Computer Graphics Project
Visuele Effecten

Andreas Van Barel

23 mei 2015

1 De raytracer

De geïmlementeerde onderdelen van de raytracer zijn als volgt. De analytische vormen waarmee de raytracer overweg kan, zijn bollen, balken, cilinders, vlakken en driehoeken. Er kunnen ook triangle meshes ingeladen worden uit .obj files. Deze laatsten kunnen gebruik maken van een normal map. Een bounding volume hierarchie kan gegenereerd worden om het ray-tracen van de triangle meshes sneller te laten verlopen. Een object in een scene bestaat uit de combinatie van een shape met een material. De materialen kunnen verschillende brdf modellen hebben, ze kunnen getextureerd zijn, spiegelend, refractief of glossy. Voor de lichtbronnen is er keuze uit directionele lichtbronnen, puntlichtbronnen en oppervlaktelichtbronnen. Verder zijn er nog anti-aliasing, reflection maps (per object afzonderlijk en/of voor de hele scene) en depth of field.

2 Brdf-modellen

2.1 Phong en glossy variant

We vergelijken eerst het Phong-brdf model en een glossy variant daarvan. De diffuse belichting gebeurt bij al de modellen besproken in deze paper op dezelfde manier:

$$L_d = k_d c_{light} L_{light} c_d \cos(\theta)$$

L_d	Uitgaande radiantie vanwege diffuse component
k_d	Diffuse coëfficiënt
c_{light}	Kleur van de lichtbron
L_{light}	Inkomende radiantie uit de lichtbron
c_d	kleur voor de diffuse belichting
θ	hoek tussen normaal en richting van uitgaande radiantie

De speculaire shading gebeurt bij het Phong-brdf model volgens

$$L_s = k_s c_{light} L_{light} c_s \cos^e(\alpha) \tag{1}$$

L_s	Uitgaande radiantie vanwege speculaire component
k_s	Speculaire coëfficiënt
c_s	kleur voor de speculaire belichting
α	hoek tussen richting van de inkomende radiantie en de perfect speculaire richting

Bij de glossy variant gebeurt de diffuse belichting op dezelfde niet-recursieve manier. De speculaire belichting wordt berekend door stralen te schieten rondom de richting van perfect speculaire reflectie volgens een distributie d (samples per ruimtehoek) die evenredig is met

$$d \sim \cos^e(\alpha)$$

Volgens sectie 7.2 uit het handboek kan men zulke distributie bekomen door twee random getallen $(r_1, r_2) \in [0, 1]^2$ te mappen volgens

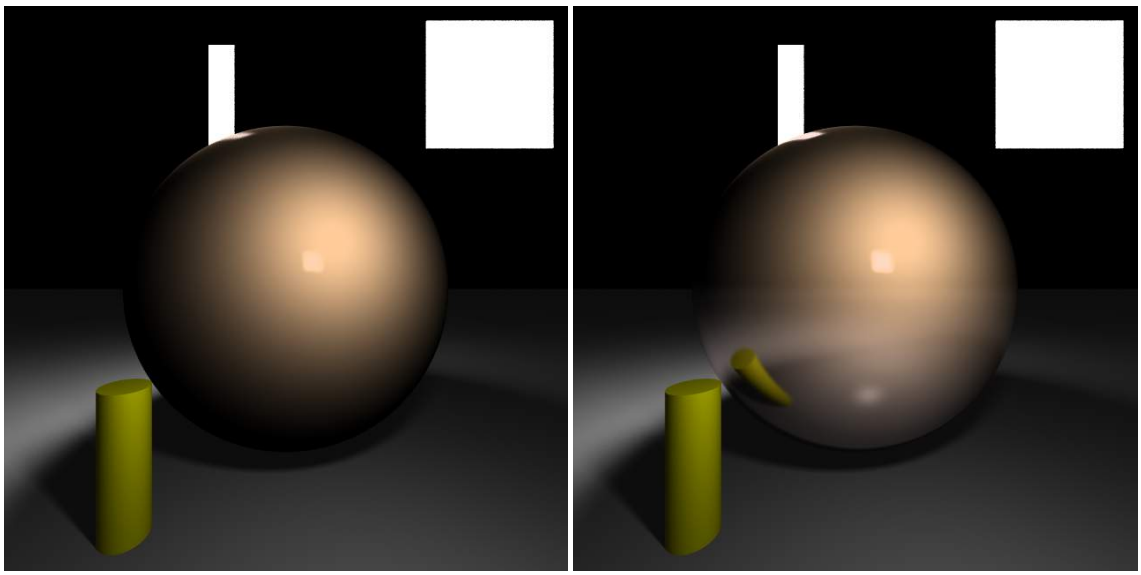
$$\phi = 2\pi r_1$$
$$\theta = \arccos((1 - r_2)^{1/(e+1)})$$

Hierbij is ϕ de hoek met de vector in de richting van de perfect speculaire reflectie, en θ de hoek rondom deze richting. M.a.w. (ϕ, θ) zijn poolcoördinaten in een assenstelsel waarin de azimuth ligt volgens de perfect speculaire reflectie. Wanneer men zulke samples genereert is het mogelijk dat deze terugvallen op het oppervlak vanwaar zij zouden moeten vertrekken. Deze slechte samples worden simpelweg niet gebruikt. Voor elk van de samples die wel gebruikt worden wordt de shading berekend via de speculaire Phong shading van vergelijking 1.

Beide modellen gebruiken de volgende concrete parameters:

$$c_d = (1, 0.8, 0.6)$$
$$k_s = 0.2$$
$$c_s = (1, 0.9, 0.9)$$
$$k_s = 0.8$$
$$e = 1000$$

Het resultaat vindt men in figuur 1. Beiden werden gerenderd met 50x anti-aliasing. De Phong



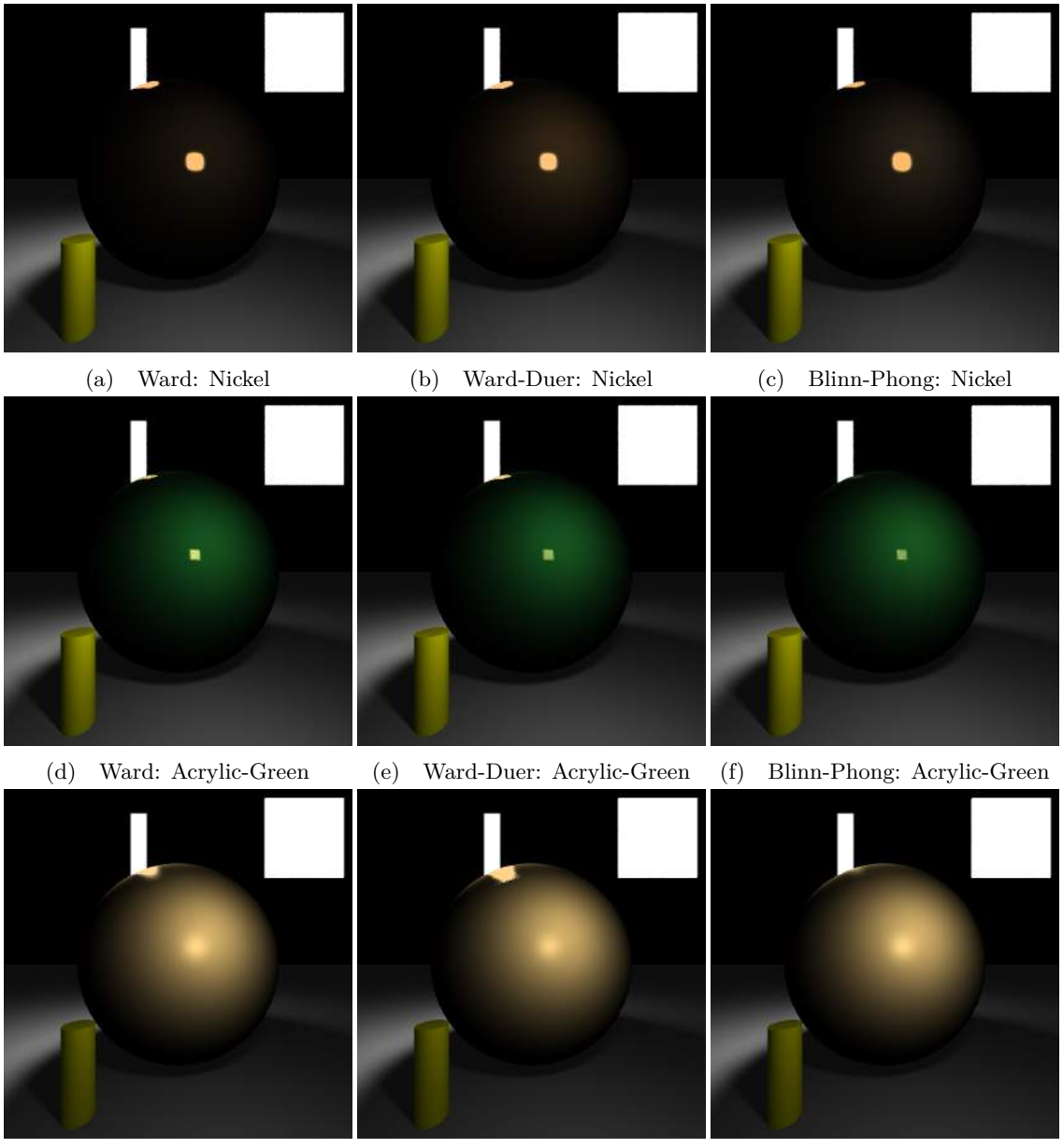
(a) Bol met Phong speculaire reflecties (b) Bol met glossy Phong speculaire reflecties.

Figuur 1: Effect van glossiness op de speculaire reflectie.

highlights werden gerenderd met 25 samples per oppervlaktelichtbron. De glossy variant werd gerenderd met 10 glossy samples per intersectie van de bol. De oppervlaktelichtbronnen werden in dit geval niet gesampled. Hun bijdrage wordt impliciet in rekening gebracht via de glossy samples. De glossy speculaire reflecties zijn duidelijk visueel veel aantrekkelijker. Niet alleen de lichtbronnen maar de hele scene reflecteert in de glossy bol. Een cilinder werd speciaal neergezet om dit zeker duidelijk te maken. Het nadeel is natuurlijk dat de glossy reflecties ongeveer 10 keer langer duren om uit te rekenen. Interessant is ook de schijn aan de rand van de bol die de oppervlaktelichtbron achter de bol produceert. Dit is vooral het resultaat van de diffuse belichting veroorzaakt door de lichtbron.

2.2 Gefitte brdf modellen

Drie brdf modellen uit de paper werden geïmplementeerd: het Ward model, het Ward-Duer model en het Blinn-Phong model. In elk van de drie gebeurt de diffuse belichting nog steeds op dezelfde manier. Ze werden vergeleken voor de gefitte materialen Nickel, Acrylic-Green en Derlin. Voor



(a) Ward: Nickel (b) Ward-Duer: Nickel (c) Blinn-Phong: Nickel

(d) Ward: Acrylic-Green (e) Ward-Duer: Acrylic-Green (f) Blinn-Phong: Acrylic-Green

(g) Ward: Derlin (h) Ward-Duer: Derlin (i) Blinn-Phong: Derlin

Figuur 2: Vergelijking van de gefitte materiaalmodellen.

Derlin werd de voorste lichtbron wat afgezwakt om kleursaturatie te voorkomen. Kleursaturatie treedt eigenlijk ook op bij Nickel, maar vermits de rest al zo donker is hebben we daar de lichtbronnen niet afgezwakt. De grootste verschillen komen voor wanneer de zichtstraal invalt onder een hoek dicht bij 90°. De meeste fysische materialen worden reflectiever in dit geval. In het Blinn-Phong model (en trouwens ook niet in het gewone Phong-model) worden deze effecten niet in rekening gebracht. Bij Ward en Ward-Duer zijn de resultaten beter.

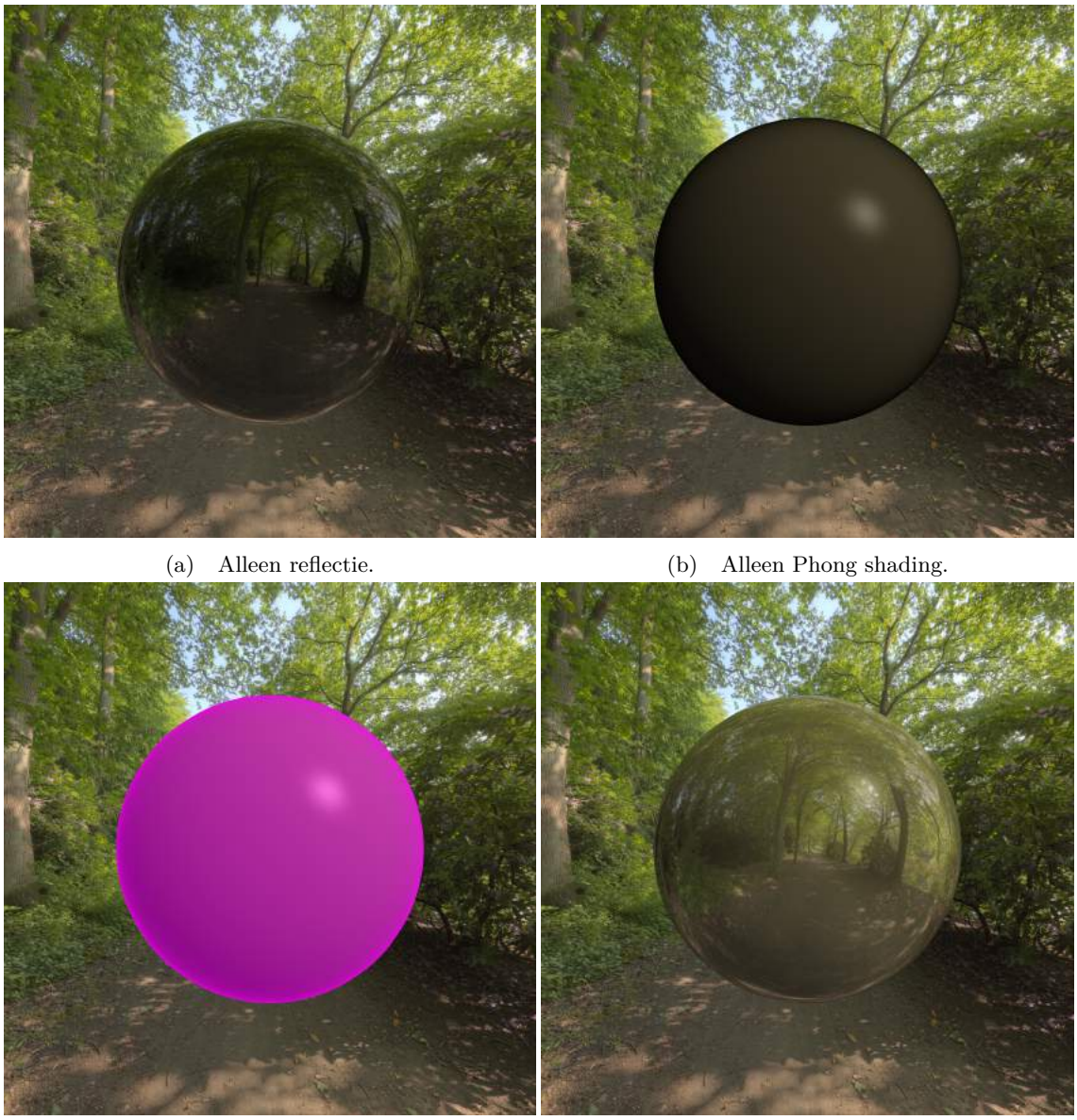
3 Fysisch correcte reflectie en refractie

We gebruiken eerst een environment map en een bol om deze effecten te bespreken. We kijken eerst naar een reflecterende bol met reflectiecoëfficiënt 0.5 en een bruine kleur. Het resultaat is te zien in figuur 3.

Figuur 3: Bol met reflectiecoëfficiënt 0.5.

De individuele termen in de shading zijn weergegeven in figuur 4. De reflectie is weergegeven in figuur 4a en de gewone Phong shading component in 4b. Het is duidelijk dat de reflectie belangrijker is wanneer de straal zeer schuin invalt. Dit kan beter gezien worden op figuur 4c waar de reflectie berekend is alsof ze gehaald zou zijn uit een volledig magenta environment map. Als deze hoekafhankelijkheid genegeerd wordt, en er dus altijd 50% reflectie is, is het resultaat zoals in figuur 6d. Het verschil is niet zo groot, maar toch voelt het origineel realistischer aan.

We bekijken nu een refracterende bol. De bol heeft refractieve index 1.5 (dus ongeveer een glazen bol), en de kleur is perfect wit, wat betekent dat er geen vermindering van de radiantie is wanneer een straal doorheen de bol loopt. Dit zodat de verschillende delen in de shading duidelijker zijn. Figuur 5 toont het resultaat.



(a) Alleen reflectie.

(b) Alleen Phong shading.

(c) Reflectie in magenta.

(d) Hoekonafhankelijke reflectie.

Figuur 4: Analyse van de refractietermen.

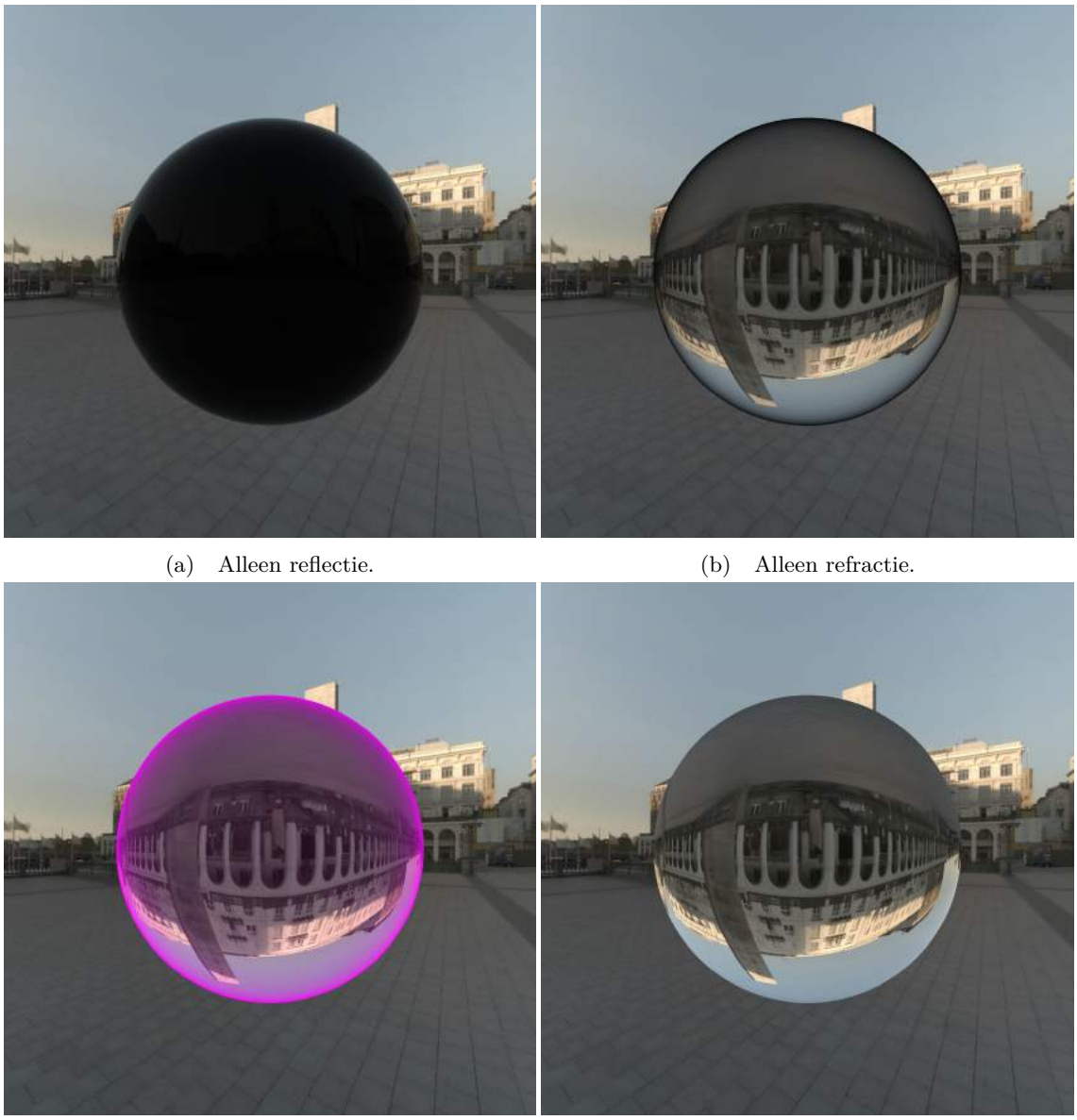
We bekijken nu de individuele termen in de shading. De reflectie is weergegeven in figuur 6a en de refractieve component in 6b. Voor de duidelijkheid is in figuur 6c de reflectie weer berekend alsof ze gehaald zou zijn uit een volledig magenta environment map. Het is duidelijk dat de reflectie vooral belangrijk is wanneer de straal zeer schuin invalt. Als de hoekafhankelijkheid genegeerd wordt, en er dus altijd 100% refractie is en geen reflectie is het resultaat zoals in figuur 6d. Dit is duidelijk minder realistisch dan het origineel.

4 Normal Mapping

Normal mapping wordt gerealiseerd door de normaal in het intersectiepunt aan te passen. Dit kan gebeuren door bij de normaal in een goed gedefinieerd assenstelsel waardes op te tellen die

Figuur 5: Bol met refractieve index 1.5.

afhangen van het intersectiepunt. In dit geval wordt de normaal simpelweg volledig overschreven door een normaalvector geëncodeerd in de RGB componenten van de normal map. Hier is normal mapping slechts geïmplementeerd voor triangle meshes. De lookup in de normal map gebeurt met de gewoonlijke UV-coördinaten verkregen uit de intersectie met de mesh. Figuur 7 toont het effect van een normal map op de shading van een appel. De normal map geeft duidelijk de illusie van veel detail tegen een zeer beperkte rekenkost. Merk op dat het silhouet van het model (en dus ook de shaduw) niet verandert. Bekijk ook de shaduwkant van de appel. De overgang van belichte naar shaduwkant verandert ook niet en is duidelijker in het model met normal map. Dit doordat de normal map zelfs dicht tegen de rand de normaal zo kan perturberen dat de shadingberekening veel licht teruggeeft. Voorbij de rand kan dit in de huidige implementatie niet. Alleen de normaal verandert, en bij het trekken van een shaduwstraal wordt de appel aan de schaduwkant altijd geïntersecteerd waardoor er geen licht kan zijn. Dit ook al zou de normaal in dat punt zo geperturbeerd zijn dat de hoek met de straal naar de lichtbron kleiner is dan 90°. Dit zou opgelost kunnen worden door uitzonderingen toe te voegen bij het intersecteren van de shaduwstralen. Zo zou een intersectiepunt dat zeer dicht bij de oorsprong van de schaduwstraal



(a) Alleen reflectie.

(b) Alleen refractie.

(c) Reflectie in magenta.

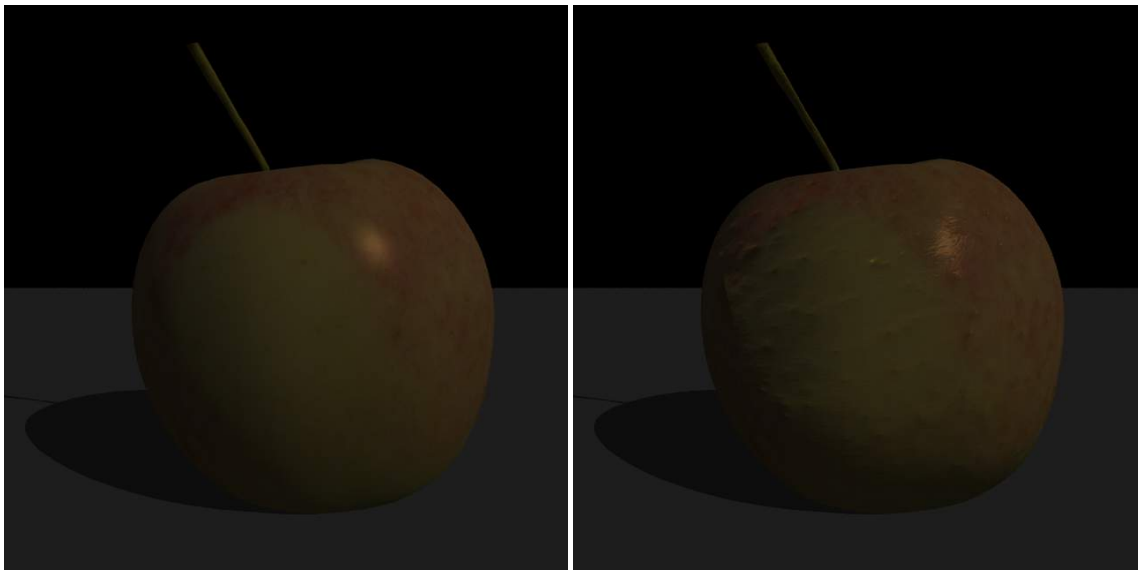
(d) Hoekonafhankelijke refractie.

Figuur 6: Analyse van de refractietermen.

gelegen is (en overeenkomstig intersectiepunt wanneer de straal de appel weer verlaat) bvb niet meer meetellen wanneer de hoek tussen normaal en schaduwstraal kleiner is dan 90°.

5 Depth of Field

Depth of field werd op de standaard manier geïmplementeerd. Enkele resultaten zijn te zien in figuur 8. Merk op dat de reflecties in de bol niet per se ook in focus zijn als de bol zelf in focus is.



(a) Appel zonder normal map. (b) Appel met normal map.

Figuur 7: Effect van een normal map op een appel.

(a) DOF: Close. (b) DOF: middle. (c) DOF: far.

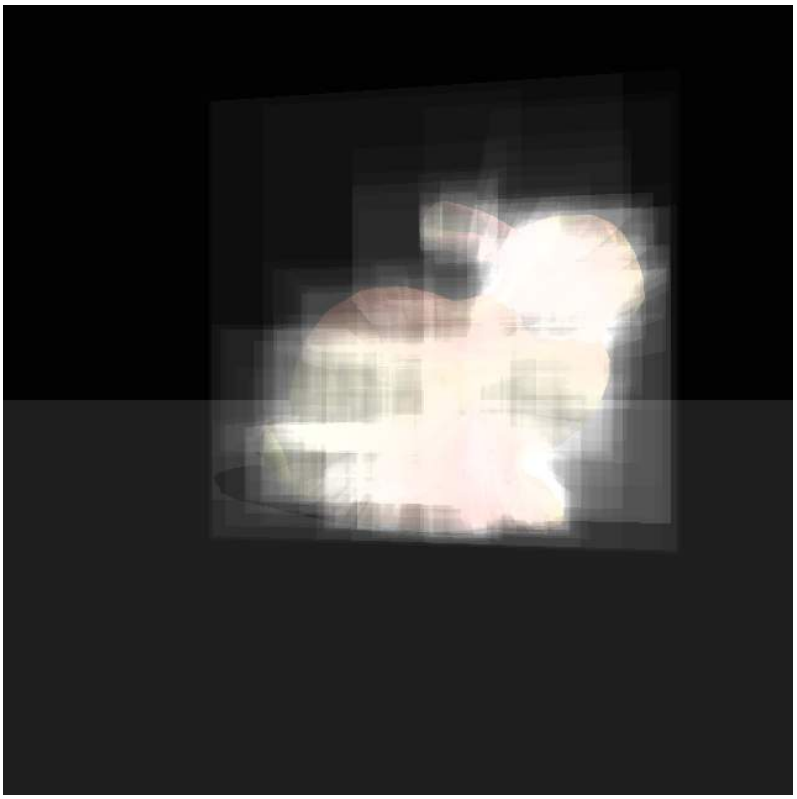
Figuur 8: Depth of Field.

6 Performantie

Er werd ook wat aandacht geschonken aan het versnellen van de raytracer.

Het inlezen van objectfiles is versneld door regex expressies te gebruiken. De draak kan in enkele seconden ingelezen worden.

De bounding volume hierarchie wordt opgesteld door telkens te splitsen volgens de langste as. Alle objecten in het parent bounding volume worden gesorteerd volgens deze as op hun middelpunt (het zwaartepunt voor driehoeken). De twee child bounding volumes worden dan opgesteld als het bounding volume dat de eerste helft van alle objecten omvat en het bounding volume dat de tweede helft omvat. Beide volumes hebben dus evenveel elementen (op één element na als het aantal elementen in de parent oneven was). Deze bounding volumes kunnen overlappen. Dit gaf de beste resultaten. Het volledige dragon model kan in een fractie van een seconde ontbonden worden in deze bounding volume hierarchie en kan in zo'n 3 seconden geraytraced worden met schaduwen en twee puntlichtbronnen. Dit gebeurde op een i5 4690K CPU. Een false color image van de bunny met deze bounding volume hierarchie is getoond in figuur 9



Figuur 9: False color image van de Stanford bunny. Hoe witter, hoe meer bounding volume intersecties het kostte om de overeenkomstige pixel te renderen.

De raytracer gebruikt een klasse ShadeRec om resultaten (hitpoint, normal, UV, ...) uit het tracen van een ray door te geven. Echter het aanmaken en verwijderen van een groot aantal objecten veroorzaakt heel wat overhead. Daarom worden ShadeRecs zoveel mogelijk herbruikt. De methodes die de heavy lifting moeten doen geven niets terug en hebben een ShadeRec als argument. De waarden van die ShadeRec worden dan overschreven. Dit in tegenstelling tot het teruggeven van ShadeRec's van methode naar methode. Daardoor kan vaak dezelfde ShadeRec herbruikt worden tussen de oproepen naar deze functies en wordt de creatie van nieuwe objecten tot een minimum teruggebracht.

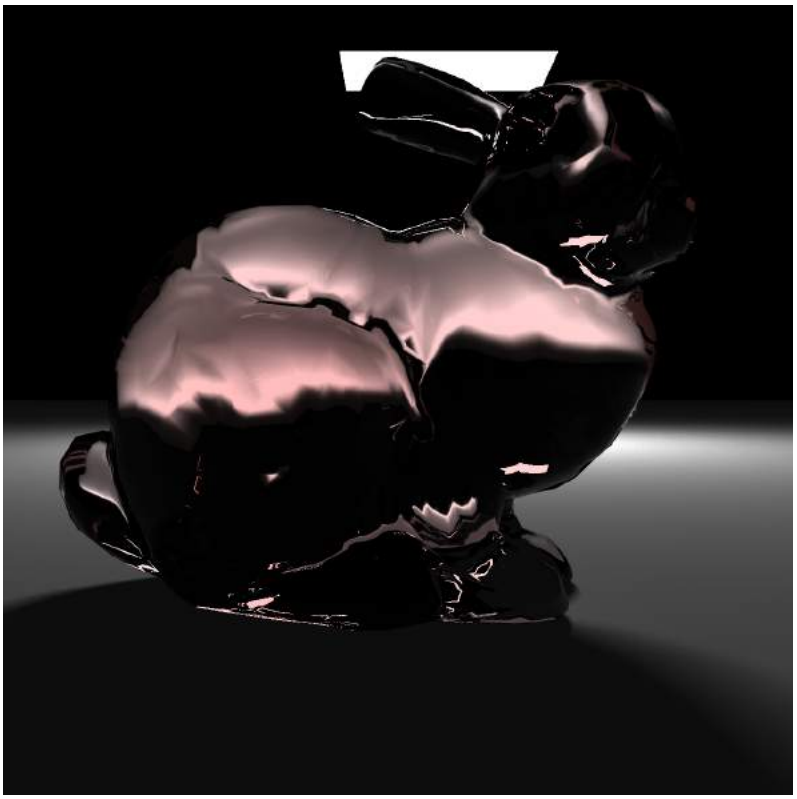
7 Andere gerenderde figuurtjes

7.1 Glazen bunny met oppervlaktelichtbron

Figuur 10 toont een glazen bunny met refractieCoëfficiënt 1.5 en met een groene kleur. Merk op dat er geen caustics zijn (hiervoor is globale belichting nodig). Deze figuur toont ook Beer's Law: Naarmate de straal langer doorheen de bunny heeft gereisd is zijn kleur roder.

7.2 Fractal in de reflectie

Beschouw figuur 11 gerenderd met deze raytracer. De fractal is afkomstig van vier reflecterende bollen die elkaar net raken. Zij liggen dus op de hoekpunten van een regelmatige tetraëder. De camera kijkt vanonder tussen de drie onderste bollen en recht naar de vierde bovenste bol. In plaats van de correcte fysische reflectie waarin de kleur bij elke reflectie afzwakt is hier de kleur een klein beetje sterker gemaakt na elke reflectie. Bovendien, als de straal een bol raakt wordt



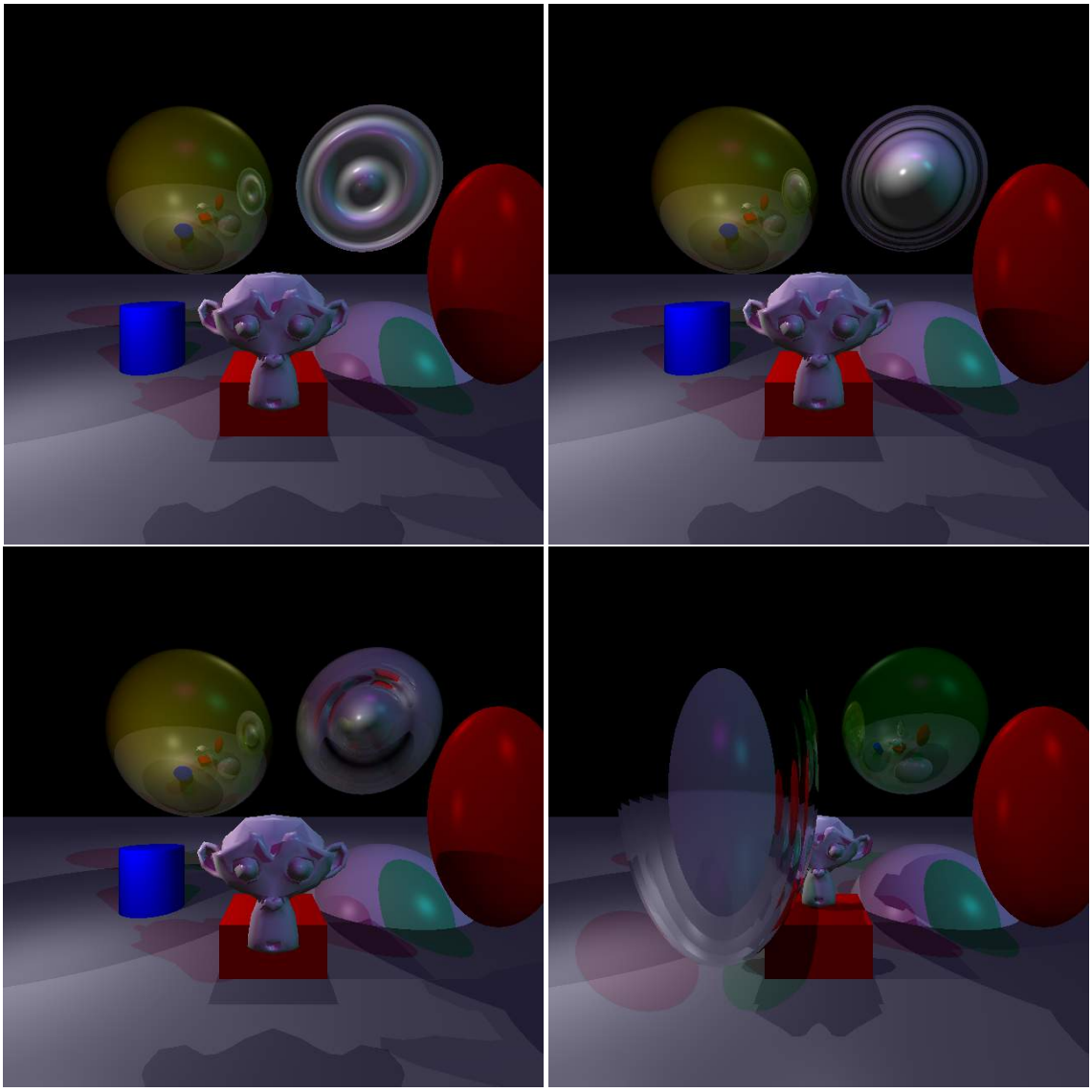
Figuur 10: Glazen bunny.

er voor de bovenste (op het prentje) wat groen toegevoegd, voor de linkse wat rood en voor de rechtse wat blauw. Zulke prentjes kunnen gebruikt worden om te kijken hoeveel keer een straal reflecteerd vooraleer zij divergeert vantussen de bollen. Blijkbaar reflecteert een straal veel keer voor punten gelegen op een sierpinski-driehoekachtige vormen. Aan de kleur kan gezien worden met welke bollen deze reflecties dan wel plaatsvinden. Zo betekent fel geel dat er veel reflecties waren met de groene en de rode bol bijvoorbeeld. De achterste bol voegt geen kleur toe als er met deze gereflecteerd wordt.

7.3 Bugs bij het implementeren van refractie

Door enkele bugs bij het implementeren van correcte refractie ontstonden volgende prentjes:

Figuur 11: Fractaal die een maat geeft voor het aantal reflecties tot divergentie uit een structuur bestaande uit vier rakende bollen.



Figuur 12: Bugs bij het implementeren van refractie.