Haptisch renderen van CSG bomen

Jeroen CORNELISSEN

promotor :
Prof. dr. Karin CONINX

Universiteit Maastricht

Eindverhandeling voorgedragen tot het bekomen van de graad
Licentiaat in de informatica
afstudeervariant multimedia

universiteit hasselt

Voorwoord

Deze thesis werd geschreven en voorgedragen ter voltooiing van mijn studie informatica, optie multimedia, aan de *transnationale Universiteit Limburg* (tUL). Hiervoor heb ik onderzoek gedaan naar het haptisch modelleren van CSG bomen in het *Expertisecentrum voor Digitale Media* (EDM).

Hierbij wil ik graag iedereen oprecht bedanken die me geholpen heeft om deze thesis en mijn studies mogelijk te maken, waarvan enkele mensen in het bijzonder. In de eerste plaats zijn dit mijn promotor prof. dr. Karin Coninx en mijn begeleider prof. dr. Chris Raymaekers, voor hun enorme hulp, geduld, tips en suggesties. Tenslotte wil ik Tom de Weyer en Lode Vanacken bedanken voor hun hulp bij enkele software- en hardwareproblemen.

Daarnaast zou ik graag mijn ouders willen bedanken voor hun morele en financiële steun, en natuurlijk mijn vriendin Mireille. Haar steun op moeilijke momenten heeft me erg geholpen.

Jeroen Cornelissen

Abstract

Tegenwoordig zijn er tal van applicaties op de markt waarmee objecten interactief gemanipuleerd kunnen worden. Deze maken hiervoor haast allemaal gebruik van een combinatie van toetsenbord en muis als input en beperken de feedback tot het visueel tonen van de modelleeractie. Om het gevoel van immersie in zulke modelleeromgevingen te vergroten, kan er een haptische component toegevoegd worden. Gebruik makend van een toestel voor haptische interactie zoals de PHANToM, wordt het mogelijk om objecten in de scène niet alleen te voelen, maar ook interactief te kunnen manipuleren.

Dit is een haptische CSG modeller die gebruik maakt van CSG om wijzigingen aan de scène door te voeren. Ze is concreet geïmplementeerd, samen met twee interactieve tools: een interactieve boor en beitel. Aan de hand van deze twee tools wordt uitgelegd welke issues zich zoal kunnen voordoen tijdens het interactief haptisch modelleren van CSG bomen.

De bedoeling van deze thesis is om te onderzoeken hoe zo'n haptische modelleeromgeving gerealiseerd kan worden. Dit gebeurt aan de hand van een literatuurstudie, de ontwikkeling van de interactieve modelleeromgeving en de concrete modelleeroperaties.

Inhoudsopgave

Lijst van Figuren

Hoofdstuk 1. Inleiding

1.1 Algemeen doel

Deze thesis heeft als doel te onderzoeken hoe er haptisch gemodelleerd kan worden in objecten die gerepresenteerd worden door middel van Constructieve Solide Geometrie (CSG). Bovendien moet zo'n modeller concreet geïmplementeerd worden, zodat aangetoond kan worden dat haptisch modelleren mogelijk is op een intuïtieve manier en tegen nteractieve snelheid.

1.2 Aanpak

De thesis omvat enerzijds een literatuurstudie en anderzijds een implementatie van de gevonden resultaten.

De literatuurstudie bestaat uit het lezen en verwerken van papers die relevant zijn aan de verschillende domeinen waartoe deze thesis behoort. De belangrijkste paper is ongetwijfeld *'Algorithms for Haptic Rendering of CSG Trees'* van Prof. Frank Van Reeth en mijn co-promotor Prof. Chris Raymaekers, waar deze thesis zich voornamelijk op baseert en waarvan de referenties een ideaal uitgangspunt vormden op zoek naar relevante artikels.

Dit leidde tot een hele reeks artikels over solid modelling en de structuur, het visueel renderen en modelleren van CSG bomen enerzijds en tot artikels over haptics, haptics, haptische interactie, renderen en modelleren anderzijds. Een andere bron van informatie over haptics en haptisch renderen waren de Siggraph 99 course notes die heel wat nuttige papers bevatten. Hierdoor kon ik stilaan een beeld vormen van wat er bij zo'n haptische CSG modeller allemaal nodig is. Een lijst van al de papers die ik uiteindelijk in de thesis kon gebruiken vindt u bij de bibliografie sectie. Over CSG modelling op een intuïtieve en dus niet-rechtstreekse manier vond ik niets.

Voor het implementatiegedeelte had ik het geluk dat ik van meet af aan kon beschikken over geïmplementeerde algoritmes voor visuele en haptische CSG rendering, gemaakt door mijn co-promotor Prof. Chris Raymaekers in het kader van zijn hierboven vermelde papar. De haptische algoritmes maakten gebruik van het haptisch framewerk e-touch, maar werden omgezet in HAL [40], een op het EDM gemaakt haptisch framewerk (zie §5.5.7).

1.3 Overzicht

Haptisch modelleren van CSG bomen is niet mogelijk zonder voorafgaand grondig onderzoek van de verschillende domeinen die hier gecombineerd dienen te worden. Daarom volgt eerst een uitgebreide synthese van de

informatie die ik vergaarde tijdens de literatuurstudie. Hierbij is elk domein in een apart hoofdstuk vervat.

Allereerst wordt CSG onder de loupe genomen. CSG is een vorm van solid modelling en dit wordt eerst doorgenomen in Hoofdstuk 2. Daarna is CSG zelf aan de beurt. Dit is een zeer intuïtieve alternatieve methode om 3D objecten voor te stellen. Bovendien onderhoudt deze techniek de constructie geschiedenis, in tegenstelling tot meer de meer conventionele en zeer populaire B-rep (polygoon meshes), en is ze dus zeer geschikt voor het interactief modelleren van bijvoorbeeld toestellen, motoren, meubelen,… We bespreken in hoofdstuk 3 uitgebreid de structuur, CSG operaties en de verschillende methodes om CSG bomen visueel te renderen. Snel zal blijken dat een performant visueel renderalgoritme niet zo eenvoudig is te verwezelijken. In Hoofdstuk 4 gaan we na hoe CSG bomen gemodelleerd kunnen worden.

Daarna bekijken we in Hoofdstuk 5 de wereld van haptics en het haptisch renderen. Haptics vormen een relatief nieuw begrip binnen de virtuele realiteit, maar zijn zeker een belangrijke manier om immersieve interactie tussen de mens en virtuele omgevingen te bevorderen. We besteden hier heel wat aandacht aan de dingen die bij haptics komen kijken zoals de interactiecyclus, vergelijking met het menselijk gevoel, maar ook concrete toestellen en framewerken worden besproken. Vervolgens wordt uitgelegd hoe er haptisch gerenderd moet worden aan de hand van een Surface Contact Point (SCP).

Hoofdstuk 6 omvat een uitgebreide beschrijving van de reeds eerder vernoemde paper over haptisch renderen en combineert hierdoor CSG met haptics. Deze vormt het belangrijkste grondwerk voor deze thesis en is de sleutel voor de haptische modeller.

Na de literatuurstudie volgt in Hoofdstuk 7 een algemene bespreking van interactieve haptische CSG modelling. Dit hoofdstuk omvat zowel een beschrijving van haptisch modelleren als een overzicht van de tools en de abstracties die ten opzichte van de realiteit altijd gemaakt moeten worden. Hierbij wordt een verschil gemaakt tussen enkelvoudige en meervoudige haptische interacties. Hoe zulke haptische modelleeracties kunnen werken wordt uitgelegd aan de hand van twee concrete tools.

Hoofdstuk 8 bevat een meervoudige haptische boor en Hoofdstuk 9 bevat een enkelvoudige haptische beitel. Beide worden uitgebreid besproken en geïmplementeerd om aan te tonen of en vooral hoe deze haptische interacties zouden kunnen werken aan interactieve snelheden.

Hoofdstuk 10 bevat een reeks algemene conclusies in verband met de haptische CSG modeller. In hoofdstuk 11 tenslotte bevindt zich nog een overzicht van mogelijk Future Work, onderwerpen die interessant zijn voor verder onderzoek.

Hoofdstuk 2. 3D Objecten via CSG Bomen

2.1 Inleiding

Al langer dan informatica bestaat, proberen mensen reële dingen en ideeën voor te stellen op een eenvoudige en voor iedereen begrijpbare manier. Een zo'n voorbeeld het schrift: gedachten worden woorden, woorden worden letters of symbolen, die dan neergeschreven worden. Deze teksten vormen een *representatie* van de oorspronkelijke gedachten, en kunnen de gedachten soms benaderen, maar zijn nooit exact hetzelfde en blijven dus steeds een *beperkte* weergave. Het verschil tussen ideeën en hun representerende teksten heeft al voor menig geschil en conflict gezorgd, wat bijvoorbeeld tot uiting komt bij het verschil tussen de letter en de geest van de wet. Zo zullen ook getekende of geschilderde 3D landschappen of voorwerpen nooit een exacte weergave zijn, maar slechts een benadering.

In de informatica probeert men ook reële dingen voor te stellen of te *digitaliseren*, maar het is steeds slechts een benadering. Zo zijn er door de jaren heen verschillende methoden bedacht om reële, solide 3D objecten digitaal voor te stellen. Elke representatiewijze heeft een bepaald uitgangspunt, zoals zijn omhulsel of ruimtelijke occupatie, en zijn eigen voor- en nadelen, maar zal nooit de enige, ideale voorstelling zijn. In §2.2 wordt het modelleren van solide 3D objecten gedefinieerd en een overzicht gegeven van verschillende representatiewijzen.

De representatie voor 3D objecten waarover het in de thesis gaat, is Constructieve Solide Geometrie. Deze methode ziet een 3D object als een combinatie en bewerking van eenvoudigere objecten, zoals u kan zien in §2.3, en wordt voorgesteld door een *boomstructuur*, de CSG boom (§2.4).

2.2. Solid Modelling

Solide objecten zijn 3D objecten die niet vervormbaar (*non-deformable*) en 'gevuld' zijn. Met dit laatste wordt bedoeld dat dat zo'n solied object niet enkel uit het zichtbare omhulsel bestaat, maar dat ook het binnenste van dit object omvat is in zijn representatie.

Solid modelling is een vorm van 3D modelleren die zich bezighoudt met het ondubbelzinnig representeren van zulke solide 3D objecten. Dit wordt ook wel eens *volume modelling* genoemd.

Feitelijk verdeelt Solid Modelling de 3D ruimte in een gedeelte dat 'binnen' het object ligt, en een gedeelte dat erbuiten ligt [1].

2.2.1 Representaties

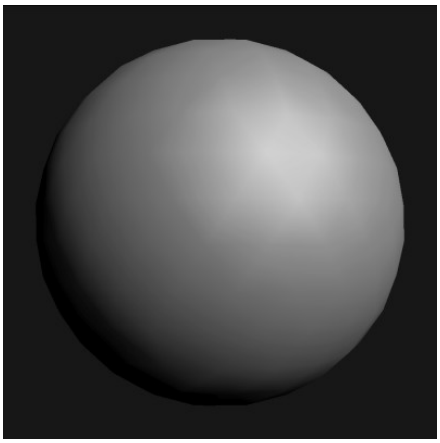
Er zijn verschillende manieren om 3D objecten voor te stellen. Hier volgen enkele veelgebruikte representatievormen.

1. Constructieve Solide Geometrie (CSG)

Bij CSG worden primitieve 3D objecten zoals een (volle) sfeer, kubus, balk, kegel, cilinder,... gecombineerd tot complexere objecten door gebruik te maken van:

- geometrische transformaties (translatie, rotatie)
- zgn. Booleaanse (binaire) operatoren (unie, doorsnede, verschil).

We komen hier veel uitgebreider op terug in §2.3.



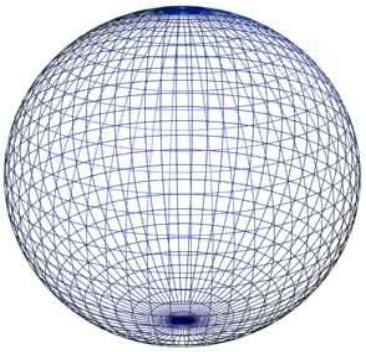
Figuur 1: Sfeer voorgesteld via CSG

2. Boundary Representatie

Bij boundary representatie (*B-rep*) wordt een solide geometrie voorgesteld via zijn geometrische of meestal door zijn topologische (*Euler*) grenzen (*boundaries*) of omhulsel. Dit omhulsel dient gesloten te zijn; er mogen met andere woorden geen *'gaten'* in voorkomen.

Een grensbeschrijving via Euler topologie bestaat uit een reeks punten (*vertices*), verbonden met lijnen (*edges*), die allemaal tegen elkaar aanliggende vlakjes vormen (*faces*).

Solide Geometrieën kunnen voorgesteld worden door een gesloten omhulsel te 'vullen' om het solied te maken. B-Rep kan men vergelijken met het maken van figuren met behulp van een mal.



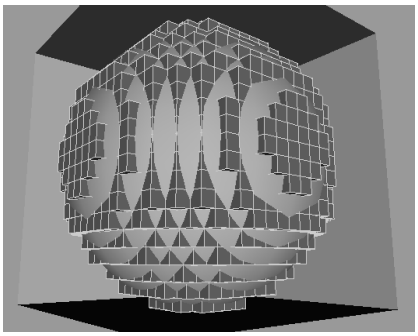
Figuur 2: Sfeer, voorgesteld door boundary representation (bron: wikipedia [15]

Vaak wordt een model gebouwd met behulp van een CSG representatie, maar wordt deze opgeslagen als een B-Rep. Tegenwoordig slaan de meeste parametrisch gebaseerde CAD systemen zowel de CSG als de B-Rep op waardoor er naderhand nog steeds constructieve wijzigingen aangebracht kunnen worden.

B-Rep objecten kunnen ondermeer opgeslagen en uitgewisseld worden met behulp van STEP (STandard for the Exchange of Product data) [16], een ISO standaard formaat dat hiervoor ontwikkeld werd.

3. Spatiale Enumeratie

Spatiale enumeratie deelt de ruimte op in ***cellen***. Deze waarin het solied object zich bevindt, vormen zijn representatie [4].



Figuur 3: Sfeer, voorgesteld via voxels (Bron : [18])

Deelt men de 3D ruimte op in reguliere, gevulde cellen, dan spreekt men van *Spatiale Occupatie*. Zulke reguliere cellen worden meestal *volumetrische elementen* of *voxels ('VOlume piXELS')* genoemd, een soort 3D versie van pixels die 2D beelden voorstellen (zie Figuur 3).

Net zoals pixels bevatten voxels meestal niet hun absolute positie in de ruimte, maar hun relatieve plaats ten opzichte van de andere voxels waaruit het object is opgebouwd.

Technieken op basis van voxels worden vaak gebruikt voor het analyseren van medische (o.a. CT scans) en wetenschappelijke gegevens, maar komen ook voor in bepaalde computer games.

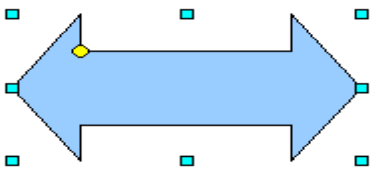
Spatiale Enumeratie kan ook *hiërarchisch geadapteerd* worden. Dit wordt *cel decompositie* genoemd. In tegenstelling met Spatiale Occupatie zijn de cellen hier noch regulier, noch geprefabriceerd. Cel decompositie wordt meestal gebruikt voor het analyseren van gegevens, als deel van de preprocessing.

4. Andere Representaties

Andere representaties om aan Solide Modellering te doen zijn ondermeer:

- ***Kerven*** (*Sweeping* of *'Sketcher Based Modelling'*) : hierbij wordt een bepaalde ruimte *uitgekerfd* door een primitief object dat zich volgens een pad verplaatst om zo een solide geometrie te vormen. Dit kerfproces voegt ofwel volume toe en vormt zo een solied object (*'extrusion'*), ofwel verwijdert het materiaal uit een reeds bestaand object (*'cutter path'*). Deze methode is analoog aan toestellen die bijvoorbeeld buizen aan de lopende meter maken.

- ***Geparametriseerde instantiëring*** van primitieven : hierbij wordt een object bepaald door een referentie naar een bibliotheek van geparametriseerde primitieve objecten en een reeks parameterwaarden. Een 2D voorbeeld hiervan zijn de blokpijltjes die in presentatie-programma's gebruikt worden (zie Figuur 4)



Figuur 4: Blokpijlen, een 2D voorbeeld van geparametriseerde primitieven

- ***Oppervlak Modellering*** (*'Facet Modelling'*): hier wordt het solied object gevormd door zijn oppervlak samen te stellen uit verschillende *veelhoekige vlakken*. Deze vorm van 'Solid Modelling' wordt vaak gebruikt bij 'reverse engineering' van fysieke modellen.

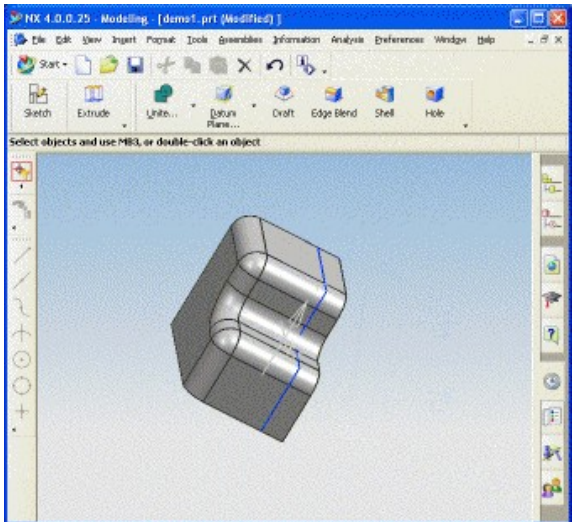
- ***Feature Based Modelling***: hierbij worden complexe combinaties van objecten en operatoren beschouwd als één geheel dat vervolgens gewijzigd en vermenigvuldigd kan worden. De volgorde van de uitgevoerde operaties wordt bijgehouden in een geschiedenis boom en parametrische veranderingen kunnen hun effect doorheen de boom laten gelden.

- ***Parametrisch Modelleren***: eigenschappen van modellen worden geparametriseerd en krijgen dus variabele in plaats van vaste waarden. Bovendien worden de verhoudingen tussen de verschillende parameters *binnen het model bijgehouden*, waardoor veranderen van parameterwaarden eenvoudiger wordt. Deze methode wordt zo goed als altijd gecombineerd met andere methodes zoals CSG in CAD toepassingen.

2.2.2 Toepassingen

Toepassingen van solid modelling situeren zich voornamelijk binnen [15] :

- *Computer Graphics* en *Computer Animatie*
- *Rapid Prototyping* (de snelle constructie van fysieke objecten zoals stereolithografie, Selective Laser Sintering (SLS), Fused Deposition Modelling (FDM), ...)
- *Medische tests*
- *Visualisatie van producten* en van wetenschappelijk onderzoek.



Figuur 5: Solid modelling in Computer Aided Design

2.2.3 Alternatieven voor solid modelling

Er zijn nog andere modelleermethodes dan Solid Modelling. Hieronder bevinden zich:

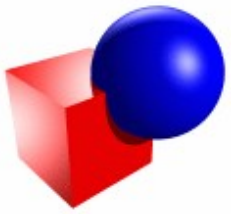
- *Surface Modelling:* objecten worden enkel bepaald door hun omhulsel. Zulke modellen kunnen wel vervormd worden en worden gebruikt bij het ontwikkelen van product design, de entertainment industrie en in 3D animaties.
- *Wireframe Modelling:* objecten worden bepaald via een *draadmodel*. Deze modellen kunnen voor solide volumes echter *dubbelzinnig* zijn.

2.3 Constructieve Solide Geometrie (CSG)

2.3.1 Algemeen

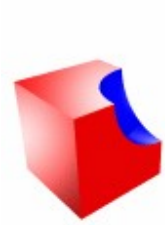
Constructieve Solide Geometrie (CSG) is een eenvoudige en intuïtieve manier om solide (m.a.w. gevulde) 3D objecten voor te stellen.

Een door CSG gevormd 3D object is een ***constructie*** van één of meer eenvoudige ***solide*** 3D objecten tot een complexer object via ***geometrische transformaties*** en ***Booleaanse*** *(binaire)* ***operatoren****.*



Figuur 6: CSG object als unie van een kubus en een sfeer. [15]

De eenvoudigste solide objecten waaruit een CSG object opgebouwd wordt noemt men ***primitieven***. Deze zijn objecten van een eenvoudige vorm en zijn vaak convex zoals de sfeer, kubus, cilinder, balk, kegel, prisma,... Soms bestaat de groep van primitieven ook uit niet-convexe objecten zoals de torus, die *2-convex* is (zie §3.3.4). Het aantal toegelaten primitieven hangt af van het softwarepakket. Sommige softwarepakketten laten CSG ook toe op glooiende objecten, anderen niet. De primitieven worden in de scène geplaatst aan de hand van de geometrische transformaties *translatie*, *rotatie* en *scalering*.



Figuur 7: CSG object als verschil van een kubus en een sfeer.

Door gebruik te maken van de ***Booleaanse operatoren*** *unie*, *intersectie* (doorsnede) en *subtractie* (verschil), verdeelt CSG, net als andere technieken voor solide modellering de 3D ruimte in een gedeelte dat *binnen* het object ligt, en een gedeelte dat *erbuiten* ligt [1] (zie respectievelijk Figuren 6, 7 en 8). Daarenboven maken deze operatoren de complexe vormen van grotere CSG objecten mogelijk.

Figuur 8: CSG object als doorsnede van een kubus en een sfeer.

Vaak is een CSG object de representatie van een model of oppervlak dat visueel complex lijkt, maar eigenlijk niet meer is dan een intelligente combinatie van primitieve objecten.

Alternatieve methodes om solide 3D objecten voor te stellen kan u te vinden in §2.3.1.

2.3.2 Toepassingen

Constructieve Solide Geometrie heeft heel wat praktische toepassingen. Het wordt vooral gebruikt wanneer :

- ***eenvoudige geometrische objecten*** gewenst zijn, of
- ***mathematische precisie*** belangrijk is.

CSG is dus een vaak procedurale modelleertechniek die gebruikt wordt in computergestuurde tekenprogramma's (CAD) en 3D computer games en graphics.

CAD: De meeste CAD programma's maken gebruik van CSG of een combinatie van CSG met andere technieken voor solid modelling zoals de B-Rep. Ze gebruiken CSG onder andere omdat

- CSG de beste benadering geeft op gebied van de accuraatheid van het model, wegens de mathematische precisie
- CSG objecten efficient opgeslagen kunnen worden
- analyze van CSG objecten vrij snel gaat.

Waar modellering op basis van polygonale en B-rep zich vaak enkel focussen op de oppervlakken van de objecten, focust CSG modellering zich op het volledige volume van de objecten, en hun inhoud. Op deze manier kunnen CAD applicaties meer dan het oppervlak van de objecten gebruiken. Zo kunnen objecten gebouwd worden met 'reële' materialen, dichtheden, en volumes zodat analysten hoermee ook fysische fenomenen kunnen bestuderen zoals kogelinslagen of transport van thermische of radioactieve aard. Een bekend open-source CAD applicatie, die veelvuldig gebruikt maakt van CSG is *BRL-CAD* [29].

Engines voor ***computer games*** maken ook gebruik van CSG bij het opbouwen van hun werelden. Level bouwers kunnen gebruik maken van een hele reeks primitieven en hierin kerven, snijden, ze combineren en manipuleren. Zo gebruikt de Unreal engine CSG, net als *Hammer*, de 'mapping engine' van de Valve's *Source Engine*.

Figuur 9: De lichtcirkels uit de film Tron werden opgebouwd met behulp van CSG [37]

Tenslotte wordt CSG ook gebruikt om 3D graphics te creëren voor andere media dan computer games. Een zo'n voorbeeld is de ***film industrie***, die soms gebruik maakt van CSG bij het genereren van bepaalde effecten, zoals de lichtcirkels achter de auto's in de virtuele race uit de film *Tron* (zie Figuur 9).

2.3.3 Voor- en nadelen

Vergeleken met de meer algemene boundary representaties hebben Constructieve Solide Geometrieën hun voordelen en hun nadelen. We zetten er hier enkele op een rij.

Voordelen tegenover B-rep zijn :

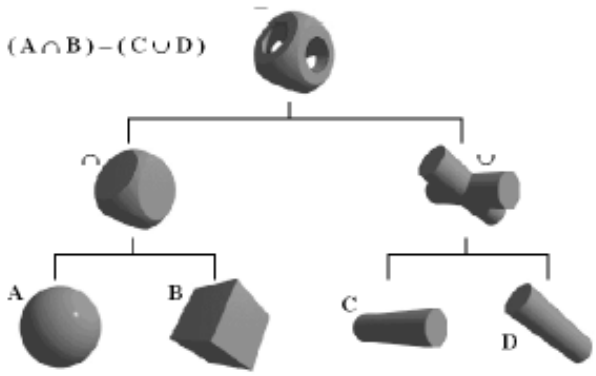
- Bepalen of een punt binnen of buiten een object ligt gaat veel sneller
- *Constructiegeschiedenis* wordt opgeslagen in de CSG boom (vervalt wel wat na normaliseren en snoeien)
- Conversie van CSG naar B-rep kan, maar deze conversie kan niet geïnverteerd worden aangezien de constructie geschiedenis van een solide model dat bij CSG bomen opgeslagen wordt, verloren geraakt tijdens de conversie naar B-rep. En er is geen algemene methode beschikbaar om vanuit deze B-rep terug een CSG boom te construeren.

Nadelen van CSG ten opzichte van B-rep zijn :

- CSG leent zich minder tot grafisch renderen dan B-rep omdat B-rep objecten reeds bestaan uit polygonen en de grafische hardware van PC's hiervoor geoptimaliseerd is.

2.4 CSG bomen

Een 3D object dat voorgesteld of geconstrueerd wordt door middel van *Constructieve Solide Geometrie* bestaat uit een *acyclische binaire boom* (zie Figuur 10). Het resultaat van deze boom, namelijk het volledige 3D object bevindt zich in zijn wortel (*root*).



Figuur 10: Voorbeeld van een CSG Boom [4]

In deze boom bestaan

- de *interne takken* enkel uit Booleaanse operatoren (U, ∩, –) en
- de *bladeren* uit (geometrisch getransformeerde) primitieven.

Op deze manier wordt een bepaalde groep primitieve objecten *bewerkt* en *gecombineerd* via Booleaanse operaties en vormt zo een complex solied object. Hiervoor wordt de boom vanaf de bladeren tot aan de wortel doorzocht via *diepte-eerst* (*depth first*). Hier wordt echter vereist dat de Booleaanse operatoren *expliciet bepaald* worden. Een CSG boom representeert dus steeds *bestaande objecten.* Bovendien zijn de operatoren *gesloten* voor CSG bomen. Zo geeft bijvoorbeeld de doorsnede van twee CSG bomen steeds opnieuw een correcte CSG boom.

Een CSG boom is *ondubbelzinnig*, maar niet *uniek*: er kunnen bijgevolg voor elk geconstrueerd object *meerdere correcte CSG bomen geconstrueerd worden*. Het is dus mogelijk om CSG bomen te normaliseren (§3.4.2) en te snoeien (*prunen*) (§3.4.3), zonder dat het resulterend geometrisch object gewijzigd wordt. Deze eigenschappen zullen tot een meer optimale en dus snellere verwerking leiden, zowel bij het visueel als haptisch renderen van CSG objecten.

CSG bomen kunnen ook voorgesteld worden als Booleaanse expressies, ook wel *producten* genoemd. De CSG boom uit Figuur 10 kan bijgevolg voorgesteld worden als (A∩B)-(CUD), waarbij A, B, C en D de verschillende primitieven zijn. Op deze manier kan makkelijk aangetoond worden dat CSG bomen niet uniek zijn: de CSG boom uit Figuur 10 kan namelijk ook voorgesteld worden door onderandere het product (((A∩B)-C)-D). Dit alternatief product is trouwens de genormaliseerde voorstelling en is uitgetekend te vinden in Figuur 11 (zie p.16).

2.5 Conclusie

Dit hoofdstuk begint met een algemene definitie van *Solid Modelling,* het digitaal voorstellen van solide 3D objecten (§2.2). Er zijn verschillende manieren om aan aan Solid Modelling te doen. Van deze verschillende representaties wordt een overzicht gegeven, alsook van enkele alternatieven voor Solid Modelling.

De manier waarop in deze thesis aan solid modelling wordt gedaan is *Constructieve Solide Geometrie*, of *CSG* (§2.3). We zagen dat deze CSG complexe 3D objecten kan voorstellen door eenvoudige objecten, *primitieven* genaamd, via de binaire operaties te combineren en te bewerken. Verder werd nagegaan welke toepassingen CSG gebruiken en welke mogelijkheden CSG biedt.

Tenslotte werd uitgelegd dat een CSG object concreet wordt voorgesteld door middel van een binaire boom, de CSG boom (§2.4). Deze boom bevat zowel de primitieven waaruit het CSG object is opgebouwd als zijn Booleaanse operaties. Bovendien kan een CSG boom ook voorgesteld worden als een binair product en kunnen er verschillende bomen geconstrueerd worden voor hetzelfde 3D object. Deze eigenschappen maken het mogelijk om deze CSG bomen snel te renderen. Deze methoden om snel visueel te renderen vindt men terug in Hoofdstuk 3. Methoden om CSG bomen haptisch te renderen staan in Hoofdstuk 6.

H3. Visueel renderen van CSG bomen

3.1 Inleiding

Voorstellingen door middel van CSG bomen zijn in 3D omgevingen pas écht van nut wanneer ze ook daadwerkelijk in beeld gebracht kunnen worden. Om zo een visueel beeld van een CSG boom te genereren, zijn er verschillende methodes ontwikkeld. Dit hoofdstuk heeft een overzicht van de verschillende methodes om CSG bomen visueel weer te geven en bespreekt hun sterke en zwakke eigenschappen.

Het sleutel element in het visueel renderen van CSG bomen is het *bepalen welke delen van primitieven in het CSG object zichtbaar zijn voor de gebruiker*. Hiervoor is een *classificatie* nodig van de *randen* van de verschillende primitieven waaruit het CSG object is samengesteld. Men noemt dit *surface clipping*. Het kan gebeuren via classificatie van *oppervlakken* of van *punten*. Aangezien de verschillende methodes die we in dit hoofdstuk behandelen steeds een keuze hebben moeten maken tussen *oppervlak classificatie* (§3.2) en *punt classificatie* (§3.3), worden ze op deze manier gecategoriseerd.

Bij de methodes met punt classificatie bevindt zich het *Goldfeather render algoritme* (§3.4). Deze belangrijke methode buit voor het eerst het *normaliseren* (§3.4.3) en *snoeien* (§3.4.4) van CSG bomen uit om zo het visueel renderen te optimaliseren tot real-time snelheden. Het algoritme van *Wiegand* (§3.5) past Goldfeather rendering aan tot een methode die geïmplementeerd kan worden op standaard 3D hardware. Verder wordt dit algoritme nog uitgebreid met extra algoritmes van *Steward en Leach* (§3.6), die het mogelijk maken om meer dan één primitieve tegelijk te clippen, waardoor de snelheid nog verhoogt. Tenslotte worden kort nog enkele verdere verfijningen besproken (§3.7).

Door deze technieken is het tegenwoordig perfect mogelijk om op gelijk welke moderne computer CSG bomen interactief visueel te renderen en te modelleren, zoals reeds gebeurt in verschillende CAD toepassingen.

3.2 Methodes met oppervlak classificatie

3.3.1 Algemeen

Bij oppervlak classificatie worden alle oppervlakken van alle primitieven in de CSG boom met elkaar vergeleken. Geometrisch gezien worden hierbij alle oppervlakken van de primitieven met elkaar geïntersecteerd. Deze paragraaf bespreekt enkele methodes die gebruik maken van oppervlak classificatie.

Oppervlak classificatie is doorgaans *trager* dan punt classificatie maar is *onafhankelijk van het standpunt van de gebruiker*. Een statisch CSG object dient dus slechts één maal berekend te worden. Om interactief te kunnen modelleren, zijn er echter methodes nodig die een wijziging van het CSG object vrijwel onmiddellijk op het scherm kunnen weergeven. Hierin schieten methodes op basis van oppervlak classificatie echter tekort.

3.3.2 Converteren naar B-rep

Eén voor de hand liggende manier om een CSG boom visueel te renderen is via een *conversie* naar de meer traditionele *boundary representation* of *B-rep* [6].

Bij het converteren naar een B-rep worden de oppervlakken van elke primitieve geclassificeerd in degene die zich *binnen*, *buiten* of *op* het oppervlak van het totale CSG object bevinden. Enkel de vlakken aan het oppervlak van het CSG object zijn effectief zichtbaar en worden gerenderd. Na omzetting bestaat de visuele voorstelling van de CSG boom dan uit een samenstelling van veelhoeken en kan ze met behulp van een standaard grafische bibliotheek gerenderd worden.

Voordelen:

- Het beeld is *onafhankelijk van het standpunt van de gebruiker.*
- Wordt *wel* door graphics hardware ondersteund.

Nadeel:

- Trage initialisatie, daarna gelukkig sneller, tenzij voor een zeer groot aantal takken.
- De omzetting van meer complexe CSG bomen naar B-reps is te traag om deze bomen te converteren tegen een tempo dat interactieve modellering mogelijk moet maken.

Omzetting naar een B-rep laat toe het solide object eenvoudigweg te bewerken en zeer snel opnieuw te renderen, maar kan niet gebruikt worden voor modellering aangezien het zoals eerder vermeld onmogelijk is een B-rep terug om te zetten in een CSG boom.

3.3.3 Spatiale Enumeratie

Een CSG boom kan ook gerenderd worden door gebruik te maken van spatiale enumeratie (zie ook §2.2.1 puntje 3). Dit kan op twee manieren: ofwel door de ruimte hiërarchisch in te gaan delen, ofwel door het object op te bouwen uit een aantal reguliere geprefabriceerde celletjes (*voxels*).

De methode van Thibault en Naylor [28] rendert een CSG boom door het opdelen van de ruimte en maakt maakt voor elke primitieve in de CSG boom een aparte BSP (*Binary Space Partitioning*) boom.

Een BSP boom verdeelt de ruimte bij elke iteratie in twee delen en kijkt dan of zo'n deel wel, niet of gedeeltelijk door de primitieve wordt ingenomen. Hoe dieper de BSP boom, hoe accurater zijn overeenkomsten met de originele CSG boom.

Daarna wordt er een oppervlak classificatie gedaan door de verschillende BSP bomen met elkaar te vermengen. Men bekomt een boom die vergelijkbaar is met een BSP boom van een B-rep van dat model.

Dit proces is te traag om aan interactieve rendering te doen, laat staan interactieve modelling. Thibault en Naylor hebben echter van dit algoritme ook een incrementele versie gemaakt die interactieve rendering binnen een modelleeromgeving wel mogelijk maken.

CSG objecten kunnen ook gerenderd worden door ze op te bouwen aan de hand van *uniforme voxels*. Hier kan men elke primitieve op voorhand definiëren aan de hand van de voxels die het inneemt. Daarna kunnen via oppervlak classificatie alle voxelverzamelingen met elkaar vergeleken worden om zo te kijken welke wel en welke geen deel uitmaken van de ruimte die ingenomen wordt door de originele CSG boom. Dit kan ook simpelweg voor elke voxel berekend worden, al wordt op deze manier de methode wel heel traag.

Voordelen:

- Het gerenderd beeld is onafhankelijk van de kijkrichting.
- Volumes kunnen op een eenvoudige manier geïntersecteerd kunnen worden.

Nadelen:

- Voor zo'n representatie moet een grote hoeveelheid data opgeslagen worden
- Data van over de oppervlakte van het object wordt slecht gerenderd. BSP bomen moeten immers zeer lang zijn of de Voxels zeer klein om een 'korrelig' resultaat te vermijden.
- Er zijn speciale algoritmes nodig om zo'n beeld daadwerkelijk op het scherm gerenderd te krijgen met gewone scan-line technieken.

3.3 Methodes met punt classificatie

3.3.1 Algemeen

Bij *punt classificatie* moeten enkel de punten op het oppervlak geclassificeerd worden die in de projectie liggen van een pixel op het scherm. Geometrisch gezien wordt dus elke primitieve geïntersecteerd met stralen vanuit elke pixel. Methodes zoals *ray casting*, *scan-line-* en *dieptebufferalgoritmes*, maken gebruik van punt classificatie. Deze methodes proberen hierbij om deze punt classificatie binnen hun algoritmes te integreren binnen een zo laag mogelijk niveau [8].

Punt classificatie is *simpeler* en doorgaans *sneller* dan oppervlak classificatie maar is *afhankelijk van het standpunt van de gebruiker* en moet dus niet alleen bij elke verandering aan het CSG object, maar ook bij elke wijziging van de kijkpositie en -richting opnieuw berekend worden. Dit lijkt een nadeel, maar aangezien er tegenwoordig methodes bestaan om CSG bomen via punt classificatie interactief te renderen, zijn deze methodes meteen ook geschikt om gebruik te worden in interactieve modellers met voortdurend wijzigende CSG objecten.

3.3.2 Ray casting

De meest voor de hand liggende methode is *ray casting* [5]: voor elke pixel van het te renderen beeld wordt een straal (*ray*) 'geschoten' (*casting*). Van elke primitieve in het CSG object waarmee deze straal intersecteert, wordt vervolgens gecontroleerd waar het intersectiepunt zich bevindt.

Het punt dat het dichtst bij het standpunt van de gebruiker ligt, en dat zichtbaar is, wordt gekozen en diens kleur wordt aan de pixel toegekend. Een primitieve in een CSG boom is niet altijd zichtbaar aangezien het ook om een gesubtraheerde of geïntersecteerde primitieve kan gaan, bijvoorbeeld een geboord gat.

Voordelen:

- Zeer mooi beeld.
- Techniek kan gebruikt worden voor een brede waaier aan objecten waaronder analytisch bepaalde en samengestelde oppervlakken.
- Eenvoudig te implementeren
- Robuuster dan de B-Rep benadering

Nadelen:

- Het beeld is afhankelijk is het standpunt van de gebruiker. Het dient dus bij elke wijziging van dit standpunt opnieuw berekend te worden.
- De methode is *zeer traag*
- Wordt *zelden* door graphics hardware ondersteund.

3.3.3 Scan-line Technieken

Het is ook mogelijk een CSG boom te renderen door de standaard rendering technieken uit te breiden. Dit kan door een een soort classificatie te maken van verschillende punten in de ruimte of door voor elke scan-line bepaalde oppervlakken in een lijst bij te houden en die te clippen over hele reeksen pixels.[4]

Nadelen:

- Het resultaat is afhankelijk is van het standpunt van de kijker.
- Er is geen hardware ondersteuning voor zo'n algoritme.

3.3.4 Algoritmes met een dieptebuffer (Z-buffer)

Om snelle visuele rendering mogelijk te maken wordt tegenwoordig gewerkt met algoritmes die gebruik maken van een *dieptebuffer* [4,7,8], de *Z-buffer*. De *surfaces* worden *geclipt* in een Z-buffer nadat de CSG boom eerst verregaand *genormaliseerd* wordt.

Deze algoritmes lenen zich het best voor het visueel renderen van CSG bomen [7,8]. Sommige methodes zijn zo efficiënt dat met momenteel mogelijk is CSG bomen interactief te renderen op standaard hardware. Het is bijgevolg mogelijk om modelleer omgevingen te creëren die CSG modellen gebruiken (§3.5). Dit biedt nieuwe perspectieven voor iedereen die bezig is met constructie van voorwerpen. Zo kan een designer een gat, gedefinieerd door een complex object, door een bestaand object boren en de nieuw ontstaande vorm observeren en verder bewerken.

Voordelen:

- Er is *geen omzetting naar een B-rep* nodig.
- De methodes zijn relatief snel.
- Er is uitgebreide hardware ondersteuning.

Nadeel:

- Is afhankelijk van de snelheid waarmee de dieptebuffer(s) gekopieerd wordt(worden). Dit kopiëren is in de meeste grafische hardware niet geoptimaliseerd.

In de volgende paragrafen worden enkele van deze visuele renderalgoritmes voor CSG bomen besproken.

3.4 Goldfeather CSG Rendering

3.4.1 Algemeen

Goldfeather CSG rendering [7] is een algoritme dat interactief renderen van CSG bomen mogelijk maakt. Het werkt net zoals de andere dieptebuffer algoritmes met *punt classificatie*.

De CSG boom wordt visueel gerenderd via *pixel-parallelle* operaties. Hierbij maakt Goldfeather rendering gebruik van een uitgebouwde frame buffer, het zogenaamd *Pixel-vlak*. Dit bestaat uit:

- twee dieptebuffers,
- twee kleurenbuffers
- bitvlaggen per pixel

Goldfeather's algoritmes vergen echter dat een CSG boom zich bevindt in een *genormaliseerde vorm*.

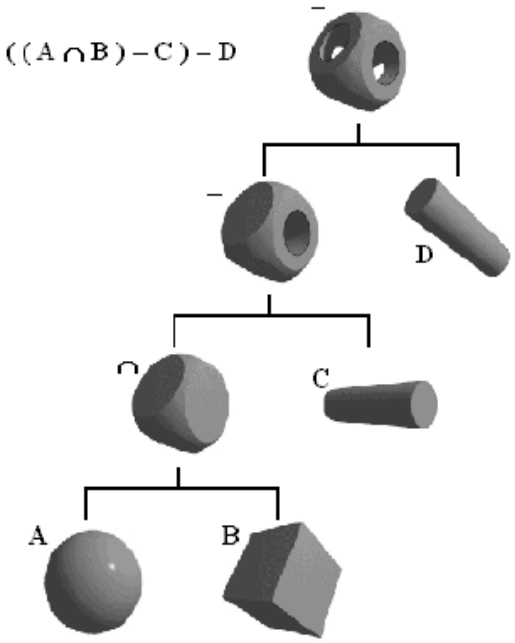
3.4.2 Normaliseren van CSG bomen

We zeggen dat een CSG boom *genormaliseerd* is wanneer de boom bestaat uit een *som van producten*.

In zo'n genormaliseerde boom hebben alle ∩ of – operatoren een primitieve als rechter kind, en een deel-boom (of ook een primitieve) als linker kind.

Deze deel-bomen bevatten geen U operatoren: alle takken met U operatoren worden vanboven in de boom geplaatst [7].

In Figuur 11 bevindt zich de genormaliseerde versie van de CSG boom uit Figuur 10 (zie p.9). We zien een vrijwel geheel naar links georiënteerde structuur die men tegenkomt in alle takken van een CSG-boom die zich bevinden *onder* de takken met U operatoren.



Figuur 11: Genormaliseerde CSG tree van deze uit Figuur 10 [4]

Werking:
Aangezien CSG-bomen equivalent zijn aan Booleaanse expressies, kunnen we normalisatie regels voorstellen als conversies hiertussen. Zo komt de structuur van de CSG boom uit figuur 10 overeen met de Booleaanse expressie (A∩B)-(CUD). Dit kan men normaliseren tot ((A∩B)-C)-D zoals te zien is in Figuur 11.

In het algemeen dient men volgende productieregels herhaaldelijk toe te passen om een *genormaliseerde CSG boom* [8, 9] te bekomen:

```
1.    X − (YUZ)    -->    (X−Y) − Z
2.    X ∩ (YUZ)    -->    (X∩Y) U (X∩Z)
3.    X − (Y∩Z)    -->    (X−Y) U (X−Z)
4.    X ∩ (Y∩Z)    -->    (X∩Y) ∩ Z
5.    X − (Y−Z)    -->    (X−Y) U (X∩Z)
6.    X ∩ (Y−Z)    -->    (X∩Y) − Z
7.    (X−Y) ∩ Z    -->    (X∩Z) − Y
8.    (XUY) − Z    -->    (X−Z) U (Y−Z)
9.    (XUY) ∩ Z    -->    (X∩Z) U (Y∩Z)
```

Een normalisatie functie ziet er dan ook ongeveer zo uit [7]:

```
functie Normaliseer (B : boom)
{
      als B een primitieve is
            keer terug ;

      herhaal
      {
            zolang B voldoet aan een van de productieregels
                  pas deze regel toe ;
            Normaliseer (B.links) ;
      }
      tot (B.operator is U)
            of
          ((B.links is een primitieve) en (B.links is geen U)) ;

      Normaliseer (B.rechts) ;
}
```

Normalisatie is belangrijk aangezien genormaliseerde CSG bomen gerenderd kunnen worden via een veel eenvoudiger algoritme. Het zet een willekeurige CSG boom om in een 'som van producten' (met U als 'som' en ∩ & – (= ∩ met het complement) als 'producten') en is de eerste stap op weg naar een geheel dat bruikbaar is dieptebuffer architectuur, waar *primitieven* met elkaar vergeleken worden in plaats van deelbomen.

Normalisatie maakt het eenvoudiger om een CSG-boom te renderen. De unie van twee of meer solide objecten kan dan gerenderd worden door gebruik te maken van een oppervlakte clipping algoritme dat via de dieptebuffer de verborgen oppervlakken verwijdert. Zulke algorites zijn standaard ingebouwd in de meeste grafische omgevingen. Het renderalgoritme dient enkel de juiste diepte en kleur te berekenen voor elke tak in de genormaliseerde boom, waarna gebruik gemaakt wordt van een dieptebuffer om daarna de data voor elke tak samen te voegen.

Het normalisatieproces kan echter veel bladeren met primitieven toevoegen aan een CSG boom waardoor de grootte van de boom in het slechtste geval exponentieel kan toenemen. Het aantal extra takken hangt hierbij af van de *structuur van de boom* en de *geometrie van de primitieven*. Daarom dient de CSG boom na normalisatie *gesnoeid* te worden.

3.4.3 Snoeien van CSG bomen

Na doorvoering van normalisatie, kan de CSG boom veel overbodige takken bevatten. In de meeste gevallen spelen een groot aantal van deze door normalisatie gegenereerde produkten echter geen rol in het uiteindelijk beeld omdat hun primitieven niet intersecteren. Deze kunnen dan *gesnoeid* (*gepruned*) worden.

Om te bepalen welke takken van de CSG boom gesnoeid kunnen worden wordt gebruik gemaakt van geometrische informatie. Hierbij wordt een omhullende kubus (een *bounding box)* berekend rond de objecten aan elke tak. Deze berekening gebeurt voor elke operator via de volgende

productieregels:

```
1. Bound (AUB) = Bound (Bound(A) U (Bound(B))
2. Bound (A∩B) = Bound (Bound(A) ∩ (Bound(B))
3. Bound (A–B) = Bound (A)
```

Indien beide bounding boxen niet intersecteren, kunnen volgende takken gesnoeid worden:

- een ∩ -tak, en
- de rechter deel-boom van een '–' -tak

Hierbij zijn A en B willekeurige takken uit de CSG boom. De boom kan dus na elke stap van het normalisatie proces gesnoeid worden. Dit gebeurt door de volgende regels op deze tak toe te passen:

```
1. (A∩B) --> Ø indien Bound (A) niet intersecteert met Bound (B)
2. (A-B) --> A indien Bound (A) niet intersecteert met Bound (B)
```

Er kan ook nog een derde regel toegepast worden, het *subtractie snoeien*.

```
3. A♭∩B  --> Ø indien Bound (A) niet intersecteert met Bound (B)
```

Hiervoor dienen eerst de produkten uit de CSG boom uitgebreid worden tot *partiële producten* (zie §3.4.4).

Performantie: Na het snoeien zal het aantal produkten van een orde zijn tussen **O(n)** en **O(n²)** ten opzichte van het totaal aantal primitieven in de boom. De *gemiddelde lengte* van een product, i, is meestal *klein* en *onafhankelijk van het totaal aantal primitieven*. Wanneer langere produkten voorkomen, zijn ze meestal van de vorm A-B-C-D... en dus vrij vatbaar voor subtractie snoeien [7,8].

3.4.4 Partiële Producten

Elk product in een CSG kan als volgt gerenderd worden: eerst wordt *elk zichtbaar oppervlak* van een primitieve gerenderd. Daarna wordt het oppervlak *getrimd* (geïntersecteerd of gesubtraheerd) met de overblijvende primitieven uit het product.

De zichtbare oppervlakken zijn dan de *naar voren gekeerde oppervlakken van de niet gesubtraheerde primitieven* en de *naar achter gekeerde oppervlakken van de wel gesubtraheerde primitieven*. Hierdoor kan de CSG boom opgesplitst worden in een som van partiële produkten.

- Een convexe primitieve heeft dan één paar oppervlakken per pixel, één naar voor gekeerd en één naar achter.
- Een niet-convexe primitieve kan meer zulke oppervlak paren hebben per pixel.

Een k-convexe primitieve wordt gedefinieerd als een primitieve die maximaal k oppervlakte paren per pixel heeft vanuit om het even welke kijkrichting.

Een k-convexe primitieve wordt genoteerd als A_k. Hierbij is

- A_{fn} zijn n-de naar *voren* gericht oppervlak tussen 0 en k-1 en
- A_{bn} zijn n-de naar *achter* gericht oppervlak tussen 0 en k-1.

Wanneer we werken met convexe objecten, kunnen we de n weglaten. Zo wordt A-B uitgebreid tot $(A_f\text{-}B)U(B_b \cap A)$ in zijn vorm van partiële producten.

Voor een k-convexe primitieve wordt A_k-B wordt dan uitgebreid tot $(A_{f0}\text{-}B)U...U\ (A_{fk}\text{-}1\text{-}B)\ U\ (B_b \cap A_k)$.

De primitieve wiens oppervlak gerenderd wordt noemen we de *beoogde primitieve* van het partieel product. De overgebleven primitieven worden *getrimde primitieven* genoemd.

Conversie naar een som van partiële produkten vereenvoudigt het probleem opnieuw. Het probleem is nu namelijk teruggebracht tot het correct renderen van partiële producten alvorens de resultaten van deze producten samen te voegen in de dieptebuffer. Het is pas op dit moment dat subtractie snoeien kan plaatsvinden.

3.4.5 Renderen en Classificeren van partiële producten.

Een partieel product wordt gerenderd in twee stappen. Eerst wordt het *beoogde oppervlak* van het partieel produkt gerenderd. Daarna wordt elke pixel op het oppervlak *parallel* geclassificeerd ten opzichte van de *getrimde primitieven*.

Om deel uit te maken van het oppervlak van een partieel produkt *moet* elke pixel:

- naar *binnen* liggen vergeleken met de *niet gesubtraheerde primitieven* en
- naar *buiten* liggen in vergelijking met de *wel gesubtraheerde primitieven*.

De pixels die niet aan deze criteria voldoen, worden weggelaten omdat ze geen deel uitmaken van het oppervlak van het CSG object. Concreet wordt hun kleur teruggezet naar de achtergrondkleur en hun diepte terug op de standaarddiepte.

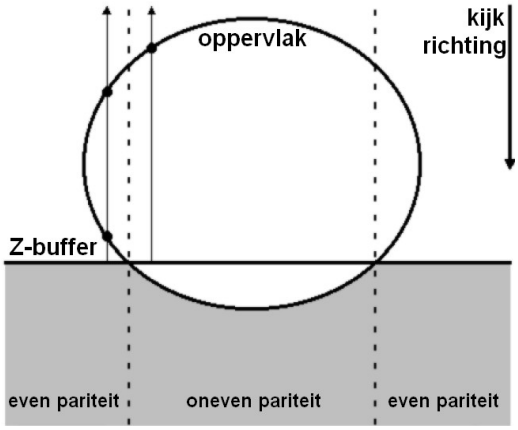
Aangezien primitieven in een CSG boom steeds bestaan uit een *gesloten omhulsel*, kunnen pixels geclassificeerd worden tegen een getrimde primitieve. Dit gebeurt via het principe van *oppervlaktepariteit*.

3.4.6 Oppervlaktepariteit

Met de oppervlaktepariteit bedoelt men dat er gekeken wordt of een bepaald vlak uit de dieptebuffer zich binnen of buiten een bepaald volume bevindt [7]. De pariteit van elk dieptebuffer element kan bepaald worden door middel van een algoritme dat erg lijkt op het standaard '*is kleiner dan*' -algoritme om dieptes te testen.

Men telt hier het aantal oppervlakken dat zich *vóór* dat bepaald vlak in de dieptebuffer bevindt.

Waar bij het gewone diepte testalgoritme de dieptebuffer van de pixel geüpdatet wordt na een succesvolle test, wordt hier zijn pariteitsbit gewijzigd (zie Figuur 12)



Figuur 12: Dieptebuffer pariteit voor oppervlakken

Is de pariteitsbit

- oneven (bit staat op 1), dan is dit een dieptebuffer element waarvan het volume zich *binnen* het object bevindt.
- even (bit staat op 0), dan bevindt dit dieptebuffer element zich er*buiten*

Voor het opslaan van deze pariteitsbit wordt gebruik gemaakt van een *stencil buffer*.

Correcte uitvoering van deze pariteitstesten is enkel mogelijk wanneer rekening gehouden wordt met deze vereisten:

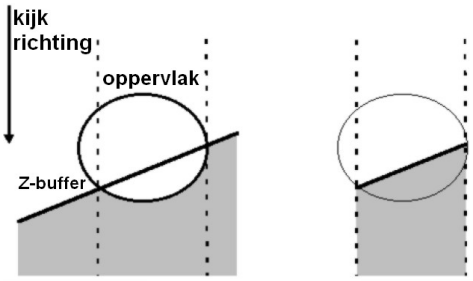
- De logica met pariteiten werkt alleen indien het object geen binnen liggende oppervlakken bevat en ook geen oppervlakken die erbuiten liggen.
- Alle grenzen van het volume moeten *bedekt zijn* door een oppervlak. Het object moet dus gesloten zijn en geen gaten bevatten die gecreëerd zijn door bijvoorbeeld dubbele vertices.
- Een oppervlak mag zichzelf niet intersecteren, geplooid zijn of lussen bevatten.

3.4.7 Clippen van oppervlakken via de dieptebuffer

Nadat elke pixel via de pariteitstest van de getrimde primitieven geclassificeerd is, kunnen de oppervlakken *geclipt worden via de dieptebuffer*. De oppervlakken van de getrimde primitieven kunnen dan uiteindelijk samengesteld worden tot het gerenderd beeld.

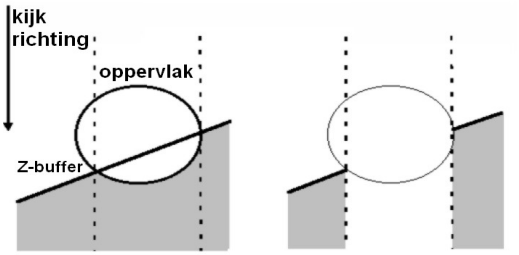
Aangezien een CSG boom in feite enkel bestaat uit intersecties (zie Figuur 13) en subtracties (zie Figuur 14), kan de oppervlaktepariteit van elke primitieve sequentieel getest worden tegenover een andere. De configuratie van de pariteitstest hangt af van de specifieke operatie tussen de primitieven:

- bij een intersectie operatie worden enkel de delen van de dieptebuffer met een *oneven* pariteit behouden en die met even pariteit dus weggeknipt. Dit komt overeen met de waarden die zich qua volume binnen de primitieve bevinden. Andere delen worden leeg gemaakt. (zie §3.4.5)



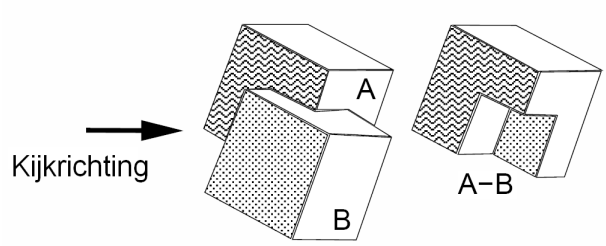
Figuur 13: Dieptebuffer clippen via oppervlakteparitеit bij intersectie.

- Bij de subtractie operatie worden de delen van de dieptebuffer met een *even* pariteit behouden. Plaatsen met oneven pariteit worden leeggemaakt. Het algoritme voor subtractie is dus het *omgekeerde* van dat voor intersectie.



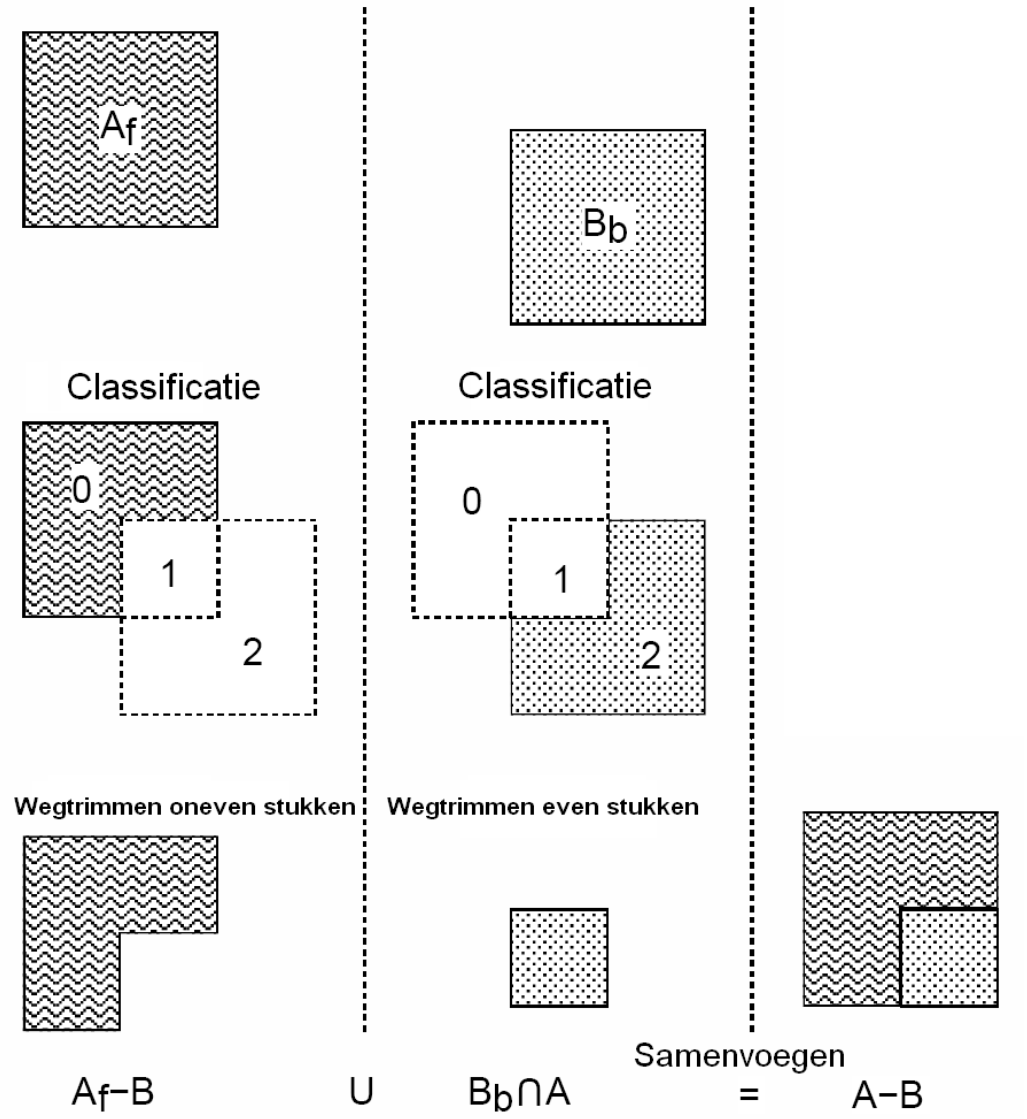
Figuur 14: Dieptebuffer clippen via opervlakteparitеit bij subtractie.

Al deze geclipte primitieven vormen samen het uiteindelijk gerenderd beeld. Het hele rendering proces wordt tenslotte hierbeneden nog eens geillustreerd aan de hand van een het clippen van een subtractie tussen twee primitieven. Wanneer gekeken wordt vanuit de kijkrichting uit Figuur 15, dan krijgen we voor de boom A-B het classificatieproces zoals men kan zien in Figuur 16.



Figuur 15: Voorbeeld van het clippen van een subtractie tussen twee primitieven.

Hier wordt eerst A_f gerenderd, daarna geclas-sificeerd ten opzichte van B en getrimd tot A_f-B. Analoog wordt B_b gerenderd, daarna geclas-sificeerd ten opzichte van A en getrimd tot $B_b \cap A$. Tenslotte worden beide gerenderde beelden samengevoegd.



Figuur 16: Renderen van Figuur 15 als twee partiële produkten

3.4.8 Performantie

Het renderen van het juiste oppervlak van een convexe primitieve is eenvoudig aangezien er per pixel steeds slechts één paar oppervlakken is. De meeste grafische bibliotheken ondersteunen dit onder de namen *front-* en *backface culling*.

Om alle mogelijke oppervlakken van een willekeurige k-convexe primitieve te renderen, hebben we niet één maar $\log_2 k$ bits nodig per pixel. Om dus het i-de naar voren (of naar achteren) gericht oppervlak van een primitieve te renderen, worden dan al de naar voren (of naar achteren) gerichte oppervlakken gerenderd. Dit verhoogt het aantal berekeningen per pixel, maar schrijft enkel data waarvoor dit aantal gelijk is aan i naar de kleurbuffer en de dieptebuffer.

Zij:

a : de snelheid waarmee een polygoon gerasterd kan worden.
b : de snelheid om naar de dieptebuffer te kopiëren.
n : het aantal primitieven in het produkt van de CSG boom
k : de convexiteit van de primitieven

Hier wordt een per aanwezige primitieve één clipping operatie uitgevoerd. De tijd die dit in beslag neemt is **an + b**.

De tijd om een CSG boom uit te tekenen is dan $\mathbf{an^2 + bn}$ wanneer er enkel convexe primitieven zijn en $\mathbf{akn^2 + bkn}$ elders.

Het algoritme heeft dus een $\mathbf{O(n^2)}$.

3.5 Wiegand CSG Rendering

3.5.1 Algemeen

Het Renderingalgoritme van Goldfeather mag dan wel interactieve visuele rendering van CSG objecten mogelijk maken, het vergt nog steeds gespecialiseerde hardware en kan dus niet geïmplementeerd worden via standaard grafische bibliotheken zoals OpenGL. Het gebruik van zulke reeds bestaande grafische bibliotheken *vereenvoudigt* de ontwikkeling, mijdt investering in specifieke grafische hardware en blijft bovendien werken met toekomstige hardware die deze bibliotheken ondersteunt. Een van de mogelijkheden om CSG bomen te visualiseren is via het omzetten naar een B-Rep (zie §3.3.2), maar zulke conversies zijn te traag voor interactieve modellering.

Wiegand CSG rendering past het algoritme van Goldfeather aan waardoor CSG rendering op standaard grafische hardware mogelijk wordt. Waar het algoritme van Goldfeather werkt met zgn. Pixel-Planes, een uitgebreide frame buffer die twee Z-buffers heeft, twee kleurbuffers en bitvlaggen per pixel, maakt Wiegands algoritme enkel gebruik van:

- een kleuren buffer,
- een dieptebuffer,
- een stencil buffer
- en de mogelijkheid om de inhoud van de diepte buffer te bewaren en terug op te halen.

Deze buffers zijn op standaard grafische hardware pertinent aanwezig en maken deel uit van grafische bibliotheken zoals OpenGL.

Wiegand integreert ook verre en nabije clipping vlakken in het algoritme. Het systeem van Wiegand onderhoudt ook volledig geëvalueerde B-rep versies van modellen en herbruikt het render algoritme voor interactieve veranderingen aan de modellen.

3.5.2 Intuïtieve benadering

Het algoritme van Wiegand [8] maakt gebruik van de op standaard grafische hardware voorziene diepte, kleur- en stencil buffers om de partiële producten te renderen, de resultaten uit de hardware terug te halen en samen te voegen in het lokale geheugen van de computer. Het uiteindelijke resultaat kan dan direct teruggestuurd worden naar de frame buffer. Deze benadering lijkt goed, maar maakt geen goed gebruik van de hardware om wille van drie redenen:

- Grafische hardware is tegenwoordig enorm *gepipelined* om de efficientie te verhogen. Om partiële producten op te halen, dient deze pipeline echter onderbroken te worden, wat een nefast effect heeft op de performantie van het renderproces.

- Grafische hardware is meestal geoptimaliseerd voor dataflow vanuit het lokaal geheugen doorheen de pipeline naar de frame buffer. Het terug halen van data uit de framebuffer naar het lokaal geheugen is meestal traag, zeker gezien de hoeveelheid aan gegevens die opgehaald moeten worden in verhouding met de korte instructies die aan de grafische hardware gegeven worden om de primitieven uit te tekenen.

- Het samenvoegen van de partiële producten tot één geheel gebeurt in het lokaal geheugen en maakt helemaal geen gebruik van hardware ondersteuning.

Wiegand CSG rendering gaat deze problemen tegen door het verkeer tussen de grafische hardware en het lokaal geheugen zo klein mogelijk te houden.

3.5.3 Geoptimaliseerde benadering

Net als Goldfeather verdeelt Wiegand hier het probleem in twee stappen: *classificatie* en het uiteindelijke *renderen*. Voordat het renderen begint, wordt de huidige inhoud van de dieptebuffer bewaard in het lokaal geheugen. Daarna wordt elk partieel product om beurt geclassificeerd. Per oppervlak wordt gebruik gemaakt van een *extra stencil buffer bit*, die de resultaten van de classificatie opslaat. Deze noemt men de ***accumulator bit***. Tijdens dit classificatieproces zijn updates naar de kleurenbuffer *niet toegelaten*.

Tenslotte wordt elk oppervlak van het partieel product opnieuw gerenderd door de opgeslagen resultaten als een ***masker*** te gebruiken om updates naar de framebuffer te controlleren. *Op hetzelfde moment* wordt de dieptebuffer gebruikt om de pixels die voor de stencil test geslaagd zijn samen te stellen met deze die reeds gerenderd zijn.

Het aantal vlakken waarvoor classificatie uitgevoerd kan worden, is *beperkt door de diepte van de stencil buffer*. Indien de capaciteit van de stencil buffer overschreden wordt, moeten de vlakken in verschillende stappen verwerkt worden. Hierbij wordt de diepte buffer tijdens elke stap bewaard en terug ingeladen.

Wiegand verkleint de hoeveelheid aan data die op deze manier gekopieerd moet worden door tijdens elke stap enkel die delen van de dieptebuffer te bewaren die gewijzigd zullen worden tijdens het classificatieproces.

Zo moet er in de eerste stap van elk frame helemaal geen dieptebuffer bewaard worden aangezien alle waarden bekend zijn; de frame buffer is immers leeg. Voor eenvoudige modellen die aan het begin van een frame gerenderd worden, moet dus al helemaal niets in de framebuffer bewaard of ingeladen worden.

Binnen een genormaliseerde CSG boom kan een vlak in meer dan één partieel product voorkomen. Wiegand buit dit gegeven uit door dezelfde accumulator bit te gebruiken voor *alle partiële producten* met hetzelfde vlak. De resultaten van de classificatie worden via een OR-operatie vermengd met de huidige inhoud van de accumulator.

De stencil bits worden dus per vlak opgedeeld in :

- $\mathbf{S_{count,}}$ de *count bits*
- $\mathbf{S_{p,}}$ een *pariteitsbit*
- $\mathbf{S_{a,}}$ een *accumulator* bit

Hierdoor zijn er dus $\mathbf{\log_2 k}$ count bits nodig waarbij k de maximale convexiteit is van alle primitieven vaarvan het oppervlak geclassificeerd wordt in de huidige stap. De count- en pariteitsbits worden onafhankelijk gebruikt en kunnen overlappen. Er dienen minstens twee bits aanwezig te zijn in de stencil buffer om één enkel oppervlak met 1-convexe en 2-convexe primitieven de classificeren en te renderen in één stap.

3.5.4 Classificatie per groep

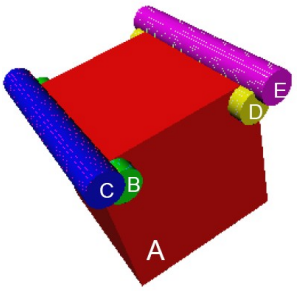
Partiële producten worden opgedeeld in groepen zodat, *naargelang de diepte van de stencil buffer*, alle partiële producten in een groep geclassificeerd en gerenderd kunnen worden in één stap.

De *capaciteit* van een groep wordt gedefinieerd als het *aantal verschillende doeloppervlakken* waaruit de partiële producten in de groep mogen bestaan.

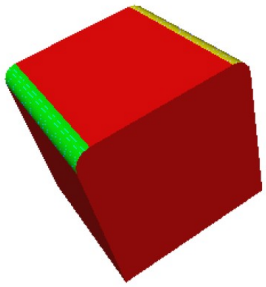
Deze capaciteit is dus *afhankelijk van :*
- *de diepte van de stencil buffer* en
- de maximale convexiteit van de doelprimitieven in de groep.

Deze groepen worden gevormd door partiële producten toe te voegen in stijgende volgorde van convexiteit van de doelprimitieve. Eens een partieel product met een bepaald doeloppervlak toegevoegd is, kunnen alle andere partiële producten met hetzelfde doeloppervlak hieraan toegevoegd worden zonder dat deze extra capaciteit gebruiken. Het toevoegen van een partieel product met een hogere convexiteit dan degenen die al in de groep zitten, zal de capaciteit van de groep verkleinen. Indien er *onvoldoende capaciteit* is een overgebleven product met een minimale convexiteit toe te voegen, dient een nieuwe groep gestart te worden.

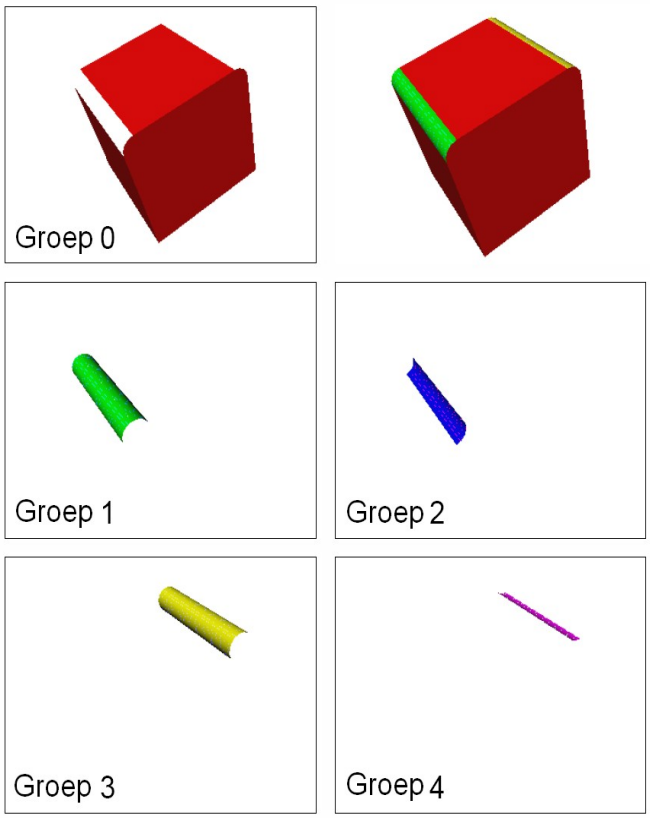
Elke groep wordt verwerkt in een *aparte stap* waarin alle doelprimitieven eerst geclassificeerd worden en dan gerenderd. Operaties op de volledige framebuffer worden beperkt tot gebieden die gedefinieerd worden door de projectie van de bounding box van de huidige groep of het partiële product.



Figuur 18: Primitieven A,B,C,D,E en F



Figuur 19: Resultaat van het product (A∩BUA-C) ∩ (A∩D UA-E).



Figuur 17: De vijf groepen apart gerenderd

Stel dat we de vijf primitieven uit Figuur 18 willen renderen tot Figuur 19 via het product **((A∩B) U (A-C)) ∩ ((A∩D) U (A-E))**.

Dit wordt dan eerst genormaliseerd tot **(A∩B∩D) U (A∩D-C) U (A-C-E)**.

De methode van Wiegand deelt dan dit product op in partiële producten en groepeert ze. Er ontstaan vijf groepen:

0 : (A_f∩B∩D) U (A_f∩D-C) U (A_f∩B-E) U (A_f-C-E)

1 : (B_f∩A∩D) U (B_f∩A-E)

2 : (C_b∩A∩D) U (C_f∩A-E)

3 : (D_f∩A∩B) U (D_f∩A-C)

4 : (E_b∩A∩B) U (E_f∩A-C)

In Figuur 17 zien we het resultaat van elke apart gerenderde groep om tot het resultaat van Figuur 19 te komen. Hierbij valt het op dat groepen 2 en 4 niet zichtbaar zijn in het uiteindelijke resultaat omdat ze zich achter de oppervlakken van groepen 1 en 3 bevinden.

3.5.5 Clippen van vlakken en halfruimtes

Om solide modellen interactief te inspecteren, kan een beroep worden gedaan op het clippen van vlakken die kunnen helpen de interne structuur te onthullen. Na *definitie* en *activatie* van een geclipt vlak, wordt alle vervolgens gerenderde geometrie tegen dat vlak geclipt en worden de buitenzijden weggelaten. Wanneer we solide objecten renderen als *gesloten omhullenden*, dan zal clippen het binnenste van een omhulsel tonen wanneer een deel van dit omhulsel reeds weggeclipt is. Dit is uiteraard niet de bedoeling.

Rossignac en co. [30] beschrijven een algoritme om deze omhulsels overal waar ze het te clippen vlak intersecteren '*af te dekken*'. Dit algoritme maakt gebruik van de stencil buffer en brengt bovendien ook intersecties tussen solide objecten aan het licht op het te clippen vlak.

Wiegand gaat nu uit van het principe dat het eerst clippen en dan afdekken van een solied object *equivalent* is met het intersecteren van een *halfruimte*. Een halfruimte wordt gedefinieerd als *het oneindig ruimtelijk volume dat zich aan één kant van een gegeven vlak bevindt*. Door deze equivalentie kan het clippen en afdekken van een solied object dus gebeuren door *een intersectie tussen een solied object **S** en een halfruimte **H*** te renderen.

Dit kan door het renderen van een convexe, *polygonale* primitieve **P**. Deze primitieve P heeft dan :

- één zijde die *op* het vlak **H** ligt
- randen die de bounding box van **S** *niet* intersecteren. De andere zijden van P zouden S helemaal *niet* moeten intersecteren.

Het renderen van **S∩P** is dus equivalent met het renderen van het solied object dat gedefinieerd wordt door **S∩H**.

Het afdekalgoritme van Rossignac kan dus eenvoudigweg geïntegreerd worden in de methode van Wiegand. Hierdoor wordt het mogelijk om gebruik te maken van te clippen hulpvlakken bij het renderen van CSG bomen die halfruimten gebruiken.

Aangezien een halfruimte *oneindig* is, nemen we aan dat het in om het even welke CSG uitdrukking steeds geïntersecteerd wordt met een eindige primitieve.

We stellen trouwens vast dat **S-H** equivalent is met **S∩(¬H)** waarbij **¬H** het complement is van de normale van het vlak dat de halfruimte definieert.

Tijdens het renderen van de doelprimitieve, gedraagt de halfruimte zich dan als een *getrimde primitieve* door een voor deze halfruimte te clippen vlak te *activeren*. De stencil buffer wordt hierbij niet gebruikt.

De verzameling halfruimtes in een product kan beschouwd worden als een 1-convexe doelprimitieve. Zijn vlak kan dan gerenderd worden door het gedefinieerd vlak van elke halfruimte te renderen, of eigenlijk een polygon die op het vlak ligt en groot genoeg is.

Intussen zijn de te clippen vlakken actief voor alle andere halfruimten. Wanneer het te clippen vlak gerenderd wordt, wordt het *tijdelijk gedeactiveerd* om te voorkomen dat het zichzelf clipt.

Deze benadering heeft drie voordelen ten opzichte van het renderen van halfruimtes als normale primitieven :

- De groep halfruimten moet enkel gerenderd worden als een doelprimitieve. Al het trimmen via halfruimtes maakt gebruik van de te clippen vlakken.
- Elke doelprimitieve wordt geclipt, waardoor de hoeveelheid naar de framebuffer geschreven gegevens *verkleind wordt ten koste van het extra verwerken van gegevens die nodig is om te clippen*.
- Een intersectie tussen een solied object en een halfruimte kan correct gerenderd worden door gebruik te maken van het algoritme voor 1-convexe solide objecten, onafhankelijk van de eigenlijke convexiteit van de primitieve.

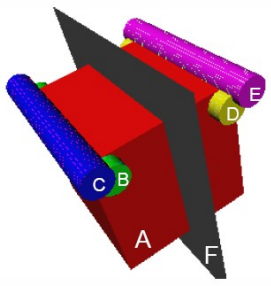
3.5.6 Clippen naar nabije en verre vlakken

Bovenop de door het gebruikersstandpunt gedefinieerde clipping vlakken, clipt de methode van Wiegand CSG objecten ook nog eens naar *"nabije"* en *"verre"* vlakken. Deze vlakken liggen loodrecht op de kijkrichting. Alle geometrie *moet verder van de gebruikerspositie liggen* dan het *nabije oppervlak* en *korterbij* dan het *verre vlak*. De nabije en verre clipping vlakken definiëren ook de indeling van de afstanden van waarden die in de dieptebuffer opgeslagen zijn naargelang de positie van de gebruiker. Zo worden punten op het *nabije* vlak ingedeeld via een *minimum dieptebufferwaarde* en punten op het *verre* oppervlak via een *maximum dieptewaarde*. Dit algoritme zal trouwens falen wanneer geen enkele primitieve geclipt wordt door zowel het nabije als het verre clipping vlak.

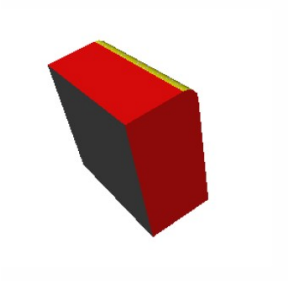
In praktijk kan het verre clipping vlak steeds *veilig* van de primitieven af gepositioneerd worden. Het nabije oppervlak geeft meer problemen:

- het kan *niet gepositioneerd worden achter het gebruikersstandpunt*
- de resolutie van de dieptebuffer is absoluut *afhankelijk van de positie van het nabije clipping vlak*.

Het nabije clipping oppervlak zou dus zo ver mogelijk van het gebruikersstandpunt moeten liggen. Beschouw hiervoor bijvoorbeeld het renderen van A-B waarbij het gebruikersstandpunt zich bevindt in een gat van A dat gevormd werd door B te subtraheren. Hier is clipping van het nabije oppervlak *onvermijdelijk*. Het algoritme kan daarom uitgebreid worden door de getrimde primitieven af te dekken indien ze onderhevig zijn aan clipping via het nabije vlak. Clippen van de doelprimitieven is geen probleem tenzij het gebruikersstandpunt zich midden in een CSG boom bevindt.



Figuur 20: Primitieven A tot E plus clipping vlak F



Figuur 21: Resultaat van het product (A∩BU-C) ∩ (A∩DUA-E)∩F

Wanneer het S_p omswitcht wordt de getrimde primitieve twee maal gerenderd: eerst met de dieptebuffertest *uitgeschakeld*, daarna met de dieptebuffertest *ingeschakeld*. De eerste renderstap stelt de *pariteitsbit* in op de plaatsen waar er afgedekt moet worden. De tweede renderstap *vervolledigt* de classificatie.

Een illustratie van dit algoritme, kan door Figuur 21 te observeren. Figuur 21 laat een CSG object zien dat bestaat uit de primitieven van figuur 18, maar dan geïntersecteerd met één enkel geclipt oppervlak of halfruimte F (zie Figuur 20). De genormaliseerde CSG omschrijving is (A∩B∩D∩F) U (A∩D∩F-C) U (A∩B∩F-C) U (A∩F-C-E).

3.5.7 Uitbreiding van Normalisatie en Snoeien

Voor Wiegand CSG rendering moeten de normalisatie- en snoeialgoritmes uitgebreid worden om te kunnen omgaan met halfruimte primitieven. Deze extensies bestaan uit *bijkomende productie regels* voor het genereren van bounding boxen, normalisatie en snoeien. Stel hierbij H voor als een halfruimte.

Genereren van Bounding Boxen:

1. Bound (A∩H) = Bound (A)

Normalisatie :

0. X - (YUZ) --> (X-Y) - Z

Snoeien :

4. (A∩H) --> Ø indien Bound (A) buiten H ligt
5. (A∩H) --> A indien Bound (A) binnen H ligt
6. (A∩H)-B --> A∩H indien Bound (B) buiten H ligt
7. (A_b∩H) --> A_b indien Bound (A) binnen H ligt
8. (H_f∩A) --> H_f indien Bound (A) H niet intersecteert

Wanneer we deze nieuwe regels bijgevolg toepassen op het voorbeeld uit Figuur 21, dan bekomen we de genormaliseerde en geometrisch gesnoeide CSG boom (A∩D∩F) U (A∩F-E). Uitgebreid naar partiële producten geeft dit (A_f∩D∩F) U (D_f∩A∩F) U (F_f∩A∩D) U (A_f∩F-E) U (F_f∩A-E) U (E_b∩A∩F).

Tenslotte zal subtractie snoeien dit nog herleiden tot (A_f∩D∩F) U (D_f∩A∩F) U (F_f∩A∩D) U (A_f∩F-E) U (F_f∩A) U (E_b∩A). Er worden in elk frame ook producten gesnoeid tegenover het vanuit het gebruikersstandpunt zichtbaar volume. Daarna worden dan de bounding boxen van de getrimde primitieve geclassificeerd ten opzichte van het nabije clipping oppervlak om te bepalen of een extra afdekstap nodig is.

3.5.8 Performantie

De performantie van Wiegand CSG rendering hangt af van het aantal uitgevoerde operaties.

Zij:

n : het aantal primitieven in het produkt van de CSG boom
k : de convexiteit van de primitieven

Dan heeft het render algoritme een tijdscomplexiteit van **O(kj²)** en is dus vergelijkbaar met de performantie van Goldfeather's renderalgoritme.

3.5.9 Voor- en nadelen

Hét voordeel van Wiegand CSG rendering, en ook de bestaansreden van deze methode is dat het geïmplementeerd kan worden op *bestaande grafische hardware,* inclusief de hardwarematige dieptebuffer. Spijtig genoeg is het algoritme afhankelijk van de snelheid waarmee de dieptebuffer gekopieerd wordt. Dit kopiëren is in de meeste grafische hardware niet geoptimaliseerd en vormt dus de '*bottleneck*' van Wiegand's methode.

3.6 Steward en Leach CSG Rendering

3.6.1 Algemeen

Het CSG Renderalgoritme van Steward en Leach [4] reduceert het aantal stappen ten opzichte van het Wiegand CSG Renderalgoritme door *meer dan één primitieve tegelijk te clippen*.

Hiervoor wordt gebruik gemaakt van de *diepte complexiteit*. De diepte complexiteit is het *aantal primitieven dat zich bevindt in de dieptebuffer van elke pixel*.

We definiëren **κ** als zijnde de *maximale diepte complexiteit* van een pixel vanuit een specifieke kijkrichting. **κ** komt dan overeen met het aantal *lagen* waarop geclipt moet worden in plaats van het aantal primitieven CSG boom **n**, dat uiteraard hoger ligt.

Het eerste wat we hiervoor moeten bepalen is **κ**. Hiervoor volstaat het om voor elke pixel *het aantal primitieven te bepalen*, op te slaan in de stencil buffer en het grootste stencil buffer waarde te zoeken.

Dit algoritme levert performantiewinst op naar gelang de grootte van **κ**, en dus ook naargelang de kijkrichting. In het slechtste geval is er dus geen winst.

Hierna kan er via een stencil test voor gezorgd worden dat enkel de **n**'de laag van een reeks primitieven wordt uitgetekend. Deze test *verhoogt een waarde in de stencil buffer voor elke pixel die door deze primitieve wordt bedekt*. Deze primitieve wordt enkel uitgetekend in de dieptebuffer indien deze waarde gelijk is aan het nummer van de gevraagde laag.

Voor de implementatie met behulp van OpenGL dient opgemerkt te worden dat de specifieke oppervlakken in elke laag bepaald worden door de volgorde waarin primitieven in OpenGL voorgesteld worden. Zo zal de eerste laag bestaan uit het oppervlak van de eerste primitieve en delen van de tweede primitieve die de eerste primitieve niet overlappen, etc...

Er kan nu per laag geclipt worden. Hierin wijkt Steward & Leach rendering duidelijk af van de vorige methodes: waar Wiegand het clippen van oppervlakken uit de dieptebuffer ten opzichte van de overeenkomende primitieve vermeed, kan dit hier wel.

Er dient echter wel rekening gehouden te worden met twee specifieke voorwaarden:

- Oppervlakken van een te *intersecteren* primitieve die naar *vóór* wijzen, moeten aangeduid worden als zijnde *binnen* de primitieve.
- Oppervlakken van een te *subtraheren* primitieve die naar *achter* wijzen, moeten aangeduid worden als zijnde *buiten* de primitieve.

Om te voldoen aan deze voorwaarden volstaat het de pariteitsbit correct in te stellen en een dieptebuffertest uit te voeren. Dit werkt, ook al hangen ze af van de driehoekjes die over verschillende stappen op exact dezelfde dieptebuffer waarden gerasterd worden.

3.6.2 Performantie

Zij:

a : de snelheid waarmee een polygoon gerasterd kan worden.
b : de snelheid om naar de dieptebuffer te kopiëren.

Hier wordt een niet per aanwezige primitieve **n** één clipping operatie uitgevoerd, maar slechts voor het aantal lagen van de dieptebuffer **κ**.

Polygonen moeten nu wel twee keer gerasterd worden: een keer om de lagen te clippen en een keer om ze te kopiëren. De tijd die dit in beslag neemt is **2an + b**.

De tijd om een CSG boom uit te tekenen is dan **aκn + bκ**. Dit verschil is belangrijk [4] omdat meestal **κ < n**. Deze diepte complexiteit **κ** hangt immers af van de kijkrichting en de plaats van de specifieke primitieven in de boom. Het visueel renderalgoritme van Steward en Leach is dus zeker beter dan **O (n²)**.

Er dient opgemerkt te worden dat in het slechtste geval, wanneer **κ = n**, er *geen enkele snelheidswinst meer overblijft*. Sterker nog: in dat geval zal het Wiegand rendering beter werken aangezien daar slechts één **a** – operatie nodig is.

3.7 Verdere verfijningen

De laatste jaren zijn er nog verdere verfijningen geweest aan deze render algoritmes. Zo verbetert het *geSequenteerde Convexe Subtractie* algoritme (SCS) van Steward & Leach uit 2000 de performantie van visueel CSG renderen door het kopiëren van de dieptebuffer, een serieuze bottleneck in moderne grafische hardware, te reduceren [31]. Zoals de naam van het algoritme al zegt, geldt hier wel dat de primitieven convex moeten zijn.

Het SCS algoritme maximaliseert de performantie door meer gebruik te maken van snelle polygoon rasterisatie. Dit is een *snel pad* en dus geoptimaliseerd om zijn taak snel uit te voeren, dit in tegenstelling tot kopiëren van dieptebuffers, die een traag pad vormen.

In 2002 werden de intersectie algoritmen van SCS door Steward en Leach nog eens verfijnd om intersecties van convexe objecten mogelijk te maken in een tijdscomplexiteit van **O(n)** voor CSG bomen met convexe objecten van de vorm A∩B∩C∩... [32]

In 2003 werd ook een snellere benadering gevonden die subtracties van grote hoeveelheden convexe objecten mogelijk maakt [33]. In het beste geval kunnen er, naargelang de ligging van de deelobjecten, subtracties van tijdscomplexiteit **O(n)** plaatsvinden. Hiervoor wordt gebruik gemaakt van een zgn. overlappingsgraaf. Deze graaf slaat informatie over ruimtelijke intersecties op en helpt zo subtracties te versnellen.

Gedetailleerde besprekingen van deze verdere verfijningen vindt men terug in de respectievelijke research papers.

3.8 Conclusie

Dit hoofdstuk gaf een overzicht van de verschillende methodes om CSG bomen visueel te renderen. Deze methodes werden opgedeeld in twee categorieën naargelang de vorm van classificatie. De oppervlak classificatie algoritmes (§3.2) omvatten onderandere conversie naar een B-rep of Spatiale Enumeratie.

Daarna werden algoritmes bekeken die gebaseerd zijn op punt classificatie (§3.3). Aangezien de algoritmes die geschikt zijn voor interactieve visuele CSG rendering allemaal gebruik maken van punt classificatie, worden deze apart besproken in de volgende paragrafen.

Het *Goldfeather render algoritme* (§3.4) is de eerste methode die interactief renderen mogelijk maakt, zij het nog op gespecialiseerde hardware. We zagen dat deze methode gebruik maakt van het feit dat een CSG boom niet uniek is. Daardoor kan een CSG boom eerst omgezet worden in een *genormaliseerde vorm* (§3.4.3) alvorens gerenderd te worden.

Hierna werd het a*lgoritme van Wiegand* (§3.5) uitgebreid besproken. Dit algoritme is zeer belangrijk omdat het op standaard grafische hardware geïmplementeerd kan worden. Dankzij dit algoritme is het dus mogelijk om CSG modelleerapplicaties op elke huiscomputer te draaien. Het spreekt voor

zich dat CSG zijn populariteit voor een stuk aan dit algoritme te danken heeft. Verder wordt dit algoritme nog uitgebreid met een reeks verfijningen (§3.6 en §3.7), bedacht door het team van *Steward en Leach.* Deze algoritmes maken het eerst mogelijk om meer dan één primitieve tegelijk te clippen. Er komt een nieuwe rendermethode genaamd ge*Sequenteerde Convexe Subtractie* (CSC) en deze wordt nog verder verfijnd zodat zowel intersectie als subtractie in lineaire tijd verwerkt kunnen worden.

Dankzij deze ontwikkelingen is het mogelijk om via CSG bomen primitieve objecten te gaan combineren en bewerken. Er is bijgevolg nood aan algoritmes die een gebruiker de mogelijkheid geeft deze CSG objecten interactief te modelleren. Hierover meer in Hoofdstuk 4.

H4. Modelleren van CSG bomen

4.1 Inleiding

Constructieve Solide Geometrieën zijn pas echt van nu indien deze ook grafisch gemodelleerd kunnen worden. Dit biedt nieuwe perspectieven voor iedereen die bezig is met constructie van voorwerpen. Zo kan een designer een gat, gedefinieerd door een complex solide object, in een bestaand object boren en de nieuw ontstaande vorm observeren en verder bewerken.

In dit hoofdstuk gaan we na wat bij het modelleren van CSG bomen zoal van belang is (§4.2). Hierbij wordt al snel duidelijk dat interactieve modelleeromgevingen intuïtiever en gemakkelijker zijn dan conventionele. We overlopen verder enkele bekende applicaties die gebruik maken zowel expliciete als impliciete modelling (§4.3). Daarna wordt er bekenen hoe zo'n modelleeractie verloopt (§4.4). Hierbij wordt het accent gelegd op de manier waarop een CSG object gewijzigd kan worden. Tenslotte worden enkele voordelen opgesomd (§4.5) van modelleren via CSG objecten ten opzichte van de standaard B-rep objecten.

4.2 Algemeen

Bij conventionele (niet-interactieve) modelleersystemen worden primitieven eerst gepositioneerd. Daarna wordt hierop een Booleaanse operatie toegepast en worden de resultaten gerenderd. Vaak is het echter zeer moeilijk om de juiste positie van een primitieve te bepalen wanneer men op dat moment enkel de andere primitieven ziet, en niet het volledige CSG object. In zulke situaties dient men gebruik te maken van trial-and-error methodes, of erger nog het modelleerproces te onderbreken om de correcte positie van zo'n primitieve op een numerieke wijze te berekenen.

Door CSG bomen interactief te renderen (zie hoofdstuk 3) is het mogelijk modelleeromgevingen te creëren waarin designers op een eenvoudige manier de mogelijkheden van het CSG concept kunnen verkennen. Zo kan een designer bijvoorbeeld via een tool (in dit geval een *virtuele boor*) een gat maken in een complex solied object en meteen kijken welke nieuwe vorm eruit ontstaat.

4.2.1 Vereisten

Visueel interactief modelleren van CSG bomen veronderstelt een performante en krachtige rendering. Dit modelleren houdt in dat de gebruiker het CSG object niet alleen een snelle manier langs alle mogelijke kanten moet kunnen bekijken, maar dit ook interactief moet wijzigen. Telkens er op een bestaand CSG object een bewerking doorgevoerd wordt, moet dit resultaat vrijwel onmiddellijk te zien zijn.

4.2.2 Performantie

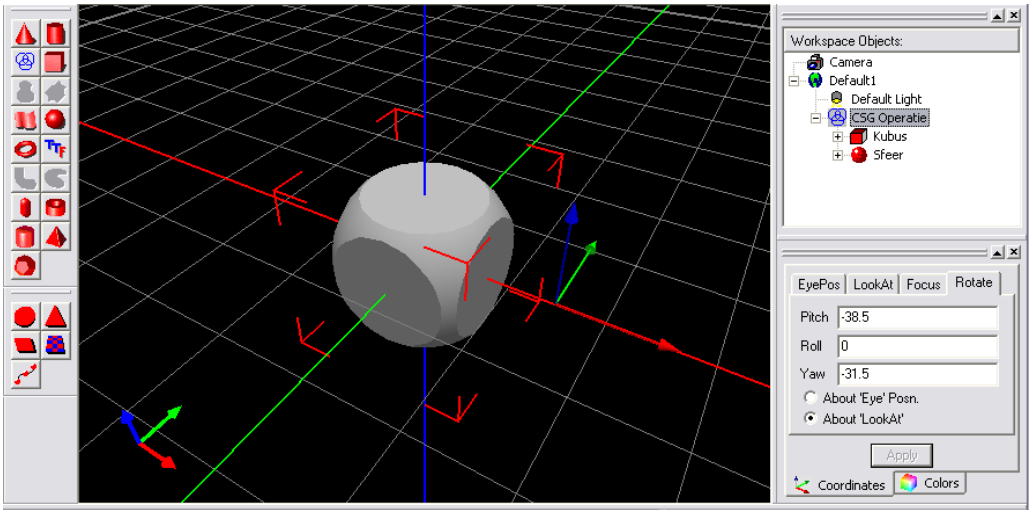
Qua performantie werken de visuele renderalgoritmes van Wiegand en Steward & Leach gelukkig behoorlijk goed wanneer ze gebruikt worden door een interactieve modelleeromgeving. Er is hier meestal sprake van slechts één zogenaamd *dynamisch gerenderd* CSG object. Dit is meestal een eenvoudige convexe "tool", meestal in de vorm van een primitieve, of een meer complex hulpmiddel.

4.3 Soorten CSG Modelling

Zoals we reeds eerder gezien hebben (§2.3.3), wordt CSG modelling vooral gebruikt in CAD en andere modelleerapplicaties zoals ModelMagic3D (zie Figuur 22). Er zijn twee benaderingen om objecten die voorgesteld worden via CSG bomen interactief te modelleren: expliciet en impliciet.

4.3.1 Expliciet modelleren

Bij expliciet moduleren wordt CSG niet op een verdoken manier gebruikt, maar laat een de applicatie duidelijk zien dat ze met CSG bezig is. De gebruiker krijgt bij deze benadering de mogelijkheid om de verschillende Booleaanse operaties op de bestaande objecten toe te passen en zo een CSG object aan te maken. De hele boomstructuur van het CSG object wordt in een apart paneel getoond en kan ook als dusdanig bewerkt worden, zoals in ModelMagic3D (zie Figuur 22). Deze methode is vaak niet interactief, maar er

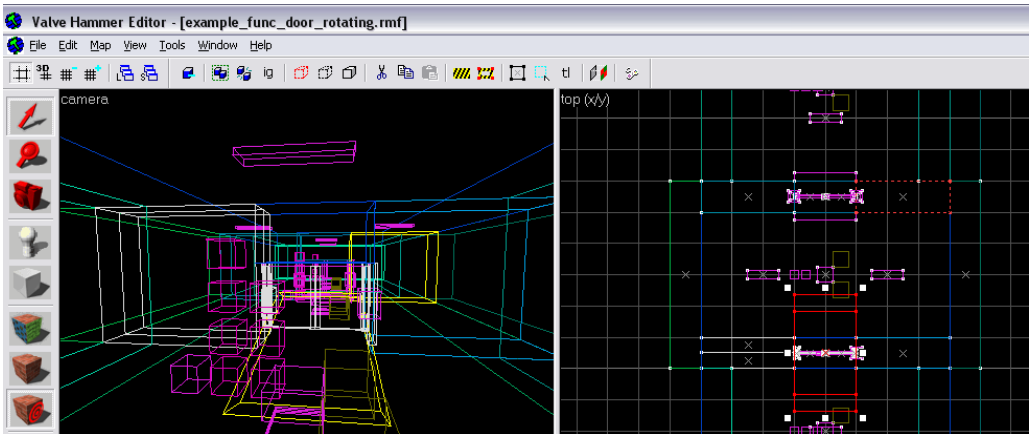


Figuur 22: Imageware ModelMagic3D, een 3D modelling applicatie die gebruik maakt can CSG bomen [35].

kunnen wel primitieven en deelobjecten op een interactieve manier verplaatst worden wanneer ze al deel uitmaken van de CSG boomstructuur.

4.3.2 Impliciet modelleren

Bij impliciet modelleren merkt de gebruiker niet dat de modelleeromgeving gebruik maakt van CSG. Objecten kunnen interactief gekerfd of op een andere manier bewerkt worden via bepaalde virtuele tools. Er kunnen ook objecten interactief samengesmolten, toegevoegd of ingepast worden. Deze methode is intuïtiever en vergt geen kennis van de CSG structuur. Een voorbeeld van een applicatie die CSG op een verdoken manier gebruikt om bestaande objecten in stukken te snijden of erin te kerven is de Hammer Editor. Dit is een modelleerapplicatie van Valve die gebruikt wordt bij het creëren van game levels (zie Figuur 23) [35].



Figuur 23: Valve's Hammer Editor maakt van CSG als deel van de pre-processing stap bij creatie van Game Levels

4.4 Verloop

Een modelleer actie verloopt in verschillende stappen. Het begint met het selecteren van het soort operatie en de tool. Deze gaat interageren met het bestaande CSG object. Dit leidt tot wijzigingen van de CSG boom. Daarna kan de gebruiker een nieuwe wijziging aanbrengen.

4.4.1 Tools en interactie

De belangrijkste bewerkingen die een gebruiker op een CSG object wil uitvoeren zijn:

- het bijvoegen van primitieven aan het CSG object
- het samenvoegen van reeds bestaande CSG objecten
- het bewerken van een reeds bestaand object door er een stuk uit te boren/snijden/kerven.

Er kunnen heel wat tools bedacht worden om CSG te wijzigen. Deze tools zijn doorgaans steeds zelf een CSG object. Voorbeelden zijn:

- beitel, hamer, mes, boor, ... Deze kerven of verwijderen delen van een bestaand CSG object.
- ingeladen object dit is meer om te combineren met andere objecten. Of om gewoon aan een bestaand CSG object te *plakken*.

De interactie bij CSG modelleren bestaat uit het correct positioneren van de tool *bij* of *in* het reeds bestaande object. Deze interactie moet precies uitgevoerd worden, anders zullen de wijzigingen in de boom fout zijn en kan dat gedeelte opnieuw gemodelleerd worden. In een goede modelleeromgeving dienen zulke interacties op een intuïtieve manier te gebeuren.

4.4.2 Wijzigen van de CSG boom

Indien de CSG boom een bewerking onderging, moet de boomstructuur gewijzigd worden. Er zijn hier uiteindelijk vier mogelijkheden:

- aan het bestaande CSG object wordt een ander object toegevoegd,
- het bestaande CSG object wordt geïntersecteerd met een ander object
- het ander object wordt gesubtraheerd van het bestaande CSG object
- het bestaande CSG object wordt gesubtraheerd van het ander object

Het spreekt voor zich dat dit ander object, hetzij een nieuw ingeladen object, hetzij een bestaande tool, ook een CSG object is. Dan kunnen combinaties van het bestaande CSG object en het andere object vrijwel onmiddelijk een nieuwe boom vormen. Zij $\mathbf{B_o}$ het reeds bestaande CSG object en $\mathbf{B_t}$ de tool of het bijgevoegd object, dan is het nieuwe object $\mathbf{B_n}$ respectievelijk gelijk aan:

- $\mathbf{B_n} = \mathbf{B_o} \cup \mathbf{B_t}$
- $\mathbf{B_n} = \mathbf{B_o} \cap \mathbf{B_t}$
- $\mathbf{B_n} = \mathbf{B_o} - \mathbf{B_t}$
- $\mathbf{B_n} = \mathbf{B_t} - \mathbf{B_o}$

We krijgen dan opnieuw een geldige CSG boom die deze nieuwe wijziging in rekening genomen heeft. Deze 'nieuwe' CSG boom moet dan opnieuw visueel gerenderd worden. Het zal hiervoor ook noodzakelijk zijn de CSG boom opnieuw te normaliseren en te *snoeien (prunen)* aangezien deze nieuwe bomen zich niet automatisch in een genormaliseerde toestand bevinden.

4.5 Voordelen van modelleren met CSG

CSG modelling wordt vaak gebruikt in CAD design omdat het enkele belangrijke voordelen heeft ten opzichte van B-reps [34]:

- Het modelleren via primitieven en Booleaanse operaties is veel intuïtiever dan het direct specifiëren van B-rep oppervlakken, tenzij in situaties waar de gegeven primitieven niet toepasbaar zijn op het te modelleren object.
- De primitieven kunnen geparameteriseerd worden waardoor ze herbruikt kunnen worden en verzameld in voorgedefiniëerde bibliotheken.
- De primitieven kunnen geassocieerd worden met andere, bijkomende informatie.
- De CSG bom bevat impliciete informatie die voor verschillende doelen gebruikt kan worden.

- De primitieven zelf verzekeren dat het de oppervlakken van het model een correcte topologie heeft. Men moet dus niet omgaan met een aantal verwarrend beperkingen aangezien deze impliciet zijn.

4.6 Conclusie

Dit hoofdstuk concentreerde zich op het modelleren van CSG objecten. Eerst werd een algemeen overzicht gegeven (§4.2), daarna werden enkele concrete voorbeelden gegeven van bestaande CSG modelleer applicaties. Hier werd een verschil gemaakt tussen interactieve en niet-interactieve modellers, en tussen impliciete en expliciete CSG modellers (§4.3).

Paragraaf 4.4 gaf een overzicht van het verloop van een modelleeractie en de wijzigingen die als gevolg van deze actie bij de CSG boom aangebracht moesten worden. Tenslotte werden een aantal voordelen van modelleren via CSG objecten ten opzichte van de standaard B-rep objecten opgesomd (§4.5).

H5. Haptics

5.1 Algemene introductie

Met haptics bedoelt men het voelen en besturen van objecten via de tastzin. Deze term wordt sinds de jaren tachtig ook gebruikt om machines en interactietechnieken te beschrijven die zich bezig houden met het realiseren van interactie tussen de mens en een elektronisch toestel via de tastzin.

Haptics in de informatica kunnen dan ook omschreven worden als "***alle aspecten van informatievergaring en object manipulatie via de tastzin door mens, machines of een combinatie van de twee in reële, virtuele of van op afstand bediende omgevingen.***"[13] In deze context zijn haptics tegenwoordig in de hele wereld een substantieel onderwerp van onderzoek en ontwikkeling geworden.

5.1.1 Indeling

Om tegemoet te komen aan deze steeds groter wordende tak van de wetenschap, kan men dit onderzoeksonderwerp indelen in drie gebieden [13]:

1. *human* haptics, of de studie van het menselijke voelen en manipuleren via de tastzin (zie §5.2).

2. *machine haptics*, de ontwikkeling, constructie en het gebruik van machines die de menselijke tastzin vervangen of verhogen. Deze machines worden *haptische interfaces* genoemd en komen verder in dit hoofdstuk (§5.4) aan bod. Daarbij passeren de voornaamste interfaces de revue. Verder wordt een overzicht gegeven van de haptische interactiecyclus die human en machine haptics samen brengt.

3. *computer haptics*, algoritmes en software om virtuele objecten te genereren en te renderen. Dit aspect van haptics komt aan bod in de paragrafen over *haptisch renderen* (§5.5)

Deze laatste onderverdeling vormt het verband tussen haptics en *informatica*, maar ook andere gebieden van de wetenschap houden zich bezig met haptics waaronder de biomechanica, neurowetenschappen, psychofysica, robotica, mathematische modellering, simulaties enz...

De aandacht die tegenwoordig aan haptics besteed wordt zullen in combinatie met de revolutie in de informatica leiden tot een steeds breder wordende waaier van toepassingen, van medische tot artistieke doeleinden (zie §5.6).

5.1.2 Haptics en Virtuele omgevingen

Virtuele omgevingen (beter bekend als *virtual reality*) zijn door de computer gegenereerde omgevingen waarin de gebruiker kan interageren met dingen die hij waarneemt. Gewoonlijk maken virtuele omgevingen enkel gebruik van *visuele* en *auditieve* mogelijkheden, waardoor de mogelijkheid tot interactie met de gebruiker zich beperkt.

Net zoals in onze interactie met de reële wereld, zal het invoeren van een *haptische component* niet alleen het voelen en aanraken van objecten aan een virtuele omgeving toevoegen; er wordt ook verwacht dat er een mogelijkheid is tot manipulatie van deze objecten wordt verwacht. In tegenstelling tot beeld en geluid is er bij de haptische component dus *zowel input* als *output*.

Figuur 24: Tastbare interactie met een virtuele omgeving [38].

5.2 Haptics bij de mens

Het menselijk gevoel bestaat uit een brede waaier van gevoelssensoren en zenuwen die eindigen in de huid, gewrichten, pezen en spieren. Deze gevoelssensoren worden geactiveerd door gepaste mechanische, thermische of chemische stimuli die dan elektrische signalen doorsturen naar het centraal zenuwstelsel. Van hieruit worden commando's naar de spieren gestuurd met een bijpassende reactie.

Daarenboven spelen ook de *fysische eigenschappen* van de huid en het *onderhuids weefsel* een belangrijke rol. De naleving en de wrijvingseigenschappen van de huid maken het samen met de gevoels- en motorische mogelijkheden van de hand mogelijk om een glijdende beweging over een oppervlak waar te nemen en te verkennen zonder contact met dat oppervlak te verliezen. Ook zachte oppervlakken kunnen hiermee gemanipuleerd worden.

Gevoelsinformatie van de hand die doorgegeven wordt aan de hersenen na contact met een object kan men opdelen twee klassen [13]:

- *Tactiele informatie*, waarmee men het gevoel bedoelt van de aard van het contact met het object.
  - ⇨ Dit wordt veroorzaakt door de signalen van zeer gevoelige handreceptoren die zich in de huid bevinden op de plaats van of rondom het contactpunt.
  - ⇨ Er wordt enkel tactiele informatie overgebracht wanneer objecten in contact komen met een *passieve*, *niet bewegende* hand, tenzij er kinestetische informatie wordt doorgegeven die altijd aanwezig is bij een ledemaat in een bepaalde houding.

- *Kinestetische informatie*, waarmee men het voelen van positie en beweging van de gewrichten bedoelt, samen met de bijpassende krachten.
  - ⇨ Dit wordt veroorzaakt door de *gevoelsontvangers in de huid* rond de gewrichten, gewrichtskappen, pezen en spieren. Deze zijn samen met *zenuwsignalen* afgeleid van motorische bewegingen.
  - ⇨ Er wordt enkel kinestetische informatie overgebracht wanneer de hand *actief is* en *vrij beweegt.* Vrije beweging betekent geen contact met een object of andere delen van de huid. De afwezigheid van tactiele informatie impliceert op zichzelf trouwens dat het om een vrije beweging gaat.

Zelfs in de uitzonderlijke gevallen hierboven, blijkt duidelijk dat alle actieve haptische gebeurtenissen zowel qua gevoel als qua manipulatie, steeds gebruik maken van *beide* klassen van informatie. Daarenboven zijn vrije uiteinden van zenuwbanen en gespecialiseerde sensoren die temperatuur, pijn (zowel van mechanische, chemische als thermale aard), of jeuk overbrengen, ook aanwezig bij de mens.

Bij het uitvoeren van om het even welke taak is de voorwaarde dat men controle moet hebben over contacten net zo belangrijk als het aanvoelen van deze voorwaarden. De controle over zo'n actie kan bij mensen variëren van een snelle spier- of ruggegraatsreflex tot een relatief trage, bewust uitgevoerde actie. Experimenten waarin objecten opgeheven worden met een grijpreflex, tonen aan dat motorische acties zoals het verhogen van de gripsterkte reeds binnen 70ms kunnen plaatsvinden nadat een object uit de hand begint te glijden. Bovendien zijn zulke signalen van huidsensoren absoluut noodzakelijk bij het uitvoeren van een taak [13].

Het is bijgevolg duidelijk dat de mechanische eigenschappen van de huid en de onderhuidse weefsels, de rijkelijke informatie die door allerlei sensoren wordt voorzien, en het koppelen van deze informatie aan de acties van het motorisch systeem, verantwoordelijk zijn voor de menselijke mogelijkheden van voelen en manipuleren.

5.3 Haptische interfaces

5.3.1 Algemeen

Haptische interfaces zijn "***toestellen die de gebruiker via krachtterugkoppeling toelaten om objecten aan te raken, te verplaatsen, te creëren of te bewerken in een gesimuleerde virtuele wereld***". [9] Ze zijn opgebouwd uit mechanische componenten die in fysisch contact komen met het menselijk lichaam met de bedoeling informatie uit te wisselen.

Ze worden gebruikt voor taken die in de reële wereld gewoonlijk door mensenhanden uitgevoerd worden zoals het onderzoeken en manipuleren van objecten. Deze interfaces krijgen hun input van een (menselijke) gebruiker en tonen de bijpassende tactiele output. Ze worden al dan niet bijgestaan door visuele of auditieve systemen. Specifieke toepassingen voor haptische interfaces kan u vinden in §5.7.

Wanneer een taak uitgevoerd wordt met een haptische interface, brengt de gebruiker de verwachte motorische acties over en wordt de interface op een fysieke manier gemanipuleerd. Deze brengt dan informatie over naar de gebruiker door zijn of haar tactiel en kinestetisch gevoelssysteem te stimuleren.

In het algemeen hebben haptische interfaces twee functies:

- meten van de posities en de contactkrachten van de gebruiker's hand
- deze contactkrachten en posities tonen

Welke positie- en contactkrachtwaarden gekozen worden voor motorische actie en welke voor het tonen hiervan, hangt af van het *ontwerp* van de interface en de taak waarvoor het bedoelt is.

5.3.2 Kenmerken

Haptische toestellen verschillen in de manier waarop de 3D omgeving gedefinieerd wordt en in de manier waarop de update procedure verloopt, maar de meeste systemen delen enkele gemeenschappelijke kenmerken. Zo moet voor het tempo van het aantal positie-updates voor de gebruiker zeer hoog liggen om een realistische en betrouwbare krachtterugkoppeling te ervaren.

De menselijke hand is ook zeer gevoelig op gebied van herkenning van minuscule oppervlakdetails (0.1µm), temperatuurvariaties (0.05°C), druk (0.03N/cm²),... Bovendien voelt de hand extreem kleine verschillen in afstand en snelheid (10%), versnelling (20%), kracht (7%),...

Haptische perceptie wordt ook versterkt door *multimodale invloeden*. Zo is aangetoond dat visuele informatie een sterke invloed heeft op de haptische perceptie. Ook extra gevoelsinformatie zoals het eerder reeds vermelde tactiele gevoel, vibraties, temperaturen, elektriciteit,... hebben een invloed.

Uiteindelijk worden de specificaties van haptische toestellen bepaald door de mogelijkheden en beperkingen van de gebruiker. Simulaties van haptische interacties met virtuele omgevingen die speciaal ontworpen zijn om reële omgevingen te immiteren, zullen dus steeds bij benadering zijn. De grenzen van de menselijke mogelijkheden zullen bepalen welke benaderingen zullen voldoen.

De gewenste mogelijkheden van haptische interfaces met krachtterugkoppeling zijn als volgt [13]:

1. Een lage inertie en wrijving wanneer er geen contact is met een voorwerp. Bovendien mogen er ook geen beperkingen zijn op het vlak van de fysische beweging van het haptisch toestel zelf. Zo voelt vrije beweging ook als dusdanig aan.

2. Het bereik, de resolutie en de bandbreedte van de haptische interface moet overeenkomen met deze van de gebruiker, zowel op vlak van het voelen van de positie als de krachtterugkoppeling, en dit voor de taken waarvoor de haptische interface moet dienen. De gebruiker mag bijgevolg :
    - niet door solide objecten kunnen gaan bij het overschrijden van de maximumkracht van de interface,
    - geen onverwachte vibraties voelen zoals deze veroorzaakt door het berekenen van de positie tegen een te traag tempo, en
    - geen stijve objecten aanvoelen als zachte wegens een te lage stijfheid van het haptisch toestel.

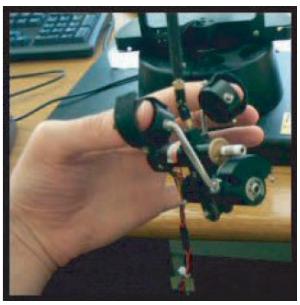
    Aan deze voorwaarden is vrij moeilijk te voldoen omdat de het menselijk zenuwstelsel een grote gevoeligheid en bandbreedte van ongeveer 1000Hz heeft. De bandbreedte van de menselijke controle over haptische interacties is gelukkig slechts ongeveer 10Hz.

3. Ergonomische factoren en comfort spelen een zeer belangrijke rol bij het dragen of manipuleren van een haptisch toestel. Dit is zeer belangrijk aangezien haptische toestellen ook oncomfortabel of zelfs pijnlijk kunnen zijn.

5.3.3 Soorten Haptische interface toestellen

Haptische interface toestellen gedragen zich als kleine robotten die energie in de vorm van krachten uitwisselen met de gebruiker. Zulke interfaces kunnen in contact komen met elk lichaamsonderdeel, maar het gros van deze toestellen spitst zich voornamelijk toe op de hand.

Men kan haptische toestellen van elkaar onderscheiden op basis van de *positie waarin ze gehouden moeten worden* [21]. Zo kunnen terugkoppelings toestellen binnen de hand *vingerspecifieke* haptische informatie lezen en krachten zetten, maar geen inertiekrachten of gewicht van objecten reproduceren (zie Figuur 25).



Figuur 25: Krachtterugkoppelende grijper

Toestellen als deze om in de hand te houden worden veel gebruikt in de game industrie en worden gebouwd met goedkope vibrotactiele converters die kunstmatige vibraties produceren.

Figuur 26: Exoskelet met krachtterugkoppeling

Haptische interfaces of exoskelet toestellen die de gebruiker rond zijn of haar arm of been kan dragen, moeten voorzien zijn van meer complexe gemotoriseerde systemen met meerdere vrijheidsgraden (zie Figuur 26).

Figuur 27: Krachtterugkoppelende Joystick
(Microsoft Sidewinder Pro)

Figuur 28: Krachtterugkoppelende Muis
(Logitech Wingman Force-Feedback mouse)

Dan zijn er nog de grondgebaseerde toestellen waaronder krachtterugkoppelende joysticks (zie Figuur 27) of muizen (zie Figuur 28) en haptische interfaces om op het bureau te zetten (zie Figuren 18, 19 en 20).

Een ander onderscheid tussen haptische toestellen wordt gemaakt door hun *intrinsiek mechanisch gedrag*. Hierin onderscheiden we de *toegankelijkheids*- van de *impedantietoestellen*. De eerste lezen een kracht en sturen een positie terug, terwijl de laatste een positie lezen en een kracht terugsturen. Impedantietoestellen komen veruit het meest voor aangezien ze eenvoudiger te ontwerpen zijn en veel goedkoper om te produceren. Toegankelijkheidstoestellen zoals de Haptic Master worden in het algemeen gebruikt voor toepassingen die zeer sterke krachten nodig hebben binnen een grote werkruimte.

Haptische interfaces worden ook vaak geclassificeerd door het aantal vrijheidsgraden waarin ze kunnen bewegen of kracht zetten. Hiermee bedoelt men het aantal dimensies waarmee bewegingen of krachten uitgewisseld kunnen worden tussen het toestel en de gebruiker. Een vrijheidsgraad kan passief of aangedreven zijn (m.a.w. met of zonder krachtterugkoppeling), gevoeld of niet gevoeld.

Er zijn haptische toestellen van één tot zes vrijheidsgraden. Toestellen met één vrijheidsgraad meten de gebruiker's positie en geven krachten terug in slechts één dimensie. Een voorbeeld zo'n toestel in één dimensie is het krachtterugkoppelend stuurwiel (zie Figuur 29).

Figuur 29: Kracht-terugkoppelend stuurwiel
(Microsoft Force-Feedback wheel)

Krachtterugkoppelende toestellen met twee vrijheidsgraden zijn 'slechts' 2D, maar kunnen al dan niet in combinatie met andere toestellen, gebruikt worden voor 3D manipulatie. Voorbeelden van zulke toestellen zijn de de krachtterugkoppelende joystick en muis (zie Figuur 27, 28).

Toestellen met één en twee vrijheidsgraden geven echter slechts een beperkte haptische perceptie, zoals trillingen en bots effecten en zijn absoluut niet te vergelijken met moderne professionele haptische toestellen met meer vrijheidsgraden.

Figuur 30: De PHANToM Premium (c) Sensable Technologies

Een van de belangrijkste toestellen voor haptische interactie met meerdere vrijheidsgraden is de PHANToM. Dit toestel bestaat uit een soort 'pen' die kan vastgehouden worden en verbonden is met een vast gedeelte. Wanneer de gebruiker in de virtuele realiteit in aanraking komt met een bepaald voorwerp, zal deze een terugkoppeling ondervinden.

De PHANToM werd oorspronkelijk ontwikkeld aan het MIT. Tegenwoordig wordt de PHANToM gemaakt door Sensable Technologies [25] en bestaan er een hele reeks PHANToM toestellen. Er zijn de eenvoudigere PHANToM Desktop en het PHANToM Omni Device met drie vrijheidsgraden (boven-onder, links-rechts, voor-achter), en de professionele PHANToM Premium lijn.

De PHANToM Premium kan bewegingen registreren in zes vrijheidsgraden (ook nog *yaw, pitch* en *roll,* de drie richtingen rond zijn eigen as) en krachtterugkoppeling geven in drie vrijheidsgraden.

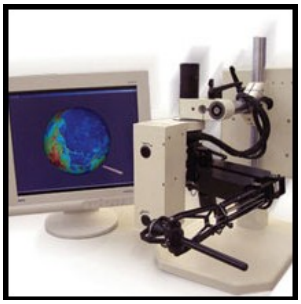
Nog zo'n type toestel is het *Omega haptic device* van Force Dimension [26]. Dit toestel biedt net zoals de PHANToM haptische interactie van drie vrijheidsgraden.

Figuur 31: Omega haptic device
(c) Force Dimension

Figuur 32: Delta haptic device met zes vrijheidsgraden
(c) Force Dimension

Van dezelfde makers is er ook het *Delta haptic device*. Dit toestel heeft naast een versie met drie ook een versie met zes vrijheidsgraden. Deze zijn allemaal actief: er kan dus in alle zes de dimensies een krachtterugkoppeling weergegeven worden.

Nog zo'n toestel met zes actieve vrijheidsgraden is de Freedom 6S van MPB (zie Figuur 33).



Figuur 33: De Freedom 6S van MPB [28]

Figuur 34: Het Haptic Workstation van Immersion[24]

Tenslotte kunnen haptische toestellen nog onderscheiden worden op basis van hun gevoeligheid bij zowel het opnemen als het teruggeven van signalen, of op basis van hun bandbreedte.

Indien haptische interfaces gecombineerd worden met toestellen voor visuele en auditieve interactie bekomt men volledige immersieve systemen zoals het Haptic Workstation van Immersion (zie Figuur 34).

5.4 Haptische interactie

Wanneer het de het menselijk haptisch systeem in contact gaat interageren met een machinaal haptisch systeem door middel van een haptische interface, spreken we van *haptische interactie*.

5.4.1 Soorten interactie

Het gebruik van haptics spitst zich vooral toe op *interactie via de menselijke hand*, wat tot een veelzijdigheid van haptische interactie leidt op gebied van:

- het aanvoelen van haptische prikkels die we kunnen ondernemen zoals
  - voelen van vormen en oppervlaktetextuur
  - vibraties,
  - het voelen van temperaturen en elektriciteit [2]
  - zachtheid, troebelheid, vocht
  - ...

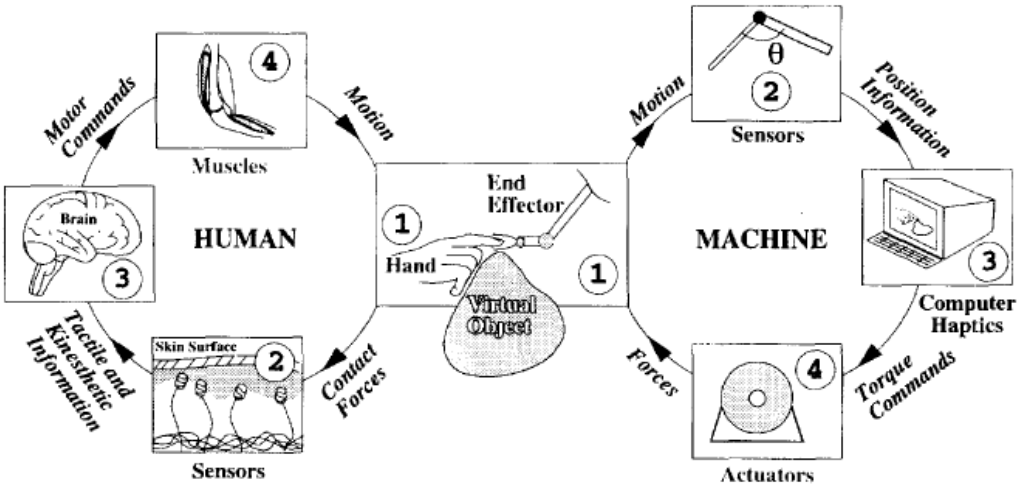
- het interageren met de virtuele wereld door objecten
  - weg te duwen
  - vast te nemen
  - knijpen
  - aaien
  - ...

- het hanteren van voorwerpen zoals een
  - pen
  - beitel
  - ...

Al deze soorten interactie kunnen in principe mogelijk gemaakt worden met behulp van haptische interfaces. Toch zijn toestellen die al deze soorten interactie samen mogelijk maken en zo voor een totaalgevoel van immersie zorgen nog niet voor vandaag. Deze thesis ziet het soort haptische interactie met de virtuele omgeving dan ook voornamelijk in de zin aanraken en manipuleren via één enkele virtuele pointer.

5.4.2 De interactiecyclus

Zowel bij de mens als bij een machine bestaat dit haptisch systeem uit een haptische lus van collision detectie en de bijpassende reactie. Deze lus van actie en reactie blijft zich voordoen tijdens zowel het *voelen* als het *manipuleren* van objecten, en wordt geïllustreerd in figuur 35.



Figuur 35: Haptische interactie via een interface tussen human en machine haptics [10].

In figuur 35 zien we duidelijk dat de haptische lus bij zowel mens als machine uit dezelfde stappen bestaat, maar dat de volgorde omgekeerd is aangezien de mens eerst *ageert* terwijl de machine *re-ageert*. Op deze reactie zal de mens dan weer reageren etcetera. Op deze manier ontstaat een wisselwerking.

Het begint wanneer de menselijke hand in contact komt met een object in de virtuele omgeving die weergegeven wordt door het haptisch toestel (1). Net zoals bij het aanraken van een object in de reële wereld, voelen de sensoren in de hand contactkrachten en nemen zo het contact met het virtuele object waar (2). De tactiele en kinestetische informatie die de sensoren opvingen, wordt via de zenuwbanen doorgegeven aan de hersenen voor interpretatie (3). De hersenen ervaren een haptische perceptie, interpreteren deze prikkels en er wordt een gepaste reactie bedacht. Deze wordt in de vorm van motorische commando's doorgegeven aan de spieren (4), welke voor een nieuwe beweging en draaiing van de hand zorgen (1).

Ondertussen hebben de sensoren van de haptische interface door het contact van de menselijke hand vastgesteld als een beweging naar een nieuwe positie en draaiing (*torque*) (2). Deze informatie wordt doorgeven aan de computer (3). Hier bevinden we ons op het actieterrein van de *computer haptics*, waarbij de voorwerpen uit de virtuele wereld *haptisch gerenderd* gaan worden (zie §5.5). De computer berekent aan de hand van de verkregen gegevens nieuwe parameters en geeft ze door aan het aandrijfsysteem van het haptisch toestel (4). De haptische interface genereert dan de gepaste krachten, die weer op de hand inwerken (1).

Wanneer deze interactiecyclus tussen de haptische lussen van mens en machine zich blijft verder zetten, is er sprake van real-time interactie. Om deze correct te genereren dienen mens en machine gesynchroniseerd te

werken. Zoals eerder vermeld is hiervoor bij de machine een vrij hoog tempo aan updates nodig om op een realistische en intuïtieve manier interactie te verzorgen.

5.5 Haptisch renderen

Definitie: Haptisch renderen is het proces waarbij men van een bepaald type 3D-model neemt en er bijbehorende krachten op berekent om zo de gebruiker via een haptisch toestel de illusie te geven van fysisch contact. **[9]**

5.5.1 Computer haptics

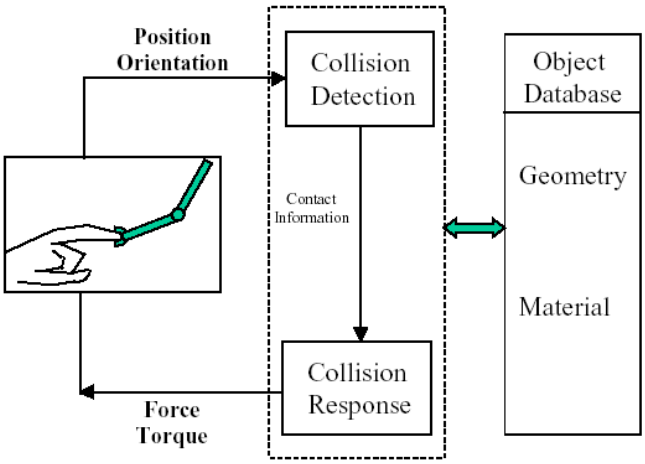
Om haptische interactie tussen de mens en een toestel mogelijk te maken, dient de interface de mogelijkheden te hebben om naargelang de omstandigheden een gepast gevoel te creëren bij het aanraken en manipuleren van virtuele objecten. *Computer haptics* is de tak van de wetenschap die zich bezig houdt met het zoeken naar technieken en processen die dit mogelijk maken.

Analoog aan *computer graphics*, houdt *computer haptics* zich bezig met modellen en gedragingen van virtuele objecten, net zoals met renderalgoritmes voor weergave in real-time. Het omvat niet enkel de software architectuur die nodig is voor haptische interacties, maar ook zijn synchronisatie met visuele of andere weergavemodaliteiten.

5.5.2 Verloop van haptisch renderen

Haptische objecten worden in virtuele omgevingen gerenderd door middel van een haptische interface. Zoals we reeds gezien hebben in de paragraaf over haptische interactie, verloopt het haptische proces zowel bij de mens als bij de machine via detectie van aanrakingen en een passende reactie. Het renderproces van een haptische interface bestaat dan ook uit dezelfde twee stappen:

- collision detectie
- collision reactie



Figuur 36: De Haptische loop waaruit het haptisch renderproces bestaat, met in elke loop detectie en reactie [10].

Telkens de gebruiker gebruiker van positie verandert, controlleert de haptische loop of er aanraking is met een virtueel object (*collision detectie*), en indien dit zo is, volgt een reactie (*collision reactie*) in de vorm van een bepaalde kracht die berekend wordt door het haptisch rendering algoritme.

Wanneer de gebruiker de pointer van het haptisch toestel aanraakt, worden de nieuwe positie en oriëntatie van deze pointer opgemerkt door de sensoren in het toestel. Is er geen collision tussen de virtuele representatie van de pointer en virtuele objecten, dan blijkt de haptische interface passief en worden er geen krachten teruggekoppeld naar de gebruiker.

Wanneer de sensoren van het haptisch toestel wel een collision met een object hebben opgemerkt, dan wordt een reactiekracht berekend. Deze is gebaseerd op de diepte waarmee de pointer in het virtueel object is binnengedrongen. De berekende krachtvectoren kunnen dan nog aangepast worden met andere gegevens zodat ook rekening gehouden wordt met details van het oppervlak. Uiteindelijk worden deze gewijzigde krachtvectoren teruggegeven aan de gebruiker via het de haptische interface. Dit verloop van collision detectie en reactie noemt men de *haptische loop*, die onophoudelijk en tegen een tempo van ongeveer duizend loops per seconde moet verwerkt worden. Indien dit niet gebeurt kunnen virtuele oppervlakken zachter aanvoelen. Erger nog, het virtueel toestel kan gewoon vibreren in de gebruiker's hand.

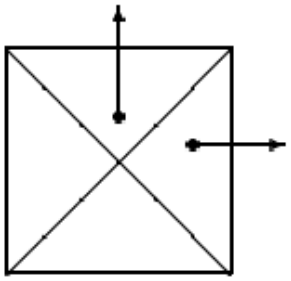
De laatste jaren werden verscheindene technieken ontwikkeld om virtuele objecten haptisch te renderen. Bij deze technieken wordt een onderscheid gemaakt worden door de manier waarop de virtuele pointer gemodelleerd wordt. Deze kan gemodelleerd worden als een

- punt
  Dit model is analoog aan het verkennen en manipuleren van echte objecten door enkel de tip van een stok te gebruiken.
- straal of lijnsegment
  Dit model is ook analoog aan het verkennen en manipuleren van echte objecten met een volledige stok, inclusief de tip.
- 3D object, bestaande uit een een aantal punten, lijnsegmenten en polygonen

Welk type gebruikt wordt hangt af van de noden en complexiteit van de applicatie waarvoor het moet dienen.

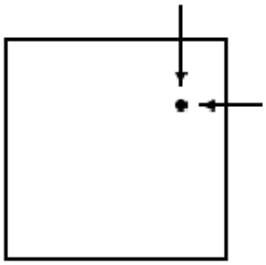
5.5.3 De Penalty Based Methode

Doorheen de jaren werden verschillende methodes bedacht om haptische krachten te berekenen. In het begin maakte men gebruik van de *penalty based methode*.



Figuur 37: Penalty methode bij een simpel geometrisch object. [11]

Deze methode associeert elk oppervlak van een object met een deel van zijn volume en bepaalt zo de richting van de kracht. De intensiteit van die kracht staat in directe verhouding met de diepte waarmee de gebruiker op het voorwerp inwerkt.



Figuur 38: Twee mogelijke paden bereiken dezelfde locatie. Welk pad werd genomen? [11]

Deze methode werkt goed bij eenvoudige objecten zoals kubussen, sferen, ... (zie Figuur 37) Ze is bovendien gemakkelijk te implementeren. Wanneer er echter interactie nodig is met meer complexe objecten, ontstaan er problemen.

Zo kan een kracht het object binnengedrongen zijn langs verschillende wegen en zich toch op dezelfde plaats bevinden (zie Figuur 38). Toch zal de *penalty based methode* automatisch het binnendring punt bepalen naargelang de plaats waar de gebruiker zich 'bevindt'. Dit veroorzaakt een foutenlast.

Verder laten *penalty based methodes* het toe om door dunnere objecten uit te duwen aangezien het binnendring punt bij het uitoefenen van een grote druk in een ander deel-volume terecht komt (zie Figuur 39).

Figuur 39: Bij dunne objecten is het mogelijk door de figuur uit te duwen. Dat zou niet mogen.

Wanneer verschillende primitieven elkaar raken of met elkaar intersecteren is het tenslotte ook moeilijk om de terugkoppelingskracht te bepalen. Het optellen van de krachten kan leiden tot zeer grote krachten die in bepaalde gevallen gevaarlijk zouden zijn voor het haptisch toestel of voor de gebruiker. Daarenboven zijn kracht en richting niet altijd uniek bepaald. Deze nadelen zijn van die aard dat er zich een betere methode opdringt om krachten te bepalen. Daarom wordt tegenwoordig nog slechts gebruik gemaakt van *constraint based methodes*.

5.5.4 De Constraint Based Methode

De *constraint based* methode biedt een oplossing aan al deze problemen door de krachtterugkoppeling niet af te laten hangen van de diepte waarmee er in het object wordt ingewerkt, maar van beperkingen of *constraints*.

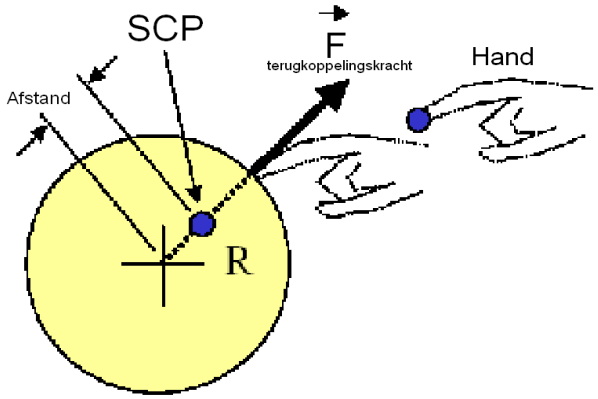
Bij *constraint based* methodes wordt de *positie* van de gebruiker in de virtuele omgeving voorgesteld door een punt, dat zich op de rand van het object bevindt en via een virtuele (stijve) veer verbonden is met de werkelijke positie van de pointer binnenin dit object. Dit object noemt men het *Surface Contact Point* (*SCP*) en heeft meestal de vorm van een kleine sfeer.

Nota: Het SCP is vaak gekend onder andere namen zoals *God Object, (Virtual) Proxy (VP), Haptic Point (HP), Haptic Interface Point(HIP), Ideal Haptic Interface Point (IHIP)*...

Definitie SCP: Een SCP wordt gedefinieerd als een eindige, massaloze sfeer, die van de objecten uit de virtuele wereld geen topologische kenmerken vereist. [9]

Omdat een SCP geen topologische kenmerken nodig heeft, kan dit systeem van haptisch renderen toegepast worden in dynamische omgevingen en ook bij objecten die voorgesteld worden door middel van een *scène Graph*, met veel intersecterende obstakels. De constraint based methode leent zich dus perfect voor het haptisch renderen van CSG objecten.

Constraint-based haptisch renderen werkt als volgt: wanneer de gebruiker zijn positie in de virtuele omgeving wijzigt, kan hij door een of meer virtuele obstakels uitgaan. Het SCP daarentegen wordt tegengehouden door de obstakels en het verplaatst zich snel naar een positie die de afstand minimaliseert met de positie van de gebruiker naargelang de constraints in de virtuele omgeving (zie Figuur 40).



Figuur 40: Haptisch contact via het SCP (hier HIP). Hierbij wordt de link tussen het SCP en de positie van de hand technisch voorgesteld door het plaatsen van een lineaire veer waarbij F=kx met k zijnde de stijfheid van het object en x de penetratiediepte. Uiteraard bevindt deze kracht zich in de richting van de oppervlaktenormaal.

Men maakt verder gebruik van het haptisch toestel om de krachten van dit SCP te berekenen. Dit komt dan bij de gebruiker over als de beperkende (*constraint*) krachten ten gevolge van contact met de verwachte omgeving.

Bij elke iteratie wordt de huidige positie van het haptisch toestel gevonden en wordt de ideale passende reactie berekend:

- ➔ Eerst wordt gebruik gemaakt van een bepaalde methode van ***collision detection*** om te kijken of het mogelijk is om het SCP te verplaatsen in de richting van de ideale passende reactie.

- ➔ Het SCP wordt dan via een lineaire beweging zo ver mogelijk verplaatst in die richting van de gebruikerspositie. Dit is de ***move stap***. Indien er zich geen object bevindt tussen het SCP en de positie van de gebruiker, wordt de SCP meteen naar zijn doel verplaatst. Ligt er wel een object in deze baan, dan wordt het SCP verplaatst tot het contact maakt met de eerste primitieve(n) langs de directe lijn naar het doel. Daar wordt dan een nieuwe mogelijke bewegingsrichting gevonden.

➔ Aan het einde van de iteratie volgt de ***update stap***, waar het SCP zijn doel zo dicht mogelijk probeert te naderen en zich verplaatst ('*glijdt*') over het oppervlak (zie Figuur 41). Hier wordt de foutmarge tussen de gebruiker's positie en het SCP berekend om zo de krachten van het haptisch toestel te bepalen. Het toestel vermindert de foutmarge door de gebruiker's positie op een fysische manier te verplaatsen in de richting van het SCP.

Figuur 41 : In de update stap verplaatst het SCP zich over het oppervlak om zo nog dichter bij het doel te komen.

Door gebruik te maken van een SCP is het steeds geweten van *waaruit* de kracht ingewerkt wordt, wat problemen zoals die in Figuur 39 onmogelijk maakt. Het SCP vereist geen topologieën meer van de obstakels in de virtuele wereld waardoor een pre-processing stap niet meer nodig is. Tenslotte zorgt de sfeervormigheid van het SCP voor meer volume waardoor het vrijwel onmogelijk wordt om door een dun object of door een *polygonsoup* [12] heen te duwen.

De grootte van het SCP moet wel correct bepaald worden: indien het SCP te groot is, is er een verlies aan precisie. Is het SCP te klein is, schiet de gebruiker wel eens door een spleet in het object, terwijl dat niet de bedoeling is. Dit komt wel eens bij polygonenmeshes die niet helemaal gesloten zijn.

Tenslotte wordt er hiervan uitgegaan dat een oppervlak steeds *perfect glad* is. In de 'echte' wereld is dit uiteraard niet zo. De constraint based methode kan uitgebreid worden om hieraan tegemoet te komen. Men kan namelijk dynamische frictie invoeren door het SCP niet te verplaatsen wanneer de hoek waarover verplaatst wordt te klein is (zie ook Figuur 41).

Er bestaan nog enkele andere uitbreidingen om de constraint based methode rekening te laten houden met 'reële' omstandigheden zoals statische en viscose frictie, textures, curves die gebruik maken van data die normaal bestemd is voor Phong of Gouraud shading, etc. Deze vallen echter buiten de intentie van deze tekst.

5.5.5 Verplaatsen en updaten van het SCP

Wanneer het volume rond het SCP bevindt zich *verplaatst* (volgens een lineair pad) in de richting van zijn doel, wordt er gecontroleerd of het niet een van de primitieven uit de omgeving binnendringt. In het begin is de bestemming van het SCP de positie van de gebruiker, maar we zullen in de update stap zien dat dit verandert.

Voor het SCP geldt namelijk het volgende:

- Indien het SCP op zijn weg een obstakel zal tegenkomen, dan wordt het verplaatst tot het in aanraking komt met de eerste primitieve(n) langs zijn pad.
- Indien het SCP op zijn weg *geen* obstakels tegenkomt, dan wordt het rechtstreeks naar zijn doel verplaatst.

De *update* stap is niet nodig indien het SCP na een move-stap aangekomen is bij de positie van de gebruiker. Indien het SCP er niet geraakt, moet een nieuwe richting gevonden worden om de afstand tot de gebruikers positie te verminderen. Er dient hier echter rekening gehouden te worden met de beperkingen van het oppervlak waarmee het SCP in aanraking gekomen is. Dit kan door de *bewegingsruimte* van het SCP te bepalen.

Binnen deze bewegingsruimte bevindt het SCP zich op punt nul en worden de primitieven die in contact staan met het SCP omgezet naar *bewegingsruimte obstakels*. Deze bestaan uit alle punten die zich bevinden binnen een afstand tot de originele obstakels die kleiner is dan de diameter van de SCP sfeer.

Men definieert vervolgens voor elke aanliggende primitieve een vlak dat gelijk loopt met het oppervlak van de bewegingsruimte en dat het SCP punt intersecteert. De mogelijke beweging van het SCP wordt door elk vlak beperkt. De overgebleven ruimte waarin het SCP verplaatst kan worden, bevindt zich in de bewegingsruimte, en dus hoofdzakelijk boven het SCP, terwijl de positie van de gebruiker *onder* het SCP ligt.

De intersectie van alle overgebleven deel-ruimtes vormt een convexe polygoon die alle punten aangeeft die vanuit het SCP in een lineaire beweging bereikt kunnen worden. Het deel hiervan dat zich het dichtst bij de gebruiker's positie bevindt, wordt dan het nieuwe doel.

5.5.6 Haptisch renderen ≠ Visueel renderen

Gezien de manier waarop haptisch renderen gebeurt, krijgt men al snel de indruk dat haptische 3D objecten gewoon op dezelfde manier voorgesteld kunnen worden als grafische 3D objecten. Toch zijn er enkele principes die visueel perfect werken, maar haptisch tekort schieten. We noemen hier enkele '*issues*' [12].

In veel grafische en CAD systemen worden polygonen individueel voorgesteld in de 3D ruimte. Men doet niet de moeite om de vertices met elkaar te verbinden, wat het teken proces sneller en eenvoudiger maakt. Zo creëert men minuscule gaatjes tussen de polygonen. Visueel is dit niet te merken. Haptisch daarentegen ontstaat er het probleem dat de gebruiker van een haptisch toestel regelmatig tussen twee polygonen door kan vallen ('***pop-through***'). Technisch gezien is er niets fout, maar het spreekt voor zich dat dit niet de bedoeling is.

Nog zo'n probleem is de ***back-face culling***. Bij deze veelgebruikte optimalisatie techniek worden vlakken met een normaal die weg van het oogpunt van de gebruiker wijst, niet getekend. Visueel kan dat perfect aangezien de gebruiker de wereld bekijkt vanuit één oogpunt. De aan de achterkant liggende vlakken leveren immers *geen bijdrage aan het visuele beeld*. Voor haptics zijn deze vlakken wel degelijk van belang om te vermijden dat alle solide objecten zonder meer langs achterom kunnen betreden worden. Indien we dus haptics en graphics samen willen renderen via grafische hardware, blijkt dus dat in de grafische pipeline het haptisch systeem zijn werk gedaan moet hebben *alvorens* aan back-face culling gedaan kan worden voor het visueel systeem.

Een laatste probleem is dat van de diepte waarde of Z-waarde. Visueel is men gewoon dat Z een oneindige waarde kan aannemen. Er kan een maximum diepte ingesteld worden indien gewenst, maar werken met een oneindige diepte is visueel gezien geen probleem. In de wereld van de haptics is dit niet mogelijk: het haptisch toestel heeft immers fysieke beperkingen. Het kan dus gebeuren dat een object groter wordt dan het fysieke bereik van het toestel, en hier dient rekening mee gehouden te worden.

Ook bij het maken van een goede haptische navigatie en bij het aanmaken van haptisch bruikbare knoppen valt het op dat niet alle principes uit de grafische wereld simpelweg toegepast kunnen worden. De verschillen tussen het haptische en het visuele zullen trouwens niet alleen duidelijk worden op het gebied van renderen, zoals duidelijk zal worden in verdere hoofdstukken (Hoofdstuk 7 en volgende).

5.5.7 Haptische Framewerken

Net zoals in andere takken van software ontwikkeling is er voor het programmeren via een haptische toepassing nood aan een framewerk, een bibliotheek van instructies die het mogelijk maakt om het haptisch toestel op een high-level manier aan te spreken. Hier volgt een kort overzicht van de belangrijkste haptische framewerken.

Het meest bekende en meest gebruikte framewerk is de ***GHOST Software Development Kit*** van Sensable Technologies [25]. Dit is een krachtige tool kit die werkt met C++. Het is opgevat als een soort haptische physics engine die zelf de complexe berekeningen uitvoert en de ontwikkelaar enkel bezig laat zijn met de eenvoudige virtuele objecten en hun fysieke eigenschappen zoals positie, massa, wrijving en stijfheid. Het bevat ook een reeks primitieve 3D objecten, polygonale objecten en haptische effecten en is zeer uitbreidbaar. Het groot probleem van de GHOST SDK is dat ze enkel werkt met haptische interfaces van eigen huis, met andere woorden de PHANToM toestellen.

Een andere SDK is ***Open Haptics***, ook van Sensable Technologies. Deze toolkit is, net als OpenAL voor geluid, gemodelleerd naar de OpenGL API. Hierdoor kan OpenGL code vergezeld worden van OpenHaptics commando's. Zo verlaagt men ook de drempel voor grafische programmeurs om ook haptisch te gaan programmeren. Ook weer stelt zich hier het probleem dat deze API enkel en alleen met PHANToM toestellen werkt, net als de ***H3D API***, nog een ander framewerk.

Hetzelfde probleem stelt zich ook bij de ***DHD API*** van Force Dimension [26]. Dit framewerk werkt met C code, is vrij uitbreidbaar en het zou volgens de makers slechts vier lijntjes C code kosten om een OMEGA of DELTA toestel in applicaties te integreren. Dit framewerk werkt ook niet met andere toestellen.

Andere API's voor haptics zijn onder meer de ***MPB API***, die vrij low-level is en enkel werkt voor toestellen van MPB zelf [28], en ***ReachIn API***, een commercieel en dus niet open source framewerk.

Het is duidelijk dat er hier nood is aan een *algemeen, open-source framewerk* dat werkt met de meeste toestellen. Elk toestel kwam met een reeks libraries die specifiek voor dat toestel geschreven waren. Het ontbreken van een algemeen framework is een serieuze beperking voor een veralgemeend gebruik van haptische toestellen. Daarom werd aan het E.D.M. (Expertise-centrum voor Digitale Media) zo een framewerk genaamd HAL ontwikkeld door mijn co-promotor Prof. Chris Raymaekers en Lode Vanacken [40].

HAL staat voor "**HA**ptic **L**ibrary" en is een zeer uitbreidbare bibliotheek en bevat reeds verschillende bruikbare componenten zoals scènegraph implementaties, een reeks voorgedefinieerde haptische objecten zoals een sfeer of polygonenmesh, en verschillende soorten krachtvelden. Bovendien wordt een driver meegeleverd voor de populaire PHANToM 3D en een pseudo driver voor test mogelijkheden. Voor de implementatie van de haptische CSG modeller wordt bijgevolg van deze library dankbaar gebruik gemaakt.

HAL is intussen niet meer de enige algemene haptische functiebibliotheek. ***CHAI 3D*** [27] is een open source, gratis beschikbare API voor C++ en werkt met de meeste haptische toestellen die vandaag op de markt zijn. Deze API werkt net zoals OpenHaptics volgens een OpenGL-achtige structuur.

5.6 Toepassingensgebieden voor haptics

Het toevoegen van een haptische component opent voor bepaalde systemen heel wat nieuwe mogelijkheden, vooral op gebied van virtuele omgevingen en besturingen vanop afstand. Hierbeneden volgen enkele belangrijke toepassingsgebieden [26].

- ***medische simulatoren***. Zulke multi-modale systemen van virtuele omgevingen kunnen gebruikt worden bij het opleiden van chirurgen, net zoals vluchtsimulatoren gebruikt worden voor het trainen van piloten. Deze kunnen voor medici in opleiding hele operaties simuleren waarbij ze realistische modellen van organen en weefsel kunnen zien, aanraken en manipuleren. Uiteraard dient hiervoor zowel specifieke software als hardware ontwikkeld te worden. Er zijn reeds zulke simulatoren in ontwikkeling waaronder de "*minimaal invasieve chirurgiesimulator*" [22], een simulator voor kijkoperaties.

- ***collaboratieve haptics*** [23]. Hier gaat men haptics als extra modaliteit gebruiken om de interactie tussen mens en computer te verhogen, of tussen mensen onderling met behulp van computers.

Hiervoor wordt een multimodale virtuele omgeving gedeeld door verschillende gebruikers. Ze kunnen hierin bepaalde taken samen uitvoeren, waarbij het niet uit maakt of ze zich in realiteit vlak bij elkaar dan wel ver van elkaar bevinden. De mogelijkheden van virtuele samenwerking gaan van het op afstand uitvoeren van medische operaties (of ander gespecialiseerd werk) tot geavanceerde multiplayer games.

- ***Geneeskunde***. Men kan robots van op afstand besturen bij kijkoperaties, een diagnose stellen vanop afstand of hulpmiddelen bieden voor gehandicapten.

- ***Ontspanning***. Er zijn video games en simulatoren denkbaar die de gebruiker toelaten om virtuele solide of vloeibare objecten aan te raken en te manipuleren en voorwerpen te gebruiken.

- ***Educatie***: men kan bijvoorbeeld studenten voeling laten krijgen met bepaalde fysische fenomenen op allerlei schalen, ...

- ***Industrie***: men kan bijvoorbeeld haptics integreren in CAD systemen zodat een designer mechanische componenten zelf kan opbouwen en assembleren in een immersieve omgeving, ...

- ***Kunst***: virtuele kunsttentoonstellingen, concertzalen, musea, muziekinstrumenten, ...

5.7 Conclusie

In dit hoofdstuk hebben we een overzicht gegeven van haptics. We hebben haptics gedefinieerd en besproken. Hierbij werd een verschil gemaakt tussen het haptisch systeem bij de mens en zijn machinale clone. Interactie tussen beide gebeurt via deze haptische interfaces of toestellen. Er kwam een overzicht van soorten interfaces, waarbij ook gefocust werd op concrete toestellen zoals de PHANToM en de DELTA.

Verder zijn we dieper ingegaan op de werking van deze haptische toestellen. Dit gaat via een haptische loop van aanvoelen en reageren, die aan een hoog tempo moet uitgevoerd worden. Tegen dit tempo moet de impact van de acties van de gebruiker op de virtuele modellen bepaald worden om vervolgens tot een passende krachtterugkoppeling te leiden. Dit proces noemt men *haptisch renderen* en werd verder onder de loupe genomen.

Na het bestuderen van enkele methodes werd het duidelijk dat de op beperkingen gebaseerde methode duidelijk de meest aangewezen is. Deze methode maakt gebruik van een oppervlakcontact punt, het SCP. Dit punt zit met een virtuele veer vast aan de virtuele pointer die de gebruiker's positie in de virtuele omgeving voorstelt. Wanneer de gebruiker de pointer van het haptisch toestel in een object steekt, blijft het SCP steeds op de rand van het object zitten. Door de virtuele veer zal de krachtterugkoppeling groter worden naarmate de pointer dich dieper in het virtuele object binnendringt.

Er is echter een hele weg af te leggen vooraleer applicaties en gebruikersinterfaces de juiste manier vinden om gebruikers door virtuele omgevingen te sturen. We hebben hierbij reeds gezien dat optimalisaties voor grafische systemen vaak niet van toepassing zijn op hun haptisch analogon. Bestaande applicaties die een haptische component inbouwen als extra feature, zullen qua *interface* problemen krijgen. Er zijn dus nieuwe en andere inzichten nodig op zowat alle gebied om ervoor te zorgen dat de haptische component iets *toevoegt* aan de bestaande systemen en geen handicap gaat vormen voor de gebruiker. [12]

Haptische toestellen zullen dus in de toekomst ongetwijfeld nog heel wat transformaties ondergaan. Want nog steeds zijn de huidige modellen van de haptisch in real-time gerenderde virtuele objecten vrij simplistisch vergeleken met het statisch en dynamisch gedrag van reële objecten waardoor haptische interactie nog steeds onrealistisch aanvoelt. [13]

Toch is het duidelijk dat haptics een extra dimensie geven aan virtuele omgevingen en een gevoel van immersie met zich meebrengen dat anders onmogelijk geweest zou zijn.

Hoofdstuk 6. Haptisch renderen van CSG bomen

6.1 Inleiding

Indien we via CSG opgebouwde 3D objecten niet alleen willen zien, maar ook kunnen voelen met behulp van een haptische interface, is er nood aan een methode om deze objecten *haptisch* te kunnen renderen.

Er bestaat zo'n haptisch render algoritme voor het renderen van CSG bomen (§6.2). Dit werd bedacht aan het E.D.M. door Raymaekers en Van Reeth [14] en een uitgebreide omschrijving van deze methode vormt de kern van dit hoofdstuk.

Aangezien CSG bomen operatoren bevatten die delen van een primitieve kunnen wegsnijden (intersectie, subtractie), moet zowel de buiten- als de binnenkant van deze primitieven gerenderd kunnen worden. Waar algoritmes voor het renderen van de buitenkant van een sfeer, kubus of cilinder reeds bekend zijn sinds de eerste haptische interfaces, zijn algoritmes voor het renderen van hun binnenkant nog niet bepaald. Paragraaf 6.3 gaat hierop in en legt uit hoe dit gebeurt. Eens dit duidelijk is, kan een inside-outside op de verschillende operatoren het juiste SCP bepalen (§6.4). De zoektocht en keuze van het SCP zal naargelang de operator ook verschillende issues met zich mee brengen, die verder besproken worden.

6.2 Algemeen

Net zoals bij andere representaties van 3D objecten, verschillen methodes voor het haptisch renderen van CSG bomen totaal van deze voor het visueel renderen van CSG bomen (zie Hoofdstuk 3).

Grafische algoritmes kunnen zoals eerder reeds aangegeven (§5.5.6), gebruik maken van het feit dat de gebruiker slechts een beeld heeft vanuit één standpunt en dus methodes zoals ray casting toepassen. Voor haptische algoritmes lukt dat niet omdat het nodig is objecten vanuit *alle* kanten voelbaar te maken, dus ook de onzichtbare kanten. Het algoritme van Raymaekers en Van Reeth [14] zal dit mogelijk maken.

Een haptisch renderalgoritme voor CSG modellen moet dus in staat zijn om het SCP te berekenen voor CSG modellen. Om ervoor te zorgen dat de basis-primitieven (sfeer, kubus en cilinder) later uitgebreid kn worden, werkt dit algoritme *onafhankelijk* van hun geometrie. Er moet op voorhand enkel geweten zijn hoe het SCP op de rand berekend moet worden indien het punt zich *binnen* zijn volume bevindt; en of een punt binnen of buiten het volume valt. Indien hieraan voldaan is, kan een primitieve gebruikt worden in een CSG boom. Uiteindelijk voegt het algoritme de SCP's van alle primitieven samen.

In het geval een primitieve van een CSG object gesubtraheerd wordt, ontstaat er een gat in het CSG object. Dit kan visueel zonder problemen gerenderd worden aangezien de rand van een de primitieve er langs binnen hetzelfde uitziet als langs buiten. Het bepalen van een SCP aan de binnenkant van een primitieve is minder eenvoudig: de bekende rendering algoritmes kunnen namelijk niet gebruikt worden indien hun oppervlak niet continu is. Voor een sfeer is er dus geen probleem, maar voor een kubus of een cilinder moest een nieuw algoritme bedacht worden voor het berekenen van het SCP indien het punt zich *buiten* zijn volume bevindt. (zie §6.4)

6.3 Haptisch renderen van de primitieven

Het haptisch renderen van CSG-primitieven vormt de basis van het renderproces van CSG-bomen. Indien elke primitieve apart gerenderd kan worden, kan de hele CSG-boom ook gerenderd worden als combinatie van meerdere zulke CSG-bomen. Van deze primitieven moet, zoals eerder vermeld, niet alleen de *buitenkant* gerenderd kunnen worden, maar ook de *binnenkant*.

- In het algemeen kan men de *binnenkant* van een primitieve haptisch renderen door het binnen-volume in verschillende segmenten op te delen namelijk één voor elk continu oppervlak.

- In het algemeen kan men de buitenkant van een primitieve ook haptisch renderen door zijn volume in verschillende segmenten op te delen. Hier is echter een grotere opdeling nodige om de hoekigheid te bewaren. Er is dus:

    - één deel-volume voor elk continu oppervlak
    - één voor elke ribbe- en puntovergang tussen deze continue oppervlakken, en
    - het binnen-volume.

We behandelen hierbeneden het render proces voor de drie standaard CSG-primitieven:

6.3.1 Sfeer

Voor de sfeer is het bepalen van het SCP een speciaal en zeer eenvoudig geval. Aangezien de sfeer een continu oppervlak heeft, hoeft dit niet opgedeeld te worden in veel deelsegmenten. Er is enkel de binnen- en de buitenkant van de sfeer.

Berekenen van het SCP is dus vrij eenvoudig: bevindt de positie van de gebruiker zich binnen de sfeer, dan wordt het SCP bepaald door deze op een afstand van de lengte van zijn straal te plaatsen in de richting vanwaar de gebruiker kwam.

Het bepalen van het SCP aan de binnenkant, met andere woorden het haptisch renderen van de buitenkant, verloopt *identiek*. Er is en blijft slechts één continu oppervlak en dus kan het SCP makkelijk op de (binnen)rand van de sfeer geplaatst worden.

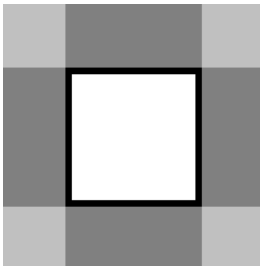
6.3.2 Kubus

In tegenstelling tot een sfeer bestaat een kubus uit meerdere oppervlakken (zes in totaal). Voor het haptisch renderen van de binnenkant van een kubus, wordt het volume opgedeeld in zes segmenten. Ieder in de vorm van een pyramide waarvan de basis overeen komt met één van de zes vlakken. (Zie Figuur 42)

Figuur 42: Segmenten voor haptisch renderen van (een dwarsdoorsnede van) de binnenkant van een kubus

Wanneer de gebruikerspositie zich binnen een van deze zes deelsegmenten bevindt, wordt het SCP op het erbij horende oppervlak geplaatst.

De algoritmes voor het haptisch renderen van de *binnenkant* van een kubus kunnen ook toegepast worden om de *buitenkant* te renderen. Toch voelen de ribben *niet hoekig* aan waardoor de gebruiker niet de indruk heeft de binnenkant van een kubus te voelen. Er is hier dus, in tegenstelling tot bij de sfeer, *een apart algoritme* nodig voor het haptisch renderen van de binnenkant.



Figuur 43: Segmenten voor haptisch renderen van (een dwarsdoorsnede van) de buitenkant van een kubus

Hiervoor wordt buitenkant van de kubus wordt hier opgedeeld in maar liefst zevenentwintig verschillende segmenten (zie Figuur 43):

- zes voor de buitenkant van de zes vlakken,
- twaalf voor de buitenkant van de twaalf ribben (edges),
- acht tegenover de acht punten (vertices) van de kubus en
- het binnen-volume.

Dit algoritme doet de ribben en hoeken van de binnenkant van een kubus wel degelijk hard aanvoelen.

6.3.3 Cilinder

Voor het haptisch renderen van een cilinder wordt gebruik gemaakt van zowel eigenschappen uit het renderen van de *sfeer* als de *kubus*. De buitenkant van de cilinder wordt bijgevolg opgedeeld in zes segmenten:

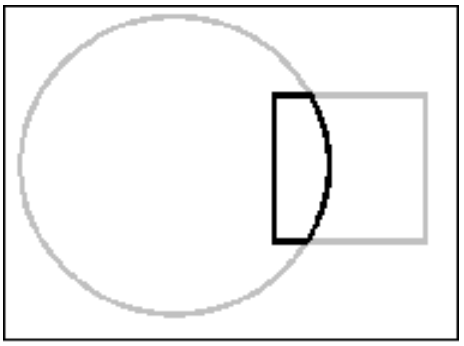
- het volume rond de (ronde) zijkant van de cilinder,
- de twee volumes die de boven en onderkant van de cilinder '*verlengen*',
- de twee volumes die overeenstemmen met de cirkelvormige ribben tussen de zijkant en de boven- en onderkant van de cilinder en tenslotte
- het binnen-volume.

6.4 De inside-outside test

Nadat het renderalgoritme een SCP kan berekenen van de buiten- en binnenkant van de primitieven, dient de *inside-outside test* toegepast te worden op de operaties tussen primitieven en tussen verschillende deelbomen. Zoals reeds eerder werd aangegeven, zijn er drie binaire zulke binaire operatoren: *intersectie*, *subtractie* en *unie*. Hieronder worden algoritmes beschreven die voor elk van deze operatoren. [14]

6.4.1 Intersectie

De intersectie van twee objecten bestaat uit het volume dat door beide objecten tegelijk ingenomen wordt (zie figuur 44). Enkel dit gedeelte mag door de gebruiker gevoeld worden en dus dient men enkel van dit gedeelte het SCP te berekenen. Indien de gebruiker zich buiten de intersectie bevindt, hoeven er geen krachten berekend te worden.



Figuur 44: Intersectie tussen twee objecten

Er moet gecontroleerd worden of de positie van de gebruiker zich binnen de linker en/of rechter deelboom bevindt. Dit wordt op voorhand berekend in de deelbomen, zoals in elk depth-first recursief algoritme. Dit algoritme klopt, want indien de gebruiker zich binnen de intersectie bevindt, liggen de oppervlakken van de doorsnede steeds korter bij dan de andere.

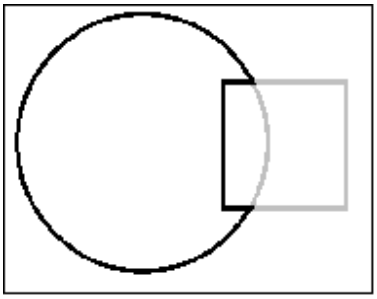
Het intersectie algoritme berekent vervolgens het SCP van beide deelbomen en kiest er een. Meestal is dat het SCP dat zich *het dichtst bij de positie van de gebruiker bevindt*, tenzij dit SCP te ver verwijderd is van het vorige SCP. Deze extra controle dient om pop-through effecten te voorkomen wanneer de gebruiker teveel druk uitoefent op het haptisch toestel.

Het pop-through effect kan echter alleen volledig vermeden worden als het haptisch renderalgoritme van de primitieven de pop-through effecten volledig vermijdt. Het intersectie algoritme gebruikt bovendien niet het vorige SCP voor de algemene controle, zoals gangbaar is bij SCP-berekeningen, maar de huidige gebruikerspositie. Dit is om *bevingen* (jitter) te vermijden bij overgangen tussen linker en rechter deelboom. In Figuur 44 geeft dit geen probleem omdat dit scherpe hoeken zijn, maar bij zachtere overgangen is dit merkbaar.

6.4.2 Subtractie

De subtractie van een object met een ander bestaat uit het volume dat door het eerste object wordt ingenomen, maar niet door het tweede object (zie Figuur 45). Enkel dit gedeelte mag door de gebruiker gevoeld worden en dus dient men enkel van dit gedeelte het SCP te berekenen. Bevindt de gebruiker zich *buiten* de subtractie, dan hoeven er *geen* krachten berekend te worden.

Er moet gecontroleerd worden of de positie van de gebruiker zich binnen de linker deel-boom bevindt en ook tegelijkertijd absoluut *niet* in de rechter rechter deel-boom zit.



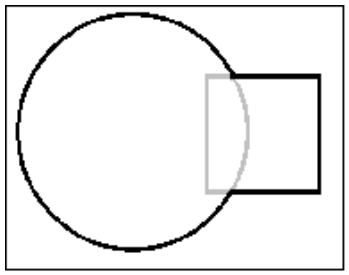
Figuur 45: Subtractie van een object met een ander

Probleem dat hoeken en randen niet hard aanvoelen komt bij subtractie niet voor. Het rechter object is namelijk steeds een primitieve, aangezien we steeds met genormaliseerde CSG bomen werken. (zie §2.7) De overgang tussen oppervlakken van de linker en de rechter deel-boom is dus nooit zacht. Pop-through effecten worden bijgevolg reeds vermeden en het nieuwe SCP kan berekend worden aan de hand van het SCP uit de vorige iteratie.

In tegenstelling tot het algoritme voor intersecties, wordt hier van het gesubtraheerd object de *binnenkant* gerenderd. Hiervoor waren nieuwe algoritmes nodig zoals reeds besproken in §6.3.

6.4.3 Unie

De *unie* van twee objecten bestaat uit het volume dat door één of beide objecten wordt ingenomen (zie Figuur 46). Indien de gebruiker zich dus bevindt binnen het volume van de linker of rechter deel-boom, dient het overeenkomend SCP berekend te worden.



Figuur 46: Unie van twee objecten

Wanneer de gebruiker zich bevindt in slechts één van de deel-bomen, dan wordt het SCP van die deelboom gebruikt. Bevindt dat SCP zich vervolgens in het andere object, dan gebruikt men het dichtstbijzijnde SCP. Het SCP van de andere deelboom wordt *niet* in overweging genomen omdat dit SCP zich ook helemaal aan de andere kant van dat object zou kunnen bevinden, wat weer voor pop-through effecten zou zorgen.

Soms kan geen enkel van beide SCP's gevonden worden. In dat geval moet het algoritme van Raymaekers en Van Reeth [14] een heuristiek gebruiken om het exacte SCP te berekenen. Dit kan gemakkelijk gerealiseerd worden door gebruik te maken van de specifieke eigenschappen van de primitieven. Dit kan echter niet gebruikt worden omdat deze informatie niet voorzien bij de operaties in de structuur van CSG-bomen. Het druist ook in tegen het opzet van deze algoritmes, die onafhankelijk van de geometrie van de primitieven proberen te werken. Bovendien zou het te veel verwerkingskracht kosten.

Er wordt bijgevolg volgende heuristiek toegepast:

- Een mogelijk SCP wordt berekend tussen de gebruikerspositie en het vorige SCP.
- Dit SCP wordt als het nieuwe SCP gebruikt en hierop worden dezelfde berekeningen gemaakt het vorige SCP.
- Indien dit een oplossing geeft hebben we ons nieuwe SCP; indien niet, dan geeft het algoritme het zoeken van het nieuwe SCP op en gebruikt het vorige SCP. Er is namelijk geen tijd meer om verder te zoeken.

Dit heuristisch algoritme geeft dus niet altijd de goede oplossing, maar het is een benadering die door de gebruiker niet of nauwelijks opgemerkt wordt.

6.5 Haptisch normaliseren

Om gerenderd te kunnen worden met het haptisch renderalgoritme van Raymaekers en Van Reeth [14] dient een CSG boom ook op voorhand genormaliseerd te worden. De vereisten voor haptisch normaliseren zijn minder streng dat zijn grafische tegenhanger aangezien de vereiste om de U operatoren vanboven in de boom te plaatsen hier wegvalt.

Werking:
Men dient volgende productieregels herhaaldelijk toe te passen om een *haptisch genormaliseerde CSG boom* te bekomen:

```
1.    X – (YUZ)    -->    (X–Y) – Z
2.    X – (Y∩Z)    -->    (X–Y) U (X–Z)
3.    X – (Y–Z)    -->    (X–Y) U (X∩Z)
```

Deze productieregels vormen een subset van de *'gewone'* normalisatieregels uit §3.4.2. en al werkt het haptisch renderalgoritme ook met de *'gewone'* regels, toch is het beter deze te gebruiken. Minder regels zorgt namelijk voor een snellere normalisatie en voegt minder nodes toe aan de boom.

Ook bij haptische normalisatie worden extra takken aan de CSG boom toegevoegd en hangt dit aantal af van de *structuur van de boom* en de *geometrie van de primitieven*. Daarom dient de CSG boom ook na normalisatie *gesnoeid* te worden. De regels die hiervoor gebruikt worden zijn echter identiek aan de algemene regels voor het snoeien van CSG bomen (§3.4.3).

6.6 Conclusie

Dit hoofdstuk van de thesis omschrijft de methode van Raymaekers en Van Reeth [14] die nodig is om CSG objecten haptisch te renderen. Er werd een overzicht gegeven van hoe deze methode te werk gaat in paragraaf 6.2. Hier wordt het duidelijk dat om het haptisch renderen van CSG bomen mogelijk te maken, haptische renderalgoritmes nodig zijn van de primitieven, zowel voor de buiten- als binnenkant (§6.3).

De tweede stap was het uitvoeren van een inside-outside test, die moest kijken of de gebruiker zich binnen of buiten een deelboom bevindt en naargelang deze uitkomst en de Booleaanse operator van de betreffende tak, het SCP bepaalt (§6.4). Op deze manier is het renderen van haptische CSG bomen met de convexe primitieven *sfeer*, *kubus* en *cilinder*, een realiteit geworden.
Tenslotte bleek dat het normalisatieproces om CSG bomen haptisch te renderen minder rigoreus is dan dat voor grafische rendering (§6.5). Dit voordeel houdt de boom compacter en maakt duidelijk dat men beter aparte representaties maakt van de ingeladen CSG boom voor grafisch én haptisch renderen waardoor beide apart geoptimaliseerd kunnen worden voor hun specifieke noden.

Het spreekt voor zich dat de algoritmes om CSG bomen haptisch te renderen het sleutelelement zijn in de bouw van een haptische CSG modeller aangezien deze CSG objecten nu in realtime tegelijkertijd zowel grafisch als haptisch gerenderd kunnen worden.

Hoofdstuk 7 Haptisch modelleren van CSG bomen

7.1 Inleiding

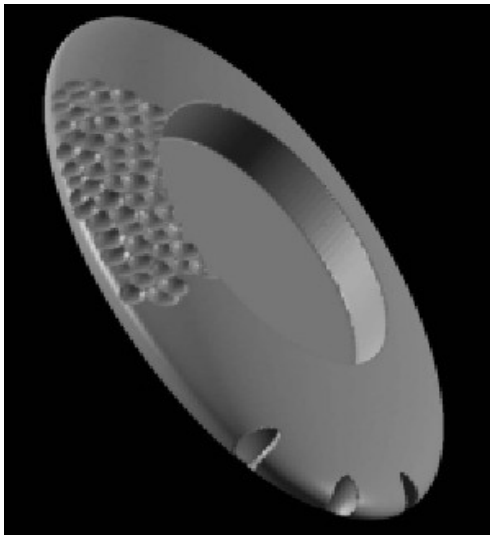
In hoofdstuk 6 werden de nodige algoritmes beschreven om CSG objecten haptisch te modelleren. Hiermee zijn alle noodzakelijke stappen gezet voor het maken van een haptische CSG modeller. Dit hoofdstuk geeft een algemeen overzicht over hoe haptisch modelleren met CSG bomen kan gebeuren.

Eerst wordt het haptisch modelleren onder de loupe genomen: er wordt gekeken waaraan deze moet voldoen en hoe zo'n interactiecyclus verloopt (§7.2). Daarna spitsen we ons toe op het haptisch modelleren van CSG objecten en kijken we welke haptische interacties haalbaar zijn (§7.3). Dan volgt er nog een overzicht en indeling van haptische tools (§7.4), die telkens een interactieve modelleeractie moeten voorstellen (§7.5).

Aangezien haptische interacties via een CSG object grotendeels vereenvoudigingen zijn van interactieve bewerkingen uit de realiteit, wordt tenslotte een overzicht gegeven van deze abstracties, en of deze ontstaan zijn uit handigheid dan wel uit noodzaak (§7.6).

7.2 Haptisch modelleren

Interactieve modelleeromgevingen maken tot op heden meestal gebruik van *toetsenbord en muis* als input en genereren uitsluitend *visuele feedback*. Deze feedback bestaat uit het tonen van een *visuele too*l en de door deze tool ontstane *aanpassingen aan de scène*. Haptisch modelleren voegt *zowel qua input als feedback een extra dimensie toe aan een modeller door objecten te te bewerken en waar te nemen*.



Figuur 47: Beitel effect op een schotel, gekerfd met een haptisch toestel. [12]

Haptics verhogen bijgevolg het gevoel van *immersie* dat reeds aanwezig is in de visuele component aanzienlijk. Door het voelbaar maken van de objecten, kan de gebruiker op een meer intuïtieve manier interageren met de objecten uit de scène. Hierdoor worden toepassingen zoals virtueel beeldhouwen (figuur 47) of boetseren met klei mogelijk.

7.2.1 Vereisten

Om zowel zinvolle krachtterugkoppelingen, haptische visualisatie als wijzigingen van de 3D modellen in een modelleeromgeving samen te laten werken tegen een tempo van minstens 1kHz, moet er voldaan worden aan enkele belangrijke voorwaarden [39]:

- **Een constant haptisch update tempo**: Grote variaties in het tempo waarin de krachten geüpdate worden, kunnen misleidende tactiele informatie produceren.
- **Snelle berekening van de krachten**: Complexe berekeningen van krachten zouden het haptisch update tempo vertragen en daardoor de verwerkingstijd die nodig is voor het grafisch renderen en het doorvoeren van wijzigingen aan de objecten.
- **Snel en incrementeel renderen**: Interactieve rendertempo's zijn nodig om de lokatie haptische pointer visueel correct op beeld te brengen. Ook moet er van de tijd die het visueel renderen, dat zich voordoet gespreid over enkele haptische iteraties, in beslag neemt, een deel afgelost worden om een consistent haptisch update tempo te hebben.
- **Snelle wijzigingen van data**: Het tempo van de datawijzigingen moet ook interactief zijn, zowel voor de visuele als haptische feedback.
- **Consistent renderen van de haptische en visuele componenten**: Deze moeten uiteraard consistent zijn met elkaar, ook al hebben beide sterk verschillende update tempo's.

Bovendien zal er rekening gehouden moeten worden met typisch haptische problemen zoals *doorduweffecten*, die onrealistisch zijn en voor problemen kunnen zorgen tijdens het interactief haptisch modelleren.

Alle interactieve haptische acties die in een modelleeromgeving gebruikt worden, zullen moeten voldoen aan deze voorwaarden om stabiel en dus werkbaar te zijn.

7.2.2 Interactiecyclus

De *haptische interactiecyclus* begint nadat het te bewerken object ingeladen is. Eerst worden de huidige positie en oriëntatie van de haptische pointer opgevraagd. Het haptisch object kan dan bewerkt worden met behulp van een *'tool'*, bijvoorbeeld een haptische beitel. Indien de tool het CSG object bewerkt, dringen zich wijzigingen op aan het object. Visueel moet enkel geupdated worden indien deze wijzigingen te zien zijn. Haptisch updates echter, moeten *onmiddelijk* uitgevoerd worden aangezien deze reeds op de volgende iteratie invloed kunnen hebben om wille van de inside-outside test (§5.5). Tenslotte worden de huidige krachten berekend en verstuurd naar het haptische toestel.

Hierbij wordt de haptische interactie steeds uitgevoerd in een *aparte thread* die minstens 1000x per seconde aangeroepen wordt vanuit het haptisch toestel. Deze thread controleert of de pointer zich binnen of buiten het object bevindt en berekent telkens een passend SCP (zie §5.5).

Worden wijzigingen aan de scène niet onmiddellijk aangebracht, dan ontstaan er ongewilde jitter effecten. Deze operaties vergen extra berekeningen, die gezien het hoge update tempo best zo kort mogelijk gehouden worden. Ze komen echter niet voor tijdens iedere iteratie van de haptische interactiecyclus.

Het tempo waaraan het visuele beeld geüpdated wordt, is ook beperkt in de tijd. In dit geval wordt er uitgegaan van een tempo van 30Hz aangezien een groter tempo niet nodig is voor het menselijk oog.

7.3 Haptisch modelleren met CSG objecten

Interactieve haptische modelleeracties wijzigen bestaande objecten. Deze wijzigingen moeten snel en efficiënt kunnen verwerkt worden. Het spreekt daarom voor zich dat enkel snel en efficiënt uit te voeren wijzigingen in aanmerking komen bij het haptisch modelleren. De aard van deze wijzigingen hangt af van de representatie die gekozen werd om deze 3D objecten voor te stellen, in casu CSG.

Zoals we reeds meermaals zagen, is de structuur van CSG een aaneenknoping van booleaanse operatoren en primitieven. Object wijzigingen die met CSG bomen relatief makkelijk verwezenlijkt kunnen worden, zijn dan ook operaties zoals een holte of een gat maken in een object (subtractie met een sfeer of een cilinder), samenvoegen van objecten (via een unie van twee CSG bomen), etc. Zulke bewerkingen zijn dan ook het meest aangewezen bij gebruik in een modeller, op voorwaarde dat de interactie op de tool voorgesteld kan worden via een puntkracht (zie §7.5.3).

Tot wijzigingen zoals freeform deformaties lenen CSG bomen zich daarentegen absoluut niet. Er wordt dan beter voor een andere representatievorm gekozen, zoals B-rep (zie ook Hoofdstuk 2).

7.3.1 Aparte grafische en haptische CSG bomen

Het doorvoeren van wijzigingen aan CSG bomen gaat enorm snel. Zo is het realiseren van bijvoorbeeld de doorsnede tussen twee CSG objecten T_1 en T_2 niet meer dan een nieuw CSG object T waarbij $T = T_1 \cap T_2$. Helaas kunnen deze objecten meestal niet gerenderd worden zonder voorafgaande normalisatie. De vereisten qua normalisatie zijn bovendien strenger voor het grafisch dan voor het haptisch renderen waardoor dit langer duurt en meer nodes creëert. Gelukkig moet er grafisch minder vaak gerenderd worden dan haptisch gezien de verschillen in update tempo's, waardoor beide normalisatievormen haalbaar zijn.

Indien we echter dezelfde CSG boom voor zowel grafisch als haptisch renderen zouden gebruiken, is het onmogelijk deze voor beide render-methodes optimaal te laten werken. Daarom dient een haptische CSG modeller gebruik te maken van twee CSG bomen: één voor de grafische en één voor de haptische operaties. Deze bomen worden *apart gerenderd en genormaliseerd* waardoor ze sterk gaan verschillen. *Toch stellen ze eenzelfde*

CSG object voor. CSG representaties zijn immers niet uniek (zie §2.4). De twee bomen blijven steeds consistent aan elkaar, of toch voor en na het uitvoeren van een haptische interactie. Tijdens een haptische interactie kunnen beide namelijk inconsistent zijn aan elkaar: haptische updates gebeuren immers constant en aan een hoog tempo, terwijl grafische updates enkel gebeuren wanneer de aangebrachte wijzigingen zichtbaar zijn.

7.4 Tools

Net zoals in een 2D tekenprogramma of een 3D CAD omgeving, wordt een scène ook in een haptische 3D omgeving opgebouwd en gewijzigd door gebruik te maken van *tools*.

Deze tools moeten *intuïtief* zijn: ze mogen niet de indruk geven dat er bijvoorbeeld een cilinder gesubtraheerd wordt van een CSG boom, maar dat er met behulp van een haptisch toestel een gat geboord is in een 3D object. Zo'n tool is dus meestal een *metafoor* die een bewerking met *reëel gereedschap* representeert en daarmee CSG objecten door middel van haptische interactie modelleert. Zo zal bijvoorbeeld de diepte van een geboord gat bepaald worden door de op de haptische tool uitgeoefende kracht in de boorrichting.

7.4.1 Indeling

Men kan de haptische tools die gebruikt worden bij het interactief bewerken van 3D objecten verdelen in twee categorieën: de ***discrete*** en de ***continue*** bewerkingen:

- Bij een *discrete of enkelvoudige bewerking* bestaat de interactie uit *één enkele actie* zoals bijvoorbeeld *beitelen* (zie ook Figuur 47): de gebruiker duwt de beitel met een bepaalde kracht in een bestaand CSG object. Er verschijnt een *uitkerving* die overeenkomt met *de aard en diepte van de 'inslag',* waarna de bewerking is voltooid. Een concrete uitwerking van een interactieve beitel inclusief implementatie kan u vinden in Hoofdstuk 9.

- *Continue of meervoudige bewerkingen* bestaan uit meer dan één contact zoals het boren van een gat. Hier is de bewerking *niet voltooid na de inslag, maar slechts het begin ervan*: de gebruiker bepaalt interactief de diepte en boort tot het geboord gat naar zijn mening diep genoeg is. Tijdens het boren voelt de gebruiker bovendien niet alleen dat hij kracht moet zetten om dieper te boren, hij voelt ook de boortool in het object zitten. Pas wanneer de gebruiker zijn boor terug uit het gat haalt, is het interactief boren klaar. Men kan argumenteren dat een meervoudige bewerking niet meer is dan een concatenatie van enkelvoudige bewerkingen. Dit kan in bepaalde gevallen wel voldoen, maar vaak leidt het tot problemen, zowel op gebied van performantie als het aanvoelen. Wat er allemaal komt kijken bij een meervoudige haptische CSG interactie, wordt bestudeerd aan de hand van de concrete uitwerking en implementatie van een interactieve boor tool. De resultaten hiervan leest u in Hoofdstuk 8.

7.5 Abstracties

Aangezien men de realiteit nooit helemaal kan kopiëren, is men in virtuele omgevingen steeds verplicht om bepaalde abstracties te maken. Zo zullen haptische interacties zoals boren in een CSG object, nooit helemaal op dezelfde manier gebeuren als wanneer er met een echte boor in bijvoorbeeld een muur wordt geboord. Zo treedt er bij het boren in een muur onder andere zijdelingse wrijving op, is het gat dat uitgeboord wordt niet altijd even glad, kunnen barsten optreden, is een object niet homogeen, etc... Sommige van deze randeffecten uit de realiteit kunnen achteraf indien gewenst nog verwezenlijkt worden om de interacties *'realistischer'* te maken.

7.5.1 Realisme of handigheid?

We kunnen ons hierbij ook afvragen of we wel een zo realistisch mogelijke haptische interactie *willen*. Streven we naar *realisme* of willen we interactie die *vooral handig* is en de bewerkingen aan het object mogelijk maakt zoals ze door de gebruiker bedoeld zijn?

Maken we dus bijvoorbeeld een *echte boor* of een *superboor*? Een echte boor heeft namelijk een *maximale boorlengte*, kan niet *'afgezet'* worden, kan soms moeilijk uit het boorgat getrokken worden, er moet soms heel hard tegen geduwd worden,... Al deze moeilijkheden zal men met de haptische boor die in het volgend hoofdstuk beschreven wordt, *niet* hebben. Dit zijn dus nuttige vereenvoudigingen.

7.5.2 Interactieve tools.

Ook de tools voor haptische interactie komen niet overeen met de realiteit. Dit is echter vaak geen keuze, maar een noodzaak.

Figuur 48: De tools uit de CSG Modeller: de pointer (boven), de boor (midden) en de beitel (onder).

Bekijken we de tools uit Figuur 48, dan zien we in het midden de *virtuele boor*. Waar een echte boor van voren spitser is en gedraaid zoals een schroef, bestaat de haptische boor enkel uit een stompe cilinder. Met zo'n boorkop is het in de praktijk zeer moeilijk om te boren. Ook zal het uitgeboord gat in de realiteit vanvoren wat ver-smallen in plaats van perfect cilindervormig te zijn. Er werd namelijk gekozen om enkel haptische interacties te doen met tools die de vorm van een *primitieve* hebben, om zo de CSG operaties niet te complex te maken.

Onderaan Figuur 48 bevindt zich de beitel. Ook deze boorkop is eenvoudigweg een 45° geroteerde cilinder waardoor er ook hier enkel interactie is met een primitieve.

7.5.3 Puntkrachten

Haptische CSG tools zijn vaak beperkt omdat ze qua interactie eigenlijk slechts voorgesteld kunnen worden door middel van een puntkracht, namelijk deze van het haptische toestel. In tegenstelling tot de interactieve tools heeft de pointer dus geen breedte. Hieraan zullen we soms tegemoet kunnen komen, bijvoorbeeld door gebruik te maken van een haptische spelingcilinder in plaats van een boorgat op volle breedte (zie §8.5.4). Toch zullen ten gevolge van het gebruik van een puntkracht steeds abstracties noodzakelijk blijven.

Bovendien is het interactiepunt uniek, waardoor de terugkoppelingskrachten die het gevolg zijn van de tool's penetratie in een CSG object, steeds berekend worden op basis van de interactie van dat ene punt. Sommige interactieve tools zoals bijvoorbeeld het duwen van een kubus in een bestaand CSG object, hebben echter nood aan een collision detectie op meerder plaatsen en vergen daardoor een veelvoud aan inside-outside tests. Dit zou de haptische load al snel verveelvoudigen waardoor het moeilijk haalbaar is.

7.6 Conclusie

Hoofdstuk 7 gaf een algemeen overzicht over hoe haptisch modelleren met CSG bomen kan gebeuren en vormt daarmee een soort algemene inleiding voor de twee volgende hoofdstukken.

Eerst kwam er een overzicht van een haptische modeller (§7.2). Er werden vereisten genoemd waaraan zo'n modeller moet voldoen en het verloop van een interactiecyclus werd beschreven. Daarbij viel op dat de grafische en haptische component eigenlijk volledig apart behandeld werden. Dit bleek alvast ook bij het haptisch modelleren via CSG objecten (§7.3): aparte voorstellingen van de grafische en haptische bomen werden immers noodzakelijk.

Net zoals in andere modelleeromgevingen, wordt ook bij haptische CSG modellers gebruik gemaakt van tools als metaforen voor echte interactieve operaties zoals haptisch boren of beitelen. Deze tools werden beschreven in §7.5.

Tenslotte volgde een overzicht van de abstracties die de virtuele tools hebben ten opzichte van tools uit de realiteit (§7.6). We leerden hier dat sommige abstracties een bewuste keuze waren en andere absoluut noodzakelijk om haalbare haptische interactie mogelijk te maken.

Dit hoofdstuk voorzag een algemene benadering, maar gaf geen gedetailleerde beschrijving van enkelvoudige (discrete) en meervoudige (continue) haptische modelleertools. Deze worden verder in deze thesis uitgebreid bestudeerd aan de hand van twee concrete tools voor haptische interactie: de meervoudige boor in Hoofdstuk 8 en de enkelvoudige beitel in Hoofdstuk 9.

Hoofdstuk 8 Interactief boren in een CSG object.

8.1 Inleiding

Tools die continue of meervoudige operaties in CSG objecten mogelijk maken zijn essentieel bij in een haptische interactief CSG modelleeromgeving. Ze laten ons bijvoorbeeld toe om niet alleen een bepaald deel uit zo'n object te verwijderen, maar ook om tijdens het modelleren heel wat parameters zoals grootte, plaats, richting,... te bepalen. Deze zijn ingewikkelder dan de operaties die slechts uit één actie bestaan, zoals beitelen, maar zijn dan ook uitgebreider en benutten de mogelijkheden van haptische interactie ten volle.

Er zijn heel wat van deze continue operaties te bedenken zoals frezen, graven, snijden,... maar de meest representatieve continue operatie is ongetwijfeld boren. Dit hoofdstuk presenteert een haptische tool waarmee interactief in CSG objecten geboord kan worden. Aan de hand van deze tool wordt onderzocht hoe zo'n continue haptische tool concreet verwezenlijkt kan worden en welke abstracties gemaakt moeten worden. Eerst bekijken we stapsgewijs hoe een reële booroperatie verloopt (§8.2). Daarna gaan we na hoe men een boorgat creëert (§8.3) en bekijken we de nodige CSG operaties (§8.4).

Aangezien er bij het boren in een CSG object heel wat komt kijken, behandelen we eerst haptisch boren in ideale omstandigheden (§8.5), en behandelen daarna de verschillende problemen (§8.6). Deze tonen aan dat rechtstreeks boren in een haptisch object niet altijd voldoet. Daarom stellen we een betere oplossing voor, waarbij concreet geboord wordt in een speciaal voor de booroperatie gecreëerd voorlopig haptisch object (§8.7). Hoe deze methode tegemoet komt aan de problemen wordt uitgelegd in (§8.8).

Tijdens het hoofdstuk wordt meermaals aandacht besteed aan de performantie van dit interactief boren, de mogelijke optimalisaties, de nodige CSG operaties, en aan zowel de visuele als de haptische feedback.

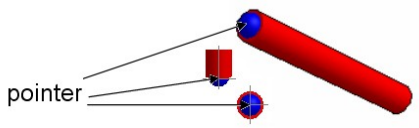
Deze tool werd in het kader van deze thesis ook effectief geïmplementeerd. De bedoeling hiervan is om een realistisch, concreet inzicht te verwerven op gebied van continue haptische tools en in welke mate zulke operaties performantiegewijs haalbaar zijn. De figuren die bij dit (en het volgende) hoofdstuk horen, zijn dan ook bewerkte screenshots uit deze implementatie.

8.2 Algemeen

Het boren van een gat in een object is een *continue* of *meervoudige* interactie tussen een tool, de boor, en een bestaand object. Meestal verloopt het boren in een object als volgt:

- Eerst wordt gekeken op welke *plaats* er ingeboord moet worden, in welke *richting* en met welke *boorbreedte*.
- Daarna wordt de boor *vastgenomen en aangezet* of *geselecteerd*.
- Door in de juiste richting druk te zetten op de boor, ontstaat een boorgat dat groter wordt telkens er genoeg druk wordt gezet in de boorrichting.
- Het boren is pas beëindigd wanneer
  - ofwel de boor nog in het gat zit, maar afgezet of gedeselecteerd wordt,
  - ofwel de boor uit het boorgat gehaald wordt.

Net zoals in de realiteit bevat het object nu een boorgat dat overeenkomt met uitgeoefende interacties. Waar dit gat in de realiteit bestaat uit gruis dat zich tijdens het boren uit het boorgat verwijderd heeft of zich nog in de rand ervan bevindt, wordt dit in de CSG modeller voorgesteld door een van het CSG object gesubtraheerde cilinder. Voor de gebruiker komt het echter over als een 3D object waar hij of zij een gat in heeft geboord.



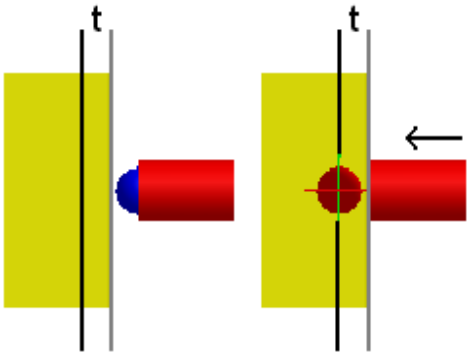
Figuur 49: Plaats van de puntkracht op de haptische boor.

De in dit hoofdstuk voorgestelde virtuele boor maakt gebuik van haptische interactie in vijf actieve vrijheidsgraden: drie voor beweging en twee voor het bepalen van de boorrichting.

8.3 Creëren van een boorgat

Wanneer de boortool geselecteerd wordt, dan verandert de pointer in een boor.
Bij selectie van de boortool, krijgt de pointer de vorm van een boor, hier voorgesteld door een rode cilinder.

De pointer die voor de haptische interactie zorgt bevindt zich in het midden van de boorkop (zie Figuur 49 en 50, links). Tijdens het boren bevindt de boorkop zich echter rond het SCP (zie Figuur 50, rechts). Haptisch gezien duwt de boor dus tegen het object zodra de pointer zich in het object bevindt.



Figuur 50: Positioneren van de boor (boven). Na overschrijden van de threshold begint zich het boorgat te vormen (onder).

Na bepaling van plaats en richting wordt de boortool tegen het object gedrukt (zie Figuur 50, links). Wanneer de gebruiker voldoende kracht uitoefent op het oppervlak van het CSG object, vormt zich een boorgat en start het boorproces.

De grootte van de kracht die nodig is om een boorgat te creëren, wordt gerealiseerd via een threshold, die in principe overeenkomt met de hardheid van het object (afstand tussen beide verticale lijnen in Figuur 50). Bij overschrijding van deze threshold wordt boorgat gecreëerd.

8.4 CSG Operaties

Boren brengt enkele nodige aanpassingen aan het CSG object in de vorm van de subtractie van het boorgat. Deze paragraaf zet de verschillende CSG operaties die nodig zijn bij het boren even op een rijtje.

8.4.1 Subtractie van het boorgat.

Een boorgat heeft de vorm van een *primitieve cilinder* waarvan *de hoogte overeenkomt met de diepte van het boorgat* en *de radius met de breedte van de boor*. We kunnen dus een boorgat creëren door van het CSG object een cilinder te subtraheren.

Zoals eerder reeds gezien (zie §7.3.1), dient een haptische CSG modeller een aparte CSG boom bij te houden voor haptisch en grafisch renderen, die beide gesubtraheerd moeten worden met een gelijkvormoge cilinder.

Tijdens het modelleren hoeft de structuur van de boom echter niet meer aangepast te worden, maar volstaat het om de lengte van de reeds aanwezige gesubtraheerde cilinders aan te passen telkens het boorgat vergroot. Hierover meer in §8.5.

8.4.2 Normalisatie

Grafisch

Om na creatie van een boorgat terug in normaalvorm te staan, moet de grafische CSG boom genormaliseerd worden, wat heel wat extra tijd in beslag neemt en extra nodes creëert. We bevinden ons echter in het specifieke geval waarin de nieuwe grafische boom T_{nieuw} bestaat uit de oude, genormaliseerde CSG boom T_{oud} gesubtraheerd met een primitieve cilinder C. Met andere woorden: $T_{nieuw} = T_{oud} - C$.

Dit betekent concreet dat deze subtractienode naar beneden in de boom wordt gebracht aangezien de U operatoren zich vanboven in de boom moeten bevinden (zie §3.4.2). Men krijgt dan een nieuwe genormaliseerde boom T_{nieuw} waarbij alle nodes R die kind zijn van een U-operator, vervangen worden door een subtractienode S met R als linker en C als rechterkind.

voor alle R kind van U-operator node: R --> (R-C)

Men kan dus specifiek gaan normaliseren door eenvoudigweg alle R-nodes op te zoeken en te vervangen met behulp van de productie uit het kadertje hierboven. Dit algoritme verloopt in real-time. Oók het opzoeken aangezien alle U nodes zich steeds vanboven in de boom bevinden. Het aantal nodes in de boom stijgt wel met 2xR.

Haptisch

Normalisatie van haptische bomen is meer een sub-normalisatie, waarin de verplichting om U-operatoren vanboven in de boom te hebben niet bestaat. Een genormaliseerde haptische boom is dus na het subtraheren met een cilinder nog steeds genormaliseerd. Inderdaad, want geen van de drie productieregels kan doorgevoerd worden op een boom van het type T-C (zie §6.5). Haptisch normaliseren is dus niet nodig.

Conclusie

Ook al lopen de berekeningen voor het aanmaken van het boorgat en het opnieuw grafisch normaliseren vrij lineair, toch is het beter deze acties reend uit te voeren wanneer de boor geselecteerd wordt en er nog geen haptische load is (omdat het de pointer zich bij selectie van de boortool niet in het object bevindt). Er wordt dan gewoon een cilinder met hoogte nul aan de scène toegevoegd.

8.4.3 Meervoudige nodes

Uit de vorige paragraaf blijkt dat de grafische boor voor heel wat extra nodes kan zorgen. Hiervan zijn alle C-primitieven exact dezelfde. Ze moeten bovendien allemaal aangepast worden telkens de boor dieper in het object binnendringt, wat onnodig tijd kost.

Daarom wordt de C-primitieve slechts één keer wordt gedefinieerd en daarna gekoppeld aan alle bijgecreëerde subtractienodes. Dit spaart geheugen uit en zorgt voor minder bewerkingen bij het verlengen van het boorgat (zie §8.5). Gebruik van zulke meervoudige nodes zorgt bovendien achteraf niet voor problemen: ze kunnen gewoon deel blijven uitmaken van de grafische CSG boom.

8.4.4 Snoeien

Na het normaliseren van de grafische boom volgt meestal een snoeioperatie. Dit is tijdens het interactief modelleren echter niet mogelijk. Het snoeialgoritme snoeit namelijk op basis van niet-intersecterende bounding volumes (zie §3.4.3). Aangezien het boorgat tijdens het modelleren van grootte verandert, wijzigen de bounding volumes eveneens en kan er niet gesnoeid worden tot het boorproces eindigt en de definitieve vorm, richting en plaats van het boorgat bepaald zijn.

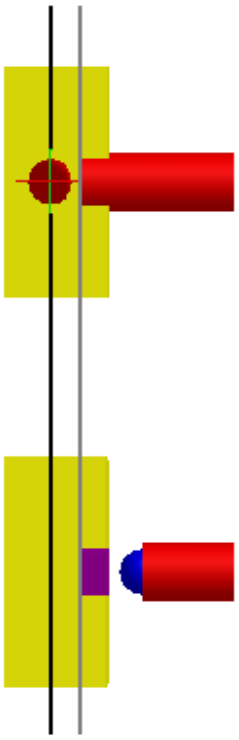
8.5 Loodrecht inboren midden in een groot object

Deze paragraaf behandelt het haptisch boorproces in ideale omstandigheden, namelijk loodrecht in het midden van een object dat voldoende groot is.

8.5.1 verloop

Na creatie van het boorgat (§8.3), wordt de gesubtraheerde cilinder die het haptisch boorgat voorstelt telkens vergroot indien de gebruiker de treshold overschrijdt. Het boorgat wordt dan verlengd tot de threshold net niet meer overschreden is, om zo jitter effecten te vermijden.

De figuur hiernaast verduidelijkt het verloop: Figuur 51, boven, toont een situatie waarbij de gebruiker de pointer van de boor doorduwt tot aan de zwarte lijn en daar stilhoudt. De boor zelf, en het boorgat zijn gevolgd tot aan de grijze lijn, dus op een afstand tot de pointer ter grootte van de threshold. In Figuur 51, beneden, wordt dan het resultaat zichtbaar na retractie van de boor.



Figuur 51: Hoe dieper de pointer inboort, hoe groter het boorgat wordt (boven). Bij retractie van de boor wordt het boorgat zichtbaar (onder).

Vanuit de situatie van Figuur 51, kan de gebruiker twee richtingen uit:

- hij of zij kan harder blijven duwen dan de kracht die nodig is om de threshold afstand te bereiken.
  --> Het boorgat wordt dan vergroot, waarbij de afstand tussen gebruiker's positie en SCP constant net kleiner dan threshold gehouden wordt.
- Hij of zij kan minder hard gaan duwen dan ervoor.
  --> Er wordt niet dieper geboord. De pointer verplaatst zich terug richting boorgat, maar de boor blijft tot achteraan in de boorholte steken tot de gebruiker volledig stopt met duwen. Pas wanneer hij of zij daarna de pointer van het haptisch toestel verder naar de ingang van het boorgat brengt, zal de boor zich gaan terugtrekken.

8.5.2 Updates

Telkens de threshold overschreden is, wijzigt het boorgat en dus ook het object en moet indien nodig zowel grafisch als haptisch opnieuw gerenderd worden zodat het na elke update visueel en gevoelsmatig overeenkomt met de huidige staat van het CSG object.

Grafisch

Bij elke update moet de grafische boom volledig opnieuw geclassificeerd en gerenderd worden (zie §3.4) om bij te blijven met zijn haptisch equivalent. Dit kost heel wat rekenkracht, zeker wanneer de gebruiker aan het boren is en er elk frame opnieuw gerenderd moet worden.

Om zo'n bottleneck te vermijden, is het best om het grafisch boorgat enkel te updaten en opnieuw te renderen indien er een zichtbaar *verschil* is ten opzichte van het vorige beeld. Zo'n zichtbaar verschil komt tijdens het inboren eigenlijk helemaal niet voor aangezien het boorgat meteen opgevuld wordt door de *boor zelf* (zie Figuur 51, boven). Zelfs als de CSG objecten gedeeltelijk doorzichtig zouden zijn, zal het boorgat pas zichtbaar zijn vanaf het moment dat de boor effectief een beetje teruggetrokken wordt (zie Figuur 51, onder).

We moeten tijdens het interactief modelleren de grafische CSG boom dus enkel updaten en renderen wanneer er :

- een verlenging van het boorgat is geweest en
- de boor wordt teruggetrokken.

Haptisch

Voor de haptische representatie vormt dit hoegenaamd geen probleem aangezien de inside-outside test in elke lus opnieuw uitgevoerd moeten worden.

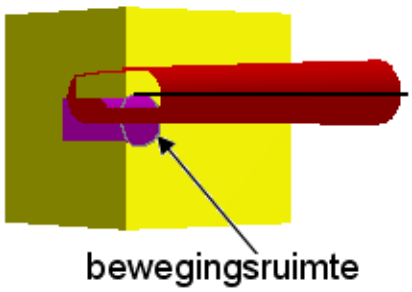
8.5.3 Beëindigen van het boren

Het boren kan zoals eerder vermeld, beëindigd worden op twee manieren, namelijk door een andere tool te kiezen of door de boor uit het boorgat te trekken. De boorgaten krijgen dan hun definitieve afmetingen en kunnen gesnoeid worden. Voor de grafische voorstelling van het CSG object zal dit het aantal nodes kunnen verkleinen. Aangezien normaliseren in de haptische voorstelling geen nodes toevoegde, is haptisch snoeien op dit moment overbodig.

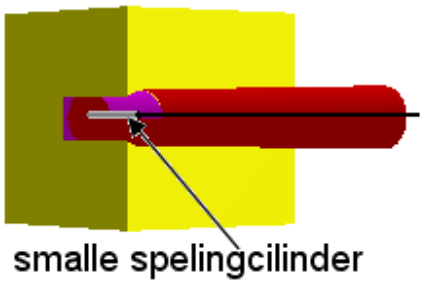
8.5.4 Voelen van een boorgat tijdens het boren.

Tijdens het boren, moet de gebruiker *voelen* dat hij of zij zich ook daadwerkelijk in het boorgat bevindt. Dit gebeurt automatisch door het de updates aan het boorgat, maar aangezien de pointer slechts een puntkracht voorstelt, kan een deel van de boor door het object uit steken (zie figuur).

Om hieraan tegemoet te komen, wordt het boorgat enkel grafisch van zijn echte breedte voorzien (zie figuur 53, onder). Het haptisch boorgat wordt echter heel smal gehouden waardoor het lijkt alsof de gebruiker met zijn brede boor weinig speling heeft binnen het boorgat terwijl hij of zij zich eigenlijk met een puntdikke pointer in een smalle holte bevindt.



Figuur 53: Te veel bewegings-ruimte bij rechstreeks gebruik van een haptisch boorgat op volle breedte.



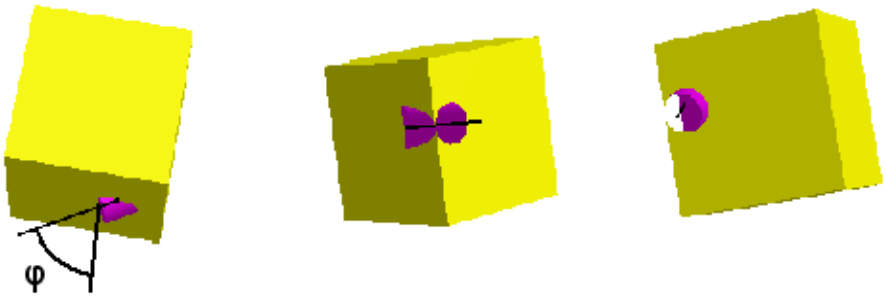
Figuur 52: Slechts beperkte speling bij een ultra smalle voorlopige cilinder als haptisch boorgat.

De haptische spelingcilinder moet net breed genoeg zijn om verplaatsing erdoor mogelijk te maken. Deze speling is bovendien niet onrealistisch aangezien een boor in een boorgat ook steeds een klein beetje speling heeft. Bij een plaatsgevoeligeid van 0.001, kwam in de testapplicatie een boorgatbreedte van 0.003 realistisch over.

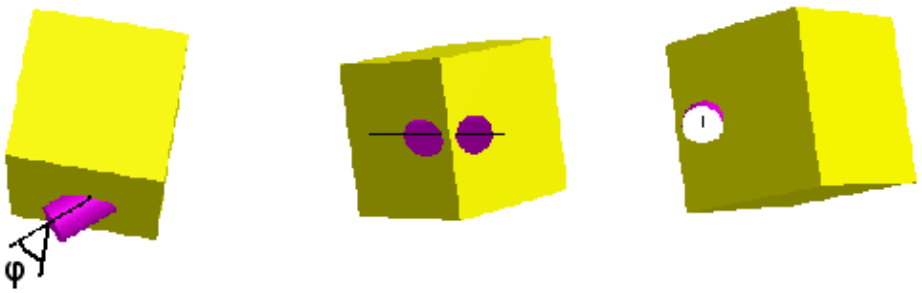
Wanneer het boren beëindigd wordt, dient men alleen nog de radius van de uitgeboorde cilinder te vergroten tot de breedte van de boor, waarna het boorgat door de gewone pointer gevoeld kan worden zoals het eruit ziet. Indien later echter opnieuw een boortool geselecteerd wordt en men opnieuw in het voormalige boorgat gaat, zal men echter niet meer het gevoel krijgen van weinig speling, maar eventueel met delen van de boor door het object kunnen gaan. Problemen zoals deze zijn onvermijdelijk wanneer men een tool met een bepaalde breedte gaat voorstellen door middel van een pointer zonder breedte (zie ook §7.5.3).

8.5.5 Schuin inboren

Tot hiertoe zijn we ervanuit gegaan dat de gebruiker loodrecht op een plat vlak of naar achter gekromd oppervlak boort (bvb een sfeer). Indien Daarentegen schuin wordt ingeboord of in een naar voren gekromd vlak (bvb een gesubtraheerde sfeer) ontstaat een probleem bij het vormen van het boorgat.



Figuur 54: Schuin inboren in een object (links) creëert een te kort boorgat (midden en rechts) omdat de boorcilinder pas bij het middenpunt begint.



Figuur 55: Voldoende marge bij het boren (links) lost dit probleem op en creëert het bedoelde boorgat (midden en rechts).

De gesubtraheerde cilinder positioneert zich namelijk vanaf het punt van binnendringen tot het punt waar er ingeboord werd. Bij het binnendringen in een recht oppervlak komt dit gat overeen met een realistisch boorgat, maar bij een schuin oppervlak is dit gat te kort (zie Figuur 54).

Daarom moet dit gat verlengd worden naargelang de hoek tussen het oppervlak en de richting van de boor. Het berekenen van de hoek waaronder geboord moet worden is bij CSG objecten vrij ingewikkeld, zeker bij naar buiten toe gekromde oppervlakken waardoor het nuttiger gewoon een marge van vaste grootte in te bouwen (zie Figuur 55).

Een marge van twee maal de boorbreedte laat inboren toe tot 60° en lijkt in de meeste gevallen meer dan voldoende, al kan deze marge makkelijk aangepast worden indien de gebruiker onder schuinere hoeken wil inboren. Een zeer grote algemene marge om zeer schuine inboorhoeken mogelijk te maken, is af te raden aangezien te grote boorgaten ook voor grotere bounding boxes zorgen, wat het snoeiproces van CSG nodes naderhand minder efficiënt maakt.

Nog een (kleiner) probleem stelt zich net voordat de gebruiker schuin gaat inboren. Op dat moment kan de rand van de cilinder zich eigenlijk al in het object bevinden terwijl het interactiepunt zich nog buiten het object bevindt. Dit is echter niet te vermijden zonder intersieve collision detectie.

8.5.6 Performantie

Deze methode voegt zowel grafisch als haptisch één (meervoudige) primitieve toe aan de CSG boom en past deze aan naargelang de gebruikerhety boorgat vergroot. De extra berekeningen tegenover *gewoon* haptisch renderen zijn zeker tijdens het modelleren *minimaal*.

selecteren van de boortool: hier worden de grafische en haptische cilinder gecreëerd. Na subtractie van de grafische CSG boom met de cilinder volgt een specifiek geoptimaliseerde normalisatie (zie §8.4.2). De haptische boom moet na subtractie met een primitieve niet opnieuw genormaliseerd worden (zie ook §8.4.2).

tijdens het boren: bij het overschrijden van de threshold wordt enkel de grootte van het de gesubtraheerde primitieve gewijzigd. De grafische boom wordt enkel geupdated bij een zichtbaar resiltaat. Haptisch wordt vervolgens,

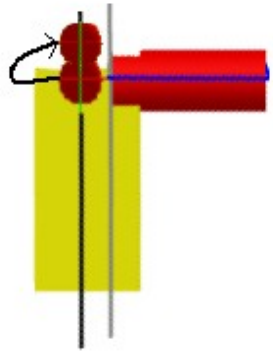
net zoals wanneer geen tool geselecteerd is, een inside-outside test uitgevoerd op het haptische CSG object, waarna het SCP van het haptisch object wordt berekend.

na het boren: na het boren dient enkel het haptisch boorgat zijn definitieve radius te krijgen, en kan de haptische boom gesnoeid worden.

8.6 Problemen van een rechtstreeks haptisch boorgat

Met de in de vorige paragraaf besproken methode kan reeds goed en efficient geboord worden in bestaande CSG objecten. Toch zijn er omstandigheden waarin het boren problematisch zal verlopen en er niet meteen, goede oplossingen te bedenken zijn. In de volgende drie gevallen schiet deze methode te kort.

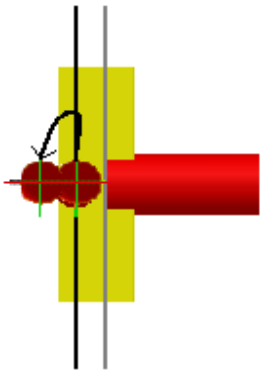
8.6.1 In de rand van een object boren



Figuur 56: Boor schiet gemakkelijk uit het object bij boren bij of in de rand.

Wanneer een gebruiker dichtbij of in de zijrand van een object boort (wat mogelijk is indien de pointer positie bijna op de rand van het CSG object ligt), ontstaat er vaak een *doorduweffect* naar opzij omdat de gebruiker immers nooit in exact dezelfde richting duwt waarin het boorgat zich bevindt. Hierdoor schiet de boor uit het boorgat, wat niet de bedoeling is (zie Figuur 56).

8.6.2 Door een object uitboren



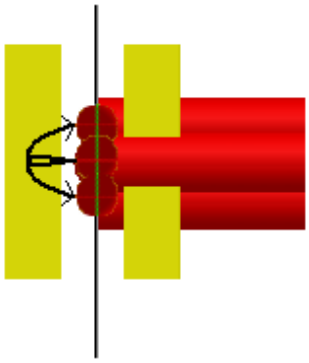
Figuur 57: Pointer schiet door object uit waardoor men niet door een object kan uitboren.

Het is simpelweg niet mogelijk om door een object uit te boren. Wanneer het boorgat zich namelijk zeer dicht bij de achterkant van een object bevindt, en er druk wordt gezet om de gesubtraheerde cilinder door het object uit te duwen, zal de gebruiker's positie reeds aan de achterkant van het object zijn uitgekomen en ontstaat een *doorduweffect* (zie Figuur 57). Gevolg is dat de pointer van de boortool zich wel aan de achterkant van het object bevindt, maar dat het boorgat het object niet volledig heeft doorboord. Dit is uiteraard niet de bedoeling.

Een oplossing hiervoor werd geïmplementeerd in de vorm van een special case, waarin de cilinder volledig door het object wordt gestoken wanneer dit doorduweffect plaatsvindt. Deze methode werkt, maar is niet realistisch: de gebruiker zal namelijk minder hard moeten duwen om het laatste stukje te overbruggen. Hierdoor wordt het dan weer moeilijk om gaten 'bijna' volledig door een CSG object heen te boren.

8.6.3 Dubbele muren, of objecten met een gat erin

Wanneer de gebruiker door een object heen boort, dient er geen kracht meer op de boor gezet te worden om deze naar voren te brengen. Toch is de booractie niet gedaan: de boor zelf steekt immers nog voor een stuk in het gat en kan dus niet méér naar opzij manoeuvreren dan wanneer de boor zich in het gat zelf bevond. Men moet de boor bovendien verder kunnen duwen tot aan een ernaast liggend object en ook daarin kunnen verder boren in exact dezelfde richting.



Figuur 58: Pointer heeft de vrijheid waardoor de boor zich vrij door het object uitbeweegt.

Een vergelijkbare situatie is een object met een gat erin: wanneer de boor het object doorboord heeft tot aan het gat, moet op exact dezelfde manier verder kunnen boren wanneer de boorpointer zich voorbij het gat heeft bewogen en in het 'tweede deel' van het object begint te boren.

In beide gevallen bevindt de boor zich op bepaalde momenten *buiten het object* en dient er toch verder geboord te worden. We zien hier duidelijk dat net zoals in de vorige paragraaf het boren nog niet beëindigd is wanneer de punt van de boor zich reeds buiten het object bevindt. Het boren is pas gedaan wanneer de boor wordt gedeselecteerd of volledig terug wordt getrokken uit het object waarin het boren begon.

8.7 Boren in een voorlopig haptisch object

De in de vorige paragraaf gestelde problemen maken duidelijk dat de methode waarin rechtstreeks in een CSG object geboord wordt in heel wat gevallen tekort schiet en dus praktisch gezien onbruikbaar is. Er is dus nood aan een andere methode waarmee men *in alle gevallen* op een correcte wijze haptisch in een CSG object kan boren. De oplossing bestaat erin het haptische gedeelte los te koppelen van het grafische, en gedeeltelijk ook van het haptische object.

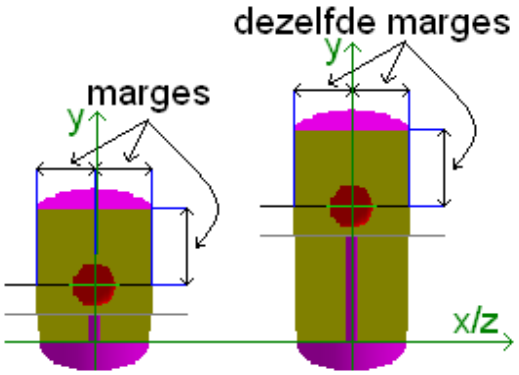
Waar het grafische gedeelte gebruik blijft maken van een visueel boorgat dat groter wordt naarmate de gebruiker verder boort, zal het haptische boorgedeelte volledig apart gebeuren in een speciaal ervoor aangemaakt *voorlopig haptisch CSG object*.

Dit voorlopig haptisch object voelt voor de gebruiker net zo aan als een echt haptisch gat in het CSG object, en in bepaalde gevallen zelfs realistischer. De gebruiker ziet en voelt dus dat hij of zij in het *'echte'* CSG object aan het boren is. We zullen zien dat boren in dit voorlopig haptisch object een goede oplossing biedt voor alle drie de in de vorige paragraaf vernoemde problemen.

8.7.1 Structuur

Het voorlopig haptisch object bestaat in feite uit een grote cilinder die altijd een stuk langer is dan het boorgat en breed genoeg opdat doorduweffecten onmogelijk worden (zie Figuur 59).
Hiervan wordt een uiterst smalle cilinder gesubtraheerd die net zoals bij de rechtstreekse methode het boorgat voorstelt en slechts een beperkte speling zoals in §8.5.4 toelaat. De cilinder bevindt zich in de richting van de Y-as en begint aan de oorsprong.

Naarmate het boorgat dieper wordt, wordt de voorlopige haptische cilinder langer om zo alle mogelijke doorduweffecten bij die bepaalde diepte te vermijden.



Figuur 59: Voorlopig haptisch object, aangepast aan de diepte van het voorlopig boorgat.

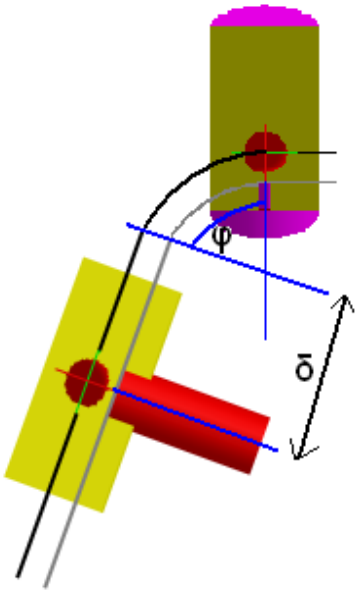
8.7.2 Verloop

De creatie van de boorobjecten gebeurt net zoals in de directe boormethode reeds bij het selecteren van de boor. Het uiteindelijk haptisch boorgat wordt hier echter nog niet aangemaakt, enkel het grafische boorgat en het haptisch CSG object.

In eerste instantie heeft het voorlopig haptisch object nog geen plaats of lengte. Bij het binnendringen van het CSG object wordt het voorlopig haptisch object geprojecteerd naar zijn juiste bestemming (zie Figuur 60) zodat

- de positie δ van de oorsprong overeenkomt met het penetratiepunt
- de richting φ van de opstaande Y-as overeenkomt met de boorrichting.

Telkens wanneer de gebruiker dus dieper in het CSG object boort, boort hij of zij eigenlijk in een geprojecteerd voorlopig haptisch object. De gesubtraheerde cilinder in dit object wordt net zoals bij de



Figuur 60: Het boorproces wordt geprojecteerd naar voorlopig haptisch object.

rechtstreekse methode langer (in de Y-richting) wanneer de gebruiker de boor dieper in het object duwt. Maar ook de errond liggende grote cilinder wordt telkens een beetje langer zodat de gebruiker in principe een oneindig lang CSG object mee kan doorboren.

Bij het beëindigen van de booractie, heeft het grafische boorgat reeds zijn definitieve afmetingen en kan de grafische CSG boom gesnoeid worden. Dan pas wordt het definitieve haptische boorgat aangemaakt aan de hand van:

- de lengte van het haptisch boorgat uit het voorlopig haptisch object,
- de geprojecteerde positie en richting
- de daadwerkelijke breedte van de haptische tool

Het voorlopige haptische object wordt tenslotte uit het geheugen verwijderd.

8.7.3 Performantie

Grafisch gezien is deze methode identiek en is er geen verschil in performantie. De haptische performantie is vergelijkbaar en zal voor grotere objecten zelfs voordeliger zijn:

selecteren van het de boortool: hier wordt het voorlopig haptisch object gecreëerd: dus de grote cilinder en de boorcilinder van het voorlopig haptisch object.

tijdens het boren: bij het overschrijden van de threshold worden nu de twee primitieven van het voorlopig haptisch object geherpositioneerd en (samen) vergroot. Daarenboven dienen de punten van het voorlopig haptisch object (het SCP en de pointerpositie) telkens omgezet te worden naar wereldcoordinaten en omgekeerd., wat extra matrixtransformaties met zich meebrengt. Daartegenover staat dat door het projecteren naar een object op de Y-as alle berekeningen binnen het voorlopig haptisch object in één dimensie kunnen gebeuren, wat afstandsberekeningen herleidt tot een '-' operatie tussen twee floating point getallen. Hier wordt dus tijd uitgespaard. Er moet nu ook een aparte inside-outside test uitgevoerd worden op het voorlopige haptische object. Dit object bevat echter slechts drie nodes, bovendien hoeft daardoor enkel het SCP van het voorlopig haptisch object berekend te worden. Zeker bij grote haptische objecten zorgt het gebruik van het voorlopig haptisch object dus voor minder berekeningen.

na het boren: na het boren dient het definitieve haptische object nog gecreëerd te worden, wat bij de rechtstreekse methode reeds gebeurde bij het selecteren van de boortool. Ook moet het voorlopig haptisch object worden afgebroken.

8.8 Voordelen van een voorlopig haptisch object

Door gebruik te maken van een voorlopig haptisch object, worden de problemen uit de rechtstreekse methode (zie §8.6) helemaal opgelost. Ze worden hier opnieuw overlopen.

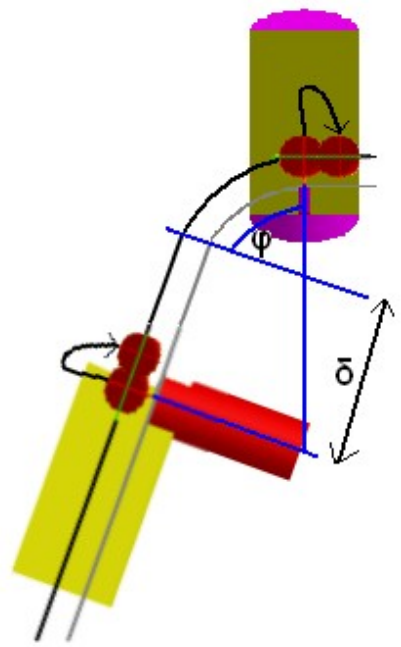
8.8.1 In de rand van een object boren

Het boren in de rand van een object verloopt nu even stabiel als het boren in het midden van een object. Dit komt omdat ongeacht de plaats waar aan het object ingeboord wordt, er steeds een even dik haptisch object klaar staat om onverwachte doorduweffecten naar opzij te vermijden.

Dit is bovendien ook veel realistischer: het is immers onmogelijk om tijdens het boren in een echt object per ongeluk de boor langs opzij door een object uit te duwen, zelfs als de zijkant van de boor zich buiten het object bevindt (zie Figuur 61, tenzij een object natuurlijk bestaat uit gelatine, breekt of barst.

De boor langs opzij door een object uitduwen kan natuurlijk wel indien het middelpunt van de boor zich nooit in het object heeft bevonden.

In dat geval zal de haptische boor echter niet boren aangezien het zich niet in het CSG object bevindt. Dit heeft meer weg van frezen en is een heel andere tool, die ook interessant kan zijn om CSG objecten te bewerken.

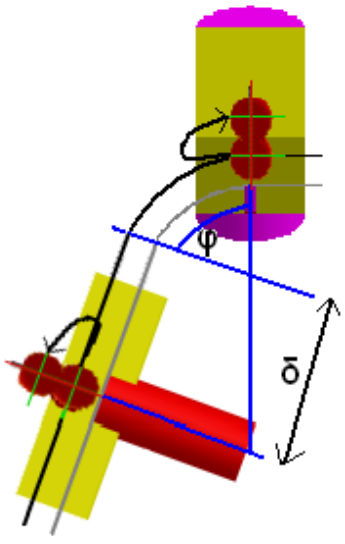


Figuur 61: Bij boren in de rand blijft de pointer binnen het voorlopig haptisch object.

8.8.2 Door een object uitboren

Ook door een object uitboren wordt mogelijk door gebruik te maken van het voorlopig haptisch object. Om te kijken of de boor zich binnen dan wel buiten het object bevindt, moet dus telkens nog een inside-outside test gebeuren.

Deze test gebeurt echter meestal op het punt waar de pointer van de gebruiker zich bevindt, maar niet altijd. Op het moment dat de gebruiker namelijk aan het duwen is om het boorgat dieper te maken en de pointer zich daardoor binnen in het voorlopig haptisch object bevindt, wordt op het 'echte' haptische object de inside-outside test uitgevoerd op de rand van het voorlopig boorgat (zie Figuur 62). Bevindt zich dit punt nog binnen het 'echte' haptische object, dan is er nog niet door het 'echte' haptische object uitgeboord. In het andere geval natuurlijk wel en boren we niet meer doorheen het CSG object.



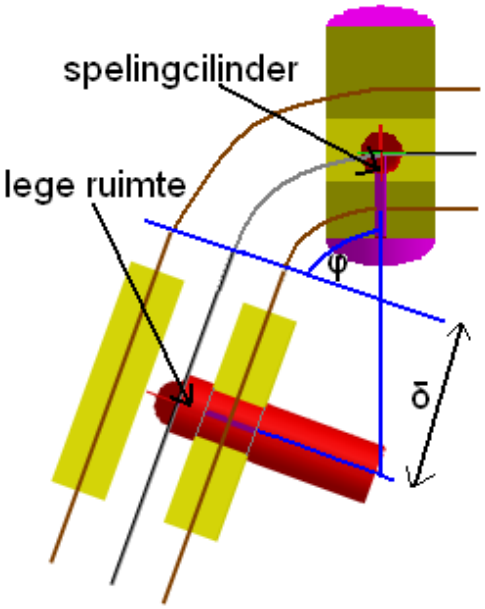
Figuur 62: Door het object uitboren wordt mogelijk want pointer blijft binnen het voorlopig haptisch object.

De gebruiker zal hier namelijk geen doorduweffect ondervinden, tenzij dit de bedoeling is en hij of zij daadwerkelijk doorheen het object geboord heeft.

8.8.3 Dubbele muren, of objecten met een gat erin

Ook boren doorheen objecten met een gat erin of doorheen dubbele muren wordt mogelijk dankzij gebruik van het voorlopig haptisch object.

Nadat de boor door het object heengedrongen is, wordt het boorgat nog steeds groter naarmate de gebruiker de boor dieper duwt. De haptische boor beweegt dan zonder terugkoppeling verder naar voren wanneer er door een object uitgeboord is en wanneer de pointer van de haptische boor zich buiten het 'echte' CSG object bevindt. De beweging naar opzij blijft echter beperkt tot een lichte speling, zoals men kan zien in Figuur 63.



Figuur 63: Na door een object uit te boren heeft de boor nog steeds een beperkte speling want pointer blijft binnen het voorlopig haptisch object.

Wanneer de gebruiker echter de boor verder blijft doorduwen en een ander object (of een ander deel van hetzelfde object) op zijn weg tegenkomt, zal er weer een terugkoppelingskracht gegenereerd worden. Net zoals bij het inboren in het eerste object (of deel van hetzelfde object), kan er nu opnieuw *'echt'* geboord kan worden in de richting waarin door het eerste object geboord werd.

8.9 Conclusie

Dit hoofdstuk bestudeerde een haptische boor als voorbeeld van een meervoudige of continue haptische CSG interactie. De keuze viel op de boor omdat dit een veelgebruikte representatieve bewerking is en realiseerbaar door middel van slechts één puntkracht en subtractie met een primitief object.

Deze boor mag dan een realistisch gevoel geven, het blijft nog steeds een abstractie van de werkelijkheid. Het was dan ook niet de intentie om een tool te maken waarmee zo realistische mogelijk geboord kon worden, maar om een werkbare haptische boor te maken die in een willekeurig CSG object kan boren.

Stapsgewijs namen we door wat er komt kijken bij zo'n booroperatie: een

algemeen overzicht (§8.2), de nodige CSG operaties, het creëren van een boorgat (§8.3), en boren in ideale omstandigheden (§8.4). Hierbij werd elke stap en de meeste alternatieven concreet geïmplementeerd om hun deugdelijkheid zowel qua interactie als qua performantie te kunnen toetsen, dus ook het rechtstreeks boren in CSG objecten, inclusief hun problemen (§8.5 en §8.6) en lapmiddeltjes. Dit maakte duidelijk dat rechtstreeks boren niet voldeed, maar een goed gericht voorlopig haptisch object met voldoende afmetingen deed dat wel (§8.7 en §8.8).

De uiteindelijk verwezenlijkte en hier beschreven tool kan door meerdere objecten achter mekaar boren vanuit elke mogelijke richting, en leverde screenshots voor dit hoofdstuk.

We kunnen in verband met continue meervoudige haptische interacties, dan ook concluderen dat alvast de interactieve haptische boor op een performante manier geïmplementeerd kan worden in een haptische modeller. Andere tools die ook een subtractie met een cilinder creëren, zoals een frees, die niet zo heel veel verschillen hebben met de boor, kunnen ook voorzien worden. Wat deze tools betreft is een interactieve haptische modeller voor CSG trees alvast realiteit.

Hoofdstuk 9 Kerven met een beitel in een CSG object.

9.1 Inleiding

Naast tools die continue of meervoudige operaties in CSG objecten mogelijk maken (zie Hoofdstuk 8), bestaan er tools die een discrete of enkelvoudige actie uitvoeren. Dat wil zeggen dat de bewerking aan het bestaande object na slechts één actie definitief is. Er is dus geen sprake van een bewerking die nadien door verdere haptische modelleeracties gespecifieerd en aangepast wordt.

Net zoals bij de meervoudige, worden ook enkelvoudige acties bestudeerd aan de hand van een tool, in casu de beitel, die bovendien net zoals de boor concreet geïmplementeerd werd om zijn deugdelijkheid aan te tonen. Er zijn natuurlijk ook andere enkelvoudige tools te bedenken zoals het vanop afstand schieten van een primitieve in een CSG object.

Dit hoofdstuk beschrijft de haptische beitelactie in het algemeen en adresseert de nodige CSG operaties en enkele issues (§9.2). Er wordt hierbij voordurend gewezen op verschillen en gelijkenissen met de meervoudige boor. Het is de lezer dan ook aangeraden om zeker eerst het vorige hoofdstuk gelezen te hebben.

Ook hier stellen we al snel vast dat haptisch boren in een voorlopig haptisch object een betere oplossing is dan rechtstreeks haptisch boren (§9.3). Ook de performantie wordt onder de loep genomen en vergeleken met de haptische boor.

9.2 Algemeen

Met een beitel kerven in een object is een interactie tussen een tool, de beitel, en een bestaand CSG object. Deze actie wordt discreet of enkelvoudig genoemd omdat er hier slechts één actie is, namelijk het duwen van de beitel in het object. Hierbij hangen

- de breedte en de vorm van het uitgekerfde gat af van de breedte en de vorm van de beitel, en hangt
- de diepte van het uitgekerfde gat af van de kracht waarmee in het object gebeiteld wordt.

Abstractie: In de realiteit klopt de beitel niet zichzelf in een voorwerp, maar gebeurt dit in combinatie met een hamer, wat eigenlijk een combinatie van twee tools is. Hier gaan we er echter vanuit dat de gebruiker met de beitel zelf in het object kerft.

Een typische actie met een beitel verloopt bijgevolg ongeveer zo :

- De gebruiker bepaalt plaats en richting waarin hij of zij moet kerven, en bepaalt het type beitel en zijn grootte.
- Daarna wordt de beitel vastgenomen of geselecteerd
- Bij de inslag van de beitel in het object, ervaart de gebruiker weerstand. Hoe meer kracht er op de beitel wordt gezet, hoe dieper de holte zal zijn die uit het object gekerfd wordt.
- Wanneer de beitel terug uit het object gehaald wordt, wordt de uitgekerfde ruimte meteen zichtbaar. En wordt de beitel terug weggeplaatst of gedeselecteerd.

Net zoals bij kerven in de realiteit, bevat het CSG object een holte, dat net zoals het boorgat een subtractie is met een primitieve. Naargelang het gewenste effect en de drie beschikbare soorten primitieven, kunnen er verschillende soorten beitels gebruikt worden.

Figuur 64: Drie soorten beitels

Figuur 64 geeft alvast drie voorbeelden: de rechte beitel (boven), die er meer uitziet als een korte boor, de gepunte beitel (midden), die eigenlijk bestaat uit een geroteerde kubus en een gekromde beitel (onder), met een geroteerde cilinder als beitelkop. Aangezien we enkel subtraheren met de primitieve die vanvoor op de boor gemonteerd is, kan bij een te diepe inslag een onrealistische holte verkregen worden. De grijze lijn in Figuur 64 toont aan tot welke diepte er ingekerfd kan worden om dit niet aan de hand te krijgen. Bij de bovenste beitel, die eigenlijk dezelfde continue vorm heeft als een boorn

De haptische beitel die hier besproken wordt, gebruikt enkel primitieven als beitel. Toch kunnen deze uitgebreid worden naar meer complexe beitels die op zichzelf echte CSG objecten zijn. Dit zou ons echter te ver leiden, maar is een interessant future work (zie Hoofdstuk 11).

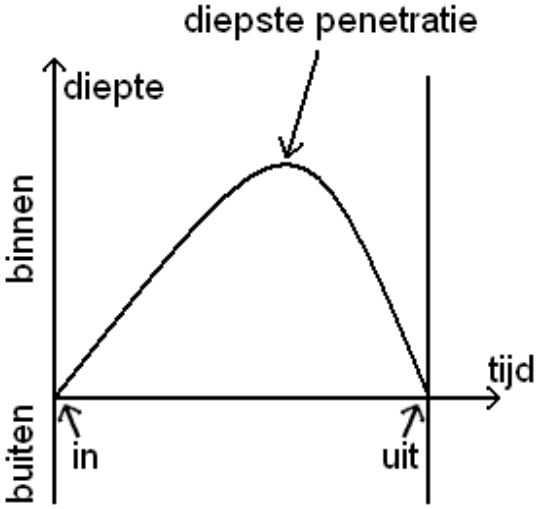
De in deze thesis voorgestelde beitel gebruikt haptische interactie. Ze maakt gebruik van zes actieve vrijheidsgraden: drie voor beweging en drie voor het bepalen van de richting waarin de beitel vastgehouden wordt. De boor heeft slechts vijf vrijheidsgraden nodig omdat deze een rond gat boort, waarbij de draaiing dus geen belang heeft. Dit is bij de beitel echter wel van belang.

9.2.1 verloop

Het verloop van een beitelactie wordt geschetst in Figuur 65. Bij de inslag in het CSG object wordt het SCP berekend en ontstaat een terugkoppelings-kracht die evenredig is aan de diepte waarmee de pointer in het object doordringt in de beitelrichting.

Aangezien de pointer en het SCP met elkaar verbonden zijn door middel van een sterke virtuele veer, zal de terugkoppelingskracht verhogen naarmate de gebruiker meer kracht zet op de beitel.

Wanneer de terugkoppelings-kracht overeenkomt met de kracht die de gebruiker op het haptisch toestel uitoefent, bereikt de pointer zijn diepste penetratie in het object. Tot deze diepte wordt in het CSG object gekerfd. Wanneer de gebruiker de uitgeoefende kracht verlaagt, zal de pointer het CSG object terug verlaten en de beitelactie is voorbij. Op dit moment creëert de modeller een grafisch en een haptisch kerfgat van dezelfde vorm als de tip van de beitel op de plaats waar de tool zijn maximale indringdiepte in het object bereikte.



Figuur 65: Verloop van een beitelactie in een CSG object. De uitkerving moet zo diep zijn als het diepste penetratiepunt.

9.2.2 CSG Operaties

Aangezien deze actie enkelvoudig is, moet er *tijdens* het beitelen niets aan het CSG object veranderd worden. Er is dus geen nood aan een *voorlopig zichtbare kerfholte*. Er zijn ook geen speciale haptische voorwaarden tijdens het beitelproces, tenzij de terugkoppelingskracht die tijdens het kerven zowiezo reeds ondervonden wordt. Deze kracht is dezelfde die een gewone pointer ondervindt en moet niet continu geupdated worden zoals bij de boor.

Pas nadat de pointer het (oorspronkelijke) object terug verlaten heeft, worden de CSG objecten aangemaakt en gepositioneerd. Net zoals bij de boor moet hier na het toevoegen van de subtractie de grafische CSG boorstelling gesnoeid worden, en de haptische niet.

9.2.3 Schuin beitelen

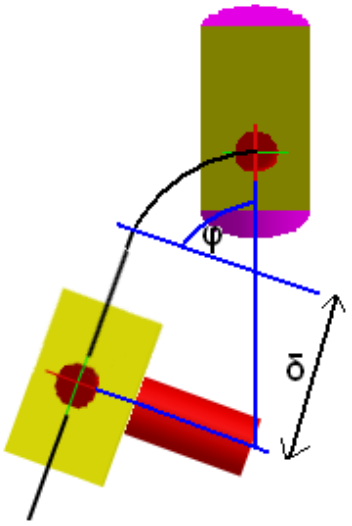
Waar schuin inboren bij de creatie van een boorgat een probleem vormde, stelt dit zich hier niet omdat we er vanuit gaan dat de beitelvorm 'groot genoeg' is. Het uitkerven met de beitel zal dus geen materie *missen*. Indien we echter werken met kleine beitels, dan acteren beitels in principe hetzelfde als een boorgat en stelt dit probleem zich wel. In dat geval is het bovendien aangewezen om als beitel zeker een rechte cilinder te gebruiken, net zoals bij een boorgat. Anders zouden er niet één object, maar een hele reeks objecten gesubtraheerd moeten worden, namelijk voor elke diepte ééntje.

Dit zou al snel leiden tot excessief grote CSG bomen, waardoor interactief modelleren onmogelijk wordt. Om deze reden wordt er ook enkel geboord met een cilinder aangezien deze slechts één (constant gewijzigd) object blijft tijdens het boorproces. Een oplossing hiervoor ligt weer in het gebruik van andere primitieven of in tools die zelf CSG bomen zijn. Deze mogelijkheden werden echter gecatalogeerd onder future work (zie Hoofdstuk 11).

9.3 Beitelen in een voorlopig haptisch object

Net zoals bij het boren, kunnen er doorduweffecten ontstaan bij te beitelen in de rand van een CSG object. Hierdoor is het niet mogelijk om dicht bij de rand van een CSG object een inkerving te maken. Ook het kerven in vrij dunne objecten genereert ongewilde doorduweffecten. De oplossing hiervoor is net dezelfde als bij de boorapplicatie, namelijk het aanmaken van een voorlopig haptisch object, waar in gebeiteld wordt (zie Figuur 66).

Toch is er een situatie waarin beitelen in een haptisch object te kort schiet. Indien er hard in een dun object gebeiteld wordt, bestaat in realiteit de kans dat de beitel vanaf een bepaald moment volledig door het object uitzit. Aangezien wij hier gebruik maken van een voldoende grote beitel, zal dit niet voorkomen omdat de beitel toch vast komt te zitten, maar bij kleine beitels vormt dit een probleem omdat er terugkoppeling gegenereerd wordt waar dit eigenlijk niet nodig is.



Figuur 66: Algemene beitelactie in een voorlopig haptisch object verloopt analoog aan een booractie.

9.3.1 Structuur

Het voorlopig haptisch object voor een beitelactie bestaat net zoals dat voor een boortool uit een grote cilinder die lang en breed genoeg is om alle mogelijke doorduweffecten op te vangen. Deze cilinder is steeds statisch door de enkelvoudigheid van de beitelactie. Het object is nodig zolang de beitel in het object kerft. Bevindt zich ook in de richting van de Y-as, beginnend aan de oorsprong en dan getransformeerd naar het penetratiepunt. Het object kan als alternatief even goed meteen ter plaatse gepositioneerd worden aangezien er buiten één afstandsberekening geen operaties uitgevoerd moeten worden en er dus niet echt een performantievoordeel is zoals bij de boor.

9.3.2 Verloop

De creatie van het voorlopig haptisch object gebeurt net zoals bij de boor best wanneer de beitel tool geselecteerd wordt, zodat het van de eigenlijke haptische interactie gescheiden blijft en dus niet interfereert wanneer de haptische balast groter wordt door het uitrekenen van het SCP. Bij het binnendringen van het object wordt het voorlopig haptisch object dan nog enkel gepositioneerd of geprojecteerd (naargelang er al dan niet getransformeerd wordt).

Op het moment dat de beitelactie voorbij is, worden dan net zoals bij een directe interactie de grafische en haptische objecten aangemaakt in de juiste positie en richting, en wordt het voorlopig haptisch object tenslotte uit het geheugen verwijderd.

9.3.3 Performantie

Zowel grafisch als haptisch gezien kost deze enkelvoudige haptische interactie minder verwerkingstijd dan een meervoudige. Ook de grafische update, met een herberekening van het CSG object als gevolg, is pas nodig nadat het interactief modelleren beëindigd is. Omdat het voorlopig haptisch object bij de beitelactie slechts uit één primitieve bestaat, is er ook tijdens een beitelactie minder te berekenen als tijdens het boren. Verder moeten tijdens het de beitelactie geen haptische objecten worden verschoven. Een overzicht:

selecteren van de beitel: hier wordt het voorlopig haptisch object gecreëerd. Dit object bestaat nu slechts uit één primitieve in plaats van twee primitieven en een subtactie node.

tijdens de beitelactie: Hier dienen de nodige parameters van het voorlopig haptisch object (het SCP, en de huidige pointerpositie) nog steeds omgezet naar wereldcoördinaten en omgekeerd, wat extra matrixtransformaties met zich meebrengt. Aangezien het voorlopig haptisch object slechts bestaat uit één primitieve, zal het berekenen van het SCP nu nog eenvoudiger worden dan tijdens het boren. Beitelen in een voorlopig haptisch object heeft ook een voordelen ten opzichte van beitelen in het bestaande CSG object; niet alleen voor de randgevallen, maar ook omdat slechts het SCP van één primitieve berekend moet worden.

na de beitelactie: na de beitelactie dient zowel het definitieve grafisch als haptisch object nog gecreëerd te worden. Positie en orientatie van beide is dezelfde waardoor deze berekening slechts één keer moet gebeuren. Tenslotte wordt het voorlopig haptisch object afgebroken.

9.4 Conclusie

Dit hoofdstuk behandelde tools die een discrete of enkelvoudige actie uitvoeren aan de hand van een concrete case study over de haptische beitel. Deze werd voortdurend vergeleken met de haptische boor uit hoofdstuk 8. Na een algemeen overzicht, waarin de het verloop en de CSG operaties van een beitelactie werden beschreven (§9.2), hebben we aandacht besteed aan onrechtstreeks haptisch beitelen (§9.3). Ook hier lijkt het gebruik van een voorlopig haptisch object een noodzaak. We eindigden §9.3 met een overzicht van de performantie van het haptisch beitelen.

Het spreekt voor zich dat zulke enkelvoudige haptische interacties eenvoudiger zijn en dus ook minder verwerkingstijd vragen, al zijn de verschillen al bij al klein. Deze enkelvoudige haptische interacties kunnen dan ook zonder problemen deel uitmaken van een interactieve haptische CSG modeller.

Hoofdstuk 10 Algemene conclusie

Tegenwoordig zijn er tal van applicaties op de markt waarmee objecten interactief gemanipuleerd kunnen worden. Deze maken hiervoor haast allemaal gebruik van een combinatie muis-toetsenbord als input en beperken de feedback tot het visueel tonen van de modelleeractie. Om het gevoel van immersie in zulke modelleeromgevingen te vergroten, kan er een haptische component toegevoegd worden. Gebruik makend van een toestel voor haptische interactie zoals de PHANToM, wordt het mogelijk om objecten in de scène niet alleen te voelen, maar ook interactief te kunnen manipuleren.

Het doel van deze thesis was te onderzoeken hoe zo'n haptische modelleeromgeving gerealiseerd kon worden wanneer de objecten de scène voorgesteld werd met behulp van Constructieve Sollide Geometrie, of kortweg CSG. Het haptisch renderen van CSG objecten, de onontbeerlijke component waarzonder een haptische CSG modeller onmogelijk kan bestaan, was immers mogelijk geworden nadat algoritmes hiervoor beschreven werden in een paper die gemaakt werd door Prof. Frank Van Reeth en mijn co-promotor Prof. Chris Raymaekers [14].

In mijn literatuurstudie ben ik me eerst gaan verdiepen in CSG, een vorm van solid modelling die 3D objecten op een vooral heel intuïtieve manier voorstelt en een mooi alternatief biedt voor conventionele B-reps, de polygonenmeshes die in veruit de meeste 3D omgevingen worden gebruikt. Bovendien heeft CSG heel wat eigenschappen die handig gebruikt kunnen worden bij het interactief modelleren, voornamelijk door zijn structuur die gebruik maakt van Booleaanse operatoren. Het visueel renderen van CSG objecten tegen interactieve snelheden was echter niet voor de hand liggend, vooral omdat het moeilijk samenwerkt met grafische hardware, die enkel op renderen van B-reps is gericht. Visuele CSG modellers komen vrij veel voor in CAD applicaties, maar maken meestal geen gebruik van interactieve tools. Ze werken rechtstreeks met de CSG structuur.

De tweede pijler van mijn literatuurstudie behandelde de wereld van haptics, haptische interfaces en interactie. Op dit gebied is de laatste jaren heel wat research verricht. Wat meteen opviel was de snelheid van de interactiecyclus van minstens 1Khz. Dit is aanzienlijk meer dan de visuele interactiecyclus die met 30hz reeds tevreden is. Haptische rendermethoden moeten bijgevolg zeer performant zijn, wat de momenteel het vaakst gebruikte constraint based methode, dan ook is.

De literatuurstudie werd afgerond met een zeer grondige analyze van de paper over haptisch CSG renderen, die de vorige twee pijlers combineert en de haptische modeller mogelijk maakt.

Aan de hand van deze kennis heb ik in het algemeen een CSG modeller beschreven en geïmplementeerd. Net zoals andere modellers maakt ook deze gebruik van specifieke tools, die abstracties zijn van tools uit de realiteit en waarmee bepaalde haptische interacties kunnen uitgevoerd worden. Ik onderscheidde twee soorten tools die een CSG object kunnen wijzigen: enkelvoudige tools die na slechts één interactie een bewerking doorgevoerd hebben en meervoudige tools die meerdere interacties nodig hadden.

Beide soorten tools werden onder de loupe genomen aan de hand van een onderzoek met implementatie naar een concrete tool uit elke categorie. Ik koos voor een enkelvoudige haptische beitel en een meervoudige haptische boor. Voor beide tools bleek het noodzakelijk om qua haptische interactie niet rechtstreeks met het CSG object te werken, maar tijdens de modelleeractie(s) gebruik te maken van een voorlopig haptisch object. Dit gebruik lijkt dan ook nodig voor de meeste interactieve tools die in beide categorieën te bedenken zijn.

Uit het bestuderen van de twee haptische CSG tools blijkt dat zowel de haptische beitel als de boor op een performante manier gerealiseerd kunnen worden op standaard hardware. Daarbij verzwaart de haptic load slechts in beperkte mate in vergelijking met gewone passieve haptische interactie.

Ik mag besluiten dat zowel enkelvoudige als meervoudige interacties, waarbij op een intuïtieve manier haptisch gemodelleerd wordt in CSG objecten, mogelijk zijn tegen interactieve snelheden.

Hoofdstuk 11 Future Work

Het is een boutade dat een thesis nooit helemaal klaar is. Er is inderdaad steeds toekomstig werk te bedenken, zoals uitbreidingen, alternatieven,... Dit hoofdstuk behandelt enkele ideeën die de haptische CSG modeller kunnen uitbreiden of verfraaien en buiten het bestek van deze thesis gevallen zijn.

Scaling van CSG primitieven: Door CSG primitieven de toe te laten gescaled te worden, breiden de mogelijheden van de haptische CSG modeller uit. Zo maakt scaling het mogelijk om ook primitiven te beschouwen in de vorm van balken en eivormige objecten. Vooral balken kunnen handig zijn, bijvoorbeeld om een continue vierkante beitel te realiseren. Hierdoor vallen de problemen bij het diep inkerven van een boorkop die bestaat uit een geroteerde cilinder of kubus weg (zie Figuur 61, midden en onder). Zelfs een vierkante boor is mogelijk. Dit is niet erg realistisch, maar kan wel eens handig zijn. Hiervoor zal het boren wel in 6DOF gebeuren aangezien de boorkop nu ook gedraaid kan worden.

Tools die zelf CSG objecten zijn: Door bij modelleeroperaties gebruik te maken van tools die zelf CSG objecten zijn, kunnen meer complexe tools gerealiseerd worden. Vooral bij het beitelen kan dit een groot voordeel zijn aangezien de beitelkoppen dan *'realistischer'* gemaakt kunnen worden. Naargelang de vorm en de complexiteit van het CSG object dat de tool voorstelt, zal interactief modelleren zwaarder worden. Onderandere grafisch normaliseren zal minder geoptimaliseerd kunnen worden en in bepaalde gevallen zal ook haptische normalisatie nodig zijn. Bovendien zullen het aantal nodes in de boom na elke operatie aanzienlijk groter zijn dan bij tools die slechts uit een enkele primitieve bestaan .

Geoptimaliseerde grafische CSG renderalgoritmes: Wanneer aan een grafisch CSG object enkel de positie of grootte van één primitief object aangepast moet worden, zou het kunnen dat de CSG boom slechts gedeeltelijk opnieuw uitgerekend zou moeten worden. Indien dit mogelijk is, zou dit alvast een snelheidswinst opleveren.

Andere primitieven: Tot op heden wordt er gebruik gemaakt van slechts drie CSG primitieven. Dit komt omdat haptische renderalgoritmes slechts voor deze drie primitieven zijn bepaald. Er zouden dus haptische renderalgoritmes gevonden kunnen worden voor andere primitieve objecten zodat zij ook in de modeller gebruikt kunnen worden. Een handige primitive is bijvoorbeeld een kegel: deze kan je op de kop van de boor plaatsen om een meer realistisch boorgat te bekomen. Een kegel kan bovendien dienen als beitelkop.

Andere tools: Het is zeker een goed idee om buiten de beitel en de boor ook andere tools te gaan implementeren. Sommige tools, zoals een pistool met kogels, verschillen niet zoveel van in casu de beitel, andere zoals een graver of een tool om objecten in elkaar te duwen, zijn pas te realiseren als minstens enkele van de hier besproken ideeën reeds verwezelijkt zijn.

Modelleren met twee haptische toestellen: Hierdoor kunnen ook tools gemaakt worden die bestaande uit twee objecten die met mekaar gecombineerd kunnen worden, zoals een beitel met een hamer. Beitelen wordt dan wel een continue operatie aangezien de gebruiker meermaals met de hamer op de beitel kan kloppen om deze dieper in het CSG object te duwen.

Bewerkingen met heterogene objecten: Heterogene objecten kunnen bijvoorbeeld gerealiseerd worden door elke primitieve zijn eigen hardheid te geven. Tijdens het boren zal er dan harder geduwd moeten worden in bepaalde delen dan in andere.

Bibliografie

[1]
F. Rottensteiner, Semi-automatic extraction of buildings based on hybrid adjustment using 3D surface models and management of building data in a TIS, PHD Thesis Vienna University of Technology

[2]
R.S. Avila, an introduction to Haptics, Course 38, 26th International Conference on Computer Graphics and Interactive Techniques "Siggraph 99", Los Angeles, US, 8-13 Augustus 1999, Course Notes.

[3]
A. A. G. Requicha, Representations for rigid solids: Theory, methods and systems.
ACM Computing Surveys 12(4), p,437-464, Dec. 1980.

[4]
N. Stewart, G Leach, en S. John van de RMIT University, Melbourne, Australia. An Improved Z-Buffer CSG Rendering Algorithm. In 1998 Eurographics/SIGGRAPH Workshop on Graphics Hardware, p. 25-30, Lissabon, PT, 31 augustus – 1 september 1998.

[5]
S.D. Roth, Ray casting for modelling solids, Computer Graphics and Image Processing, **18**(2), pp. 109-144 (1982)

[6]
A. Requicha, H Völcker, Boolean Operations in Solid Modelling: Boundary Evaluation and Merging Algorithms, Proc. Of the IEEE, Vol. 73, No. 1, Januari 1985, p. 30-44

[7]
J. Goldfeather, S. Molnar, G. Turk, H. Fuchs, Near-realtime CSG rendering using tree normalisation and geometric pruning, IEEE Computer Graphics and Applicationsn 9(3), Mei 1989, p. 20-27

[8]
T. Wiegand, Interactive rendering of CSG models, Computer Graphics Forum, 15(4), oktober 1996, p. 249-261

[9] Diego Ruspini, Haptic Rendering of Graphical Models, Course 38, 26th International Conference on Computer Graphics and Interactive Techniques "Siggraph 99", Los Angeles, US, 8-13 Augustus 1999, Course Notes.

[10]
M. Smolens, Haptic Rendering Course Notes, University of North Carolina

[11]
C.B. Zilles, J.K. Salisbury, "A constraint-based god-object method for haptic display." Proc. IEE/RSJ International Conference on IntelligentRobots and Systems, Human Robot Interaction, and Cooperative Robots, Vol. 3, p. 146-151, 1995.

[12]
D. Staples, Haptic Application Issues, Course 38, 26th International Conference on Computer Graphics and Interactive Techniques "Siggraph 99", Los Angeles, US, 8-13 Augustus 1999, Siggraph 99 Course Notes.

[13]
Mandayam A Srinivasan, Laboratory for Human and Machine Haptics: The Touch Lab, Massachisetts Institute of Technology; "What is Haptics?", Course Notes.

[14]
C. Raymaekers, F. Van Reeth, Algorithms for Haptic Rendering of CSG Trees, Proceedings of Eurohaptics 2002, Edinburgh, GB, 8-10 July 2002 / Wall S. [edit.], e.a., s.l., , 2002, p. 86-91

[15]
Wikipedia (www.wikipedia.org).

[16]
STEP formaat (ISO[17] 10303) (http://wheger.tripod.com/step)

[17]
ISO (Standard for the Exchange of Product model data) (http://www.iso.org/)

[18] Voxel 3D : Eenvoudig 3D Modelleer Softwarepakket dat gebruikt maakt van Voxels (http://www.everygraph.com/voxel3d/)

[19] STEWART, N., LEACH, G., AND JOHN, S. Linear-time CSG rendering of intersected convex objects. In 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG 2002 (2002), vol. II, pp. 437--444.

[20] W.C. Thibault, B.F. Naylor, Set operations on polyhedra using binary space partitioning trees, SIGGRAPH '87, pp. 153-162, 1987.

[21] K. Salisbury, F. Conti, F. Barbagli, Haptic Rendering: Introductory concepts, IEEE Computer Graphics and Applications, Vol. 24, Nr. 2, p. 56-64, 2004.

[22] C. Basdogan, S. De, J. Kim, M. Muniyandi, H. Kim, M. A. Srinivasan, Haptics in Minimally Invasive Surgical Simulation and Training, IEEE Computer Graphics and Applications, Vol. 24, Nr. 2, p. 24-32, 2004

[23] J. Kim, H. Kim, B. K. Tay, M. Muniyandi, J. Jordan, J. Mortensen, M. Oliveira, M. Slater, Transatlantic Touch: A Study of Haptic Collaboration over Long Distance, Presence: Teleoperators & Virtual Environments, Vol. 13, Nr. 3, p. 328-337, 2004

[24] Het Immersion haptic workstation.
(http://www.immersion.com/3d/docs/HWS2_datasheet.pdf)

[25] Sensable Technologies, makers van de PHANToM
(http://www.sensable.com/)

[26] Force Dimension, makers van de Omega en Delta haptische toestellen
(http://www.forcedimension.com/)

[27] Chai 3D, een open-source haptisch framewerk dat werkt met meerder types van haptische toestellen.
(http://www.chai3d.org/)

[28] MPB, makers van onderandere de MPB Freedom en de MPB API.
(http://www.mpb-technologies.ca/)

[29] BRLCAD, een open-source CAD programma dat gebruik maakt van CSG
(http://sourceforge.net/projects/brlcad)

[30] J. Rossignac, A. Megahed, B.-O. Schneider, Interactive inspection of solids: Cross-sections and interferences, SIGGRAPH '92, vol. 26, p. 353-360, 1992.

[31] N. Stewart, G. Leach, S. John, A CSG Rendering Algorithm for Convex Objects, The 8-th International Conference in Central Europe on Computer Graphics, Visualisation and Interactive Digital Media '2000 – WSCG 2000, Volume II, pp. 369-372

[32] N. Stewart, G. Leach, S. John, Linear-time CSG Rendering of Intersected Convex Objects, The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision '2002 – WSCG 2002, Volume II, pp. 437-444

[33] Improved CSG Rendering using Overlap Graph Subtraction Sequences, International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE 2003, pp. 47-53

[34] C. Brenner, Modelling 3D Objects Using Weak CSG Primitives, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 35, ISPRS, Istanbul, 2004.

[35] Imageware ModelMagic3D, een 3D modelling applicatie die gebruik maakt can CSG bomen. Gratis voor persoonlijk en educatief gebruik.
(http://www.imagewaredev.com/)

[36] Valve's Hammer Editor, een freeware editor voor het bouwen van mappen voor de Source Engine die gebruik maakt van CSG.
(http://collective.valve-erc.com/)

[37] G. Chappell, Implementing Scene Graphs, CSG Trees, Course Notes, University of Alaska, Fairbanks, Slide 11
(http://www.cs.uaf.edi/~cs481/slides/200401246scene-impl.ppt)

[38] R. Stone, Haptic feedback: A potted history, from telepresence to virtual reality. In the first International Worksh op on Haptic Human-Computer Interaction, Glasgow, UK. Springer-Verlag Lecture Notes in Computer Science 2000, pp. 1-7

[39] Ricardo S. Avila, Lisa M. Sobierajski, "A Haptic Interaction Method for Volume Visualization," *vis*, p. 197, Seventh IEEE Visualization 1996 (VIS'96), 1996.

[40] HAL, HAptic Library, het haptisch framewerk van Chris Raymaekers en Lode Vanacken waar mijn implementatie gebruik van maakt bij het aansturen van het haptisch toestel.
(http://www.edm.luc.ac.be/software/hal/)

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen en uw akkoord te verlenen.

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Haptisch renderen van CSG bomen
Richting: **Licentiaat in de informatica** Jaar: **2006**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Deze toekenning van het auteursrecht aan de Universiteit Hasselt houdt in dat ik/wij als auteur de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij kan reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

U bevestigt dat de eindverhandeling uw origineel werk is, en dat u het recht heeft om de rechten te verlenen die in deze overeenkomst worden beschreven. U verklaart tevens dat de eindverhandeling, naar uw weten, het auteursrecht van anderen niet overtreedt.

U verklaart tevens dat u voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen hebt verkregen zodat u deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal u als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze licentie

Ik ga akkoord,

Jeroen CORNELISSEN

Datum: