die uiteindelijk het punt (x,y,z) bereikt waarbij de straal van de lichtbron wordt gekruisd. De vector vanaf (x,y,z) naar het linkeroppervlak is i1 en die naar het rechter oppervlak i2.

De stralen komen van buiten en worden door een oppervlak (object) tegengehouden. Een deel van het licht stopt en wordt geabsorbeerd. Een ander deel zal door het oppervlak worden gereflecteerd en nog een ander deel zal zijn weg (met of zonder verandering van de invalshoek in drie dimensies) vervolgen in het object zelf.
Bij Ray Casting wordt bekeken welk deel van het licht wordt toegekend aan één van de drie mogelijkheden.
In de jaren '80 van de vorige eeuw deed Ray Casting opgang bij de animatie industrie (TV en film). Daarna volgden de video games.

Waarom Ray Casting sneller is dan Ray Tracing?
Bij Ray Casting zal een 320 x 200 pixel rendering alleen door 320 stralen worden uitgerekend. De stralen worden in groepen bewerkt.

Met dezelfde resolutie van 320 x 200 pixels zijn bij Ray Tracing 64.000 stralen nodig.



Waardoor deze bijna 200 keer zo langzaam is. Zie de literatuur "05-Raycast.pdf" en "Raycasting03.pdf".

Pseudo code voor Ray Casting

```
Image RayCast (Camera camera, Scene scene,
int width, int height)
{
    Image image = new Image (width, height);
    Set P0;
    for (int i = o; i < height, i++ ) {
    p = P0
    for (int j = 0; j < width, j++) {
    Ray ray = E + t * (p - E);
    Intersection hit = FindIntersection (ray, scene);
    image [i][j] = GetColor(hit);
    p += Vx;
    }
    P0 += Vy;
    }
    return image;
}
```

http://leftech.com/raycaster.htm
Javascript RayCaster

Tracing

Tracing is in tegenstelling tot bv. Ray Tracing een sneller en minder geavanceerd renderproces. Waardoor het zeer geschikt is om als Preview te dienen en bij de voorbereiding van animaties en Real Time Editing.

Scanline rendering en rasterisation

Een rendering van een 3D scene omvat elementen uit verschillende domeinen naar de uiteindelijke render pixels toe. Deze elementen worden als 'primitives' gezien. Een 'primitive' kan bv. een lijn segment, een driehoekig oppervlak of een curve zijn.

Een 'primitive' naar 'primitive' aanpak kan handig zijn om te ontkomen aan het pixel-by-pixel systeem dat veel langzamer is. In één lus wordt elke primitive bij langs gelopen en bepaald welke pixels in de afbeelding worden beïnvloed en overeenkomstig gewijzigd moeten worden. (= rasterization). Daarbij wordt een rij-voor-rij methode toegepast in plaats van een polygoon-voor-polygoon methode. De polygonen worden eerst gesorteerd om vervolgens rij-voor-rij te worden afgetast. Het is de standaard renderingsmethode die door grafische kaarten wordt toegepast. Het voordeel is dat het aantal vergelijkingen langs de randen kan verminderen. En een omzetting van alle coördinaten vanuit het hoofdgeheugen naar het werkmodel is niet nodig. Alleen de hoekpunten die elkaar snijden worden in het actieve werkmodel vastgelegd en ze worden maar eenmaal ingelezen. Daarmee wordt voorkomen dat het relatief langzame RAM geheugen een beperking vormt, omdat het veel snellere Cache geheugen met de CPU deze bewerkingen uit kan voeren.

Grote delen van het 3D model bevatten wellicht geen primitieven waardoor het proces sneller kan verlopen. Rasterization wordt dan ook vaak ingezet bij interactieve renderingen (bv. Previews) waarbij het 3D-model snel en redelijke goed op het scherm moet worden weergegeven.
Het scanline rendering algoritme kan gemakkelijk worden uitgebreid met het Z-buffer, het Phong reflectie model en vele andere aanvullende verbeteringen.

Met Z-buffering kan een hybride methode worden gebouwd (zichtbare pixels worden maar een keer ingelezen), waarmee de actieve hoekpunten tabel sortering komt te vervallen. Daarvoor in de plaats wordt een scanlijn per keer in het Z-buffer geladen. Er zijn legio varianten op deze technieken waaronder ook het ID buffer systeem.

Zo begon het
De eerste scanlijn methode werd in 1967 beschreven door Rommey, Evans, Erdahl en Wylie. Twee jaar daarna door Bouknight en Newel. Tenslotte Sancha in 1972.
De universiteit van Utah had zich in dit onderwerp vastgebeten en veel van het ontwikkelingswerk werd door de Ivan Sutherland's Graphics Group uitgevoerd en ook door z'n inmiddels opgerichte bedrijf Evans & Sutherland company in Salt Lake City.
Z'n bedrijf ontwikkelde een image-generator (IGs) die dat in hardware 'on the fly' kon uitrekenen. In zijn bedrijf was ook één van de medeoprichters van het huidige Adobe aanwezig John Warnock en Jim Clark van Silicon Graphics werkzaam.
De naam **Ivan Sutherland** kwamen we ook al tegen bij de "Geschiedenis.pdf", waar hij in z'n proefschrift in één jaar een werkend en intelligent tekenprogramma van de grond kreeg, een enorme prestatie.

Er zijn verschillende grote namen die scanline rendering hebben toegepast in diverse uitvoeringen: Nintendo DS met gerasterde afbeeldingen in VRAM. Sony was bezig met een tweede processor voor PlayStation 3 maar besloot uiteindelijk voor de standaard CPU/GPU oplossing. Verder de medio 1980 toegepast sprites in games.

Ook bij 'tiled rendering' worden primitives in de schermruimte gesorteerd en vervolgens vliegensvlug gerenderd, één tegeltje per keer.
Bij de universiteit van Utah vinden we een opsomming van ontdekkingen, waaronder Phong lighting model, computer animatie en

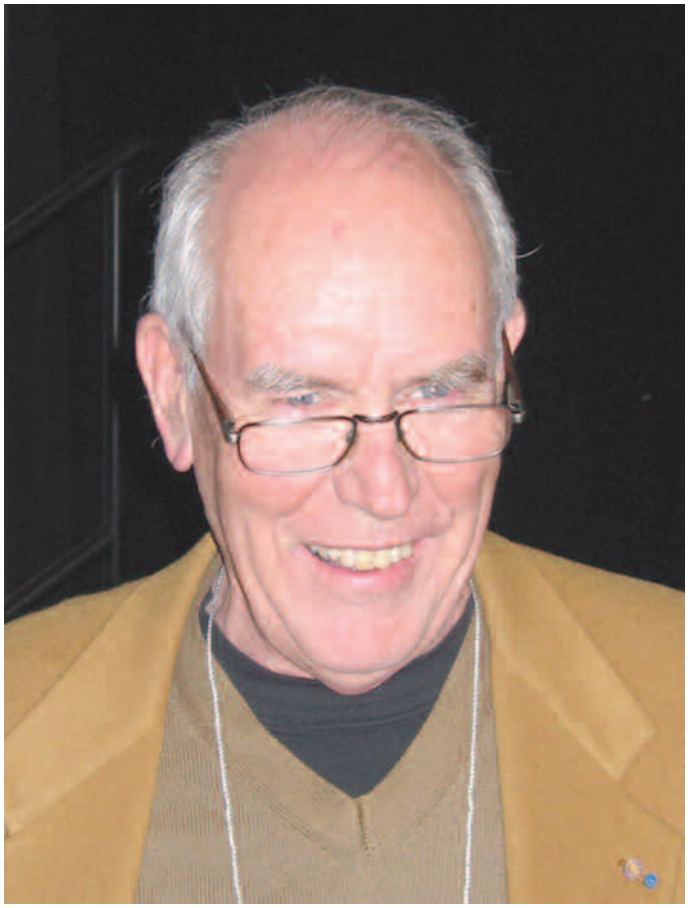
art, grafische gebruikers interfaces en Sketch-pad etc. Ivan Sutherland (Computer Science Faculty van 1968 - 1974) heeft daar een prominente rol in. Hij ontving vele prijzen.

Literatuur

"50700049.pdf"
Half-tone perspective drawings by computer door Chris Wylie, Gordon Romney, David Evans en Alan Erdahl. Universiteit van Utah.

"50750001.pdf"
An algorithm for producing half-tone computer graphics presentations with shadows and movable light sources.
door J. Bouknight en K. Kelley. Universiteit van Illinois.

"p443-newell.pdf"
A solution to the Hidden Surface Problem.
door M.E. Newell, R.G. Newell, T.L. Sancha.
The Computer-Aided Design Center. Met een referentie naar J.E. Warnock.



Ivan Sutherland op z'n 70ste verjaardag bij het Computer History Museum op 22 mei 2008. Wikipedia.org.

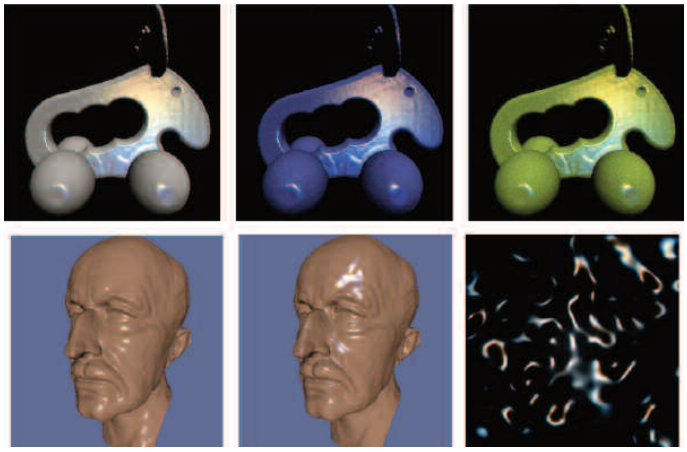
Schlick's benadering

Bi-drectional reflection distribution function (BRDF) is de term bij 3D computer graphics om de benadering aan te geven, van de Fresnel termen, bij spiegelende reflecties van licht op gladde oppervlakken.
De reflectie coëfficient spiegelende reflectie factor R wordt door Schlick weergegeven:

$$R(0) = Ro + (1 - Ro)\,(1 - \cos phi)$$

waarin phi de halve hoek is tussen de inkomende- en de uitgaande lichtstralen.
R0 is de reflectie bij de normaal van het punt van aanraking.

"PG.02.pdf"
A Simple Layerd RGB BRDF Model
University of British Columbia, Department of computer Science.



Uit "A Simple Layerd RGB BRDF Model" Geheel rechts het Phong model.

Photon Mapping

Film over interactive progressive photon mapping:

http://www.youtube.com/watch?v=o-rKLs64Qk8&feature=player_embedded

van deze site:
http://cg.alexandra.dk/category/realistic-rendering/

De twee films laten zien hoe razendsnel Photon Mapping kan werken indien de algoritmes volgens nieuwe inzichten zijn aangepast.

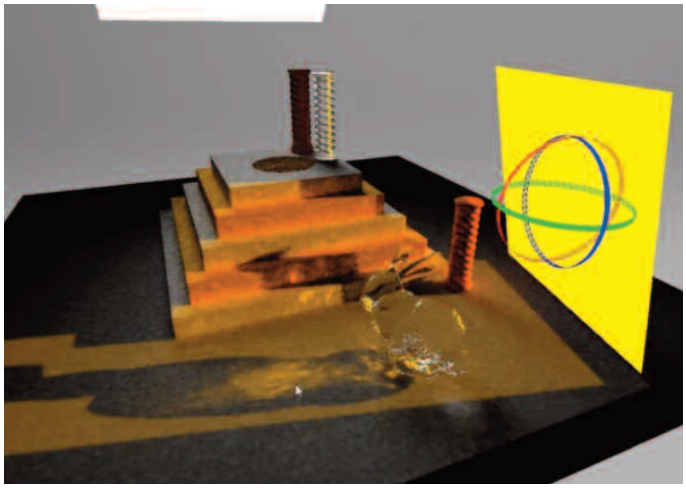
Wat is Photon mapping?
Een renderingssysteem waarbij in twee stappen virtuele lichtstralen op een 3D model worden afgestuurd, de trefpunten daarvan worden opgeslagen in een tussengeheugen en later met elkaar vergeleken.

Het is een algoritme dat bedacht is door ***Henrik Wann Jensen*** en een nieuwe ingang geeft tot het maken van een fotorealistische renderingen. Het is echter wel een zgn. 'biased' rendering algoritme zoals Path tracing, bi-directional Path tracing en de nieuwere Metropolis Light Transport methoden.

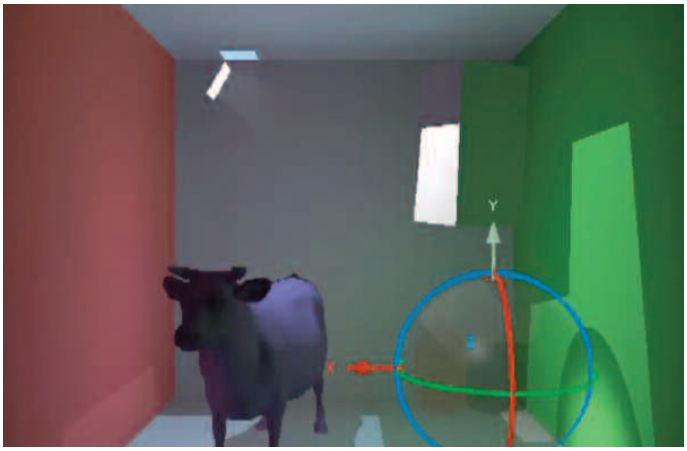
Photon Mapping is een uitbreiding van Ray Tracing [Glassner89, die in 1989 al bepaalde dat diffuuse refelecties, caustic of golven in het water nog niet best naar voren kwamen in die tijd].

Het Photon Map algoritme werd in 1993-1994 ontwikkeld en de eerste publicaties werden in 1995 uitgebracht. En prettig voor commerciële renderfirma's: *er berust geen patent op het gebruik van Photon Maps.*

Photon Mapping is een twee-traps raket, waarbij de stralen van twee kanten komen. Zowel van de lichtbronnen als van de camera standpunt worden stralen in de 3D scene geschoten. Ze worden daarbij onafhankelijk van elkaar vastgelegd totdat één van de beïndigings kriteria wordt bereikt. In dat geval worden de twee afzonderlijke stralen aan elkaar gekoppeld en bij het trefpunten of meerdere trefpunten wordt de stralingswaarde en kleur opgeslagen voor de uiteindelijke renderingsafbeelding. Daarmee wordt gepoogd om een zo realistisch en natuurlijk mogelijke



Terwijl u kijkt en een lichtbron of een object wordt gewijzigd wordt het proces van het renderen life in beeld gebracht.
Interactive progressive Photon Mapping. Experimenten met een snelle bvh rebuilder.



Film en presentatie
http://www.youtube.com/watch?v=GckOkpeJ3BY
Uploaded by BCEMCOCATb on Feb 20, 2010
Een verdere uitbreiding van Photon Mapping: Image Space Photon Mapping (ISPM). Waarbij BSDF's ook kunnen worden toegepast.

To be or not to be . . .

http://www.maxwellrender.com/index.php/technology
Maxwell Render
"Maxwell Render is physically accurate"
OctaneRender
"Physically based GPU renderer"
Artlantis Maxwell Engine
"Artlantis is the fastest 3D rendering application available today for architects and designers. And Maxwell Render Engine is considered to be the most physically accurate rendering engine on the market. "

http://simplymaya.com/autodesk-maya-video-tutorial/lighting-and-rendering/physically-accurate-lighting-in-mental-ray/?tut_id=207

"Physically Accurate Lighting in **Mental Ray** *"*

http://forums.luxology.com/topic.aspx?f=36&t=66395
"Modo also does not take transparency into account when conserving energy, so materials are not physically accurate."
"I agree. Now with 601 we have new custom materials, so it would be great if luxology (or a 3rd party) could make a new physically accurate custom material with the features we want :)"

http://www.luxrender.net/en_GB/project_goals
Free software
*"***LuxRender** *is a free and unbiased renderer. The program is the result of the dedication of a loosely nit team of passionate people around the world."*

http://www.jigsaw3d.com/articles/physically-accurate-linear-workflows-part-1
V-Ray & 3ds Max
"That pretty much concludes all the considerations you need to make when setting up 3ds Max and V-Ray for a physically accurate workflow."

http://forums.cgarchitect.com/21183-vray-physically-correct.html
Is V-Ray eigenlijk wel fysisch correct?
"You see that is an interesting concept because what the human eye can see is somewhat subjective... I would say that even Greg Ward (the creator of radiance) would agree to that. The closest thing that a computer program can do to simulate things that are "physically accurate" is to simulate photography. Programs like Maxwell would be a good choice for that. Or, with Vray, using an unbiases system with PPT and Vray Cameras would be able to achieve that. The most interesting things I have seen to simuate human vision that Greg did was the tonal mapping stuff which shows how the human eye deals with the high dynamic range of light."

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.106.422
Coherent Raytracing
"It turns out, however, that as ray tracing is accelerated, other parts of physically accurate rendering algorithms tend to become bottlenecks. In this paper, we introduce a coherent physically accurate rendering approach, which allows taking advantage of SIMD capabilities of modern CPUs at every stage of rendering computations."

In literatuur:
s2010_phyysically_based_shading_hofman_b.pdf
pagina 5 van de 66 stuks.
"Less tweaking and 'fudge factors' ".

Vergelijking
http://www.keldysh.ru/pages/cgraph/articles/pals/index.htm
Physically Accurate Lighting Simulation in Computer Graphics Software
This paper compares three different computer graphics systems claiming to physically accurate global lighting simulation. Physical accuracy of these systems is the main criterion for this comparison.

The following systems are compared:

* LVS ver. 1.2.4 distributed by Lightscape Technologies Inc.
* Specter system ver. 4.83.1 distributed by Integra, Inc.
* **Radiance** ver. 2.5, a public system developed by G.Ward

Conclusie:
http://www.keldysh.ru/pages/cgraph/articles/pals/conclusion.htm

Literatuur: gdc.pdf
"Fundamentals of Lighting and Perception: The Rendering of Physically Accurate Images".

http://en.wikipedia.org/wiki/LuxRender
LuxRender is a free and open source software rendering system for physically correct image synthesis"
"LuxRender is based on PBRT, a physically based ray tracing program."

Indigo Render
http://www.indigorenderer.com/documentation/manual/techniques/physically-correct-modelling-glass
"Physically correct modelling of glass"
"This would eventually converge to the previous image, however it would take very long to do so, making Glass Acceleration a crucial feature in such scenes.
Convergence could also be improved by using Bi-Directional Path Tracing, possibly with MLT enabled, however Path Tracing and Glass Acceleration is an excellent fit for exterior architectural shots. For interiors, Bi-Directional Path Tracing would likely be superior, however enabling Glass Acceleration is still strongly suggested if much of the illumination comes from outside."

http://www.simplyrhino.co.uk/products/maxwell.html
"**Maxwell Render™** v2.6 is an unbiased rendering engine based on the mathematical equations governing light transport, meaning that all elements in the scene are derived from physically accurate models."

http://www.spectralpixel.com/
"Spectral Studio is a photo realistic renderer, working on both the GPU or the CPU, un-biased and based on physically correct rules. It is a standalone tool that provides a complete solution for those who want great image quality and full control."
"Physically correct materials supporting diffuse, fresnel IOR, dielectrics, and many more..."
Physically correct ?
"Spectral Studio simulates the flow of light like in the real world, according to physical equations, thus producing image of outstanding quality."

http://www.thearender.com/cms/
"**Thea Render** is the most versatile render solution featuring multiple state-of-the-art biased and unbiased engines. "

Antwoord op vragen bij Thea Render
• I have seen the term "physically-plausible", what does this mean?

• I see some engines described as "physically-based", does it differ from being physically-plausible?

• Is Thea Render a physically-based engine?

• What does the term "bias" mean?

• So, an unbiased render engine will render with zero error and all unbiased engines will have the same solution?

• I see the images being very noisy and refined over time, is this a sign of being unbiased?

http://www.thearender.com/cms/index.php/features/faqs/48-glossary-of-terms.html

http://forums.cgsociety.org/archive/index.php/t-420003.html
Bij literatuur als html pagina bijgevoegd.

Fysisch correct:
3D modellen om wolken te maken, weergave van glas en water, materialen, zelfs kubussen, texturen, foto's, camera's etc. etc.

weergave te verwezenlijken door het samenspel van licht en objecten in de scene.

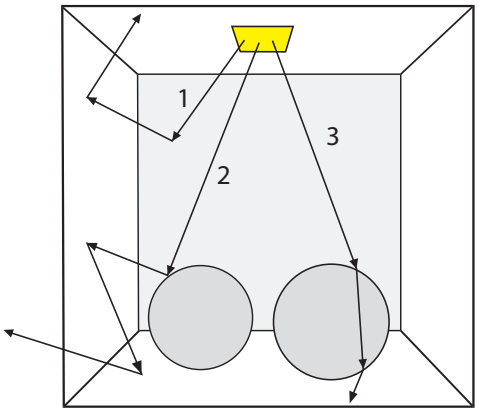
Met deze geheugen- en tijd intensieve methode kunnen een aantal optische eigenschappen worden ingebouwd, die bij andere renderings algoritmen slechts kunnen worden gesimuleerd.
We denken daarbij aan de simulatie van lichtbreking door een transparante oppervlakken of vloeistof, bv. een glas met water. Spiegelende oppervlakken of semi-spiegelend. Verder diffuse onderlinge reflecties tussen verlichte objecten. Plus lensvervormingen en allerlei andere effecten die we uit de fotografie kennen, oppervlakte onefenheid reflecties en natuurlijke weergave van water en rook.
Het kan worden uitgebreid met een nog nauwkeuriger eindresultaat door het gebruik van "Spectrale Rendering", tenkoste van extra renderingstijd.

Photon Mapping is dus een 'biased' rendermethode waarbij het gemiddelde niet direct tot de exacte render vergelijking leidt. Het is echter wel een consequente methode waarbij een steeds toenemend aantal photonen een beter resultaat oplevert. Het staat daarmee voorlopig haaks op de wens van de gebruikers om de renderingstijd terug te dringen met gebruik van redelijk goede computers en grafische kaarten.

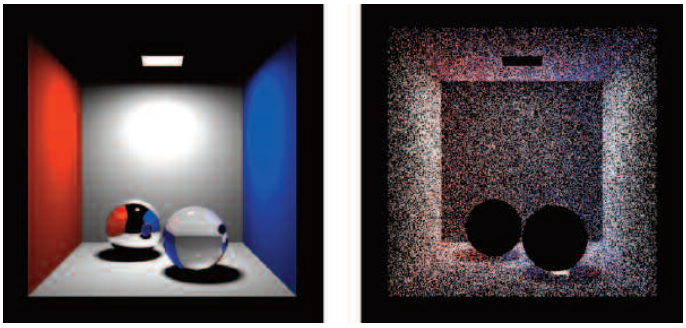
Photon Map voor grote 3D modellen
De Photon map is losgemaakt van de geometrische weergave van de scene, waardoor het mogelijk werd om complexe scenes te renderen die vele miljoenen driehoeken en andere oppervlakken bevatten. In tegenstelling tot Radiosity algoritmes is er hier geen 3D Mesh nodig. Toch blijft Radiosity (bv. gebruikt in Artlantis vers. 4) aanzienlijk sneller voor eenvoudiger diffuuse scenes. Zodra de 3D scenes complexer worden, kunnen de Photon Maps beter de schaal aan, waarbij ook niet diffuuse oppervlakken en caustics worden verkregen.

Een standaardwerk over Photon Mapping is:
Realistic Image Synthesis using Photon Mapping, AK Peters, 2001 van Henrik Wann Jensen.

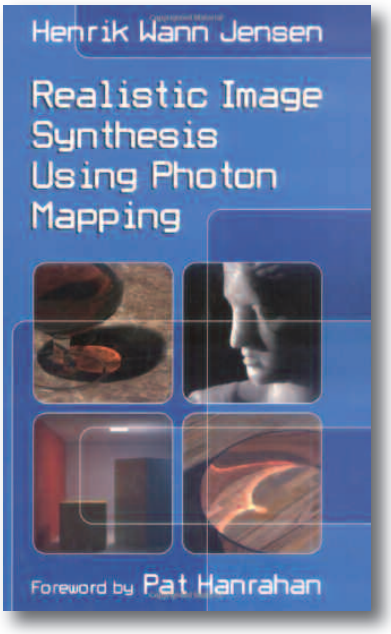
Gedurende het Photon Tracing traject worden de Photonen opgeslagen bij de diffuuse oppervlakken die ze op hun weg aandoen.



Cornell Box met Photonen banen. Links een chrome bol, rechts glas.
1. Twee difuuse reflecties plus absorptie. 2. een spiegelende reflectie met 2 difuuse. 3. twee spiegelende uitstralingen en daarna absorptie.



Cornell Box met glas en chroom.
Links: Ray Traced afbeelding met directe verlichting en spiegelende refelctie en uitstraling.
Rechts: de photonen in de overeenkomende Photon Map. [ref. Course43sig02.pdf]



Standaard werk met uitgebreide beschrijving render principes plus Photon Mapping incl. C++ listings.

Elk uitgestraalde photon kan meerdere keren langs het pad worden vastgelegd.

Wat wordt er opgeslagen?
In principe zo weinig mogelijk om het geheugen niet te veel te belasten. De plaats, de richting bij het trefpunt, kracht van de Photon en één of meerdere vlaggen. Volgens Jensen [Jensen96b] zou dat in 20 bytes kunnen, maar 18 bytes blijken ook genoeg. 12 voor de plaats, 4 voor de gecomprimeerde kracht en 1 voor de richting en een vlag. De Photonen die een diffuus oppervlak treffen die vanaf een spiegelende oppervlak afkomstig zijn worden niet in de Global Photon Map opgeslagen, maar in de Caustic Photon Map. Nadat de gegevens zijn vastgelegd volgt een sorteer routine om zo snel mogelijk het dichtstbijzijnde verbindingspunt op te zoeken.

In principe kunnen drie Photon Maps worden toegepast, maar daar is ook al een uitbreiding op:

1. Caustic Photon map via een spiegelende oppervlak
2. Global Photon map, globale verlichting voor alle diffuuse oppervlakken
3. Volume Photon map, indirecte verlichting

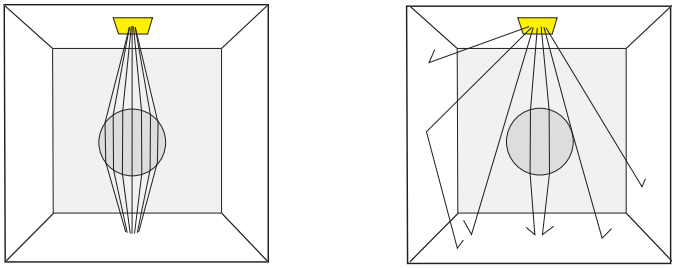
Zie bovenaan de pagina de gele inzet voor de pseudo code. Het verschil tussen een caustic map en een Photon Map in de afbeelding daaronder.

Ter illustratie en voor de geschiedenis:
Christensen in zijn "*Faster Photon Map Global Illumination.pdf*" deed zijn testen met een Sony Vaio laptop met een enkele **233 MHz Pentium** processor en slechts 32 MB aan RAM geheugen! Dat hij daarmee prachtige Cornell Box renderingen kon maken is bijgaand te zien.

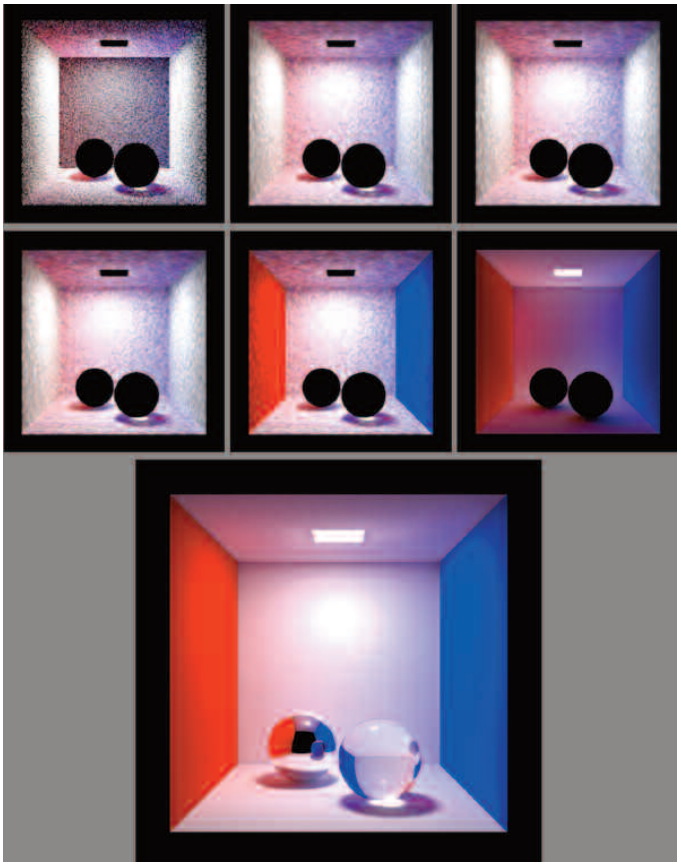
De complete rendering (meer dan 400.000 Photonen) met directe verlichting, spiegelende reflectie en refractie, caustic en zachte directe verlichting. december 1999.
http://www.acm.org/jgt/papers/christensen99
werkt niet meer !
wèl: http://www.acm.org/
Association for Computing Machinery
[Christensen_Faser Photon Map Global Illumination.pdf] Per H. Christensen, Honolulu

```
struct photon {
    float x, y, z;     // position ( 3 x 32 bit floats )
    char p[4];         // power(rgb) packed as 4 chars
    char phi, theta;   // compressed incident direction
    short flag;        // flag used for kd-tree
}
```

Voor elke Photon wordt de plaats, kracht en richting vastgelegd. Volgens Jensen:
4 bytes voor de RGB kleur en kracht, de phi en theta zijn de hoeken van de bol coördinaten met 65536 mogelijke richtingen. Wellicht dat de plaats nog in een 42-bit fixed point format kan worden gecomprimeerd.



Links de opbouw van een caustic map, rechts die van een global Photon Map.



Cornell box. 1. Globale Photon map. 2. Stralings benadering berekend op sample punten. 3. Stralings benaderingen van alle 400.000 Photonen. 4. Idem met 100.000 Photonen. 5. Straling gebaseerd op 6. Zachte indirecte verlichting van geheel.
7. Complete rendering met directe verlichting.

Caustic
Caustics is de naam voor licht dat gereflecteerd of gebroken wordt waardoor patronen ontstaan. Vaak te herkennen aan geconcentreerde hooglichten op de nabij gelegen oppervlakken. Als licht door een wijnglas heen gaat, dan is dat niet alleen in het glas te zien, maar ook op de tafel. Begrijpelijk dat een renderingsprogramma daar moeite mee heeft om dat zo natuurgetrouw weer te geven. Op de volgende pagina het wijnglas van Tobias R. Metoc, Wikipedia 2006.Caustic kan bij een aantal renderprogramma's worden aan- of uitgezet. Bijvoorbeeld default Quality rendering, High Quality Rnderingen, Viewport 2.0 and Caustic Visualizer.

zie afbeelding rechts onder.

Lichtbronnen
Indien een 3D scene meerdere lichtbronnen heeft, dan dienen de Photonen naar elk daarvan worden uitgestraald. En met helderder licht zijn dat er meer dan met donkerder ingestelde lichtbronnen. Met het gebruik van bv. 30 lichtbronnen zou het aantal Photonen ook evenredig toenemen plus de opslag capaciteit in de Photon Map. Maar dat blijkt niet zo te zijn. Elke lichtbron zorgt voor zijn eigen bescheiden bijdrage aan het verlichten van de scene (het uiteindelijke resultaat) of te wel het maken van de afbeeldingsrendering.

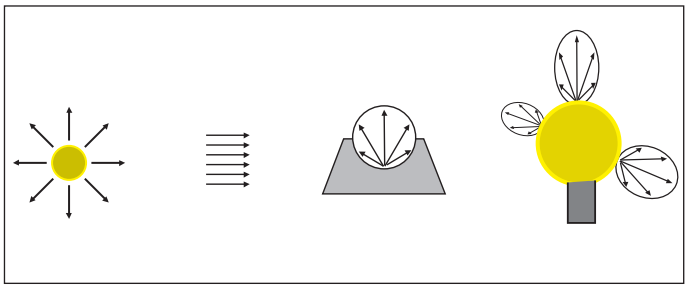
Om een helder beeld te krijgen is het dus niet nodig dat in vergelijk met 1 lichtbron veel meer Photonen te gebruiken.

Maar het vraagt wel aandacht van de ontwerper van het programma. Als er slechts enkele lichtbronnen in de scene zijn die ook nog eens belangrijk zijn, dan kunnen ze toegevoegd worden in een zgn.

Sampling Map [literatuur Peter98] om de Photonen in dat gebied te concentreren. Daarbij is het interessant om juist die lichtbronnen als belangrijk te bestempelen die dat ook werkelijk voor de toeschouwer zijn. En de totale hoeveelheid licht ten opzichte van de rest van de scene dient in balans te worden gebracht.

Victor Loba, straalbreking in een glas.

Wijnglas, Tobias R. Metoc, Wikipedia 2006.
Het glas is met NURBS oppervlakken opgebouwd. Renderprogramma Metal Ray (caustic & Raytracing) plus Maya.



Lichtbronnen
Van links naar rechts:
puntlicht, gericht licht, verstrooïd licht en algemene (lamp) verlichting.
Een lichtbron kan elke vorm aannemen en uitzendend karakter. De intensiteit van het licht varieert zowel wat oorsprong betreft als richting.

Simpele scenes met Projection Map
In simpele scenes met slechts enkele objecten zullen een groot aantal Photonen in het geheel geen object tegenkomen en verloren gaan. Het berekenen van dergelijke Photonen is dus kostbare renderingstijd verdoen. Om dat te verbeteren is de "Projection Map" [Jensen 93, Jensen95a] ontwikkeld. Een eenvoudige geometrische kaart van het 3D model, zoals deze vanuit de lichtbron wordt gezien. Deze kaart bestaat uit kleine cellen die aan of uitgezet kunnen worden. De cel is 'aan' als er geometrie in die richting is. Tevens wordt een schatting gemaakt van het totale aantal Photonen die nodig is in een bepaalde richting.

Hoe gaat het in z'n werk?

De eerste straal
Lichtstralen (voor het gemak Photonen genoemd) worden de scene ingestuurd vanuit de lichtbronnen. Zodra een Photon een oppervlak bereikt van het 3D model wordt het trefpunt daarvan opgeslagen, samen met de hoek waaronder het oppervlak wordt bereikt. De Photon map is een soort cache geheugen voor deze gegevens. In de praktijk worden zelfs twee (of meer) Photon maps aangelegd, één voor caustic en één voor globale voor het andere licht. Zodra een Photon weggestuurd is wordt het door de Photon tracing gevolgd ('Light Ray tracing', ook wel 'backward ray tracing' Forward Ray Tracing en Backward Path Tracing)

Photon Tracing werkt identiek aan het normale Ray Tracing met één belangrijk verschil:

- Photonen planten zich voort en kunnen licht uitstralen
- Rays verzamelen straling

Bij het trefpunt zal het effect van een Photon dus kunnen afwijken met die van een Ray.

Eén van de voorbeelden is de lichtbreking van een straal waarbij de relatieve brekingsindex wordt gebruikt, dat gebeurt niet met Photonen. Indien een Photon een oppervlak bereikt dan kan het ofwel worden gereflecteerd, uitgezonden of geabsorbeerd. De eigenschappen van het oppervlak bepalen wat er gebeurt. Maar dat gebeurt dan op een semi willekeurig manier waarop dat

$$P\ photon = \frac{P\ licht}{np}$$

P photon = de kracht van elk individueel Photon
P licht = Totale kracht van de Photonen (power / vermogen)
np = aantal uitgezonden Photonen

[course43sig02.pdf]

Pseudo code voor uitstraling van photonen van een diffuus puntlicht

```
emit photons from diffuse point light() {
  ne = 0 // number of emitted photons
   while (not enough photons) {
   do { use simple rejection sampling to
   find diffuse photon direction
     x = random number between -1 and 1
     y = random number between -1 and 1
     z = random number between -1 and 1
    } while ( x2 + y2 + z2 > 1 )
    d = < x; y; z >
    p = light source position
    // trace photon from p in direction d
    ne = ne + 1
  }
  scale power of stored photons with 1/ne
}
```

[course43sig02.pdf]

plaats vindt. De techniek staat bekend onder de naam van Russische roulette [Arvo90] of Monte Carlo methode. Met een dobbelsteen wordt bepaald of een Photon het overleefd of niet.

Bij het trefpunt gebeurt het een en ander. Niet alleen de aangestuurde richting en de afstand tot de lichtbron, maar ook de eigenschappen van het materiaal (oppervlak) komen naar voren. Dat kan zijn de reflectie van de straal, absorptie, uitzenden en straalbreking etc.

Monte Carlo
Met de Monte Carlo methode wordt één van de opties vrij willekeurig gekozen. Waarbij er een weging plaats vindt. Bij absorptie zal de Photon bij het trefpunt ophouden om verder te gaan. Het doel van de Monte Carlo methode lijkt op het eerste gezicht vreemd. De oppervlakken hebben hun bepaalde eigenschappen en het ligt

Test	Algorithm	Number of samples per pixel	Total number of rays	Time (sec)
1	Path tr.	60	45×10^6	15 897
2	Bi-dir. path tr.	20	30×10^6	11 036
3	Path tr.	60	35×10^6	18 251
4	Bi-dir. path tr.	20	31×10^6	14 929

Test resultaten 400 x 400 pixels
[Bidirectional_path_tracing.pdf]

voor de hand om deze dan bij reflectie ook in zijn geheel te benutten en mee te nemen in de berekening.

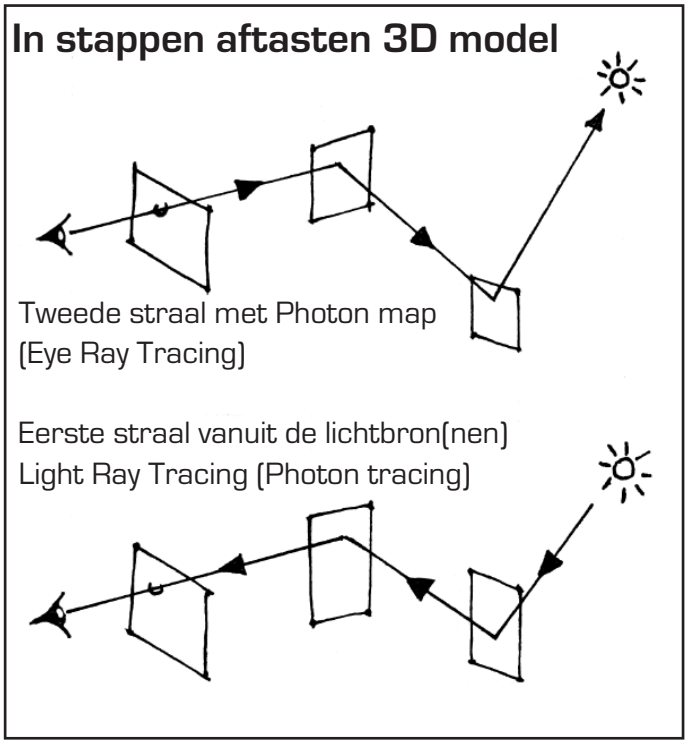
Waarom extra routines inlassen om te bepalen wat de Photonen doen?
In de eerste plaats is het bijzonder prettig om photonen met gelijke vermogens (helderheid) in de Photon Map op te slaan. Daardoor kan de uitstralingsweging met slechts enkele Photonen in balans worden gebracht. Het tweede punt is dat met bv. twee Photonen op hetzelfde oppervlak het 2^8 photonen na 8 interacties oplevert. Er zijn dus na 8 interacties 256 Photonen om uit te rekenen in plaats van 1 Photon die direct vanuit de lichtbron komt.

Overzicht van testresultaten voor Path Tracing en Bi-directional Path Tracing.

[shirley90]
[pattanaik93]
[glassner95]

Bij reflectie wordt de "Bidirectional Distribution Function" gebruikt om de verhouding van de gereflecteerde straling te bepalen. En als de photon zelf licht uitstraalt, dan wordt de richting berekend afhankelijk van de eigenschappen van de uitstraling.

Tijdens de aanmaak van de **Photon map** of daarna is het zaak om de gegevens zo goed en vooral zo **snel** mogelijk met elkaar te vergelijken. Welke stralen (van lichtbron en camera) komen zo dicht bijelkaar dat ze in een bepaald gebied tesamen kunnen worden gebruikt? Jensen heeft daar het gebruik van *kd-trees* geïntroduceerd. Dit algoritme zoekt in een zo'n kort mogelijke tijd de nabije k-buurman op (k-nearest neighbor algorithm).



Is dit uitzoekwerk klaar dan wordt de Photon map op de harde schijf of in het geheugen bewaard voor evt. later gebruik.

De tweede straal
Bij dit proces wordt een schatting gemaakt van de helderheid en kleur van elke pixel van de uiteindelijke rendering aan de hand van de eerder gemaakte Photon map. Voor elke pixel wordt de scene afgetast (Ray Traced) totdat de dichtstbijzijnde oppervlakte in de buurt van het trefpunt is gevonden.

En daar wordt de bekende renderingsvergelijking ingezet om de straling te berekenen die het trefpunt verlaat in de richting van de straal die er bij de eerste keer op was neergestreken.

Er zijn vier afzonderlijke routines bedacht om dat proces te versnellen:
directe verlichting, specular reflectie, caustic en diffuse indirecte verlichting.

Om de direct verlichting zo goed mogelijk te benaderen wordt een straal vanaf het trefpunt naar elk van de lichtbronnen gestuurd. Indien daarbij geen andere objecten worden aangetroffen, dan zal de lichtbron met z'n eigenschappen uiteindelijk mede de helderheid bepalen.
Bij indirecte verlichting wordt de Photon map nagegaan om de straling uit te rekenen.

Specular reflectie kan met Ray tracing routines worden uitgevoerd omdat deze reflecties goed kunnen weergeven.

Caustic is een ander verhaal, deze wordt middels de afzonderlijk aangemaakte Caustic Photon map uitgerekend. Daarbij is het belangrijk dat het aantal lichtstralen ruim voldoende is omdat juist deze Photon map als enige zal worden benut om aan zijn gegevens te komen. Te weinig opgeslagen stralen geven een slechtere kwaliteit van het eindresultaat (onscherpte in bepaalde partijen en ruis).

Bij zachte indirecte verlichting wordt ook de andere Photon Map aangesproken. Deze behoeft echter lang niet zo nauwkeurig te zijn als die van de caustic waarvoor de Global Photon map wordt gebruikt.

Pixels berekenen aan de hand van de Photon map
Voor de bepaling van de helderheid en kleur op het trefpunt wordt een van de opgeslagen Photon maps gebruikt.

a.
Verzamel een bepaald n aantal photons die er het dichtst bij komen met behulp van de *nearest neighbor search* functie.

b.
Groepeer deze als een soort bol die n Photons bevat

c.
Deel voor elke Photon de hoeveelheid licht door het gebied van bestreken wordt en vermenigvuldig dat met de BRDF die op deze Photon van toepassing is

d.
De som van al deze resultaten per Photon geeft de totale oppervlakte straling weer die op de trefpunten in de richting van de straal die er op kwam.

Versnellingen
Optimalisaties van deze methode is nodig omdat het proces anders erg veel tijd vraagt om enige kwaliteit te kunnen leveren. En in 2011 - 2012 zijn er enkele nieuwe bijgekomen, die veelbelovend lijken in de beschreven deelgebieden.

En dat is meteen de algemene problematiek van een renderingsprogramma: onderzoeken bij universiteiten wereldwijd worden vaak uitgevoerd op één onderdeel van een deel van het probleem (kwaliteit versus renderingstijd). En bij het afstuderen is er meestal maar één doel waar naar toegewerkt dient te worden. De situatie beschrijven zoals die bestaat en de nieuwe situatie, beschreven in het proefschrift of dissertaat waarbij de nieuwe methode bijna altijd extra voordelen (moet) opleveren.
De praktijk van het omwerken van een deel algoritme naar programmeertaal en inpassen in een bestaand programma is veel complexer. Vaak vallen de eerder gemelde deelvoordelen weg of laten op een ander terrein een gat vallen. Vandaar dat het proces van publicatie en werkelijke integratie in een commercieel renderingsprogramma meestal meer dan 6 - 12 maanden duurt.

1.
Om zoveel mogelijk te voorkomen dat lichtstralen (Photonen voor het gemak genoemd) worden uitgerekend die uiteindelijk niet bijdragen (er buiten vallen) aan de uiteindelijke rendering wordt de richting van de uitgaande photonen vaak beperkt. In plaats van ze in het wilde weg in allerlei richtingen te sturen, worden ze gebundeld gestuurd in de richting van een bekend object van een gewenste Photon manipulator om het licht te bundelen of juist diffuus te maken. Ook het aantal in een bepaalde richting gestuurde Photons kan worden getuurd. Meer Photons in de buurt van het trefpunt versturen betekent een grotere Photon map, waar-

door er een onbalans kan ontstaan van de stralingssterkte ten opzichte van andere punten waar minder Photons naar toe werden gestuurd.

2.
Indien een oppervlak "Lambertian" is met zachte indirecte verlichting, kan Irradiance caching (stralingssterkte opslag) worden gebruikt om te middelen tussen de waarden van voorgaande berekeningen.

3.
Om onnodige trefpunt controles en testen te voorkomen bij directe verlichting kunnen speciale schaduw Photons worden toegepast.
Indien een Photon op een oppervlak komt wordt een schaduw Photon opgewekt die dezelfde richting opgaat als de richting van de eerste en door het object heen gaat. Het volgende object dat wordt aangedaan zal een schaduw Photon opleveren die in de Photon map wordt vastgelegd. Bij het uitrekenen van de directe verlichting wordt dan niet de gebruikte techniek vanuit het oppervlak naar het licht uitgevoerd, maar eenvoudig nagekeken in de Photon map voor de schaduw Photons. Als deze er niet zijn, dan zal het object een direct zichtverbinding hebben met de lichtbron; de overige berekeningen kunnen dan achterwege blijven.

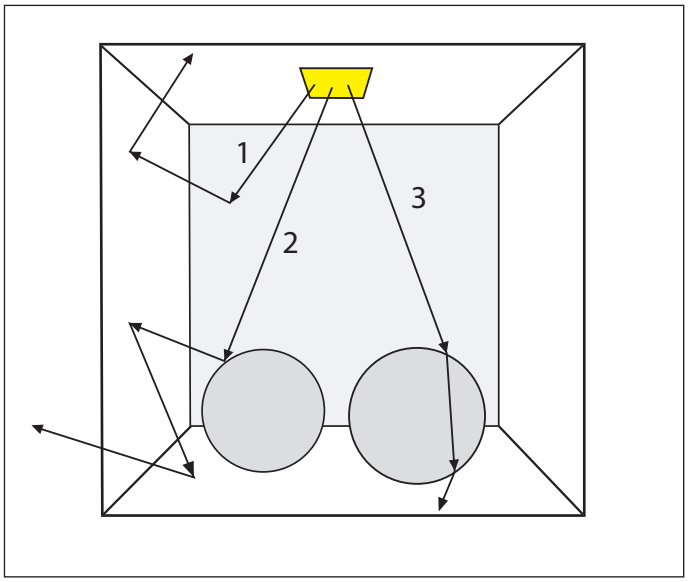
4.
Jensen adviseert het gebruik van een 'cone filter' in het bijzonder bij caustics. De lichtstralen worden daarbij gewogen en ingesteld afhankelijk van de afgelegde afstand tot het trefpunt. Daarmee kunnen scherpere beelden worden geproduceerd.

5.
Met behulp van een GPU raster routine is het mogelijk om bijna een real time afbeelding te produceren door het gebruik van de eerste en de laatste verstrooïng. (image space Photon mapping).

6. In de herhaalde berekeningen van de Photonen wordt soms gebruik gemaakt van opzoek tabellen met de hoek, richting in 3 dimensies of bol coördinaten. Daarmee wordt tijd bespaard die anders nodig zou zijn voor de altijd kostbare cos() en sin() functies.
Zie ook het uitstekende Siggraph paper.

[literatuur nr. 157]
course43sig02.pdf
A Practical Guide to Global Illumination using Photon Mapping, Siggraph 2002, course 43, july 2002. Henrik Wann Jensen, Stanford University

In de bekende 'Cornell Box' met een glazen bol rechts en een chrome bol links zijn er verschillende Photonen paden in beeld gebracht.
1. twee diffuuse reflecties met uiteindelijk tegen het plafond absorptie



Vrij naar afbeelding 2.3 van Course43sig02.pdf van Jensen.

2. een spiegelende reflectie die daarna twee maal een diffuuse reflectie oplevert

3. twee spiegelende overgangen gevolgd door absorptie op de grond.

Schaduwen vastleggen
Voorwerpen die schaduwen produceren zijn vanaf het begin een aandachtspunt geweest. Enerzijds levert een schaduw een diepte in het 2D model op, dat een betere benadering van het 3D model moet opleveren. Anderzijds is er in principe geen informatie beschikbaar, wat is er eigenlijk in de schaduw te zien en hoe halen we dat naar voren?

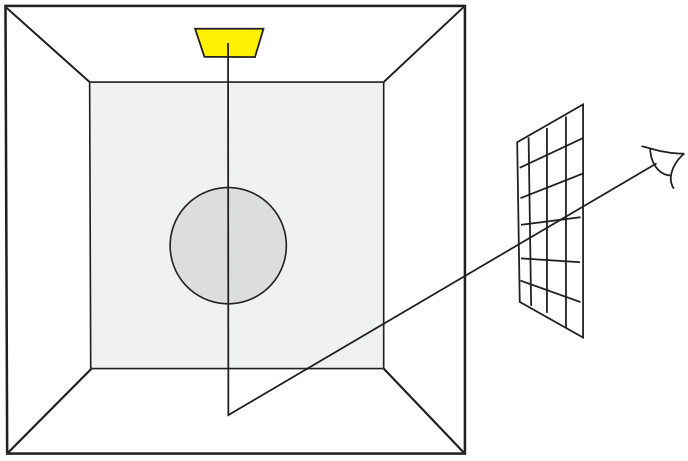
In Ray Tracing wordt een **schaduwstraal** in de richting van het licht gestuurd om te controleren of er naar de lichtbron toe een object wordt ontmoet. Indien de schaduwstraal geen object raakt zullen de gegevens van de lichtbron in de berekening worden meegenomen. Anders worden deze genegeerd.

Bij grote 'area light sources' moeten meerdere schaduwstralen worden weggestuurd om te bepalen waar de schaduw gedeelten zich bevinden. Met het gebruik van de Photon map kunnen schaduwstralen efficiënter worden toegepast zonder te veel extra berekeningen.

De benadering daarvoor is stochastisch, waardoor soms bewust schaduwen over het hoofd worden gezien bij kleine objecten.

Monte Carlo heeft het nadeel dat een aanzienlijke hoeveelheid ruis ontstaat zelfs met hoge Photon dichtheden door de grote afwijkingen van de hoge frequentie ruis die door caustic wordt veroorzaakt.
Bij **area lighting** en gecompliceerde geometrie, bv. bij licht dat door een bladerdak van een boom valt is de Monte Carlo methode niet altijd maatgevend. De integratie van verschillende delen van het model verloopt niet uniform.
K. Egan e.a. van de universiteit van Columbia heeft daar een oplossing voor gevonden:
Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders
[papers-first-pages-siggraph-2011.pdf - page 35]
Monte Carlo is prima geschikt voor productie renderingen vanwege zijn eenvoud. De integratie van de schaduwgedeelten geeft vaak een relatief zachte weergave van de schaduw, maar de samples zelf (schaduw stralen) hebben een hoge frequentie met een aanzienlijke hoeveelheid ruis zelfs bij hoge sample aantallen en tijd.
We denken hierbij onwillekeurig aan de rendering van een Maxwell Engine gebruiker die 139 uur besteede aan een animatie waarbij in een aantal ogenblikken van de animatie duidelijk vervelende ruis in de rendering te zien was.

http://www.gk.dtu.dk/photonmap link werkt niet meer, Technische universiteit Denemarken

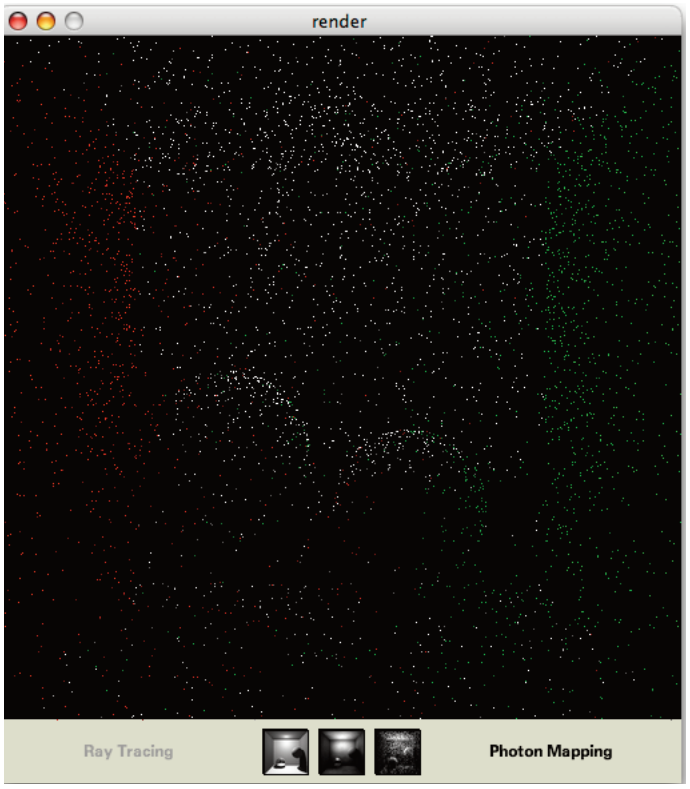


Schadow Ray:
is er een object tussen lichtbron en camera?

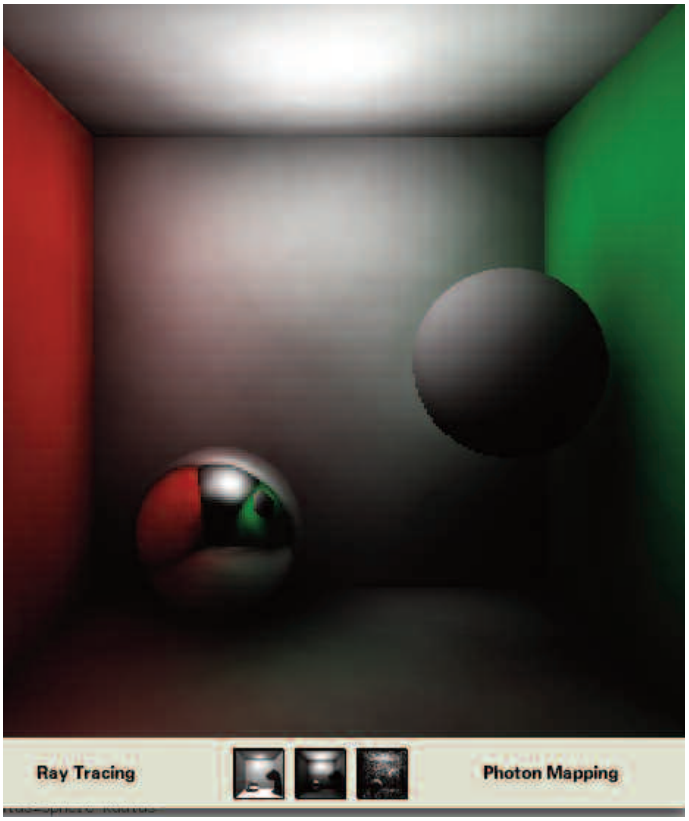
Render programma in Processing.
Met Ray Tracing, Photon mapping uitgeschakeld.

http://www.dtu.dk/

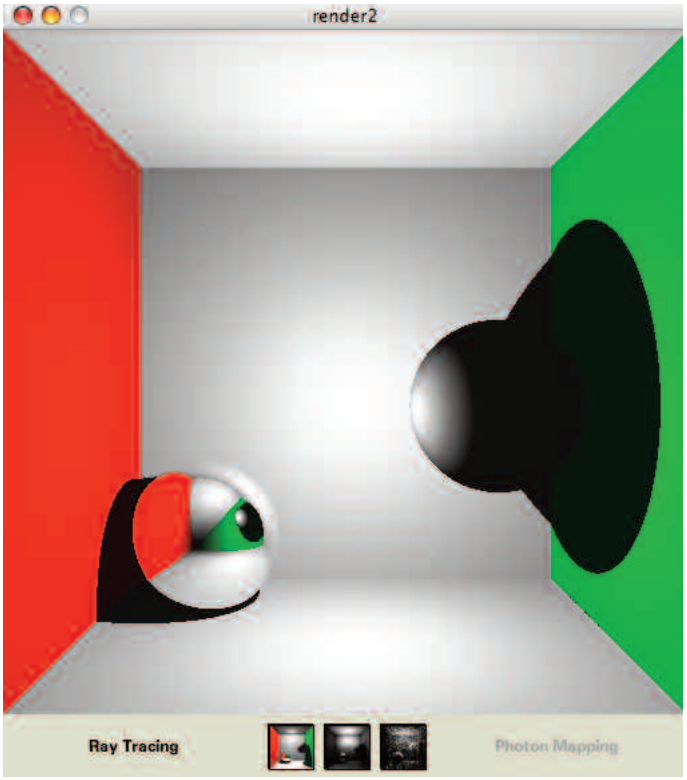
http://en.wikipedia.org/wiki/Ray_tracing
wikipedia ray tracing



Render programma in Processing. Corresponderende Photon Map.



Render programma in Processing.
Ray Tracing plus Photon Mapping.



Render programma in Processing. Ray Tracing.

Wiskundige aanduidingen

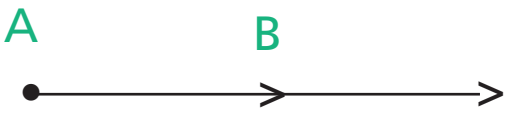
De pijlpunten geven aan dat deze lijn van links naar rechts oneindig doorloopt

Rays / lichtstraal

Een lichtstraal is een deel van een lijn met een vast eindpunt en die oneindig lang doorloopt naar de andere kant.

Een lichtstraal met één eindpunt

In wiskundige formules wordt een vector op een lijn aangeduid met een streep erboven plus een pijl boven de tweede letter.



Dit is een lichtstraal met een vector B
In de wiskunde aangeduid met

$\overrightarrow{AB}$

De vector start boven de letter A en passeert B

Opposite Rays

Er zijn ook tegengestelde stralen (opposite rays), de aanduiding daarvan maakt gebruik van dezelfde principes.
Het begin van de lijn is het startpunt van de vector en de pijl boven de letter het eindpunt van de vector. Alleen als de twee vectoren op één lijn liggen, maar tegengesteld zijn wordt over **tegengestelde stralen** gesproken.
Het startpunt Y is hetzelfde. De punten X, Y en Z zijn collineair.

We schrijven de vector tussen Y en Z als volgt

$\overrightarrow{YZ}$ $\overrightarrow{YX}$

Deze wijst naar rechts, precies 180° tegengesteld daaraan (Opposite Rays) van Y naar X.
Waarbij telkens het beginpunt een streep is boven de start letter en de pijl boven het eindpunt van de vector, de lijn zelf loopt wel door aan alle beide kanten < >.

Lijn segment (lijnstuk)

Een lijnstuk bestaat uit twee eindpunten. In de afbeelding zijn dat A en B.
Een lijn segment is dus een deel van een 'normale' lijn.
Om aan te geven dat het hier om een afgebakende afstand gaat wordt dat vaak aangegeven met een horizontaal streepje boven de vector.

$\overline{AB}$

Een lijn wordt onderverdeeld door twee punten in een lijn segment. Hetgeen onderdeel uitmaakt van de complete lijn.
A en B zijn de eindpunten van het lijnsegment met alle punten daartussenin.

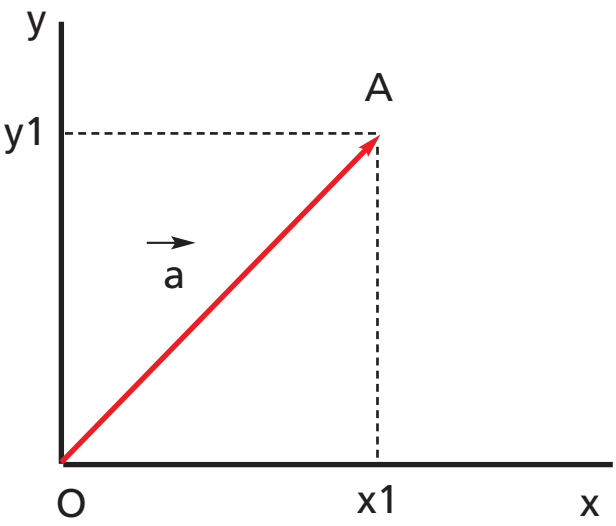
$\overline{AB}$

De vijf axioma's van Euclides

1. Twee punten kunnen slechts door één rechte lijn worden verbonden.
2. Elke lijn kan eindeloos worden uitgebreid in een nog langere lijn.
3. Bij een punt en lijnsegment die op dat

punt begint, is er een cirkel met het punt als middelpunt en het lijnsegment als straal (halve diameter) van de cirkel.
4. Alle rechte hoeken zijn gelijk aan elkaar.
5. Als een punt buiten een lijn staat, is er maar één lijn, die de punt snijdt zonder de eerste lijn te snijden.

Polygoon is een gesloten stelsel lijnsegmenten die samen een plat vlak omsluiten.
Veelhoekig basisfiguur waaruit binnen 3-programma's driedimensionale voorwerpen worden opgebouwd.



Plaatsvector
In het platte 2D vlak gaan we uit van het bekende xy-assenstelsel. Y postief omhoog, X positief vanaf de oorsprong naar rechts.

De vector $\vec{a}$ wordt bepaald door de oorsprong O en het punt A.
A = (x1, y1)
De x1 en y1 zijn de coördinaten van de van de vector of A = (a1, a2)
De lijn van O naar A heet de drager van de vector a.
Er zijn twee manieren waarop de vector kan worden geschreven:
als kolomvector

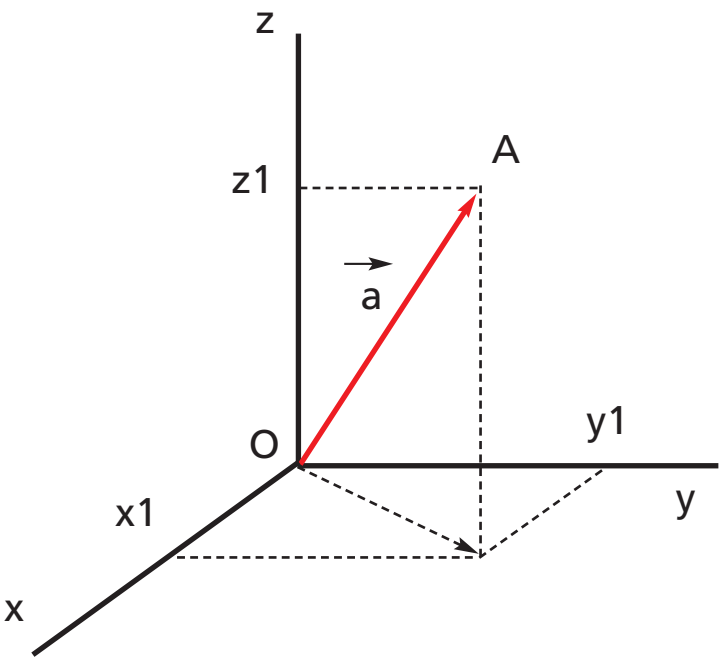
$$\vec{a} = \begin{pmatrix} x1 \\ y1 \end{pmatrix}$$

als rijvector

$$\vec{a} = (x1, y1)$$

De verzameling van vectoren wordt aangegeven met a1, a2 $\mathbb{R}^2$

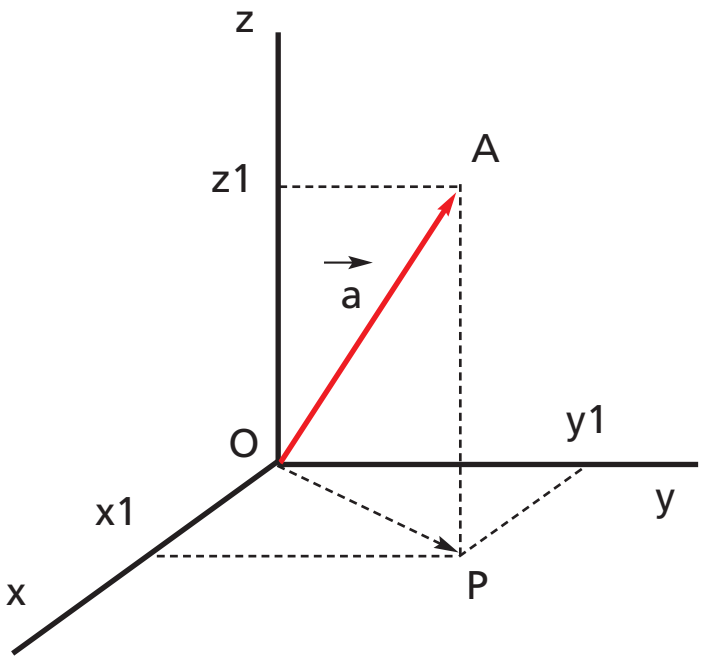
Vectoren in een 3D vlak, zoals bij renderprogramma's worden voorgesteld in een assenstelsel met drie onderling met elkaar verbonden assen die 90° op elkaar staan.
De Z omhoog, de X naar de kijker toewijzend en de Y horizontaal naar rechts. Dit is een rechtsdraaiend assenstelsel. De vingers van de rechterhand draaien vanaf de x-as in de richting van de y-as (beiden positief) tegen de klok in, waarbij de duim de positieve z-as aangeeft. Er zijn ook andere systemen in omloop.



De aanduiding van een 3D-vector is overeenkomstig de schrijfwijze van de 2D-vector. En dat geldt ook voor de verdere rekenwijzes bij drie dimensionale ruimte. Ook bij een vier of meer dimensionale ruimte komen dezelfde formules en wetmatigheden voor, maar de tekening met meerdere ruimtes wordt moeilijker.

$$\vec{a} = \begin{pmatrix} x1 \\ y1 \\ z1 \end{pmatrix}$$

$$\vec{a} = (x1, y1, z1)$$



De verzameling van vectoren wordt aangegeven met a1, a2, a3 $\mathbb{R}^3$

De lengte van de vector $\vec{a}$ is zowel bij de tweede dimensie, maar ook bij de derde dimensie (render scene) belangrijk. Het is de afstand van de lichtbron naar een object of van de camera naar een object of lichtbron.

De absolute lengte van de vector wordt aangegeven met twee verticale streepjes en in Amerikaanse literatuur ook wel met twee dubbele streepjes.

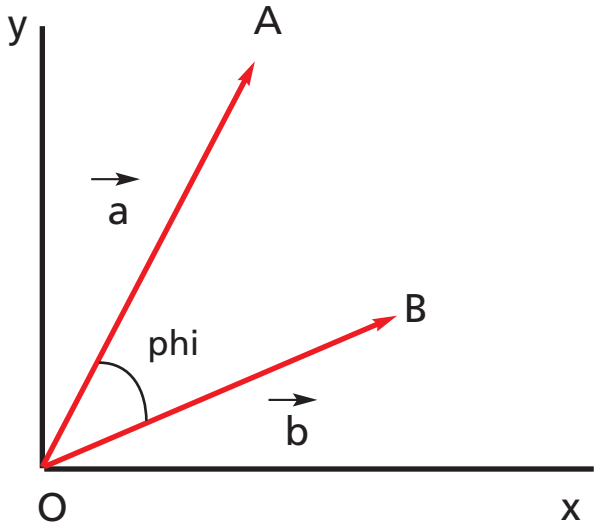
Bij een 3D vector is de afstand van de vector

$$|a| = \sqrt{x^2_1 + y^2_1 + z^2_1} \qquad \mathbb{R}^3$$

In het twee dimensionale vlak is de vector P

$$|p| = \sqrt{x^2_1 + y^2_1} \qquad \mathbb{R}^2$$

Een belangrijke formule die bij renderprogramma's wordt toegepast:



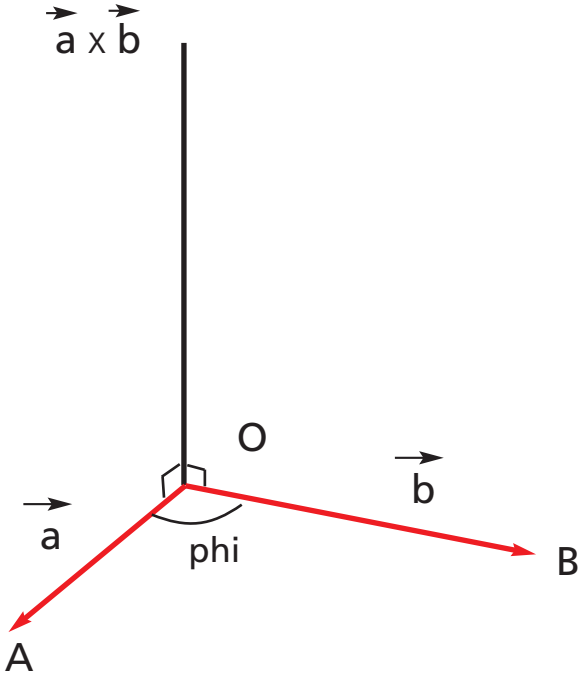
$\vec{a} \bullet \vec{b} = |a| \ |b| \cos(phi)$

Het inwendig product (*dot product*) van twee vectoren (a , b) is gelijk aan de lengte van a maal de lengte van b maal de cosinus van de ingesloten hoek. Voor de 3de en 4de dimensie is de hoek (phi) nog net voor te stellen. Voor hogere dimensies kan dat niet meer. Indien a • b = 0 dan staan de twee vectoren a en b loodrecht op elkaar. De uitkomst van het inwendig product is een getal (scalar) en dus geen vector.

Het kruisproduct (*cross product*) is niet in de 2de dimensie mogelijk, alleen bij de derde is deze te gebruiken. De uitkomst van het kruisproduct (ook wel uitwendig product, of kortweg uitproduct genoemd) is wèl een vector. De notatie bestaat uit een "x" teken tussen de twee vectoren:

$\vec{a} \times \vec{b} = |a| \, |b| \sin(\text{phi})$

Ook deze variant komen we in renderingsprogramma's tegen.



Indien we de uitkomst van het kruisproduct op c stellen volgt:

$\vec{c} = \vec{a} \times \vec{b} = |a| \, |b| \sin(\text{phi})$
waarbij geldt dat de nieuwe vector $\vec{c}$ loodrecht staat op zowel a als b. De lengte van c wordt ook wel zo aangeduid:

$\vec{c} = |a \times b| = |a| \, |b| \sin(\text{phi})$

Ook hier geldt de rechterhandregel. Leg de vingers zo dat A wordt bereikt en vervolgens B. De duim wijst dan in de richting van de crossproduct.

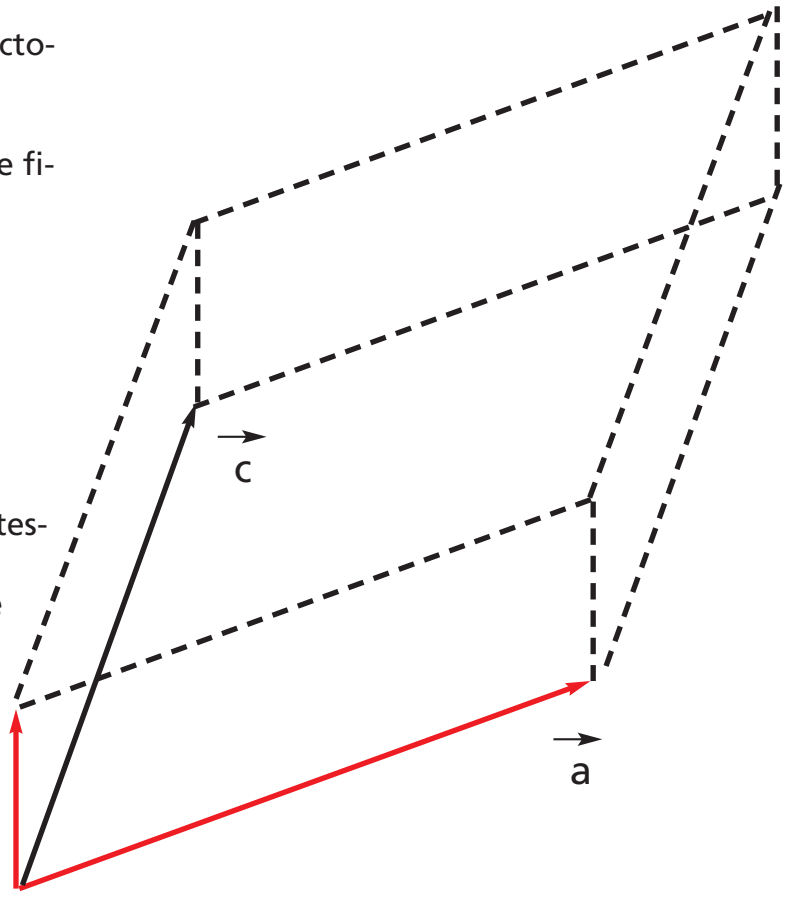
Indien $\vec{c} = \vec{a} \times \vec{b} = 0$ dan lopen de twee vectoren a en b parallel langs elkaar.

De oppervlakte van het drie dimensionale figuur is gelijk aan:

Oppervlakte = $|\vec{a} \times \vec{b}|$

En de inhoud door:

Inhoud = $|\vec{a} \bullet (\vec{b} \times \vec{c})|$



Het Dot product wordt toegepast om te testen dat vectoren 90° op elkaar staan, het product is dan nul. Als ze in verschillende richtingen wijzen dan is het Dot product -1. En in dezelfde richting + 1. Het meest belangrijke is het uitzoeken van de hoek tussen twee vectoren. Of het berekenen van de hoek tussen een vector en het gebruikte coördinaten stelsel.

Dimensies

Dimensie is een aspect van een object dat onafhankelijk van andere aspecten kan worden beoordeeld. Bijvoorbeeld een punt is een object dat geen eigenschappen heeft behalve dan een plaats (coördinaten).
Het aantal getallen dat de plek van een punt vastlegd komt precies overeen met de dimensie van de ruimte waar de punt aanwezig is.

Een punt in een twee dimensionale ruimte heeft twee coördinaten die de plaats vastleggen, een x en een y (x, y).
In de drie dimensionale ruimte wordt dat (x, y, z).
Denk aan de coördinaten van een punt als een graad van vrijheid, omdat ze vrij zijn om onafhankelijk van elkaar te wijzigen.

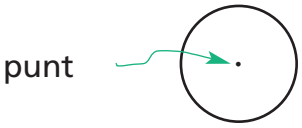
Een punt zelf is eigenlijk een nul dimensionaal object, als een ruimte. Alleen deze is nu beperkt tot één mogelijke plaats met nul graad van vrijheid.

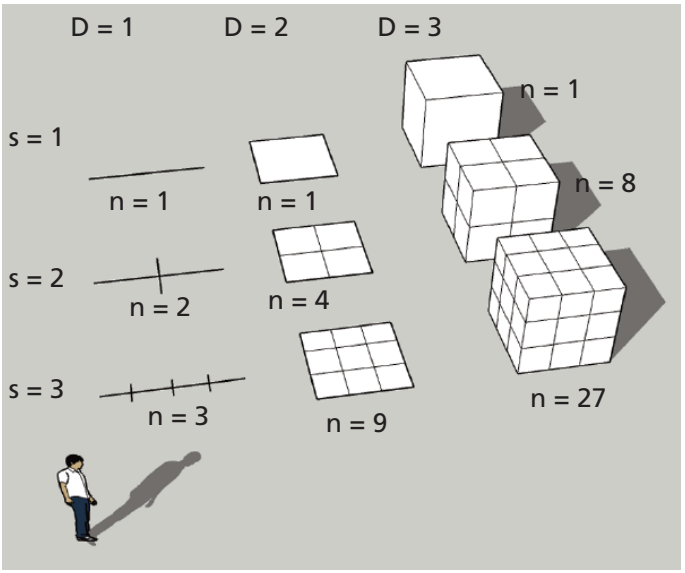
Door een punt in één-dimensionale ruimte te brengen wordt deze eenvoudig in de eerste dimensie geschoven.
Twee punten vormen een lijnstuk, die op zijn beurt verschoven kan worden tot een oppervlak in een twee dimensionale ruimte.

Verschuiven we een vlak in de hoogte dan wordt een vlak in de drie dimensionale ruimte gebracht, het vierkant is nu een kubus geworden, hetgeen op zijn beurt weer deel uitmaakt van een groter drie dimensionale ruimte. Elk hoekpunt heeft nu drie coördinaten om deze vast te leggen een x, y en z waarde.

Het weergeven van een drie dimensionale figuur op bv. een beeldscherm wordt als een soort illusie beschouwd. De derde as kan nu eenmaal niet loodrecht op de beide andere assen worden weergegeven op een twee dimensionaal scherm. Meestal zien we dat alleen als een schuine lijn. Hogere dimensionale objecten kunnen dus in een lagere dimensie vervormd overkomen!

Een drie dimensionale kubus kan worden ge-





$$n = s^{D}$$

$$D = \frac{\log (n)}{\log (s)}$$

volgens Hausdorff (1919)
s = schaalfactor
d = dimensie
n = aantal gelijke kopieën

D = 2 en n = 9 (2de dimensie) naar 3-d:
schaalfactor = 3 -> s x n = 3 x 9 = 27 kubussen

De Grande Arche in Parijs in het stadsdeel La Défance staat een soort weergave van een hyperkubus. Von Spreckelsen (Deense architect) ontwierp deze in 1982.

projecteerd op een 2 dimensionaal vlak als een oppervlak. De schaduw is dus eigenlijk veroozaakt door een kubus die van draad is gemaakt.

Elk van de 8 hoekpunten (vertices) wordt op het 2-dimensionale vlak geprojecteerd. De projectie van een 3D vlak is dus één dimensie minder dan het originele object. Met een zaklantaarn of een lichtbron uit een render programma kan dat aanschouwelijk worden gemaakt.

Een stap verder is om de kubus naar een vier dimensionale ruimte te schuiven door deze in een richting loodrecht op de x, y en z as te verplaatsen. De richting is in een 2-dimensie tekening niet meer aan te geven.

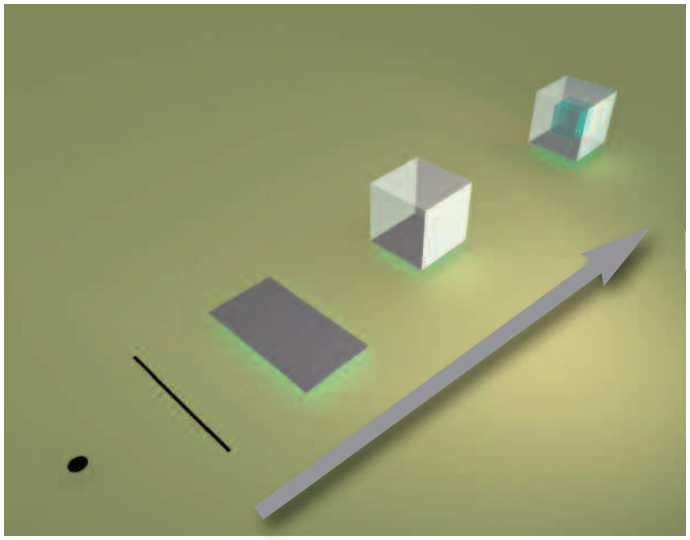
Een dergelijk object heet in de wiskunde hypercube, welke deel uitmaakt van een grotere vier-dimensionele ruimte. Een hoekpunt heeft hierbij vier coördinaten (x, y, z en w assen). De bewegingsrichting van een drie-dimensionele ruimte naar een vier-dimensionale ruimte kan worden voorgesteld door de rechts-links of omhoog-omlaag bewegingen in een lagere dimensie.

Sommige wiskundigen geven deze vier dimensie richtingen aan met "ana" en "kata" van het Griekse woord voor "in" en "out". Indien u bv. in de x-as naar rechts schuift dan komt dat overeen met 'ana' in de w-as richting.

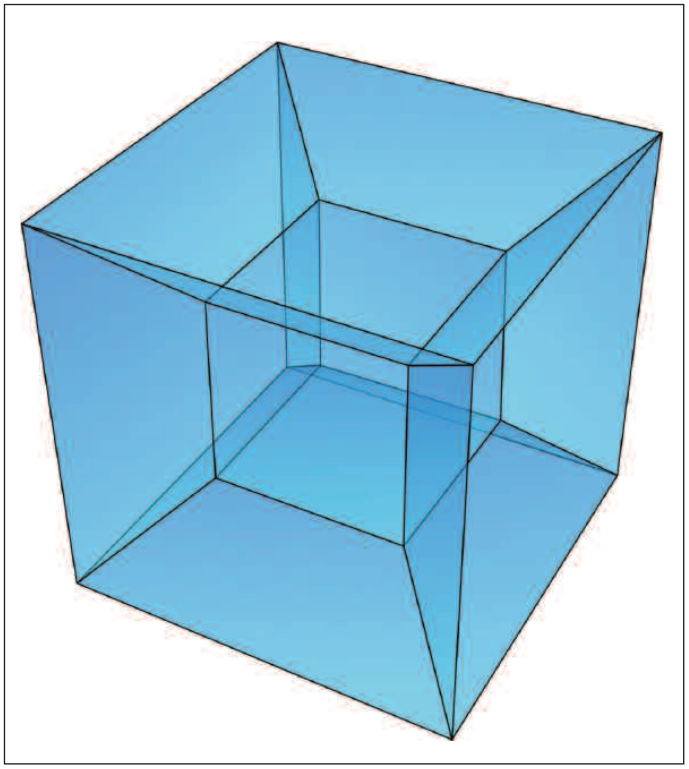
De bijbehorende vlakken in een 4-dimensie ruimte zijn als volgt:
xy, yz, xz, xw, yw, zw,

Vier dimensies zijn niet ongebruikelijk in wiskundige formules en codes voor renderprogramma's. De vierde staat dan vaak op 1.

Maar dat kan ook worden gebruikt voor de tijdfactor bij het maken van render animaties.



Van links onder naar rechts boven:
Punt
Lijn segment
Oppervlak
Kubus
Hyperkubus



Bovenstaand: projectie van een Tesseract

Hyperkubus is een voorwerp in een n-dimensionale ruimte:

n = 0 punt
n = 1 lijn
n = 2 oppervlak
n = 3 kubus
n = 4 Tesseract

Punten, vectoren en normalen

Het principe van renderen is dat een denkbeeldige lichtstraal het 3D model aftast en daarbij direct optekent op welke plek de kleur en helderheid waarneembaar is. Kort door de bocht want het maken van een renderingsprogramma gaat uit van een een groot aantal wiskundige formules om te weten te komen waar de trefpunten met het 3d model zijn en waar de lichtbronnen en de vergelijkbare schaduwdelen in het 3d vlak zichtbaar zijn.

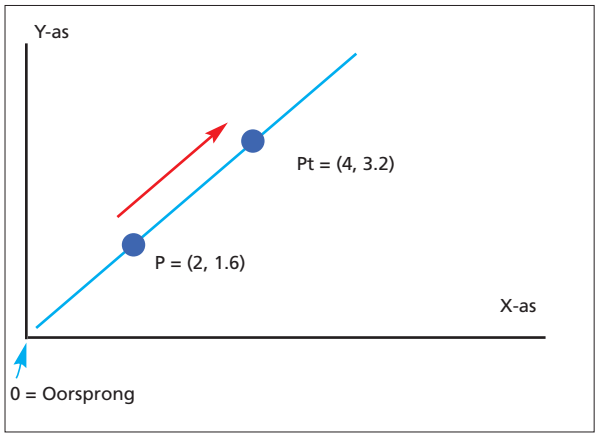
Om dat hele proces van aftasten te kunnen doen is ondermeer kennis van het werken met vectoren, punten, normalen, 3d coördinaten, matrixen en perspectief modellen nodig.

Het is niet de opzet van deze uitgave om dat in detail te behandelen, als daar al iemand in geïnteresseerd zou zijn. Het is bijna altijd strikt geheime informatie, die alleen beschikbaar is bij programmeurs van commerciële renderingsprogramma's. En marketing mensen laten weinig los, of weten het zelf niet. Firma's verschuilen zich vaak achter hoogdragende termen met telkens weer nieuwe gigantische snelheids- en kwaliteitsverbeteringen om de markt maar gewillig te maken het product te kopen. Met techniek en echte technische achtergrond heeft dat weinig van doen. Maar via de achterdeur van OpenSource kunnen we wel degelijk een kijkje nemen in diverse onderdelen van renderprogramma's. Twee OpenSource programma's zijn bv. SunFlow en POV-Ray, waarvan de complete broncode met een simpele tekstverwerker is te openen (zie literatuur).

Een PUNT
Een punt is een plaats in een twee of drie- of hogere dimensionele ruimte.

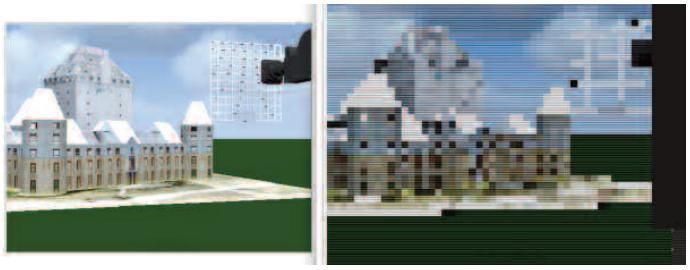
Een punt P kan worden gedefinieerd door bv.
P (x1, y1, z1) - 3 dimensies
Waarbij x1 de x waarde vanaf de oorsprong is, y1 eveneens vanaf de oorsprong en z1 ook. x en y liggen in hetzelfde horizontale vlak (haaks op elkaar) en de z vector wijst omhoog.
Punten kunnen op allerlei manieren worden aangepast middels een translate
P -> Translate -> Pnieuw

Een veel voorkomende omzetting is die waarbij een punt langs een bepaalde lijn wordt verschoven, dit wordt translatie genoemd.



En in wezen is dat het gehele renderingsproces. We starten een lijn bij de oorsprong (het oog) en trekken een lijn door een van de vele vakken van het gezichtsvenster, het 3D model in. De blauwe punten kunnen als photonen worden voorgesteld die zich langs de rechte lijn bewegen. Totdat een object wordt geraakt. Het trefpunt.
Bij lichtstralen (bv. met een zaklantaarn in het donker) zien

Vanuit de camera gaan de stralen door het venster naar het 3D model toe.



Links de Preview voor het renderen. Rechts het renderingsproces in volle gang, waarbij de grote pixels langzamerhand steeds kleiner worden en de rendering scherper oogt.

we direct waar een object wordt geraakt. Bij 3D modellen is dat van te voren onvoorspelbaar en zal er een proces van aftasten aan vooraf gaan om dat te weten te komen. De trefpunten (intersection) zijn essentieel, daarvandaan vertrekken er weer nieuwe lijnen het 3D model naar volgende objecten of naar lichtbronnen. Het aftasten kost bijzonder veel tijd in het hele renderingsproces, vandaar dat juist daar vele 'versnellingen' zijn bedacht om dat te verbeteren. De meest optimale versnelling is die waarbij een rendering in een onderdeel van een seconde kan worden weergegeven, maar dat gaat helaas vaak samen met een inkrimping van de kwaliteit, afbeeldingsgrootte en extra verfijnende mogelijkheden.

Coördinaten stelsels
Er zijn een aantal systemen die vaak worden toegepast in renderingsprogramma's, waarbij het standaard lineaire stelsel het meest bekend is, daarnaast wordt het bol coördinaten stelsel toegepast waarbij een punt wordt bepaald door een hoek vanuit de x-as berekend, een afstand en een hoogte z.
Het omrekenen van het ene stelsel in het andere gebeurt in afzonderlijke subroutines en komt vrijwel bij alle te berekenen stralen terug.

Vector
Een vector kan worden weergegeven door een lijnstuk van een bepaalde lengte die in een bepaalde richting wijst, aangeduid met een pijlpunt. Indien de vector genormaliseerd is

dan betekent dat deze een standaard lengte van 1 heeft meegekregen.

Vector lengte

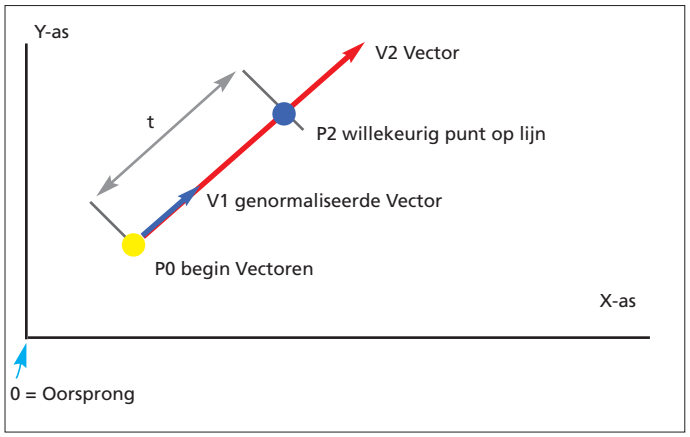
Een vector kan worden gezien als twee punten met daartussen een lijnstuk, waarbij een van de punten is voorzien van een pijl: de richting. In de wiskunde wordt de absolute waarde (lengte) van een vector aangegeven tussen twee vertikale lijntjes (absolute waarde). In engelse literatuur ook wel door 2 x 2 vertikale lijntjes geplaatst.
In twee dimensies is het berekenen van de lengte een vrij simpele zaak, maar ook in drie dimensies blijkt het opvallend eenvoudig. Zelfs in meer dan drie dimensies indien gewenst kan van dezelfde structuur van de formule worden uitgegaan.

Eenheids vector

In het engels 'Unit vector', deze heeft een standaard lengte van 1. Het 'normaliseren' van een bestaande vector in een eenheidsvector gebeurt als volgt:

$$\hat{V} = \frac{V}{|V|}$$

Bereken eerst de lengte van de vector (onder de streep in de formule). Deel de vector (lengte en richting) door deze lengte eenheid. "V met dakje" is de eenheidsvector.
Een Normaal is een technische term voor een loodlijn recht vanuit een oppervlak bekeken. En haaks op de oppervlakte zelf. Een Normaal is ook een vector met een lengte en richting, maar ze worden afzonderlijk behandeld omdat de lengte geen rol speelt, alleen de richting waar het vlak naar toewijst is belangrijk. Normalen spelen ondermeer een belangrijke rol bij het weergeven van schaduw in het renderingsproces.



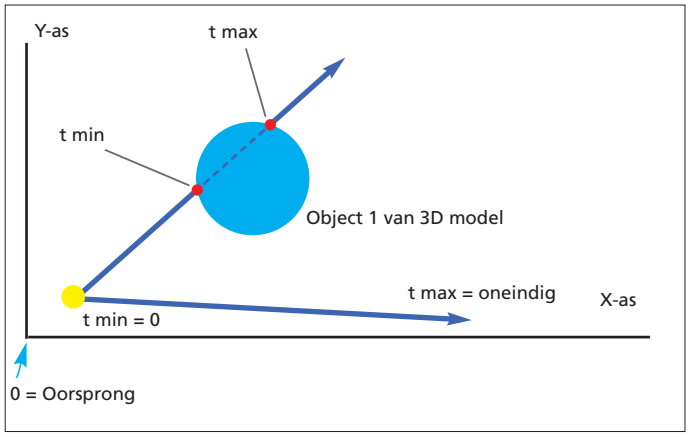
Lichtstralen kunnen worden voorgesteld door vectoren. Het is in wezen een lijn met een beginpunt (P0) en een richting plus een grootte. Starten we met de V1 vector dan heeft deze een beginpunt (P0) en een lengte van 1 plus een richting naar rechtsboven. De afstand die het punt P2 heeft afgelegd richting rechtsboven is gelijk aan t. Deze tijdfactor geeft in wezen het verloop van de fotonen aan vanaf het begin, het 3D model in. Indien P2 zo ver is opgeschoven tot de max. waarde van V2 is bereikt dan is dat t_{max}.
Geven we een lichtstraal aan met de letter L dan volgt (niet wiskundig) daaruit:

L einde van V2 = L begin + L richting x t max

Een ander punt op deze lichtstraal volgt uit:

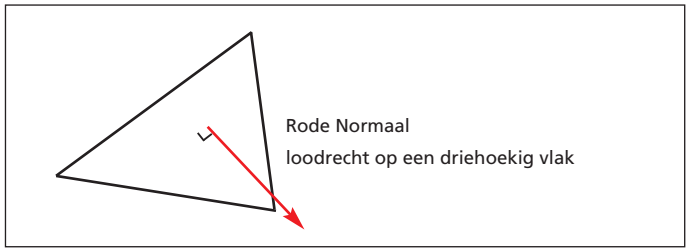
P3 (t) = P begin + V1 richting x t waarbij $t <= t_{max}$

Het willekeurig gekozen P2 punt komt dus overeen met P2 = P0 + t x V1



Bij het berekenen van de lichtstralen gaan we uit van $t_{min} = 0$, dat is het begin van de lichtstraal. Bij render programma's wordt vaak uitgegaan van twee tijdwaarden (momenten waar een denkbeeldige straal zich op dat moment bevindt) "t_{min}" en "t_{max}". De eerste straal vanuit $t_{min} = 0$ bereikt op t min een object (intersection point). De afstand tussen $t_{min} = 0$ en t min is belangrijk voor de berekening tijdens het renderen. Het volgende trefpunt wordt t max aangeduid. In een aantal programma's zijn deze waarden ook handmatig in te stellen. Als er binnen t_{min} en t_{max} één of meerdere trefpunten zijn, dan wordt dat als trefpunt vastgelegd in het geheugen. Als t echter kleiner is dan t min waarde. Of als t groter is dan t_{max} dan zal er geen trefpunt zijn (zie de onderste blauwe straal in de afbeelding hierboven). Bij het renderingsproces vanuit een lichtbron zullen er veel stralen in het geheel niet voldoen aan deze twee gestelde grenzen en verloren gaan voor de uiteindelijke rendering. Echter het kost wel processortijd.
In een renderprogramma wordt t min vaak op 0 of een heel kleine waarde ingesteld en t max op een heel hoge waarde, zodat ook bij grote 3D scenes voorwerpen op soms tientallen meters of zelfs honderden toch worden meegenomen in de rendering. Staat de instelling van t_{max} te laag dan zullen alle objecten die verderweg (in de tijd) zijn gelegen niet zichtbaar worden.

In de programmeer codes worden niet alleen de trefpunten opgeslagen ook het omgekeerde (inverse) van de richting en het teken van elke coördinaat van de richtingsvector uitgerekend en tijdelijk opgeslagen. Deze getallen komen later van pas bij het verder verwerken van de straal.



Dot Product

Ook wel 'scalar product', 'inner of 'in-product' (inwendig product) genoemd wordt bijzonder veel toegepast bij een renderingsprogramma. In wezen is het de projectie van de ene vector op de andere zie afbeelding op de volgende pagina.

Bijzonder is wel dat het antwoord van het Dot product een reël getal R is en geen dubbele bodem, zoals bij een normale vector met lengte en richting in één vector.

A • B = R

De aanduiding van een **Dot product** gebeurt met een dikke stip halverwege de letters

A • B of als < a, b>

De berekening die wordt uitgevoerd bestaat uit het vermenigvuldigen van elk element van vector A met vector B, waarbij de som van beide producten het uiteindelijke Dot product oplevert:

A • B = A.x * B.x + A.y * B.y + A.z * B.z (3 dimensies)

A • B = A.x * B.x + A.y * B.y (2 dimensies)

De * geeft hier het maalteken aan.

Het Dot Product is verbonden met de hoek tussen twee vectoren volgens de vergelijking:

$A \bullet B = |A|\,|B| \cos \alpha$

aangezien de cos van een 90° hoek gelijk is aan 0 zal de vergelijking gelden:

$A \bullet B = |A|\,|B| \cos 0° = 0$

A • B = 0 als de 2 vectoren loodrecht op elkaar staan

De volgende vergelijkingen zijn ook geldig:

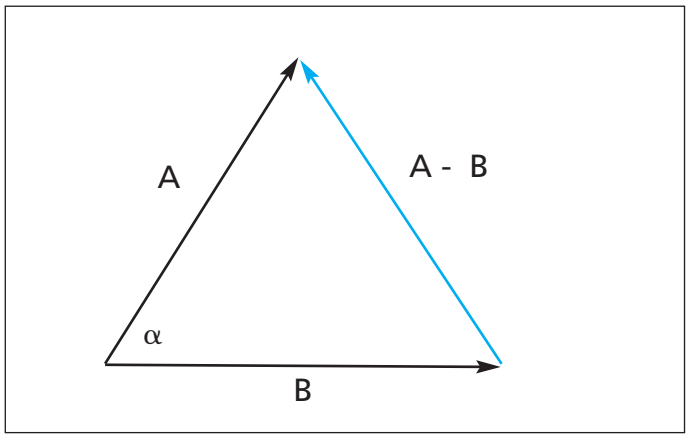
A • B = B • A

(dA) • B = d (A • B)

A • (B + C) = A • B + A • C

$A \bullet A = |A|^2$

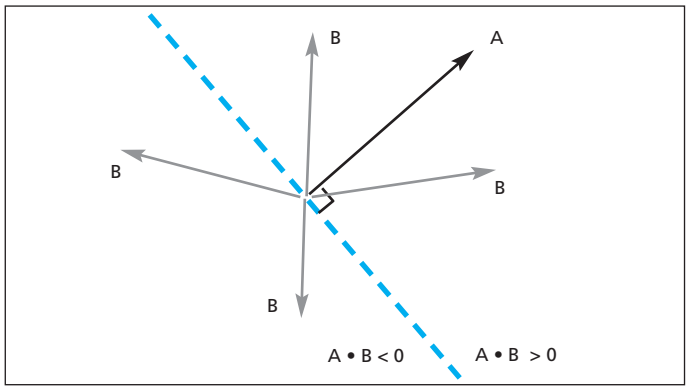
Het DOT product is belangrijk bij het renderen omdat het resultaat overeenkomt met de cosinus van de hoek tussen twee vectoren. Verder wordt DOT product gebruikt om te onderzoeken of twee vectoren haaks op elkaar staan en in welke richting ze wijzen.



Het Dot-product is gekoppeld met de hoek tussen de twee vectoren (hier A en B). De vergelijking van een Dot product is:

$A \bullet B = |A|\ |B|\ \cos \alpha$

Maar een Dot-product van twee vectoren vertelt nog meer, afhankelijk van het teken van de uitkomst. Daarmee wordt bepaald of twee vectoren aan dezelfde kant liggen òf aan de tegengestelde kant van een vlak.



Java Code voor Vector Dot Product

```
double dot (sfvec3f other) {
  return (x * other.x) + (y * other.y) +
  (z * other.z) ;
}
```

C++ Code voor Vector Dot Product

```
Double sfvec3f::dot(sfvec3f * other) {
  return (x * other-> x) + ( y * other -> y)
  + ( z * other -> z) ;
}
```

C++ code voor Intersection punt

```
double intersect(const Ray &r) const { // re-
turns distance, 0 if nohit
Vec op = p-r.o; // Solve t^2*d.d + 2*t*(o-
p).d + (o-p).(o-p)-R^2 = 0
double t, eps=1e-4, b=op.dot(r.d), det=b*b-
op.dot(op)+rad*rad;
if (det<0) return 0; else det=sqrt(det);
return (t=b-det)>eps ? t : ((t=b+det)>eps ? t
: 0);
```

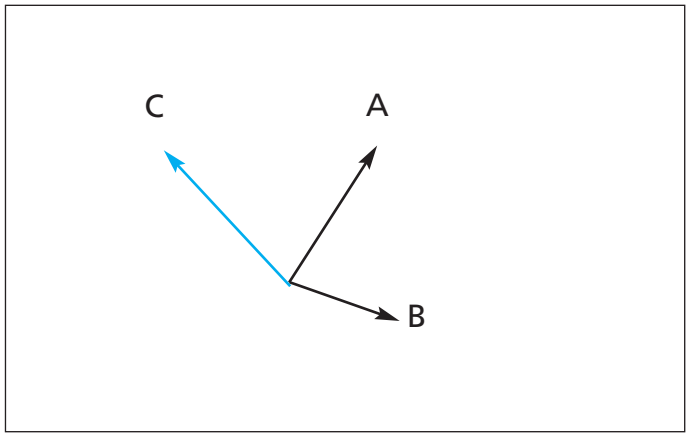
Kruis product

Ook bij dit product (*cross product*) gaat het om twee vectoren, maar de uitkomst bij een kruisproduct is een nieuwe vector, dit in tegenstelling met het Dot product dat een getal (scalar) oplevert.

A x B = C

Het bijzondere bij het kruisproduct is dat de nieuwe vector C loodrecht op de andere twee vectoren staat. Dat betekent dan ook dat een kruisproduct alleen met drie of meer dimensies kan worden gebruikt. Gegeven twee vectoren in 3D levert als kruisproduct:

A x B een nieuwe vector op die loodrecht op de eerste twee staat. Van de definitie van het kruis-



product kunnen we sin (phi) berekenen van de ingesloten hoek tussen A en B.

| A x B | = | A | | B | sin (phi)

Daar volgt uit dat het kruisproduct van twee loodrechte eenheidsvectoren zelf ook weer een eenheidsvector oplevert.
Het kruisproduct wordt regelmatig in render programma's toegepast alhoewel minder dan het Dot product. Zo kan er bv. worden getest of twee vectoren parallel lopen A x B = 0 (verbasterde vector). Verder is de oppervlakte van een parallelogram met kruisvector te berekenen.

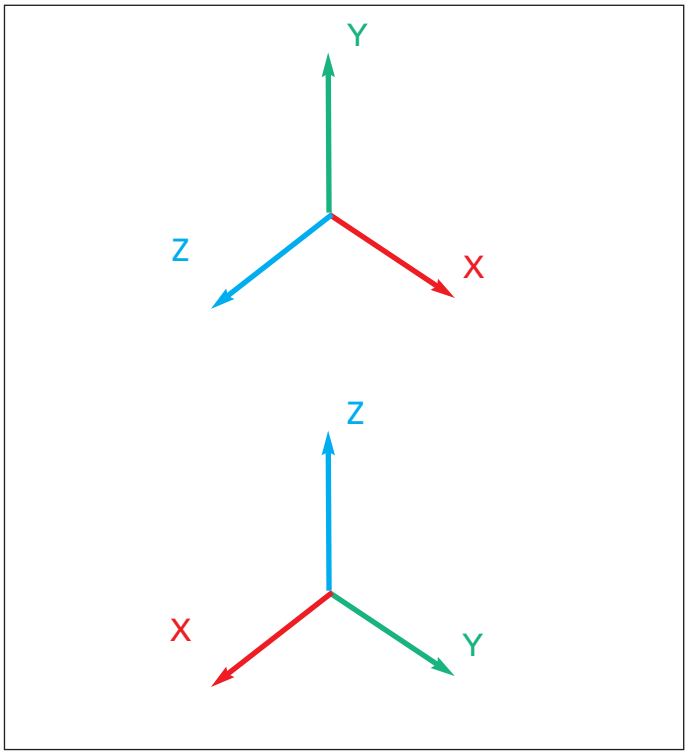
Assenstelsels

Bij geometrie wordt de derde dimensie weergegeven door drie kleuren:

blauw = z-as
rood = x-as
groen = y-as

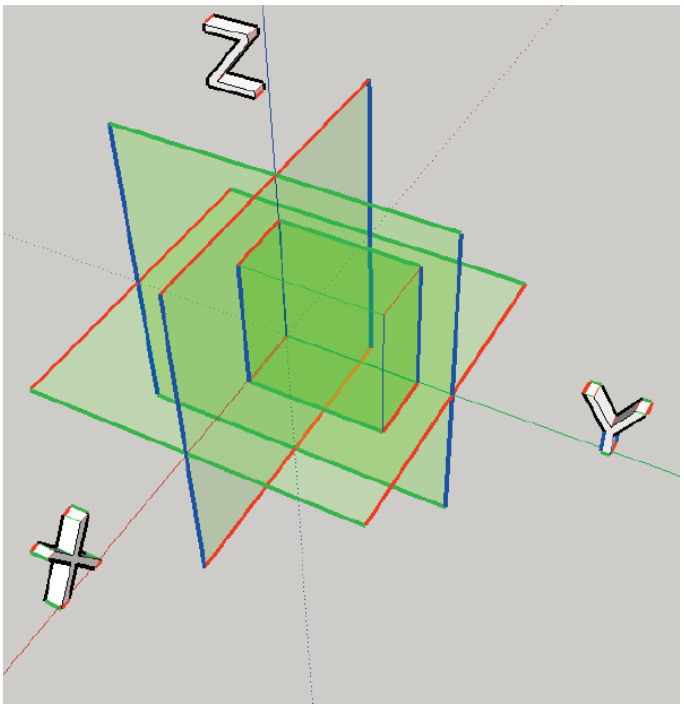
Maar de manier waarop deze op papier worden gezet en worden gebruikt kan sterk verschillen.
Indien we de naar boven gerichte z-as als voorbeeld aanhouden dan zal de x as eerst komen en daarna de y-as met een 'rechter-hand coördinatie systeem'. Bij een linkerhand coördinatie systeem is dat juist andersom. Hou de rechter duim in de richting van de omhoog wijzende vector de wijsvinger geeft dan de x-as aan en de wijsvinger de y-as.
Maar de z-as wordt niet altijd naar omhoog weergegeven.
John Ambrose Fleming bedacht de linker- en rechter handregel om de twee afspraken vast te leggen. Maya en OpenGL gebruiken het rechterhand coördinatie systeem.
Maar Open GL gebruikt ook het linkerhand systeem, het is net hoe het wordt ingevoerd. Het omzetten van het ene coördinatie stelsel in het andere kan met matrixen gebeuren bv. matrix A (1, 1, -1);

afb. beide rechterhand coördinatie stelsels, waarbij b) uit de wiskunde komt. Omdat de coördinaten stelsels in het renderingsprogramma zitten ingebakken is er weinig verwarring mogelijk. Dat wil zeggen dat de conversieslag van een bepaald merk 3D programma naar renderingsprogramma (vaak een plug-in of import omzetting) wel met deze verschillen rekening moet worden gehouden.



Cartesisch coördinatenstelsel (Orthogonaal coördinatenstelsel)
De meest gebruikte methode voor het aangeven van het assenstel is de onderste variant. De Z-as heeft een blauwe kleur als referentie van de lucht. Dit wordt bv. toegepast in SketchUp. Het wordt ook wel wereldcoördinatenstelsel genoemd.

In computer graphics en de programma's wordt echter vaak uitgegaan van de bovenste variant waarbij de Y-as de groene as is en de Z-as omhoog wijst.
Beide stelsels zijn RECHTSHANDIG.
Duim wijst naar Z-as, de vingers van de links draaiende richting eerst de x-as en dan de y-as.
Indien in de onderste tekening de Y en X met elkaar wisselen spreken we van Linkshandig (minder in omloop).



Dot product in OpenGL/ES
Dot product van twee vectoren
float dot (genType x, genType y);
x, y de eerste en tweede vector

$$\begin{pmatrix} x[1]\cdot y[2]-y[1]\cdot x[2] \\ x[2]\cdot y[0]-y[2]\cdot x[0] \\ x[0]\cdot y[1]-y[1]\cdot x[1] \end{pmatrix}$$

Cross
vec3 cross(vec3 x, vec3 y);
waarin x de eerste van de twee vectoren is.
cross geeft (zie schermafb) het volgende weer:

De lengte van een vector
float length (genType x);
met uitkomst zie schermafdruk

$$\sqrt{x[0]^2 + x[1]^2 + \ldots}$$

literatuur: **OpenGL-ES-2_0-Reference-card.pdf**
OpenGL ES 2.0 API Quick Reference Card.
ES 3.0 is van augustus 2012.

Lichtstralen (rays)
Een lichtstraal is een eenvoudig lijn segment dat wordt bepaald door één punt (oorsprong van de straal, ook wel endpoint genoemd), een richting EN een parametrische waarde t.
De formule die kan worden gebruikt in parametrische vorm is een functie van de scalar waarde t, waarbij het aantal punten die de straal opbouwd (uitvouwd of ontwikkeld)

$$r(t) = o(r) + t\,\mathbf{d}(r) \qquad 0 < t < \text{oneindig}$$

** < met streepje eronder 2 x

Met behulp van de t min en t max wordt de oneindige lijn beperkt in het aantal punten. Deze velden worden als *mutable* verklaard (ze kunnen worden gewijzigd, zelfs als de lichtstraal structuur constant is.

```
mutable Float min t, max t;
```

Voor het simuleren van bv. bewegings onscherpte wordt elke straal een unieke eigen tijdwaarde meegegeven.
min t wordt meestal niet gelijk aan 0 ingesteld, maar krijgt een kleine waarde mee, waardoor foutieve zelf-trefpunten bij de beperkingen van de reken nauwkeurigheid worden verminderd. Er is geen waarde te vinden waardoor ze geheel worden voorkomen. Het getal voor min t moet klein genoeg zijn om de eerste 3D objecten in beeld te krijgen en groot genoeg om de meeste nauwkeurigheids problemen voor te zijn. Maar zoals het vaak gaat is er wel altijd wel een 3D scene te bedenken die juist met de gekozen constante niet, of beperkt resultaat oplevert. Vandaar dat diverse andere algoritmes dit zo veel mogelijk moeten zien te voorkomen.

De algemene code voor een punt in een bepaalde plaats langs de lijn:

```
Point operator ()  (Float t) const { return o + d * t ; }
```

```
R einde = R oorsprong + R richting * tmax
```

of in het algemeen:

```
R(t) = R o + R richting * t       met t  <= t max
```

Een straal kan worden voorgesteld door:

O + Rt

waarin O overeenkomt met de Oorsprong van de straal en R de richting. t staat voor een reël positief of negatief getal of zelfs in de buurt van 0. Door het simpel veranderen van de factor t kunnen we elk punt maken op de lijn, waarbij de richting hetzelfde blijft.

Dit is in wezen de manier van berekenen van een lichtstraal vanuit de camera door het pixel venster naar het 3D model toe.
Elke plaats op de straal kan worden berekend door een R normalized te vermenigvuldigen met een factor t.

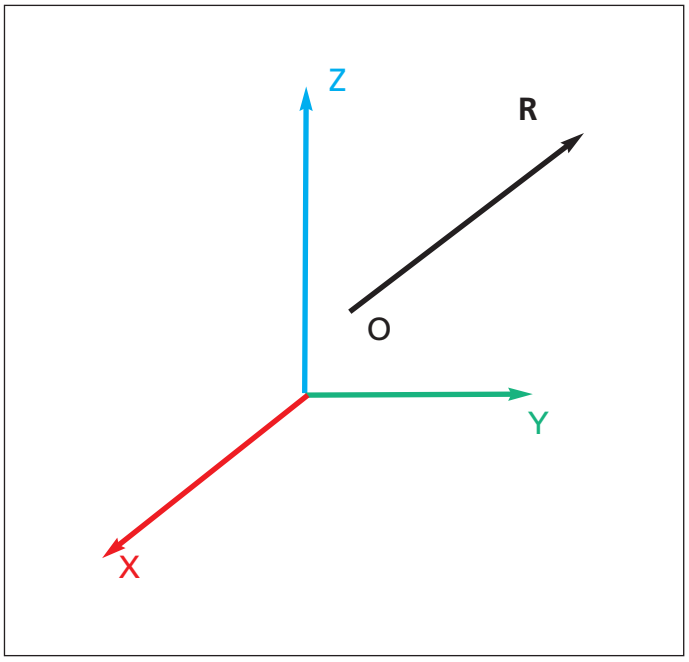
In een Ray tracer is t altijd groter dan 0.

Java code voor vector Cross Product

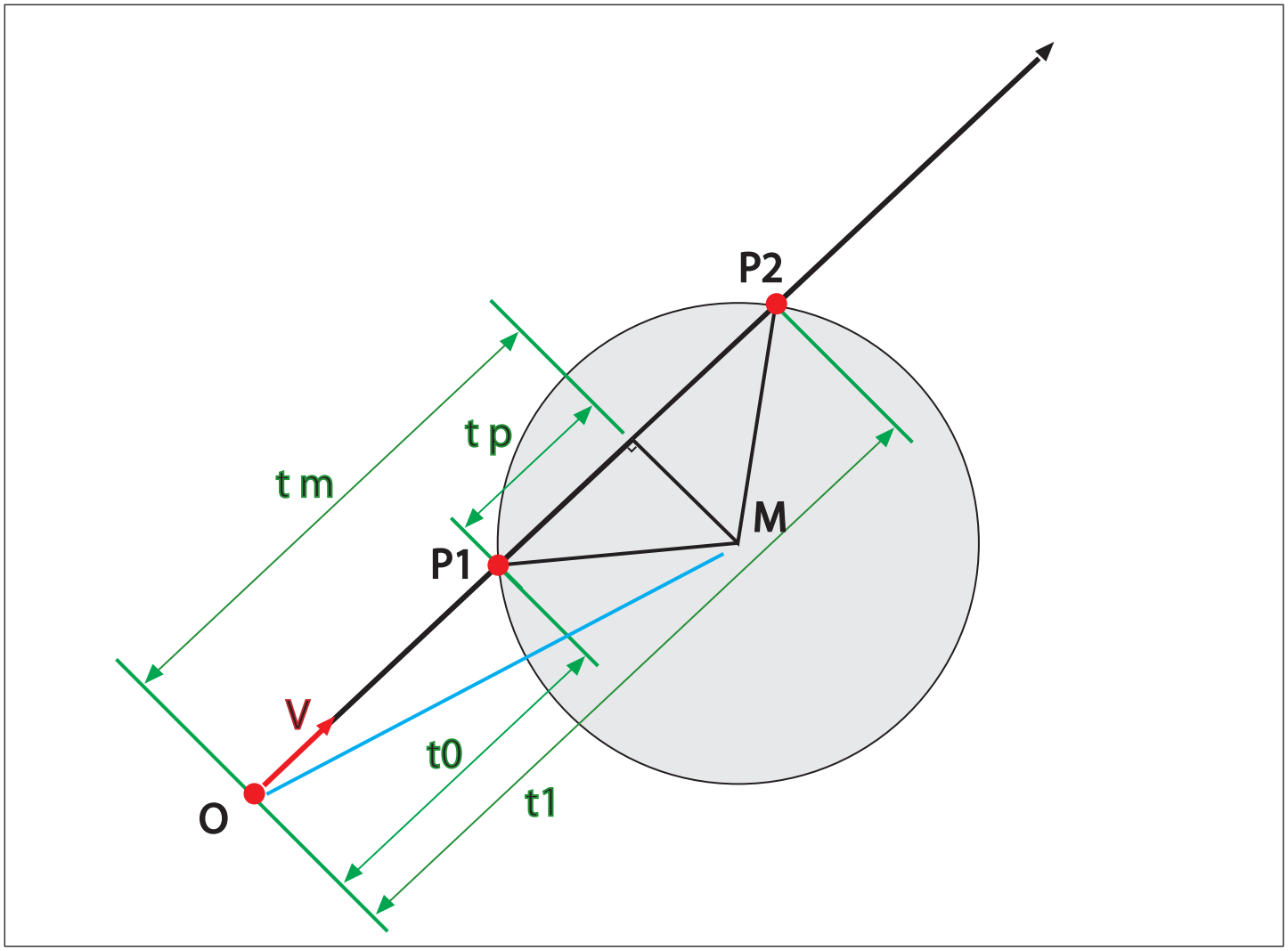
```
Void cross(sfvec3f other) {
  double xh = y * other.z - other.y * z;
  double yh = z * other.x - other.z * x;
  double zh = x * other.y - other.x * y;
  x = xh;
  y = yh;
  z = zh;
}
```

C++ code voor vector Cross Product

```
Void sfvec3f::cross(sfvec3f* other) {
  double xh = y * other->z - other->y * z;
  double yh = z * other->x - other->z * x;
  double zh = x * other->y - other->x * y;
  x = xh;
  y = yh;
  z = zh;
```



Een lichtstraal is een half oneindige lijn die door zijn oorsprong en richting wordt bepaald.



Een eerste lichtstraal (primary ray) kan bv. een bol raken, waarbij er twee trefpunten zijn P1 en P2, zie afbeelding.

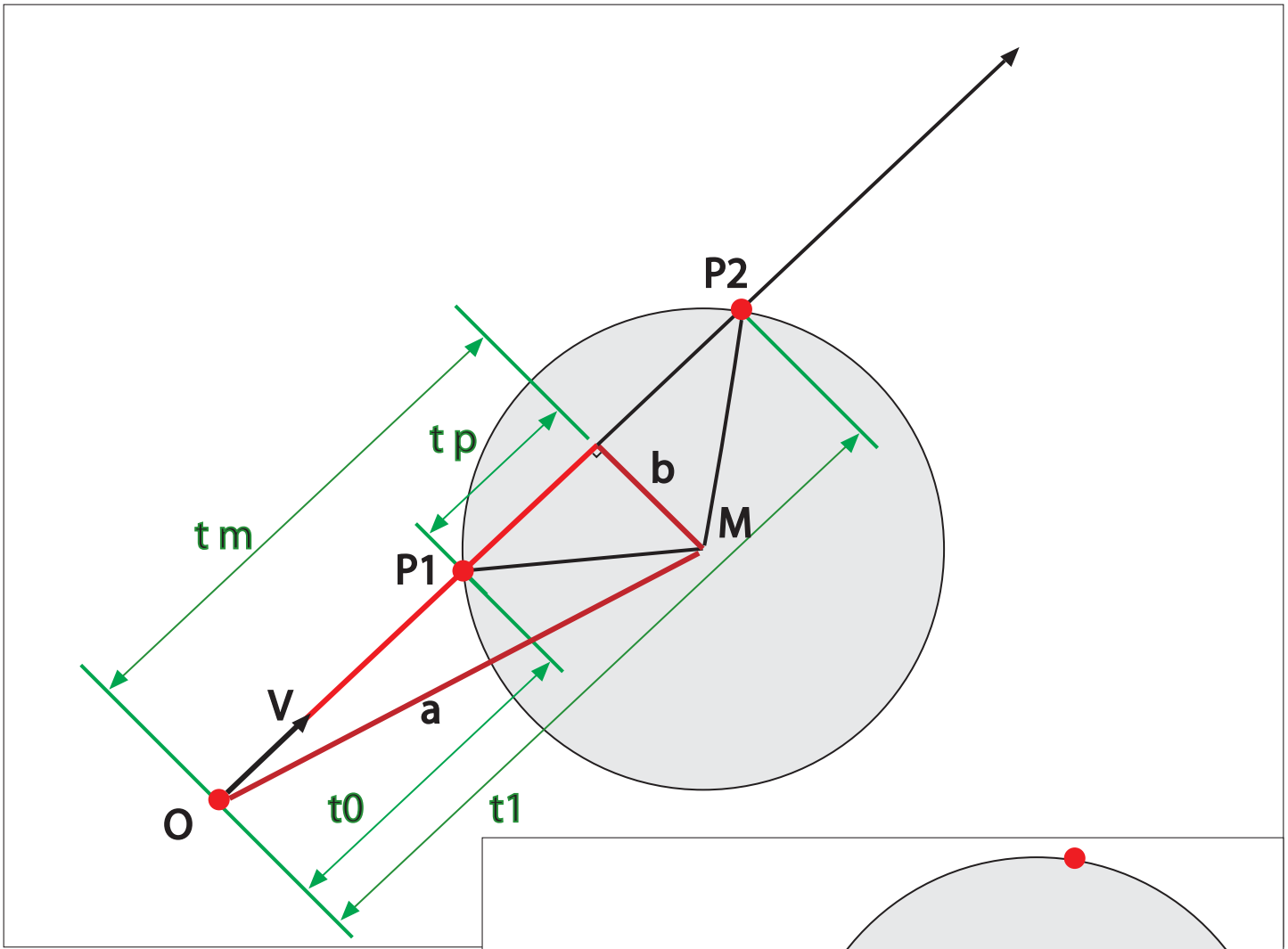
Punt P1 vertegenwoordigd dan de t min waarde en P2 de t max waarde. Indien er een treffen is met een 3d object tussen de grensen van tmin en tmax dan kunnen we deze berekenen. Valt het trefpunt binnen t min dan is er geen trefpunt, maar ook als t max waarde wordt overschreden wordt er geen trefpunt meer
t max wordt vaak op een zeer hoog getal gezet en t min meestal iets boven de nul wordt ingesteld.

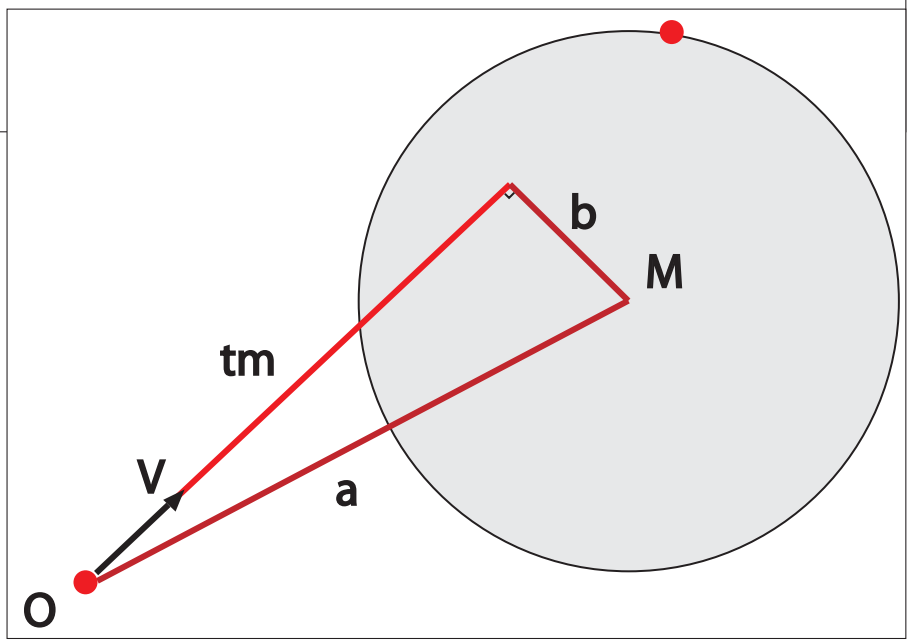
Het trefpunt (intersection)

Dit is een van de meest cruciale punten in het hele renderingsproces. Met een zaklantaarn ‘s avonds schijnen levert een bundel op waar we in het schijnsel een bepaalde 3D scene kunnen aftasten. Maar een programma werkt niet met een zaklantaarn maar met formules, algoritmes en programma codes.

Op welke manier is het mogelijk om een trefpunt met een object in het 3D model te berekenen?

Aan de hand van een simpel 2D model met een cirkel en een lichtstraal wordt dat stap voor stap bekeken. De lichtstraal kan worden voorgesteld door O + t V. De beginpositie (Oorsprong) plus de tijd t maal de vector (meestal eenheidsvector). Door het aanpassen van t kan het punt op de lijn naar boven bewegen. Met t waarde die positief is wordt vanuit de oorsprong naar het object ‘gekeken’. Bij P1 wordt een object (de cirkel) bereikt.





waarbij uit de tekening blijkt
t0 = tm - tp
verder t1 = tm + tp

De loodlijn vanuit het middelpunt van de cirkel deelt het lijnstuk P1 en P2 precies doormidden. De vraag is hoe we aan *tm* en *tp* komen.

P1 = O + t0 V
P2 = O + t1 V

Daartoe kijken we naar de driehoek die wordt gevormd door: tm , blauwe lijn en loodlijn vanaf middelpunt M. Bovenstaande tekening is de driehoek rood gekleurd. Rechts een detail daarvan. Bekend is dat bij de rechthoek

$b^2 + a^2 = tm^2$ ->

$tm^2 = b^2 + a^2$

a is bekend (coördinaten van de oorsprong en het middelpunt M van de cirkel zijn bekend). tm is echter niet bekend. De hoek ingesloten bij O:
cos (phi) = tm / a of te wel tm = a cos (phi).
cos (phi) is de scalar product van de twee vectoren
Het dot product van a en r :
a • r = |a| |r| cos (phi) schrijven we dit anders op:
a cos (phi) . |r| = a • V
tm . |r| = a • V
tm = a • V
levert de waarde van tm op
Er kan alleen een trefpunt zijn als tm een positieve waarde heeft. Met een negatieve waarde zou het trefpunt achter

de startpositie van de straal zijn, hetgeen van geen belang is.
if (tm < 0) return false

tm en a zijn nu bekend.
a = M - O
tm = a • V
De tweede rechthoek wordt gevormd door tp, b en de straal van de cirkel r. De straal van de cirkel is bekend.
$a^2 = b^2 + tm^2$

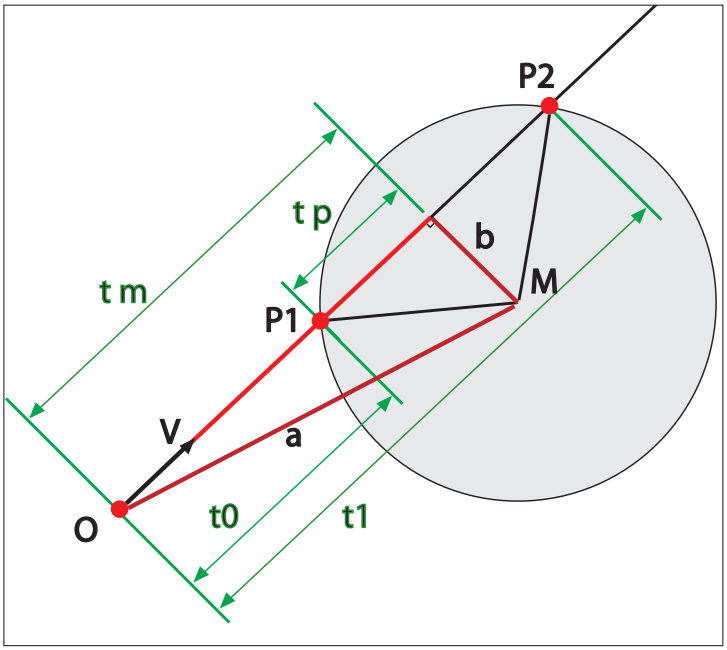
$$a = \sqrt{b^2 - tm^2} = \sqrt{b \bullet b - tm \bullet tm}$$

indien (a < 0) return false
Indien a groter is dan de straal van de cirkel gaat de lichtstraal langs de cirkel en is er dus geen trefpunt.

$b^2 + tp^2 = r^2$

$$tp = \sqrt{r^2 - b^2}$$

t0 = tm - tp
t1 = tm + tp



Trefpunten vinden deel 2
Er zijn andere methoden waarop de trefpunten met een bol kunnen worden gevonden. Een daarvan gaat er van uit dat de algemene vormen (polygonen) kunnen worden voorgesteld door een algebraïsche vorm.
De lichtstraal kan worden voorgesteld door
O + t r [1]
Waarbij O de oorsprong is van de lichtstraal en r de eenheidsvector en t de tijdfactor.

Al eerder zagen we de lijn met y = ax + b
een cirkel wordt omschreven als
$x^2 + y^2 = R^2$

Een de bol in 3 dimensies is daar een simpele uitbreiding op:

$x^2 + y^2 + z^2 = R^2$
$x^2 + y^2 + z^2 - R^2 = 0$
Elk punt op de bol voldoet aan deze formule dus ook een willekeurig punt P op de bol (kan een trefpunt zijn).
De coördinaten van een punt P op een bol kunnen worden ingevuld met x, y, z
de vergelijking wordt daardoor:
$P^2 - R^2 = 0$ [2] (impliciete functie)

Vullen we [1] in [2] in dan volgt:
$|O + t\,r|^2 - R^2 = 0$
en
$O^2 + 2\,ort + rt^2 - R^2 = 0$ [3]
Deze vergelijking voldoet aan de algemene vierkantsvergelijking
$f(x) = ax^2 + bx + c$
de tijdfactor t is hierbij x

en $a = r^2$
$b = 2\,or$
$c = o^2 - r^2$

http://nl.wikipedia.org/wiki/Vierkantsvergelijking

Met de bekend veronderstelde oplossing van de vierkantsvergelijking (abc formule) zijn er een aantal mogelijkheden:

$D = a^2 (x1 - x2)^2 = b^2 - 4\,ac$

waarbij x1 en x2 complexe wortels van de oplossing zijn.

* Als D > 0 dan zijn er twee verschillende reële oplossingen
* Als D = 0 dan zijn er twee gelijke reële oplossingen x1 = x2, de wortel = - b / 2a
* Als D < 0 geen oplossingen, de straal raakt de bol niet maar schiet er voorbij.

Programma code op volgende pagina

De Shape Class voorziet in twee afzonderlijke intersection routines. De eerste Shape::Intersect() geeft door of een enkele lichtstraal trefpunt overeenkomt met de eerste intersectie volgens de ingestelde min t en max t waarden.

De andere routine
Shape :: IntersetP ()
is een predicate functie die bepaald of er wel of geen trefpunt optreedt, zonder dat er details van het trefpunt zelf worden doorgegeven.
Sommige vormen kunnen een goede en efficiënte invulling voor IntersectP() geven zonder ze eigenlijk te berekenen.

Intersection algoritme tips

1. De Ray structuur bevat Ray::min t en Ray:: max t variabelen waarbinnen het lijnsegment (zie vooraan deze uitgave hoodstuk over lijnen, vectoren en rays) is vastgelegd. Routines behoren alle andere trefpunten zoveel mogelijk te vermijden die niet langs dit segment lopen.

2. Als een trefpunt wordt gevonden dan dient de parametrische waarde van de afstand in een pointer t_hitp te worden opgeslagen die op zijn beurt aan de trefpunt routine wordt doorgegeven. Als er meerdere trefpunten aanwezig zijn dan dient de dichtstbijzijnde te worden aangehouden.

3. Informatie over een intersectie positie wordt in DifferentialGeometry
structuur opgeslagen. Waarmee de locale geometrische eigenschappen van een oppervlak worden gevangen. Deze wordt veel gebruikt met lrt
en dient om het geometrische gedeelte van de ray tracer van de schaduw- en verlichtingsdelen te scheiden. Zie inzet 1 met uitleg.

4. De stralen die in de trefpunten routine wordt doorgegeven is in World Space. Vormen zorgen voor omzetting naar Object Space indien dat nodig is voor de intersectie testen. De geometrie die wordt afgeleverd dient in World Space coördinaten te zijn.
De Shape Class voorziet in standaard methoden voor de intersect routines die een foutmelding opleveren als ze aangeroepen worden.
Alle Shapes die '*true*' afleveren van Shape::CanIntersect() moeten daar gebruik van maken. Diegene die '*false*' afleveren kunnen doorgesluisd worden naar 'lrt' om deze routines niet te gebruiken indien de vorm geen trefpunt oplevert.
Indien het volledig virtuele functies zouden zijn, dan zou elke vorm die geen trefpunt oplevert, een vergelijkbare standaard functie moeten bezitten.

Inzet 1

Bijna alle Ray Tracers gebruiken dit algemeen toepgepaste principe om geometrische informatie over trefpunten met oppervlakken vast te leggen.

Als optimalisatie zullen velen de trefpunt informatie gedeeltelijk opslaan en dan alleen als een trefpunt wordt gevonden. Dus slechts datgene opslaan wat hoognodig is. Zodat de rest van de scan voorgang kan vinden.

Deze manier van werken bespaart tijd als een dichtbijzijnd trefpunt op een later tijdstip wordt gevonden bij een ander oppervlak.

De extra berekeningen om deze informatie ook vast te leggen valt in het niet en voor render programma's met complexe scenes die data management algoritmes gebruiken zal deze aanpak wellicht mislukken, omdat het oppervlak niet langer in het geheugen aanwezig is.

Geheugen is zoals altijd een kostbaar goed, enerzijds vanwege de beperkte schrijf- en leessnelheid van normaal RAM geheugen, anderzijds vanwege de schaarste die er bijvoorbeeld bij GPU kaarten aanwezig is. Een enkele GPU kaart uitgezonderd, die vele duizenden Euro's kost heeft max. 6 GB geheugen. Bij veel GPU render programma's dient het hele 3D model in een van de grafische kaart geheugens te passen.

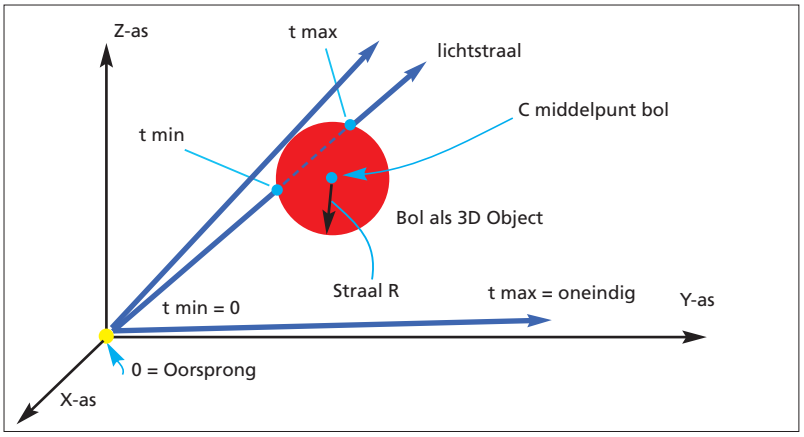
Uitleg op de volgende pagina

Java code intersection:

```
void intersection(sfvec3f rayStart,sfvec3f rayDirection,sfnode nearestNode,sfvec3f distance) {
        sfvec3f c = getSphereCentre().clone();
        c.sub(rayStart); // c=centre of sphere relative to start of ray
        double r = getSphereRadius(); // r=radius of sphere
        double distanceToSphere = Math::sqrt((c.x * c.x) + (c.y * c.y) + (c.z * c.z));
        if (r> distanceToSphere) { // the start point is already inside the sphere so the distance to it is zero
                nearestNode.setNode(this);
      distance.set(0.0);
                return;
        }
        double vx = rayDirection.x * distanceToSphere;
        double vy = rayDirection.y * distanceToSphere;
   double vz = rayDirection.z * distanceToSphere; // v = end point of ray when projected to sphere orbit
        double distanceFromEnd=Math::sqrt(((c.x - vx)*(c.x - vx)) + ((c.y - vy)*(c.y - vy)) + ((c.z - vz)*(c.z - vz)));
        if (distanceFromEnd > distanceToSphere) return; // the distance of the centre is greater than the distance to the beginning so its pointing away
        if (R < distanceFromEnd) return; // The distance of V from C is greater than R so its outside
        double d=distance.getValue();
        if ((d < 0.0) | (d > distanceToSphere - R)) {
        distance.set(distanceToSphere - R); // return distance to the nearest point
                nearestNode.setNode(this);
        }
}
```

C++ code intersection:

```
void sphereBean::intersection(sfvec3f* rayStart,sfvec3f* rayDirection,sfnode* nearestNode,sfvec3f* distance) {
        sfvec3f* c = getSphereCentre()->clone();
        c->sub(rayStart); // c=centre of sphere relative to start of ray
        double r = getSphereRadius(); // r=radius of sphere
        double distanceToSphere = Math::Sqrt((c->x * c->x) + (c->y * c->y) + (c->z * c->z));
        if (r> distanceToSphere) { // the start point is already inside the sphere so the distance to it is zero
                nearestNode->setNode(this);
      distance->set(0.0);
                return;
        }
        double vx = rayDirection->x * distanceToSphere;
        double vy = rayDirection->y * distanceToSphere;
   double vz = rayDirection->z * distanceToSphere; // v = end point of ray when projected to sphere orbit
        double distanceFromEnd=Math::Sqrt(((c->x - vx)*(c->x - vx)) + ((c->y - vy)*(c->y - vy)) + ((c->z - vz)*(c->z - vz)));
        if (distanceFromEnd > distanceToSphere) return; // the distance of the centre is greater than the distance to the beginning so its pointing away
        if (R < distanceFromEnd) return; // The distance of V from C is greater than R so its outside
        double d=distance->getValue();
        if ((d < 0.0) | (d > distanceToSphere - R)) {
        distance->set(distanceToSphere - R); // return distance to the nearest point
                nearestNode->setNode(this);
        }
}
```



Uitleg bij de JAVA en C++ code voorbeelden

In de test code voorbeelden wordt er van uitgegaan dat de coördinaten allemaal relatief tot zijn van de camera positie.
Absolute coördinaten vragen meer rekenwerk.

C = middelpunt van cirkel / bol
Een lichtstraal zal door de bol (object in 3D) gaan indien de nieuw te berekenen vector V binnen in de bol is aangekomen, door een verlenging vanuit het brandpunt / eyepoint.

De afstand van vector V tot C (middelpunt) bedraagt
De straal van de bol bedraagt R

$$\sqrt{((Cx - Vx)^2 + (Cy - Vy)^2 + (Cz - Vz)^2)}$$

Indien deze uitkomst groter is dan R dan zal de straal niet door de bol heen gaan.

De test om te controleren of er sprake is van intersection (trefpunt):

$$R > \sqrt{((Cx - Vx)^2 + (Cy - Vy)^2 + (Cy - Vy)^2)}$$

Voordat deze test wordt uitgevoerd is het handig om eerst te controleren of het startpunt zich al binnen de bol bevindt. En vervolgens of de straal naar buiten wijst waardoor geen trefpunt wordt bereikt.

De testen (zie codes op vorige pagina) zijn dan als volgt:

$$\text{if } (R > \sqrt{((Cx)^2 + (Cy)^2 + (Cy)^2)}) \text{ return } 0$$
// start punt binnen de bol, afstand is 0

$$\text{if } (\sqrt{((Cx-Vx)^2 + (Cy-Vy)^2 + (Cy-Vy)^2)} > \sqrt{(Cx^2 + Cy^2 + Cy^2)}) \text{ return } -1$$
// afstand van het middelpunt is groter dan de afstand begin, wijst naar buiten

$$\text{if } (R < \sqrt{((Cx-Vx)^2 + (Cy-Vy)\text{^}2 + (Cy-Vy)^2)}) \text{ return } -1$$
// De afstand van V vanuit C is groter dan de straal R, buiten de bol

$$\text{return } \sqrt{(Cx*Cx + Cy*Cy + Cz*Cz)} - R$$
// geef de afstand tot het dichtsbijzijnde punt weer

Render programma's begin 2013

met ondermeer de gegevens fabrikant

Thea Render

Een nieuwe GPU gebaseerd renderingsprogramma. Solid Iris Technologies is de maker van deze renderer. Het is een hybrid biased/unbiased programma. Voor Mac OSX, Linux en Windows.

Thea Render maakt het mogelijk voor de gebruiker om tussen twee render systemen te schakelen, zelf een keuze makend tussen 'fysisisch correct' en 'rendertijd'. Het programma wordt geleverd als plug-in voor ondermeer SketchUp, Softimage, Cinema 4D, Rhino, 3ds Max, Blender, Mode.

Een Thea Render licentie is nodig alvorens u de plug-in kunt aanschaffen (= ook een licentie). Thea Render is een spinoff van de makers van het bekende **Kerkythea** programma dat gratis is. Belangrijke features: **Displacement** maps en **Relighting**. Daarnaast is vanaf v.1.1 Tea Render gecombineerd met **Motiva Colimo** (alleen voor Windows) waardoor het mogelijk is om een 'oneindig' aantal variaties achteraf in Real Time te maken. We hebben deze extra mogelijkheid nog niet bij andere render programma's opgemerkt. De tijd die nodig is voor die variaties is vele malen minder, dan voor de volledige render, het bestand wordt in Colimo gehaald om alle mogelijkheden snel te kunnen kiezen. Een soort Post-processing van de gemaakte rendering, maar met snel resultaat en bijzonder aantrekkelijke extra mogelijkheden.

Vraag en antwoord

Vrg. De term "physically-plausible", wat betekent dat?
Antw. Indien er sprake is van een "physically-plausible", wordt ermee bedoeld dat de geometrie, materialen en de 'emitters' gebaseerd zijn op modellen die bepaalde fysische eigenschappen bezitten. De meest opvallende eigenschap is die van het energie concept: een materiaal kan nooit meer licht reflecteren, dan dat er via de diverse lichtbronnen wordt geleverd.
Andere eigenschappen bestaan uit het gebruik van hoeveelheden en wederkerigheid bij de verstrooïng van licht.

'Making of MS House' bij zonsondergang. Ronenbekerman, architectural Visualization Blog. Door Madjid Rekeb opnieuw in Thea Render bewerkt.

Vrg. Bij sommige renderprogramma's wordt de term "physically-based" gebruikt, wijkt dat af van "physically-plausible"?
Antw. Als een programma "physically-based" is dan is het zeker ook "physically-plausible", dat betekent dat er geen natuurkundige wetten worden overtreden. De laatste zorgt ervoor dat de modellen nog nauwkeuriger zijn dan de "physically-plausible" die vaak op waarneming berusten.

Vrg. Is Thea Render een "physically-based engine"?
Antw. Thea Render maakt gebruik van 'state-of-art' materialen, die gebaseerd zijn op een erg goede theorie en daarmee wordt een nauwkeurige fysisch juiste Engine gemaakt.
De materiaal primitieven en het lagen systeem zijn gebaseerd op kortgeleden uitgevoerd onderzoek. "*Waardoor Thea Render één van de meest fysisch nauwkeurig renderprogramma's is van deze markt*". Daarnaast gebruikt Thea Render

bepaalde toevoegingen (Shading normals, Point lights) die het gemak van de gebruiker ten goede komen. Het zijn extensies / uitbreidingen op de strak aangehouden fysische manier van renderen.

Vrg. Thea Render was in oktober 2012 aanwezig op Basecamp 2012 in de VS van SketchUp Trimble. Wat werd daar vertoond?
Antw. Nieuwe mogelijkheden van Thea Render en een aankondiging van de nieuwe aanstaande versies met de geïntegreerde plug-in voor Trimble's **SketchUp**.

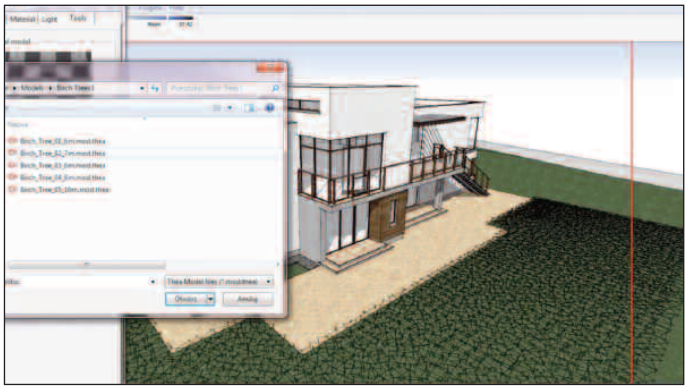
http://www.thearender.com/cms/

YouTube:
http://youtu.be/C2Zo4BZGq-c
Thea Showreel met aanstekelijke muziek.

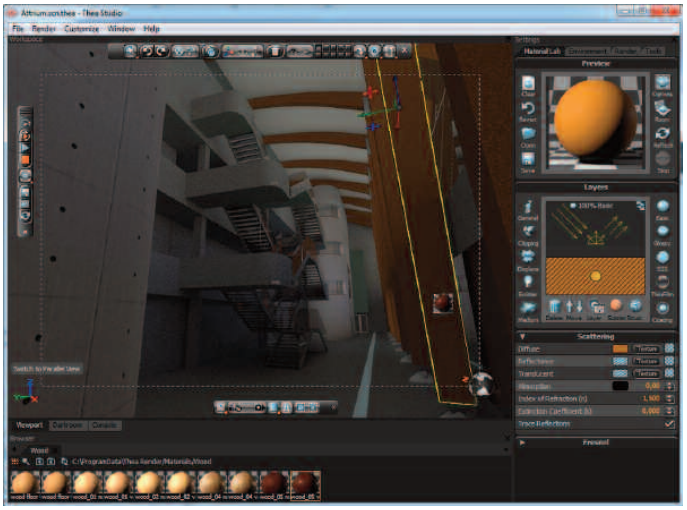
Displacement mapping
http://en.wikipedia.org/wiki/Displacement_mapping

http://youtu.be/PHXlApa3Mjw
Basecamp 2012 met SketchUp plug-in

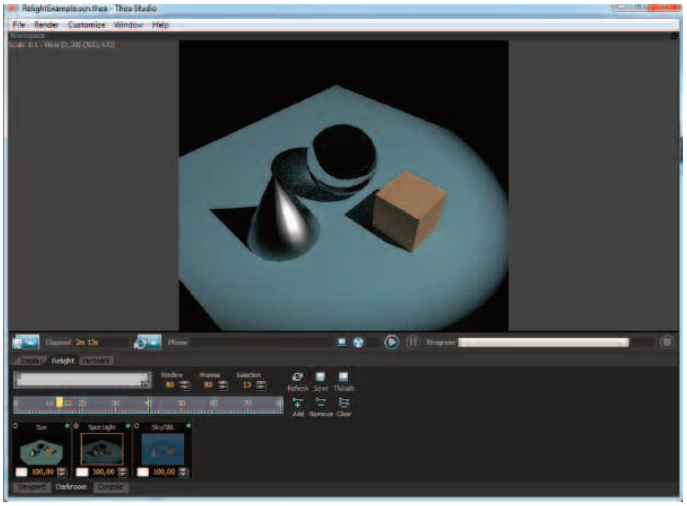
Thea gaat van CPU naar GPU !
http://youtu.be/MZUKi8tlycs
Voor het eerst is de nieuwe GPU render engine te bewonderen, deze wordt in de volgende versie van Thea uitgebracht (datum niet bekend gemaakt). Bij de film werd gebruik gemaakt van de **GeForce NVIDIA GTX 580** grafische kaart. Het ligt daarbij voor de hand dat het nieuwe programma CUDA georiënteerd is. Rendersnelheid ziet er goed uit met de GTX 580. Nieuwe mogelijkheden: ParticleFlow, Proxy systeem, Xfrog plant bibliotheek voor Thea. Er volgen nog meer opties.



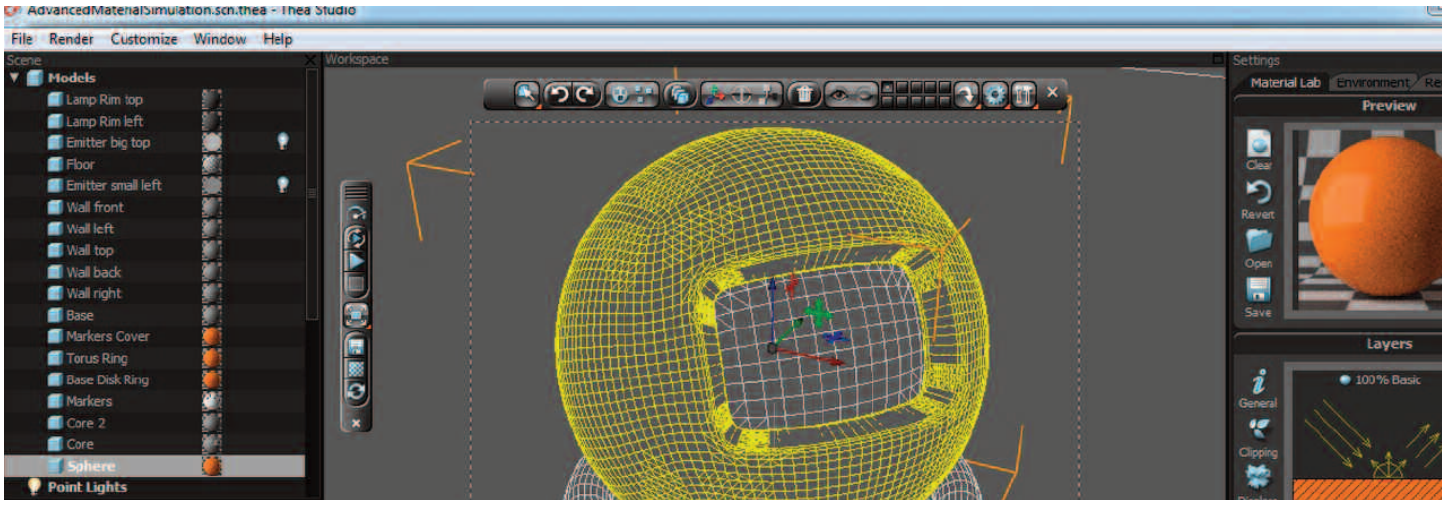
Nieuwe plug-in voor SketchUp.



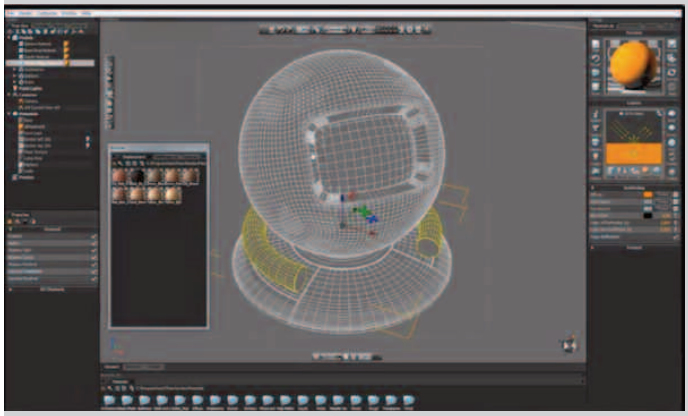
Interactieve Thea Render.

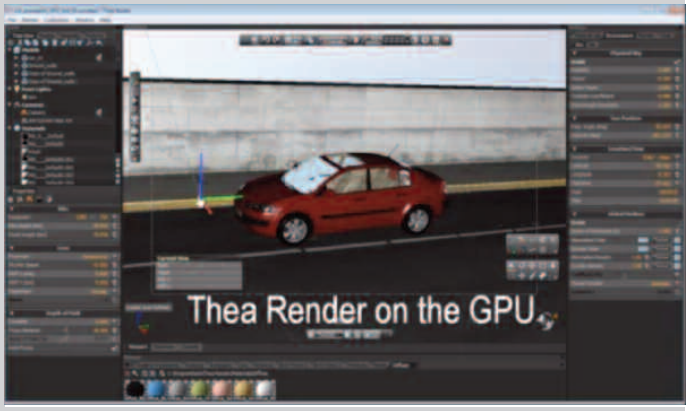


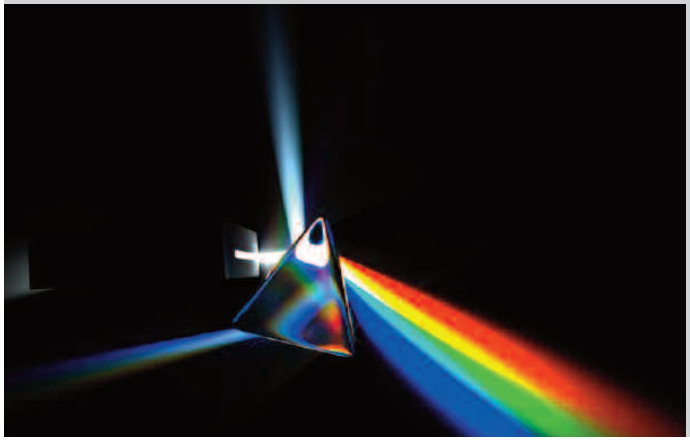
Relight editor.

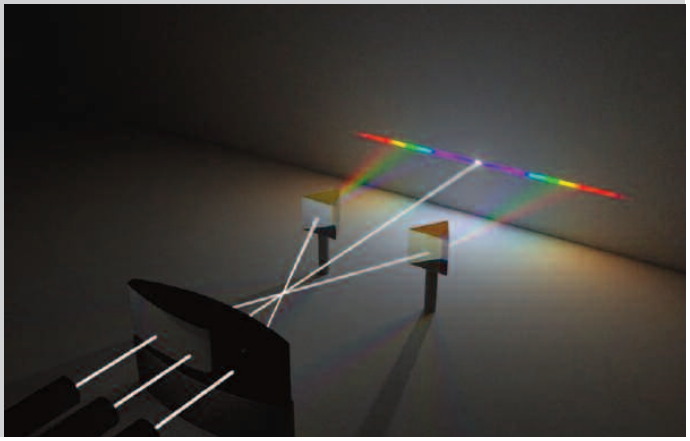


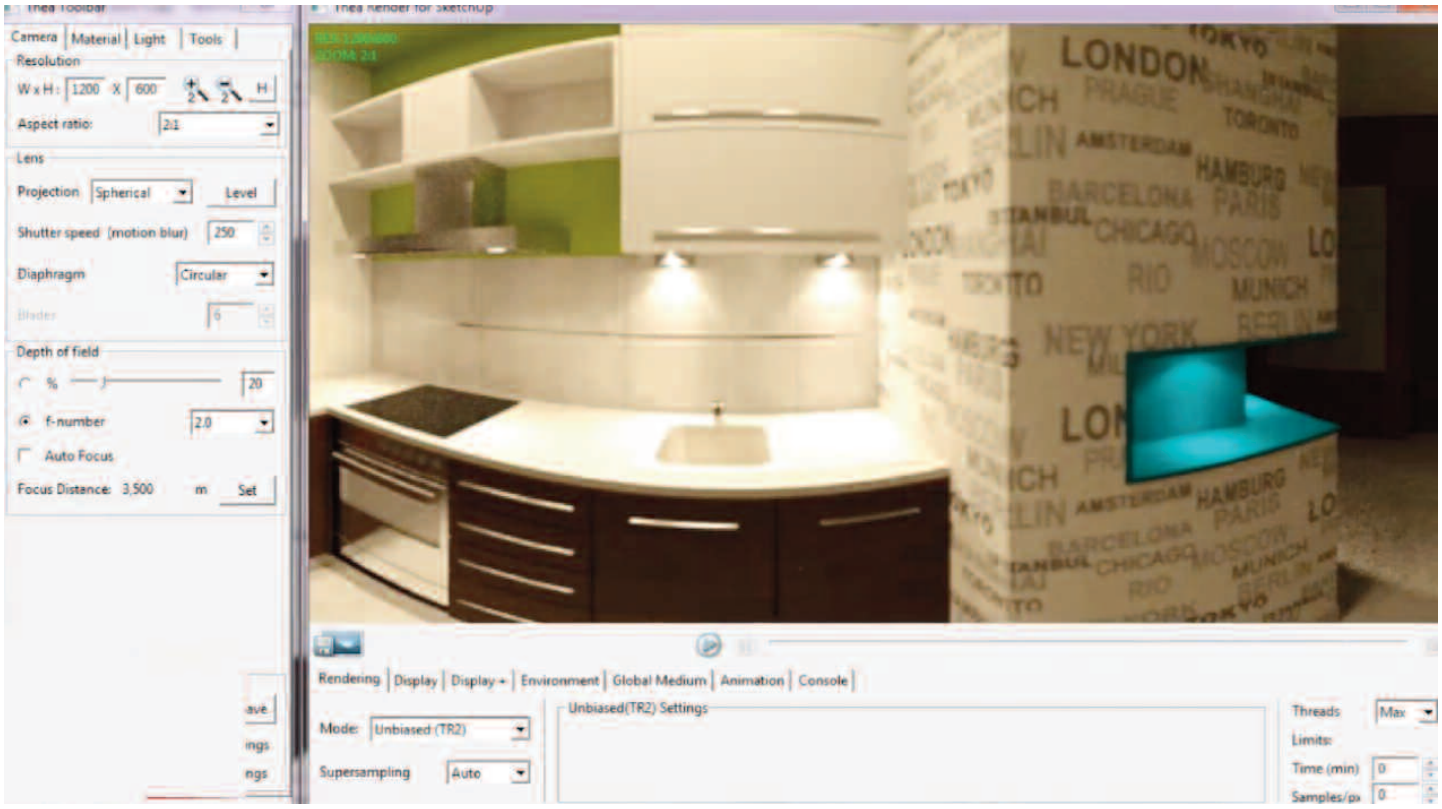
Thea Render schermafdukken



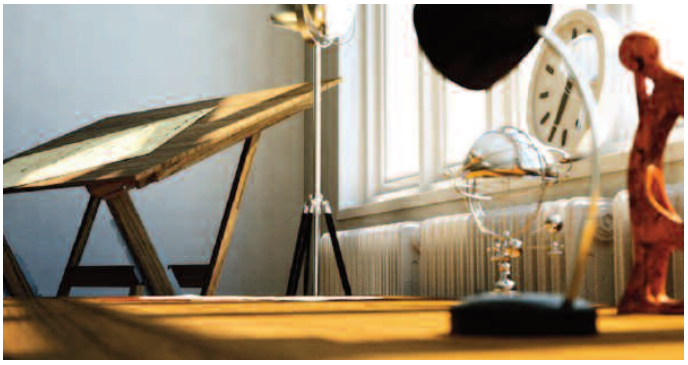








© *Thea Render een programma om te blijven volgen, het bezit potentie.*



Octane Render V 1.0

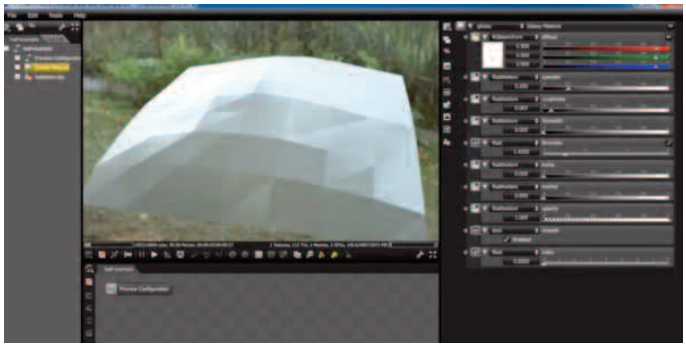
http://render.otoy.com/

26 november 2012 verscheen Octane Render Versie 1.0. Volgens de makers een "Physically based GPU renderer".
De makers zijn bijzonder trots dat Octane Render na een lange tijd Beta te zijn geweest nu als officiële eerste versie kon worden uitgebracht. De betatesters en gebruikers worden van harte bedankt. De nieuwe prijs is $ 199,- en dat is een enorm verschil met bv. een programma dat nog van de oude langzame techniek gebruik maakt zoals Maxwell Render waar een prijskaartje van $ 995,- aan hangt. Let op prijzen en mogelijkheden kunnen in de loop van de tijd veranderen.

Venturebeat.com:
"Bottom line, with Octane Render, it is now possible to render a photorealistic scene at 1080px in seconds with an off the shelf $350 video game card," said Urbach. "The quality is no different — in fact, it is perhaps even better — than what comes out of a cluster of high-end CPU render nodes rendering movie frames for a $200 million-Studio production."

Platform
Octane Render wordt uitgebracht voor
Macintosh OSX 32-64 bit, vanaf 10.5 en hoger
Windows XP, Vista, 7 en 8 (32- en 64 bit)
Linux 64-bit

Benodigd
Dit is een van de belangrijkste onderdelen van elk render programma. Op welke manier en met welke middelen (hardware & operating systeem) kunt u het programma optimaal gebruiken?
Octane Render heeft zondermeer NVIDIA video driver 301.42 nodig of een hogere uitvoering. En voor Mac OSX (10.5 en hoger) gebruikers is ook de laatste CUDA driver noodzakelijk, de links staan op de Octane site en hiernaast vermeld.



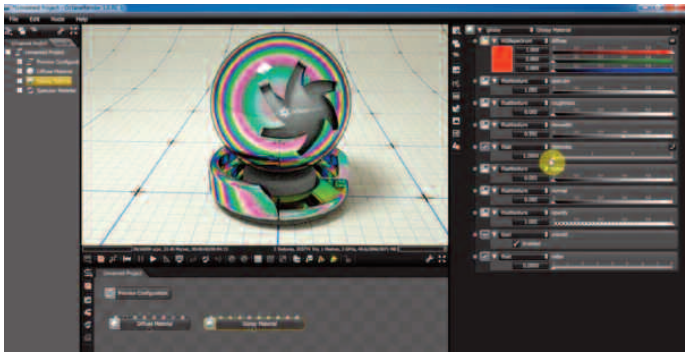
Alle afbeeldingen © Otoy, Octane Render of een van de makers van de renderingen die bij de rendering onderaan staan vermeld.

http://render.otoy.com/forum/viewtopic.php?f=7&t=25376

Drivers bij NVIDIA
http://www.nvidia.com/Download/index.aspx?lang=en-us
CUDA driver archief voor Mac OSX
http://www.nvidia.com/object/mac-driver-archive.html

Van de fabrikant:
Features Octane Render V 1.0

SNELHEID
Octane Render gebruikt de kracht van de GPU met CUDA cores, in plaats van de CPU bij andere render programmas.
Octane kan renderingen tussen de **10 - 50 x** zo snel maken t.o.v. welke CPU met meerdere cores dan ook. U kunt meer dan één GPU inzetten om nóg sneller te kunnen werken.
Er is een echte WYSIWYG omgeving, zodat



de gebruiker zich kan richten op de instellingen en direct feedback krijgt.
Het beeld op het scherm IS de uiteindelijke render afbeelding. Waardoor verschillen bij diverse renderingsprogramma's tussen Preview (met bv. OpenGL) en de uiteindelijke rendering (CPU) wordt voorkomen.

KWALITEIT
Octane Render levert "*uncompromising quality*" aldus de makers. Bekijk de website en de gallerie om zelf een indruk van te kunnen vormen van deze uitspraak.

FLEXIBEL
Octane kan met vele 3D programma worden toegepast in tegenstelling tot de specifieke renderprogramma's die slechts een kleine lijst van compatibele 3D programma's ondersteund.
Daarbij wordt dankbaar gebruik gemaakt van de universele .**OBJ** en .**Collada** formats.
Octane Render licentie kan ook worden gedeeld tussen verschillende operating systemen, die op dezelfde computer draaien of zelfs op verschillende computers.

MATERIALEN
Materialen kunnen een renderprogramma maken en breken. Octane Render staat de gebruiker toe om de volledige controle over de materialen te houden en met zijn Real-Time systeem wordt het zoek- en testwerk sterk verminderd. De veranderingen aan de materialen kunnen direct worden bekeken.

Met de **Node Editor** kunnen complexe materialen met textures worden samengesteld.
SSS (Subsurface Light Scattering) wordt ondersteund evenals complexe IOR, Chromatic Dispersion en Absorption om bijzondere goed uitziende materialen te ontwikkelen

OctaneLive material database staat het de gebruiker toe om snel via internet de toegevoegde materialen te raadplegen en te gebruiken. Allemaal binnen de standaard Octane interface.

VERLICHTING
Of u nu HDRI files gebruikt, Mesh emmitters, IES files of zon en lucht, Octane maakt complete besturing van het licht mogelijk. Stel de plaats van de zon in om de juiste schaduw te creëren in een kamer, gekoppeld aan de kalender en de tijd op elke locatie op aarde.

Elk object kan in een Mesh Light worden omgezet door de stralingseigenschap van het materiaal te gebruiken. De kleur kan door een texture, of door het kleuren spectrum worden aangegeven. Met het inladen van IES files kan een nog realistiser lichtpatroon worden gecreëerd.

Een HDRI file kan worden gedraaid als aanvulling op bestaande verlichting. Of stel de helderheid van het licht in en bekijk direct het resultaat.

REAL TIME (RT)
We hebben het over deze term al meerdere keren gehad. De term “RT” ligt niet vast. Maar uit eigen proeven en die van YouTube films met vermelding van de gebruikte hardware is af te leiden dat “RT” hier zeker niet oneigenlijk wordt gebruikt.

Met WYSIWYG kan de gebruiker zich concentreren op het resultaat. De gebruiker ziet direct het direct op het scherm.

GEOMETRIE VASTLEGGEN
Octane maakt het hergebruik van geometrie mogelijk. Verander alleen de schaal, plaats,

© Florinmocanu - Griekse vaas - Octane Render

© Octane Render

Onder:
Schermafdruk Windows computer.
Onvoorstelbaar veel mogelijkheden met glooiende oppervlakken en materialen.
Zie filmlink op volgende pagina.

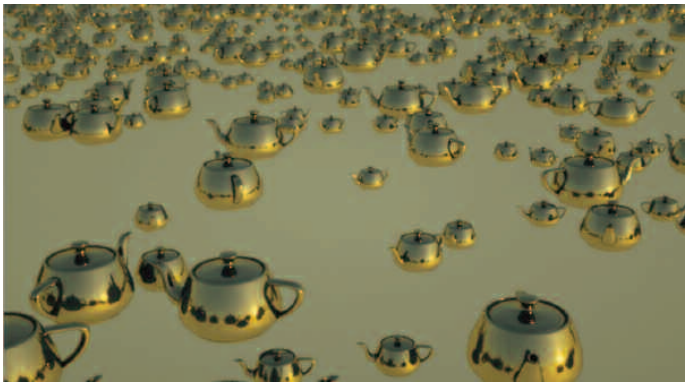
opstelling en materiaal instelling. Er kan een geometrie kopie worden bewaard waarmee kostbare VRAM geheugen wordt bespaard. Meerdere geometrie modellen zijn mogelijk evenals kopieën van gras, haar en (bos) resp. boomonderdelen.

PLUGINS
Octane Render wordt nu ook als geïntegreerde plug-in voor populaire 3D programma's uitgebracht zoals Autodesk **3ds Max**, Autodesk **Maya**, Smith Micro **Poser** en binnenkort ook Autodesk AutoCAD, Autodesk Softimage, Daz3D Daz Studio, Newtek's Lightwave en Maxon Cinema 4D.
In het eerste kwartaal van 2013 zullen nog meer plug-ins volgen.

EXPORT SCRIPTS
Gratis export scripts voor **SketchUp**, Blender, Modo, Rhino, Softimage, 3ds Max, Cinema 4D en Daz3D. Daarmee is het mogelijk om vanuit de 3D programma te exporteren en Octane Render als een standalone Render programma te gebruiken, gelijk aan bv. Artlantis R en S.

Eerste conclusie
De eerste reactie is er een van verwondering over het grote aantal features en mogelijkheden, die nu in Versie 1.0 worden geboden voor deze interessante prijs. We kunnen niet anders zeggen dat de jarenlange Beta periode kennelijk zijn vruchten heeft afgeworpen. Daar blijkt ook uit dat Octane (Otoy, Australië) het geduld heeft genomen om alles op orde te hebben. Een bijna Engelse aanpak en dat spreekt in deze tijd bijzonder aan. Dat wil niet zeggen dat er bij en na de introductie geen bijzonderheden naar voren zullen komen, dat is onvermijdelijk voor elk programma. Ook de interface vraagt de nodige studie.
Octane heeft niet zoals te vaak voorkomt het programma of versie te vroeg op de markt gebracht. Voor minder dan $ 100,- konden de mensen eerst de beta testen.

DEMONSTRATIE FILMS
In de films hebben we gezien dat Octane met meerdere high-end NVIDIA CUDA kaarten werkt. U dient bij het gebruik van één goede



NVIDIA CUDA kaart dat terug te schalen.

http://youtu.be/F7r69634fVM
Octane Render Introduction to Materials
Fraaie mogelijkheden met de standaard materialen.

http://youtu.be/K3wdJplu0xQ
Octane Render Standalone Textures Tutorial
Wederom een nieuwe manier om met texturen om te gaan.

Onze uitgave "GPU-Toekomst" van augustus 2012 een aantal wetenswaardigheden over Octane Render toen nog in de BETA UITVOERING.

Octane Render

Het eerste programma voor Windows, Linux en Mac dat gebruik maakt van CUDA om te renderen. U kunt een demo ophalen om eerst eens te proberen. Onderstaand de stappen die we maakten bij een Windows 7 computer om het programma te laten werken:

1. Controleer of u een grafische kaart hebt die CUDA ondersteund (alleen bij NVIDIA kaarten, GeForce en Quadro). In dit geval hadden we een NVIDIA Quadro 600 kaart die CUDA ondersteund.

2. Zoek de bijpassende CUDA driver (NVIDIA site) op die bij uw kaart past.

3. Bij onze Windows computer werd direct nog voor het downloaden gevraagd om het bijpassende Java programma. Wellicht dat het bij Mac al geïnstalleerd is. Het is nodig bij het werken met de CUDA driver.

4. Download Java.

5. Download de CUDA driver (175 MB).

6. Geef na het installeren van de CUDA eenmaal één herstart voor de computer.

7. U kunt een demo programma van Octane Render ophalen:
http://render.otoy.com/

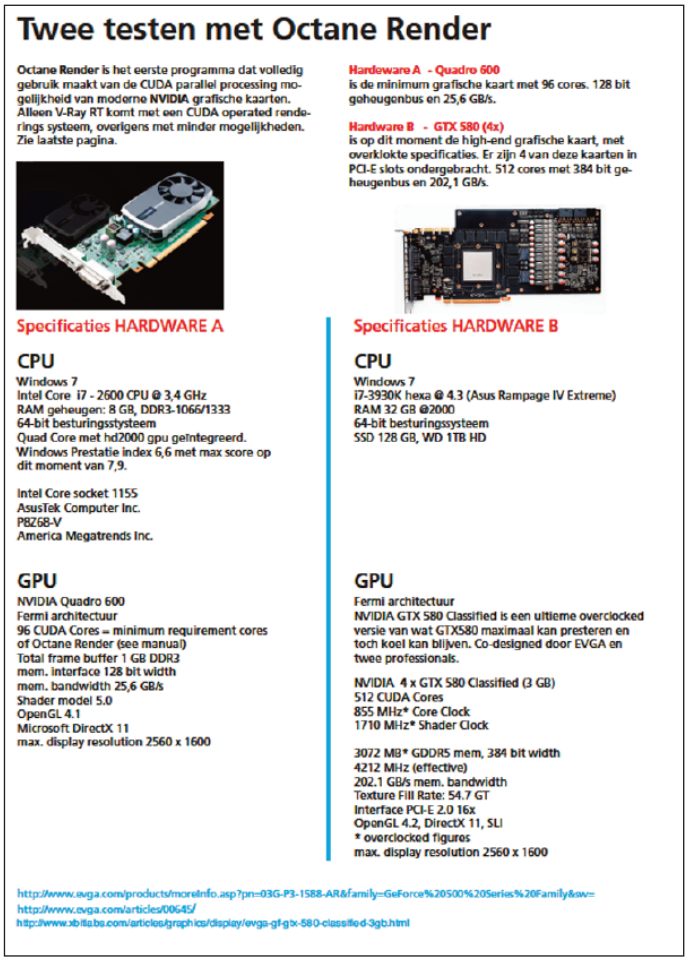
8. Installeer Octane Render.

9. Als het goed is gegaan wordt de grafische kaart direct herkend in de Preferences van Octane, zie schermafdruk van de Windows 7 computer met grafische kaart Quadro 600. Verder blijkt dat Cuda Driver versie 4.20 wordt herkend met runtime version 4.00. Er zijn 96 cores met een kloksnelheid van 1250 MHz beschikbaar. In het onderste vak van het Preference menu de gegevens van het gebruikte geheugen.
Het programma is klaar om te gebruiken ('save as' is uitgeschakeld in de demo versie van Octane). Experimenteren maar.

Twee snelheidstesten Octane Render *B*
We deden toen ook uitgebreide testen om de snelheid van het programma met heel uiteenlopende hardware te testen.
In Literatuur de pdf met het complete verslag. **"twee testen_4 Octane.pdf"**

Huidige Versie 1.0 Octane Render
Het is best mogelijk dat de werkvolgorde bij de officiële 1.0 uitvoering iets anders verloopt. Bij Literatuur treft u de nieuwe handleiding aan van Versie 1.0.
"OctaneRenderUserManual.pdf"

Verkoopkanalen
Octane Render zal Online worden verkocht, maar men denkt ook aan een dealernetwerk, aangezien juist **goede support vooraf en achteraf** nodig is. Locale support zal nodig zijn, meer dan bij andere renderings programma's. We blijven het volgen.



Octane Render voor LightWave

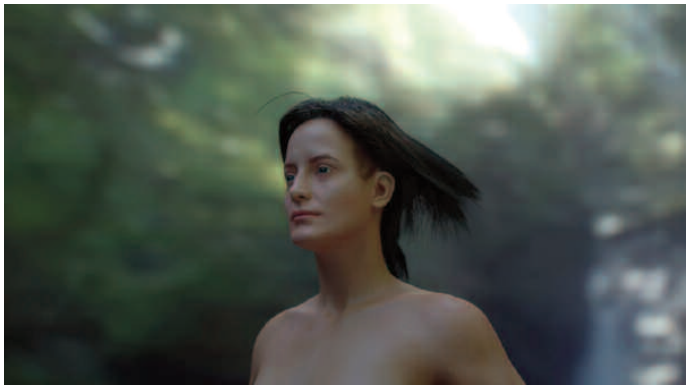
Ontwikkelingen . . .
Snelle, PGU gebaseerde render programma's laten zich kennelijk gemakkelijk inkopen bij bestaande 3D programma's.

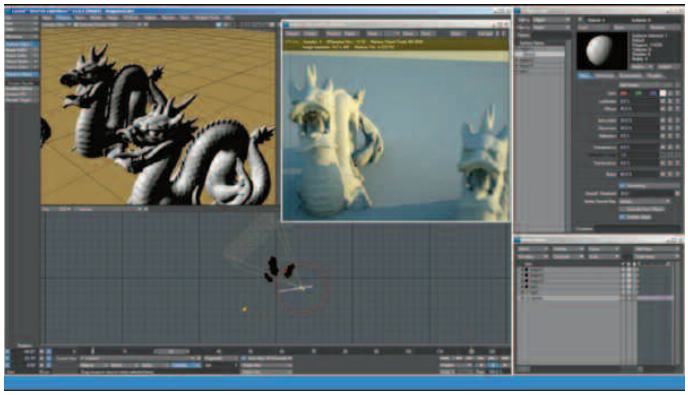
In januari 2010 schreef Octane het volgende: *"Het OctaneRender team is erg trots om te vermelden dat er opnieuw een geïntegreerde plug-in wordt ontwikkeld."* Ze boekten daarmee hun zesde plug-in succes voor hun ontwikkelingswerk van het OctaneRender op GPU gebaseerd programma. Inmiddels zijn dat er meer.
In de bijbehorende films kunt u zelf zien hoe dat in zijn werk gaat. In een aanvulling zien we dat de films opgenomen zijn met een GTX 670 NVIDIA kaart.
http://render.otoy.com/forum/viewtopic.php?f=7&t=24959

Zònder deze gegevens kan de toekomstige klant geen enkele indruk krijgen hoe het programma bij hem of haar zal werken.

En voor LightWave, dat altijd aan de onderkant van de computereisen bungelde, waardoor een grote groep mensen met middelmatige computers ook het programma kunnen gebruiken, is het een hele ommezwaai. Zij dienen te investeren in GPU kaarten, die wellicht niet in hun onderkant-van-de-markt-moederbord passen. Of te wel zij dienen over te stappen naar een nieuwe computer, die zowel CPU als GPU (NVIDIA CUDA) goed ondersteunt.

oktober 2012
Meer dan een miljoen polygonen voor haar.

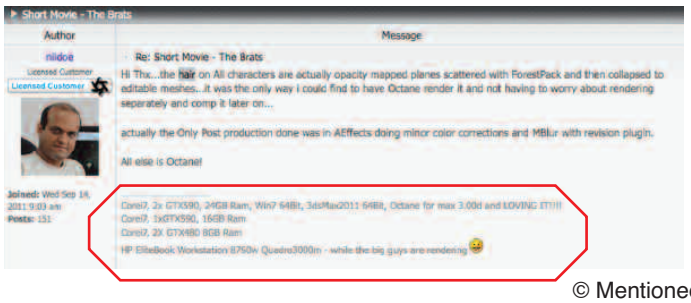




LW met Octane render plugin

Gras met OctaneRender; programma blijft interactief.

Zó zou het bij andere Forums ook moeten gaan
Octane Forum: onder de streep de technische gegevens van de gebruikte computer(s) plus GPU etc. dragen bij aan eerste goede communicatie.



IRAY NVIDIA

http://www.nvidia.com/object/nvidia-iray.html

iray® is de foto realistische render oplossing. Waarbij goede renderingstijden kunnen worden bereikt door de combinatie met een van de nieuwste GPU's van NVIDIA. In tegenstelling (aldus de fabrikant) tot andere programma's wordt met iRay resultaten bereikt die met de 'werkelijkheid overeenkomen'.

Fotorealistische renderingen met een 'druk-knop' systeem maakt het mogelijk om zowel kwaliteit en fotorealistisch te combineren met gebruikersgemak en snelheid.

iRay verfijnt tijdens het renderen de kwaliteit zowel bij de preview als bij de uiteindelijke rendering. En maakt daarbij gebruik van de voordelen van parallele processen middels NIVDIA CUDA GPU systemen.
Vereist een CUDA kaart met 2 GB of meer geheugen, zie snelheidsvergelijkingen.

Voor ontwikkelaars is er een gratis download beschikbaar.

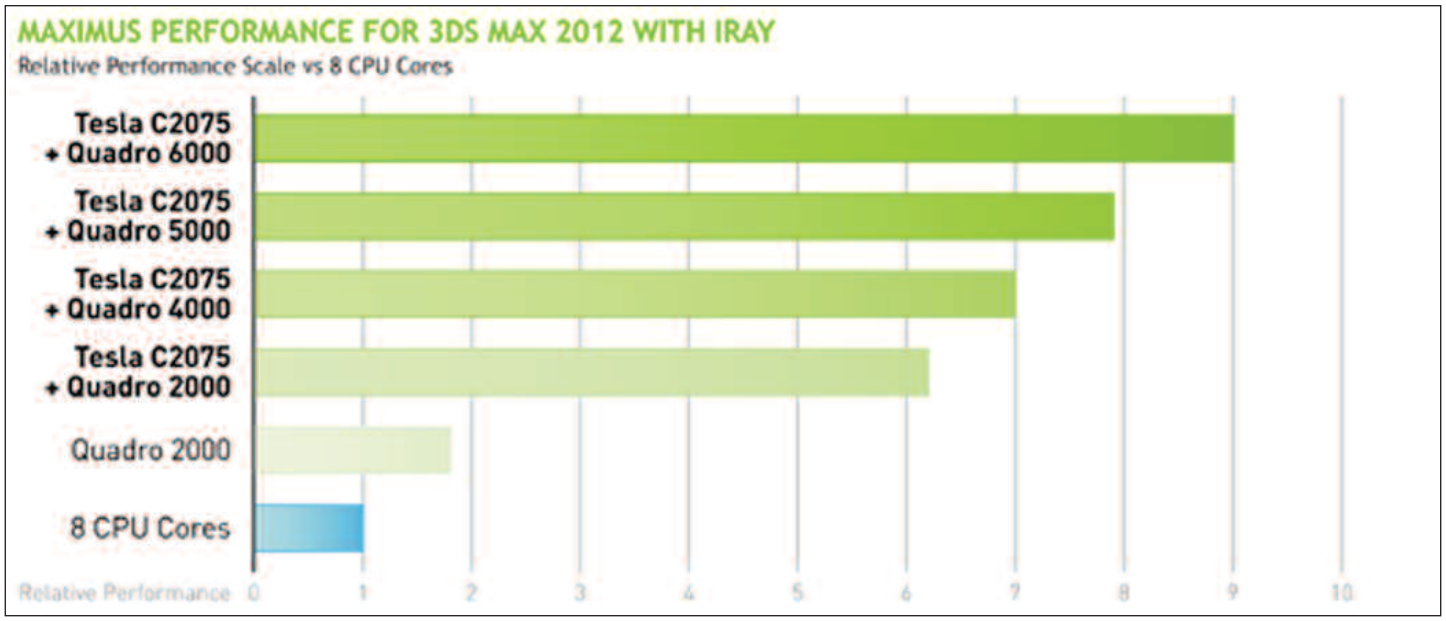
3ds Max met iRay met Fermi-gebaseerde **GPU** na 2 minuten.

3ds Max met iRay met **CPU** na 2 minuten.

Bekende rendermerken die iRay toepassen:
Autodesk 3ds Max en 3ds Max design
Bunkspeed
Dassault systemes CATIA v6
Cinema 4D met m4d van at2.

Voor MacPro gebruikers is bv. de Quadro 4000 een interessante aanschaf.

Een snelle GPU kaart levert direct enorme tijdwinst op ten opzichte van bv. CPU 8 core processoren.
Groene staven zijn de snelheidsfactoren, blauw een 8 core CPU.



iRay Gallery

http://www.mentalimages.com/products/iray/iray-gallery.html

iRay rendering
© courtesy Delta Tracing

iRay rendering
© courtesy Delta Tracing

iRay rendering

© courtesy Dassault Systemes, CATIA V6 Live Rendering

Mental Ray

Een bijzonder programma met veel mogelijkheden en toepassingen. Zowel in productie omgevingen (render farms en Cloud) als standalone met GPU aansturing of zonder.
De standalone versie is te koop via Autodesk. De software voor OEM's via NVIDIA. Daarmee is Autodesk een van de grootste klanten van Mental Ray.

Ondersteund de correcte Global Illuminiation incl. Radiosity met een uitbreiding daarop.

Evenals iRay dat via NVIDIA wordt uitgebracht heeft ook Mental Ray een imposante klantenkring:

Autodesk met AutoCAD, Revit, Inventor® Series, VIZ, 3ds Max, Maya, Softimage®
Dassault Systèmes
Dassault: CATIA, Maticad, Domus3D, PTC, Pro/ENGINEER, Cinema 4D m4d plugin, SensAble Technologies FreeForm, ViSoft.
Ook deeltjes en haar kunnen optimaal worden gerenderd.

Soft- en hardware:
Ondersteund zowel Windows 7, Vista als XP voor 64-bit en 32 bit. Mac OSX Leopard, Snow Leopard, Tiger Intel processoren. Plus de belangrijke LINUX distributies en Unix zoals Solaris. Daarmee één van de meest universeel toe te passen renderers.

Ook hier wederom 'fysisch juiste afbeelding van goede kwaliteit en natuurgetrouw'.

Daarnaast worden er acceleratie algoritmes toegepast om zelfs noch met de eenvoudigste enkele processor nog te kunnen werken.

Mental Ray wordt als een standalone product uitgebracht, maar ook als biblitheek om direct te integreren in andere render programma's. Er worden scene data maps aangelegd in het .mi format.

Met de MetaSL of C/C++ taal kunnen texturen, materialen en nieuwe lichtbronnen worden ingepast plus camera efecten. Ook Globale verlichting met Photon shaders en

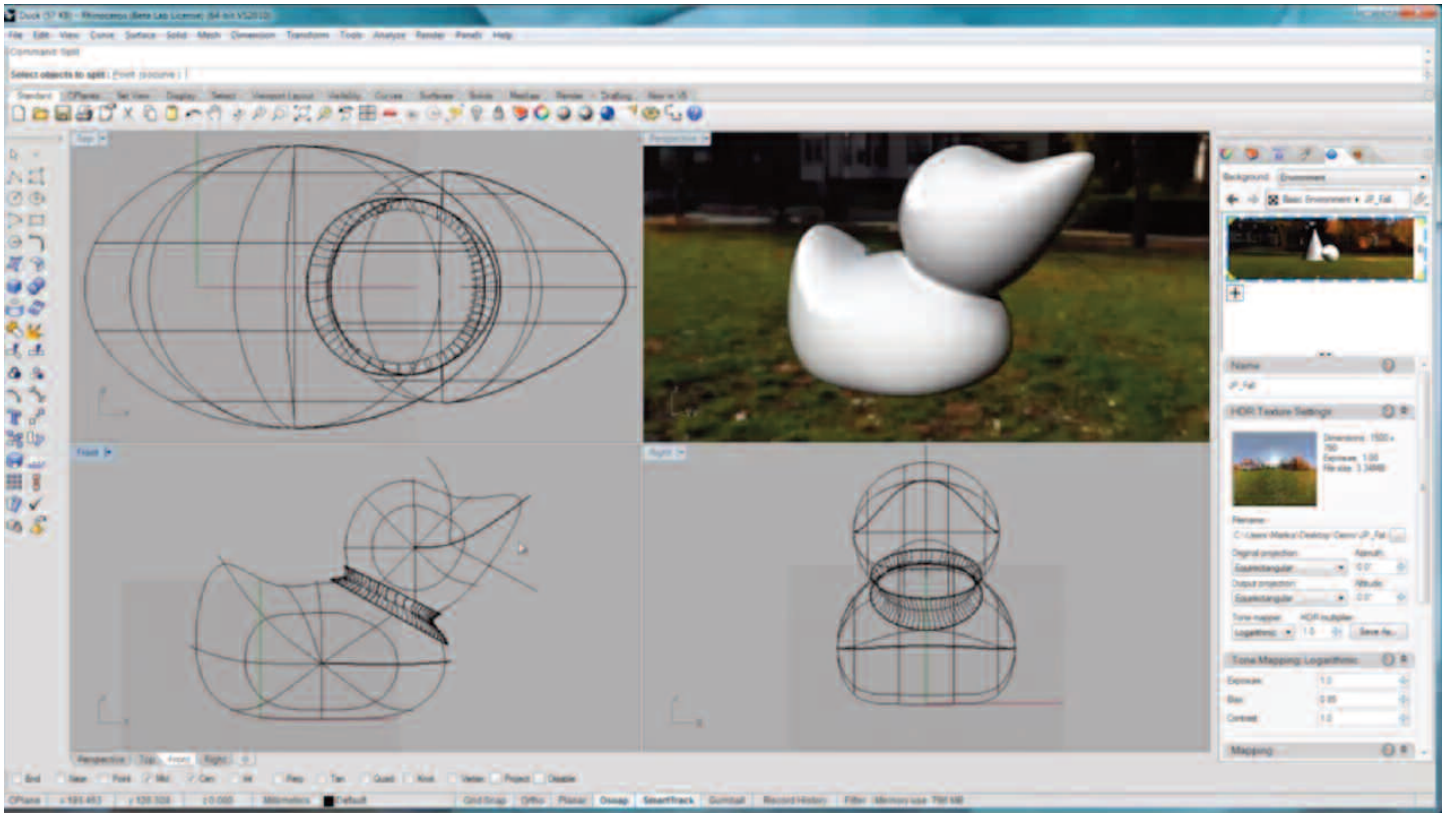
© Mies van der Rohe Farnsworth House
Artiest Alessandro Prodan
gerenderd met Mental Ray.

een Photon map worden gebruikt.

Ook de algoritmes zijn bijzonder te noemen:

1.1 Algorithms

* ray tracing architecture for automatic, accurate, and general simulation of reflections, refractions, shadows, and complex illumination
* rasterizer for fast, high quality rendering of directly visible objects, with efficient support for motion blur and displacement mapping, and with separate control of shading quality and visibility anti-aliasing
* scanline rendering method for fast rendering of scenes in low to medium complexity
* physically correct simulation of global illumination
• photon mapping combines forward and backward ray tracing and simulates all possible light paths, without the need to prepare and combine each effect separately
• final gathering for efficient and easy to use computation of one or more diffuse indirect light bounces, optionally combined with photon mapping
• computation of caustics using dedicated photon maps (light patterns caused by reflected or refracted light)
• interaction with participating media: volume caustics and multiple volume scattering in diffuse media like fog (e.g. light beams in fog)



Neon Project for Rhino

Een “full raytraced viewport” plug-in for Rhino 5.0 in ontwikkeling.

Deze plug-in werd in samenwerking met Caustic Professional gemaakt. Het biedt ondermeer een "Raytraced neon" dat bijzonder wordt aangeprezen. Volgens de makers is er geen vergelijkbare software, die hetzelfde presteert.

Neon maakt het de ontwerper mogelijk om snel vorm, kleur, textuur, verlichting en de materiaal benodigdheden van het object te kiezen binnen het 3D ontwerp.

Brazil instellingen worden ondersteund als Brazil 2.0 SR2 Beta is geïnstalleerd. Brazil SDK is de huidige interne SDK voor de ontwikkeling van plug-ins zoals bv. Neon.

Systeemeisen: Neon werkt het best op krachtige hardware, maar is ook bruikbaar op wat mindere (zelfs laptops). Alleen Windows. Geadviseerd 64-bit Rhino 5.0 met minimaal 4 GB RAM en i7 Intel processor (Sandybridge) met Windows 7. Medio dec. 2012 was het gratis te downloaden. Een goede grafische kaart is niet nodig, het zal dus CPU gebaseerd zijn. Zie het YouTube filmpje:

http://youtu.be/JDm4xosy5IQ

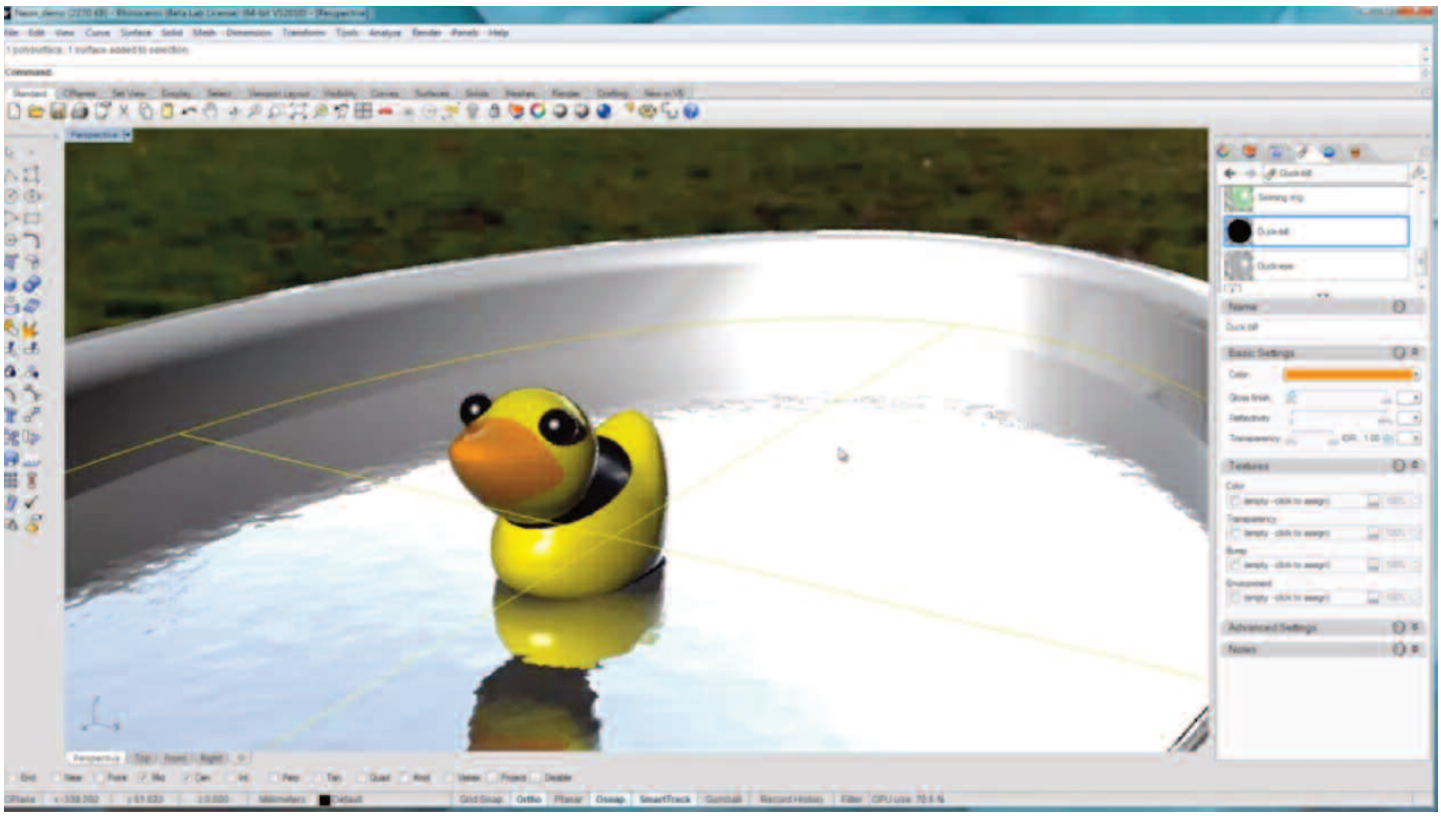
http://brazil.mcneel.com/
Brazil for Rhino

https://vimeo.com/3926832
introductiefilm van Brazil

Prijs 895,- Euro (begin 2013) bij McNeel Europe SL, Barcelona Spanje. Brazil is de commerciële en uitgebreide spinoff van OpenRL dat gratis is. Visualizer met Global illumination voor Autodesk Maya maakt ook gebruik van Caustic Professional voor de “fully interactive real time raytracer to Maya”.
https://caustic.com/brazilsdk.php
http://www.imgtec.com/powervr/powervr_brazil_sdk.asp
PowerVR Brazil SDK (beta) is de Ray Traced rendering Toolkit die gebruik maakt van het OpenRL platform. En wordt toegepast bij 3ds Max en Rhinoceros.

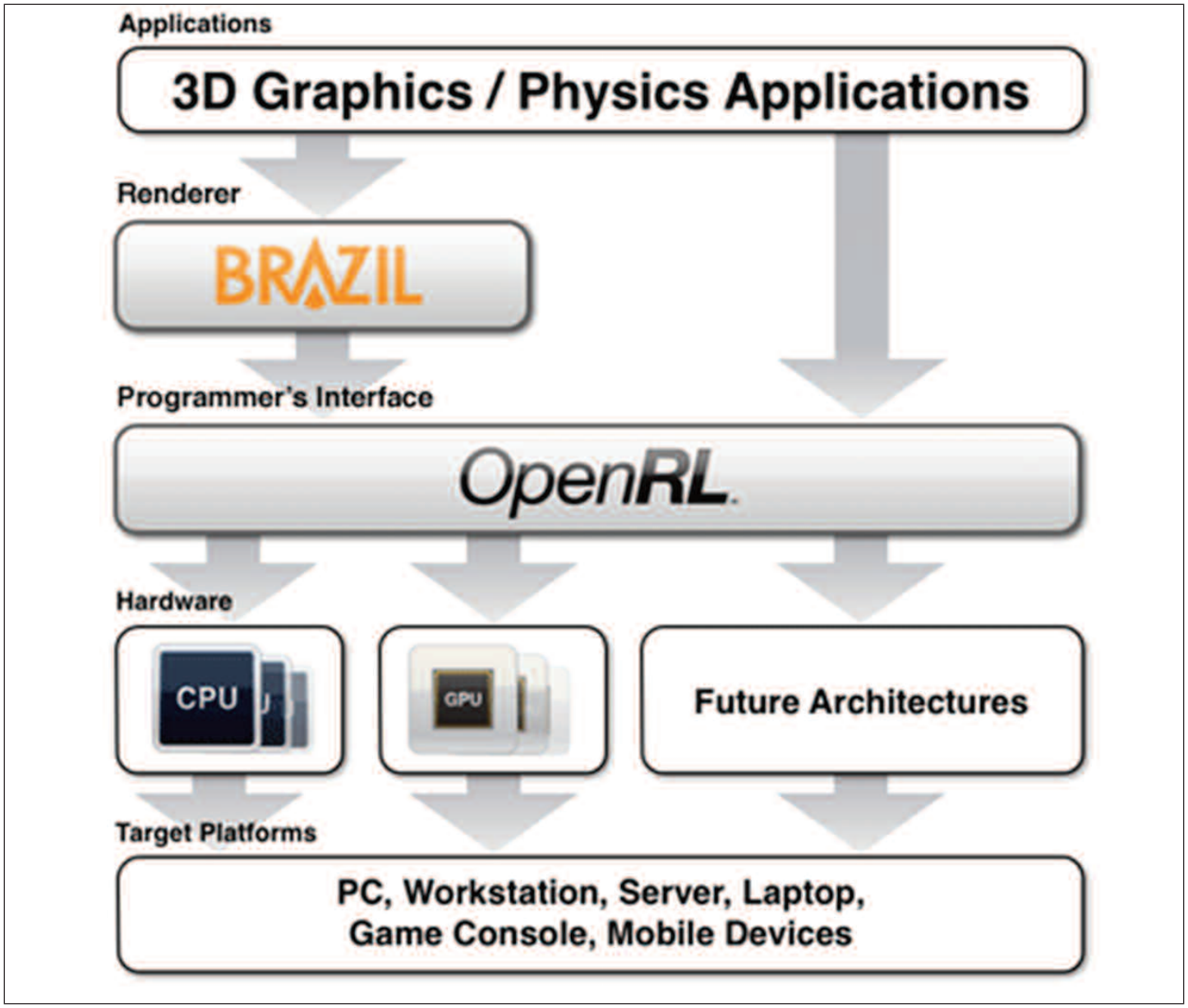
Op de volgende pagina nog twee schermafdrukken van Rhino in samenwerking met Neon. En een diagram over de werking van Brazil en OpenRL.

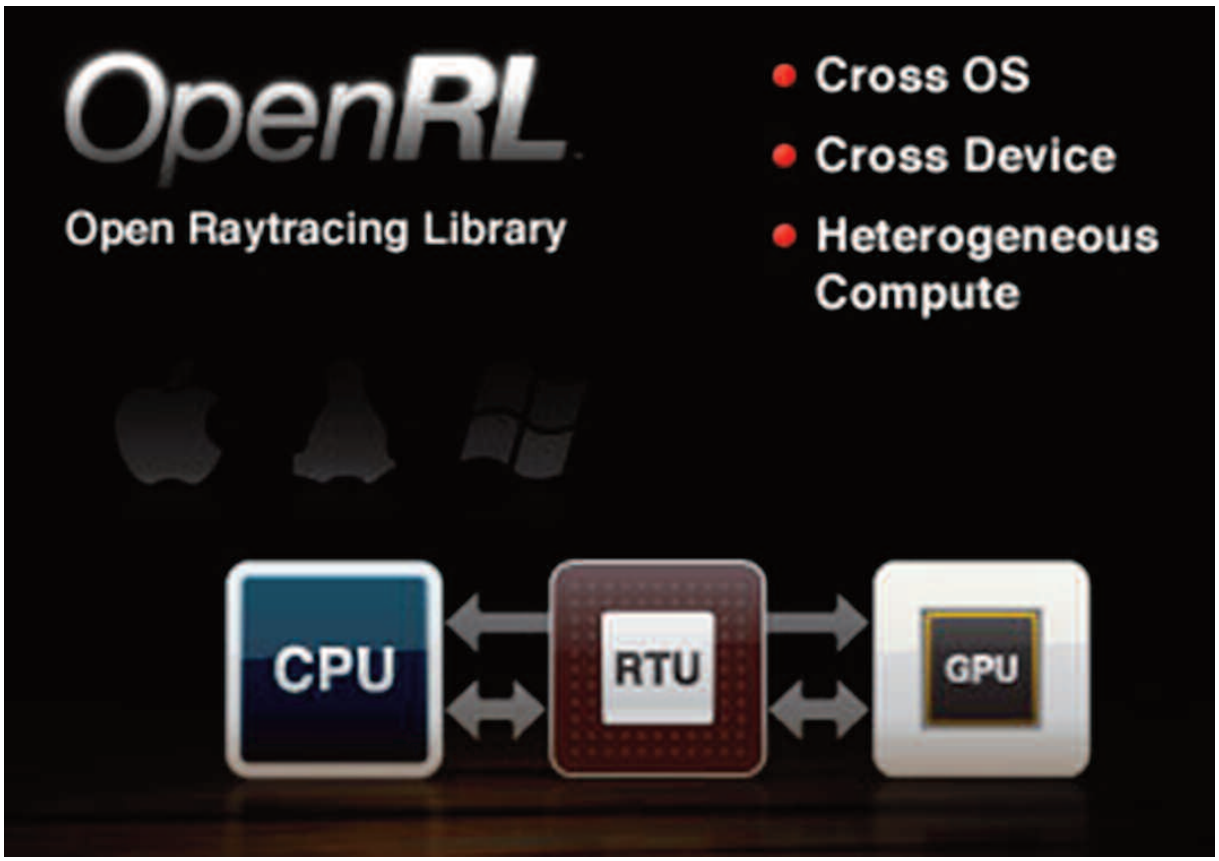
Met dank aan Alex Kelley van imgtec.com voor de verstrekte informatie.



Renderen binnen Rhino met Neon render programma.

Onderstaand het eindresultaat.


Applications
3D Graphics / Physics Applications
Renderer
BRAZIL
Programmer's Interface
OpenRL
Hardware
CPU
GPU
Future Architectures
Target Platforms
PC, Workstation, Server, Laptop,
Game Console, Mobile Devices


OpenRL
Open Raytracing Library
Cross OS
Cross Device
Heterogeneous Compute
CPU
RTU
GPU

LumenRT

Een renderprogramma uit de stal van e-on software producten (met o.m. Vue, Carbon Scatter en Ozone).
Het achtervoegsel RT, dat normaal voor *Real Time* staat wordt hier oneigenlijk gebruikt. De koper verwacht toch dat het renderings-proces 'life' gebeurt òf heel snel. Dat is met dit programma zeker niet het geval.

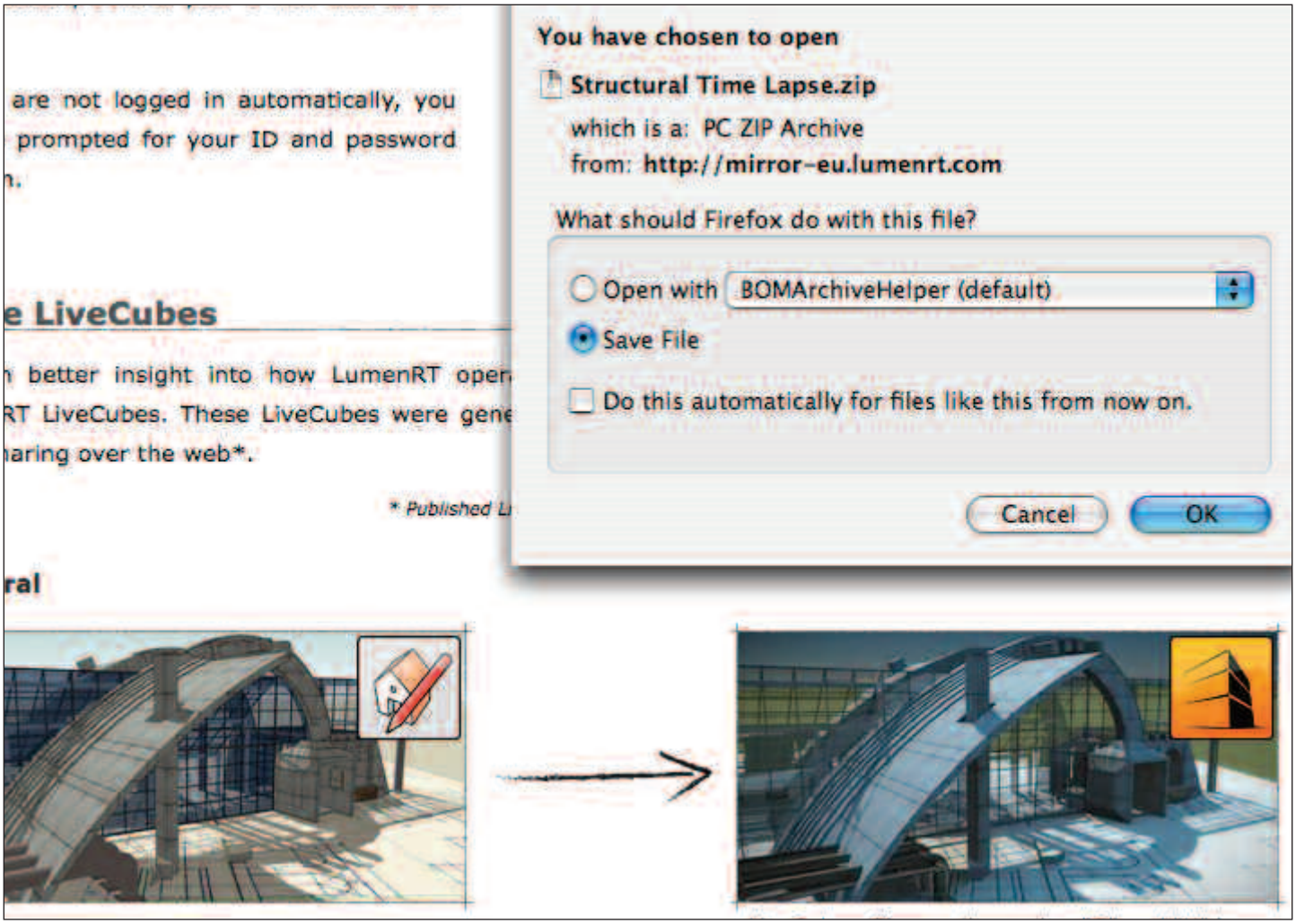
Om de verwarring nog groter te maken staat er op de website: "*LumenRT software creates real-time 3D visualizations of architectural designs*". Een onduidelijk begin voor een programma dat redelijk zou kunnen scoren omdat het één belangrijke feature heeft die andere ontberen. Daar had veel meer de aandacht op gericht moeten worden.

LumenRT website: "**The best picture quality on the market**" of "**LumenRT is the only solution that combines real-time rendering with a full, physical simulation of light and nature, and walk around your projects as if they existed**". Aan de hand van de bijgeleverde Showcase is gemakkelijk het antwoord zelf in te vullen. De fabrikant maakt zoals zo veel gebeurt de *Minimum System Requirements* zó laagdrempelig dat zelfs een digitale slak bruikbaar is. Daar komt bij dat de doorsnee MacOSX gebruiker geen keuze heeft NA aanschaf van zijn computer voor soort grafische kaart. De Mac Pro (4-, 6- of 12 cores) biedt die keuze wel, maar is begin 2013 een gedateerd model, met smart wordt op een goede upgrade gewacht. Geadviseerd wordt om, indien mogelijk, een NVIDIA kaart te kopen, waardoor de weg naar CUDA renderingsprogramma's in ieder geval later mogelijk is.

Valse starts
Er blijft dan voor deze bespreking, na de diverse valse starts waarbij de toekomstige klant op het verkeerde been wordt gezet, niets anders over dan de feature te bespreken: een methode om animaties, 'walk through' e.a. van de scene geschikt te maken voor zowel Windows als Mac OSX gebruikers met een redelijk goede grafische kaart. U levert daartoe een compleet bestand aan (van

behoorlijke grootte) waarmee de interactieve animatie zonder overige software kan worden gestart.
Uit het Forum blijkt dat er nogal wat klanten van renderingsbedrijven toch nèt niet over de juiste grafische kaart beschikken, waardoor de feature voor hen althans, niet werkt. En nu blijkt ook waar de term "RT" vandaan komt, men beroept zich erop dat NA HET MAKEN VAN DE RENDERING er **Real Time** door de scene kan worden gelopen door de klant. Het blijkt ook mogelijk om vanuit de LifeCubes schermafdrukken te maken, maar dat was natuurlijk altijd al mogelijk.

Systeem vereisten
De download bedraagt een forse 1,4 GB voor de 32- en 64-bit Windows en Macintosh (Intel) uitvoeringen in de Engelse taal. Een demo blijft 15 dagen werken met een watermerk eroverheen en zonder de 'Content Pack'. Een 64-bit uitvoering van de computer wordt herhaald aanbevolen, waarbij 32-bit nog kan worden gebruikt (buiten de deur) voor het afspelen van de LiveCubes. Zowel versie 7 als 8 van SketchUp (free en Pro) kunnen worden gebruikt om het 3d model te maken. Telkens wordt de schuld gelegd bij 32-bit alsof daar de grens van het programma wordt bepaald, dat lijkt ons niet juist, de problemen van grote modellen renderen liggen elders, bv. bij de juiste keuze van een grafische kaart plus 8 GB of meer aan RAM geheugen.
Het maken van een rendering vraagt een



goede grafische kaart en snelle computer. Men adviseert NVIDIA GeForce 8800 of nieuwer of een ATI Radeon 4000 of nieuwer met minimaal 512 MB VRAM geheugen.
Multi-core processor aanbevolen met 4 GB geheugen of meer. Hoe krachtiger het systeem hoe sneller de 'pre-processing' fase wordt uitgevoerd. Dat betekent dat de processor met meerdere cores plus de parallel processing van de grafische kaart samenwerken. Er wordt niet gesproken over CUDA, waar de fabrikanten kennelijk nog steeds bang voor zijn. (manufacturer lock)

De 8800 GT biedt 112 CUDA cores en 256-bit frame buffer, een PCI Express 2.0 bus is noodzakelijk (Windows). De prijs bedraagt tussen de 500 - 600 euro. De kloksnelheid bedraagt 1500 MHz met een minimale voeding van 400 Watt en hoger.
De Radeon 4000 duidt op een serie kaarten, De 4870 X2 levert 2 GB geheugen bij een klok van 1800 MHz. De 4870 heeft 512 MB geheugen en is daarmee minder geschikt voor grote modellen.
De prijzen zijn aanzienlijk lager dan bij NIVDIA maar de keuze zal toch niet moeilijk zijn.

Macintosh gebruiker
De Macintosh gebruiker komt er bekaaid vanaf. Deze snel groeiende groep van professionele gebruikers komt in de specificaties niet verder dan MacOSX Intel compatibel, zonder enige grafische kaart te noemen. MacOSX 10.4.11? 10.5 ? 10.6, 10.7 of 10.8, het blijft een open vraag op de internetpagina. Na lang zoeken komen we op een andere internetpagina een magere systeem "10.6" tegen, waarbij vermeld wordt dat andere versies zullen volgen. De 13" MacBook Pro's gebruikers met ingebouwde Intel processor kunnen we zo al afschrijven, dat gaat niet werken met SketchUp en LumenRT.

LiveCubes
Terug naar de feature: de LiveCubes. Een aanwinst zo lijkt het
http://www.lumenrt.com/download/#LiveCubes
Voor een demonstratie download tussen de 40 - 60 MB. Hou er wel rekening mee dat de renderingstijden voor een dergelijke LiveCube's tot vele uren kunnen oplopen, afhanke-

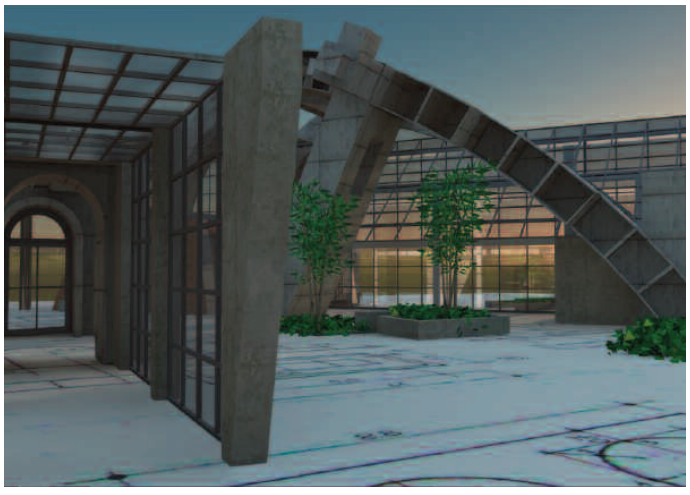
lijk van de kwaliteit van de gebruikte hardware. U kunt ze publiceren in een Mac of Windows format of beiden. De Mac uitvoering is een zip file die uitgepakt dient te worden. De fabrikant stelt zelf met een gemiddelde resolutie, dat het een kwartier of langer duurt voor een rendering klaar is.

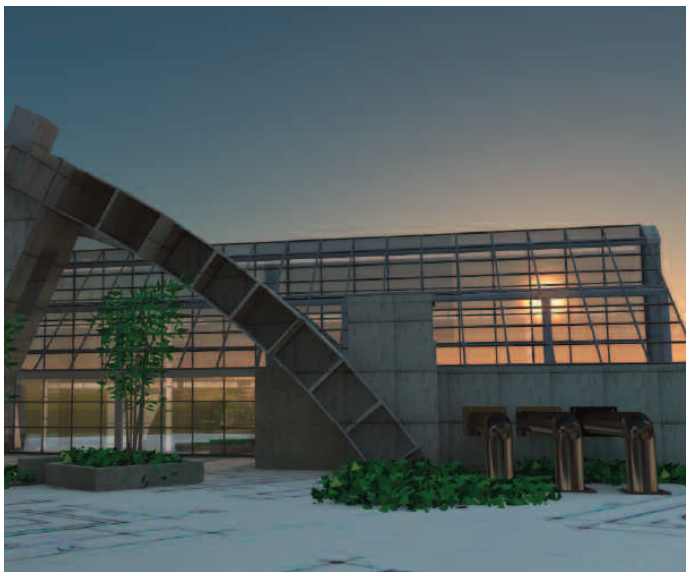
Met een MacPro met Intel 10.4.11 en GeForce 7300 GT NVIDIA kaart wilde de LiveCube niet starten.
Uitgepakt is de zip 59 MB file nu 153 MB.
In een iMac met 10.6.8, 2.06 GHz Intel duo core, 4 GB SDRAM NVIDIA GeForce 8800 GS grafische kaart met 512 MB geheugen start de cube wel.
De voorkeuze voor de functie toets F1 werkte echter niet, waardoor er geen keuzemogelijkheid voor animatie en Walk through direct beschikbaar komt, de animatie holt alle kanten op, verwarrend.

Na enige tijd is die controle er wel, alleen het beeld blijft springerig en bijzonder lastig te navigeren. Of dat aan de hardware ligt is niet bekend, het is in ieder geval ook voor deze hardware (bij de klant) geen goede eerste kennismakingsoptie. Bij het beëindigen van het programma kon de gebruikelijke Command Q of de muis niet meer worden gebruikt. Deze reageerden niet.

Het programma kon niet anders worden gestopt dan met "Geforceerd stoppen" via het dock en rechtermuisknop optie.

Wellicht dat in de loop van de tijd (wij beken LumenRT 3.0) er verandering komt in de manier van werken. De website vraagt om aanpassing met duidelijker informatie voor toekomstige gebruikers. (Meetdatum februari 2013)





LumenRT renderprogramma
Bloemlezing van enkele claims.

- physical lighting
- best picture quality on the market
- physically accurate lighting
- real-time visualizations
- real-time rendering
- physical simulation of light
- accurately simulates light, interacts with volume
- the only solution capable of lighting in real-time

LumenRT software creates real-time 3D visualizations of architectural designs.
Thanks to **physical lighting**, LumenRT produces the **best picture quality** on the market:

- **Take photos, shoot videos of your designs in real-time 3D**
  With photo-realistic quality and physically accurate lighting

Create high quality **real-time visualizations** with the push of a button, add **life and nature**, and let your **clients experience** your designs at their own pace!

LumenRT is the only solution that combines **real-time rendering** with a full, **physical simulation of light**, letting you accurately convey the true volume and lighting of your designs.

Pre-Processing Phase

LumenRT's pre-processing engine, built on e-on patented technologies, is capable of rapidly computing physically accurate lighting on arbitrary mesh geometries. These technologies virtually eliminate the need to optimize geometry prior to rendering.

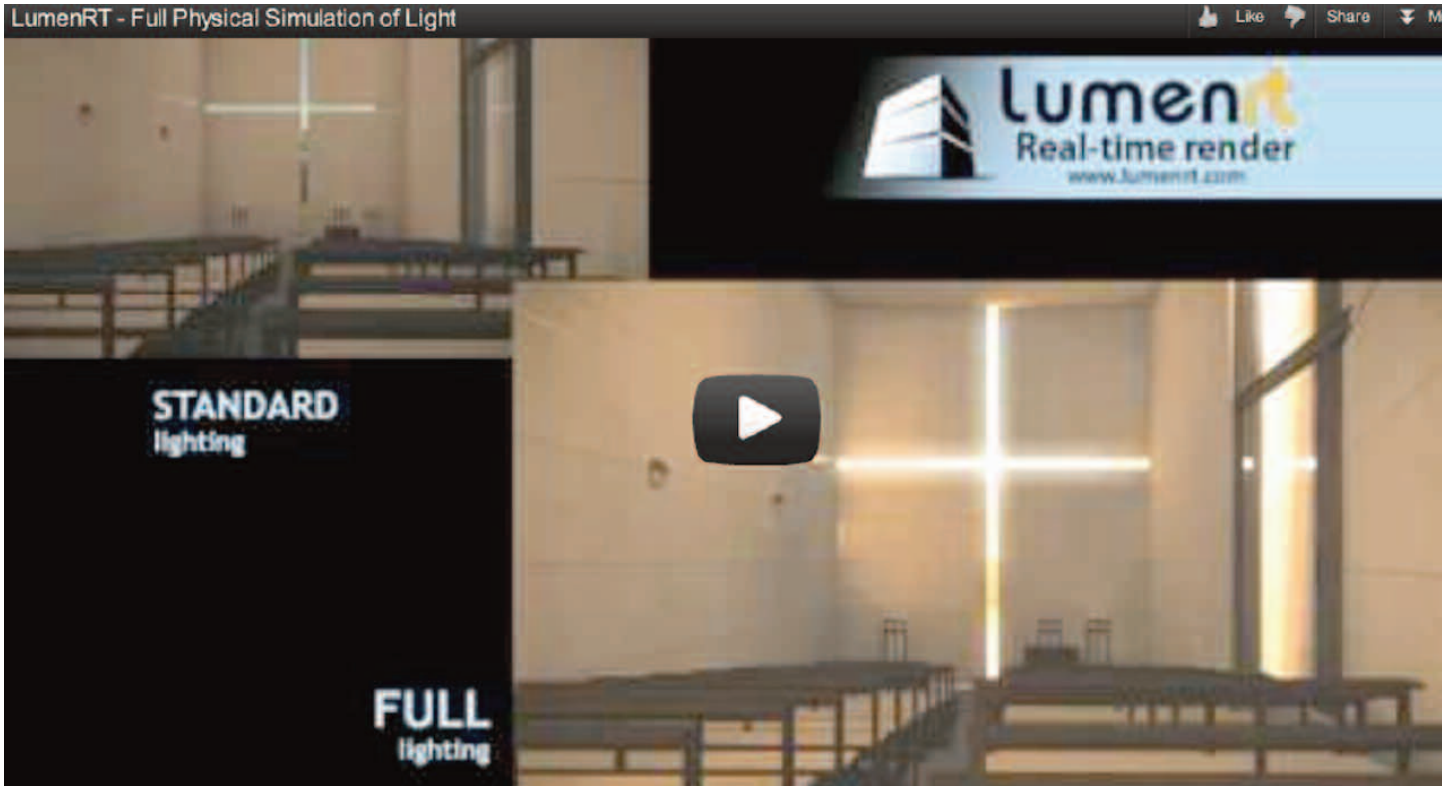
The Only Solution with Physical Lighting

Because light is an essential component of architectural designs, Architects should rely on tools that represent light with greatest accuracy.

Thanks to e-on software's patented Physical Lighting technologies*, LumenRT accurately simulates the way light interacts with volume. LumenRT is the only solution capable of reproducing all the subtelties of direct and indirect lighting in real-time!

** Multiple patents pending*

NB.
Opsomming en schermafdrukken begin januari 2013. Deze kunnen inmiddels zijn aangepast.



Lumion

http://lumion3d.com/

Lumion komt in twee uitvoeringen: Standard en Pro. Alleen voor Windows computers. De meest bekende 3D formats kunnen worden geïmporteerd. Dus ook die van SketchUp, AutoCAD, Revit Architecture, 3DS Max en vele andere.

De prijs voor de Pro is net even onder de 3.000 euro. De standaard uitvoering komt op 1.500 euro met een beperkte resolutie van 1920 x 1080 voor video frames. De resolutie voor de afbeeldingen bedraagt bij beide uitvoeringen 7680 x 4320 maximaal.

Update van de 3D-scene in renderingsprogramma vanaf de 3D software mogelijk met behoud van gemaakte instellingen.

Dit programma ondersteund en leunt geheel op de GPU kaart, waardoor veranderingen in 'Real Time' kunnen worden bekeken.
De Pro wordt met 1826 modellen met 459 bomen, planten en andere elementen uitgerust.

"All you need is a computer with a modern graphics card."
Veel meer technische informatie wordt echter niet verstrekt (!)

Hardware vereisten
minimaal:
OS: Windows XP, Vista, 7 (32- or 64-bit) en
DirectX 9.0c of nieuwer
3 GB RAM (voor eenvoudige scenes)
NVidia GeForce 9800 GTX of vergelijkbare ATI / AMD, met tenminste 1 GB VRAM.

Aanbevolen:
OS: Windows 7 64-bit
ten minste 6 GB RAM
NVidia 460GTX / ATI 5850 of sneller, met tenminste 1 GB VRAM.

Versie 3.0 Pro
De nieuwe Global Illumination werkt voor de zon en voor puntlichtbronnen. Het is een snelle manier waardoor het niet te veel aan renderingstijd kost.
GI is echter niet hetzelfde als bv. programma's als VRay, maar de winst is wel dat een hogere snelheid wordt bereikt.

De oude versie van Lumion geeft op YouTube een interessante film:
http://youtu.be/gr57E2YzQtQ
En deze ziet er ook goed uit:
http://youtu.be/pYHVVtdeCas

Jammer dat er geen Mac versie beschikbaar is, de opmerking van Virtueel Bootcamp gebruik lijkt, vanwege het probleem van een geschikte grafische kaart, niet juist.
Er zijn behoorlijk wat Middellandse Zee entourages bijgeleverd, maar wellicht dat er ook andere meer westerse optionele bibliotheken beschikbaar zijn. De mensen zijn van alle delen van de wereld.
Door het aan kunnen maken van een eigen materialen bibliotheek tesamen met een interessante Paint functie is dat aantrekkelijk.

De personen zien er goed uit en ze bewegen ook aanzienlijk beter dan bij sommige andere renderingsprogramma's.
De waterhoogte is in te stellen, waardoor er overstromings scenarios mogelijk zijn.
Vogels zijn eenvoudig in te stellen en ze vliegen direct. Stereoscopic 3d.

Snelheid
Wat opvalt is het gemak en de snelheid van het werken met het programma (in de YouTube films te zien), maar helaas is ook hier '***vergeten***' om de hardware te vermelden waarbij dat is vastgelegd. Maar zelfs zonder deze vermelding gaat het allemaal even vlot zo lijkt het. Het maken van animaties gaat op

een vergelijkende manier als bij SketchUp, de gewenste aanzichten worden gekozen en de daartussen in te vullen frames worden automatisch berekend.

Het water lijkt in de animatie vaak aardig, maar net niet natuurlijk. Wellicht is de snelheid van veranderen nog in te stellen waardoor het een betere indruk zal maken.

High End Grafische Kaart
Een goede (NVIDIA) kaart wordt aanbevolen waarbij de 'passmark punten' worden aangehouden van deze site

http://www.videocardbenchmark.net

Een redelijk goede kaart heeft tenminste 1500 punten, 2500 en liefst meer wordt aanbevolen.
Quadro en FirePro kaarten worden niet geadviseerd.
Voor deze applicatie van renderprogramma is het dus echt nodig om de Windows machine op te tuigen met een High-end Grafische kaart.

Over CUDA
In het Lumion Forum wordt door de Lumion Staff het volgende vertelt:
"We are using the cores throught DirectX so not CUDA. That doesn't mean it doens't use all those cores only in a different method."
Het wordt er niet duidelijker op. Ook het feit dat Quadro kaarten niet worden geadviseerd is vreemd te noemen. Kennelijk wordt voor een deel op DirectX teruggevallen, waarbij enig rekenwerk (vandaar de Passmark punten) door parallele processoren in de grafische kaart wordt uitgevoerd.

Ook hier zou de directie van Lumion wat duidelijkheid kunnen verschaffen aan de toekomstige koper.

Via het NVIDIA System Panel (GTX 670, GTX 680 en hoger) zou het mogelijk zijn om Anti-Alias aan te zetten, waardoor de renderingen duidelijk aan kwaliteit winnen. Door gebrek aan optimale informatie en professionele beantwoording van de Forum vragen blijven de klanten afhankelijk van de meer- of minder professionele kennis van mede renderprogramma gebruikers. Aangezien niemand exact weet hoe het werkt blijven er blinde spots. Mede gezien de vrij stevige prijs zouden we hier toch iets anders verwachten.

Lumion Forum
"I'm currently running a Nvidia Quadro 4000 and would like to know if a GTX series card would offer greater performance."

"Do you guys at Lumion have benchmark stats posted anywhere for this kind of stuff?

I would use this site...i do not think that the Dev team has access to all configurations of cards...but generally the higher the passmark score the better. Make sure you look at single core cards only if that is what runs best for Lumion".

http://www.videocardbenchmark.net/

ARTISAN

Een geheel op de processor georiënteerd render programma dat baat heeft bij veel cores en snelle kloktijden.
Artisan wordt als programma code (OEM) verkocht aan diverse renderprogramma's van naam.

Interessant om te zien hoe de diverse rendermerken omspringen met het geven van informatie over in wezen één het zelfde renderonderdeel in hun eigen interface. Sommige merken bespreken het onderwerp niet en het woord "Artisan" duikt daarbij soms op in Forums. Anderen hebben het logo (rechts boven) op hun website staan. Weer anderen verzinnen er gewoon een andere naam bij om aan te geven dat hun programma nu is voorzien van een render module.

http://www.belightsoft.com/products/liveinterior/renderboost.php

LiveInterior 3D modeller met de nieuw ondergebrachte "Render Boost" plugin van Live Interior 3D.

Work Between Live Interior 3D and Render Boost.

Weggestopt in de pagina's: *"powered by Artisan, Focused Rendering Solutions"*
Bij Liveinterior 3D heet de render programma plugin "Render Boost". Ze vertellen er wel bij dat in het hart van Render Boost het Artisan Render Engine is ondergebracht, een volgende generatie product van Lightworks. Wat het volgende daarin betekent is niet duidelijk. CPU gebaseerde renderers zijn bepaald niet toekomst gericht.
Pricing: Starting from $150.

System Requirements:
Live Interior 3D Pro or Standard Edition installed.
Mac OS X 10.5.8 or later, Mac OS X 10.8 Mountain Lion compatible.
Intel-only processor.
Discrete AMD Radeon or NVidia graphics with 256 MB or more.

http://www.novedge.com/products/2132/tab/1

Winkel met allerlei render programma's, waaronder ook Artisan wordt genoemd.

Novedge met **Kompas Artisan Rendering Plug-in**. Het is alleen voor Windows.

Kompas-3D V13 voor $ 3.499 'complete solid modeling solution. Er zijn ook licentievormen vanaf 6 maanden, 9, 12 maanden, maar ondanks deze segmentering blijft de prijs fors. De Rendering Kompas Artisan plug-in wordt ook los verkocht voor $ 599,-

http://www.zwsoft.com/artisanforzw3d/

ZWsoft is de ongekroonde Chinese tegenhanger van **AutoCad**, een 3D rendering oplossing wordt nu geboden middels Artisan for ZW3D CAD / CAM.
Het Artisan render programma wordt aangeprezen als de *"next generation rendering plug-in from Lightworks"*. Waarbij Lightworks, zoals zovele wordt gebombardeerd tot *"World's leading supplier of rendering solutions for developers of advanced 3D computer graphics software."*

Zie http://www.zwsoft.com/artisanforzw3d/
Alleen voor het Windows platform.

http://www.lightworkdesign.com/new-kompas-3d-v13-from-ascon-integrates-lightworks-artisan.html

New **KOMPAS-3D V13** from ASCON integrates Lightworks Artisan.
In december 2011 wordt vertelt dat Lightworks de laatste versie heeft uitgebrach van Kompas 3D van ASCON. Ascon is een ontwikkelaar en professionele CAD / AEC / PLM oplossingen.
Ascon zou een van de eerste CAD ontwikkelaars zijn op de Russische markt, het programma kan bogen op 40.000 klanten over de hele wereld. Samen met de Artisan render eenheid is Ascon ook de eeste Artisan gebaseerde renderer in Rusland.

Lightworks zou in deze pagina's:
"Renowned for providing physically accurate visualisation of real-world objects and environments, Lightworks is the ideal choice for companies wanting to target the Architectural, Interior and Mechanical Design markets."

http://www.imsidesign.com/Products/Renditioner/RenditionerPro/UITutorials/tabid/1754/Default.aspx

Renditioner v3 voor SketchUp

Artisan is ondergebracht in "Renditioner Pro", prijs $ 200,00.
Waarbij wordt vertelt dat de gebruikers nu ook toegang krijgen tot de nieuwe Lightworks Artisan Website met bonus materiaal, textures en achtergronden.
Alleen Windows.

Renditioner Pro v3 'maakt goede renderingen mogelijk van uw SketchUp modellen'.

http://twilightrender.com/phpBB3/viewtopic.php?f=16&t=2388

TwilightRender.com
Prijs $ 99,-.

Renderen binnen SketchUp met deze plugin Met **Twilight Render**, is het nu mogelijk om foto realistische beelden binnen SketchUp te maken.

Biased en Unbiased methoden met een onbeperkt aantal processoren (cores).

Twilight biedt " Physically Accurate Materials, photometric (IES) lights, image based (HDR) lighting etc."

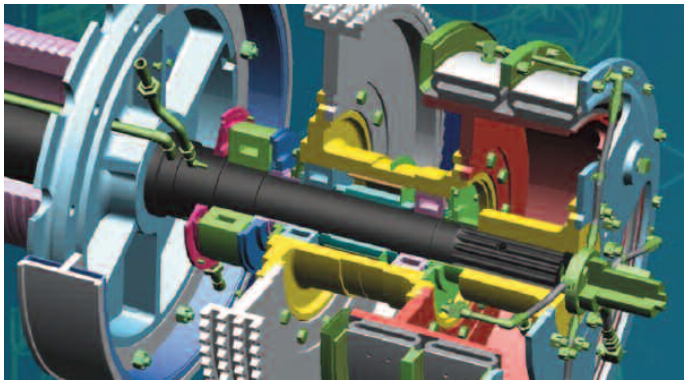
Twilight Render
* Ray Tracing
* Photon Mapping
* Path Tracing
* Bi-directional Path Tracing
* Metropolis Light Transport
* Clay, Depth, Alpha Mask, and more
* Use the selected geometry or the entire scene

Camera Setup
* Render exactly what you see in SketchUp
* Perspective, parallel, cylindrical, spherical
* Tightly integrated with SketchUp's projection, FOV, Aspect Ratio
* Interactive Tone Mapping that updates while you render.

Siemens
http://www.plm.automation.siemens.com/en_us/products/open/parasolid/comp-toolkit-providers/

Complementary Toolkit Poviders
Parasolid

Lightwork gebruikt Parasolid om een integratie te ontwikkelen met hun Author, Aspect en Artisan geïntegreerde render onderdelen. Duidelijker wordt het niet.



Parasolid van Siemens

Artisan zou contracten hebben afgesloten met meer dan 80 OEM firma's. Bovenstaand hebben we er een aantal opgesomd. Waar het getal 80 vandaan komt blijft echter onduidelijk. Wij hebben ze niet kunnen vinden. Of er zijn nog render programma's die het woord "Artisan" nergens vermelden, maar toch het render gedeelte van de codes inkopen.

Het lijkt onwaarschijnlijk want de gebruikte zwarte interface en de bijbehorende toegang tot materialen en texturen zou het een en ander snel verraden.

Het komt zo lekker over tijdens een beurs (CEBIT), in een verkoop gesprek, op een internet pagina of tijdens een webinar:

*"Real-Time Ray Tracing simulates the **physics of light** in an environment so whenever . . ." Maar is wetenschappelijk op geen enkele wijze te onderbouwen. Er is geen enkele renderfirma te vinden, die dat heeft overwogen of geprobeerd.*

We zien bij renderprogramma's die een deel van de codes inkopen een gemengde filosofie. De één probeert de naam te wijzigen en er een eigen commerciële draai aan mee te geven.

De ander verzwijgt het zo veel mogelijk en nog een ander zet het logo op zijn website en prijst zijn programma met deze OEM toevoeging juist aan.

SpectralPixel

http://www.spectralpixel.com/

Volgens de fabrikant: “Simulates the flow of light like in the real world, according to physical equations, thus producing image of outstanding quality”.
“Great image quality and full control”.
De films gaan over auto’s, terwijl er wel wordt gesproken over renderingen voor architectuur, film, media, entertainment, TV en aanverwante doelgroepen. Het programma lijkt nog in de beta 0.8 fase.

CPU en GPU gebaseerde voordelig geprijsd renderprogramma*.
Zowel op de CPU als op de GPU. Het is dus aan de klant en diens hardware welke, of beide systemen worden gebruikt.

Het aanpassen van materialen en alle instellingen in real-time bekijken is volgens de fabrikant realiteit. In de YouTube film is de rendersnelheid goed te noemen, alleen naar hardware specificaties kunnen we alleen maar raden, ze worden niet vermeld.
“Don’t waste time waiting for an image”.

De Preview is op OpenGL gebaseerd met een WYSIWYG editing methode.

Camera effecten kunnen worden gesimuleerd met ‘echte optische’ instellingen, scherpte diepte, brandpuntsafstand en lens grootte instellingen, auto focus.

De materialen zijn op OSL (Open Shading Language) gebaseerd, waardoor verbetering en aanpassingen mogelijk zijn.

CPU of GPU gebaseerd
CPU : Multi-processor /
Multi-core / Multi-thread
CPU sub-core
CPU : SSE support
GPU : Multi-GPU support

Voor CPU dient u OpenCL SDK te downloaden van Intel.
En voor GPU dient u te beschikken over een goede NVIDIA grafische kaart en dito driver.

*** Koop nu?**
Van wie is het programma of waar is de firma gevestigd, blijft niet alleen vaag, maar ook onvindbaar.

Bij internet onder **whois** komen we uit bij:

Domain Discreet Privacy Service
ATTN: spectralpixel.com
12808 Gran Bay Pkwy, West
Jacksonville, FL 32258
US

Vreemd om dat niet te vermelden, dat schept in het geheel *geen vertrouwen*. Mede door de beta fase is het wellicht te verklaren dat het Forum vrijwel leeg is.

POV-Ray renderer

http://www.povray.org/

POV-RAY STUDIE PROJECT VOOR RENDEREN

Bij het bekijken van de website ziet u al dat het om een bijzonder renderprogramma gaat, zo u wilt een *studie project*. Aan de bovenzijde missen we de bekende kopjes: "buy" en "resellers".
POV-Ray is een soort Open source programma (lees de speciale licentie voorwaarden: NOT PUBLIC DOMAIN**) voor Windows, MacOS en MacOSX plus i86 Linux computers. De broncode is ook beschikbaar.**

Het is in eerste instantie door David K. Buck en Aaron A. Collins geschreven (2001, het "POV-Team" genoemd). Het 'hoofdkantoor' is het internet. David Buck begon interesse in renderen te krijgen toen een vriend van hem de C-code voor een RayTracer voor Unix van internet had gedownload. David bouwde het om naar de Amiga (DKBTtrace) en schreef de drivers om het op het scherm te kunnen weergeven. Aaron Collins bracht middels C-code het naar het Windows platform en voegde het Phong lichtmodel er aan toe.

Gedateerd
De laatste POV-Ray versie van Windows dateerd al weer van juni 2009, hetgeen betekend dat de vaart er uit is geraakt. Het feit dat zelfs OX 9.2 van Mac wordt ondersteund en OSX 10.2.8 als laatste versie wordt opgevoerd, doet vermoeden dat de ontwikkeling een aantal jaren geleden stil is komen te liggen. Wel prettig dat het nog steeds op internet wordt aangeboden. Mac Intel gebruikers zouden moeten wachten tot POV-Ray 3.6 Mac OSX zou uitkomen. De PDF handleiding (256 pagina's in A4 PDF format) is overigens van POV-Ray Version 3.6.1.

Op zich interessant voor mensen die nog kunnen beschikken over een oudere Mac, waarbij de ondergrens OS 8.1 is, 8.6 is een stabielere versie met CarbonLib 1.0.4 of nieuwer voor een PowerMac. (iMac, iBook, Mac G3, Mac G4, Mac G5, Cube en de dubbele nummers 5200, 6300, 7200, 8100 en 9600 die

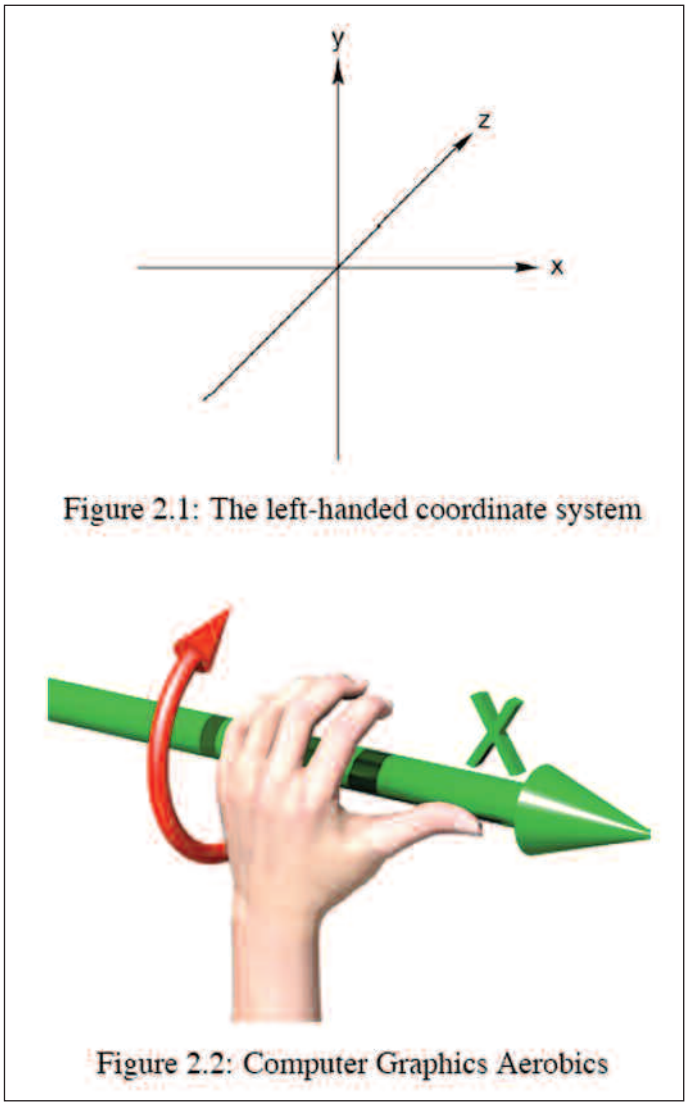
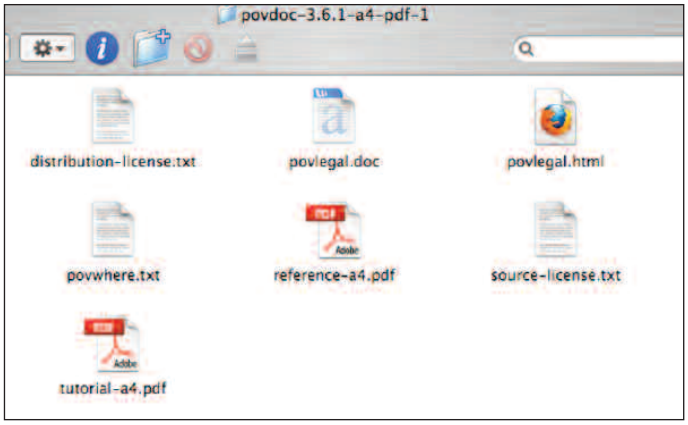


Figure 2.1: The left-handed coordinate system

Figure 2.2: Computer Graphics Aerobics

© Afbeeldingen Reference manual. Het gebruikte coördinaten stelsel bij POV-Ray.



Bij de literatuur "Tutorial-a4.pdf" en de "Reference-a4.pdf". Beide PDF's aantrekkelijk indien u de basis principes van het renderen en de corresponderende programmeercodes wilt leren kennen.

overeen komen met de eerste Power Mac serie computers. Windows gebruikers kunnen nog met Windows 95, 98 of NT aan de slag. Er zijn ook nòg oudere versies beschikbaar via FTP, waarbij de start "official-1.0 van juli 1992" dateert. De laatste versie bij die pagina is van augustus 2009 de "Official-3.6".

Ray Tracing
POV-Ray maakt gebruik van Ray-Tracing, de algemene al omvattende naam om een afbeelding van een 3D scene op te bouwen door simulatie van de manier waarop lichtstralen zo'n virtueel landschap zouden kunnen schijnen. Ook hier wordt het proces van lichtstralen niet vanaf de lichtbron, maar vanuit het oog en renderingsmasker gedaan. POV-Ray maakt voor Ray Tracing gebruik van de computer processor en niet van de grafische kaart. De grafische kaart dient er voor om de beelden op het scherm te krijgen, bijvoorbeeld met OpenGL.

Reference-a4.pdf
Naast de Tutorial handleiding is er ook nog een 393 pagina tellende "Reference-a4.pdf" beschikbaar met referentie naar POV-Ray opties, de scene programmeertaal en Quick Reference. Zie fig 4.5 (pag. 127) met de verhouding tussen pixels en driehoeken vanuit de 3D scene. Duidelijk is weergegeven welk coördinatenstelsel ze bij POV hebben gebruikt.
Fig 4.14 Geometrie van spotlight.
Fig 4.19 Area light 4 x 4, plaatsing met vectoren.
pag. 194 uit de Reference pdf: **Ambient light**. Dit licht kan niet direct met Ray Tracing worden opgeroepen, maar met een truc "*Ambient Lighting to simulate the light inside a shadowed area*" wel.
Photon map het eerst geïntroduceerd door Henrik Wann Jensen, zie pagina 261.
Fig 6.5 Reflective caustics pag. 261.

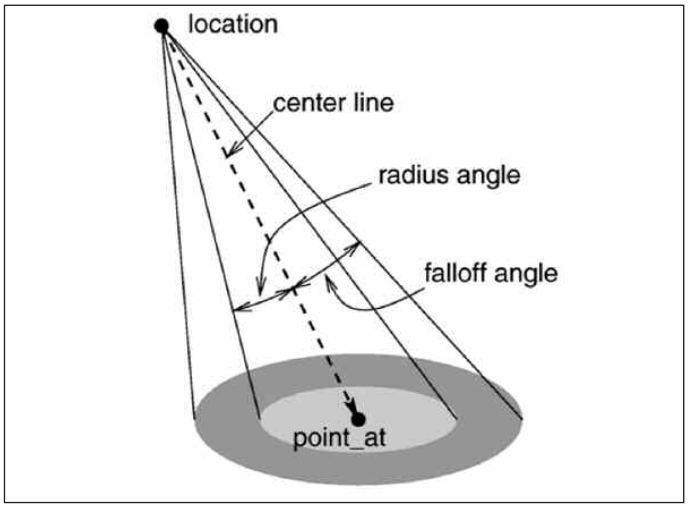


Fig. 4.14 Geometrie van een 'spotlight'.

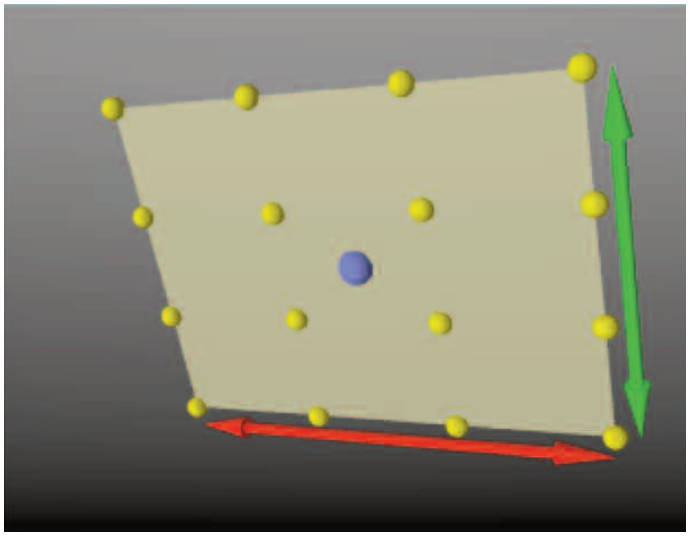


Fig. 4.19 Area Light 4 x 4, plaatsing met vectoren.

You know you have been raytracing too long when ...
... You have gone full circle and find your self writing a scene that contains only a shiny sphere hovering over a green and yellow checkered plane ...
– Ken Tyler

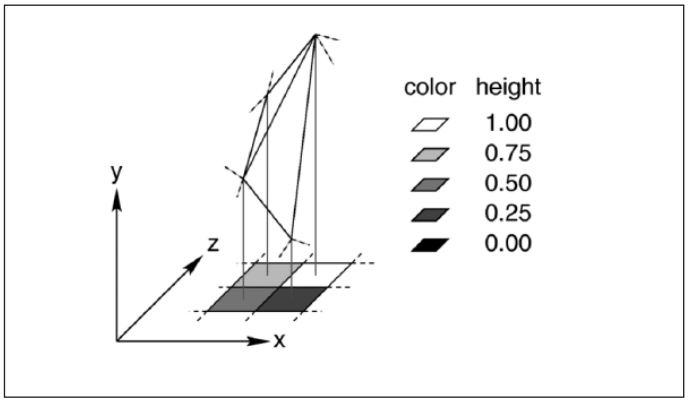


Fig. 4.5 Pixels en driehoeken in hoogte veld.

De handleiding POV-Ray "Tutorial-a4.pdf"
Voor het coördinaten stelsel werd gekozen voor het links-handig systeem, zie afbeelding aan het begin van dit hoofdstuk. De duim wijst daarbij naar de X-as in het 3D stelsel x-y-z. Het standpunt van de camera brengt al meteen een complexe eenheid in de 3d scene met zijn x-, y- en z-coördinaten.
De coördinaten worden als een driedimensionale vector beschreven.
Bij POV-Ray wijst de positieve y-as recht omhoog, de positieve x-as naar rechts en de positieve z-as naar het scherm. Bij andere 3d programma's en render programmas komen we vaker tegen dat de positieve z-as recht omhoog wijst.

```
camera {
  location < 0, 2, -3 >
  look_at  < 0, 1, 2 >
}
```

De eerste plaatst de camera twee eenheden omhoog en 3 eenheden terug vanuit het midden van de ray-tracing ruimte met coördinaten $< 0, 0, 0 >$
Standaard wordt +z aangenomen als binnen het scherm en -z als buiten het scherm.

Met de tweede programmaregel wordt de camera gedraaid naar het punt $< 0, 1, 2 >$
Er wordt een simpel object en texture bijgevoegd om bij de volgende aan te komen: de lichtbron:

```
light_source { <2, 4, -3> color White }
```

pag. 160 van de reference handleiding
De **Photon mapping** techniek kan in POV-Ray worden gebruikt om fantastische afbeeldingen met reflecterend licht en spiegeling van objecten te maken. Zie afbeelding 3.72 waar het 'echte' licht ook zichtbaar gemaakt is in de scene. Met reflectie en Radiosity verkrijgen we het beeld van 3.73.

pag. 195 “The Trace macro”, waar de virtuele stralen in het 3D model worden geschoten om te beoordelen wat er in de rendering per pixel moet worden opgenomen.

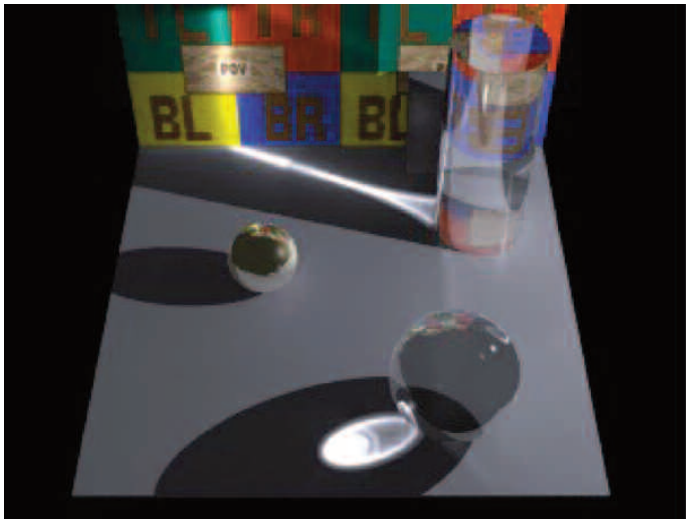


Fig. 6.5 Reflectie caustics met POV-Ray.

Fig. 3.72 Geavanceerd verstrooiings voorbeeld met POV-Ray.

Fig. 3.73 Verstrooiings voorbeeld met Photon maps in POV-Ray.

Tips en trucs *pag. 224*
Hoe kan met POV-Ray de render snelheid worden verhoogd? Ook in die tijd was daar al aandacht voor:

1. toepassen van bounding boxes
2. aantal lichtbronnen beperken
3. Area lights zijn langzaam om uit te rekenen, gebruik ze zo weinig mogelijk
4. Voeg texturen op diverse objecten bijelkaar
5. Als een object zowel reflecteert als refractie geeft zoals glas, dan wordt het renderen aanzienlijk langzamer. Stralen kunnen in het object heen en weer worden gestuurd.
5b. Probeer in dat geval het aantal lichtstralen (max_trace_level) te beperken.

You know you have been raytracing too long when ...
... Your friends are used to the fact that you will suddenly stop walking in order to look at objects and figure out how to do them as CSGs.
– Jeff Lee

You know you have been raytracing too long when ...
... You find yourself wishing you'd paid attention in math class to all those formulae you thought you'd never have any use for in real life.
– Jeff Lee

You know you have been raytracing too long when ...
... You actually read all the documentation that comes with programs.
– AmaltheaJ5

De in de reference handleiding aangegeven literatuurlijst:

1. "An Introduction to Ray tracing" Andrew S. Glassner (editor)
ISBN 0-12-286160-4; Academic Press; 1989
2. "Realistic Image Synthesis Using Photon Mapping" Henrik Wann Jensen
ISBN: 1568811470; AK Peters; July 2001
3. "3D Artist" Newsletter, "The Only Newsletter about Affordable PC 3D Tools and Techniques")
Publisher: Bill Allen; P.O. Box 4787; Santa Fe, NM 87502-4787; (505) 982-3532
4. "Image Synthesis: Theory and Practice" Nadia Magnenat-Thalman and Daniel Thalmann;
Springer-Verlag; 1987
5. "The RenderMan Companion" Steve Upstill;
Addison Wesley; 1989
6. "Graphics Gems" Andrew S. Glassner (editor);
Academic Press; 1990
7. "Fundamentals of Interactive Computer Graphics" J. D. Foley and A. Van Dam;
ISBN 0-201-14468-9; Addison-Wesley 1983
8. "Computer Graphics: Principles and Practice (2nd Ed.)" J. D. Foley, A. van Dam, J. F. Hughes;
ISBN 0-201-12110-7; Addison-Wesley; 1990
9. "Computers, Pattern, Chaos, and Beauty" Clifford Pickover;
St.Martin's Press;
10. "SIGGRAPH Conference Proceedings"; Association for Computing Machinery Special Interest Group on Computer Graphics
11. "IEEE Computer Graphics and Applications"; The Computer Society;
10662, Los Vaqueros Circle; Los Alamitos, CA 90720
The POV-Team no longer recommends books from CRC Press. Go read Eric's commentary13 to find out
why.

websites:
9 http://www.povray.org/
10 ftp://ftp.povray.org/
11 http://www.povray.org/groups.html
12 http://www.povray.org/

CentiLeo renderer

http://www.centileo.com/about.html

Nieuw OEM renderprogramma met een wel zeer bijzondere specialiteit: fotorealistisch interactief renderen van heel grote 3D modellen met goedkope personal computers of zelfs labtops.

De aloude droom blijkt dan toch eindelijk in vervulling te gaan, maar dit programma is niet zó te koop, het is programma code die door renderprogramma's kunnen worden aangeschaft om hun eigen programma GPU proof te maken èn voor een groot hardware aanbod geschikt te maken.

Homepage:

- about
- problemen bij (alle) andere renderingsprogramma's

About
Een misleidende kop voor een bedrijf dat zich zelf totaal onzichtbaar heeft gemaakt. U kunt wel in 'contact' met ze treden, maar u dient al uw n.a.w. gegevens door te sturen, zij vertellen niet waar ze zijn gevestigd, in welke plaats, in welk land of werelddeel.

Elke website heeft een domeinnaamregistratie en via deze omweg komen we er achter dat het domein op 12 april 2012 is geactiveerd. En het domein administration met een beschermd E-mail adres is gevestigd op ID#10760, PO Box 16, Nobby Beach, null, QLD 4218 in Australië.
De oostkant van Australië aan de goudkust onder Brisbane, zie schermafdruk verderop.

Omdat Octane Render in Nieuw Zeeland is gevestigd zou er een verband kunnen zijn, via Whois voor render.otoy.com komen we niet verder, want dit domein in de Los Angeles USA aangevraagd op 10 april 2002. Die zijn al veel langer aan het ontwikkelen. Er lijkt eerder een weg te zijn naar de *Russian Academy of Sciences*, waar een deel van ontwikkeling is gedaan.

Perpetuum mobile
Latijn = voordurend of eeuwig bewegend. Te mooi om waar te zijn. Gaat dat ook op voor dit renderprogramma?

Problemen bij andere programma's
CentiLeo vertelt: "een aantal oude render systemen hebben beperkingen, ze kunnen langzaam zijn, beperkte fotorealistische eigenschappen hebben en ze kunnen alleen een middelmatige complexe geometrie en texture ondersteunen vanwege beperkingen van het systeen. Wat 'systeem' ook moge inhouden, hardware of software òf beiden?"

"Veel getalenteerde artiesten worden gedwongen om dure computers aan te schaffen en soms is dat kostbaar, tegelijk krijgen ze veel beperkingen in huis. Ze kunnen een beperkte hoeveelheid data verwerken of ze kunnen niet snel genoeg renderen of ze kunnen niet zo creatief zijn als ze zouden willen! Ze worden verplicht om hun 3D wereld aan te passen, de vrijheid van beelden maken te laten varen en creatieve ideeën tijd en geld over de balk ... " u begrijpt het al.

Centileo biedt uitkomst: "wij ontwikkelen een profesioneel render systeem CentiLeo waarbij de standaard render taken (VFX Movie of CAD) worden geoptimaliseerd. CentiLeo levert fotorealistisch Ray Tracing met ondersteuning van allerlei animatie geometrie (?), onbeperkte hoeveelheid textures, programmeerbare shaders. CentiLeo onder-

steund het werken met honderden miljoenen unieke polygonen waarvan naar eigen inzicht animaties kunnen worden gemaakt.

Inclusief deformatie en expansie met veranderende topologie. De hoeveelheid textures die vastgelegd wordt voor de geometrie wordt bepaald door de hoeveelheid harde schijf ruimte.

En al deze aantrekkelijke eigenschappen werken interactief samen op standaard PC's of laptops."

Ze vervolgen: "Technisch gebruiken we de rekenkracht van de GPU (CUDA), de meest krachtige rekenbron voor de massa markt. Een aantal andere programma's hebben óók een snelle interactie door het gebruik van de GPU. Maar, grafische kaarten hebben een beperkte hoeveelheid onboard geheugen, dat altijd onvoldoende is voor de 3D artiest in zijn uitdagend beroep om modellen en renderingen te maken."

"CentiLeo technologie is speciaal en de SNELSTE IN DE WERELD om dat probleem op te lossen. Wij hebben een hoog vermogen virtueel geheugensysteem ontwikkeld om algemene Stochastische Ray Tracing mee te kunnen doen.

Daarmee wordt het mogelijk om 10 - 20 GB (of hogere) 3D modellen in te laden en met GPU's te laten renderen met slechts 1 - 2 GB GPU geheugen per kaart."

"CentiLeo rendert grote 3D scenes net zo snel als kleinere, die bij andere firma's in het werkgeheugen van de GPU kaart(en) moeten passen."
Stelt u zich eens voor: dat een enkele GPU kaart fysiek 25 GB of 100 GB aan geometrie kan verwerken? Hoeveel zou dat moeten kosten? En wanneer zou dat beschikbaar komen? Wilt u daar op blijven wachten? Met CentiLeo kunt u nog steeds uw huidige computer blijven gebruiken, niet gedwongen om een kostbare nieuwe PC aan te schaffen en toch uw uitdagende (grote) 3D projecten te ontwikkelen."

Duidelijk een bedrijf 'in opkomst' met een wellicht een postbus ergens in Australië. CentiLeo vervolgt: "In toekomstige uitvoeringen van het programma hebben we nog op onze lijst staan:

1. ook niet polygonen renderen
2. het bereiken van het hoogste fotorealistisch niveau
3. het opzetten van eigen render applicaties
4. het integreren van onze individuele versnellings stappen in bestaande oplossingen.

Demonstratie CentiLeo op de SIGGRAPH 2012 (stand 1012).
http://www.centileo.com/news.html

Met de kop: "CentiLeo rendert video beelden voor grote scenes op een laptop"

In twee YouTube filmpjes wordt de kijker meegezogen in de heerlijkheden van grote modellen en animaties maken. De ene is de bekende New York scene met **8 GB aan textures en 25 miljoen polygonen**.
Tussendoor verschijnen de teksten:

Global Illumination Reflections
GPU Ray Tracing performance
Over 8 Gigabyte textures
Over 25 million triangles
360 million dynamic polygones
Full Frame Production
You never know it is possible
On a single laptop GPU

"Boeing's vliegtuig bestaat 360 miljoen polygonen (die worden ondergebracht in 14 GB's geometrie data). In beide films werd elk frame gerenderd met 1000x met progressieve Path Tracing (Ray Tracing) op 720 px, resulterende in een glade Global Illumination met alle reflecties in de stand AAN. In de laptop duurt elk frame ongeveer 5 minuten."

De laptop is uitgerust met de **NVIDIA GeForce GTX 485M**.
Deze snelle Game kaart (met oude generatie PCIe 2.0 aansluiting) werkt met CUDA, DirectX, Optimus, 3D Vision, Verde drivers, SLI en PhysX.

384 CUDA cores, 1150 MHz kloksnelheid, 36,8 miljoen/sec texture fill rate.
De geheugen kloksnelheid bedraagt 1500 MHz, GDDR 5 geheugen met 2 GB aan geheugen. Op de NVIDIA pagina's is dat getal niet te vinden. Bij testen van deze kaart in de NIVIDA Windows menu is 2 GB af te lezen.
De 485M is gebaseerd op de GF104 chip, waarbij in andere uitvoeringen een aantal CUDA cores niet meewerken.
De 348M presteert ongeveer 1/3 van wat de nieuwe GeForce GTX 670 (PCIe 3.0) doet in een desktop computer.

http://www.notebookcheck.net/NVIDIA-GeForce-GTX-485M.42883.0.html

Conclusie
Gebaseerd op de CentiLeo website en PDF bestanden januari 2013.
Kennelijk heeft men de flessenhals van elk GPU render programma geprobeerd op te lossen door het geheugen van de grafische kaart(en) niet voor de opslag van de geometrie te gebruiken, maar daar het grotere RAM geheugen van de computer in te zetten. Dat heeft het grote voordeel dat RAM geheugen veel beter en goedkoper beschikbaar is dan geheugen van de grafische kaart, die in zijn meest kostbare uitvoering niet verder dan 6 GB komt voor desktop computers. Het nadeel is het verschil in snelheid van RAM ten opzichte van VRAM van de grafische kaart. Kaartgeheugen is vele malen sneller en uitwisseling tussen deze twee snelheden kost tijd.

Daar komt bij dat de geometrie in het voorbeeld van de Boeing van 360 miljoen wordt teruggebracht naar 14 GB (RAM geheugen).

Indien we voor het gemak uitgaan van minimaal 20 byte voor opslag van geometrie (pag. 41) dan komen we met 360 miljoen polygonen uit op 7200 miljoen bytes = 7,2 GB. Zij komen uit op 14 GB hetgeen zou betekenen dat ze ongeveer 40 bytes per polygoon opslaan. Om daar snel mee te kunnen werken zal een gedeelte daarvan toch in de grafische kaart moeten worden geladen (bij de voorbeelden wordt met een laptop met 2 GB kaartgeheugen gewerkt en dat kan alleen indien een verdere reductie van 14 GB naar

minder dan 2 GB wordt toegepast. Een soort compressie. Het maximale 2 GB kaartgeheugen is uiteraard niet in zijn geheel beschikbaar, wellicht dat 1,5 GB direct beschikbaar is voor gecomprimeerde geometrie. Ofwel men past reductie in het aantal polygonen toe zoals met bijvoorbeeld DirectX. Ofwel men heeft een eigen routine daarvoor geschreven of er is nog een andere oplossing.
Ook de textures zullen in RAM moeten worden gevangen om de grootte hoeveelheid in onder te brengen. Overzetten en ophalen naar de grafische kaart kost wederom relatief veel tijd. In OpenGL zijn tegelijk slechts 120 stuks mogelijk.

Polygoon reductie behoeft niet altijd direct in een verminderde kwaliteit van de render afbeelding zichtbaar te zijn. Verderweg gelegen objecten worden het meest gereduceerd, dichterbij gelegen objecten het minst.

http://www.blitzbasic.com/Community/posts.php?topic=75365
Any good reduction tools?

De **New York** scene is niet zo interessant, de resolutie is laag en met 5 minuten per frame komen we voor deze film toch aan een forse rendertijd. De kwaliteit / sturing van licht en contrast is onvoldoende.

De **Boeing film** is indrukwekkend, mede door de uitslag van het 3D model en door de grote hoeveelheid polygonen die toch maar op het scherm worden getoverd in een resolutie van 1024 x 768 px. Met een lage 100 samples per pixel volgens Raytracey.blogspot.nl die zijn gegevens van de You Tube film SIGGRAPH uit april 2011 haalde.

Met de onduidelijke waarde van Frame Rate van **You Tube** films is het toch wel aannemelijk dat deze in de buurt van de 30 frames/sec ligt. Dat wil niet zeggen dat de renderinganimatie met dezelfde framerate is opgenomen. Met FRAPS kan een en ander worden gemeten (Frame Rate Programma). Nemen we aan dat het 30 is geweest dan komen we aan 30 x 5 x 66 seconden (duur You Tube film) is een behoorlijke tijd.
Leg daar dan wel de gebruikte resolutie en


Whois
Identity for everyone

centileo.com Registry Whois

Domain Name: CENTILEO.COM
Registrar: PDR LTD. D/B/A PUBLICDOMAINREGISTRY.COM
Whois Server: whois.PublicDomainRegistry.com
Referral URL: http://www.PublicDomainRegistry.com
Name Server: NS1.SPACEWEB.RU
Name Server: NS2.SPACEWEB.RU
Status: clientTransferProhibited
Updated Date: 12-apr-2012
Creation Date: 12-apr-2011
Expiration Date: 12-apr-2013

centileo.com Registrar Whois

Registration Service Provided By: WHOIS.COM
Website: http://www.whois.com

Domain Name: CENTILEO.COM

Registrant:
PrivacyProtect.org
Domain Admin (contact@privacyprotect.org)
ID#10760, PO Box 16
Note - All Postal Mails Rejected, visit Privacyprotect.org
Nobby Beach
null,QLD 4218
AU
Tel. +45.36946676

Creation Date: 12-Apr-2011
Expiration Date: 12-Apr-2013

Domain servers in listed order:
ns1.spaceweb.ru
ns2.spaceweb.ru

Administrative Contact:
PrivacyProtect.org
Domain Admin (contact@privacyprotect.org)
ID#10760, PO Box 16
Note - All Postal Mails Rejected, visit Privacyprotect.org
Nobby Beach
null,QLD 4218
AU
Tel. +45.36946676

de uiteindelijke kwaliteit naast, die laag is te noemen.

Voor grote 3D modellen zou dit programma in de toekomst een oplossing kunnen zijn. De kwaliteit van dit moment is onvoldoende voor serieuze architectuur of interieur renderingen. Voor animatie films ligt dat natuurlijk anders. Daar komt bij dat de technische achtergrond gegevens onvoldoende zijn om een indruk te krijgen van de toegepaste techniek. Omdat er in de tekst wordt gesproken over een soort OEM programma ligt het voor de hand dat men samenwerking zoekt met renderprogramma's om de technologie in onder

te brengen. Er is geen forum aanwezig en in de menubalk ontbreekt "Buy". Aan de duidelijkheid, ook wat naam, adres, woonplaats gegevens ontbreekt het helemaal.
In de renderforums van andere programma's wordt het al opgepakt en zal het voorlopig nog wel even blijven rondzingen, met meer of minder kennis van de materie.

Luxology
http://forums.luxology.com/topic.aspx?f=4&t=68788

Blog
http://raytracey.blogspot.nl/2011/04/centileo-brand-new-interactive-out-of.html
Ray Tracey's blog met interessante neven informatie van een medewerker van OTOY (met ondermeer Octane Render). Waar de achtergrond gegevens voor de tijdberekening van de Boeing film van 2 minuten bij 30 Frames/s op 15 uur vandaan komt is niet duidelijk. Maar wellicht via onderstaande links en samenvattingen wordt meer informatie over dit thema bekend.

CentiLeo SIGGRAPH 2012 demo
http://www.cgrecord.net/2012/08/centileo-siggraph-2012-demo.html

Blender
http://blenderartists.org/forum/showthread.php?261987-CentiLeo-GPU-memory-limit-no-longer-exists

GPGPU site (General Purpose Computation on Graphics Hardware)
http://gpgpu.org/2011/08/04/centileo-out-of-core-ray-tracer

SIGGRAPH 2012
http://www.siggraph.org/s2011/content/out-core-gpu-ray-tracing-complex-scenes

CGSociety (Society of digtital Artists)
http://forums.cgsociety.org/archive/index.php/t-1063001.html

7 augustus 2012
Huge-Scene Interactive Rendering on a Laptop **CentiLeo** is a high-performance rendering engine that interactively renders huge scenes composed of hundreds of millions of polygons on a laptop accelerated with a GPU. Using the fastest ray tracing to render photorealistic images, the system supports arbitrarilly dynamic 3D scenes (deformed or exploding) and a huge number of large textures. Scenes can contain dozens or hundreds of gigabytes, and still the frames are interactively updateable on a laptop with a GPU.

CentiLeo LLC
beschrijving bedrijf
http://sis.siggraph.org/cgi-bin/procform?&SESS=u2hdd7xpc1RwVGh%2Bm2enEgMohsvoGjxAV%2B8RWYTU4na4A/2JTmj1LTgzr0lsSswnTvnv7tr9pbn0%2BL2sX9SGt66ej11gOnv8jzntJQ4O6Mn9RKvx2pUG9H6T6KSqAH%2BF&preparsed=1

Octane Render
http://render.otoy.com/forum/viewtopic.php?f=25&t=15078&start=10

Documentatie
http://www.tabellion.org/et/paper11/index.html
Coherent Out-of-Core Point-Based Global Illumination.

OutOfCorePBGI.pdf 8 pagina's

gc2011Garanzha.pdf 4 pagina's
Simple Geometry Compression for Ray tracing on GPU

http://dl.acm.org/citation.cfm?id=2037826.2037854
ACM heeft betaalde PDF documentatie beschikbaar over dit onderwerp. De gratis op te halen verkorte PDF heet *"Out-of-core Ray Tracing on Memory Limited Throughput Architectures".*

De ontwikkelde techniek:

Out-Of-Core Ray Tracing
Een nieuwe methode om complexe grote scenes met CUDA grafische kaarten met beperkte geheugens te renderen. Waarbij film wordt genoemd als eindproduct. Met een efficiënt werkend cache geheugen en page-

swapping van RAM naar grafische kaart geheugen. Verder wordt een eenvoudige data compressie toegepast. Het gebruik van de CUDA GPU maakt het mogelijk om tussen de 10 - 20 keer zo snel te rekenen dan met de CPU mogelijk is. De aangeboden Ray Tracing Engine is in staat om een grote verscheidenheid aan rendering en licht algoritmes te herbergen. Inclusief Photon mapping, Metropolis Light Transport, bi-directional Path Tracing etc. Om toegang te krijgen tot de opslag van de trefpunten, textures en andere Ray Tracing taken werd een speciaal Data Management Systeem ontwikkeld.
We nemen aan dat de complete gecomprimeerde geometrie in RAM kan worden opgeslagen en met de Data Manager worden pages gemaakt waarbij een deel daarvan (klein genoeg om VRAM van GPU te passen) naar de GPU worden gestuurd, waarbij bestaande data wordt overschreven.

De pages worden in batches van ca. 1 MB samengevoegd om de page data overdrachtsnelheid te verbeteren. Bij de (oude) PCIe 2.0 bus zal elke 1 MB batch met 5,6 GB/s kunnen worden vervoerd. Met de beperkte bandbreedte van het naar RAM kopiëren (ca. 4 GB/s) komen de makers op een gemiddelde van 3 - 3.5 GB/s voor de overdracht van CPU naar GPU.
De data management cache bevat 700 MB aan geo-informatie, hetgeen experimenteel bepaald is door een groot aantal praktijk testen uit te voeren.

Data reductie
De bedoeling is om zoveel mogelijk virtuele geometrie in de GPU te brengen. Daarmee wordt de beperkte transportsnelheid beter benut. Met een efficiënt geometrie quantization algoritme wordt een eenvoudige data reductie bereikt.

Inspiratiebronnen worden vermeld
De ontwikkelaars hebben een uitgebreid verslag gemaakt van de inspiratiebronnen voor de ontwikkeling van deze techniek. Daarbij zien we duidelijk dat de ontwikkeling bij render programma's wordt gestimuleerd door ondermeer SIGGRAPH congressen. Eén van de genoemde inspiratiebronnen is OptiX.

NVIDIA blijkt al bij versie 2.5 (2011) een soort out-of-core systeem te hebben toegevoegd. Waardoor **OptiX** (NVIDIA PDF) een aantrekkelijke keuze is geworden voor Render programma's. OptiX wordt nl. continu ontwikkeld en de licentie is geheel gratis!

OptiX is momenteel in versie 3.0 waarbij zelfs het idee van *CUDA render only* geheel vaarwel is gezegd. Deze render onderdelen incl. het Ray Tracing gedeelte kunnen ook ZONDER GPU functioneren op een normale CPU.

OptiX is het Ray Tracing's kader voor programmeurs. Om vlot Ray Tracing toepassingen te ontwikkelen met goede resultaten. OptiX is na aanmelding licentievrij.

Een aantal CentiLeo schermafdrukken van 12 april 2011 SIGGRAPH.

In 2011 ook al bij SIGGRAPH aanwezig:
Out-of-Core GPU Ray Tracing of Complex Scenes - this talk covers the CentiLeo GPU ray tracer (based on Kirill Garanzha's PhD research at the Keldysh Institute of Applied Mathematics), which can render models composed of several hundred million polygons in real-time. More information on CentiLeo (including a video of it in action) can be found here.

CentiLeo

Current capabilities

- Path tracing at HD resolution
- For scenes with 100s million polygons such as Boeing 777
- Works interactively on desktop GPU with only 1.5GB of memory

CentiLeo

Implemented

- Out-of-core ray-polygon intersection search
- Out-of-core texture fetch
- Large ray queues
- NVIDIA CUDA code
- The scene should fit into DDR3 memory

Algorithms

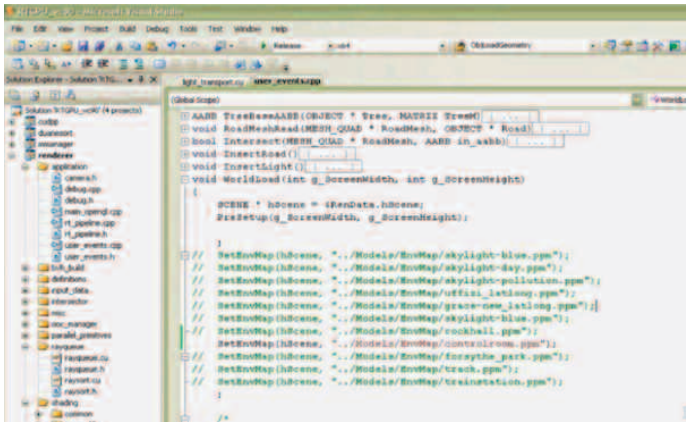
- The system is built around a novel GPU virtual memory manager (VMM)
- High-performance and scalable GPU memory cache for large models
- Precise: no any LOD or any other approximation tricks
- The design is not limited to polygons or path tracing only

CentiLeo

Demo: Interactive work (5.5 minutes):

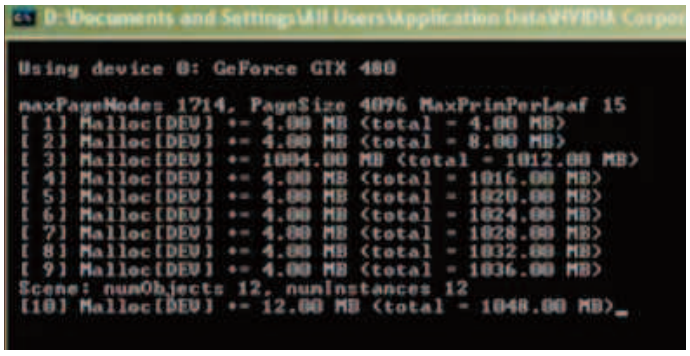
- Interactive navigation through Boeing 777 model (360 million polygons)
- Interactive Global Illumination preview (path traced)
- Increased rays throughput (Million rays / second) when more samples per pixel (SPP) are shoot - used for faster final image convergence
- Works on GeForce 480 GTX (also requires 16GB of DDR3 RAM)

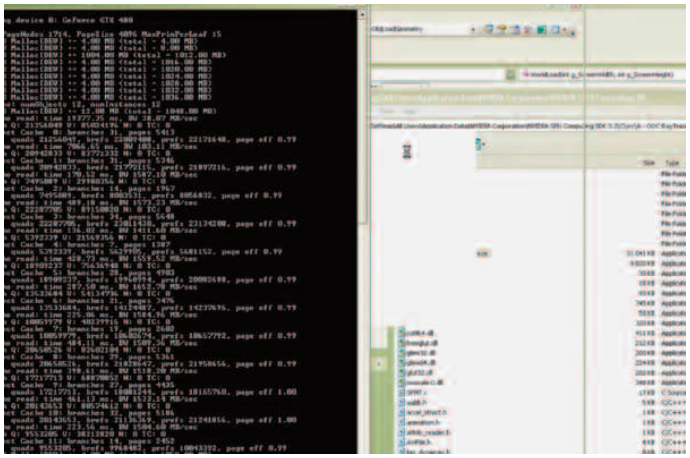
Rechts het uiteindelijke render resultaat van de achtergrond alleen.

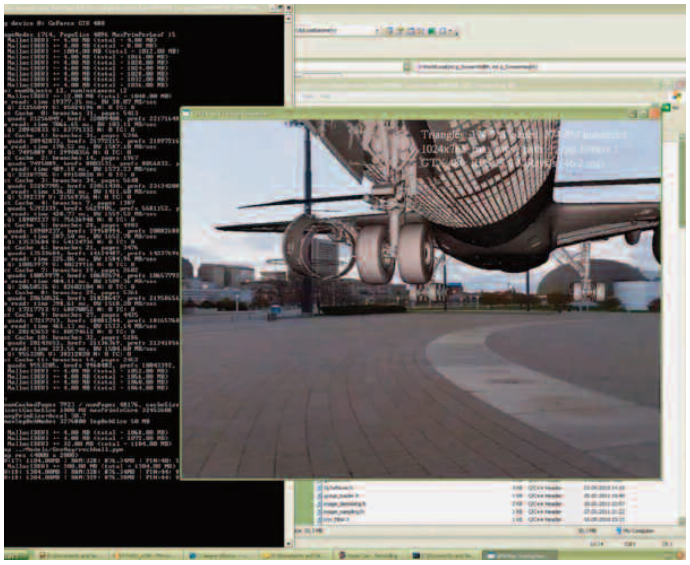


CentiLeo programmacodes.

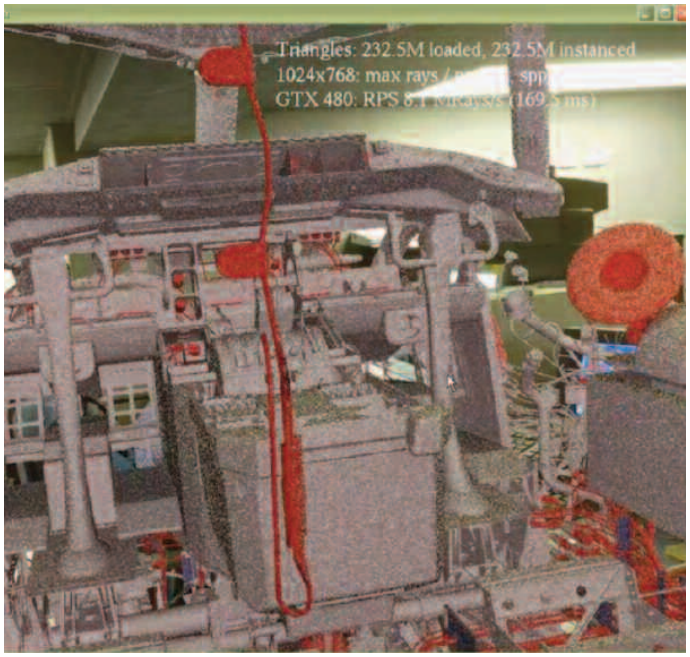
Afbeelding onder:
Links van de codes is een terminal waarbij de snelheid van de dataloader wordt weergegeven.

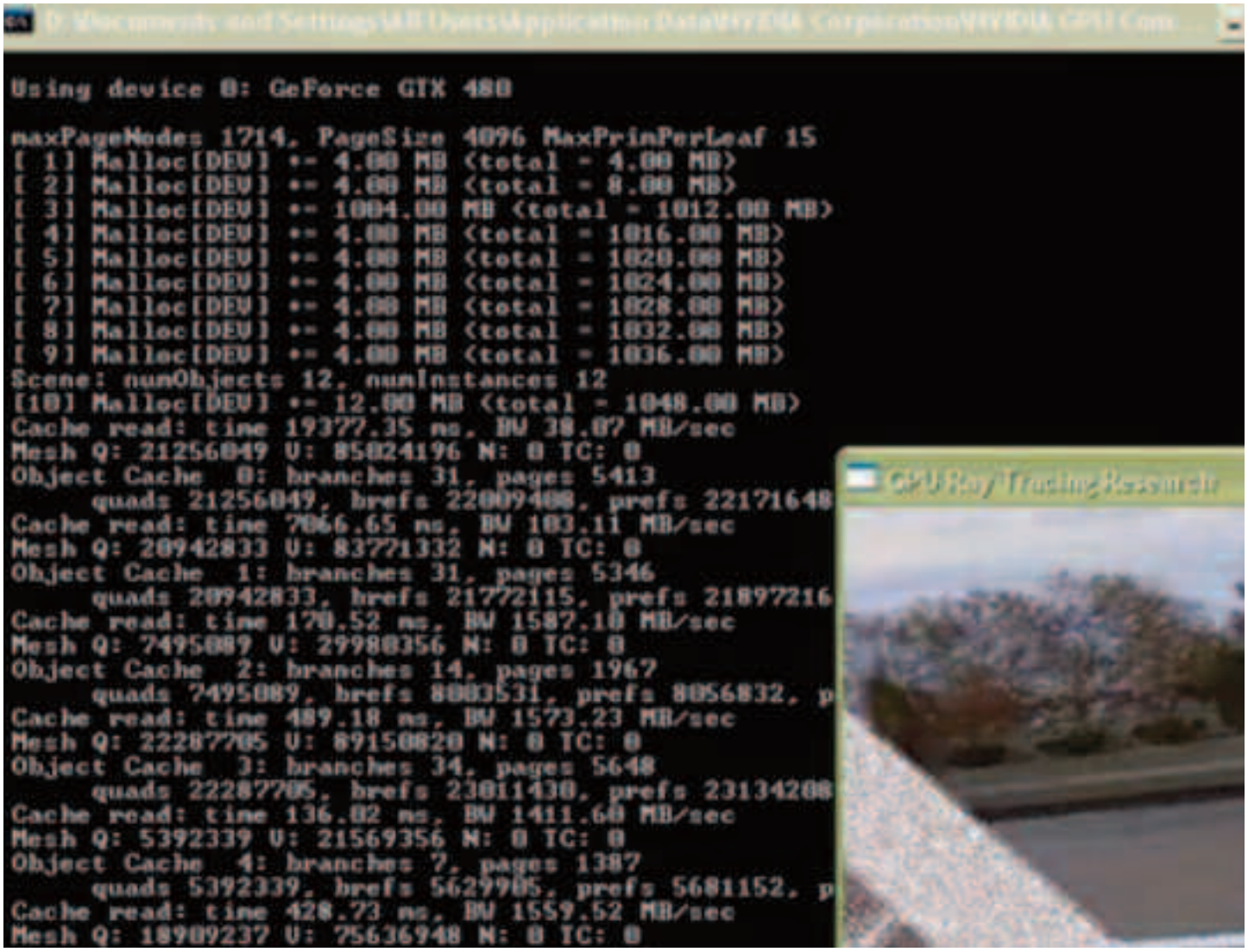






Verander de programma codes en lees het resultaat in de terminal af en zie direct het render resultaat in beeld verschijnen.
374,9 miljoen driehoeken geladen, res. 1024 x 768. max rays /path 1, spp 1, iters 1, GTX 480, RPS 13,6, MRays / s 16,6 ms.
SPP = number concurrent samples (paths) per pixel.
Iters = number progressive image iterations, every additional iteration reduce image noise. Bij het verhogen van de SPP waarde wordt de doorvoersnelheid van de stralen vergroot. De ruis in het beeld verdwijnt sneller, het uiteindelijk frame wordt sneller opgebouwd.



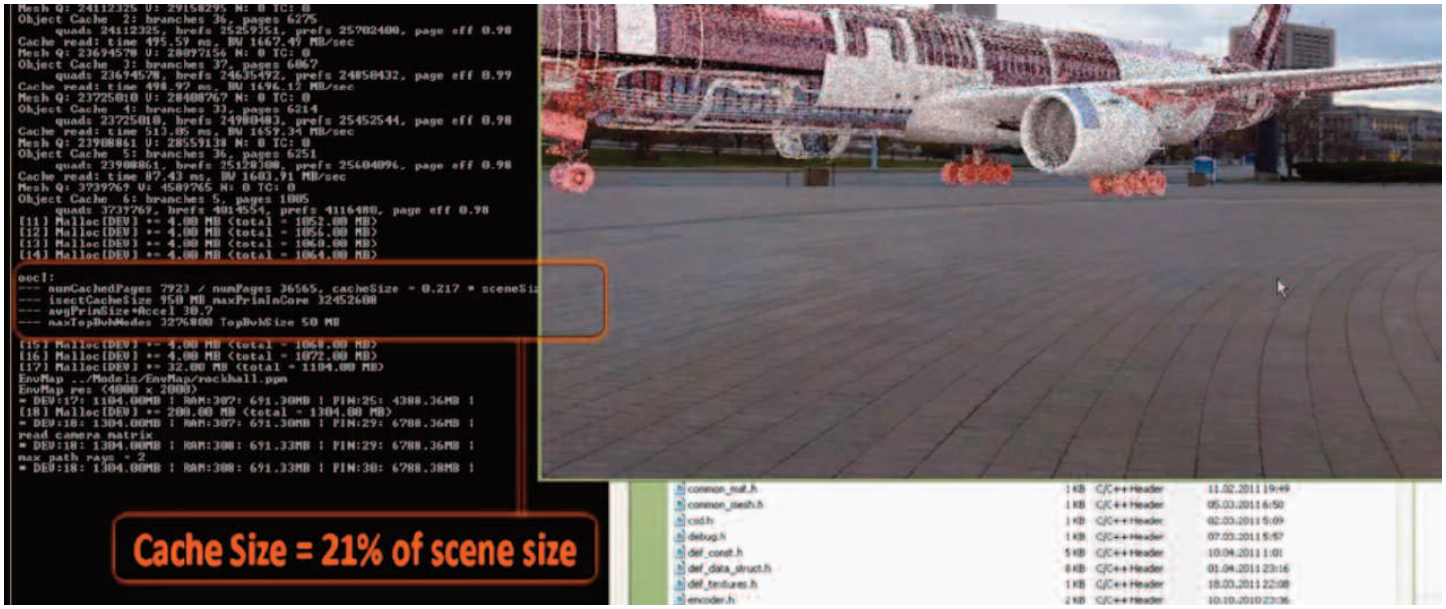


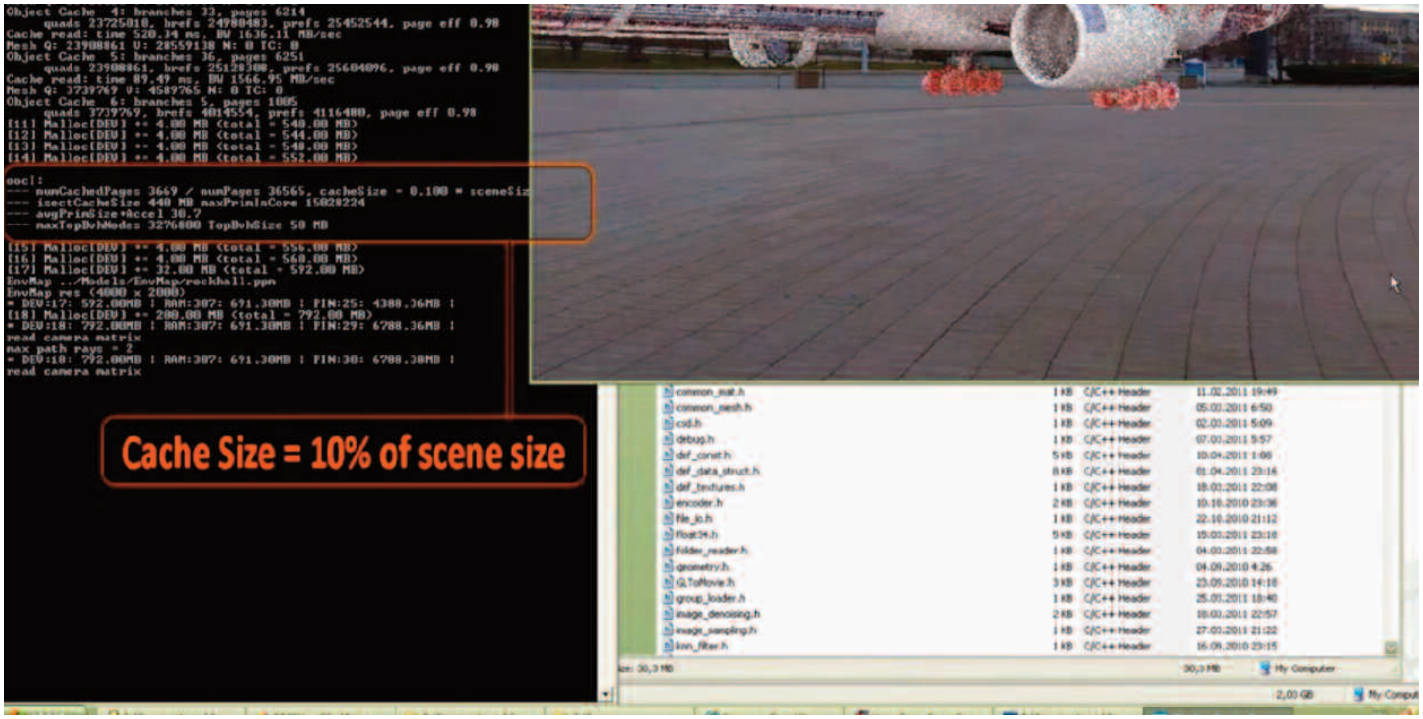
CentiLeo

Demo: Virtual Memory Cache Scalability

- Test performance of stochastic path tracing
- for GPU memory cache size = [21% of scene size … 0.1% of scene size]
- We can't allocate more GPU memory than 950 Mb (21% of tested scene size)

Alle schermafdrukken © CentiLeo.

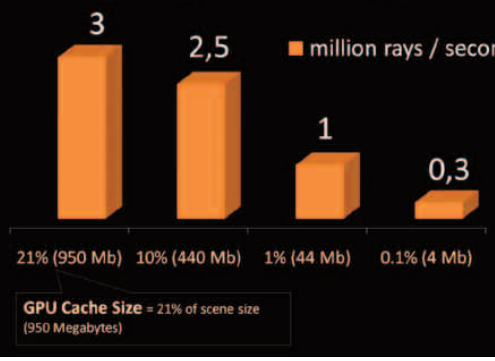




Global Illumination aangezet, afbeelding al na enkele seconden met snel afnemende ruis.

CentiLeo

Path tracing throughput for 280 million poly scene



CentiLeo

- Even for the cache that is 1000x smaller than the scene size the ray tracer works faster compared to the modern editors working with such huge scenes

- For the fixed GPU memory cache (such as 950Mb) and increased DDR3 (just more slots in MoBo) it is possible to accelerate the GPU rendering of the scenes with many billions of polygons

CentiLeo

Future features (already designed)

- 1 billion poly scene loader
- Disc-level for the cache hierarchy
- Fastest out-of-core scene data base builder
- Crack-free curved patch tessellation + displacement mapping
- Hair & Fur
- Texture filtering
- Advanced light-transport algorithms (bidirectional path tracing, MLT)
- Programmable material shaders

CentiLeo

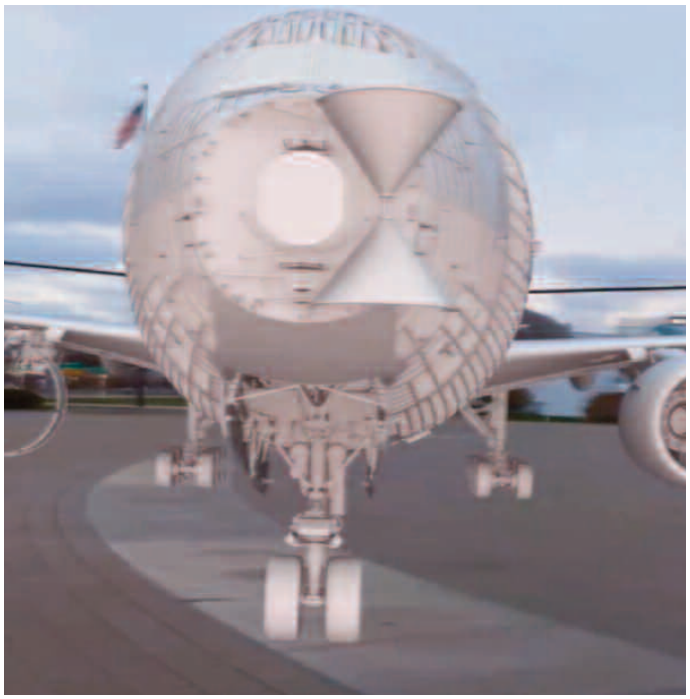
Potential scalability

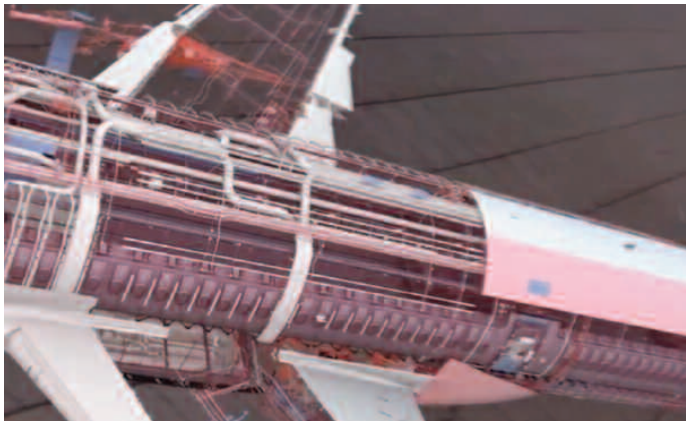
- Render interactively the scenes of Movie industry on the desktop PC
- Increase render performance with more DDR3 memory
- Increase render performance with more GPUs

CentiLeo

Demo: Final video Playback

- Final frame: 1024x768 @ 100 path traced samples per pixel
- Boeing 777 (360 million polygons)
- average frame computation takes 15 seconds on GTX480
- 2 minute video @ 30FPS was computed in 15 hours





Conclusie

Het schrijven over Render programma's is moedwillig op de rand van de afgrond gaan staan, waarbij duidelijk tekenen van erosie zichtbaar zijn. De in deze PDF beschreven programma's zijn geen statische objecten, maar veranderen continu met gratis- en betaalde updates om nieuwe opties toe te voegen en/of om bestaande problemen te verminderen of zelfs op te lossen. De korte samenvattingen van de render programma's zijn dan ook momentopnamen die in de loop van de tijd snel door de werkelijkheid kunnen zijn ingehaald. Hou daar bij het lezen altijd rekening mee! En controleer steeds de gegevens van actuele websites.

Verder is er een trend dat bestaande CPU programma's, meestal zonder enige vooraankondiging van de een op de andere dag een andere uitvoering uitbrengen die CUDA GPU gebaseerd is. Heeft u daar met de aanschaf van een nieuwe computer GEEN rekening mee gehouden, dan is helaas deze interessante snellere optie niet te gebruiken, totdat u zelf nieuwe hardware hebt aangeschaft. (zie ondermeer PDF werkstations).

Maxwell Render Forum
Ook bij dit programma (2004 verschijningsdatum voorafgaande aan 2 jaar ontwikkeling) is er een duidelijke trend van de gebruikers richting GPU om lange rendertijden een halt toe te roepen. Dave Morley: "*... We acknowledge that it may be a slow render*". Zie inzet rechts.

Overweegt u een renderprogramma aan te schaffen?
Controleer dan eerst de huidige hardware gegevens en noteer deze.

Operating systeem versie / serv. pack
Processor / cores
Snelheid processor
Hoeveelheid RAM geheugen
PCIe slots, *alle* specificaties
Grafische kaart(en)
Geheugen grafische kaart
CUDA ondersteund?

MAXWELL RENDER FORUM

Any plans for Maxwell to run on GPU processor ?
PostPosted: Thu Aug 16, 2012 3:42 pm

It will be the best for this render !!!!

All compagny go to GPU rendering see Octane, Cycles, Arion, Indigo, Iray, Centileo ....

I think this is one of the most awaited features for Maxwell, the power of the GPU and SLI working for make awesome renders in minutes, no hours. +1

I am interested in fast workflow without limitations. If that includes GPU, cool, if not, oh well. Last time i tried Arion a year or so ago it was not substantially faster than Maxwell for final renders. VRAY RT is cool, when it works, and when your scene does not use any of the myriad functions it does not support.

Can we have a reaction from Nextlimit team?

And if iRay may have less accuracy of Maxwell (you wont notice it in most productons), there are other engines with all kind of accuracy - i.e. Octane, which is spectral-based as well.
This is not to say these engines are better than Maxwell. I love Maxwell. But production-wise, GPU road *IS* a concrete possibility for a large improvement, right now. All the rest are words.

Ook hier blijft een officiële reactie met een serieuze beantwoording uit.

Wilt u met de bestaande hardware gaan renderen, bepaal dan de opties van de diverse programma's. Wilt u verder kijken en evt. nieuwe hardware overwegen, kijk dan eerst naar de op dat moment beschikbare programma's en prijzen. En bepaal welke hardware daar het best voor geschikt is. Bekijk de bijgevoegde PDF over "Werkstations" voor uitvoerige hardware informatie.

Standaard en goed ingevoerde renderprogramma's niet in deze PDF uitgave genoemd
We kennen eind 2012 een reeks van renderprogramma's die goed zijn ingevoerd in de Benelux :

sa = stand alone
pl = plugin v. SketchUp

Artlantis Render	(sa)
Artlantis Studio	(sa)
Render[in]	(pl)
SU-Podium	(pl)
Shaderlight	(pl)
V-Ray	(pl)

Dit zijn veel gebruikte render programma's. Zonder uitzondering zijn het **CPU** gebaseerde programma's. Waarbij **V-Ray** in de toekomst zou worden omgebouwd naar GPU en RT. Informatie van **Shaderlight** wijst erop dat ze al twee jaar achter de schermen bezig zijn met een GPU versie. Als tussenoplossing werd een Cloud render via een bedrijf in de VS aangeboden voor een te hoge stuksprijs. Een officiële aankondiging of bevestiging voor GPU uitvoering is er nog niet geweest. **Artlantis** heeft het in het Forum over "het staat op onze wensenlijst" en daar moet de klant al jaren maar mee doen.

Deze renderprogramma's met een prijs vanaf ruim anderhalf honderd- tot meer dan duizend Euro aan toe, hebben op dit moment nog het voordeel dat ze aansluiten bij een breed terrein van hardware. En in de praktijk (lees de Forums er maar op na) blijken veel render gebruikers over middelmatige hardware te beschikken met een aantal uitschieters naar boven van specifieke render bedrijven, die wel aanzienlijk hebben geïnvesteerd in de optimalisatie van de hardware. Zowel op het gebied van CPU (snelheid en aantal cores) als GPU met NVIDIA kaarten. Maar ook met het gebruik van relatief 'simpele' hardware, zelf tot op laptop niveau, met een NVIDIA CUDA grafische kaart en slimme programma's (CentiLeo) kunnen nu al opzienbarende resultaten worden bereikt.

Nieuwe programma's bieden vaak een ruime keuze aan hardware, waarbij de klant zelf zijn keuze kan bepalen.

Een aantal van deze uitspraken is juist, gedeeltelijk juist of geheel onjuist !
Lees ook de opmerking van programmeur *Mike Farnsworth* over Buzzwords met de 180° draai die een technische term kan nemen.

Maxwell Render
"Maxwell Render is physically accurate. Maxwell Render emerged from pure physics algorithms that treat light and its interaction with objects as in the real world. Our engineers have continuously rejected the use of tricks and shortcuts that could sacrifice quality or add artificial settings"

Arion
"Is a hybrid-accelerated and physically-based production render engine. It inherits all our expertise in light simulation and makes it run on steroids, thanks to our very unique massively parallel GPU+CPU approach. Is a high performance production renderer capable of producing hyper-real images and animations."

Indigo
"Indigo Renderer is an unbiased, physically based and photorealistic renderer which simulates the physics of light to achieve near-perfect image realism."

Fryrender
"The natural evolution of both Arion 1.x and Fryrender 1.x. It is a coninuation of both products. Arion is the new renderer."

OctaneRender
"Physically based GPU renderer"
Physically based / Spectral Light Transport-Unbiased and Direct Lighting / Ambient Occlusion
Custom Sampling Algorithm (Custom MLT-like implementation)

Artlantis Maxwell Engine
"Considered to be the most physically accurate rendering engine on the market. With Artlantis 4.1 you get the best of both worlds: Speed and Physical Accuracy."

Mental Ray
"Physically Accurate Lighting"
"is a high performance, photorealistic rendering software that produces images of unsurpassed realism from computer-aided design and digital content creation data, by relying on patented and proprietary ray tracing algorithms."

LuxRender
"is based on PBRT, a physically based ray tracing and unviased program. Is a physically based and unbiased rendering engine. Based on state of the art algorithms, LuxRender simulates the flow of light according to physical equations, thus producing realistic images of photographic quality."

Indigo Render
"Physically correct modelling of glass". With V-Ray for SketchUp, users now have one of the most powerful rendering tools available to visualize their models with the utmost quality and realism."

Spectral Studio
Photo realistic renderer, unbiased and physically correct.

Thea Render
"Multiple state-of-the-art biased and unbiased engines."

Shaderlight
"Accurate lighting. Illuminate your scene using SketchUp's sun, image based lighting or our physical sky. For realistic interiors simply add a point, spot, area, portal or IES light from Shaderlight's simple light editor."
Current versions (Win/Mac) are 2.3.2
The site is undergoing maintenance and is currently unavailable.
Please check back later. From december 2 2012 on. Inmiddels is de website weer in de lucht.

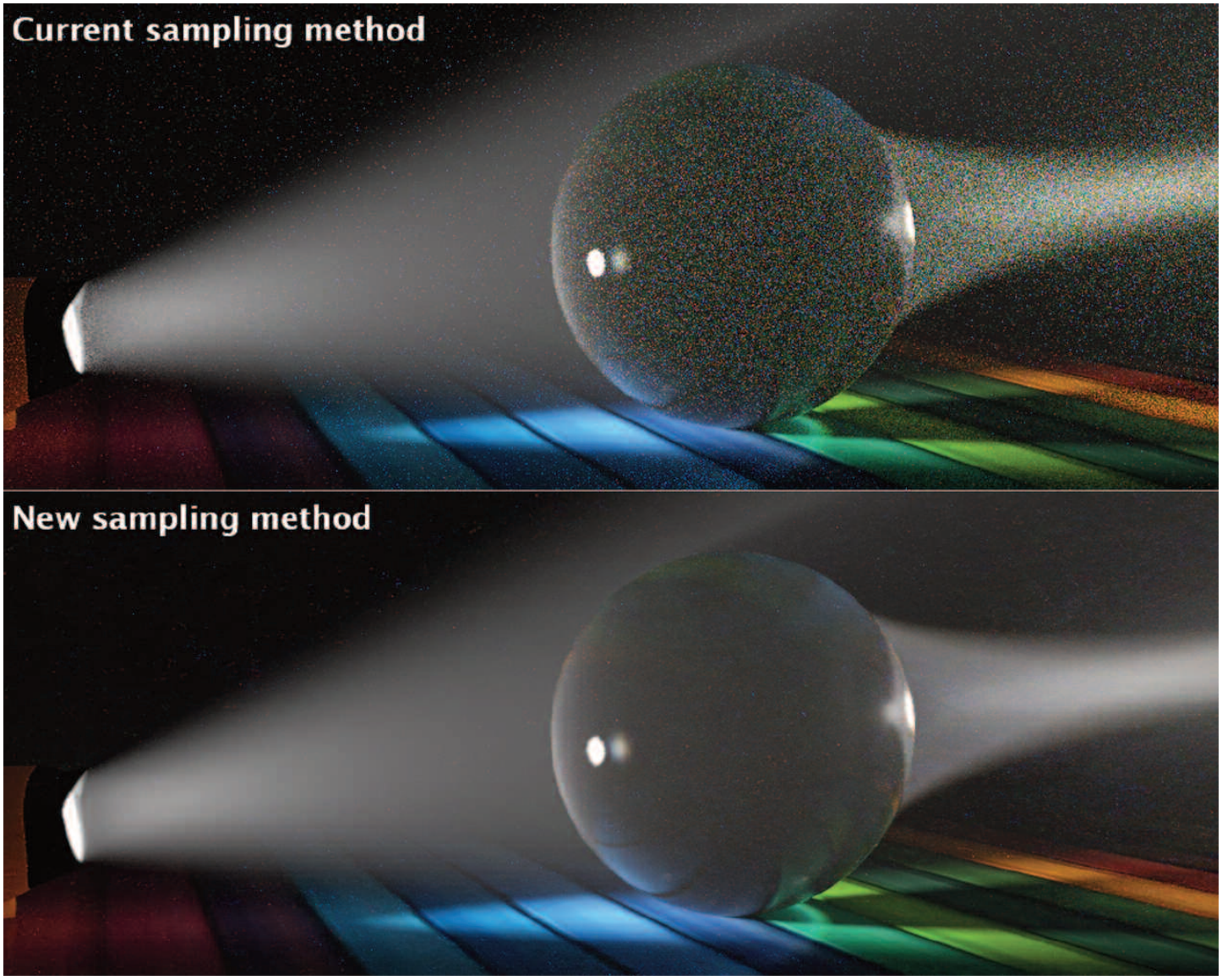
SU-Podium
"SU Podium will turn your SketchUp model into a photo-real image. SU Podium V2 employs advanced rendering technology but sticks to it's original intention of making photo-realistic rendering for SketchUp an easy and rewarding task."

Render[in]
"Render[in] 2's new Radiosity engine improves images for a better perception of colors, textures, and materials. Thanks to both the ISO and Shutter parameters, it is now easier than ever to fine-tune a scene's lighting."

iray
"Unlike current ray-tracing renderers, iray does not depend on complex renderer specific shaders and settings to approximate global illumination. iray achieves its high level of performance by taking full advantage of the CUDA programming model, allowing interactive previewing on both single and multiple NVIDIA GPU platforms. iray balances intuitive ease of use and scene setup with the highest quality photorealistic final frame output and interactive performance."

Kerkythea Render
"Caustics (based on Photon Maps or quick heuristics). Global Illumination support based on various methods (Path Tracing, Photon Maps/Mesh Maps + Irradiance Caching, Diffuse Interreflection). Is a standalone freeware renderer that can be used to create photorealistic renderings."

Twilight Render for SketchUp
Betaalde spinoff van Kerkythea:
"Supports physically accurate materials like fuzzy reflection and subsurface scattering, specialized render methods like Alpha Mask, and a wide range of fine tuned render presets. Also supports animation with almost no setup time, including third-party plugins like Proper Animation and SketchyPhysics!
Physically accurate materials, Fresnel reflections, Fuzzy Metal Reflections, SubSurface Scattering, Volumetrics and more. Render SketchUp models with biased and unbiased methods on an unlimited number of processors. Ray Tracing, Photon Mapping, Path Tracing, Bi-directional Path Tracing, Metropolis Light Transport."



Arion 2.0.4 vooruitblik.
Een lastig 3D model in een GPU gebaseerd render programma. Beide afbeeldingen zijn met dezelfde hardware gedaan en de rendertijd is gelijk. Arion is CUDA 4 georiënteerd.
http://www.randomcontrol.com/newsletter-2011-12-06

Zo herkennen we de hybride programma's die de keuze open laten voor:

- CPU renderen
- GPU CUDA renderen
- OpenCL renderen

Dit lijkt de ideale mix voor een zo groot mogelijk publiek, dankzij aanzienlijke extra software investeringen van de fabrikant. Maar daar krijgt de fabrikant wel voor terug dat zijn programma met allerlei soorten hardware kan worden gedraaid.

Render programma's en 'Unbiased'
"Een lage kwaliteit Photon Map is een kwestie van enkele seconden. Een gemiddelde kwaliteit Photon Map zal meerdere minuten duren. Maar het maken van een rendering in hoge kwaliteit zal dagen . . . kunnen duren."

Monte Carlo methode voor Global Illumination wordt vaak gekoppeld aan "unbiased". Dat betekent dat voor elk pixel in de uiteindelijke rendering er bijzonder veel semi-willekeurige virtuele lichtstralen in de 3d-scene worden gestraald. Dit zou een goede nauwkeurigheid met zich mee brengen, maar ook slakken-traag verlopen en **ruis** introduceren.

Dit is in wezen de hoofdmethode die door o.m. Maxwell, Arnold, LuxRender, Indigo Render, OctaneRender, Kerkythea en FPrime worden toegepast. Veel andere render programma's gebruiken dezelfde methodiek, maar met slimme aanpassingen in algoritmes. Het is dus niet voorbehouden (= marketingpraat) aan één enkel programma zoals we op websites kunnen lezen. FPrime geeft

op haar website aan dat het programma Windows en Mac compatibel is maar vergeet de versie nummers van de operating systemen te vermelden.

Renderen zonder dat Global Illumination aan staat, kan al in enkele minuten resultaat opleveren, maar met GI aangezet kan dat al gauw oplopen tot enkele uren (brut force). Eén van de oplossingen die men heeft gevonden is 'Irradiance Caching'. Niet elk pixel wordt daarin meegenomen in de berekeningen, bv. bij egale gelijkende oppervlakken. Evenals bij Photon Mapping is ook hier een tussenstap noodzakelijk (pre-pass), waardoor het risico van detailverlies en plotselinge storende vlekken of stippen groter wordt. Met de brute kracht van Global Illumination alleen, zien we **ruis** en dat varieert afhankelijk van hoe lang er wordt gerenderd. In veel gevallen ziet een lichte ruis er beter uit dan een met willekeurige spikkels uitgedoste rendering.

De ruis van Photon Mapping samen met de kans op spikkels met Irradiance caching GI en de lage rendersnelheid bracht "Final Gathering" in beeld. Allereerst wordt een Photon Map berekend (incl. spikkels) die alleen als basis opzet voor de 3D scene wordt gebruikt. Om vervolgens een 'langzamer' methode, maar wel nauwkeuriger toe te passen: 'Final Gathering'. Deze zorgt voor details en hopenlijk ook voor het wegwerken van evt. spikkels in de rendering.

Nadeel: **Photon Mapping** is al langzaam, deze methode met FG is nog langzamer, maar wel 'nauwkeuriger'. Daarbij worden de 'Shadow rays' voor een deel overgeslagen.

Final Gathering (bekend van o.m. Mental Ray en Turtle) wordt nogal eens verward met Irradiance Caching. In principe streven alle render programma's naar het zo goed mogelijk weergeven van een 3d model, de middelen en gebruikte algoritmes kunnen echter sterk uiteenlopen. Waardoor ook de kwaliteit en de rendersnelheid grote variaties kent.

Maxwell Render wordt vaak geassocieerd met "unbiased", miljoenen stralen worden berekend, die niet in de uiteindelijke rendering terugkomen voor de goede GI weergave. "Unbiased" is dat er geen vreemde oneigenlijke dingen in de rendering voorkomen (spikkels, licht lekken etc.), maar daarvoor in de plaats komt dan wel ruis. Met Irriance Map en Quasi Monte Carlo wordt dezelfde methode toegepast als bij Maxwell Render maar nu worden minder bijdragende pixels berekend, waardoor het aantal berekening kleiner kan zijn, met snelheidswinst voor de gebruiker.

Zelfs met 'biased' render programma's zijn goede renderingen te maken waarbij *meer Photonen gebruiken* niet gelijk staat aan een beter resultaat.

'Irriance Caching' en 'Final Gathering' zijn slechts enkele van de vele toe te passen methoden om tot snellere renderingstijden te komen met behoud van zo veel mogelijk nauwkeurigheid.

Op diverse plaatsen in deze PDF hebben we het gehad over termen (buzz-words), die door Render programma's worden gebruikt om aan te duiden dat alléén hun programma aan al deze speciale opties voldoet.

Daarvoor worden zelfs nieuwe woorden gebruikt om de klant te verleiden dat specifieke en unieke programma aan te schaffen.

In de praktijk zijn vrijwel alle programma's afgeleid van enkele basisprincipes die door Appel, Turner Whitted, Jensen, Glassner en vele anderen decenia geleden zijn uitgedacht.

S. Metzger (CG supervisor) grapt er op los door te zeggen dat er niet over renderingen kan worden gesproken zonder dat iemand of een groep zich daar over opwind: "Renderingen maken is een soort religie aan het worden met een enorm aanbod van renderingsprogramma's en heel veel nieuwe ontwikkelingen".

Mike Farnsworth (RenderSpud) denkt over het gebruik van kreten voor renderingsprogramma's het volgende:

"We start hearing marketing people, and then subsequently users, talk about a particular technique or new technology as if it were just the greatest thing since sliced bread.

They throw the terms associated with it around and make it sound really important. Sometimes it really is important, but most of the time we geeks just laugh about all the hype and talk."

En om het ingewikkelder te maken geeft Mike Farnsworh op een vraag uit 2007 over Photon Mapping het volgende antwoord:

"I've thought about that, and I think you are right. The bias is introduced in photon mapping when you are sampling already-stored photons, and the stored photons are what you've got (no chance to get other sample locations)."

"I'll update the post, because while in the limit you could theoretically get to the same image (photon mapping does cover all light paths when done right, after all), that doesn't mean it is unbiased statistically."

In de literatuur (zie PDF bestanden) zijn diverse inzendingen, die speciaal over dit onderwerp gaan.

OTOY (Nieuw Zeeland) heeft veel divisies met een grote klantenkring en diversiteit op het gebied van renderen. Octane Render maakt daar inmiddels ook deel van uit waarbij de focus op GPU renderen is gericht. Maar ook Cloud renderen is met Octane mogelijk. Daarnaast hebben ze een plug-in voor 3ds Max, Maya, Poser uitgebracht. OTOY werkt voor Autodesk (en LightWave, Cinema 4D, AutoCAD, Poser en Daz Studio) om Cloud render tools te ontwikkelen, waarbij Brian Matthews (vice pres. Autodesk Labs) vertelde dat de mogelijkheden van de OTOY's software een breed aantal gebruikers en de industrie aanspreken. Venturebeat.com vertelt dat Autodesk echter niet op één paard wed en ook investeert in OnLive (streaming-gaming bedrijf), een belangrijke concurrent van OTOY, maar OnLive is wel in financiële problemen. In oktober 2012 werd deze voor een fractie overgedaan aan een venture capitalist. Zie de film met twee GTX 580 en de nieuwe Brigade Game Engine van OTOY. Sneller dan V-RAY met z'n RT. De verlichting is een combinatie van omgevings- en direct licht. Het model wordt door de lucht en de zon verlicht.
http://youtu.be/Ig6FoTezKi8

Vervolg op deze "Render principes" PDF:

programma codes

Extra:
Diagram computer en grafische kaart keuze

literatuur opgave met originele pdf's

Extra:
Geschiedenis Grafische ontwikkeling

NVIDIA LIJN

OptiX interactieve Ray Tracing Engine voor elk programma om te integreren?

http://www.nvidia.com/object/optix.html

Werkstations

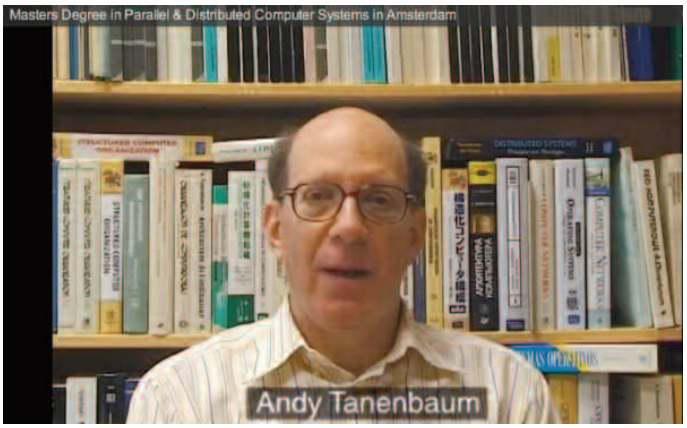
Andrew S. Tanenbaum

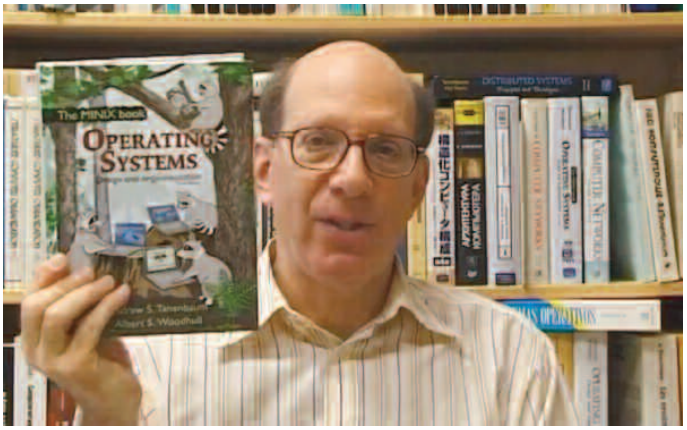
Wat heeft A. Tanenbaum (professor aan de VU Amsterdam) met renderen te maken? Weinig en toch alles. Deze render uitgave is geschreven met als inspiratiebron Tanenbaum. Mede door de lezingen en boeken van Andrew Tanenbaum.

Als we teruggaan in de tijd dan komen we in de ontwikkeling van computer operating systemen **Linux** tegen. En Andrew Tanenbaum is in mijn beleving de grondlegger van het eerste Linux operatingsysteem. Linus Torvalds uit Finland (sinds 2008 woonachtig in Amerika) krijgt meestal de eer toebedeeld, dat lijkt mij gezien de geschiedenis onjuist. Wel heeft Linus de aanzet voor de Open Source gedachte gegeven verder uitgewerkt door Richard Stallman.

Geschiedenis
De voorganger UNIX V7 werd afgeschermd en was niet voor educatieve doeleinden beschikbaar. AT&T corporation maakte het commercieel.

Professor Tanenbaum starte in 1984 met een compleet nieuwe UNIX clone, **Minix** genaamd. In 1987 kwam MINIX uit en het was voornamelijk bedoeld als een leer operatingsysteem voor studenten. Het was geheel van de grond af aan opnieuw ontwikkeld en geschreven. In 1997 kwam **MINIX 2** uit samen met de tweede druk van het boek en in 2004 MINIX 3. In 2006 verscheen de derde editie van het MINIX boek. In 2008 doneerde de European Union de ERC Advanced Grant van $ 3,5 miljoen voor de ontwikkeling van een BETROUWBAAR operating systeem Minix 3. Dat was des te meer een teken dat de Europese gemeenschap bijzonder veel vertrouwen had in de kennis van hoogleraar Tanenbaum.
U kunt alles over **MINIX 3** vinden op deze site http://www.minix3.org
Hij heeft als geen ander een eerste aanzet gegeven voor een KLEIN operating systeem dat ook BETROUWBAAR moest zijn.
Klein betekend dat het razend snel is opgestart. En dat er weinig bugs inzitten. Volgens professor Andrew Tananbaum zitten in elke





1000 regels programma code 3 bugs. Naarmate het aantal code regels toeneemt, is het aannemelijk dat er veel meer bugs in voorkomen. Betrouwbaar betekend dat het een aantal problemen in diverse processen zo goed mogelijk zonder interventie zelf kan oplossen. Maar ook dat er een perfecte afscherming is. Bij betrouwbaar hoort ook dat veel processen buiten de kernel worden gehouden, waardoor deze beter beschermd is en minder afhankelijk van “de eerste de beste Taiwaneese programmeur die de code van de driver niet helemaal op tijd kon afronden en een aantal steken had laten vallen” (verhaal van Andrew Tanenbaum tijdens Fosdem 2010).

Professor Tanenbaum tijdens de presentatie van zijn vakgebied in Amsterdam:
http://www.youtube.com/watch?v=7zAruZ_jFzY&feature=youtu.be

Tijdens de Fosdem lezing:
http://www.youtube.com/watch?feature=endscreen&v=bx3KuE7UjGA&NR=1

De nieuwe opleidingen parallele processen en het CUDA NVIDIA Teaching Center in Amsterdam vallen onder prof. A. Tanenbaum zijn afdeling.

Wat heeft de wis- en natuurkundige Gauss met renderen te maken?

Op het eerste gezicht weinig, Carl Friedrich Gauss (1777-1855) ontdekte een methode van de 'least squares' om de banen van planeten te berekenen. In 1809 publiceerde hij zijn techniek in het Latijn:
"Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium"
In het Engels: *"Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections"*.
Hij vertelde erbij dat hij deze methode al sinds 1795 gebruikte en dat hij ook een theorie had die bekend staat onder de naam:

The normal law of error
"... the most probable system of values of the quantities ... will be that in which the sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision, is a minimum." (Gauss 1809).

De meest interessante interpretatie kwam van:
Following Wells and Krakiwsky (1971, pp.8-9), it is interesting to analyse the following quotation from Gauss' Theoria Motus (Gauss, 1809, p.249).

"If the astronomical observations and other quantities, on which the computation of orbits is based, were absolutely correct, the elements also, whether deduced from three or four observations, would be strictly accurate (so far indeed as the motion is supposed to take place exactly according to the laws of KEPLER), and, therefore, if other observations were used, they might be confirmed, but not corrected. But since all our measurements and observations are nothing more than approximations to the truth, the same must be true of all calculations resting upon them, and the highest aim of all computations made concerning concrete phenomena must be to approximate, as nearly as practicable, to the truth. But this can be accomplished in no other way than by a suitable combination of more observations than the number absolutely requisite for the determination of the unknown quantities. This problem can only be properly undertaken when an approximate knowledge of the orbit has been already attained, which is afterwards to be corrected so as to satisfy all the observations in the most accurate manner possible."

Deze tekst is meer dan 200 jaar geleden geschreven, maar nog steeds relevant en bijzonder actueel.

1. Wiskundige modellen kunnen niet compleet zijn.

2. Natuurkundige metingen zijn inconsequent.

3. Alle uitkomsten die kunnen worden bereikt uit berekeningen die gebaseerd zijn op inconsequente metingen zijn inschattingen of waarderingen van de "werkelijkheid" of zo u wilt "waarheid".

4. Oorspronkelijke benaderingen van de uiteindelijke inschattingen / waarderingen moeten worden gebruikt en tenslotte:

5. **Oorspronkelijke benaderingen behoren gecorrigeerd te worden op een manier die de inconsistenties tussen de meetgegevens (waarmee Gauss zijn kleinste kwadraten-methode bedoelde) zo minimaal mogelijk maakt.**

En juist deze vijfde regel gaat voor het renderen in zijn geheel op.

Het gaat er in het geheel niet om, dat een render programma een one-liner meekrijgt zoals de voorbeelden op o.m. pag 93 en 94.

Pysically accurate
Pure physics algorithms"
Hybrid-accelerated and physically-based production render engine
Unbiased, physically based
Physically based GPU renderer
Most physically accurate rendering engine on the market
Speed and Physical Accuracy
Physically Accurate Lighting
Physically based ray tracing and unbiased program
Physically correct modelling of glass
Photo realistic renderer, unbiased and physically correct
Image based lighting or our physical sky
Highest quality photorealistic final frame output
Supports physically accurate materials

Het gaat er om dat een renderprogramma routines (algoritmes) heeft ontwikkeld en toepast, die de verschillen en meer of minder grote fouten zo veel mogelijk minimaliseerd. 100% Natuur- en wiskundig betrouwbare programma's bestaan eenvoudig niet, omdat er altijd compromissen moeten worden gesloten tussen:

snelheid
beschikbare hardware
kwaliteit
verkoopprijs
oplossen van problemen
neveneffecten
interface

Het renderprogramma (fabrikant) die dat niet doet zou in theorie een 'echt natuurlijke' rendering kunnen afleveren, waar niemand enige waardering voor kan opbrengen omdat aan bovenstaande zeven punten niet voldoende werd voldaan.

Het gaat al mis bij het maken van een 3D model, het is een interpretatie van de maker met de mogelijkheden die een 3D modeller biedt. Het is een interpretatie van de werkelijkheid of van een schijnwerkelijkheid bij het maken van een virtueel ontwerp. Deze interpretatie is eigenlijk nog heel concreet en betrouwbaar . . .

Rendering zonder naamsvermelding in 35 minuten en 45 seconden. Dit is de 'simpele' rendering.

*Rendering zonder naam in 12 uur 10 minuten en 30 seconden. Dit is de '**physically accurate**' rendering, oordeel zelf. Let op de flessen en de textuur van de tafelranden en eronder. En op de verschillen in licht en schaduw interpretatie.*
3D model is identiek.

Het renderproces is vele malen waziger en neemt afstand van alle concrete 3D vectoren en polygonen door een eigen interpretatie te geven van licht, schaduw, textuur, reflectie, absorbtie, glasbreking, caustic en nog tientallen andere zaken.

'Renderen is de kunst om compromissen zoveel mogelijk te verdoezelen'

Renderfabrikanten zouden zich meer op hun eigen sterkte moeten concentreren en voor de diverse doelgroepen die men wil aansturen een zo goed mogelijke interface en programma moeten ontwikkelen.
Hetgeen als een verrijking kan worden gezien in de *overvolle rendermarkt*.