

Het gebruik van de normalen bij het bouwen van BSP acceleratiestructuren

Thesisverdediging

Jesse Hoobergs
KU Leuven
Juni 2019

Promotor:
Prof. dr. ir. Ph. Dutré

0 Overzicht

- ① Inleiding
- ② BSP bomen
- ③ BSP_{SWEEP}
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

1 Outline

① Inleiding

② *BSP* bomen

③ *BSP_{SWEEP}*

④ Implementatie

⑤ Resultaten

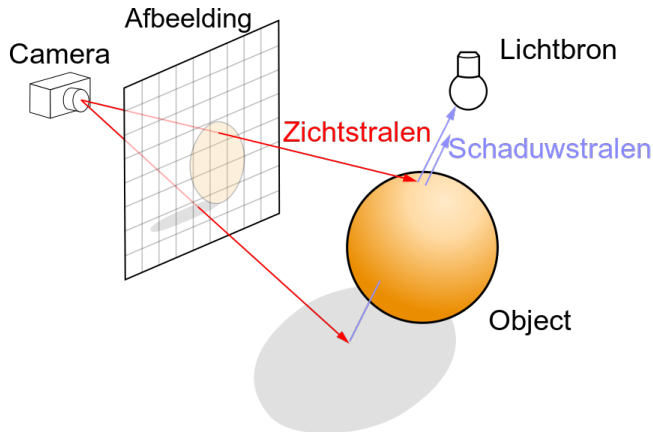
⑥ Conclusie

1 Ray tracing

- ▶ Fysisch gebaseerd renderen
- ▶ Stralen volgen door een scene

1 Ray tracing

- Fysisch gebaseerd renderen
- Stralen volgen door een scene



1 Ray tracing

► Praktische aantallen:

- 1 miljoen pixels
- 1 miljoen driehoeken (mogelijks veel meer)
- 100 stralen per pixel

1 Ray tracing

► Praktische aantallen:

- 1 miljoen pixels
- 1 miljoen driehoeken (mogelijks veel meer)
- 100 stralen per pixel

⇒ 10^{14} straal-driehoekintersecties

1 Ray tracing

► Praktische aantallen:

- 1 miljoen pixels
- 1 miljoen driehoeken (mogelijks veel meer)
- 100 stralen per pixel

⇒ 10^{14} straal-driehoekintersecties

⇒ Acceleratiestructuren

1 Acceleratiestructuren

► Doel:

- Totale rendertijd minimaliseren
- Straal-driehoekintersecties te verminderen

1 Acceleratiestructuren

► Doel:

- Totale rendertijd minimaliseren
- Straal-driehoekintersecties te verminderen

► Simpelste versie

- Omhullende volume van scene (balk, bol, etc)
- Test intersectie met omhullend volume
 - Intersectie: Test alle driehoeken
 - Geen intersectie: Test nul driehoeken
- Recursief opdelen tot boomstructuur

1 Acceleratiestructuren

► Doel:

- Totale rendertijd minimaliseren
- Straal-driehoekintersecties te verminderen

► Simpelste versie

- Omhullende volume van scene (balk, bol, etc)
- Test intersectie met omhullend volume
 - Intersectie: Test alle driehoeken
 - Geen intersectie: Test nul driehoeken
- Recursief opdelen tot boomstructuur

► Twee manieren van opdelen:

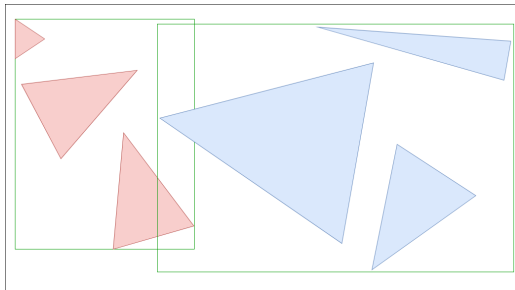
- Volgens objecten
- Volgens volume

1 Opdelen volgens object

- ▶ Driehoeken opgedeeld in disjuncte groepen
- ▶ Kindvolumes = omhullende volumes groepen
- ▶ Elke driehoek in exact één kindvolume
- ▶ Kindvolumes kunnen overlappen in de ruimte

1 Opdelen volgens object

- ▶ Driehoeken opgedeeld in disjuncte groepen
- ▶ Kindvolumes = omhullende volumes groepen
- ▶ Elke driehoek in exact één kindvolume
- ▶ Kindvolumes kunnen overlappen in de ruimte

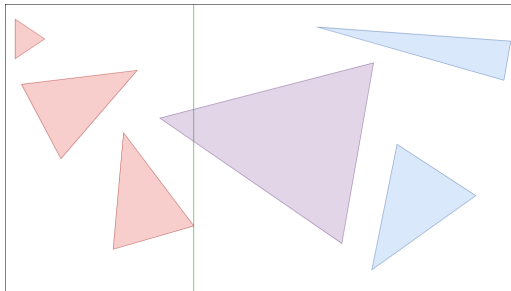


1 Opdelen volgens ruimte

- ▶ Volume opgedeeld in disjuncte groepen
- ▶ Splitsing via *splitsingsvlakken*
- ▶ Elke driehoek in minstens één kindvolume
- ▶ Kindvolumes overlappen niet in de ruimte

1 Opdelen volgens ruimte

- ▶ Volume opgedeeld in disjuncte groepen
- ▶ Splitsing via *splitsingsvlakken*
- ▶ Elke driehoek in minstens één kindvolume
- ▶ Kindvolumes overlappen niet in de ruimte



1 *BSP* bomen

- ▶ *Binary Space Partitioning* bomen
- ▶ Delen volgens ruimte
- ▶ Delen steeds in 2 kindvolumes
- ▶ Splitsing via willekeurige vlakken in de ruimte
- ▶ *Kd* boom:
 - Enkel asgealigneerde vlakken
 - Computatieve voordelen bij bouwen en renderen
 - Meest gebruikte *BSP* boom
- ▶ Algemene *BSP* boom:
 - Veel meer mogelijke splitsingsvlakken
 - Moeilijk om goede te vinden

1 Doel thesis

- ▶ Algemene BSP boom
- ▶ Normalen van driehoeken
- ▶ Goede splitsingsvlakken

- ▶ BSP_{SWEEP} boom

2 Outline

- ① Inleiding
- ② *BSP* bomen
- ③ *BSP_{SWEEP}*
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

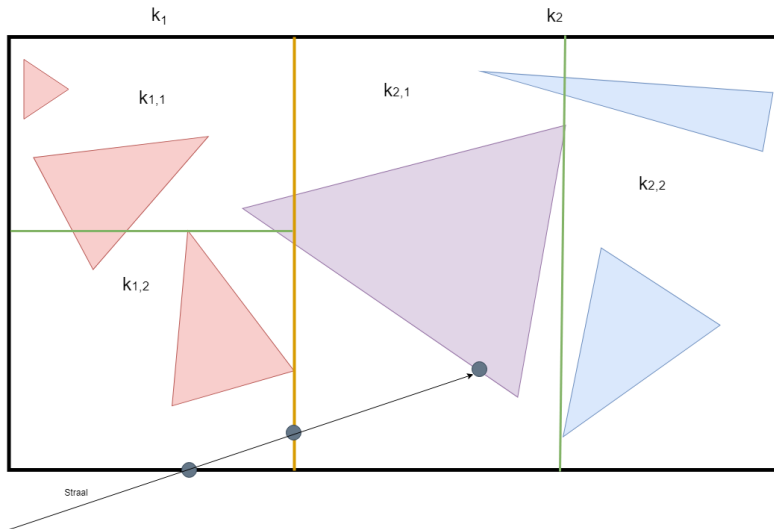
2 *BSP* bomen

- ▶ Bouwen
- ▶ Intersecteren
- ▶ Bestaande *BSP* bomen

2 Bouwen *BSP* boom

- ▶ Wortelknoop
 - Volume = omhullende balk scene
 - Bevat alle driehoeken
- ▶ Splits in twee kindknopen
 - Splits het volume volgens een splitsingsvlak
 - Maak twee kindknopen, één voor elk volumedeel
 - Bepaal voor elke kindknoop de driehoeken
- ▶ Splits kindknopen recursief tot stopconditie

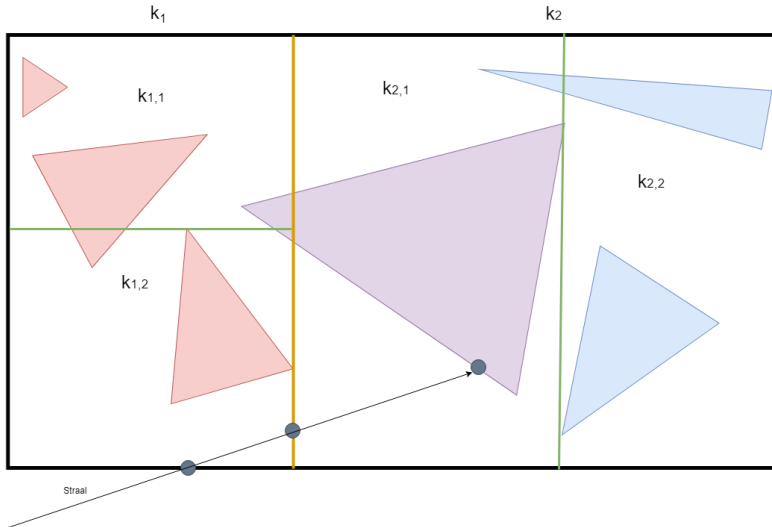
2 Bouwen *BSP* boom



2 Intersecteren *BSP* boom

- ▶ Bepaal intersectie straal met wortelknoop:
 - geen intersectie: geen intersecterende driehoek
 - anders: doorkruis de wortelknoop
- ▶ Doorkruisen knoop
 - Inwendige knoop
 - Bepaal intersectie straal met splitsingsvlak
 - Bepaal de volgorde waarin de straal door de kindknopen gaat
 - Doorkruis de kindknopen in deze volgorde
 - Bladknoop
 - Bepaal voor elke driehoek de straal-driehoek intersectie

2 Intersecteren *BSP* boom

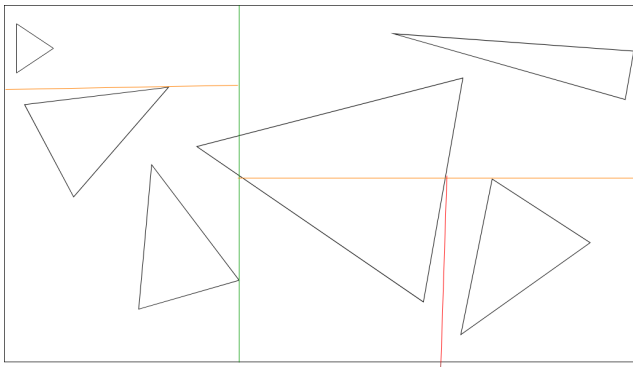


2 *Kd boom*

- ▶ Enkel asgealigneerde splitsingsvlakken
 - Volume elke knoop = asgealigneerde balk
 - Goedkoper doorkruisen inwendige knoop
 - Kunnen zich minder goed aanpassen aan scene

2 *Kd boom*

- Enkel asgealigneerde splitsingsvlakken
 - Volume elke knoop = asgealigneerde balk
 - Goedkoper doorkruisen inwendige knoop
 - Kunnen zich minder goed aanpassen aan scene



2 Bepalen beste splitsingsvlak

- ▶ Bepaal kost na splitsen volgens vlak
- ▶ Kies beste splitsingsvlak of splits niet

2 Bepalen beste splitsingsvlak

- ▶ Bepaal kost na splitsen volgens vlak
- ▶ Kies beste splitsingsvlak of splits niet
- ▶ *Surface Area* heuristiek
 - Kans om kindknoop te doorkruisen is evenredig met oppervlakte
 - Kost knoop evenredig met aantal driehoeken
 - Na splitsing zijn beide kindknopen bladknopen
 - $\mathcal{K}_p = \frac{SA(l)}{SA(p)} * n_l * \mathcal{K}_i + \frac{SA(r)}{SA(p)} * n_r * \mathcal{K}_i + \mathcal{K}_d$
 - Kost om niet te splitsen: $n_l * \mathcal{K}_i$

2 Bepalen beste splitsingsvlak

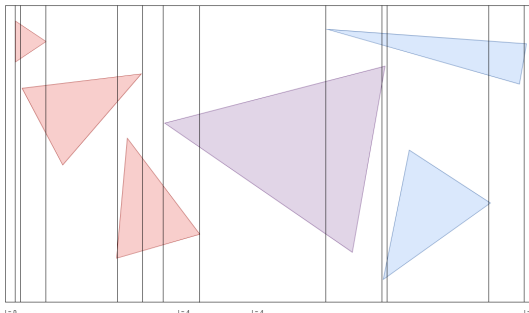
- ▶ Bepaal kost na splitsen volgens vlak
- ▶ Kies beste splitsingsvlak of splits niet
- ▶ *Surface Area* heuristiek
 - Kans om kindknoop te doorkruisen is evenredig met oppervlakte
 - Kost knoop evenredig met aantal driehoeken
 - Na splitsing zijn beide kindknopen bladknopen
 - $\mathcal{K}_p = \frac{SA(l)}{SA(p)} * n_l * \mathcal{K}_i + \frac{SA(r)}{SA(p)} * n_r * \mathcal{K}_i + \mathcal{K}_d$
 - Kost om niet te splitsen: $n_l * \mathcal{K}_i$
- ▶ Alle asgealigneerde vlakken testen is onhaalbaar
 - Havran: slechts $2n$ mogelijke splitsingsvlakken per richting
 - *SA* kost stijgt/daalt monotoon tussen eindpunten driehoeken langs die richting
 - Enkel asgealigneerde vlakken door eindpunten testen

2 Bepalen beste splitsingsvlak

- ▶ SA kost berekenen
 - Aantal driehoeken in beide kindknopen nodig
 - Oppervlaktes beide kindknopen nodig

2 Bepalen beste splitsingsvlak

- ▶ *SA* kost berekenen
 - Aantal driehoeken in beide kindknopen nodig
 - Oppervlaktes beide kindknopen nodig
- ▶ *Sweeping*
 - Sorteer driehoeken volgens eindknopen langs as
 - 'Veeg' over de as en update n_l en n_r

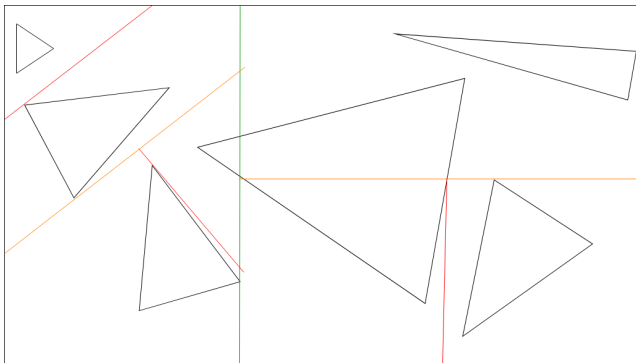


2 *RBSP* boom

- ▶ Enkel splitsingsrichtingen uit vaste verzameling van k richtingen
 - Volume elke knoop = $k - DOP$
 - Duurder doorkruisen inwendige knoop
 - Kunnen niet alle niet-intersecterende driehoeken scheiden

2 *RBSP* boom

- ▶ Enkel splitsingsrichtingen uit vaste verzameling van k richtingen
 - Volume elke knoop = $k - DOP$
 - Duurder doorkruisen inwendige knoop
 - Kunnen niet alle niet-intersecterende driehoeken scheiden



2 Praktisch

- ▶ Bepalen vaste verzameling splitsingsrichtingen
 - Belangrijk dat ze samen de eenheidsbol goed bedekken
- ▶ SA kost kan gebruikt worden, inclusief *sweeping*
- ▶ Oppervlakte $k - DOP$ berekenen is duurder
- ▶ Ten opzichte van Kd boom
 - Minder straal-driehoekintersecties
 - Tragere inwendige knoopdoorkruising
 - Tragere rendertijd

2 BSP_{IZE} boom

- ▶ Enige bestaande algemene BSP boom bij rendering
- ▶ Geometrie-afhankelijke splitsingsvlakken
 - De asgealigneerde vlakken van Kd boom
 - Vlak door elke driehoek
 - Drie vlakken door zijde driehoek en loodrecht op driehoek
- ▶ Volume elke knoop = convex veelvlak
- ▶ Sweeping
 - Mogelijk voor de Kd richtingen
 - Niet mogelijk voor de vier andere vlakken per driehoek
 - BVH hulpstructuur nodig om n_l en n_r efficiënt te berekenen
 - Tragere bouwtijd

2 BSP_{IZE}^{Kd} boom

- Optimalisatie

2 BSP_{IZE}^{Kd} boom

- ▶ Optimalisatie
- ▶ Inwendige Kd knopen bevoordelen
 - Sneller te doorkruisen dan BSP knopen
 - Lagere doorkruiskost in SA kost dan inwendige BSP knopen

2 BSP_{IZE}^{Kd} boom

- ▶ Optimalisatie
- ▶ Inwendige Kd knopen bevoordelen
 - Sneller te doorkruisen dan BSP knopen
 - Lagere doorkruiskost in SA kost dan inwendige BSP knopen
- ▶ Aanpassing SA heuristiek
 - Aparte $\mathcal{K}_{d,Kd}$ en $\mathcal{K}_{d,BSP}$

2 BSP_{IZE}^{Kd} boom

- ▶ Optimalisatie
- ▶ Inwendige Kd knopen bevoordelen
 - Sneller te doorkruisen dan BSP knopen
 - Lagere doorkruiskost in SA kost dan inwendige BSP knopen
- ▶ Aanpassing SA heuristiek
 - Aparte $\mathcal{K}_{d,Kd}$ en $\mathcal{K}_{d,BSP}$
 - Rechtstreeks gebruiken in SA kost werkt niet
 - SA kost varieert lineair in aantal driehoeken
 - BSP knopen splitsen beter
 - Bijna enkel BSP knopen gebruikt

2 BSP_{IZE}^{Kd} boom

- ▶ Optimalisatie
- ▶ Inwendige Kd knopen bevoordelen
 - Sneller te doorkruisen dan BSP knopen
 - Lagere doorkruiskost in SA kost dan inwendige BSP knopen
- ▶ Aanpassing SA heuristiek
 - Aparte $\mathcal{K}_{d,Kd}$ en $\mathcal{K}_{d,BSP}$
 - Rechtstreeks gebruiken in SA kost werkt niet
 - SA kost varieert lineair in aantal driehoeken
 - BSP knopen splitsen beter
 - Bijna enkel BSP knopen gebruikt
 - $\mathcal{K}_{d,BSP}$ lineair afhankelijk van aantal driehoek
 - $\mathcal{K}_{d,BSP} = \alpha * \mathcal{K}_i * (n - 1) + \mathcal{K}_{d,Kd}$
 - Beste splitsingsvlak zoeken
 - Indien niet gevonden, vaste $\mathcal{K}_{d,BSP}$

2 Vergelijking

- ▶ Ten opzichte van Kd boom
 - Geen volledige sweeping mogelijk
 - Minder straal-driehoekintersecties
 - Gemiddeld lichtjes tragere inwendige knoopdoorkruising
 - Lichtjes snellere rendertijd

2 Vergelijking

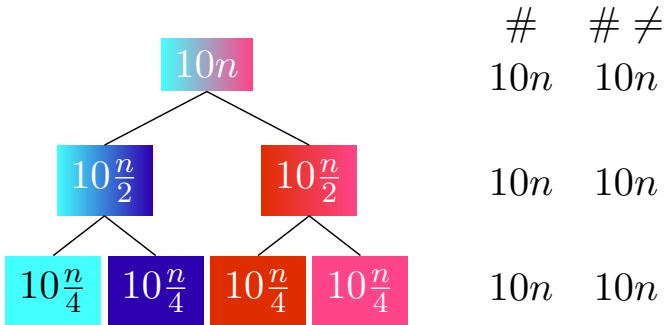
- ▶ Ten opzichte van Kd boom
 - Geen volledige sweeping mogelijk
 - Minder straal-driehoekintersecties
 - Gemiddeld lichtjes tragere inwendige knoopdoorkruising
 - Lichtjes snellere rendertijd
- ▶ Ten opzichte van $RBSP$ boom
 - Duurdere bouwtijd
 - Beide kunnen snelle Kd doorkruising gebruiken

2 Vergelijking

- ▶ Aantal splitsingsvlakken per niveau
 - Kd : $6n$
 - $RBSP$: $2kn$
 - BSP_{IZE} : $10n$
- ▶ Totaal aantal verschillende geteste splitsingsvlakken
 - Kd : $6n$
 - $RBSP$: $2kn$
 - BSP_{IZE} : $10n$
- ▶ Zelfs BSP_{IZE} gebruikt niet volledige vrijheid

2 Aantal splitsingsvlakken

- Zelfde splitsingsvlakken op elk niveau (bv BSP_{IZE})



- Driehoeken die in het bovenste niveau niet gesplitst kunnen worden
 - Kunnen in geen enkel niveau van elkaar gesplitst worden

3 Outline

- ① Inleiding
- ② BSP bomen
- ③ BSP_{SWEEP}
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

3 Concept

- ▶ Algemene *BSP* boom
- ▶ Geometrie-afhankelijke splitsingsrichtingen
 - In elke knoop k richtingen bepaald
 - Richtingen kunnen afhankelijk zijn van driehoeken in knoop
 - *Sweeping* over deze richtingen
 - Geen hulpstructuur nodig
- ▶ Drie ontwerpbeslissingen
 - Methode gebruikt om de k -richtingen te bepalen
 - Waarde van k
 - Kd richtingen altijd gebruiken of niet ?
- ▶ *RBSP* boom is BSP_{SWEEP} boom met steeds zelfde richtingen

3 Bepalen k-richtingen

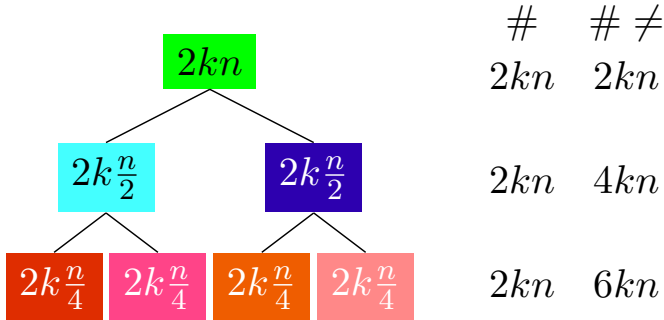
- ▶ BSP_{random}
 - Willekeurige richtingen (uniform op hemisfeer)
 - Idee: met veel verschillende (mogelijks slechte) vlakken proberen te splitsen
 - Kans op splitsing door willekeurige richting even groot als door Kd richting
- ▶ BSP_{wn}
 - Normalen van willekeurige driehoeken in de knoop
 - Idee: splitsen volgens oriëntatie driehoeken
 - Maakt gebruik van welke driehoeken samen in een knoop zitten
- ▶ BSP_{cn}
 - Clustercentra van normalen van de driehoeken in de knoop
 - Idee: splitsen volgens veelvoorkomende oriëntaties
 - Maakt gebruik van welke driehoeken samen in een knoop zitten

3 Kd richtingen gebruiken ?

- ▶ k richtingen genereren
 - BSP_{random}
 - BSP_{wn}
 - BSP_{cn}
- ▶ Kd richtingen en $k - 3$ richtingen genereren
 - Kd richtingen behandelen als BSP richtingen
 - $BSP_{random+}$
 - BSP_{wn+}
 - BSP_{cn+}
 - Kd richtingen apart behandelen
 - $BSP_{random+}^{Kd}$
 - BSP_{wn+}^{Kd}
 - BSP_{cn+}^{Kd}

3 Aantal splitsingsvlakken

- ▶ Andere splitsingsvlakken op elk niveau



- ▶ Driehoeken die in het bovenste niveau niet gesplitst kunnen worden
 - Kunnen op lagere niveaus misschien wel gesplitst worden
 - $\mathcal{O}(n \log(n))$ verschillende splitsingsvlakken ipv $\mathcal{O}(n)$

4 Outline

- ① Inleiding
- ② BSP bomen
- ③ BSP_{SWEEP}
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

4 Implementatie

- ▶ Pbrt-v3 renderer
- ▶ Enkel op de CPU
- ▶ Bouwen is niet geparallelliseerd
- ▶ Geïmplementeerde BSP bomen:
 - Kd
 - $RBSP$ en $RBSP^{Kd}$
 - BSP_{IZE} en BSP_{IZE}^{Kd}
 - $BSP_{random(+)}^{(Kd)}$, $BSP_{wn(+)}^{(Kd)}$ en $BSP_{cn(+)}^{(Kd)}$

5 Outline

- ① Inleiding
- ② BSP bomen
- ③ BSP_{SWEEP}
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

5 Scenes



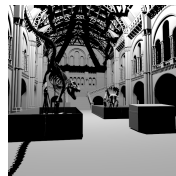
(a) Killeroo Been scene



(b) Sponza scene



(c) Conference scene



(d) Museum scene

- ▶ Killeroo: 33264 driehoeken
- ▶ Sponza: 227309 driehoeken
- ▶ Conference: 123651 driehoeken
- ▶ Museum: 1462840 driehoeken

5 Twee experimenten

- ▶ Afhankelijkheid van het aantal richtingen k bij BSP_{SWEEP}
 - Eerste drie scenes zeven keer gerenderd
 - Voor elk k -waarde van 2 tem 10
 - Voor elk van de negen BSP_{SWEEP} bomen
 - Uitvoering met rendertijd = mediaan zeven rendertijden gebruikt als representatieve
- ▶ Vergelijking met de andere BSP bomen
 - Alle vier scenes zeven keer gerenderd
 - Voor elke besproken bestaande BSP boom
 - Voor $BSP_{random+}^{Kd}$, BSP_{wn+}^{Kd} , BSP_{cn+}^{Kd} met optimale k -waarde
 - Uitvoering met rendertijd = mediaan zeven rendertijden gebruikt als representatieve

5 Afhankelijkheid van het aantal richtingen

► Bouwtijd

- Twee ordegroottes groter dan bouwtijd Kd boom
- Afhankelijkheid van k
 - $BSP_{random(+)}^{(Kd)}$ lineair
 - $BSP_{wn(+)}^{(Kd)}$ en $BSP_{cn(+)}^{(Kd)}$: sublineair
- Op zelfde wijze afhankelijk van aantal driehoeken als Kd boom
- Gebruik convex veelvlak is ongeveer 25 keer trager dan asgealigneerde balk

5 Afhankelijkheid van het aantal richtingen

► Aantal knopen

- Stijgt met stijgende k , maar vlakt snel af
- $BSP_{random(+)}^{(Kd)}$: 1.8 - 2 keer zoveel knopen als Kd boom
- $BSP_{wn(+)}^{(Kd)}$ en $BSP_{cn(+)}^{(Kd)}$: 1.2 - 1.3 keer zoveel knopen als Kd boom

► Procentueel aantal Kd knopen

- Daalt met stijgende k , maar vlakt snel af
- $BSP_{random+}^{Kd}$: 20% - 35 %
- BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd} : 30% - 45 %
- Afhankelijkheid van de diepte
 - Bovenste niveaus: bijna enkel Kd knopen
 - Onderste niveaus: bijna enkel BSP knopen

5 Rendertijd

- ▶ BSP_{SWEEP} boom zonder Kd richtingen
 - Daalt sterk met stijgende k
 - Enkel sneller dan Kd boom bij Killeroo Been scene
 - Daalt tot ongeveer 1.8 keer (BSP_{random}) en 1.3 keer (BSP_{wn} en BSP_{cn}) trager
- ▶ BSP_{SWEEP} boom met Kd richtingen als BSP richtingen
 - Daalt lichtjes met stijgende k en stijgt dan voor grotere k -waarden
 - Bij minstens één k -waarde sneller dan Kd boom
 - $BSP_{random+}$ trager dan BSP_{wn+} en BSP_{cn+}
- ▶ BSP_{SWEEP} boom met Kd richtingen apart behandeld
 - Daalt met stijgende k
 - Bij elk k -waarde sneller dan Kd boom
 - BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd} : daalt al minstens 15% voor $k = 4$
 - $BSP_{random+}^{Kd}$ trager dan BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd}

5 Afhankelijkheid van het aantal richtingen

► Aantal intersecties

- Verloop in functie van k zoals rendertijd
- BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd} :
 - daling tot 40%
 - vlak vanaf een k -waarde van 5
- $BSP_{random+}^{Kd}$:
 - daling tot 40%
 - daalt trager maar monotoon in functie van k

5 Afhankelijkheid van het aantal richtingen

► Aantal doorkruisingen

- BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd} :
 - evenveel als Kd boom
 - onafhankelijk van k
- $BSP_{random+}^{Kd}$:
 - stijgt tot 10% meer dan Kd boom
 - stijgt traag en monotoon in functie van k

► Procentueel aantal Kd doorkruisingen

- BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd} :
 - 85% - 90%
 - vlak vanaf k -waarde van 5
- $BSP_{random+}^{Kd}$:
 - 80% - 90%
 - daalt traag en monotoon in functie van k

5 Vergelijking bestaande bomen

► Nieuwe bomen

- $BSP_{random+}^{Kd}$ met $k = 10$
- BSP_{wn+}^{Kd} met $k = 6$
- BSP_{cn+}^{Kd} met $k = 6$
- $RBSP^{Kd}$ met $k = 13$

► Bestaande bomen

- Kd
- $RBSP$ met $k = 13$
- BSP_{IZE}
- BSP_{IZE}^{Kd}

5 Rendertijd en Bouwtijd

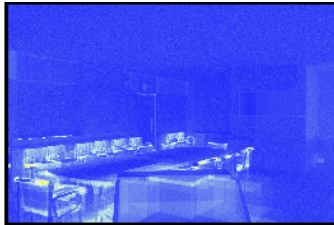
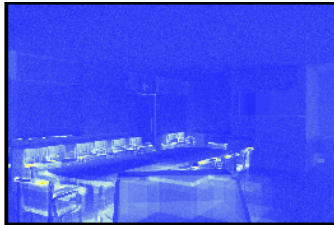
Boom	k	Killeroo Been		Sponza		Conference		Museum	
		R	B	R	B	R	B	R	B
Kd		100%	100%	100%	100%	100%	100%	100%	100%
BSP_{IZE}		94.1%	21 100%	165%	35 900%	105%	26 700%	88.2%	90 100%
BSP_{IZE}^{Kd}		95.0%	19 600%	109%	35 300%	92.5%	25 600%	84.7%	91 800%
BSP_{wn+}^{Kd}	6	67.7%	11 300%	80.4%	9460%	78.6%	8650%	63.4%	11 500%
BSP_{cn+}^{Kd}	6	67.8%	11 700%	80.3%	9740%	79.3%	8840%	65.9%	12 100%
$BSP_{random+}^{Kd}$	10	80.2%	17 800%	85.5%	16 100%	87.1%	15 400%	78.4%	21 800%
$RBSP$	13	82.7%	25 100%	131%	23 700%	101%	22 300%	482%	25 000%
$RBSP^{Kd}$	13	83.5%	25 700%	94.2%	23 600%	92.9%	22 200%	78.1%	24 900%

5 Straal-driehoekintersecties

Boom	k	Killeroo Been		Sponza		Conference		Museum	
		ZI	SI	ZI	SI	ZI	SI	ZI	SI
Kd		100%	100%	100%	100%	100%	100%	100%	100%
$BSP_{I\bar{Z}E}$		73.1%	72.7%	390%	483%	128%	140%	73.1%	46.6%
$BSP_{I\bar{Z}E}^{Kd}$		76.2%	75.2%	157%	142%	83.4%	81.9%	57.5%	61.4%
BSP_{wn+}^{Kd}	6	19.0%	16.1%	58.6%	54.3%	47.1%	34.2%	27.0%	40.2%
BSP_{cn+}^{Kd}	6	18.7%	16.4%	59.3%	54.5%	53.7%	41.3%	28.5%	44.8%
$BSP_{random+}^{Kd}$	10	35.2%	35.8%	49.9%	53.8%	52.5%	52.0%	36.8%	52.3%
$RBSP$	13	44.3%	45.2%	209%	239%	98.3%	111%	1380%	600%
$RBSP^{Kd}$	13	42.4%	43.0%	78.0%	75.8%	73.3%	78.4%	55.1%	33.1%

5 *False color* afbeeldingen intersecties

► BSP_{wn+}^{Kd} vs Kd



5 Inwendige knoopdoorkruisingen

Boom	k	Killeroo Been		Sponza		Conference		Museum	
		ZD	SD	ZD	SD	ZD	SD	ZD	SD
Kd		100%	100%	100%	100%	100%	100%	100%	100%
BSP_{IZE}		97.3%	101%	130%	128%	117%	115%	109%	98.4%
BSP_{IZE}^{Kd}		100%	102%	117%	109%	107%	107%	107%	103%
BSP_{wn+}^{Kd}	6	83.2%	83.9%	98.6%	99.6%	96.7%	96.5%	94.1%	90.6%
BSP_{cn+}^{Kd}	6	83.0%	85.0%	98.6%	99.5%	96.0%	97.0%	95.0%	94.0%
$BSP_{random+}^{Kd}$	10	95.8%	99.1%	108%	109%	108%	108%	106%	105%
$RBSP$	13	91.5%	96.1%	141%	138%	125%	127%	109%	97.1%
$RBSP^{Kd}$	13	95.3%	98.1%	117%	112%	110%	108%	105%	96.4%

5 Kd knoopdoorkruisingen

Boom	k	Killeroo Been			Sponza		
		Kd	ZD Kd	SD Kd	Kd	ZD Kd	SD Kd
Kd		100%	100%	100%	100%	100%	100%
BSP_{IZE}		49.4%	0%	0%	50.9%	0%	0%
BSP_{IZE}^{Kd}		55.1%	88.3%	88.3%	55.6%	88.2%	90.9%
BSP_{wn+}^{Kd}	6	29.5%	72.8%	72.3%	36.6%	88.8%	89.3%
BSP_{cn+}^{Kd}	6	29.2%	72.8%	71.3%	36.5%	88.5%	89.0%
$BSP_{random+}^{Kd}$	10	30.2%	74.6%	72.3%	34.5%	83.5%	84.9%
$RBSP$	13	22.0%	0%	0%	26.3%	0%	0%
$RBSP^{Kd}$	13	29.8%	71.2%	70.9%	33.2%	77.9%	79.2%

Boom	k	Conference			Museum		
		Kd	ZD Kd	SD Kd	Kd	ZD Kd	SD Kd
Kd		100%	100%	100%	100%	100%	100%
BSP_{IZE}		49.8%	0%	0%	38.5%	0%	0%
BSP_{IZE}^{Kd}		58.4%	89.7%	91.0%	41.7%	87.1%	87.1%
BSP_{wn+}^{Kd}	6	44.1%	92.3%	92.6%	32.0%	88.8%	87.3%
BSP_{cn+}^{Kd}	6	43.2%	90.0%	90.8%	30.5%	87.1%	86.8%
$BSP_{random+}^{Kd}$	10	35.3%	84.1%	87.1%	21.3%	83.1%	83.6%
$RBSP$	13	25.1%	0%	0%	18.3%	0%	0%
$RBSP^{Kd}$	13	34.3%	80.4%	82.6%	22.2%	78.9%	78.4%

6 Outline

- ① Inleiding
- ② BSP bomen
- ③ BSP_{SWEEP}
- ④ Implementatie
- ⑤ Resultaten
- ⑥ Conclusie

6 Conclusie

- ▶ BSP_{SWEEP}
 - Nuttig en uitbreidbaar concept voor algemene BSP bomen
 - Lokale geometrie makkelijk in rekening te brengen
 - Beperkte bouwtijd door *sweeping*
- ▶ Variant met snelle Kd richtingen is superieur
- ▶ $BSP_{random+}^{Kd}$
 - In elke knoop andere splitsingsvlakken kiezen maakt de boom beter
- ▶ BSP_{wn+}^{Kd} en BSP_{cn+}^{Kd}
 - Splitsingsvlakken afhankelijk van lokale geometrie (normalen) maken de boom nog beter
 - Vermindering rendertijd met meer dan 20%
 - Vermindering straal-driehoekintersecties met meer dan 40%
 - Clustering lijkt geen voordeel te bieden

6 Toekomstig onderzoek

- ▶ Verbeteren bouwtijd
- ▶ Bepalen beste aantal richtingen (per diepte)
- ▶ Bepalen betere richtingen
- ▶ Focus op splitsen in disjuncte delen ipv SA kost

Bedankt voor jullie aandacht