KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Distributiefunctie van de geometrische normalen gebruiken voor het bouwen van BSP acceleratiedatastructuren

Jesse Hoobergs

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie Mens-machine communicatie

Promotor:
Prof. dr. ir. Philip Dutré

Assessoren:
Ir. W. Eetveel
W. Eetrest

Begeleiders:
Ir. M. Moulin
Ir. P. Bartels

Academiejaar 2018 – 2019

Voorwoord

Dit is mijn dankwoord om iedereen te danken die mij bezig gehouden heeft. Hierbij dank ik mijn promotor, mijn begeleider en de voltallige jury. Ook mijn familie heeft mij erg gesteund natuurlijk.

Jesse Hoobergs

Inhoudsopgave

Samenvatting

In dit `abstract` environment wordt een al dan niet uitgebreide samenvatting van het werk gegeven. De bedoeling is wel dat dit tot 1 bladzijde beperkt blijft.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Lijst van figuren en tabellen

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

BSP	Binary Space Partitioning
BVH	Bounding Volume Hierarchy
$k - DOP$	Discrete Oriented Polytope
$RBSP$	Restricted Binary Space Partitioning Tree
$\mathcal{SA}$	Surface Area

Symbolen

BSP_{wn}	BSP boom die per knoop random normalen als splitsrichtingen gebruikt.
BSP_{wn+}	BSP_{wn} boom waarbij de drie richtingen volgens de hoofdassen altijd deel uitmaken van de splitsrichtingen.
BSP^{Kd}_{wn+}	BSP_{wn+} boom waarbij Kd knopen efficiënter doorkruist worden dan BSP knopen.
BSP_{wn}	BSP boom die per knoop een clustering van de normalen berekend en de centrums van deze clusters als splitsrichtingen gebruikt.
BSP_{wn+}	BSP_{wn} boom waarbij de drie richtingen volgens de hoofdassen altijd deel uitmaken van de splitsrichtingen.
BSP^{Kd}_{wn+}	BSP_{wn+} boom waarbij Kd knopen efficiënter doorkruist worden dan BSP knopen.
BSP_{random}	BSP boom die per knoop random richtingen als splitsrichtingen gebruikt.
$BSP_{random+}$	BSP_{random} boom waarbij de drie richtingen volgens de hoofdassen altijd deel uitmaken van de random splitsrichtingen.
$BSP^{Kd}_{random+}$	$BSP_{random+}$ boom waarbij Kd knopen efficiënter doorkruist worden dan BSP knopen.
$\mathcal{K}_{d,BSP}$	De kost om een BSP knoop te doorkruisen.
$\mathcal{K}_{d,Kd}$	De kost om een Kd knoop te doorkruisen.
$RBSP^{Kd}$	$RBSP$ waarbij Kd knopen efficiënter doorkruist worden dan BSP knopen.

Hoofdstuk 1

Inleiding

In dit hoofdstuk wordt het werk ingeleid. Het doel wordt gedefinieerd en er wordt uitgelegd wat de te volgen weg is (beter bekend als de rode draad).

Als je niet goed weet wat een masterproef is, kan je altijd Wikipedia eens nakijken.

1.1 Ray tracing

Stralen volgen Door elke pixel één (of meerdere stralen), kleur intersectiepunt is kleur pixel -> path tracing.

Acceleratiestructuren Doel: aantal straal-driehoek intersecties verminderen.

1.2 Doelstelling

Betere Acceleratiestructuur bouwen door een algemene BSP te maken die gebruik maakt van de geometrische normalen bij het splitsen. Aantal intersecties nog doen dalen, traversals stijgen, rendertijd dalen.

1.3 Methodologie

1.4 Contributie

1.5 Overzicht

Hoofdstuk 2

Voorgaand werk

Raytracing vereist acceleratiestructuren om efficiënt driehoeken in de scene te kunnen zoeken. Dit hoofdstuk start met een algemene uitleg over acceleratiestructuren en wijdt dan uit over één specifieke acceleratiestructuur: de *Binary Space Partitioning* (BSP) boom. De varianten van de BSP boom worden één voor één besproken.

2.1 Acceleratiestructuren

Het doel van acceleratiestructuren is om het aantal straal-driehoek intersecties te verminderen. De simpelste acceleratiestructuur bestaat uit het omhullende volume van de scene. Testen op intersectie met de driehoeken in de scene gebeurt dan enkel als dit omhullende volume intersecteert met de straal. Deze acceleratiestructuur kan worden uitgebreid tot een boomstructuur door dit volume recursief op te delen in kindvolumes. Binaire bomen delen elk omhullend volume op in twee nieuwe volumes, andere acceleratiestructuren zoals bijvoorbeeld octrees, delen het volume op in meer dan twee volumes.

Het volume kan worden opgedeeld op twee manieren: volgens objecten of volgens ruimte. Bij opdeling volgens objecten worden de objecten binnen het volume opgedeeld in meerdere disjuncte groepen en de kindvolumes zijn de omhullende volumes van deze groepen. Na deze opdeling zit elk object in exact één van deze nieuwe volumes, maar de volumes kunnen overlappen. Een voorbeeld van een acceleratiestructuur waarbij de opdeling volgens objecten gebeurt is de *Bounding Volume Hierarchy* (BVH). Opdeling volgens ruimte betekent dat de ruimte in het volume wordt opgedeeld in meerdere delen. Na deze opdeling overlappen deze nieuwe volumes niet, maar een object ligt nu in minstens één (en mogelijks in meerdere) kindvolume(s). De BSP boom deelt de ruimte van het volume steeds op in twee kindvolumes.

2.2 BSP bomen

De BSP boom deelt de ruimte van het omhullende volume recursief op door te splitsen volgens een willekeurig georiënteerd vlak totdat een bepaalde stopconditie bereikt is. Het feit dat de BSP volgens willekeurig georiënteerde vlakken splitst, is zowel een voor- als een nadeel. Het zorgt ervoor dat de BSP zich heel goed kan aanpassen aan de scene en alle niet-intersecterende driehoeken in principe kan scheiden. Maar het zorgt er ook voor dat het heel moeilijk is om deze goede splitsingsvlakken te vinden. In de praktijk wordt vaak een specifieke soort BSP boom gebruikt: de Kd boom.

Het bouwen van de boom Het splitsingsvlak dat een knoop opdeelt in twee delen, kan een grote impact hebben op het aantal doorkruisstappen en het aantal driehoekintersecties. Het bouwalgoritme moet voor elke knoop het beste splitsingvlak bepalen en als splitsen volgens dat vlak voordelig is, de knoop opsplitsen in twee kindknopen. Heuristieken voorspellen hoe goed een splitsing volgens een bepaald splitsingsvlak is. Het algoritme stopt met het splitsen van de knoop als het aantal driehoeken in de knoop lager is dan een vastgelegde limiet, bijvoorbeeld als er maar één driehoek in de knoop zit.

2.3 Kd Bomen

De Kd boom is een BSP boom waarbij alle splitsingsvlakken asgealigneerd zijn. Hiermee wordt bedoeld dat de splitsingsrichting (de normaal op het splitsingsvlak) evenwijdig is aan één van de drie hoofdassen. Dit zorgt ervoor dat elke knoop van de boom een asgealigneerde balk voorstelt. Het doorkruisen van een knoop uit een Kd boom is daardoor goedkoper dan het doorkruisen van een knoop uit een algemene BSP boom. De reden hiervoor is dat het goedkoper is om het intersectiepunt van een straal en een asgealigneerd vlak te vinden (verschil en vermenigvuldiging) dan het intersectiepunt van een straal en een willekeurig vlak (twee scalaire producten en deling). Door de beperking op mogelijke splitsingsvlakken kunnen Kd bomen zich minder goed aanpassen aan de scene. Ze kunnen bijvoorbeeld niet alle niet-intersecterende driehoeken scheiden.

Ondanks dit nadeel, worden in de praktijk Kd bomen verkozen boven algemene BSP bomen. Ize et al [IWP08] geven drie (volgens hun foute) ruimverspreide aannames over algemene BSP bomen die ervoor zorgen dat Kd bomen hoger ingeschat worden. Ten eerste wordt er aangenomen dat algemene BSP bomen nooit sneller kunnen zijn dan Kd bomen omdat het doorkruisen van een BSP boom beduidend duurder is. De beperkte precisie van vlottende komma getallen wordt gezien als het tweede probleem omdat het BSP bomen numeriek onstabiel zou maken. De derde aanname is dat door de grotere flexibileit (= grotere verzameling van mogelijke splitsingsvlakken) het veel moeilijker is om een BSP boom te bouwen dan om een Kd boom te bouwen.

Voor een Kd-boom is de Surface Area Heuristiek (SAH) de beste gekende methode om bomen met minimale verwachtte kost te bouwen. De SAH is oorspronkelijk ontwikkelt door Goldsmith en Salmon voor de BVH [GS87] en later aangepast door MacDonald en Booth voor de Kd boom [MB90]. De SAH schat de kost van een splitsingsvlak door te veronderstellen dat beide kindknopen, bladknopen worden en dat de kost om een bladknoop te intersecteren afhankelijk is van het aantal driehoeken en de oppervlakte van het omhullende volume. De verwachte kost $\mathcal{K}$ om een knoop p te splitsen in kindknopen l en r is dan: $\mathcal{K}_p = \frac{\mathcal{SA}(l)}{\mathcal{SA}(p)} * n_l * \mathcal{K}_i + \frac{\mathcal{SA}(r)}{\mathcal{SA}(p)} * n_r * \mathcal{K}_i + \mathcal{K}_d$ met kost $\mathcal{K}$, oppervlakte $\mathcal{SA}()$ en aantal driehoeken n. De subscripts i en d staan voor respectievelijk intersectie en doorkruising. De kost van het beste splitsingsvlak (laagste kost) wordt dan vergeleken met de kost om niet te splitsen: $n * \mathcal{K}_i$. Als splitsen voordelig is, wordt de knoop opgesplitst volgens dit splitsingsvlak.

Alle mogelijke asgealigneerde splitsingsvlakken testen is ondoenbaar. Havran toonde aan dat de kost van de SAH langs een richting ofwel lineair stijgt ofwel linear daalt tussen eindpunten (de minimale en maximale waarde) van de driehoeken langs die richting [Hav00]. Dit impliceert dat het beste splitsingsvlak door een eindpunt van een driehoek gaat. Het is dus voldoende om enkel de splitsingsvlakken te bekijken die door de eindpunten van de driehoeken gaan. Hieruit volgt dat er voor elke richting in een knoop met n driehoeken, $2n$ splitsingsvlakken bekeken moeten worden. Bij een Kd boom is het dus voldoende om $6n$ splitsingsvlakken te bekijken. Om de SAH kost te berekenen voor een splitsingsvlak moet het aantal driehoeken dat (deels) aan de ene kant van het splitsingsvlak ligt (n_l) en het aantal driehoeken dat (deels) aan de andere kant ligt (n_r) gekend zijn. Om deze aantallen efficiënt te kunnen berekenen, worden de eindpunten van de driehoeken in de knoop gesorteerd volgens hun projectie op de splitsingsrichting. Een veegbeweging (*sweep*) over deze gesorteerde lijst kan dan in lineaire tijd de SAH kost van alle splitsingsvlakken langs de splitsingsrichting berekenen.

2.4 RBSP-bomen

De Restricted Binary Space Partitioning ($RBSP$) boom is een uitbreiding van de Kd boom die toelaat om knopen op te splitsen volgens splitsingsrichtingen uit een vaste verzameling richtingen. Het omhullend volume van een $RBSP$ knoop is hierdoor geen balk maar een Discreet Geöriënteerde Polytoop met k richtingen: een k-DOP. Zowel Klosowski et al [KHM+98] als Zachmann [Zac98] hebben hiërarchiën van k-DOPs gebruikt binnen het domein van botsherkenning (*collision detection*). De $RBSP$ boom is voor het eerst gebruikt in ray tracing door Kammaje en Mora [KM07] en verder onderzocht door Budge et al [BCNJ08]. Aangezien de splitsingsrichtingen niet asgealigneerd moeten zijn, lijkt de $RBSP$ boom meer op de algemene BSP boom dan de Kd boom.

Een belangrijke ontwerpbeslissing bij $RBSP$ bomen is het kiezen van de verzameling splitsingsrichtingen. Volgens Budge et al werkt in principe elke verzameling

met minstens drie niet-evenwijdige richtingen, maar is het gewenst om richtingen te kiezen die samen de eenheidsbol goed bedekken. Kammaje en Mora genereren punten langs een spiraal op gelijk verdeelde breedtegraden volgens de gouden ratio. Budge et al gebruiken de verzamelingen die ze de standaard richtingen van Klosowski et al noemen. Deze richtingen gebruiken enkel waarden uit de verzameling $\{-1, 0, 1\}$ als x, y en z coördinaat.

Kammaje en Mora toonden aan dat de SAH ook gebruikt kan worden voor $RBSP$ bomen. Net zoals bij de Kd boom moeten voor elke splitsingsrichting de eindpunten van de driehoeken gesorteerd worden en moeten $2n$ splitsingsvlakken bekeken worden. In totaal bekijkt de $RBSP$ boom dus $2kn$ splitsingsvlakken. Het bouwen van de $RBSP$ boom is computationeel duurder dan het bouwen van de Kd boom door het grotere aantal richtingen en het feit dat het berekenen van de oppervlakte van een k-DOP computationeel duurder is dan het berekenen van de oppervlakte van een balk. Budge et al vonden een oplossing voor dit tweede probleem door een methode te ontwikkelen waarmee de oppervlaktes van de kind k-DOPs incrementeel berekend kunnen worden tijdens het sweepen.

De huidige $RBSP$ implementaties zijn superieur ten opzichte van de Kd boom in termen van het aantal driehoekintersecties en het aantal doorkruisingen, maar moet onderdoen in termen van rendertijd. Ize et al menen dat de $RBSP$ boom de slechtste eigenschappen van zowel de Kd boom als de algemene BSP boom overneemt. Net als de Kd boom kan het geen rekening houden met de lokale geometrie en zich dus niet aanpassen aan complexe geometrie. Net als bij de algemene BSP boom moet de intersectie met een willekeurig georiënteerd vlak berekend worden om knopen te doorkruisen. Budge et al tonen aan dat deze tweede eigenschap minder erg is bij de $RBSP$ boom dan bij de algemene BSP boom omdat alle scalaire producten op voorhand (en dus maar één keer) berekend kunnen worden.

2.5 Algemene BSP-Bomen in de praktijk

Ize et al ontwikkelden de eerste algemene BSP boom voor raytracing [IWP08]. In tegenstelling tot de Kd en $RBSP$ boom kan BSP_{IZE} wel splitsingsvlakken kiezen afhankelijk van de geometrie. De BSP_{IZE} boom gebruikt de splitsingsvlakken van de Kd boom en elke driehoek in de knoop bepaalt nog vier extra splitsingsvlakken. Deze vier vlakken zijn: het vlak van de driehoek zelf (autoparitie) en de drie vlakken loodrecht op de driehoek door elk van de drie zijden. Merk op dat deze laatste vier vlakken rekening houden met de geometrie en niet mogelijk zouden zijn bij een $RBSP$ boom. Het omhullend volume van de knopen bij een BSP boom is een convex veelvlak. Voor de eenvoud wordt de omhullende balk gebruikt als omhullend volume voor de wortelknoop. Dit heeft als extra voordeel dat er een zeer snelle test is voor stralen die niets raken.

De SAH kan ook gebruikt worden voor algemene BSP bomen. De BSP_{IZE}

boom controleert de $6n$ Kd splitsingsvlakken door te sweepen zoals bij de Kd en $RBSP$ boom. De SA berekening is duurder aangezien het omhullende volume een convex veelvlak is. De BSP_{IZE} boom controleert ook vier extra vlakken per driehoek. Voor deze vlakken is het berekenen van de SAH kost moeilijker omdat het aantal driehoeken links en rechts van het vlak bepaald moet worden. Om dit efficiënt te bepalen gebruiken Ize et al de BVH als hulpstructuur. In elke knoop wordt een BVH boom gebouwd en die wordt bij elke niet-Kd splitsing gebruikt om het aantal driehoeken in beide kindknopen te bepalen. Hierdoor is het bouwen van de BSP_{IZE} boom computationeel beduidend duurder dan het bouwen van $RBSP$ en Kd bomen. De inwendige knopen van de boom kunnen worden opgedeeld in twee groepen: de knopen die gesplitst worden door een Kd vlak en de knopen die gesplitst worden door een geometrie-afhankelijk BSP vlak. Met Kd knopen worden de knopen uit de eerste groep bedoeld, met BSP knopen die uit de tweede groep.

Ize et al verminderen het probleem van de tragere BSP boom doorkruising door Kd knopen apart te behandelen bij het doorkruisen. Als een Kd knoop doorkruist wordt, kan de intersectie met het splitsingsvlak berekend worden als de intersectie van de straal met een asgealigneerd vlak. De boom die deze optimalisatie toepast wordt aangeduid met BSP^{Kd}_{IZE}. Het feit dat Kd knopen goedkoper zijn om te doorkruisen dan BSP knopen, zorgt ervoor dat er voor deze twee soorten knopen een aparte doorkruiskost gebruikt moet worden in de SAH: $\mathcal{K}_{d,BSP}$ en $\mathcal{K}_{d,Kd}$. Deze kosten rechtstreeks in de SAH gebruiken, zorgt ervoor dat voornamelijk BSP knopen gecreëerd worden. De reden hiervoor is dat de intersectieterm van de SAH lineair varieert in het aantal driehoeken en die intersectieterm domineert snel de constante doorkruisterm waardoor BSP splitsingen, die de driehoeken beter splitsen, gekozen worden. Deze lineaire afhankelijkheid is geen probleem als de SAH enkel moet beslissen of een knoop gesplitst wordt of niet, maar wel als het moet bepalen of een knoop al of niet gesplitst wordt en of dit best door een goedkoep Kd vlak of door een duur BSP vlak gedaan wordt. Ize et al lossen dit probleem op door ook $\mathcal{K}_{d,BSP}$ lineair te laten variëren in het aantal driehoeken: $\mathcal{K}_{d,BSP} = \alpha * \mathcal{K}_i * (n-1) + \mathcal{K}_{d,Kd}$ waarbij α een instelbare parameter is. Als er na het testen van alle splitsingsvlakken geen splitsingsvlak gevonden is dat zorgt dat de kost om te splitsen lager is dan de kost om een bladknoop te creëren, worden alle BSP vlakken opnieuw getest, maar deze keer met een constante $\mathcal{K}_{d,BSP}$. Dit blijkt veel beter te werken dan een constante kost [IWP08].

De BSP^{Kd}_{IZE} boom is in veel scenes even snel of zelfs sneller dan de Kd boom. De niet geöptimaliseerde BSP_{IZE} boom is voor sommige scenes ook al beter dan de Kd boom. De winst wordt behaald door het lagere aantal straal driehoek intersecties. Het aantal knoop doorkruising stijgt echter, waardoor de totale verbetering wordt afgezwakt. De BSP^{Kd}_{wn+} boom toont veel minder variatie in rendertijd per pixel dan de Kd boom die duidelijke hotspot regio's heeft. Dit toont aan dat de BSP boom beter om kan met complexe geometrie.

2.6 Hiërarchie

Aantal verschillende splitsingsvlakken Eén van de belangrijkste eigenschappen van een BSP algoritme is zijn vermogen om zich aan te passen aan complexe geometrie. Het totaal aantal verschillende splitsingsvlakken dat bekeken wordt tijdens het bouwen van de boom, draagt bij tot dit vermogen. Voor een scene met n driehoeken, bekijkt een Kd boom $6n$ verschillende splitsingsvlakken. Elk niveau van de boom bekijkt exact dezelfde splitsingsvlakken. Een $RBSP$ boom met k discrete richtingen doet hetzelfde, maar bekijkt $2kn$ verschillende splitsingsvlakken. De BSP_{IZE} boom gebruikt $10n$ verschillende splitsingsrichtingen, $6n$ voor de Kd richtingen en n voor elk van de vier andere richtingen. De BSP_{IZE} boom bekijkt op elk niveau ook exact dezelfde splitsingvlakken en is dus niet zo algemeen als een BSP boom kan zijn. De reden hiervoor is dat de gebruikte BSP splitsingsvlakken enkel afhankelijk zijn van de driehoeken zelf en niet van welke driehoeken samen in een knoop zitten. Geen van bovenstaande bomen gebruikt de volledige vrijheid van een BSP boom om op elk niveau andere splitsingsvlakken te nemen en op die manier beperken ze hun vermogen om zich aan te passen aan complexe geometrie.

Hoofdstuk 3

BSP_{SWEEP}

3.1 Algemeen idee

De bekeken richtingen zijn verschillend per node Richtingen kunnen rekening houden met lokale geometrie (kan niet bij RBSP, wel bij BSPize) Sweeping van richtingen Kd-richtingen + aantal richtingen of puur die richtingen Snelle traversal voor kd-richtingen

3.2 Gebaseerd op geometrische normalen

3.2.1 Willekeurige normaal

Extra richtingen door random normalen te kiezen Autopartitie van Ize maar gesweeped Waarom zou dit werken ? : ...

3.2.2 Geclusterde normalen

(Extra) richtingen via K-means clustering Sweepen volgens die richtingen Waarom zou dit werken ...

3.3 Gebaseerd op random richtingen

(Extra) richtingen door random richtingen te kiezen Ter controle dat de richtingen met behulp van normalen, nuttige richtingen zijn Sweeping Waarom zou dit werken ? : Random per node itt vast bij Kd Driehoeken proberen te worden gesplitst volgens meer verschillende richtingen, Kd probeert steeds hetzelfde Kd heeft maar 3 opties, als het volgens geen kan -> nooit mogelijk Kans splitsbaar door Kd richting of random is even hoog in uniform geval. Scenes hebben wel veel asgealigneerde delen, dus daarom die extra.

Hoofdstuk 4

Implementatie

In dit hoofdstuk wordt het werk ingeleid. Het doel wordt gedefinieerd en er wordt uitgelegd wat de te volgen weg is (beter bekend als de rode draad).

Als je niet goed weet wat een masterproef is, kan je altijd Wikipedia eens nakijken.

4.1 Outline BSP-algoritmes

4.1.1 Bouwalgoritme

4.1.2 Intersectie-algoritme

4.2 Kd boom

4.3 $RBSP$ boom

4.4 BSP_{IZE}

4.5 BSP_{SWEEP}

4.5.1 Algemeen

4.5.2 BSP_{random}

4.5.3 BSP_{wn}

4.5.4 BSP_{wn}

Hoofdstuk 5

Resultaten

In dit hoofdstuk wordt het werk ingeleid. Het doel wordt gedefinieerd en er wordt uitgelegd wat de te volgen weg is (beter bekend als de rode draad).

Als je niet goed weet wat een masterproef is, kan je altijd Wikipedia eens nakijken.

5.1 Praktische aspecten

5.2 Afhankelijkheid van aantal richtingen

5.3 Vergelijking

	Feet		Sponza		Conference Hall		Museum	
K	Mediaan	Stdev	Mediaan	Stdev	Mediaan	Stdev	Mediaan	Stdev
4								
5								
6								
7								
8								
9								
10								

TABEL 5.1: Statistieken over de rendertijd voor $BSP^{Kd}_{random+}$ voor verschillende waarden van K. Voor elke waarde van K is het algoritme 6 keer uitgevoerd.

Bibliografie

[BCNJ08] B. C. Budge, D. Coming, D. Norpchen, and K. I. Joy. Accelerated building and ray tracing of restricted bsp trees. In *2008 IEEE Symposium on Interactive Ray Tracing*, pages 167–174, Aug 2008.

[GS87] Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, 1987.

[Hav00] Vlastimil Havran. *Heuristic ray shooting algorithms*. PhD thesis, Ph. d. thesis, Department of Computer Science and Engineering, Faculty of . . . , 2000.

[IWP08] T. Ize, I. Wald, and S. G. Parker. Ray tracing with the bsp tree. In *2008 IEEE Symposium on Interactive Ray Tracing*, pages 159–166, Aug 2008.

[KHM+98] James T Klosowski, Martin Held, Joseph SB Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[KM07] R. P. Kammaje and B. Mora. A study of restricted bsp trees for ray tracing. In *IEEE/ EG Symposium on Interactive Ray Tracing 2007(RT)*, volume 00, pages 55–62, 09 2007.

[MB90] J David MacDonald and Kellogg S Booth. Heuristics for ray tracing using space subdivision. *The Visual Computer*, 6(3):153–166, 1990.

[Zac98] Gabriel Zachmann. Rapid collision detection by dynamically aligned dop-trees. In *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No. 98CB36180)*, pages 90–97. IEEE, 1998.

Fiche masterproef

Student: Jesse Hoobergs

Titel: Distributiefunctie van de geometrische normalen gebruiken voor het bouwen van BSP acceleratiedatastructuren

Engelse titel: Using the distribution function of the geometric normals to build BSP acceleration data structures.

UDC: 621.3

Korte inhoud:

Hier komt een heel bondig abstract van hooguit 500 woorden. LaTeX commando's mogen hier gebruikt worden. Blanco lijnen (of het commando `\par`) zijn wel niet toegelaten!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie Mens-machine communicatie

Promotor: Prof. dr. ir. Philip Dutré

Assessoren: Ir. W. Eetveel
W. Eetrest

Begeleiders: Ir. M. Moulin
Ir. P. Bartels