Törmäysten havaitseminen tietokoneanimaatiossa

Jyrki Rasku

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi /
Vuorovaikutteinen teknologia
Pro gradu -tutkielma
Elokuu 2003

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyopin laitos / Vuorovaikutteinen teknologia
Tekijän Nimi: Jyrki Rasku
Pro gradu -tutkielma, 65 sivua, 2 liitesivua
Elokuu 2003

Tässä tutkielmassa tarkastellaan aluksi törmäysten havaitsemiseen liittyvien ongelmien luonnetta, jonka jälkeen luodaan lyhyt katsaus näiden ongelmien yleisimpiin ratkaisumenetelmiin. Seuraavaksi tutkitaan useasta kohteesta muodostuvan animaatiosysteemin törmäysten havaitsemisen periaatteita. Luku 6 käsittelee Stefan Gottschalkin [12] väitöskirjaa mukaillen kohteen mukaan suunnatun rajoitustilavuushierarkian käyttöä törmäyskyselyissä. Luvussa 7 esitellään yleisimpiä geometristen primitiivien välisiä leikkauksia, joiden määrittämiseen törmäyksien havaitseminen lopulta johtaa. Luku 8 perustuu myös Gottschalkin väitöstutkimukseen, ja se käsittelee rajoitustilavuushierarkiassa suoritettavien törmäyskyselyiden oikeellisuutta sekä niiden lukumäärien ala- ja ylärajoja. Lopuksi tarkastellaan törmäyskyselyiden suorittamisen vaikutusta tietokoneanimaation ruudunpäivitysnopeuteen pienen testiohjelman avulla.

Avainsanat ja -sanonnat: Hierarkkiset tietorakenteet, törmäysten havaitseminen, geometrian approksimointi, fysikaalinen mallinnus.

Sisällys

1. Johdanto

Nykyaikaisilla tietokoneilla pystytään tuottamaan erittäin näyttävää ja nopeaa animaatiota. Erilaiset kulmapiste- ja pikselivarjostustekniikat ovat mahdollistaneet luonnolliselta näyttävien animaatioiden toteuttamisen. Animaatioita käytetään paljon eri tarkoituksissa. Esimerkiksi CAD/CAM ympäristöissä erilaisten osakokonaisuuksien asennusten simuloinneissa, tietokonepeleissä ja virtuaalitodellisuussovelluksissa. Kaikissa sovelluksissa on tärkeää, että animaatiossa mukana olevat kohteet vaikuttavat toisiinsa luonnonlakien mukaisesti. Ei ole suotavaa, että kohteet kulkevat toistensa läpi vaikuttamatta toisiinsa millään tavalla.

Kohteiden välinen vuorovaikutus tietokoneanimaatiossa voidaan jakaa kolmeen osaan. Ensimmäisessä vaiheessa tutkitaan, voivatko kohteet törmätä toisiinsa. Tämän jälkeen kiinnitetään huomio täsmällisten törmäyskohtien löytämiseen. Viimeisessä vaiheessa suoritetaan kappaleiden fysikaalisten ominaisuuksien muutoksien tarkastelu löydettyjen törmäyskohtien perusteella. Törmäysten tapauksessa kappaleiden liikettä tarkastellaan tyypillisesti liike- ja pyörimismäärän säilymisen lakien avulla. Tässä tutkielmassa keskitytään ainoastaan törmäysten havaitsemiseen. Kohteiden fysikaalisten tilojen muutokset eivät kuulu tämän tutkielman aihe alueeseen.

Animoitavien kohteiden välisten törmäysten tarkasteleminen on ollut tutkimuskohteena jo pitkään ja on edelleen. Tämä johtuu siitä, että törmäysten havaitsemien on aikakompleksisuudeltaan $O(n^2)$ ja se muodostuu usein sujuvan animaation pullonkaulaksi. Tässä n tarkoittaa animoitavissa kohteissa olevien monikulmioiden lukumäärää. Luotettavan ja tehokkaan törmäysten havaitsemisalgoritmin toteuttaminen on erittäin vaikeaa, varsinkin jos halutaan, että sitä voitaisiin käyttää kirjastotyyppisesti useissa eri sovelluksissa. Näin on lähinnä siksi, että animaatiossa käytettävien mallien tietorakenne on erilainen sovelluksien välillä. Lisäksi eri laiteympäristöissä saattaa tulla erilaisia lukujen pyöristystarkkuuksista johtuvia ongelmia. Algoritmien pitäisi myös pystyä käsittelemään syötteitä, joissa saattaa olla degeneroituneita piirteitä. Esimerkkinä on kolmion surkastuminen viivaksi tai pisteeksi. Joissain algoritmeissa suuri osa koodista käsittelee juuri syötteen poikkeustapauksia.

Tietokoneanimaatiota ajetaan tyypillisesti silmukassa, jossa kohteiden ominaisuuksiin, esimerkiksi sijaintiin ja asentoon, voidaan vaikuttaa käyttäjän antamien syötteiden mukaisesti. Toisaalta kohteita voidaan liikuttaa myös ennalta määrättyä rataa pitkin ajan funktiona. Koodikatkelma 1 kuvaa yksinkertaisen animaatiorungon, jossa fysiikkamallinnuksessa rajoitutaan ainoastaan kappaleiden pysäyttämiseen törmäyksen sattuessa.

```
while(1)
{
        HandleInput();
        UpdateLoacations();
        RenderScene();
}

 UpdateLocations()
{
       for(every object)
        {
          RestoreOldPosition();
          CalculateNewPosition();
       if(collision)
         newposition=oldposition;
        }
}
```

Koodikatkelma 1: Yksinkertainen animaatiorunko.

2. Ongelman luonne

Tyypillisesti tietokoneanimaatiossa on mukana M kappaletta paikallaan olevia kohteita ja N kappaletta liikkuvia kohteita. Näiden kohteiden kesken voi tapahtua $NM+\binom{N}{2}$ erilaista törmäystä. Tässä N liikkuvaa kohdetta voi törmätä M paikallaan olevan kohteen kanssa ja muodostaa törmäyspareja N yli 2 kappaletta. Tietenkin törmäykseen voi osallistua enemmän kuin kaksi kohdetta kerrallaan, mutta käsittelyn helpottamiseksi suoritetaan näille kohteille pareittainen törmäystarkastelu.

Jos animaatioon kuuluu useita kohteita, niin näiden muodostamien mahdollisten törmäysparien lukumäärä kasvaa nopeasti niin suureksi, että kaikkien parien tarkastelu hidastaa animaation käyttökelvottomaksi. Lähtökohta ongelman ratkaisemiseksi on siis sellaisten kohteiden karsiminen tarkastelusta, jotka eivät voi törmätä keskenään. Tässä tutkielmassa tarkastellaan aluksi periaatetasolla yleisimpiä törmäyksen havaitsemisen perusmenetelmiä erilaisille kohteiden esitysmuodoille. Tämän jälkeen keskitytään tarkastelemaan monikulmioverkoista muodostuvien kohteiden välisiä törmäyksiä. Näin on siksi, että monikulmioverkkoihin perustuvat

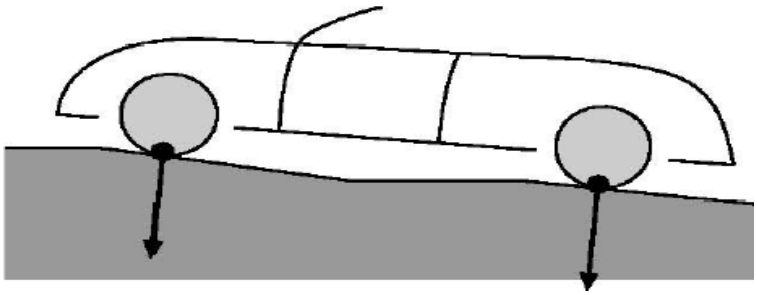
esitysmuodot ovat yleisimmin käytettyjä. Lisäksi muut esitysmuodot, esimerkiksi vokselien ja neliöllisten parametristen lappujen käyttö, palautuvat lopulta monikulmioverkkoesityksiksi. Erilaisista esitysmuodoista kerrotaan enemmän seuraavissa teoksissa: Alan Watt [28] sivuilla 27-122, Edward Angel [1] sivuilla 477-527 sekä Slater *et al*. [27] sivuilla 383-433.

3. Ratkaisumenetelmiä

Törmäystarkasteluissa käytettävien menetelmien valinta riippuu ensisijaisesti käytettävästä sovelluksesta ja sovelluksessa käytettävästä kohteen esitysmuodosta. Yksinkertaisissa tapauksissa, joissa törmäysten luonne ja suunnat tunnetaan etukäteen voidaan käyttää esimerkiksi säteen ja tason leikkaukseen perustuvia menetelmiä. Monimutkaisissa järjestelmissä, joissa törmäysten suunnat ja kohdat voivat olla mielivaltaisia, vaaditaan yleensä hierarkkiseen geometrian alijakoon perustuvia tekniikoita. Näissä tapauksissa käytetään kohteiden esitysmuotoina yleensä monikulmioverkkoja. Geometrian alijakoon perustuvissa menetelmissä voidaan olla kiinnostuneita täsmällisten törmäyskohtien löytämisestä. Joissain aikakriittisissä tapauksissa saattaa riittää pelkkä törmäyksen havaitseminen. Yksinkertaisista konvekseista monitahokkaista muodostuville kohteille voidaan käyttää Voronoin alueisiin perustuvia menetelmiä.

3.1. Säteen ja tason leikkaukseen perustuva menetelmä

Oletetaan, että auto liikkuu mäkeä ylös tasaisella alustalla. Oletetaan lisäksi, että autoon kohdistuu ainoastaan tien pinnasta aiheutuvat tukivoimat, eikä törmäystä voi tapahtua korin ja ympäristön välillä. Nyt voidaan asettaa säteet lähtemään auton renkaista siten, että ne ovat tien pinnan normaalin suuntaiset. Jos renkaasta lähtevän säteen pituus on nolla, auto kulkee täsmällisesti tien pintaa pitkin. Jos säteen pituus on positiivinen, auto on irti tien pinnasta ja sitä pitää siirtää alaspäin säteen suunnassa sen pituuden verran. Säteen ollessa negatiivinen auto on painunut tien pinnan sisään ja sitä tarvitsee nostaa säteen suunnassa sen pituuden verran ylöspäin. Erilaisia geometristen primitiivien leikkaustapauksia tarkastellaan myöhemmin. Kuva 1 esittää tason ja säteen leikkaukseen perustuvan menetelmän periaatteen. Tämä menetelmä soveltuu hyvin vastaavanlaisiin tilanteisiin, joissa kohdetta liikutetaan ympäristön geometrian mukaan. Säteen ja tason leikkaukseen perustuvassa menetelmässä approksimoidaan siis monimutkaista geometriaa joukolla säteitä. Esimerkissä auton rengas voisi hyvinkin sisältää satoja monikulmioita.



Kuva 1: Säteen ja tason leikkaukseen perustuva törmäys

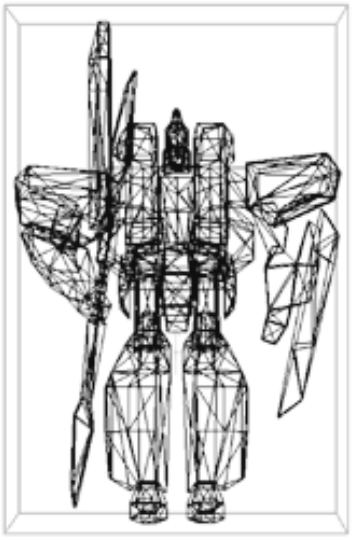
Säteen ja tason leikkaukseen perustuvia menetelmiä tarkastellaan enemmän Hainesin ja Möllerin kirjassa [15] sivuilla 343-344.

3.2. Geometrian alijakoon perustuvat menetelmät

Animaatiossa olevat kohteet muodostuvat usein sadoista, jopa sadoista tuhansista monikulmioista. Jos tarkastellaan kahden tällaisen kohteen törmäystä yksinkertaisella ”raa’an voiman” menetelmällä, joudutaan tutkimaan kaikki monikulmioparit kohteiden kesken. Mikäli kohteessa on n kappaletta monikulmiota päädytään $O(n^2)$-ongelman ratkaisemiseen. Laskentatarpeen vähentämiseksi on kehitetty menetelmiä, joiden avulla voidaan jättää tarkastelematta kokonaisia kohteen osia. Näiden menetelmien suoritusaikavaatimukset ovat tyypillisesti luokkaa $O(\log n)$, missä n on kohteessa olevien monikulmioiden lukumäärä. Tässä oletetaan, että törmäystarkastelu suoritetaan kahden sellaisen kohteen välillä, joissa on yhtä paljon monikulmiota. Suoritusajoista kerrotaan tarkemmin myöhemmin.

3.2.1. Rajoitustilavuudet

Monimutkaisen geometrian approksimoinnissa käytetään erilaisia yksinkertaisia rajoitustilavuuksia. Tämä tarkoittaa sitä, että monimutkaisen geometrian omaava kohde suljetaan yksinkertaisen geometrian sisään. Tällöin kohteille voidaan suorittaa karkean tason tömäystarkastelu testaamalla, onko niiden rajoitustilavuuksilla yhteisiä pisteitä. Jos rajoitustilavuudet ovat erillisiä, ei lisätarkastelua törmäyksen löytämiseksi tarvita. Rajoitustilavuuksien käyttö vähentää huomattavasti laskentatarvetta, koska kohteiden kaikkien monikulmioparien ei tarvitse välttämättä suorittaa. Kuva 2 esittää tilannetta, jossa paljon monikulmioita sisältävä kohde on sijoitettu suorakulmaisen särmiön muotoisen rajoitustilavuuden sisään.



Kuva 2: Kohteen rajoittaminen suorakulmaisella särmiöllä.

3.2.2. Rajoitustilavuustyyppejä

Rajoitustilavuudeksi voidaan valita mikä tahansa geometrinen muoto. Valinnassa on kuitenkin huomioitava että rajoitustilavuuden pitää pystyä peittämään kohde mahdollisimman tarkasti. Lisäksi rajoitustilavuus on pystyttävä päivittämään nopeasti kohteen liikkeen mukaan. Nämä kaksi vaatimusta ovat kuitenkin keskenään ristiriitaisia, joten oikean rajoitustilavuustyypin valinta ei ole välttämättä kovin helppoa. Seuraavaksi esitellään käytetyimpiä rajoitustilavuuksia. Vähemmän käytettyjä rajoitustilavuustyyppejä käsittelevät mm. Klosowski *et al*. [19] ja Larsen *et al*. [20].

3.2.3. Pallo

Pallon käyttö rajoitustilavuutena perustuu sen nopeaan siirtämiseen rajoitettavan kohteen mukana. Pallon siirrossa sen keskipisteelle **c** suoritetaan sama siirto-operaatio kuin siirrettävälle kohteelle. Siirroista enemmän teoksissa: Alan Watt [28] sivuilla 1-10 sekä Edward Angel [1] sivuilla 143-215. Pallon koko valitaan siten, että se pystyy rajoittamaan mielivaltaisessa asennossa olevan kohteen. Tällöin palloa ei tarvitse kiertää kohteen mukana. Tyypillisesti pallo esitetään neljällä parametrilla muodossa:

$$P = \{(x, y, z) \mid (x - \mathbf{c}_x)^2 + (y - \mathbf{c}_y)^2 + (z - \mathbf{c}_z)^2 < r^2\},$$

missä r on pallon säde ja **c** on pallon keskipiste.

Pallon käytön huono puoli on se, että sen ja rajoitettavan kohteen väliin jää paljon tyhjää. Tällöin karkean tason törmäystarkastelu voi raportoida törmäyksen vaikka kohteet eivät kuitenkaan törmää.

Pallon käyttöä rajoitustilavuutena kuvaavat mm. Bradshaw ja O'Sulliwan [7] sekä P. Hubbard [16].

3.2.4. Koordinaattiakseleiden mukaan suunnattu laatikko

Koordinaattiakselien mukaan suunnattu laatikko voidaan esittää muodossa:

$$L = \{(x,y,z) \mid |\mathbf{c}_x - x| \le r_x, |\mathbf{c}_y - y| \le r_y, |\mathbf{c}_z - z| \le r_z\},$$

missä **c** on laatikon keskipiste , r_x, r_y ja r_z ovat laatikon x-, y - ja z-akselien suuntaiset särmien pituuksien puolikkaat. Kohteen siirron ja kierron yhteydessä laatikon särmät joudutaan laskemaan uudestaan siten, että laatikko on aina koordinaattiakselien suuntainen.

Koordinaattiakseleiden mukaan suunnatun laatikon käyttöä kuvaa yksityiskohtaisesti van den Bergen [6].

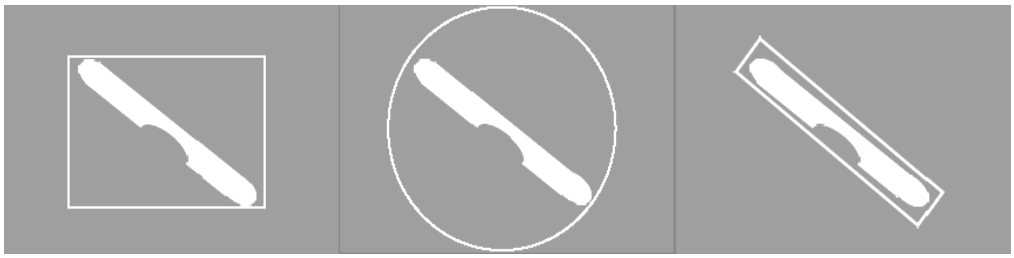
3.2.5. Kohteen mukaan suunnattu laatikko

Kohteen mukaan suunnattu laatikko muodostetaan siten, että se rajoittaa kohteen mahdollisimman tarkasti. Tällöin kohteen ja rajoitustilavuuden väliin jäävä tila minimoituu. Kohteen mukaan suunnatun laatikon tyypillinen esitysmuoto on:

$$L = \{\mathbf{c} + a_1\mathbf{v}^1 + a_2\mathbf{v}^2 + a_3\mathbf{v}^3\},$$

missä **c** on laatikon keskipiste, a_1, a_2 ja a_3 ovat laatikon sivujen pituuksien puolikkaat ja vektorit $\mathbf{v}^1$, $\mathbf{v}^2$ ja $\mathbf{v}^3$ ilmaisevat laatikon suunnan avaruudessa. Kohteen mukaan suunnatun laatikon muodostamisesta kerrotaan myöhemmin. Kohteen siirron ja kierron yhteydessä joudutaan myös rajoitustilavuudelle suorittamaan samat operaatiot. Kohteen mukaan suunnatun laatikon käyttöä on kuvattu Gottschalkin ja Manochan konferenssijulkaisussa [13].

Kuvassa 3 sama kohde on rajoitettu eri rajoitustilavuuksilla. Ensimmäiseksi on esitetty koordinaattiakselien suuntainen laatikko, toisena pallo sekä viimeisenä kohteen mukaan suunnattu laatikko.

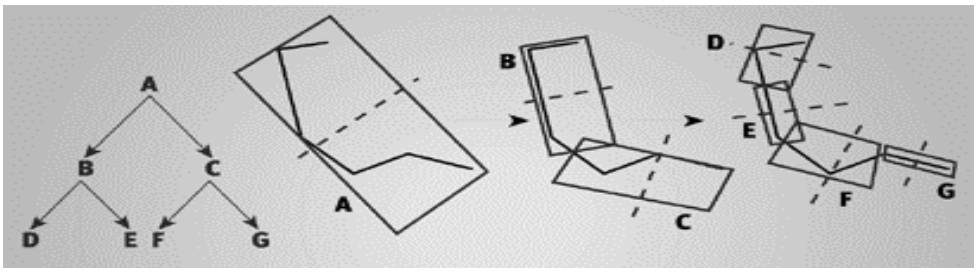


Kuva 3: Kohteen rajoittaminen eri rajoitustilavuuksilla

3.2.6. Rajoitustilavuushierarkiat

Jos kohteiden rajoitustilavuudet menevät päällekkäin, on suoritettava tarkastelu rajoitustilavuuksien sisältämien monikulmioiden välillä. Yleensä geometriaa jaetaan rekursiivisesti pienempiin rajoitustilavuuksiin siten, että päätason geometriaa pilkotaan pienempiin osiin niin kauan, että lehtitason rajoitustilavuus sisältää ainoastaan yhden monikulmion. Geometrian jakaminen muistuttaa läheisesti pikalajittelua. Aluksi etsitään tilavuus, joka sulkee sisäänsä päätason geometrian. Tämän jälkeen valitaan jokin taso, jonka suhteen monikulmiot jaetaan. Monikulmiot sijoitetaan jakotason mukaan uusiin rajoitustilavuuksiin sen mukaan, kummalle puolelle jakotasoa monikulmion painopiste sijoittuu. Näin jatketaan, kunnes jakoa ei voida enää suorittaa.

Geometrian alijaon toteuttamisessa käytetään yleensä binääripuuta. Valinta on tehty helpohkon toteutettavuuden takia. Muita alijaon toteuttamistapoja kuvataan myöhemmin. Kuvassa 4 havainnollistetaan viidestä segmentistä muodostuvan murtoviivan alijaon suorittamista. Päätasolla rajoitustilavuus A sisältää koko murtoviivan. Päätaso A jaetaan osiin B ja C katkoviivalla osoitetun jakotason mukaisesti. Jakoa jatketaan rekursiivisesti tasolle D, E, F ja G asti. Rajoitustilavuudelle F on vielä suoritettava yksi jako, jotta kaikkiin solmutason rajoitustilavuuksiin saadaan ainoastaan yksi alkeistason piirre.



Kuva 4: Murtoviivan alijako rajoitustilavuuksiin.

3.2.7. Törmäyskyselyt rajoitustilavuushierarkiassa

Oletetaan, että animaatiossa on kaksi kohdetta A ja B, joille on muodostettu rajoitustilavuushierarkiat. Ensimmäisessä vaiheessa tutkitaan, menevätkö päätason rajoitustilavuudet päällekkäin. Mikäli päällekkäisyyttä ei havaita, testaaminen voidaan lopettaa välittömästi. Jos rajoitustilavuuksien välillä havaitaan päällekkäisyys, kuljetaan hierarkioita A ja B rekursiivisesti lehtitasolle saakka. Rekursion haara lopetetaan, jos se tuottaa erilliset rajoitustilavuudet. Lehtitasolla käytetään metodia, joka löytää monikulmioiden väliset leikkaukset. Joissain tapauksissa monikulmioiden välisiä leikkauksia ei etsitä, vaan pelkkä rajoitustilavuuksien päällekkäisyyksien viimeisen tason löytäminen riittää. Tällainen menettely on ominaista erityisen aikakriittisille sovelluksille. P. Hubbard [16] on tarkastellut tämän tyyppisiä tilanteita. Rajoitustilavuushierarkian lehtitason primitiivit ovat yleensä kolmiota. Tehokas menetelmä kolmioiden leikkaamiseen esitetään myöhemmin. Koodikatkelma 2 esittää törmäyskyselyn pseudokoodina.

```
1.      QueryCollision(R,A,B)
2.      {   if(!CollBV(A,B) return;
3.          if(Leaf(A) AND LEAF(B))
4.            { CollTris(R,A,B);
5.              return;
6.            }
7.          if(DescendA(A,B))
8.            { QueryCollision(R,Left(A),B);
9.              QueryCollision(R,Right(A),B);
10.           }
11.         else
12.           { QueryCollision(R,A,Left(B));
13.             QueryCollision(R,A,Right(B));
14.           }
```

Koodikatkelma 2: Törmäyskysely rajoitustilavuushierarkiassa.

Koodikatkelmassa 2 rivillä 4 oleva metodi CollTtris(R,A,B) sijoittaa lehtitasolta löydetyt kolmioiden törmäykset listaan R. Rivillä 7 oleva metodi DescendA(A,B) tutkii, mihin suuntaan hierarkiassa edetään seuraavaksi. Rajoitustilavuushierarkiassa kulkemista voidaan suorittaa monilla eri tavoilla, mutta yleisin tapa on, että lähdetään kulkemaan siihen haaraan, jonka rajoitustilavuus on suurempi.

Ennen törmäyskyselyn suorittamista kohteiden rajoitustilavuudet pitää sijoittaa samaan tarkasteluavaruuteen. Tarkastelu voidaan suorittaa siirtämällä toinen kohteista toisen paikalliseen koordinaatistoon tai siirtämällä molemmat kohteet yhteiseen maailmankoordinaatistoon. Visualisoinnin kannalta on luonnollisempaa sijoittaa kohteet maailman koordinaatistoon. Tällöin kohteiden paikat määräytyvät valmiiksi animaation mukaan, ja vältytään ylimääräiseltä kohteen siirtämiseltä toisen kohteen paikalliseen koordinaatistoon.

3.2.8. Kustannusfunktio

Rajoitustilavuushierarkioihin perustuvissa törmäyskyselyissä kulutettua aikaa t kuvataan tyypillisesti funktiolla $t=N_vC_v+N_pC_p+N_uC_u$. Tässä N_v kuvaa rajoitustilavuuksien välillä tehtyjen vertailujen lukumäärää ja C_v yhden vertailun kustannusta. Vastaavasti N_p ja C_p kuvaavat primitiivitasolla tehtyihin tarkasteluihin kuluvaa aikaa. Viimeisenä summassa mukana olevat N_u ja C_u kuvaavat rajoitustilavuuksien päivittämiseen kuluvaa aikaa. Monessa yhteydessä rajoitustilavuuksien päivittämiseen kuluvaa aikaa ei ole merkitty näkyviin, mutta se on erittäin tärkeä animaatiossa, jossa kohteet ovat jatkuvassa liikkeessä. Näin on siksi, että rajoitustilavuutta pitää päivittää aina, kun kohde liikkuu tai kiertyy.

Kustannusfunktion suuruuteen vaikuttavat asiat ovat keskenään ristiriitaisia, joten optimaalisen systeemin löytäminen on vaikeaa. Jos rajoitustilavuudeksi valitaan pallo, kasvaa tekijä N_v nopeasti suureksi, mutta C_u pysyy pienenä, koska pallo on invariantti kierrolle. Kohteen mukaan suunnatun särmiön tapauksessa N_v on edellä mainittua pienempi, mutta C_v ja C_u kasvavat huomattavasti.

3.3. Voronoin alueisiin perustuvat menetelmät

Voronoin alueisiin perustuvia menetelmiä käytetään tyypillisesti sellaisissa sovelluksissa, joissa kohteet muodostuvat monitahokkaista, jotka sisältävät suhteellisen vähän monikulmioita. Tällaisten monitahokkaiden sanotaan muodostuvan piirteistä X, joita ovat kulmapisteet, monikulmion särmät sekä monikulmion pinnat.

Monitahokkaan P piirteen X Voronoin alueella VP(X) tarkoitetaan sellaista avaruuden osaa, jossa monitahokkaan P ulkopuolella olevat avaruuden pisteet ovat lähempänä piirrettä X kuin mitään muuta monitahokkaan P piirrettä Y. Olkoon A ja B kaksi monitahokasta, joissa on piirteet $P_A \in A$ ja $P_B \in B$. Jos nyt on olemassa avaruuden pisteet $\mathbf{x} \in P_A$ ja $\mathbf{y} \in P_B$ siten, että $\mathbf{x} \in VB(P_B)$ ja $y \in VA(P_A)$,

niin pisteet **x** ja **y** ovat lähimmät mahdolliset pisteet monitahokkaiden A ja B välillä. Pistepari **x** ja **y** ei välttämättä ole yksikäsitteinen, mutta väliä $|\mathbf{x}-\mathbf{y}|$ lyhyempää väliä monitahokkaiden A ja B välillä ei ole olemassa. Näin on esimerkiksi silloin, kun kaksi kuutiota asetetaan siten, että niiden tahkot ovat yhdensuuntaisilla tasoilla. Törmäyksen tapauksessa $\mathbf{x}=\mathbf{y}$. Brian Mirtitch [23] on kehittänyt tehokkaan Voronoin alueisiin perustuvan törmäysten havaitsemismenetelmän, jonka nimi on V-Clip. Voronoin alueiden käyttöä kuvataan tarkasti Min Linin [21] väitöstutkimuksessa.

4. Usean kohteen muodostama animaatio

Tietokoneanimaatiossa kohteiden välinen törmäyksien havaitseminen jaetaan yleensä kahteen eri vaiheeseen. Ensimmäisessä vaiheessa engl. (Broad Phase) etsitään sellaiset kohdeparit, jotka mahdollisesti törmäävät toisiinsa. Toisessa vaiheessa engl. (Narrow Phase) suoritetaan mahdollisille törmäyspareille täsmällisten törmäyskohtien etsiminen. Jos täsmällisten törmäyskohtien etsimiseen käytetään rajoitustilavuushierarkioihin perustuvia menetelmiä, voidaan kohteen osien poissulkemista jatkotarkastelusta pitää myös ensimmäiseen vaiheeseen kuuluvana toimenpiteenä. Näin ollen törmäys-tarkastelun jakaminen kahteen vaiheeseen on hiukan ontuva. Ajatus on kuitenkin se, että mahdollisimman varhaisessa vaiheessa pyritään karsimaan laskennasta sellaiset kohteet tai kohteiden osat, jotka eivät voi törmätä toisiinsa.

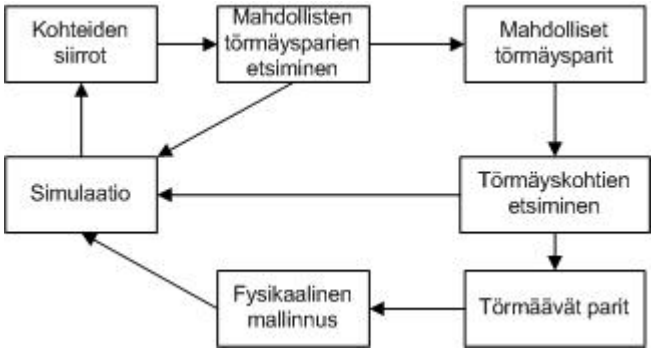
Kuvassa 5 simulaatio tarkoittaa animaation visualisointia. Simulaatiossa olevilla kohteille suoritetaan mahdollisten törmäysparien etsiminen, jos ne ovat liikkuneet edellisen tarkastelun jälkeen. Mahdollisille törmäyspareille suoritetaan edelleen täsmällisten törmäyskohtien etsiminen. Tämän jälkeen törmääville pareille voidaan tehdä sovelluskohtainen fysikaalinen mallinnus, jonka jälkeen palataan takaisin simulaatiovaiheeseen.

Mahdollisten törmäysparien etsiminen suoritetaan kohteiden päätason rajoitustilavuuksien avulla. Kohteiden päätason rajoitustilavuuksien päällekkäisyys ei vielä tarkoita, että kohteet törmäisivät, joten lisätarkastelu tarvitaan täsmällisten törmäyskohtien löytämiseksi. Tarkastelussa saattaa käydä myös niin, että täsmällisten törmäyskohtien etsimisessä päädytään tilanteeseen, jossa kohteet eivät törmää.

Animaatiosysteemissä on tärkeää, että simulaatiovaiheeseen palataan aina mahdollisimman nopeasti. Joissain menetelmissä [16] törmäystarkastelu-vaiheelle annetaan tietty aikaraja. Kun aikaraja täyttyy, niin palataan takaisin simulaatioon. Muissa tapauksissa paluu animaatioon suoritetaan heti, kun se

on mahdollista. Näin on esimerkiksi silloin, jos mahdollisia törmäyspareja ei löydy. Seuraavaksi tarkastellaan menetelmää, jonka avulla mahdollisten törmäysparien löytämien voidaan suorittaa keskimäärin ajassa $O(n \log n + k)$, missä n on kohteiden lukumäärä ja k on mahdollisesti törmäävien kohteiden lukumäärä. Parhaassa tapauksessa menetelmä toimii ajassa $O(n + k)$.

Usean kohteen muodostaman animaation törmäyksiä kuvaavat mm. Cohen *et al*. [8] ja Hudson *et al*. [17].



Kuva 5: Animaatiosysteemin periaate.

5. Mahdollisten törmäysparien etsiminen

Mahdollisten törmäysparien etsimisessä voidaan hyödyntää animaation ajallista koherenssia. Tämä tarkoittaa sitä, että kohteiden liike on suhteellisen pientä, eivätkä niiden paikat ja asennot muutu ratkaisevasti lyhyellä aikavälillä.

Valitaan rajoitustilavuudeksi sellainen koordinaattiakseleiden suuntainen kuutio, joka pystyy sulkemaan sisäänsä mielivaltaisessa asennossa olevan kohteen. Tällöin kuutiota ei tarvitse kiertää kohteen mukana, vaan pelkkä siirtäminen riittää. Kuutiota siirretään siten, että sen keskipisteelle suoritetaan sama siirto-operaatio kuin kuution rajoittamalle kohteelle. Tämän jälkeen kuution särmille lasketaan uudet alku- ja loppupisteet keskipisteen ja särmän pituuden avulla. Jos nyt kaksi tällaista kuutiota sisältää yhteisiä pisteitä, niin kaikki niiden yksiulotteiset koordinaattiakselien suuntaiset särmät menevät pareittain päällekkäin. Ongelma voidaan nyt yksinkertaistaa kolmen yksiulotteisen särmälistan tarkasteluksi. Menetelmää kutsutaan kirjallisuudessa nimellä sweep-and-prune.

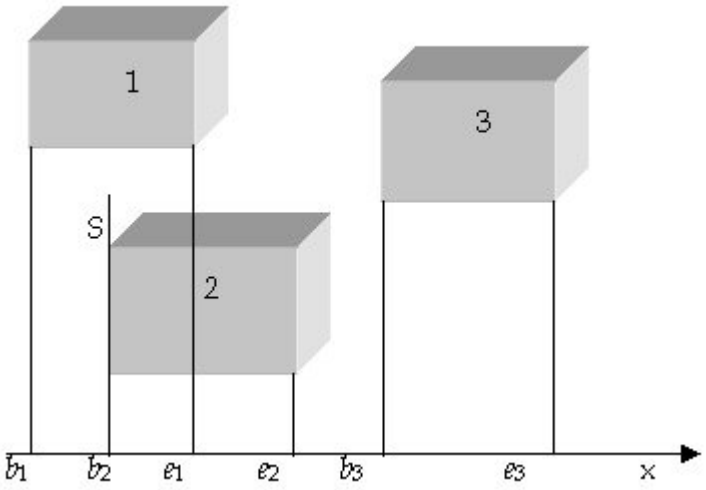
Jokaiselle animaatiossa mukana olevan kohteen rajoitustilavuudelle muodostetaan aluksi koordinaattiakselien suuntaiset intervallit. Intervallille annetaan alkupiste, loppupiste ja tunnistenumero, joka kertoo mihin

kohteeseen se kuuluu. Seuraavaksi muodostetaan kolme intervallilistaa x-, y- ja z-akselin suuntaisia intervalleja varten. Näihin listoihin laitetaan kaikkien animaatiossa mukana olevien kohteiden ko. akselien suuntaiset intervallit, jonka jälkeen listat lajitellaan nousevaan järjestykseen. Tässä on hyvä käyttää aputietorakennetta, jossa on intervallin tunniste, intervallin pisteen arvo ja intervallin pisteen tyyppi, joka voi olla alku- tai loppupiste.

Mahdollisten törmäysparien löytämiseksi kuljetaan x-, y- ja z-akselien suuntaiset intervallilistat alusta loppuun. Listan kulkemiseksi muodostetaan aktiivisten intervallien lista, johon laitetaan intervalli aina, kun sen alkupiste löydetään. Jos toisen intervallin alkupiste löydetään intervallilistasta ennen aktiivisessa listassa olevan intervallin loppupistettä, merkitään nämä kaksi intervallia päällekkäisiksi. Jos aktiivisessa listassa on useita intervalleja, merkitään viimeiseksi lisätty intervalli päällekkäiseksi kaikkien aktiivisessa listassa olevien intervallien, paitsi itsensä kanssa. Intervalli poistetaan aktiivisesta listasta, kun sen loppupiste tulee vastaan intervallilistan selaamisessa. Jos kahden kohteen intervallien välille tulee kolme päällekkäisyyttä, niin näistä kohteista muodostetaan törmäyspari, jolle suoritetaan myöhemmin täsmällisten törmäyskohtien etsiminen. Ohjelmoitaessa menetelmää on luonnollista käyttää törmäysparille struktuuria, jossa on törmäävien kohteiden yksilöivät kokonaisluvut. Kolme päällekkäisyyttä kohteiden intervallien kesken voi löytyä siis silloin, kun kaikkien kolmen koordinaattiakselin suuntaiset intervallilistat on käyty läpi.

Tämä menetelmä lajittelee intervallilistan ajassa $O(n\log n)$, mikäli käytetään jotain muuta yleistä lajittelumenetelmää kuin kuplalajittelu. Lisäksi se tarvitsee ajan $O(n)$ intervallilistan läpikäymiseen sekä ajan $O(k)$ k:n intervallin päällekkäisyyden merkkaamiseen. Näin ollen menetelmän suoritusaika kokonaisuudessaan on $O(n\log n + k)$. Ajallisen koherenssin vuoksi intervallilista ei muutu kovin paljoa vertailujen välillä, joten käyttämällä lisäyslajittelua voidaan lähes järjestyksessä oleva intervallilista lajitella ajassa $O(n)$. Tällöin parhaassa tapauksessa menetelmän suoritusaika on $O(n + k)$.

Kuvassa 6 esitetään menetelmän periaate x-akselin osalta. Selaussuoran S kohdalla kohteen 1 x-akselin suuntainen intervalli on aktiivisessa listassa, kun saavutaan kohteen 2 intervallin alkukohtaan b_2. Tällöin raportoidaan päällekkäisyys kohteille 1 ja 2 x-akselin osalta. Kohde 3 ei mene päällekkäin minkään kohteen kanssa eikä siis voi törmätä mihinkään. Selaussuora S kuvaa tässä intervallilistan läpikäyntiä.



Kuva 6: Sweep-and-Prune menetelmän periaate.

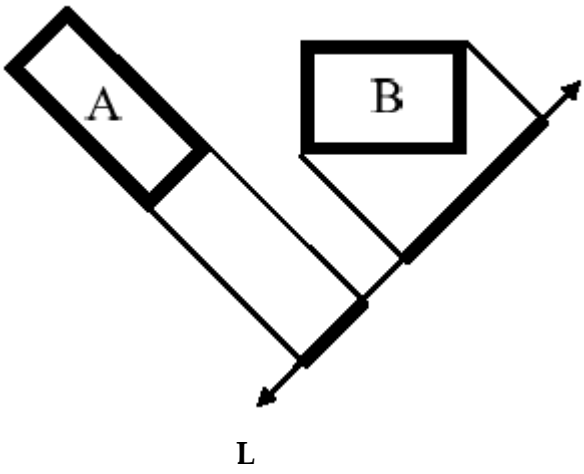
Tässä kuvatun menetelmän tuloksena saadaan lista, joka sisältää ne kohdeparit, jotka voivat törmätä toisiinsa. Listassa oleville kohteille on vielä suoritettava täsmällisten törmäyskohtien etsiminen. Tässä esitettyä menetelmää on käsitelty myös Hainesin ja Möllerin kirjassa [15] sivuilla 358-360. Seuraavaksi tarkastellaan menetelmää, jonka avulla voidaan etsiä kolmioverkoista muodostuvien kohteiden kolmiot, jotka törmäävät keskenään.

6. Kohteen mukaan suunnatun rajoitustilavuuden menetelmä

Kohteen mukaan suunnatun rajoitustilavuuden menetelmä perustuu kolmioverkoista muodostuvien kohteiden geometrian alijakoon. Jos kohteet muodostuvat monikulmioverkoista, niin niille pitää aluksi suorittaa kolmiointi. Monikulmioiden kolmiointia kuvaavat hyvin Fournier ja Montuno [11]. Kohteille muodostetaan aluksi rajoitustilavuushierarkia kohdassa 3.2.5 kuvatuista rajoitustilavuuksista. Näille hierarkioille suoritetaan koodikatkelman 2 mukaisia kyselyitä törmäyksien havaitsemiseksi. Koodikatkelmassa 2 rivillä 2 oleva metodi CollBV(A,B) testaa, ovatko rajoitustilavuudet A ja B päällekkäisiä. Päällekkäisyyden testaamisessa kohteet projisoidaan erottavalle akselille L kuvan 6 mukaisesti. Jos kohteiden projektiot ovat erillisiä, niin myös kohteet ovat erillisiä. Erottavasta akselista ja päällekkäisyyden testaamisesta kerrotaan myöhemmin. Jos testattavana on lehtitason rajoitustilavuudet, niin tällöin tutkitaan, leikkaavatko ko. rajoitustilavuuksien sisältämät kolmiot toisensa.

Suurin ongelma kohteen mukaan suunnatun rajoitustilavuuden menetelmässä on rajoitustilavuuden sovittaminen kohteen geometriaan siten, että tyhjä tila kohteen ja rajoitustilavuuden välissä jäisi mahdollisimman pieneksi. Seuraavaksi tarkastellaan menetelmiä rajoitustilavuuden

sovittamiseksi kohteen geometriaan. Tässä luvussa esitetyt asiat perustuvat Stefan Gottschalkin [12] väitöstutkimukseen.



Kuva 7: Rajoitustilavuuksien projektiot erottavalle akselille **L**.

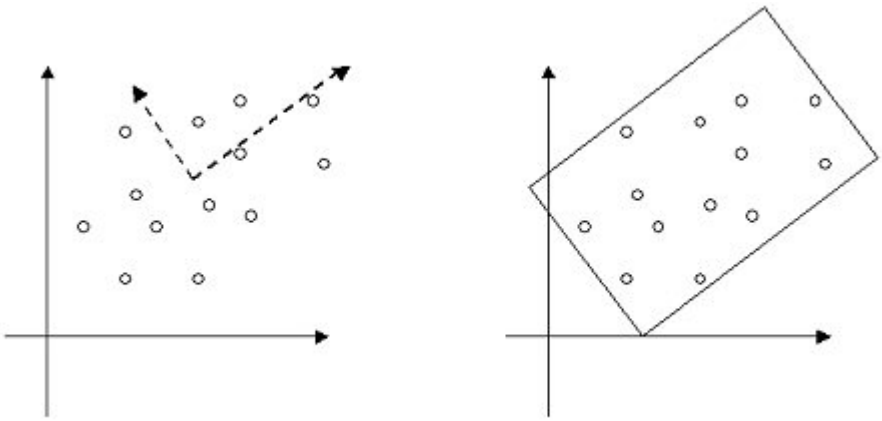
6.1. Kohteen geometrian sovittaminen rajoitustilavuuteen

Tehokkaan rajoitustilavuuden löytämiseksi tarkastellaan kohteen geometristen ominaisuuksien tilastollista jakaumaa. Geometrisena ominaisuutena voidaan pitää monikulmioverkon kulmapisteitä, särmiä tai itse monikulmioita. Nämä ominaisuudet muodostavat avaruuteen pistejoukon, jonka muotoa voidaan approksimoida kovarianssimatriisilla **C** ja keskipistettä vektorilla **m**. Kolmiulotteisessa avaruudessa kovarianssimatriisi on symmetrinen ja reaalinen 3x3-matriisi.

Kovarianssi voidaan ajatella normaalijakauman siirtämisenä yksiulotteisesta avaruudesta kolmiulotteiseen avaruuteen. Tällöin normaalijakauman tasokuvaus muuttuu kolmiulotteisen muodon kuvaamiseksi. Tämä muoto kuvaa kohteen muodon tilastollisen sijoittumisen kolmiulotteiseen avaruuteen. Nyt kohteen muodon jakauman projektio $P_v=\mathbf{v}^T\mathbf{C}\mathbf{v}$ ilmaisee kohteen pisteiden varianssin vektorilla **v**. Kovarianssi-matriisin ominaisvektorit antavat ne suunnat, joissa pisteiden varianssit saavat maksimi- ja minimiarvonsa.

Geometrian sovittamiseksi rajoitustilavuuteen on ensin määritettävä kovarianssimatriisi ja laskettava tälle ominaisvektorit. Tämän jälkeen rajoitustilavuus voidaan määrittää kovarianssimatriisin ominaisvektorien

suunnissa olevien kulmapisteiden minimi- ja maksimiarvojen avulla. Kuvassa 8 nuolet esittävät pistejoukon kovarianssimatriisin ominaisvektoreita.



Kuva 8: Pistejoukon suunnan luonnehtiminen kovarianssimatriisin ominaisvektoreilla.

6.1.1. Rajoitustilavuuden sovittaminen kulmapisteiden avulla

Ensimmäinen tehtävä rajoitustilavuuden sovittamiseksi kulmapistejoukolle on kovarianssimatriisin määrittäminen. Oletetaan, että kohde K muodostuu kulmapisteistä $\mathbf{p}^1,\mathbf{p}^2,\ldots,\mathbf{p}^n$. Tämän pistejoukon kovarianssimatriisin **C** alkio $\mathbf{C}_{ij}$ voidaan ilmoittaa muodossa

$$\mathbf{C}_{ij}=\mathrm{Cov}(\mathbf{x}_i,\mathbf{x}_j)=E[(\mathbf{x}_i-\mathbf{m}_i)(\mathbf{x}_j-\mathbf{m}_j)].$$

Tässä **x** on kohteen K mielivaltaisesti valittu kulmapiste ja **m** on kulmapistejoukon keskipiste. E tarkoittaa kulmapistejoukon odotusarvoa, ja keskipiste **m** voidaan ilmoittaa myös muodossa $\mathbf{m}=E[\mathbf{x}]$. Alaindeksit i ja j kuvaavat kulmapisteen **x** yksittäisiä koordinaattiarvoja. Jos i=0, niin $\mathbf{x}_0$ tarkoittaa kulmapisteen **x** x-koordinaatin arvoa. Kulmapisteen **x** y-ja z-koordinaatit saadaan i:n arvoilla 1 ja 2. Vastaavasti käyttäytyy myös alaindeksi j. Kovarianssi tarkoittaa kahden muuttujan yhteisvaihtelua, joten tässä tarkastellaan kahden kulmapisteen eri koordinaattiarvojen keskinäistä vaihtelua.

Kovarianssi ilmaistaan yleensä muodossa:

$$\mathbf{C}_{ij}=\mathrm{Cov}(\mathbf{x}_i,\mathbf{x}_j)=E[\mathbf{x}_i\mathbf{x}_j]-\mathbf{m}_i\mathbf{m}_j,$$

jolloin kovarianssi *n* kulmapisteelle

$$\mathbf{C}_{ij}=\mathrm{Cov}(\mathbf{x}_i,\mathbf{x}_j)=\frac{1}{n}\sum_{k=1}^{n}[\mathbf{p}_i^k\mathbf{p}_j^k]-\mathbf{m}_i\mathbf{m}_j,$$

$$\text{missä } \mathbf{m}_i=\mathrm{E}[\mathbf{x}_i]=\frac{1}{n}\sum_{k=1}^{n}\mathbf{p}_i^k .$$

Tämä kuvaa kulmapistejoukon keskipisteen i-koordinaattia. Jos i=0 saadaan keskipisteen x-koordinaatti. Vastaavasti y- ja z-koordinaatit saadaan i:n arvoilla 1 ja 2.

Kovarianssimatriisin **C** määrittämisen jälkeen lasketaan sen ominaisarvot, joiden avulla lasketaan edelleen ominaisvektorit. Matriisin **C** ominaisarvo λ on sellainen skalaari arvo, jolla $\mathbf{Cv}=\lambda\mathbf{v}$ silloin, kun vektori **v** ei ole nollavektori. Nyt ominaisarvot voidaan ratkaista kirjoittamalla ominaisarvoyhtälö muodossa $(\mathbf{C}-\lambda\mathbf{I})\mathbf{v}=0$, missä **I** on identtinen 3x3-matriisi. Ominaisarvoyhtälö voidaan nyt ratkaista λ:n polynomina, jos $\det(\mathbf{C}-\lambda\mathbf{I})=0$. Ominaisarvojen ja ominaisvektoreiden määrittämisessä voidaan käyttää esimerkiksi Jacobi-Davidson [18] menetelmää.

Ominaisarvojen määrittämisen jälkeen ominaisvektorit saadaan ratkaisemalla yhtälö $\mathbf{Cv}=\lambda\mathbf{v}$. Tässä tapauksessa ominaisarvoa λ vastaa ominaisvektori **v**, joka kuuluu oikeakätiseen koordinaattijärjestelmään.

Rajoitustilavuuden ulottuvuuksien laskemiseksi projisoidaan kohteen kaikki kulmapisteet $\mathbf{p}^k$ ominaisvektorien $\mathbf{v}^1$,$\mathbf{v}^2$ ja $\mathbf{v}^3$ suuntiin. Ylä- ja alarajat rajoitustilavuuden ulottuvuuksille saadaan nyt seuraavasti pistetulolla:

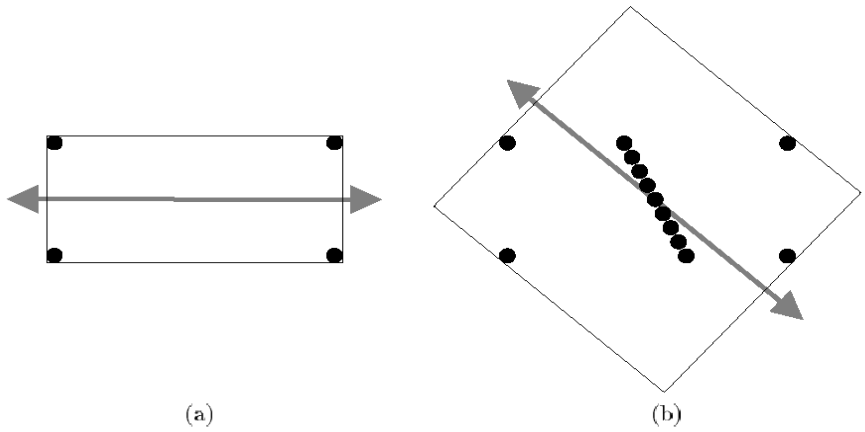
$$u^1=\max(\mathbf{v}^1\bullet\mathbf{p}^k)$$
$$u^2=\max(\mathbf{v}^2\bullet\mathbf{p}^k)$$
$$u^3=\max(\mathbf{v}^3\bullet\mathbf{p}^k)$$
$$l^1=\min(\mathbf{v}^1\bullet\mathbf{p}^k)$$
$$l^2=\min(\mathbf{v}^2\bullet\mathbf{p}^k)$$
$$l^3=\min(\mathbf{v}^3\bullet\mathbf{p}^k).$$

Nyt rajoitustilavuuden keskipiste **c** voidaan laskea ulottuvuuksien avulla

$$\mathbf{c}=\frac{1}{2}(l^1+u^1)\mathbf{v}^1+\frac{1}{2}(l^2+u^2)\mathbf{v}^2+\frac{1}{2}(l^3+u^3)\mathbf{v}^3.$$

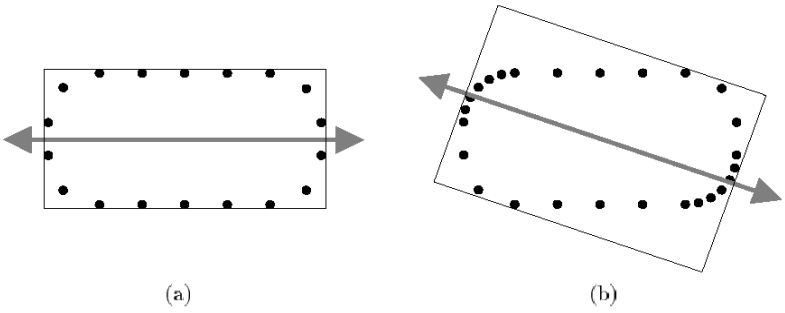
Kulmapisteiden jakaumaan perustuva rajoitustilavuuden määrittäminen on kaikkein yksinkertaisin menetelmä kohteen rajoittamiseksi. Tämä menetelmä toimii hyvin silloin, kun kulmapisteet ovat jakautuneet suhteellisen tasaisesti

avaruuteen. Jos kohteessa on joillain alueilla tiheästi kulmapisteitä, niin nämä voivat vaikuttaa epäedullisesti rajoitustilavuuden suuntaan. Suunnan kiertymisen lisäksi rajoitustilavuuden koko kasvaa liian suureksi. Kuva 9 havainnollistaa tätä tapausta. Kohdassa (a) olevan kohteen sisään on lisätty joukko kulmapisteitä, jotka aiheuttavat virheellisen rajoitustilavuuden muodostumisen kohdassa (b).



Kuva 9: Kulmapistetihentymän vaikutus rajoitustilavuuteen.

Kuvan 8 mukaista tilannetta voidaan yrittää parantaa siten, että jätetään kohteen sisimmäiset kulmapisteet huomiotta ja tarkastellaan ainoastaan äärimmäisiä kulmapisteitä. Yleensä tilanne paranee huomattavasti, mutta joskus voidaan joutua tilanteeseen, jossa kohteen reunoilla olevien kulmapisteiden alueelliset tiheys erot voivat tuottaa huonosti sopivan rajoitustilavuuden. Kuva 10 havainnollistaa tätä tapausta.



Kuva 10: Kohteen reunalla olevien kulmapisteiden tiheyserojen vaikutus rajoitustilavuuteen.

Kulmapisteiden käyttäminen rajoitustilavuuksien määrittämisessä johtaa yleensä epäedulliseen lopputulokseen. Tämän vuoksi rajoitustilavuuden määrittämiseksi käytetään tyypillisesti kohteen konveksia peitettä tai kohteen muodostavia monikulmiota. Jos kohteen monikulmioverkolla ei ole sisäistä rakennetta, niin monikulmioverkon pinnan kolmioita voidaan käyttää kohteen konveksin peitteen approksimoinnissa. Seuraavaksi tarkastellaan rajoitustilavuuden sovittamista kohteelle sen pinnan kolmioiden avulla.

6.1.2. Rajoitustilavuuden sovittaminen kolmioiden avulla

Oletetaan, että kohde muodostuu kolmioverkosta, jossa on n kolmiota. Kohteen yhteen kolmioon viitataan yläindeksillä k ja kolmioverkkoon yläindeksillä M. Vektori $\mathbf{x}$ kuvaa yksittäisen kolmion mielivaltaisesti valittua pistettä ja vektori $\mathbf{m}^k$ ilmaisee kolmion k keskipisteen. Olkoon kolmioverkon keskipiste vektori $\mathbf{m}^M$ ja skalaari A^k kolmion k pinta-ala. Merkitään lisäksi kolmion k kulmapisteitä vektoreilla $\mathbf{p}^k$, $\mathbf{q}^k$ ja $\mathbf{r}^k$ sekä kyseisen kolmion integrointialuetta merkinnällä Ω. Nyt kolmioverkon kolmion k piste $\mathbf{x}$ voidaan esittää muodossa:

$$\mathbf{x}^k(s,t)=\mathbf{p}^k+s\mathbf{u}^k+t\mathbf{v}^k, \text{ missä } s\in[0,1] \text{ ja } t\in[0,1-s], \mathbf{u}^k=\mathbf{q}^k-\mathbf{p}^k \text{ ja } \mathbf{v}^k=\mathbf{r}^k-\mathbf{p}^k.$$

Pintaintegraali kolmiolle k alueessa $\Omega=\{(s,t) \mid s\in[0,1], t\in[0,1-s]\}$ on nyt

$$\int_{\Omega} f(s,t)dA = |\mathbf{u}\times\mathbf{v}| \int_0^1 \int_0^{1-s} f(s,t)dtds\,.$$

Aikaisemmasta muistetaan, että kovarianssimatriisille $\mathbf{C}$

$$\mathbf{C}_{ij}=E[\mathbf{x}_i\mathbf{x}_j]-E[\mathbf{x}_i]E[\mathbf{x}_j].$$

Nyt muuttujat $\mathbf{x}_i$ ja $\mathbf{x}_j$ kuvaavat kolmioiden pinta-alojen suuruudet suunnissa $\mathbf{x}_i$ ja $\mathbf{x}_j$, joten odotusarvoiksi saadaan kolmioverkossa M

$$E[\mathbf{x}_i]=\frac{\int_M \mathbf{x}_i dA}{\int_M dA}$$

$$\mathrm{E}[\mathbf{x}_j] = \frac{\int_M \mathbf{x}_j dA}{\int_M dA}$$

$$\mathrm{E}[\mathbf{x}_i \mathbf{x}_j] = \frac{\int_M \mathbf{x}_i \mathbf{x}_j dA}{\int_M dA}.$$

Edellä esitetyt integraalit kolmioverkossa M voidaan esittää kolmioverkon kolmioiden integraalien summana. Odotusarvojen laskemisessa tarvittava nimittäjä saadaan seuraavasti:

$$\begin{aligned}
\int_M dA &= \sum_k \int_\Omega dA \\
&= \sum_k |\, \mathbf{u}^k \mathrm{x} \mathbf{v}^k \,| \int_0^1 \int_0^{1-s} dt ds \\
&= \sum_k |\, \mathbf{u}^k \mathrm{x} \mathbf{v}^k \,| \int_0^1 (1-s) ds \\
&= \sum_k |\, \mathbf{u}^k \mathrm{x} \mathbf{v}^k \,| \frac{1}{2} \\
&= \sum_k A^k \\
&= A^M.
\end{aligned}$$

Tässä A^k on yksittäisen kolmion k pinta-ala ja A^M on kolmioverkon pinta-ala. Nyt odotusarvojen lausekkeet yksinkertaistuvat muotoon:

$$\mathrm{E}[\mathbf{x}_i] = \frac{1}{A^M} \int_M \mathbf{x}_i dA$$

$$\mathrm{E}[\mathbf{x}_i] = \frac{1}{A^M} \int_M \mathbf{x}_j dA$$

$$\mathrm{E}[\mathbf{x}_i \mathbf{x}_j] = \frac{1}{A^M} \int_M \mathbf{x}_i \mathbf{x}_j dA.$$

Nyt odotusarvo $\mathrm{E}[\mathbf{x}_i]$ voidaan laskea seuraavasti

$$\mathrm{E}[\mathbf{x}_i] = \frac{1}{A^M} \int_M \mathbf{x}_i dA = \frac{1}{A^M} \sum_k \int_\Omega \mathbf{x}_i^k dA,$$

ja edelleen

$$\frac{1}{A^M}\sum_k \int_\Omega \mathbf{x}_i^k dA = |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 \int_0^{1-s} \mathbf{x}_i^k dt ds$$

$$|\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 \int_0^{1-s} \mathbf{x}_i^k dt ds = |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 \int_0^{1-s} \mathbf{p}^k + s\mathbf{u}^k + \mathrm{t}\mathbf{v}^k dt ds$$

$$= |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 (1-s)\mathbf{p}^k + (1-s)\mathrm{s}\mathbf{u}^k + \frac{1}{2}(1-s)^2 \mathbf{v}^k ds$$

$$= |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \left(\frac{1}{2}\mathbf{p}^k + \frac{1}{6}\mathbf{u}^k + \frac{1}{6}\mathbf{v}^k \right)$$

$$= |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \left(\frac{1}{2}\mathbf{p}^k + \frac{1}{6}\left(\mathbf{q}^k - \mathbf{p}^k\right) + \frac{1}{6}(\mathbf{r}^k - \mathbf{p}^k) \right)$$

$$= |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \frac{1}{6}(\mathbf{p}^k + \mathbf{q}^k + \mathbf{r}^k)$$

$$= \left(\frac{1}{2} |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \right) \left(\frac{1}{3}(\mathbf{p}^k + \mathbf{q}^k + \mathbf{r}^k) \right)$$

$$= A^k \mathbf{m}_i.$$

Tässä $\mathbf{m}_i$ on kolmion k keskipiste ja A^k on sen pinta-ala. Nyt saadaan odotusarvolle kaava

$$\mathrm{E}[\mathbf{x}_i] = \frac{1}{A^M}\sum_k A^k \mathbf{m}_i^k,$$

joka on edelleen koko kolmioverkon pinta-alajakauman keskipisteen koordinaatti suunnassa $\mathbf{x}_i$ eli

$$\mathrm{E}[\mathbf{x}_i] = \mathbf{m}_i^M.$$

Odotusarvon $\mathrm{E}[\mathbf{x}_j]$ laskeminen tapahtuu vastaavalla tavalla. Laskennan tuloksena saadaan

$$\mathrm{E}[\mathbf{x}_j] = \frac{1}{\mathrm{A}^M}\sum_k \mathrm{A}^k \mathbf{m}_j^k,$$

ja edelleen

$$\mathrm{E}[\mathbf{x}_j] = \mathbf{m}_j^M.$$

Kovarianssimatriisin määrittämiseksi täytyy vielä laskea $\mathrm{E}[\mathbf{x}_i\mathbf{x}_j]$. Nyt

$$\mathrm{E}[\mathbf{x}_i\mathbf{x}_j] = \frac{1}{A^M}\int_M \mathbf{x}_i \mathbf{x}_j dA$$

$$= \frac{1}{A^M}\sum_k \int_\Omega \mathbf{x}_i^k \mathbf{x}_j^k dA,$$

josta saadaan

$$\int_{\Omega} \mathbf{x}_i^k \mathbf{x}_j^k dA = |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 \int_0^{1-s} \mathbf{x}_i^k \mathbf{x}_j^k dt ds$$
$$= |\,\mathbf{u}^k \mathrm{x} \mathbf{v}^k\,| \int_0^1 \int_0^{1-s} f(s,t) dt ds\,.$$

Tässä $f(s,t)=\mathbf{x}_i^k \mathbf{x}_j^k$ ja edelleen $f(s,t)=(\mathbf{p}_i^k + s\mathbf{u}_i^k + t\mathbf{v}_i^k)(\mathbf{p}_j^k + s\mathbf{u}_j^k + t\mathbf{v}_j^k)$. Kun $f(s,t)$ kerrotaan auki, niin saadaan:

$$\mathbf{p}_i^k \mathbf{p}_j^k + s(\mathbf{u}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{u}_j^k) + t(\mathbf{v}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{v}_j^k) +$$
$$st(\mathbf{u}_i^k \mathbf{v}_j^k + \mathbf{v}_i^k \mathbf{u}_j^k) + s^2(\mathbf{u}_i^k \mathbf{u}_j^k) + t^2(\mathbf{v}_i^k \mathbf{v}_j^k)\,.$$

Kun tämä integroidaan t:n suhteen, niin saadaan:

$$\int_0^{1-s} f(s,t) dt = (1-s)\mathbf{p}_i^k \mathbf{p}_j^k + (1-s)s(\mathbf{u}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{u}_j^k) + \frac{1}{2}(1-s)^2(\mathbf{v}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{v}_j^k)$$
$$+ \frac{s}{2}(1-s)^2(\mathbf{u}_i^k \mathbf{v}_j^k + \mathbf{v}_i^k \mathbf{u}_j^k) + (1-s)\mathrm{s}^2(\mathbf{u}_i^k \mathbf{u}_j^k) + \frac{1}{2}s^3(\mathbf{v}_i^k \mathbf{v}_j^k)\,,$$

ja edelleen integroidaan s:n suhteen:

$$\int_0^1 \int_0^{1-s} f(s,t) dt ds = \frac{1}{2}\mathbf{p}_i^k \mathbf{p}_j^k + \frac{1}{6}(\mathbf{u}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{u}_j^k) + \frac{1}{6}(\mathbf{v}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{v}_j^k)$$
$$+ \frac{1}{24}(\mathbf{u}_i^k \mathbf{v}_j^k + \mathbf{v}_i^k \mathbf{u}_j^k) + \frac{1}{12}(\mathbf{u}_i^k \mathbf{u}_j^k) + \frac{1}{12}(\mathbf{v}_i^k \mathbf{v}_j^k)\,.$$

Sijoittamalla $\mathbf{u}^k=\mathbf{q}^k-\mathbf{r}^k$ ja $\mathbf{v}^k=\mathbf{r}^k-\mathbf{r}^k$ saadaan:

$$\mathbf{u}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{u}_j^k = \mathbf{q}_i^k \mathbf{p}_j^k - 2\mathbf{p}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{q}_j^k$$
$$\mathbf{v}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{v}_j^k = \mathbf{r}_i^k \mathbf{p}_j^k - 2\mathbf{p}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{r}_j^k$$
$$\mathbf{u}_i^k \mathbf{v}_j^k = \mathbf{q}_i^k \mathbf{r}_j^k - \mathbf{p}_i^k \mathbf{r}_j^k - \mathbf{q}_i^k \mathbf{p}_j^k + \mathbf{p}_i^k \mathbf{p}_j^k$$
$$\mathbf{v}_i^k \mathbf{u}_j^k = \mathbf{r}_i^k \mathbf{q}_j^k - \mathbf{r}_i^k \mathbf{p}_j^k - \mathbf{p}_i^k \mathbf{q}_j^k + \mathbf{p}_i^k \mathbf{p}_j^k$$
$$\mathbf{u}_i^k \mathbf{u}_j^k = \mathbf{q}_i^k \mathbf{q}_j^k - \mathbf{q}_i^k \mathbf{p}_j^k - \mathbf{p}_i^k \mathbf{q}_j^k + \mathbf{p}_i^k \mathbf{p}_j^k$$
$$\mathbf{v}_i^k \mathbf{v}_j^k = \mathbf{r}_i^k \mathbf{r}_j^k - \mathbf{r}_i^k \mathbf{p}_j^k - \mathbf{p}_i^k \mathbf{r}_j^k + \mathbf{p}_i^k \mathbf{p}_j^k\,.$$

Kun integraali

$$\int_0^1 \int_0^{1-s} f(s,t) dt ds$$

lasketaan sijoitusten jälkeen, niin odotusarvoksi $\mathrm{E}[\mathbf{x}_i\mathbf{x}_j]$ saadaan:

$$E[\mathbf{x}_i\mathbf{x}j]=\frac{1}{A^M}\sum_k\frac{A^k}{12}[9\mathbf{m}_i^k\mathbf{m}_j^k+\mathbf{p}_i^k\mathbf{p}_j^k+\mathbf{q}_i^k\mathbf{q}_j^k+\mathbf{r}_i^k\mathbf{r}_j^k].$$

Nyt voidaan laskea kovarianssimatriisi kolmioverkon pinnan kolmioiden mukaan määritetylle rajoitustilavuudelle seuraavasti:

$$\mathbf{C}_{ij}=E[\mathbf{x}_i\mathbf{x}_j]-E[\mathbf{x}_i]E[\mathbf{x}_j]=$$
$$\frac{1}{A^M}\sum_k\frac{A^k}{12}[9\mathbf{m}_i^k\mathbf{m}_j^k+\mathbf{p}_i^k\mathbf{p}_j^k+\mathbf{q}_i^k\mathbf{q}_j^k+\mathbf{r}_i^k\mathbf{r}_j^k]-\mathbf{m}_i^M\mathbf{m}_j^M.$$

Kun kovarianssimatriisille on määritetty ominaisvektorit, voidaan rajoitustilavuuden ulottuvuudet määrittää samalla tavalla, kun kohdassa 6.1.1. Tämän jälkeen voidaan muodostaa yksittäinen rajoitustilavuus, joka voidaan kirjoittaa c++ struktuurina koodikatkelman 3 mukaisesti.

```
struct triangle
{
  double vertex1[3], vertex2[3], vertex3[3];
};
struct OrientedBox
{
  public:
  double orientations[3][3]; //Kovarianssimatriisin ominaisvektorit
  double center[3];          //Rajoitustilavuuden keskipiste
  double dimensions[3];      //Rajoitustilavuuden koko
  triangle *trilist;         //Rajoitustilavuuden sisältämät kolmiot
};
```

Koodikatkelma 3: Yksittäinen kohteen mukaan suunnattu rajoitustilavuus.

Rajoitustilavuuden sovittaminen kohteelle voidaan suorittaa lineaarisessa ajassa O(n), kun sovittamisessa käytetään kolmioverkkoa. Sovittamista varten voidaan muodostaa koodikatkelman 4 mukainen apustruktuuri acc, johon lasketaan yhteen kaikkien kohteessa olevien kolmioiden pinta-alat, keskipisteet ja kovarianssit. Tästä kertymästä saadaan kolmioverkon keskipiste jakamalla struktuuriin kertynyt keskipiste acc.m kolmioverkon pinta-alalla acc.A. Kovarianssimatriisin $\mathbf{C}$ alkio $\mathbf{C}_{ij}$ saadaan kertyneestä kovarianssista acc.C seuraavasti

$$\mathbf{C}_{ij} = \text{acc.}C_{ij} - \text{acc.m[i]} * \text{acc.m[j]} * \frac{1}{\text{acc.A}}.$$

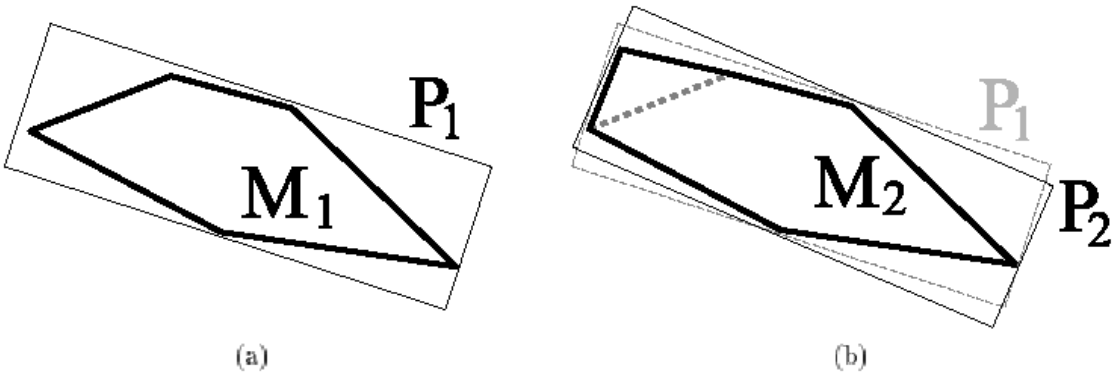
```
struct acc
{
  double A; //Pinta-alan kertymä
  double m[3]; //Keskipisteen kertymä
  double C[3][3]; //Kovarianssin kertymä
};
```

Koodikatkelma 4: Apustruktuuri rajoitustilavuuden laskemiseksi.

Jos halutaan määrittää kohteelle paras mahdollinen rajoitustilavuus on käytettävä kohteen kulmapisteiden konveksia peitettä. Tämän laskeminen on astetta vaativampi tehtävä, kun edellä esitetyn apustruktuurin käyttö. Tällöin rajoitustilavuuden muodostamisen aikavaatimukseksi tulee O(*n*log*n*). Konveksin peitteen käytöstä kertovat mm. M. de Berg *et al*. [5] ja Barber *et al*. [4].

6.2. Kovarianssimatriisin avulla sovitetun rajoitustilavuuden sopivuus

Kovarianssimenetelmällä määritetyn rajoitustilavuuden suunta ja ulottuvuudet muuttuvat herkästi kohteen geometristen ominaisuuksien mukaan. Tämän vuoksi menetelmä ei välttämättä tuota optimaalista rajoitustilavuutta. Tarkastellaan asiaa kuvan 11 avulla.



Kuva 11: Kulmapisteen siirtämisen vaikutus rajoitustilavuuteen.

Kuvan 11 (a)-kohta esittää kohteen M_1 kovarianssimenetelmällä muodostettua rajoitustilavuutta P_1. Siirretään nyt kohteen M_1 yhtä kulmapistettä siten, että kohteen M_1 muoto muuttuu (b)-kohdan mukaiseksi.

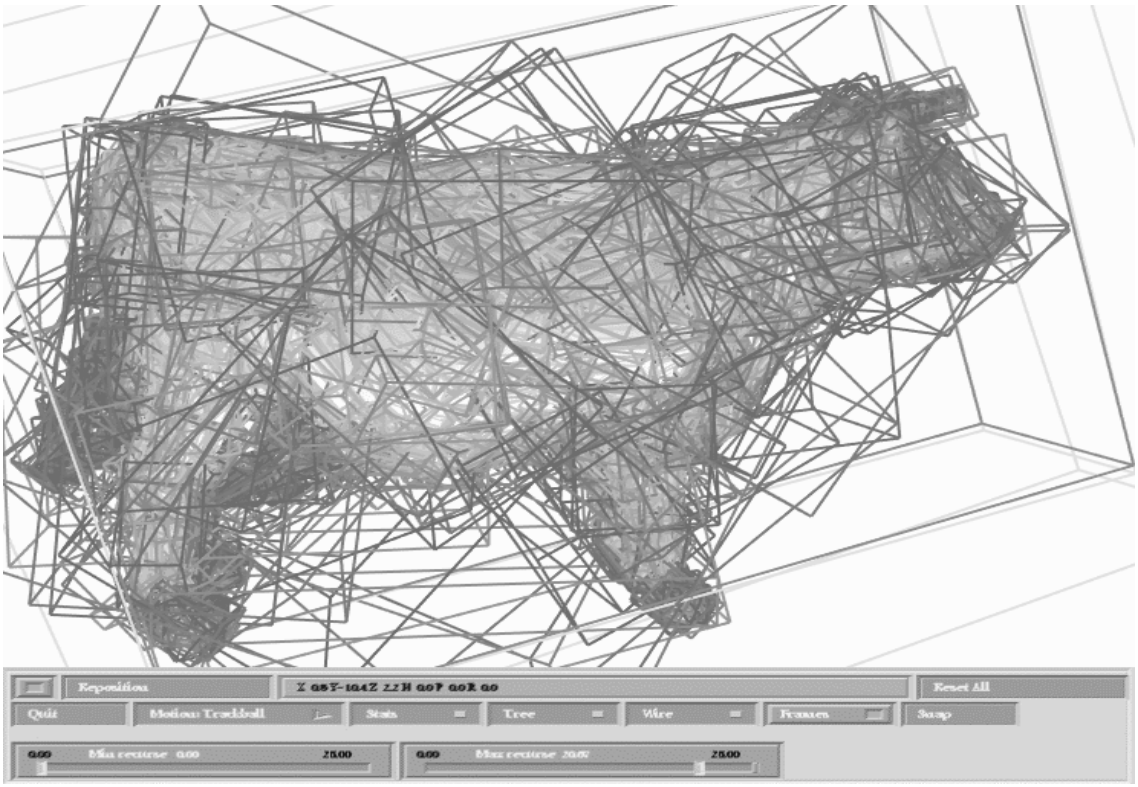
Tällöin kohde M_1 muuttuu kohteeksi M_2. Rajoitustilavuus P_1 pystyy edelleen sulkemaan sisäänsä kohteen M_2, mutta kohteelle M_2 muodostettu rajoitustilavuus P_2 ei ole enää sama kuin P_1. Näin ollen rajoitustilavuuden muoto on kohteen geometrian jakauman funktio. Kovarianssimenetelmän tuottama rajoitustilavuus ei siis välttämättä ole paras mahdollinen.

6.3. Rajoitustilavuushierarkian muodostaminen

Animaatiossa mukana olevien kohteiden geometria ladataan yleensä tiedostosta, joka on tehty jollain 3D-mallinnusohjelmalla. Rajoitustilavuushierarkian muodostamiseksi kohteen kolmiot ladataan listaan L. Jos kohde muodostuu monikulmiosta, joissa on enemmän kuin kolme särmää, suoritetaan näille monikulmioille kolmiointi [11]. Yleisin tapa rajoitustilavuushierarkian toteuttamiseksi on binaaripuu ja sen rakentaminen siten, että syötelistaa L jaetaan kahteen osaan niin kauan, kuin lista on jaettavissa. Tätä menetelmää käytetään myös pikalajittelussa.

Päätason rajoitustilavuudelle R lasketaan aluksi kolmiolistan L avulla keskipiste **m** ja suuntavektorit $\mathbf{v}^1$, $\mathbf{v}^2$ ja $\mathbf{v}^3$. Oletetaan, että vektori $\mathbf{v}^1$ on rajoitustilavuuden R se suuntavektori, jonka suunnassa kohteen K geometrian jakaumalla on suurin varianssi. Tämän jälkeen kolmiolistaa L jaetaan rekursiivisesti osalistoihin L_1 ja L_2. Listan L jakaminen osalistoihin L_1 ja L_2 tapahtuu siten, että projisoidaan listan L kolmioiden keskipisteet vektorille $\mathbf{v}^1$ ja katsotaan, kummalle puolelle rajoitustilavuuden R keskipisteen **m** projektiota vektorilla $\mathbf{v}^1$ ne sijoittuvat. Rajoitustilavuudet R_1 ja R_2 määritetään nyt osalistojen L_1 ja L_2 mukaan vastaavasti kuin rajoitustilavuus R. Kun listaa ei enää voida jakaa, niin lehtitason rajoitustilavuuksissa on kussakin vain yksi kolmio.

Jos oletetaan, että kohteen geometria jakaantuu tasaisesti, niin lopullisen rajoitustilavuushierarkian syvyydeksi saadaan $O(\log n)$. Yksittäisen rajoitustilavuuden muodostamiseen kuluu aika $O(k)$, missä k on rajoitustilavuuden sisältämien kolmioiden lukumäärä. Kaikkien rajoitustilavuuksien muodostamiseen kuluu nyt aika $O(n)$ joten kokonaisaika hierarkian muodostamiseksi on $O(n \log n)$. Kuva 12 havainnollistaa kohteen geometrian alijakoa rajoitustilavuushierarkiaan.



Kuva 12: Geometrian alijako rajoitustilavuushierarkiaan.

Edellä esitetyn menetelmän lisäksi rajoitustilavuushierarkia voidaan muodostaa myös siten, että lähdetään rakentamaan binaaripuuta yksittäisten kolmioiden rajoitustilavuuksista ja lomittamalla näitä suurempiin tilavuuksiin kunnes saavutetaan juuritaso. Tämä tapa on kuitenkin vaikeampi toteuttaa, eikä sen soveltamista juuri esiinny törmäystarkasteluja käsittelevässä kirjallisuudessa.

Geometrian alijakoon perustuvassa törmäystarkastelussa tutkitaan kahden rajoitustilavuushierarkian rajoitustilavuuksien erillisyyttä. Jos rajoitustilavuudet ovat erillisiä, voidaan testaaminen lopettaa heti. Kahden kohteen välisessä törmäyksessä edetään niiden rajoitustilavuushierarkioiden lehtitasoilla olevien kolmioiden leikkauksien tarkasteluun. Seuraavaksi tarkastellaan menetelmää, jonka avulla rajoitustilavuuksien erillisyyden voi todeta.

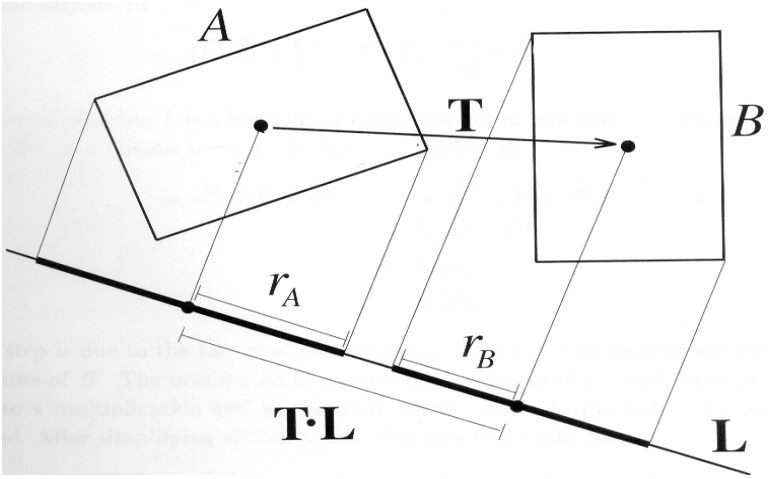
6.4. Rajoitustilavuuksien erillisyyden testaaminen

Kohdassa 3.2.5 todettiin, että kohteen mukaan suunnatun rajoitustilavuuden kuvaamiseksi tarvitaan keskipisteen **c** lisäksi suuntavektorit $\mathbf{v}^1$, $\mathbf{v}^2$ ja $\mathbf{v}^3$.

Edelleen tarvitaan vielä rajoitustilavuuden särmien pituuksien puolikkaat a_1, a_2 ja a_3. Nyt voidaan määritellä rajoitustilavuudelle A säde **r** seuraavasti:

$$\mathbf{r} = a_1\mathbf{v}^1 + a_2\mathbf{v}^2 + a_3\mathbf{v}^3 .$$

Rajoitustilavuuksien A ja B erillisyys voidaan nyt todeta projisoimalla niiden keskipisteiden $\mathbf{c}_A$ ja $\mathbf{c}_B$ erotus **T** sekä säteet $\mathbf{r}_A$ ja $\mathbf{r}_B$ erottavalle akselille **L**. Jos tässä projektiossa $|\mathbf{T}\bullet\mathbf{L}| > |\mathbf{r}_A\bullet\mathbf{L}| + |\mathbf{r}_B\bullet\mathbf{L}|$, niin rajoitustilavuudet A ja B ovat erillisiä. Kuvassa 13 kuvataan erillisiä rajoitustilavuuksia.



Kuva 13: Erilliset rajoitustilavuudet A ja B.

Erottava akseli L voi olla kumman tahansa rajoitustilavuuden A tai B jonkin suuntavektorin $\mathbf{v}_A^1, \mathbf{v}_A^2, \mathbf{v}_A^3, \mathbf{v}_B^1, \mathbf{v}_B^2$ tai $\mathbf{v}_B^3$ suuntainen. Lisäksi erottavan akselin suunta voi määräytyä jonkin ristitulon $\mathbf{v}_A^i \times \mathbf{v}_B^i$ mukaan. Tässä $i \in [1,3]$. Tällöin erottava akseli voi määräytyä yhteensä viidellätoista eri tavalla.

Rajoitustilavuuksien erillisyyttä tutkittaessa toimitaan yleensä siten, että rajoitustilavuus B siirretään rajoitustilavuuden A paikalliseen koordinaatistoon. Tällöin on luonnollista ilmoittaa rajoitustilavuuksien suuntavektorit matriisimuodossa siten, että vektorista $\mathbf{v}_A^1$ tulee matriisin **A** ensimmäinen, vektorista $\mathbf{v}_A^2$ toinen ja vektorista $\mathbf{v}_A^3$ kolmas sarake. Tällöin rajoitustilavuuden B suuntavektorit voidaan ilmaista rajoitustilavuuden A paikallisessa koordinaatistossa muodossa **B=AC,** missä $\mathbf{C}=\mathbf{A}^T\mathbf{B}$. Tässä $\mathbf{A}^T$ on matriisin **A** transpoosi. Olkoon c_{ij} matriisin **C** komponentti ij. Nyt rajoitustilavuuden B suuntavektori $\mathbf{v}_B^i$ voidaan kirjoittaa rajoitustilavuuden A suuntavektorien avulla seuraavasti

$$\mathbf{v}_B^i = c_{1i}\mathbf{v}_A^1 + c_{2i}\mathbf{v}_A^2 + c_{3i}\mathbf{v}_A^3.$$

Tässä oletetaan, että matriisin indeksointi aloitetaan kohdasta 1 eikä 0. Rajoitustilavuuden B keskipiste $\mathbf{c}_B$ rajoitustilavuuden A koordinaatistossa voidaan nyt kirjottaa muodossa $\mathbf{c}_B^A = \mathbf{A}^T\mathbf{c}_B$. Rajoitustilavuuden A suuntavektorit voidaan vastaavasti ilmoittaa rajoitustilavuuden B suuntavektorien avulla muodossa

$$\mathbf{v}_A^i = c_{i1}\mathbf{v}_B^1 + c_{i2}\mathbf{v}_B^2 + c_{i3}\mathbf{v}_B^3.$$

Nyt voidaan kaikille viidelletoista testille muodostaa taulukko, josta ilmenee yhteydet muuttujille **L**, **T**, $\mathbf{r}_A \bullet \mathbf{L}$ ja $\mathbf{r}_B \bullet \mathbf{L}$. Tässä oletetaan, että molemmat rajoitustilavuudet ovat samassa koordinaatistossa.

$\mathbf{L}$	$\mathbf{r}_A \bullet \mathbf{L}$	$\mathbf{r}_B \bullet \mathbf{L}$	$\mathbf{T}$
$\mathbf{v}_A^1$	a_1	$b_1 \lvert c_{11} \rvert + b_2 \lvert c_{12} \rvert + b_3 \lvert c_{13} \rvert$	$\lvert \mathbf{v}_A^1 \bullet \mathbf{L} \rvert = A1$
$\mathbf{v}_A^2$	a_2	$b_1 \lvert c_{21} \rvert + b_2 \lvert c_{22} \rvert + b_3 \lvert c_{23} \rvert$	$\lvert \mathbf{v}_A^2 \bullet \mathbf{L} \rvert = A2$
$\mathbf{v}_A^3$	a_3	$b_1 \lvert c_{31} \rvert + b_2 \lvert c_{32} \rvert + b_3 \lvert c_{33} \rvert$	$\lvert \mathbf{v}_A^3 \bullet \mathbf{L} \rvert = A3$
$\mathbf{v}_B^1$	$a_1 \lvert c_{11} \rvert + a_2 \lvert c_{21} \rvert + a_3 \lvert c_{31} \rvert$	b_1	$\lvert \mathbf{v}_B^1 \bullet \mathbf{L} \rvert$
$\mathbf{v}_B^2$	$a_1 \lvert c_{12} \rvert + a_2 \lvert c_{22} \rvert + a_3 \lvert c_{32} \rvert$	b_2	$\lvert \mathbf{v}_B^2 \bullet \mathbf{L} \rvert$
$\mathbf{v}_B^3$	$a_1 \lvert c_{13} \rvert + a_2 \lvert c_{23} \rvert + a_3 \lvert c_{33} \rvert$	b_3	$\lvert \mathbf{v}_B^3 \bullet \mathbf{L} \rvert$
$\mathbf{v}_A^1 \times \mathbf{v}_B^1$	$a_2 \lvert c_{31} \rvert + a_3 \lvert c_{21} \rvert$	$b_2 \lvert c_{13} \rvert + b_3 \lvert c_{12} \rvert$	$\lvert c_{21}A3 - c_{31}A2 \rvert$
$\mathbf{v}_A^1 \times \mathbf{v}_B^2$	$a_2 \lvert c_{32} \rvert + a_3 \lvert c_{22} \rvert$	$b_1 \lvert c_{13} \rvert + b_3 \lvert c_{11} \rvert$	$\lvert c_{22}A3 - c_{31}A2 \rvert$
$\mathbf{v}_A^1 \times \mathbf{v}_B^3$	$a_2 \lvert c_{33} \rvert + a_3 \lvert c_{23} \rvert$	$b_1 \lvert c_{12} \rvert + b_2 \lvert c_{11} \rvert$	$\lvert c_{23}A3 - c_{33}A2 \rvert$
$\mathbf{v}_A^2 \times \mathbf{v}_B^1$	$a_1 \lvert c_{31} \rvert + a_3 \lvert c_{11} \rvert$	$b_2 \lvert c_{23} \rvert + b_3 \lvert c_{22} \rvert$	$\lvert c_{31}A1 - c_{11}A3 \rvert$
$\mathbf{v}_A^2 \times \mathbf{v}_B^2$	$a_1 \lvert c_{32} \rvert + a_3 \lvert c_{12} \rvert$	$b_1 \lvert c_{23} \rvert + b_3 \lvert c_{21} \rvert$	$\lvert c_{32}A1 - c_{12}A3 \rvert$
$\mathbf{v}_A^2 \times \mathbf{v}_B^3$	$a_1 \lvert c_{33} \rvert + a_3 \lvert c_{13} \rvert$	$b_1 \lvert c_{22} \rvert + b_2 \lvert c_{21} \rvert$	$\lvert c_{33}A1 - c_{13}A3 \rvert$
$\mathbf{v}_A^3 \times \mathbf{v}_B^1$	$a_1 \lvert c_{21} \rvert + a_2 \lvert c_{11} \rvert$	$b_2 \lvert c_{33} \rvert + b_3 \lvert c_{32} \rvert$	$\lvert c_{11}A2 - c_{21}A1 \rvert$
$\mathbf{v}_A^3 \times \mathbf{v}_B^2$	$a_1 \lvert c_{22} \rvert + a_2 \lvert c_{12} \rvert$	$b_1 \lvert c_{33} \rvert + b_3 \lvert c_{31} \rvert$	$\lvert c_{12}A2 - c_{22}A1 \rvert$
$\mathbf{v}_A^3 \times \mathbf{v}_B^3$	$a_1 \lvert c_{23} \rvert + a_2 \lvert c_{13} \rvert$	$b_1 \lvert c_{32} \rvert + b_2 \lvert c_{31} \rvert$	$\lvert c_{13}A2 - c_{23}A1 \rvert$

Taulukko 1: Rajoitustilavuuksien säteiden ja erottavan akselin riippuvuus.

Taulukossa 1 on sarakkeessa **T** kolmella ensimmäisellä rivillä käytetty yhtä suuruus merkkiä loppujen rivien lyhentämistä varten.

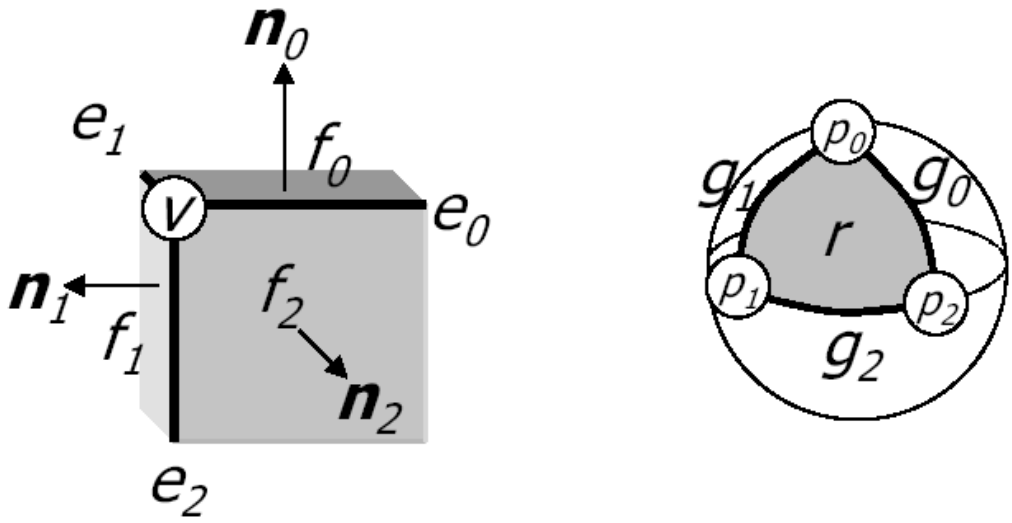
Kahden rajoitustilavuuden A ja B erillisyyden toteamiseksi on löydettävä yksi erottava akseli **L** siten, että $|\mathbf{T} \bullet \mathbf{L}| > |\mathbf{r}_A \bullet \mathbf{L}| + |\mathbf{r}_B \bullet \mathbf{L}|$ taulukon 1 mukaisesti. Jos tällaista akselia ei löydy, niin rajoitustilavuudet A ja B menevät päällekkäin. Pahimmassa tapauksessa joudutaan tutkimaan kaikki vaihtoehdot, mutta keskimäärin testataan vain 7,5 akseliehdokasta. Testin tehostamiseksi se lopetetaan heti, kun ensimmäinen erottava akseli löydetään.

Tämä on huomattava parannus naiviin menetelmään, jossa joudutaan tutkimaan 144 erilaista rajoitustilavuuden särmän ja tahkon mahdollista leikkausta. 144 testiä naivissa menetelmässä tulee siitä, että rajoitustilavuuksia on kaksi ja kummassakin on kuusi tahkoa ja kaksitoista särmää. Eli 2x6x12=144.

Seuraavaksi tutkitaan miten ehdokkaat erottavaksi akseliksi on valittu. Tämän selvittämiseksi tutustutaan ensin Gaussin kuvaukseen ja Minkowskin erotukseen.

6.4.1. Gaussin kuvaus

Monitahokkaan Gaussin kuvauksessa monitahokkaan kulmapisteet **v**, särmät **e** ja tahkot **f** kuvautuvat kolmiulotteisen pallon pinnalle. Tässä tapauksessa rajoitutaan suorakulmaisen särmiön Gaussin kuvauksen muodostamisen. Suorakulmaisen särmiön tahko $\mathbf{f}_0$ kuvautuu yhdeksi pisteeksi $\mathbf{p}_0$, joka ilmaisee kyseisen tahkon suuntavektorin. Kahden tahkon välissä oleva särmä $\mathbf{e}_0$ kuvautuu pallon pinnalla olevaksi kaareksi $\mathbf{g}_0$. Tämä kaari ilmaisee ne suunnat, jotka ovat kohtisuorassa särmän $\mathbf{e}_0$ kanssa. Kulmapiste **v** kuvautuu konveksiksi alueeksi r, joka muodostuu kolmen risteävän kaaren $\mathbf{e}_1$, $\mathbf{e}_2$ ja $\mathbf{e}_3$ rajoittamasta alueesta. Gaussin kuvausta havainnollistaa kuva 14.



Kuva 14: Suorakulmaisen särmiön Gauss-kuvaus.

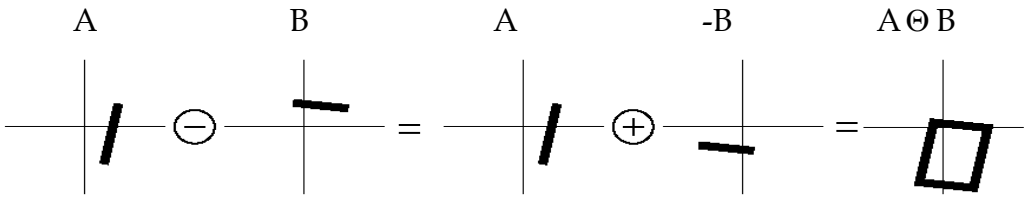
Gaussin kuvauksessa saman pallon kehälle voidaan kuvata useampia monitahokkaita. Erottavan akselin määräytymisen tarkastelemisessa kuvataan rajoitustilavuudet A ja B pallon kehälle. Tämän kuvauksen tuloksena voidaan muodostaa rajoitustilavuuksille A ja B Minkowskin erotus $A \Theta B$, jonka jälkeen voidaan määrittää yhteys erottavalle akselille ja rajoitustilavuuksille.

6.4.2. Minkowskin erotus

Kahden rajoitustilavuuden A ja B välinen Minkowskin erotus $A \Theta B$ on sellainen pistejoukko P, missä P:n pisteet ovat muotoa **a**-**b**. Tässä $\mathbf{a} \in A$ ja $\mathbf{b} \in B$. Tämä voidaan ilmoittaa myös muodossa

$$P=\{ \mathbf{a}\text{-}\mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}.$$

Minkowskin erotus voidaan tulkita geometrisesti siten, että rajoitustilavuuden B pisteistä muodostetaan uusi suorakulmainen särmiö vastavektoreiden **–b** avulla. Tämän jälkeen kuljetetaan rajoitustilavuutta A vektoreiden **–b** muodostamaa uraa pitkin siten, että A säilyttää suuntansa ja sen keskipiste $\mathbf{c}_A$ on uralla. Kuvassa 15 viivasegmentti B kuvataan ensin segmentiksi –B. Tämän jälkeen viivasegmenttiä A kuljetetaan viivasegmenttiä –B pitkin siten, että A:n keskipiste on viivasegmentillä –B. Minkowskin erotusta $A \Theta B$ kuvaa nyt muodostunut pinta-ala. Kolmiulotteisessa tapauksessa muodostuu luonnollisesti tilavuus.



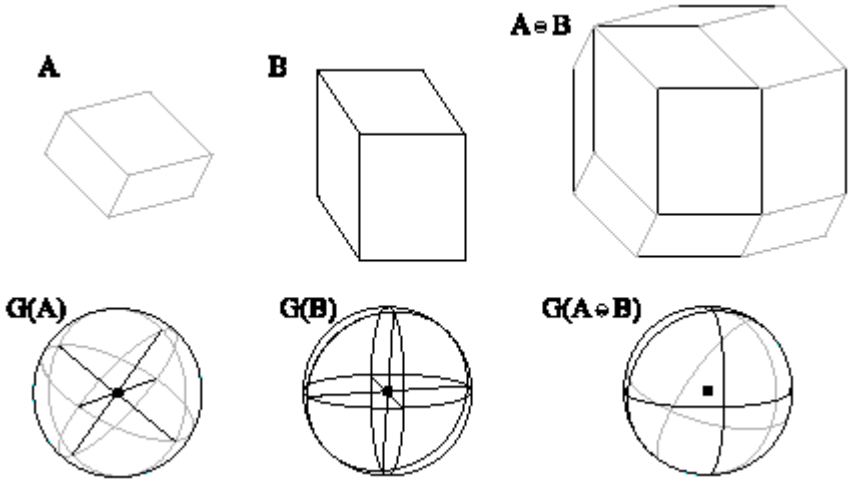
Kuva 15: Minkowskin erotuksen geometrinen tulkinta.

Minkowskin erotuksella on keskeinen osa erilaisissa törmäystarkastelumenetelmissä. Tämä johtuu siitä, että pistejoukko P sisältää tarkasteluavaruuden origon **0** silloin ja vain silloin, kun monitahokkaat A ja B koskettavat toisiaan. Tämä pätee mihin tahansa monitahokkaaseen ja erityisesti tässä tarkasteltaville suorakulmaisille särmiöille. Jos suorakulmaiset särmiöt A ja B koskettavat toisiaan, niin silloin on olemassa pisteet **a** ja **b** siten, että niiden erotus **a-b=0**. Nyt **a=b**, jolloin määritelmän $P=\{ \mathbf{a}-\mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}$ mukaan $\mathbf{b} \in A$, $\mathbf{a} \in B$.

Seuraavaksi tarkastellaan erottavan akselin määräytymistä Gaussin kuvauksen ja Minkowskin erotuksen avulla.

6.4.3. Erottavan akselin määräytyminen

Tarkastellaan erottavan akselin määräytymistä kuvan 16 avulla



Kuva 16: Rajoitustilavuuksien A, B ja niiden Minkowskin erotuksen $A \Theta B$ Gaussin kuvaukset.

Kuvan 16 yläosassa on esitetty monitahokkaat A, B ja $A \Theta B$ sekä alaosassa niiden Gaussin kuvaukset. Aikaisemmasta muistetaan, että $A \Theta B$ muodostuu kun särmiötä A liikutetaan särmiön –B särmiä pitkin siten, että A:n keskipiste kulkee B:n särmällä ja A:n suunta avaruudessa pysyy muuttumattomana. –B on siis särmiö, jonka särmävektorit ovat $V=\{-\mathbf{b} \mid \mathbf{b} \in B\}$.

Kuvassa 16 rajoitustilavuuden A tahkot kuvautuvat kolmeksi keskenään kohtisuorassa olevaksi ympyräksi Gaussin kuvauksessa G(A). Vastaavasti kuvautuu myös rajoitustilavuus B. Rajoitustilavuuksien A ja B Minkowskin erotuksen $A \Theta B$ Gaussin kuvauksessa $G(A \Theta B)$ syntyy kuusi ympyrää kuvauspallon pinnalle. Näistä kolme on keskenään kohtisuorassa toisiaan vastaan, ja ne kuuluvat rajoitustilavuudelle A. Toiset kolme ovat myös keskenään kohtisuorassa toisiaan vastaan, ja ne kuuluvat rajoitustilavuudelle B. Nämä on piirretty kuvaan 16 siten, että A:sta tulevat ympyrät on piirretty harmaalla värillä ja B:stä tulevat ympyrät mustalla värillä. Jos oletetaan, että ainoastaan kaksi ympyrää voi leikata toisensa yhdessä pisteessä, niin kuudesta ympyrästä voidaan nyt valita pareittain $\binom{6}{2}$=15 erilaista ympyräparia. Jokainen ympyräpari leikkaa toisensa täsmälleen kahdessa pisteessä, jotka ovat pallon vastakkaisilla puolilla. Näin ollen leikkauspisteitä on nyt 30. Koska pisteet ovat pallon vastakkaisilla puolilla, ovat pisteille kuvautuvan Minkowskin erotuksen $A \Theta B$ tahkojen normaalit näissä kohdissa vastakkaissuuntaiset. Näin ollen $A \Theta B$ rajoittuu kahden yhdensuuntaisen tason väliin. Koska leikkauspisteitä on kaikkiaan 30, rajoittuu $A \Theta B$ yhteensä 30 keskenään pareittain yhdensuuntaisen tason rajoittaman avaruuden osan sisään. Tämä tarkoittaa

sitä, että rajoitustilavuuksien A ja B Minkowskin erotuksessa $A \Theta B$ on 30 tahkoa.

Gaussin kuvauksen kaarien ja monitahokkaan särmien välillä on yksi yhteen vastaavuus. Koska jokainen kuvauksen $G(A \Theta B)$ kuudesta kaaresta voidaan jakaa kahteen osaan viidellä muulla kuvauksen kaarista, saadaan rajoitustilavuuksien A ja B Minkowskin erotukselle $A \Theta B$ särmien lukumääräksi 6x5x2=60. Nyt voidaan $A \Theta B$:n kulmapisteiden lukumäärä laskea Eulerin kaavan V-E+F=2 avulla. Tässä E on särmien lukumäärä, F on tahkojen lukumäärä ja V on kulmapisteiden lukumäärä. Nyt siis $A \Theta B$:n kulmapisteiden lukumäärä on 2+60-30=32. Jos kahta useampi ympyrä leikkaa toisensa yhdessä pisteessä, niin muodostuvassa Minkowskin erotuksessa $A \Theta B$ särmien, tahkojen ja kulmapisteiden määrä jää edellä esitettyä pienemmäksi.

Rajoitustilavuuksien A ja B Minkowskin erotus on nyt siis sellainen monitahokas, jossa on 30 pareittain yhdensuuntaista tahkoa, 60 särmää ja 32 kulmapistettä. Aikaisemmasta muistetaan, että rajoitustilavuudet A ja B ovat päällekkäisiä, jos niiden Minkowskin erotus $A \Theta B$ sisältää tarkasteluavaruuden origon $\mathbf{0}$. Jos rajoitustilavuudet A ja B ovat erillisiä, niin origon $\mathbf{0}$ etäisyys lähimmästä aluetta $A \Theta B$ rajoittavasta tasosta on rajoitustilavuuksien A ja B välinen etäisyys. Samalla etäisyyden suunta on kyseisen tason suunta.

Kuvassa 16 kuvauksen $G(A \Theta B)$ kolme harmaata ympyrää ilmaisee rajoitustilavuuden A tahkojen suunnat. Kolme mustaa ympyrää ilmaisee rajoitustilavuuden B tahkojen suunnat. Jäljelle jää yhdeksän mustan ja harmaan kaaren leikkausta, joissa pisteissä suunta on kohtisuorassa sekä rajoitustilavuuden A, että rajoitustilavuuden B jotakin särmää vastaan.

Edellä mainitun perusteella erottava akseli voi olla olemassa niissä suunnissa, joissa Minkowskin erotusta $A \Theta B$ rajoittavat tasot sijaitsevat. Erottavaa akselia ei ole olemassa, jos tarkasteluavaruuden origo $\mathbf{0}$ sijaitsee kaikkien Minkowskin erotusta $A \Theta B$ rajoittavien tasojen välissä.

Rajoitustilavuuksien A ja B erottavan akseli L voi siis olla olemassa niissä suunnissa, joissa rajoitustilavuuksien A ja B Minkowskin erotuksen $A \Theta B$ rajoittavat tasot sijaitsevat. Tasoja on yhteensä 30, mutta ne ovat pareittain yhdensuuntaisia, joten tarkastelussa riittää 15 tason suunnan tutkiminen.

7. Leikkauksista

Edellä on kuvattu menetelmää, jonka päätarkoituksena on karsia laskennasta sellaisia geometrisia primitiivejä, jotka eivät voi leikata toisiaan. Lehtitason rajoitustilavuuksien päällekkäisyyksien tapauksessa joudutaan kuitenkin lopulta laskemaan täsmälliset leikkauskohdat primitiivien välillä.

Seuraavaksi tarkastellaan tyypillisimpiä geometristen primitiivien välisiä leikkauksia. Näitä ovat kahden säteen välinen leikkaus, säteen ja tason välinen leikkaus, säteen ja kolmion välinen leikkaus ja kahden kolmion välinen leikkaus. Kahden kolmion välistä leikkausta käytetään myös kohteen mukaan suunnatun rajoitustilavuuden lehtitason kolmioiden leikkaustarkastelussa.

7.1. Kahden säteen välinen leikkaus

Kolmiulotteisessa avaruudessa säde annetaan yleensä parametrimuodossa $\mathbf{r}(t)=\mathbf{p}+t\mathbf{d}$, missä $\mathbf{p}$ on säteen alkupiste, $\mathbf{d}$ on säteen suuntavektori ja t on skalaari, jonka avulla säteen pituutta määritellään suunnassa $\mathbf{d}$. Tyypillisesti $\mathbf{d}$ on yksikkövektori, mutta näin ei välttämättä tarvitse olla. Lisäksi parametria t ei tarvitse rajoittaa, vaan sen avulla säteestä voidaan muodostaa sellainen, että sillä on alkupiste $\mathbf{p}$, mutta se jatkuu äärettömyyteen asti suunnassa $\mathbf{d}$.

Olkoon annettuna kaksi sädettä $\mathbf{r}_1$ ja $\mathbf{r}_2$ siten, että

$$\mathbf{r}_1(t_1)=\mathbf{p}_1+t_1\mathbf{d}_1$$
$$\mathbf{r}_2(t_2)=\mathbf{p}_2+t_2\mathbf{d}_2.$$

Nämä säteet voivat sijaita kolmiulotteisessa avaruudessa neljällä eri tavalla toisiinsa nähden. Säteet $\mathbf{r}_1(t_1)$ ja $\mathbf{r}_2(t_2)$ voivat leikata toisensa yhdessä pisteessä, tai niillä voi olla ääretön määrä yhteisiä pisteitä. Säteet voivat myös olla yhdensuuntaisia siten, että niillä ei ole yhteisiä pisteitä. Edellisten lisäksi säteet voivat sijaita kahdella samansuuntaisella tasolla siten, että tasot ovat eri etäisyyksillä origosta. Tällöin säteillä ei ole yhteisiä pisteitä.

Kahden säteen välisen leikkauksen löytäminen on käsitteellisesti yksinkertaista, mutta tietokoneiden pyöristystarkkuus saattaa aiheuttaa sen, että leikkausta ei välttämättä löydetä, vaikka sellainen olisi olemassa. Jos oletetaan, että säteiden leikkaus voidaan löytää käytössä olevalla pyöristystarkkuudella, niin leikkauskohta löydetään sijoittamalla seuraavasti

$$\mathbf{r}_1(t_1)= \mathbf{r}_2(t_2)$$
$$\mathbf{p}_1+t_1\mathbf{d}_1=\mathbf{p}_2+t_2\mathbf{d}_2$$
$$t_1\mathbf{d}_1=\mathbf{p}_2+t_2\mathbf{d}_2-\mathbf{p}_1$$
$$(t_1\mathbf{d}_1)\times\mathbf{d}_2=(\mathbf{p}_2+t_2\mathbf{d}_2-\mathbf{p}_1)\times\mathbf{d}_2$$
$$t_1(\mathbf{d}_1\times\mathbf{d}_2)=(t_2\mathbf{d}_2)\times\mathbf{d}_2+(\mathbf{p}_2-\mathbf{p}_1)\times\mathbf{d}_2$$
$$t_1(\mathbf{d}_1\times\mathbf{d}_2)=t_2(\mathbf{d}_2\times\mathbf{d}_2)+(\mathbf{p}_2-\mathbf{p}_1)\times\mathbf{d}_2$$
$$t_1(\mathbf{d}_1\times\mathbf{d}_2)=t_2\mathbf{0}+(\mathbf{p}_2-\mathbf{p}_1)\times\mathbf{d}_2$$
$$t_1(\mathbf{d}_1\times\mathbf{d}_2)=(\mathbf{p}_2-\mathbf{p}_1)\times\mathbf{d}_2$$
$$t_1(\mathbf{d}_1\times\mathbf{d}_2)\bullet(\mathbf{d}_1\times\mathbf{d}_2)=((\mathbf{p}_2-\mathbf{p}_1)\times\mathbf{d}_2)\bullet(\mathbf{d}_1\times\mathbf{d}_2)$$

$$t_1 = \frac{((\mathbf{p}_2 - \mathbf{p}_1)\mathrm{x}\mathbf{d}_2 \bullet (\mathbf{d}_1 \mathrm{x} \mathbf{d}_2)}{|\mathbf{d}_1 \mathrm{x} \mathbf{d}_2|^2}.$$

Parametrin t_2 arvo voidaan laskea vastaavasti, jolloin saadaan

$$t_2 = \frac{((\mathbf{p}_2 - \mathbf{p}_1)\mathrm{x}\mathbf{d}_1 \bullet (\mathbf{d}_1 \mathrm{x} \mathbf{d}_2)}{|\mathbf{d}_1 \mathrm{x} \mathbf{d}_2|^2}.$$

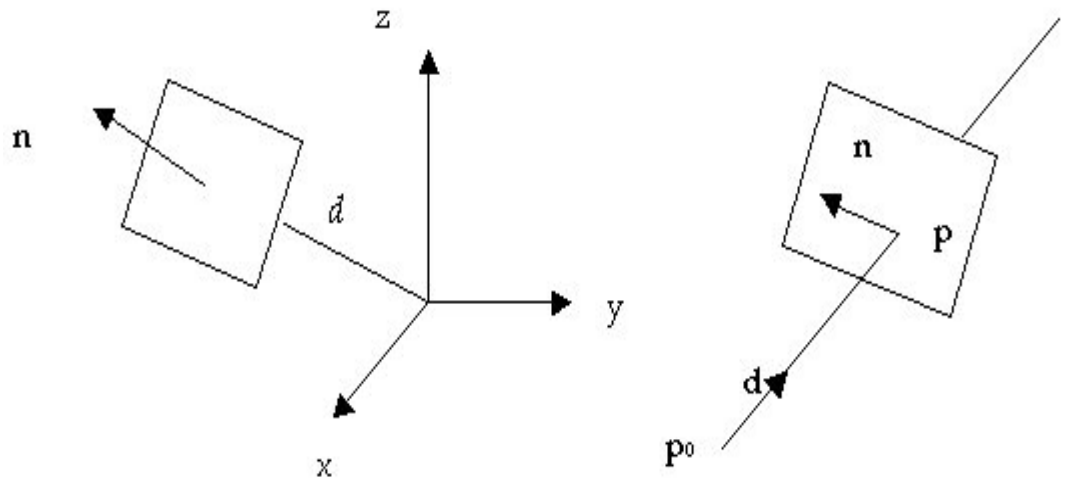
Edellä esitetyssä säteiden leikkauspisteen etsimisessä on oletettu, että säteet $\mathbf{r}_1$ ja $\mathbf{r}_2$ jatkuvat äärettömyyteen suuntavektorien osoittamissa suunnissa. Jos säteiden pituudet ovat rajoitettuja on tutkittava kuuluuko saatu leikkauspiste säteille, vai niiden jatkeille. Tämä voidaan tutkia siten, että sijoitetaan saadut arvot t_1 ja t_2 vastaavien säteiden $\mathbf{r}_1$ ja $\mathbf{r}_2$ yhtälöihin. Jos nyt $|\mathbf{p}_1 + t_1\mathbf{d}_1| \leq |\mathbf{r}_1|$ ja $|\mathbf{p}_2 + t_2\mathbf{d}_2| \leq |\mathbf{r}_2|$, niin säteet leikkaavat toisensa. Koska kyse on säteiden pituuksista, niin aina täytyy olla $t_1 > 0$ ja $t_2 > 0$.

7.2. Säteen ja tason leikkaus

Tason yhtälö annetaan yleensä muodossa Ax+By+Cz+D=0. Tietokonegrafiikassa on käytännöllisempää ilmoittaa taso P sen yksikkönormaalin **n** avulla. Yksikkönormaali **n** saadaan muodostamalla yksikkövektori

$$\mathbf{n} = \frac{\mathrm{A}\mathbf{i} + \mathrm{B}\mathbf{j} + \mathrm{C}\mathbf{k}}{|\mathrm{A}\mathbf{i} + \mathrm{B}\mathbf{j} + \mathrm{C}\mathbf{k}|}.$$

Tämän lisäksi tarvitaan tason yksikkönormaalin **n** suuntainen etäisyys d koordinaatiston origosta **O**. Jos A, B ja C muodostavat yksikkövektorin **n**, niin tason etäisyys origosta d=D. Nyt kaikille tason P pisteille **p** on voimassa $\mathbf{p} \bullet \mathbf{n} = d$. Kuvan 17 vasen puoli selventää asiaa.



Kuva 17: Tason etäisyys origosta sekä säteen ja tason leikkaus.

Olkoon nyt säde $R(t)=\mathbf{p}_0+\mathbf{d}t$ ja taso P muodossa $\mathbf{p}\bullet\mathbf{n}=d$. Sijoittamalla säde R tason yhtälöön saadaan:

$$
\begin{aligned}
(\mathbf{p}_0+\mathbf{d}t)\bullet\mathbf{n} &= d \\
\mathbf{p}_0\bullet\mathbf{n}+t\mathbf{d}\bullet\mathbf{n} &= d \\
t\mathbf{d}\bullet\mathbf{n} &= d-\mathbf{p}_0\bullet\mathbf{n} \\
t &= \frac{d-\mathbf{p}_0\bullet\mathbf{n}}{\mathbf{d}\bullet\mathbf{n}}.
\end{aligned}
$$

Säde R leikkaa siis tason P pisteessä **p**, mikäli on olemassa sellainen $t \in R$ siten, että $t>0$. Tason ja säteen leikkauspisteen määrittämisessä voidaan laskentaa nopeuttaa tutkimalla aluksi onko leikkaus ollenkaan mahdollinen. Jos pistetulo $\mathbf{d}\bullet\mathbf{n}=0$, niin taso ja säde ovat yhdensuuntaisia eikä leikkausta tapahdu. Jos taas $\mathbf{d}\bullet\mathbf{n}>0$, niin tason normaali **n** osoittaa poispäin säteestä. Tällöin voidaan leikkauksen laskeminen lopettaa, jos kysymyksessä on yksisuuntainen taso. Esimerkkinä on lasipinta, joka läpäisee valon vain toiselta puolelta.

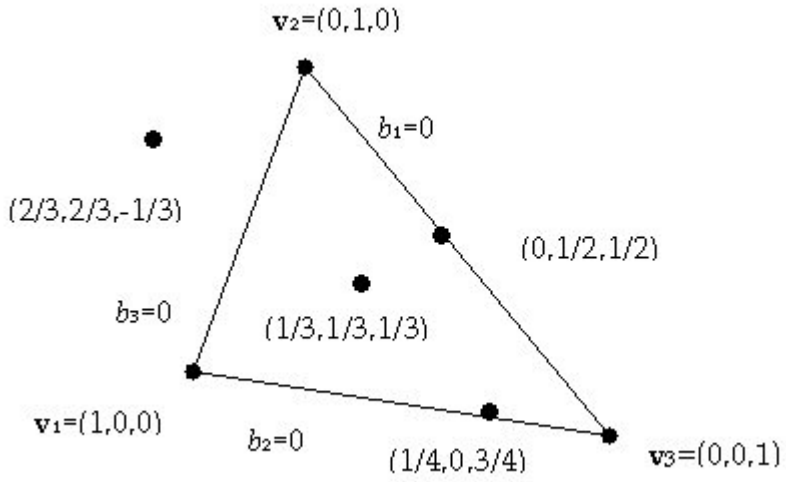
Parametrin t laskeminen kannattaa jättää suorittamatta, jos $\mathbf{d}\bullet\mathbf{n}$ ja $d-\mathbf{p}_0\bullet\mathbf{n}$ ovat eri merkkisiä. Tällöin t ei ole mielekäs, koska säteen ja tason leikkaus tapahtuisi säteen origon takana, eikä tämä vastaisi mitään todellista tilannetta. Tässä tapauksessa säästytään turhalta jakolaskun suorittamiselta, joka sinällään on suhteellisen paljon aikaa vievä operaatio. Parametrin t laskemisen jälkeen voidaan täsmällinen tason ja säteen leikkauskohta laskea sijoittamalla $\mathbf{p}=\mathbf{p}_0+\mathbf{d}t$.

7.3. Kolmion barysentriset koordinaatit

Kolmion pinnan tarkasteleminen kolmiulotteisessa avaruudessa on suhteellisen hankalaa. Tämän vuoksi on otettu käyttöön kolmion pinnan määräämä avaruus, jota kutsutaan barysentriseksi avaruudeksi. Barysentrisessä avaruudessa jokainen kolmion piste voidaan ilmaista sen kulmapisteiden painotettuna keskiarvona. Näitä keskiarvoja kutsutaan kolmion barysentrisiksi koordinaateiksi. Barysentristen koordinaattien (b_1,b_2,b_3) ja kolmiulotteisen avaruuden koordinaattien $\mathbf{v}_1$, $\mathbf{v}_2$ ja $\mathbf{v}_3$ välillä vallitsee yhteys:

$$(b_1,b_2,b_3)=b_1\mathbf{v}_1+b_2\mathbf{v}_2+b_3\mathbf{v}_3,$$

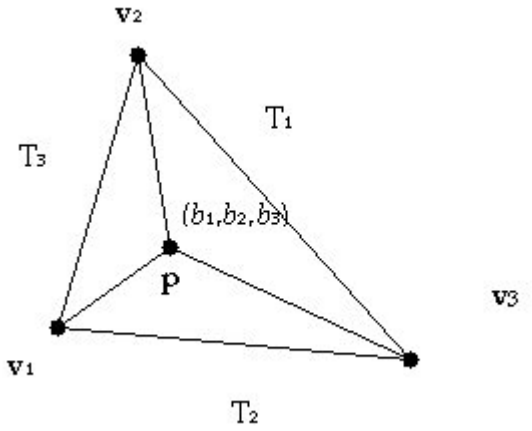
missä $b_1+b_2+b_3=1$. Kuva 18 havainnollistaa barysentristen koordinaattien ja kolmiulotteisen avaruuden koordinaattien yhteyttä.



Kuva 18: Barysentristen ja karteesisten koordinaattien yhteys

Kuvasta 18 voidaan huomata, että jokaisen kulmapisteen $\mathbf{v}_1$, $\mathbf{v}_2$ ja $\mathbf{v}_3$ barysentriset koordinaatit ovat samat kuin kyseisten kulmapisteiden koordinaatit. Tämän lisäksi voidaan huomata, että kulmapisteen $\mathbf{v}_i$ vastapäisellä janalla oleva barysentrinen koordinaatti on nolla. Esimerkiksi kulmapisteen $\mathbf{v}_1$ barysentrinen koordinaatti b_1 vastapäisellä janalla $\mathbf{v}_2$-$\mathbf{v}_3$ on 0. Kolmion sisäpuolella olevat barysentriset koordinaatit b_1, b_2 ja b_3 ovat kaikki suurempia kuin nolla. Jos jonkin kulmapisteen $\mathbf{v}_i$ barysentrinen koordinaatti $b_i<0$, niin kyseinen koordinaatti on kolmion ulkopuolella. Esimerkiksi kuvassa 18 kulmapisteen $\mathbf{v}_3$ barysentrinen koordinaatti $b_3=-1/3$, kun piste liikkuu kolmion ulkopuolelle janan $\mathbf{v}_2$-$\mathbf{v}_1$ yli. Kolmion barysentrinen koordinaatisto muodostaa tason, joka koostuu keskenään samanlaisista kolmioista. Tätä havainnollistavat Dunn ja Barberry kirjansa [10] sivulla 261.

Kun kolmion kulmapisteet tunnetaan, on kolmion barysentristen koordinaattien muuttaminen karteesisiksi koordinaateiksi suoraviivainen toimenpide kaavan $(b_1,b_2,b_3)=b_1\mathbf{v}_1+b_2\mathbf{v}_2+b_3\mathbf{v}_3$ avulla. Jos halutaan muuttaa kolmion mielivaltainen piste $\mathbf{p}$ barysentrisiksi koordinaateiksi, joudutaan tekemään enemmän työtä. Kuva 19 esittää barysentristen koordinaattien määrittämistä kaksiulotteisessa tapauksessa.



Kuva 19: Mielivaltaisen pisteen $\mathbf{p}$ barysentriset koordinaatit

Kuvan 19 mielivaltaiselle pisteelle $\mathbf{p}$ saadaan kaksiulotteisessa tapauksessa yhtälöt:

$$\mathbf{p}_x=b_1x_1+b_2x_2+b_3x_3$$
$$\mathbf{p}_y=b_1y_1+b_2y_2+b_3y_3$$
$$1=b_1+b_2+b_3.$$

Kun tämä yhtälöryhmä ratkaistaan, niin pisteeen $\mathbf{p}$ barysentrisiksi koordinaateiksi b_1,b_2 ja b_3 saadaan

$$b_1=\frac{(\mathbf{p}_y-y_3)(x_2-x_3)+(y_2-y_3)(x_3-\mathbf{p}_x)}{(y_1-y_3)(x_2-x_3)+(y_2-y_3)(x_3-x_1)}$$

$$b_2=\frac{(\mathbf{p}_y-y_1)(x_3-x_1)+(y_3-y_1)(x_1-\mathbf{p}_x)}{(y_1-y_3)(x_2-x_3)+(y_2-y_3)(x_3-x_1)}$$

$$b_3=\frac{(\mathbf{p}_y-y_2)(x_1-x_2)+(y_1-y_2)(x_2-\mathbf{p}_x)}{(y_1-y_3)(x_2-x_3)+(y_2-y_3)(x_3-x_1)}.$$

Jos nyt tarkastellaan saatujen yhtälöiden osoittajia, niin huomataan, että ne vastaavat kuvan 19 osakolmioiden T_1, T_2 ja T_3 kaksinkertaisia pinta-aloja.

Lisäksi jokaisen yhtälön nimittäjä antaa koko kolmion T kaksinkertaisen pinta-alan. Näin ollen kolmion pisteen **p** barysentriset koordinaatit voidaan lausua myös muodossa:

$$b_1 = \frac{\mathrm{A}(\mathrm{T}_1)}{\mathrm{A}(\mathrm{T})}$$
$$b_2 = \frac{\mathrm{A}(\mathrm{T}_2)}{\mathrm{A}(\mathrm{T})}$$
$$b_3 = \frac{\mathrm{A}(\mathrm{T}_3)}{\mathrm{A}(\mathrm{T})}.$$

Kolmiulotteisessa avaruudessa barysentristen koordinaattien laskeminen ei onnistu edellä esitetyllä tavalla, koska ratkaistavaksi tulee yhtälöryhmä, jossa on neljä yhtälöä ja kolme tuntematonta. Tyypillinen ratkaisutapa on kolmion projisoiminen jollekin tasolle xy, xz tai yz. Näin voidaan tehdä, koska projisoitujen kolmioiden pinta-alat ovat suhteessa alkuperäisien kolmioiden pinta-aloihin. Projektiotasoksi valitaan taso, jolle kolmion projektion pinta-ala on suurin. Tällöin vältytään mahdollisilta kolmioiden surkastumisilta viivoiksi. Suurimman projektion omaava taso saadaan tutkimalla kolmion normaalivektoria **n** ja jättämällä yhtälöryhmästä pois se komponentti $\mathbf{p}_x$, $\mathbf{p}_y$ tai $\mathbf{p}_z$, jonka suunnassa normaalivektorin **n** komponentti $\mathbf{n}_x$, $\mathbf{n}_y$ tai $\mathbf{n}_z$ on suurin. Tällöin laskenta palautuu edellä esitettyyn kaksiulotteiseen tapaukseen. Kolmion barysentrisiä koordinaatteja tarvitaan seuraavaksi esiteltävässä säteen ja kolmion leikkauksen määrittämisessä.

7.4. Säteen ja kolmion leikkaus

Säteen ja kolmion leikkauksen määrittämisessä haetaan yleensä ensin säteen ja kolmion muodostaman tason leikkauskohta, jonka jälkeen tutkitaan sisältyykö leikkauskohta tarkasteltavaan kolmioon vai ei. Leikkauskohta pystytään kuitenkin määrittämään ilman erillistä tason ja säteen leikkauksen tutkimista. Kätevän tavan kolmion ja säteen leikkauksen määrittämiseksi ovat esittäneet Möller ja Trumbore [25]. Tässä menetelmässä etsitään arvot kolmion barysentrisille koordinaateille u ja v sekä etäisyys t säteen origosta leikkauspisteeseen.

Olkoon kolmion T mielivaltainen piste annettu muodossa:

$$\mathbf{T}(u,v)=(1-u-v)\mathbf{v}_0+u\mathbf{v}_1+v\mathbf{v}_2.$$

Tässä u ja v ovat barysentriset koordinaatit, joille on voimassa $u \geq 0$ ja $v \geq 0$ sekä $u+v \leq 1$. Nyt säteen $R(t)=\mathbf{p}_0+t\mathbf{d}$ ja kolmion $\mathbf{T}(u,v)$ leikkauspiste saadaan sijoittamalla $R(t)= \mathbf{T}(u,v)$, jolloin saadaan:

$$\mathbf{p}_0+t\mathbf{d}=(1-u-v)\mathbf{v}_0+u\mathbf{v}_1+v\mathbf{v}_2.$$

Kun yhtälö kerrotaan auki ja termit järjestetään uudelleen, saadaan lineaarinen yhtälöryhmä

$$\begin{bmatrix} -\mathbf{d} & \mathbf{v}_1-\mathbf{v}_0 & \mathbf{v}_2-\mathbf{v}_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0-\mathbf{v}_0 \end{bmatrix}.$$

Kyseessä on siis 3x3- ja 3x1-matriisin tulo. Selvyyden vuoksi jätetään vektorien $-\mathbf{d}$, $\mathbf{v}_1-\mathbf{v}_0$ ja $\mathbf{v}_2-\mathbf{v}_0$ yksittäiset komponentit kirjoittamatta. Yhtälöryhmän yksinkertaistamiseksi voidaan merkitä $\mathbf{e}_1=\mathbf{v}_1-\mathbf{v}_0$, $\mathbf{e}_2=\mathbf{v}_2-\mathbf{v}_0$ ja $\mathbf{T}=\mathbf{p}_0-\mathbf{v}_0$. Nyt yhtälöryhmä voidaan ratkaista esimerkiksi Cramerin säännöllä, jolloin saadaan:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{|-\mathbf{d} \quad \mathbf{e}_1 \quad \mathbf{e}_2|} \begin{bmatrix} |\mathbf{T} & \mathbf{e}_1 & \mathbf{e}_2| \\ |-\mathbf{d} & \mathbf{T} & \mathbf{e}_2| \\ |-\mathbf{d} & \mathbf{e}_1 & \mathbf{T}| \end{bmatrix}.$$

Yllä olevassa merkinnässä $|-\mathbf{d} \quad \mathbf{e}_1 \quad \mathbf{e}_2|$ tarkoittaa siis 3x3-matriisin determinanttia. Samoin on $|\mathbf{T} \quad \mathbf{e}_1 \quad \mathbf{e}_2|$ jne. Laskettaessa arvoa t käytetään determinanttia $|\mathbf{T} \quad \mathbf{e}_1 \quad \mathbf{e}_2|$ u:n laskemisessa determinanttia $|-\mathbf{d} \quad \mathbf{T} \quad \mathbf{e}_2|$ ja v:n laskemisessa determinanttia $|-\mathbf{d} \quad \mathbf{e}_1 \quad \mathbf{T}|$.

Säteen ja kolmion leikkauskohdan laskemisen nopeuttamiseksi lasketaan arvot t, u ja v yksitellen. Laskenta lopetetaan heti, jos jokin arvoista on pienempi kuin nolla. Tällöin säde ja kolmio eivät voi leikata toisiaan. Lähdekoodi tässä kuvatun menetelmän toteutukselle on Möllerin ja Trumboren [10] esityksessä.

7.5. Kahden kolmion välinen leikkaus

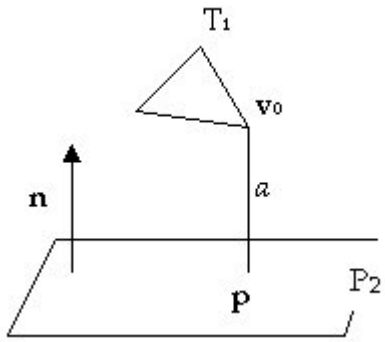
Kohdassa 6 esitetyssä kohteen mukaan suunnatun rajoitustilavuuden menetelmässä suoritetaan lehtitason rajoitustilavuuksissa olevien kolmioiden leikkaustarkastelu, jos kyseiset rajoitustilavuudet ovat päällekkäisiä. Kahden kolmion välinen leikkauspiste voidaan määrittää säteen ja kolmion

leikkauksena, jossa toisen kolmion särmiä käytetään säteinä ja niiden mahdollisia leikkauksia tutkitaan toisen kolmion suhteen. Tarkastelussa käytetään vuoronperään molempien kolmioiden särmiä säteinä. Tämän menetelmän heikkous on kuitenkin siinä, että kolmiot voivat leikatessaan muodostaa myös viivasegmentin.

Tomas Möller [24] esittää ratkaisun kahden kolmion leikkauksen löytämiseksi siten, että leikkaus voi olla joko piste tai viivasegmentti. Oletetaan että annettuna on kaksi kolmiota T_1 ja T_2, joilla on kulmapisteet $\mathbf{v}_0^1$, $\mathbf{v}_1^1$, $\mathbf{v}_2^1$ ja $\mathbf{v}_0^2$, $\mathbf{v}_1^2$, $\mathbf{v}_2^2$. Kolmiot T_1 ja T_2 sijaitsevat tasoilla P_1 ja P_2 alaindeksiensä mukaisesti. Tason P_2 yhtälö on $\mathbf{n}_2 \bullet \mathbf{p} = d_2$, missä $\mathbf{p}$ on tason P_2 mielivaltainen piste ja d_2 on tason P_2 etäisyys origosta. Kolmion T_2 määräämän tason yhtälö voidaan laskea ristitulon avulla seuraavasti:

$$\mathbf{n}_2 = (\mathbf{v}_1^2 - \mathbf{v}_0^2) \mathrm{x} (\mathbf{v}_2^2 - \mathbf{v}_0^2)$$
$$d_2 = -\mathbf{n}_2 \bullet \mathbf{v}_0^2 .$$

Vastaavalla tavalla määritetään tason yhtälö myös kolmion T_1 määräämälle tasolle. Nyt voidaan laskea kolmion T_1 kulmapisteiden etäisyydet tasosta P_2. Kuva 20 selventää asiaa.



Kuva 20: Kolmion T_1 kulmapisteen $\mathbf{v}_0$ etäisyys tasosta P_2.

Pisteen $\mathbf{v}_0$ etäisyys tasosta P_2 saadaan nyt seuraavasti kuvan 20 merkintöjen avulla:

$$\mathbf{p} + a\mathbf{n} = \mathbf{v}_0$$
$$(\mathbf{p} + a\mathbf{n}) \bullet \mathbf{n} = \mathbf{v}_0 \bullet \mathbf{n}$$
$$\mathbf{p} \bullet \mathbf{n} + (a\mathbf{n}) \bullet \mathbf{n} = \mathbf{v}_0 \bullet \mathbf{n}$$
$$d + a = \mathbf{v}_0 \bullet \mathbf{n}$$
$$a = \mathbf{v}_0 \bullet \mathbf{n} - d.$$

Nyt siis minkä tahansa kolmion T_1 kulmapisteen $\mathbf{v}_i^1$ etäisyys a_i tasosta P_2 voidaan kirjoittaa muodossa:

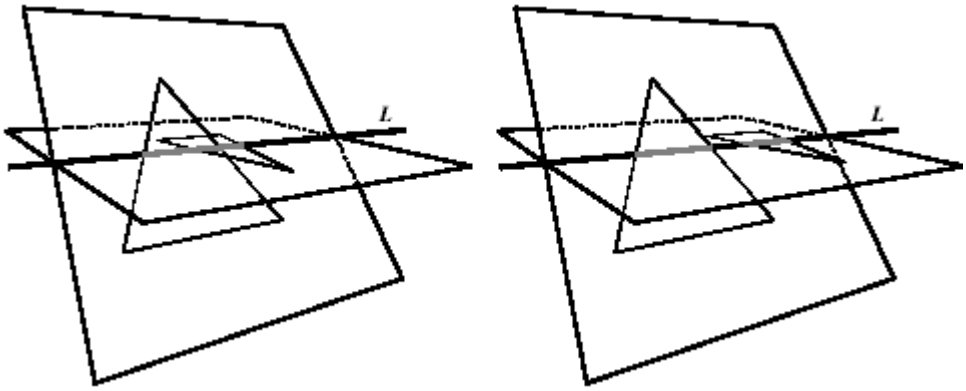
$$a_i = \mathbf{n}_2 \bullet \mathbf{v}_i^1 + d_2 .$$

Kolmion T_2 etäisyydet tasosta P_1 lasketaan vastaavalla tavalla. Jos nyt kaikilla $a_i \neq 0$ ja kaikki etäisyydet a_i ovat samanmerkkisiä, niin kolmio T_1 sijaitsee kokonaan tason P_2 toisella puolella. Tällöin kolmiot T_1 ja T_2 eivät voi leikata toisiaan ja testi voidaan lopettaa. Jos kuitenkin on niin, että löytyy erimerkkisiä etäisyyksien arvoja, niin tasot P_1 ja P_2 leikkaavat toisensa suoraa **L** pitkin. Suora **L** on tällöin vektorin $\mathbf{n}_1 x \mathbf{n}_2$ suuntainen.

Jos kaikki etäisyyksien arvot $a_i=0$, niin kolmiot sijaitsevat samalla tasolla. Tällöin voidaan tarkastaa, leikkaavatko kolmioiden särmät toisiaan. Tämä voidaan toteuttaa projisoimalla kolmiot jollekin tasolle *xy*, *xz* tai *yz*. Kolmiot projisoidaan sellaiselle tasolle, jolla niiden projektioiden pinta-ala on suurin. Projisoinnin jälkeen voidaan kolmioiden särmien leikkaaminen määrittää kaksiulotteisena viivasegmenttien leikkauksena. Jos leikkauskohtia ei löydetä, täytyy vielä tutkia sisältyykö toinen kolmioista toiseen. Yleisessä tapauksessa on suhteellisen harvinaista, että kaksi kolmiota on samalla tasolla, joten tässä tapauksessa leikkauksen määrittämisessä voidaan käyttää tavallista raa'an voiman menetelmää.

Sellaisessa tapauksessa, missä kolmiot sijaitsevat eri tasoilla saadaan tasojen leikkausviivan yhtälöksi $\mathbf{L}(t)=\mathbf{p}_0+t\mathbf{d}$. Leikkausviivan yhtälössä suuntavektori $\mathbf{d}=\mathbf{n}_1 x \mathbf{n}_2$, missä $\mathbf{n}_1$ ja $\mathbf{n}_2$ ovat kolmioiden $\mathbf{T}_1$ ja $\mathbf{T}_2$ normaalivektorit. Piste $\mathbf{p}_0$ on leikkausviivan origo, joka voidaan asettaa halutulla tavalla, koska sen muuttaminen ei vaikuta lopulliseen leikkaustarkasteluun.

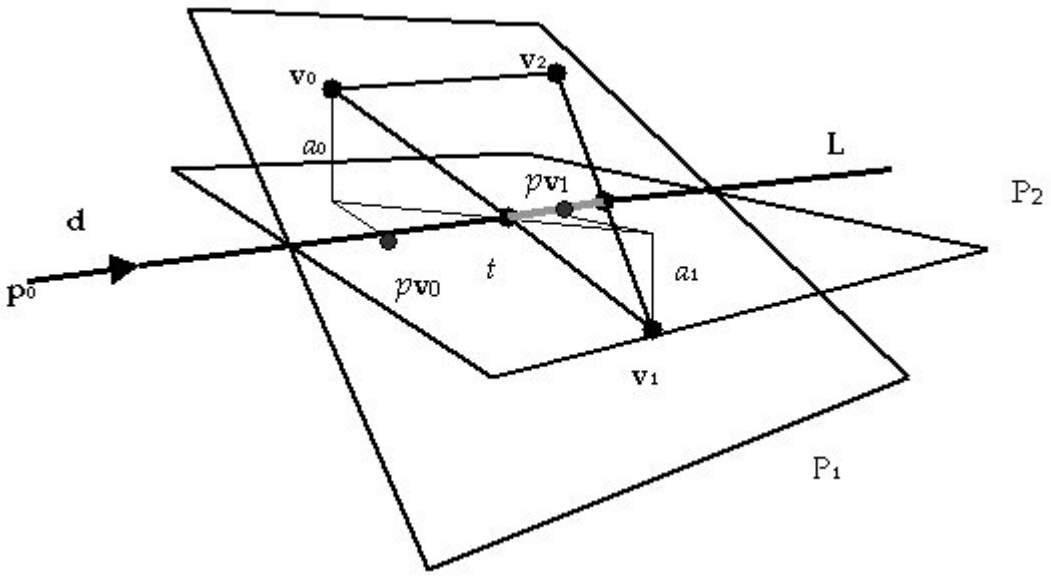
Kaksi eri tasoilla olevaa kolmiota voivat sijaita avaruudessa kahdella eri tavalla toisiinsa nähden. Nämä tapaukset on esitetty kuvassa 21. Kuvassa 21 viivasegmentin **L** harmaalla esitetty alue kuvaa kolmioiden ja viivasegmentin **L** leikkausta. Jos kolmioilla on yhteisiä pisteitä viivasegmentillä **L**, niin ne leikkaavat toisensa.



Kuva 21: Kolmioiden sijoittuminen kahdella risteävällä tasolla.

Kahden kolmion leikkauksen toteamiseksi lasketaan kummallekin kolmiolle intervallit viivasegmentillä **L**. Jos nämä intervallit ovat päällekkäisiä, niin kolmiot leikkaavat toisensa.

Intervallien laskemiseksi projisoidaan kolmion T_1 kulmapisteet viivasegmentille **L**. Lisäksi lasketaan kolmion T_1 kulmapisteiden etäisyydet a_0, a_1 ja a_3 tasosta P_2. Kuvassa 22 kulmapisteiden projektioista viivasegmentillä **L** käytetään merkintää $p\mathbf{v}_0$ ja $p\mathbf{v}_1$. Harmaalla värillä merkityn intervallin alkupiste t ilmaisee kolmion särmän $\mathbf{v}_0$-$\mathbf{v}_1$ leikkauskohdan skalaariarvon viivasgmentillä $\mathbf{L}(t)=\mathbf{p}_0+\mathbf{d}t$.



Kuva 22: Kolmion kulmapisteiden projisoiminen viivasegmentille L ja niiden etäisyydet tasosta P_2.

Kuvasta 22 voidaan huomata, että kolmion T_1 kulmapisteiden projektiot ovat $p\mathbf{v}_0$, $p\mathbf{v}_1$ ja niiden etäisyydet a_0 ja a_1 tasosta P_2 muodostavat kaksi keskenään yhdenmuotoista kolmiota. Näiden avulla voidaan laskea intervallin alkupiste t seuraavasti:

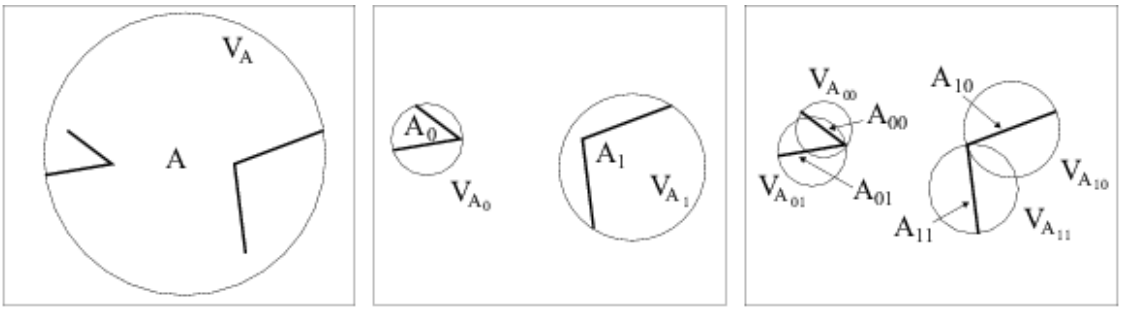
$$t = p\mathbf{v}_0 + (p\mathbf{v}_1 - p\mathbf{v}_0)\frac{a_1}{(a_0 - a_1)}.$$

Vastaavalla tavalla lasketaan intervallin loppupiste, mutta tässä käytetään tietenkin kulmapisteitä $\mathbf{v}_0$ ja $\mathbf{v}_2$. Tämän jälkeen suoritetaan samat laskutoimenpiteet kolmiolle T_2 tason P_1 suhteen. Lopuksi tutkitaan, onko saaduilla kahdella intervallilla yhteisiä pisteitä. Jos yhteisiä pisteitä on olemassa, niin kaksi kolmiota leikkaavat toisensa.

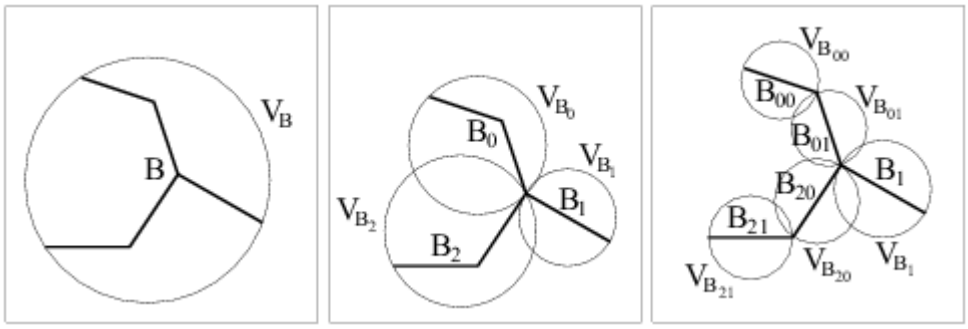
Tässä tutkielmassa on esitetty periaatetasolla tyypillisimpiä geometristen primitiivien välisiä leikkauksia. Sovelluskohtaisesti erilaisia leikkaustyyppejä voi olla kuinka paljon tahansa. Dunn ja Barberry [12] käsittelevät erilaisia leikkauksia kirjansa sivuilla 283-318. Lisää erilaisia leikkausalgoritmeja esittävät mm. Arenberg [2] ja Badouel [3].

8. Törmäyskyselyiden tarkasteluja rajoitustilavuushierarkiassa

Olkoon annettuna kaksi rajoitustilavuushierarkiaa V_A ja V_B, joiden sisältämien kohteiden välisiä törmäyksiä tutkitaan. Kuva 23 esittää murtoviivoista koostuvan kohteen A rajoitustilavuushierarkian V_A kolme eri tasoa. Ensimmäisenä esitetään juuritaso, joka sisältää kohteen kokonaisuudessaan, ja viimeisenä on esitetty lehtitaso, jossa jokaisella viivasegmentillä on oma rajoitustilavuutensa. Kuva 24 esittää vastaavan tilanteen rajoitustilavuushierarkialle B. Tämän luvun kuvat Gottschalk [12].

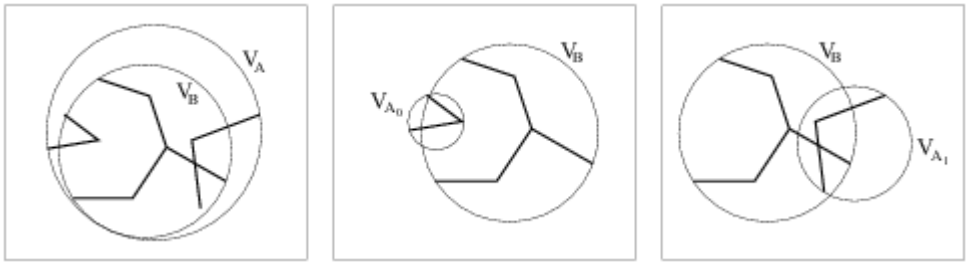


Kuva 23: Rajoitustilavuushierarkian V_A kolme tasoa.



Kuva 24: Rajoitustilavuushierarkian V_B kolme tasoa.

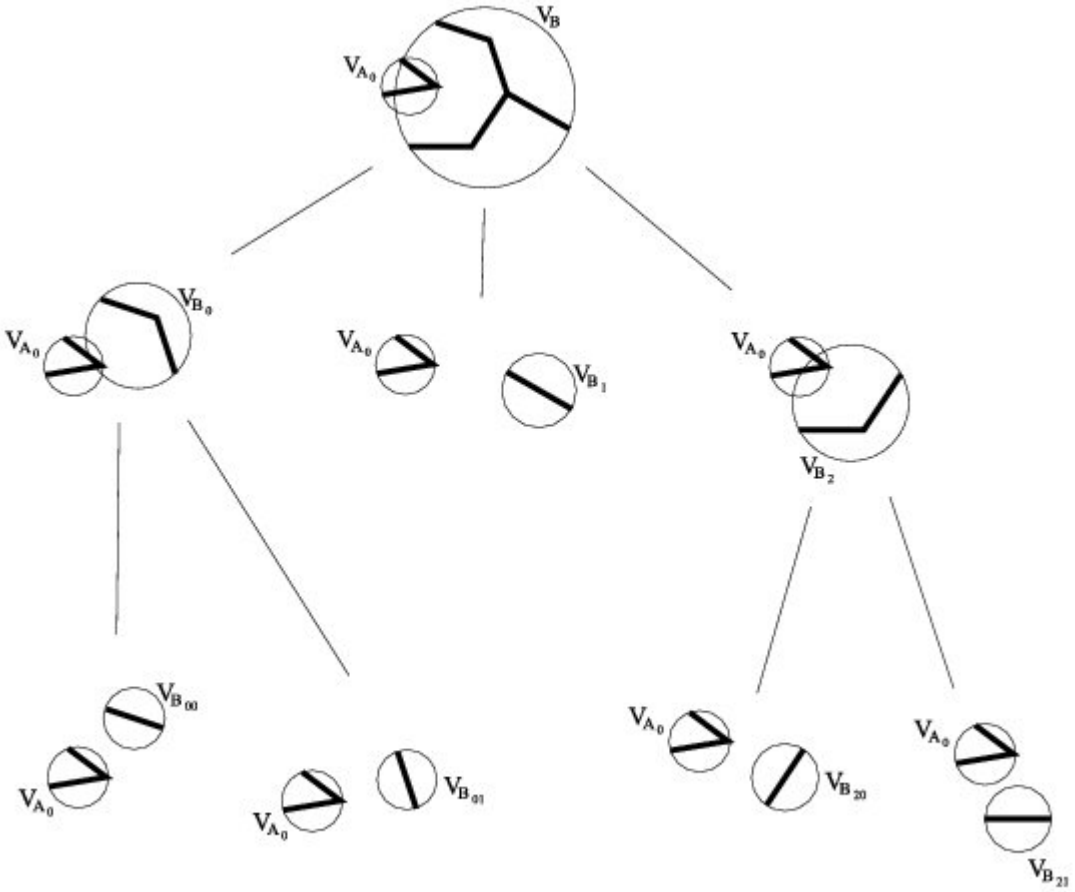
Törmäyskyselyiden suorittamiseksi siirretään kohteet A ja B yhteiseen maailmankoordinaatistoon. Ensimmäisessä törmäyskyselyssä testataan kohteiden A ja B juuritason rajoitustilavuuksien päällekkäisyyttä. Käytetään tästä testistä merkintää $T(V_A,V_B)$. Jos juuritason rajoitustilavuudet menevät päällekkäin, joudutaan suorittamaan uusia törmäyskyselyitä, joissa verrataan kohteen A rajoitustilavuuden V_A lapsien V_{A0} ja V_{A1} päällekkäisyyttä rajoitustilavuuden V_B kanssa. Tällöin suoritetaan siis testit $T(V_{A0},V_B)$ ja $T(V_{A1},V_B)$. Kuvassa 25 havainnollistetaan päätason törmäyskyselyn jakautumista kahdeksi alemman tason kyselyksi. Kuvassa 25 testiparit $T(V_{A0},V_B)$ ja $T(V_{A1},V_B)$ on valittu siten, että siirrytään testaamaan suuremman rajoitustilavuuden lapsia ensimmäisen testin pienempää rajoitustilavuutta vasten. Testattavien rajoitustilavuuksien lapsisolmut voidaan valita myös muilla tavoilla, mutta kirjallisuudessa tässä esitetty tapa on yleisin.



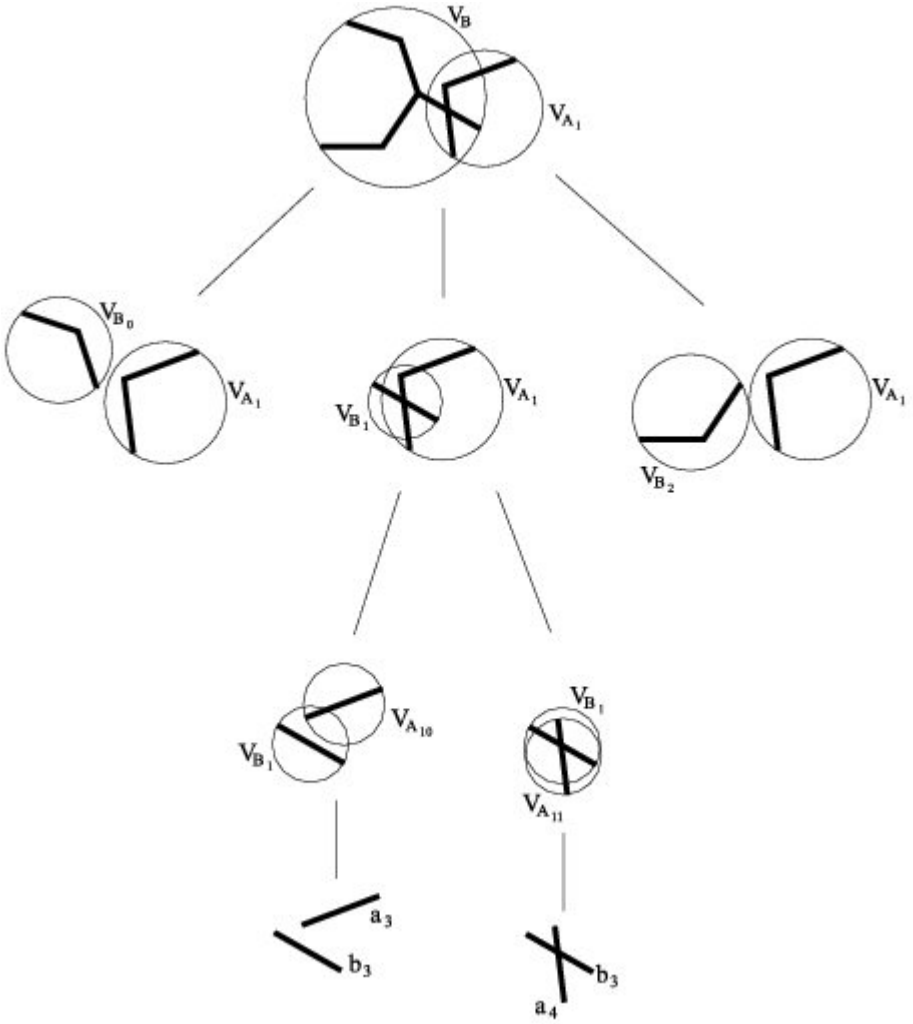
Kuva 25: Päätason törmäyskyselyn jakautuminen kahdeksi alikyselyksi.

Aikaisemmasta muistetaan, että törmäyskysely lopetetaan heti, jos testattavat rajoitustilavuudet ovat erillisiä. Muussa tapauksessa testiä jatketaan lehtitasolle saakka, jossa voidaan määrittää kohteiden A ja B ne primitiivit, jotka leikkaavat toisensa. Kuvassa 26 on kuvattu kyselyn $T(V_{A0},V_B)$ ja kuvassa 27 testin $T(V_{A1},V_B)$ eteneminen lehtitasolle saakka. Kysely $T(V_{A0},V_B)$ ei löydä

törmäyksiä rajoitustilavuuksien V_{A0} ja V_B sisältämien primitiivien välillä, mutta kysely $T(V_{A1},V_B)$ tuottaa leikkauksen primitiivien a_4 ja b_3 välillä. Kuvista 26 ja 27 voidaan huomata, että kaikki alikyselyt haarautuvat suuremman rajoitustilavuuden lapsien vertaamiseen pienempään rajoitustilavuuteen.



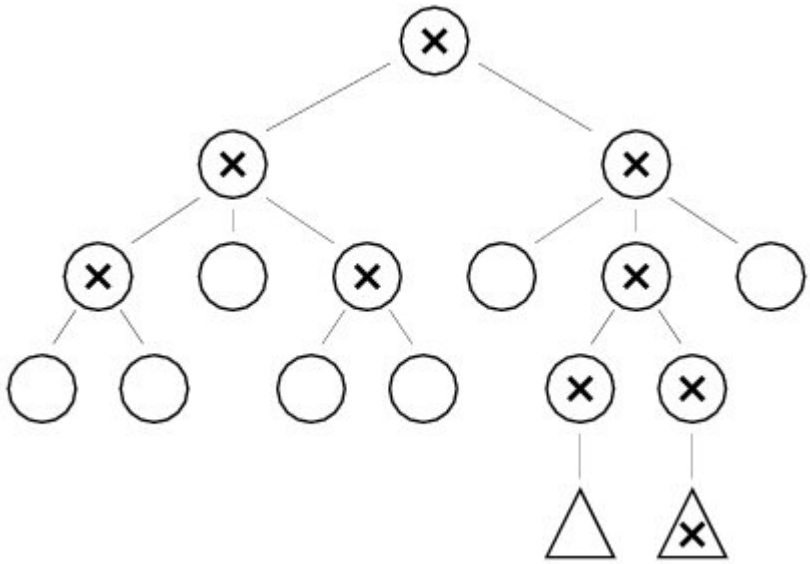
Kuva 26: Törmäyskyselyn $T(V_{A0},V_B)$ eteneminen lehtitasolle.



Kuva 27: Törmäyskyselyn $T(V_{A1},V_B)$ eteneminen lehtitasolle.

Rajoitustilavuuksien V_A ja V_B välillä suoritettavat törmäyskyselyt muodostavat puurakenteen edellä kuvatulla tavalla. Tällaista puurakennetta voidaan käyttää törmäyskyselyjen lukumäärän alarajan määrittämiseen tietyntyyppisten optimointitekniikoiden yhteydessä. Tämän lisäksi puurakennetta voidaan käyttää törmäyskyselyjen lukumäärän määrittämiseksi, jos halutaan löytää $k>0$ leikkausta kohteiden A ja B primitiivien välillä. Puurakenteesta käytetään englannin kielessä lyhennettä BVTT (bounding volume test tree). Tässä tutkielmassa rakennetta kutsutaan nimellä testipuu.

Edellä esitetty törmäyskysely rajoitustilavuuksien V_A ja V_B välillä voidaan ilmaista symbolisessa muodossa kuvan 27 avulla. Kuvassa 27 tyhjä solmu tarkoittaa kyselyä, joka tuottaa erilliset rajoitustilavuudet ja ”X” tarkoittaa kyselyä, joka tuottaa päällekkäiset rajoitustilavuudet. Lehtitasolla oleva tyhjä kolmio tarkoittaa sellaista primitiivien välistä kyselyä, jossa leikkausta ei todeta. Kolmion sisältämä ”X” tarkoittaa sellaista primitiivien välistä kyselyä, jossa primitiivit leikkaavat toisensa. Kuvan 28 kolmion sisältämä ”X” tarkoittaa siis törmäyskyselyn $T(V_A,V_B)$ tuottamaa primitiivien a_4 ja b_3 välistä leikkausta.



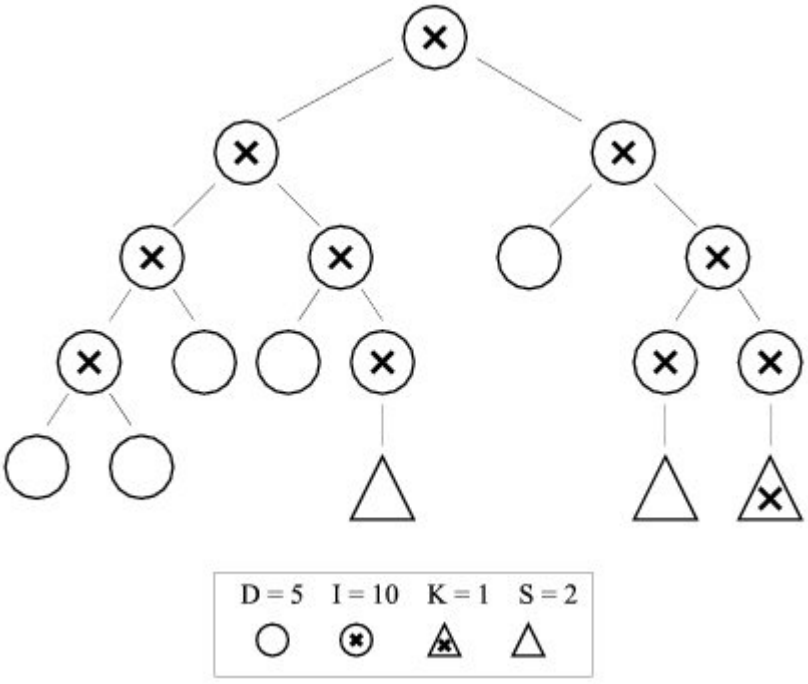
Kuva 28: Törmäyskyselyn T(V_A,V_B) testipuu.

Törmäyskysely T(V_A,V_B) tuottaa testipuun, joka ei ole binaarinen. Yleensä kohteista pyritään rakentamaan sellaisia rajoitustilavuushierarkioita, jotka ovat binaarisia. Tällöin myös testipuusta tulee binaarinen. Seuraavaksi tarkastellaan binaariseen testipuuhun liittyviä invariantteja. Näiden invarianttien avulla voidaan määrittää törmäyskyselyiden lukumäärille alarajat, kun käytetään kyselyiden optimoinnissa varhaiseen lopettamiseen sekä ajalliseen koherenssiin perustuvia tekniikoita.

8.1. Binaarisen testipuun invariantit

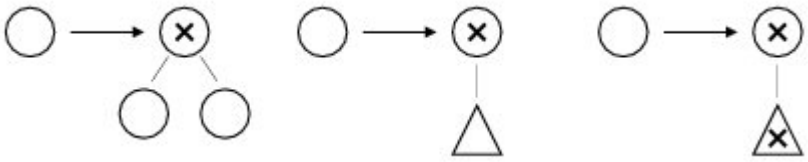
Tarkastellaan kuvan 29 mukaista binaarista testipuuta. Kyseinen testipuu kuvaa tilannetta, jossa kahden kohteen välillä on suoritettu kymmenen kyselyä, joiden tuloksena on saatu päällekkäiset rajoitustilavuudet. Viiden kyselyn tuloksena on saatu erilliset rajoitustilavuudet. Näiden lisäksi lehtitason tarkasteluissa on suoritettu kaksi kyselyä, joiden tuloksena on erilliset primitiivit, sekä yksi kahden primitiivin leikkauksen tuottanut kysely.

Jos testipuu on binaarinen, niin se sisältää invariantin, joka liittää erityyppisten törmäyskyselyjen lukumäärän toisiinsa siten, että $D-I+2S+2K=1$. Tässä D on niiden kyselyjen lukumäärä, jotka tuottavat erilliset rajoitustilavuudet. I on niiden kyselyiden lukumäärä, jotka tuottavat päällekkäiset rajoitustilavuudet. Lehtitason kyselyissä S antaa sellaisten kyselyjen lukumäärän, jotka tuottavat erilliset primitiivit, ja K on leikkauksen tuottavien kyselyiden lukumäärä.



Kuva 29: Binaarinen testipuu.

Jokainen rajoitustilavuushierarkiassa suoritettu törmäyskysely voi päättyä kolmella eri tavalla kuvan 30 mukaisesti. Törmäyskysely päättyy, jos tutkittavat rajoitustilavuudet ovat erilliset, lehtitason kysely tuottaa erilliset primitiivit tai lehtitason primitiivit leikkaavat toisensa.



Kuva 30: Erityyppisten törmäyskyselyiden päättymisehdot.

Invariantti $D-I+2S+2K=1$ on nyt todistettavissa induktiolla testipuuhun suoritettavien törmäyskyselyiden lukumäärän suhteen. Kuvataan testipuuhun tehtävää kyselysarjaa merkinnällä $(D,I,S,K)=(a,b,c,d)$, missä a, b, c ja d kuvaavat vastaavien törmäyskyselyiden D, I, S ja K lukumääriä. Otetaan aluksi perusaskel ja suoritetaan törmäyskysely sellaisille kohteille, jotka ovat erillisiä. Tällöin saadaan $(D,I,S,K)=(1,0,0,0)$, joka toteuttaa invariantin.

Edellisen kyselyn kanssa yhtäpitävä lopetusehto saadaan, kun kysely korvataan kuvan 30 ensimmäisen kohdan kyselyllä. Tällöin alkuperäiseen kyselysarjaan tulee lisäksi yksi erilliset rajoitustilavuudet tuottava kysely *D* sekä yksi päällekkäiset rajoitustilavudet tuottava kysely *I*. Nyt kyselysarja on siis muotoa *(D,I,S,K)*=(2,1,0,0), mikä toteuttaa invariantin. Jos ensimmäinen kysely korvataan kuvan 30 toisen kohdan kyselyllä, niin tällöin kyselysarjasta vähennetään yksi *D*-kysely ja siihen lisätään yksi *I*-ja yksi *S*-kysely. Nyt kyselysarja saadaan kirjoitettua muodossa *(D,I,S,K)*=(0,1,1,0), joka toteuttaa invariantin. Viimeinen erilainen, mutta alkuperäisen kyselyn kanssa yhtäpitävä lopetusehto saadaan, kun alkuperäisestä kyselystä poistetaan yksi *D* kysely ja siihen lisätään yksi *I*- ja yksi *K*-kysely. Tällöin kyselysarja on muotoa *(D,I,S,K)*=(0,1,0,1), joka toteuttaa invariantin.

8.2. Yläraja varhaiseen lopettamiseen perustuvalle törmäyskyselylle

Törmäyskyselyissä voidaan käyttää varhaiseen lopettamiseen perustuvaa optimointia, jos on epätodennäköistä, että kaksi kohdetta törmää toisiinsa. Tällaisessa tapauksessa edellä esitetty invariantti voidaan kirjoittaa muodossa $D=I+1$. Nyt siis lehtitason törmäyskyselyiden *K* ja *S* lukumäärät ovat nollia.

Varhaiseen lopettamiseen perustuvassa optimoinnissa käytetään kerroksellista rajoitustilavuushierarkiaa. Tämä tarkoittaa sitä, että monimutkaisempi rajoitustilavuus suljetaan yksinkertaisemman rajoitustilavuuden sisään. Rajoitustilavuuksien päällekkäisyyksien testaaminen suoritetaan ensin yksinkertaisemmalla rajoitustilavuudella. Varsinaista rajoitustilavuutta käytetään ainoastaan, jos on pakko. Yleensä yksinkertaisin käytetty rajoitustilavuus on pallo, jonka sijainti avaruudessa voidaan päivittää nopeasti sen keskipisteen ja säteen avulla. Pallon sijainnin päivitys on huomattavasti nopeampaa kuin esimerkiksi kohteen mukaan suunnatun rajoitustilavuuden päivitys. Oletetaan, että pallon sijainnin päivittäminen on ilmainen toimenpide ja että pallolla testaaminen havaitsee erilliset rajoitustilavuudet samalla tavalla kuin kohteen mukaan suunnatun rajoitustilavuuden testi. Lisäksi oletetaan, että rajoitustilavuustestien lukumäärä pysyy muuttumattomana. Nyt törmäyskyselysarjalle saatu nopeutus, joka on saatu käyttämällä yksinkertaisempia rajoitustilavuustestejä, voidaan kirjoittaa muodossa

$$\mathrm{s} = \frac{IT_\mathrm{I} + DT_\mathrm{D}}{IT_\mathrm{I}} = 1 + \frac{DT_\mathrm{D}}{IT_\mathrm{I}} \approx 1 + \frac{T_\mathrm{D}}{T_\mathrm{I}},$$

missä I on niiden törmäyskyselyiden lukumäärä, jotka tuottavat päällekkäiset rajoitustilavuudet ja D on niiden törmäyskyselyiden lukumäärä, jotka tuottavat erilliset rajoitustilavuudet. T_I ja T_D ovat vastaavasti kyseisten testien suoritusajat. Tässä on huomattava, että D ja T_D tarkoittavat alkuperäisen rajoitustilavuuden testien lukumääriä ja suoritusaikaa. Nimittäjästä on jätetty pois yksinkertaisemman testin suoritusaika, joka oletettiin nollaksi. Invariantti $D=I+1$ voidaan nyt yksinkertaistaa muotoon $D=I$.

Kohteen mukaan suunnatun rajoitustilavuuden menetelmässä rajoitustilavuuksien päällekkäisyyden havaitsemiseksi joudutaan suorittamaan 15 eri testiä. Jos rajoitustilavuudet ovat erillisiä, niin testejä tarvitaan vähemmän. Näin ollen $T_D>T_I$. Jos yksinkertaisuuden vuoksi oletetaan, että $T_D=T_I$, niin saadaan varhaisen lopettamisen menetelmän tuoman törmäyskyselyn nopeutukseksi $s=2$. Ihanteellisissa olosuhteissa varhaiseen lopettamiseen perustuva törmäyskyselyn optimointi tuottaa siis kaksinkertaisen nopeutuksen alkuperäiseen törmäyskyselyyn verrattuna. Todellisessa tilanteessa kohteiden välillä on kuitenkin törmäyksiä ja joudutaan tutkimaan myös lehtitason primitiivien väliset leikkaukset. Tällöin varhaiseen lopettamiseen perustuvan optimoinnin nopeutus on paljon pienempi kuin 2.

8.3. Yläraja ajalliseen koherenssiin perustuvalle törmäyskyselylle

Jos oletetaan, että animaatiossa olevien kohteiden sijainnit eivät muutu ratkaisevasti peräkkäisten animaatioruutujen välillä, niin tätä voidaan hyödyntää törmäyskyselyiden suorittamisessa. Tässä tapauksessa törmäyskyselyä ei tarvitse aloittaa juuritasojen testaamisella, vaan edelliseen animaatioruutuun liittyvää törmäyskyselyä voidaan käyttää hyväksi. Tämän toteuttamiseksi ylläpidetään listaa edelliseen animaatioruutuun perustuvan törmäyskyselyn lehtisolmuista. Ihanteellisessa tapauksessa kohteet eivät törmää, ja lehtisolmut ovat testejä, jotka tuottavat erilliset rajoitustilavuudet. Uudelle animaatioruudulle suoritetaan törmäystarkastelu kohteen uuden sijainnin mukaan käyttämällä edellisen ruudun törmäyskyselyn lehtisolmuja. Jos kohteiden välinen törmäystilanne ei muutu, säästytään juuri- ja lehtisolmujen väliin jääviltä törmäyskyselyiltä sekä kohteiden primitiivien välisiltä leikkauskyselyiltä. Tällöin törmäyskyselyn nopeutus voidaan kirjoittaa muodossa

$$s = \frac{IT_\mathrm{I} + DT_\mathrm{D}}{DT_\mathrm{D}} = 1 + \frac{IT_\mathrm{I}}{DT_\mathrm{D}} \approx 1 + \frac{T_\mathrm{I}}{T_\mathrm{D}}.$$

Tässäkin tapauksessa invarianttia $D=I+1$ on yksinkertaistettu siten, että $D=I$. Lisäksi oletetaan, että $T_D=T_I$. Näin ollen huomataan, että ajalliseen koherenssiin perustuvan törmäyskyselyn nopeutus ihannetapauksessa alkuperäiseen nähden on siis $s=2$.

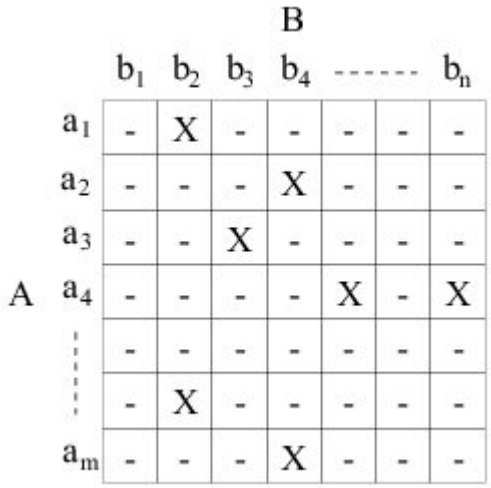
Todellisessa tilanteessa nopeutuksen tuoma hyöty on pienempi, koska jos törmäystilanne muuttuu kahden animaatioruudun välillä, niin joudutaan suorittamaan myös leikkaustarkasteluja lehtitason primitiivien välillä. Joissain tapauksissa tömäyskysely voidaan joutua myös aloittamaan edellisen animaatioruudun lehtisolmua ylemmältä tasolta. Tässä tapauksessa on kuitenkin niin, että ajallisen koherenssin oletus ei enää ole voimassa.

8.4. Kontaktiparimatriisi

Kontaktiparimatriisi on edellä esitellyn testipuun ohella työkalu, jonka avulla voidaan arvioida törmäyskyselyitä rajoitustilavuushierarkiassa. Kontaktiparimatriisin avulla voidaan todeta, että rekursiivinen törmäyskysely löytää kaikki mahdolliset primitiivien leikkaukset kahden kohteen kolmioverkon välillä. Tämän lisäksi kontaktiparimatriisin avulla voidaan löytää ylä- ja alarajat törmäyskyselyiden lukumäärälle, kun kohteiden välillä on $k>0$ törmäystä.

Oletetaan, että kohde A sisältää m kolmiota ja kohde B sisältää n kolmiota. Tällöin on olemassa mn mahdollista törmäysparia kohteiden A ja B välillä. Kohteiden A ja B välillä tapahtuvat primitiivien leikkaukset on esitetty kuvassa 31 olevassa kontaktiparimatriisissa. Tässä matriisissa kohteen A primitiivit $a_1,\ldots a_m$ on esitetty matriisin riveinä ja kohteen B primitiivit $b_1,\ldots,b_n$ matriisin sarakkeina. Kuvan 31 kontaktiparimatriisin ruutuun merkitty ”-” tarkoittaa, että kohteiden A ja B primitiivit eivät leikkaa toisiaan. Merkintä ”X” puolestaan tarkoittaa kohteiden A ja B välillä tapahtuvaa primitiivien välistä leikkausta.

Kontaktiparimatriisin läpikäyminen raa’an voiman menetelmällä on aika-vaatimukseltaan $O(mn)$ toimenpide, mutta rajoitustilavuushierarkiaan perustuvat törmäyskyselyt voivat sulkea kokonaisia matriisin lohkoja jatko-tarkastelun ulkopuolelle. Seuraavaksi tutkitaan rajoitustilavuushierarkiaan perustuvan törmäyskyselyn kykyä löytää kaikki leikkaukset kahden kohteen välillä. Tämän jälkeen etsitään ylä- ja alarajat törmäyskyselyiden lukumäärälle $k>0$ törmäyksen löytämiseksi.



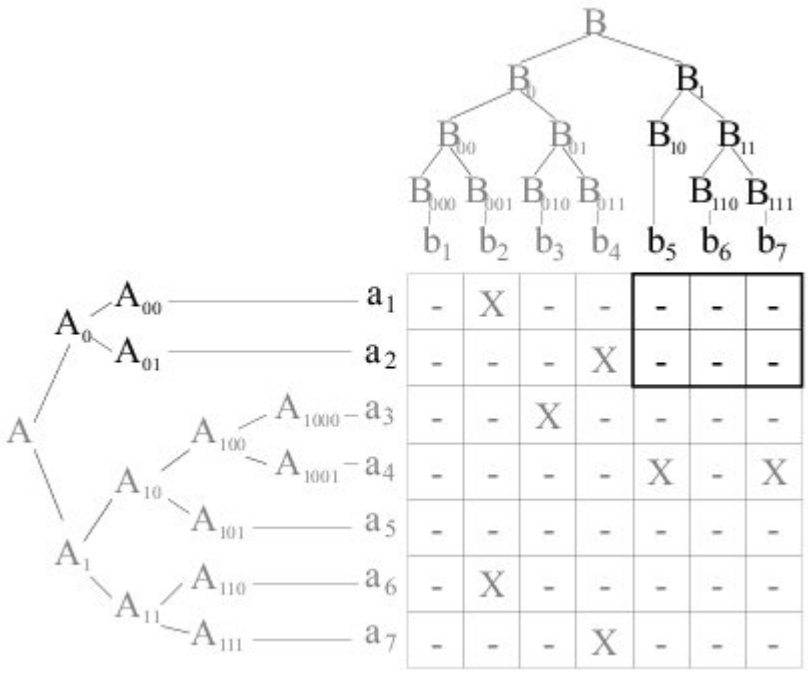
Kuva 31: Kontaktiparimatriisi.

8.5. Törmäyskyselyn oikeellisuus rajoitustilavuushierarkiassa

Tarkastellaan kohteiden A ja B välistä törmäyskyselyä $T(V_A,V_B)$. Tämä johtaa alikyselyihin, joissa vertallaan suuremman rajoitustilavuuden lapsia pienempään rajoitustilavuuteen. Tässä kyselysarjassa verrataan siis vuorotellen rajoitustilavuushierarkian V_A lapsia rajoitustilavuushierarkiaan V_B ja päinvastoin. Tämä kyselytapa johtaa lopulta molempien rajoitustilavuushierarkioiden V_A ja V_B osittamiseen rajoitustilavuuksien kokojen määräämillä tavoilla. Tämä osittaminen määrää myös kontaktiparimatriisin lohkojen määräytymisen.

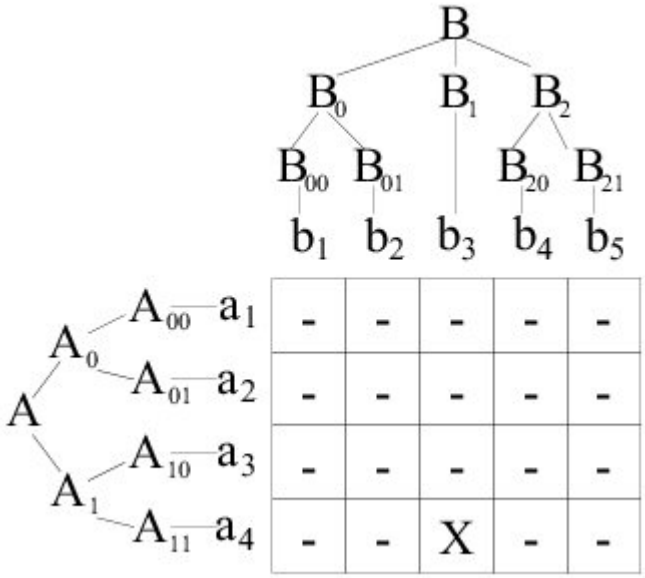
Kuva 32 esittää kontaktiparimatriisin lohkon muodostumisen kyselyllä $T(V_{A0},V_{B1})$. Kuvassa 32 oletetaan, että kohteiden A ja B primitiivit $a_1,\ldots,a_7$ ja $b_1,\ldots,b_7$ ovat puiden A ja B esijärjestyskulkemisen määräämissä järjestyksissä. Kysely $T(V_{A0},V_{B1})$ tuottaa erilliset rajoitustilavuudet V_{A0} ja V_{B1}. Tämä tarkoittaa sitä, että yksikään niistä primitiiveistä, jotka sisältyvät rajoitustilavuuteen V_{A0}, ei voi leikata rajoitustilavuudessa V_{B1} olevia primitiivejä.

Kontaktiparimatriisissa tämä näkyy 2x3-kokoisena alilohkona, joka on merkitty arvoilla ”-”. Nyt on huomattava, että kontaktiparimatriisissa saattaa olla leikkauksista vapaa lohko, vaikka testi tuottaisi päällekkäiset rajoitustilavuudet. Joka tapauksessa erilliset rajoitustilavuudet tuottava testi takaa sen, että testiä vastaava lohko kontaktiparimatriisissa ei sisällä primitiivien välisiä leikkauksia.



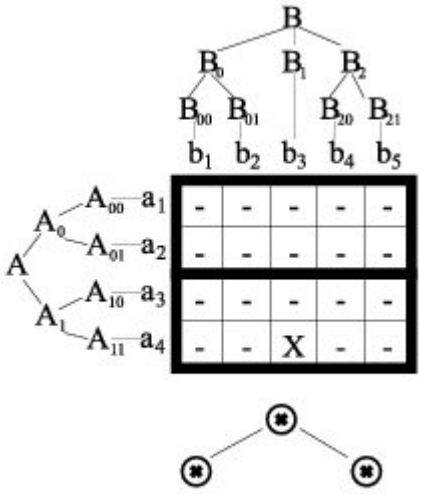
Kuva 32: Törmäyskyselyn $T(V_{A0},V_{B1})$ tuottama 2x3-kokoinen lohko, jossa ei ole kohteiden A ja B primitiivien välillä leikkauksia.

Käydään seuraavaksi läpi kuvissa 23, 24, 25, 26 ja 27 esitetty törmäyskysely $T(V_A,V_B)$. Kuvan 23 esittämä kohde A sisältää neljä viivasegmenttiä ja kuvan 24 esittämä kohde B sisältää viisi viivasegmenttiä. Näin ollen kohteiden A ja B välille saadaan kontaktiparimatriisi, jossa on neljä riviä ja viisi saraketta kuvan 33 mukaisesti.



Kuva 33: Rajoitustilavuushierarkioita A ja B vastaava kontaktiparimatriisi.

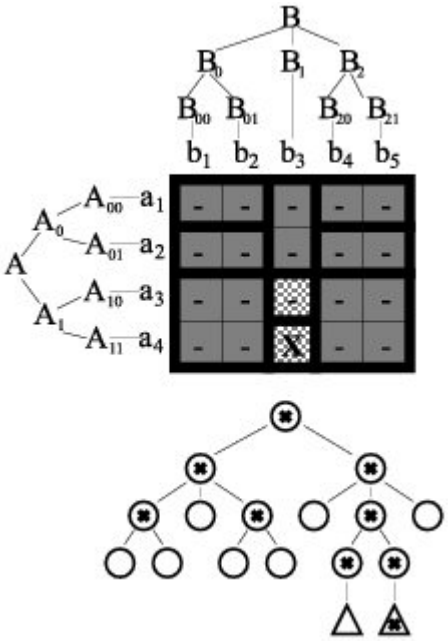
Kuvassa 33 rajoitustilavuushierarkioiden A ja B välisessä kontaktiparimatriisissa on havaittu yksi leikkaus primitiivien a_4 ja b_3 välillä. Kuvan 33 kontaktiparimatriisi kuvaa ensimmäistä törmäyskyselyä kohteiden A ja B välillä. Tämä kysely johtaa kahteen uuteen törmäyskyselyyn kuvan 34 mukaisesti.



Kuva 34: Kyselyiden $T(V_{A0},VB)$ ja $T(V_{A1},VB)$ tuottama kontaktiparimatriisin jako.

Kuvassa 34 on esitetty tilanne, jossa kyselyt $T(V_{A0},VB)$ ja $T(V_{A1},VB)$ jakavat alkuperäisen kontaktiparimatriisin kahteen lohkoon. Molemmat lohkot tuottavat päällekkäiset rajoitustilavuudet. Kuvista 25 ja 26 voidaan nähdä, että seuraavaksi suoritetaan kuusi törmäyskyselyä rajoitustilavuuksien (V_{A0}, V_{B0}), (V_{A0}, V_{B1}), (V_{A0}, V_{B2}), (V_{A1}, V_{B0}), (V_{A1}, V_{B1}) ja (V_{A1}, V_{B2}) välillä. Näiden kyselyiden suorittamisen jälkeen kontaktiparimatriisi on kuvan 34 mukainen. Harmaalla värillä sävytetyt lohkot tarkoittavat erillisiä rajoitustilavuuksia. Tämä tarkoittaa sitä, että näissä lohkoissa ei voi olla leikkaavia primitiivejä, eikä näitä lohkoja tarvitse enää testata.

Viimeisenä suoritetaan kaksi kyselyä kuvan 35 mukaisesti. Kyselyt $T(V_{A10},V_{B1})$ ja $T(V_{A11},V_{B1})$ tuottavat kumpikin päällekkäiset rajoitustilavuudet. Näiden testien jälkeen rajoitustilavuushierarkioita V_{A10}, V_{A10} ja V_{B1} ei enää voida jakaa pienemmiksi, vaan on suoritettava leikkaustestit kyseisissä rajoitustilavuuksissa olevien primitiivien välillä. Primitiivien välisiä leikkaustestejä on merkitty viivoituksella kuvassa 35.



Kuva 35: Kontaktiparimatriisi primitiivien leikkauksen löytymisen jälkeen.

Törmäyskyselyjen suorittaminen voidaan nähdä kontaktiparimatriisin pilkkomisena pienempiin osiin rajoitustilavuuksien suuruuden avulla. Joka tapauksessa törmäyskyselyjä jatketaan niihin testipuun haaroihin, joista rajoitustilavuuksien päällekkäisyydet löytyvät. Näin ollen rajoitustilavuushierarkioihin perustuva törmäyskysely löytää kaikki kohteiden A ja B välillä olevat primitiivien leikkaukset. Näin on siksi, että yksikään kontaktiparimatriisin osiointi ei voi tapahtua siten, että jokin lohko joutuisi eristetyksi tulevien alikyselyiden ulkopuolelle.

Seuraavaksi tarkastellaan rajoitustilavuushierarkiaan perustuvien törmäyskyselyiden lukumäärän ylä- ja alarajojen määräytymistä kontaktiparimatriisin avulla.

8.6. Törmäyskyselyiden lukumäärien rajat rajoitustilavuushierarkiassa

Rajoitustilavuushierarkiassa suoritettavien törmäyskyselyiden lukumäärien ylä- ja alarajojen määrittämiseksi tarvitaan ideaalisen rajoitustilavuuden käsitettä. Ideaalinen rajoitustilavuus on sellainen kuvitteellinen rajoitustilavuus, jota käytettäessä törmäyskysely tuottaa erilliset rajoitustilavuudet, jos ja vain jos kyseisten rajoitustilavuuksien sisältämät primitiivit eivät leikkaa toisiaan. Tällaisen ideaalisen rajoitustilavuuden tuottama törmäyskyselyiden lukumäärän alaraja on myös minkä tahansa rajoitustilavuustyypin törmäyskyselyiden alaraja. Seuraavissa tarkasteluissa oletetaan, että käytettävissä on ideaalinen rajoitustilavuus.

8.7. Yhden leikkauksen löytäminen törmäyskyselyillä

Oletetaan, että kontaktiparimatriisi sisältää ainoastaan yhden leikkauksen kahden kohteen primitiivien välillä. Oletetaan lisäksi, että kummankin kohteen ideaalinen rajoitustilavuushierarkia on täydellisesti tasapainoinen binaaripuu. Täydellisesti tasapainoinen binaaripuu tarkoittaa siis sitä, että kohteen rajoitustilavuushierarkiassa on $n=2^m$ primitiiviä, missä m on binaaripuun tasojen lukumäärä. Tällaisessa tapauksessa kontaktiparimatriisissa on n^2 solua ja yhden leikkauksen löytämiseen tarvitaan $2\log n^2+1$ törmäyskyselyä.

Törmäyskyselyiden lukumäärä voidaan todeta seuraavasti. Ensimmäinen törmäyskysely tuottaa koko kontaktiparimatriisin, jossa on n^2 solua. Toinen törmäyskysely jakaa kontaktiparimatriisin kahteen yhtä suureen osaan, joissa molemmissa on $n^2/2$ solua. Toinen osa sisältää leikkauksen ja toinen on tyhjä, eikä sitä tarvitse enää tutkia. Leikkauksen sisältämä kontaktiparimatriisin lohko jakautuu seuraavalla törmäyskyselyllä edelleen kahteen lohkoon, joissa on molemmissa $n^2/4$ solua. Leikkauksen löytämiseksi törmäyskyselyitä jatketaan, kunnes kontaktiparimatriisissa on jäljellä lohko, jossa on kaksi solua, joista toinen sisältää haettavan leikkauksen. Tällaisen kaksisoluisen lohkon löytämiseksi kontaktiparimatriisia täytyy jakaa $s=\log n^2$ kertaa. Jokainen kontaktiparimatriisin jako johtaa kahteen törmäyskyselyyn siten, että toinen lohkoista sisältää leikkauksen ja toinen ei, joten törmäystestien lukumääräksi saadaan $2\log n^2+1$. Tässä ykkösen lisääminen tarkoittaa ensimmäistä törmäyskyselyä, joka ei vielä jaa kontaktiparimatriisia osiin.

8.8. Alaraja törmäyskyselyiden lukumäärälle

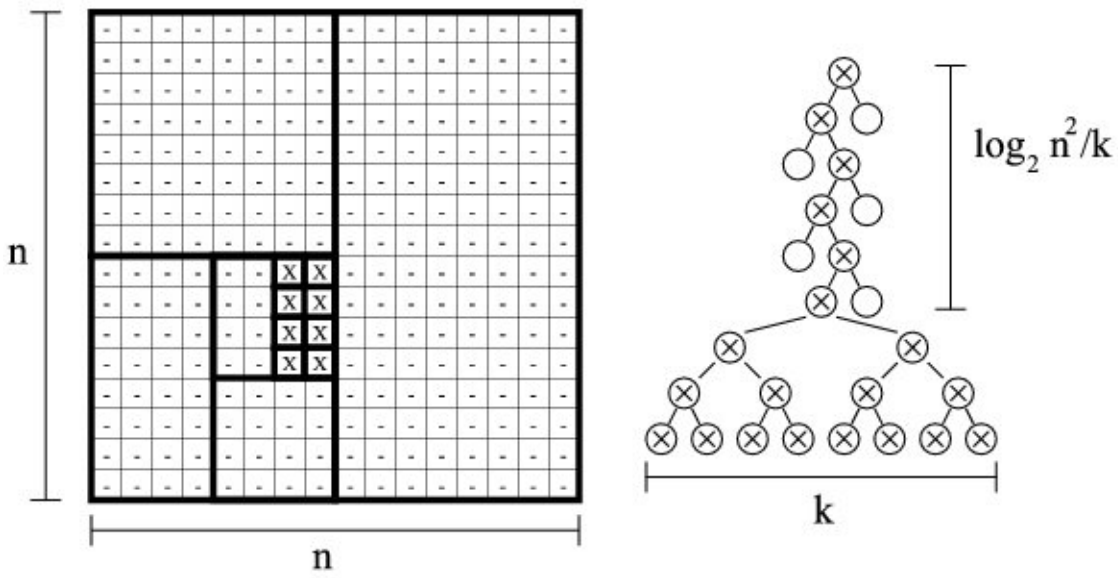
Oletetaan, että tutkittavana on kaksi täydellisesti tasapainoista ideaalista rajoitustilavuushierarkiaa, joissa on lehtitasoilla n primitiiviä. Jos lehtitason primitiivien välillä on $k>0$ leikkausta, niin alaraja törmäyskyselyille näiden leikkausten löytämiseksi on

$$a(n,k)=2\log(n^2/k)+2k-1.$$

Tämä voidaan todeta helposti kuvan 35 kontaktiparimatriisin avulla. Ihanteellisessa tapauksessa kohteiden primitiivien leikkaukset sisältyvät kaikki samaan lohkoon ja aluksi voidaan mahdollisimman pienellä törmäyskyselyiden määrällä sulkea kontaktiparimatriisin muut lohkot törmäyskyselyiden ulkopuolelle. Tässä tapauksessa kontaktiparimatriisin lohkomisten lukumääräksi saadaan $\log(n^2/k)$. Leikkaukset sisältävän lohkon eristämisen

jälkeen tarvitaan vielä k-1 lohkomista kaikkien erillisten leikkausten löytämiseksi. Tämä voidaan todeta kuvan 36 alaosan sisäsolmujen lukumäärän avulla. Koska jokainen lohkominen aiheuttaa kaksi törmäyskyselyä saadaan törmäyskyselyiden lukumäärän alarajaksi

$$a(n,k)=2\log(n^2/k)+2k\text{-}1.$$

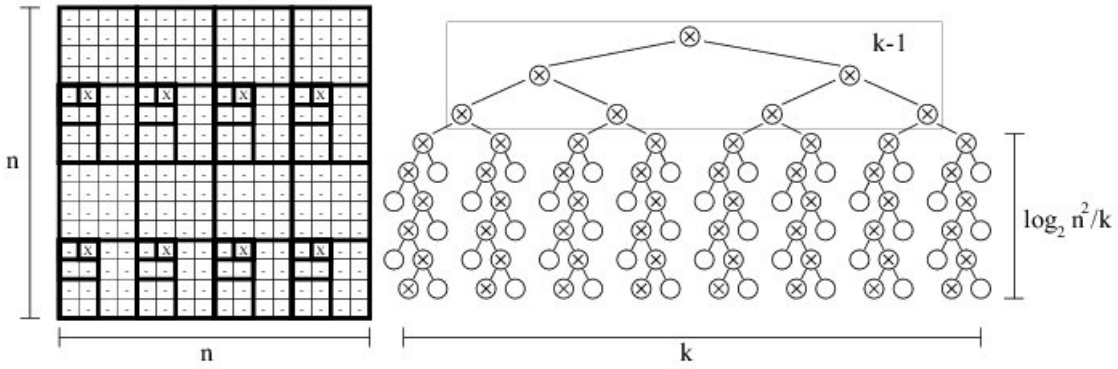


Kuva 36: Leikkaukset sisältyvät yhteen lohkoon.

8.9. Yläraja törmäyskyselyjen lukumäärälle

Tilanne on huonoin törmäyskyselyiden lukumäärän kannalta silloin, kun kaikki kohteiden väliset leikkaukset sijoittuvat kontaktiparimatriisissa omaan lohkoonsa. Tällainen tilanne on esitetty kuvassa 37. Tässä tilanteessa kontaktiparimatriisi sisältää k lohkoa, joiden eristämiseksi tarvitaan k-1 lohkomista. Näiden lohkojen sisältämien leikkausten löytämiseksi tarvitaan jokaiselle lohkolle $\log(n^2/k)$ lohkomista. Aikaisemmasta muistetaan, että jokainen lohkominen aiheuttaa kaksi törmäyskyselyä, joten törmäyskyselyjen ylärajaksi saadaan

$$a(n,k)=2k\log(n^2/k)+2k\text{-}1.$$



Kuva 37: Leikkausten sijoittuminen omiin erillisiin lohkoihin.

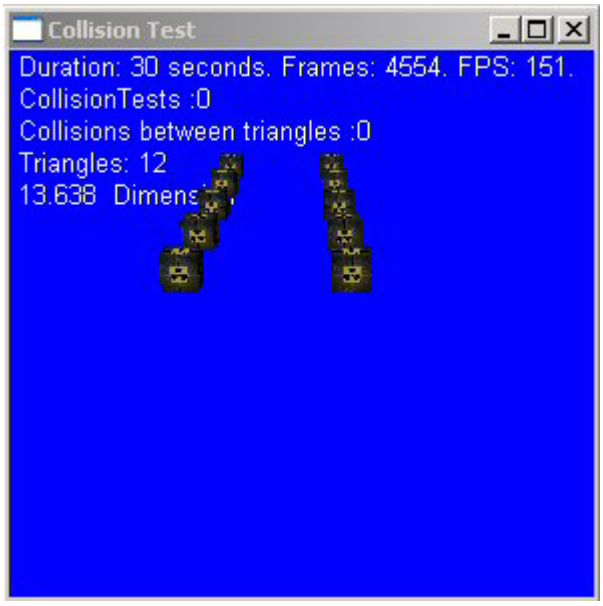
9. Testijärjestely

Seuraavaksi tarkastellaan testijärjestelyä, jonka tarkoituksena on tutkia törmäyskyselyiden suorittamisen vaikutusta ruudunpäivitysnopeuteen. Testiä varten on toteutettu c++-ohjelmointikielellä pieni ohjelma, johon voidaan ladata 3D Studio Max-ohjelmalla [9] tehtyjä kolmioverkkomalleja. Ohjelma muodostaa malleille kuution muotoiset rajoitustilavuudet siten, että mallit mahtuvat niihin mielivaltaisissa asennoissa. Tämän lisäksi mallien kolmioista muodostetaan rajoitustilavuushierarkiat, joiden solmut ovat kohteen mukaan suunnattuja rajoitustilavuuksia. Ohjelma siirtää ja kiertää kohteita mielivaltaisesti. Jokaisen siirron jälkeen tutkitaan kuutionmuotoisten rajoitustilavuuksien päällekkäisyys ja muodostetaan lista päällekkäisistä kohdepareista. Tämän jälkeen suoritetaan listassa oleville kohdepareille primitiivitasolle menevä törmäyskysely Rapid [26]-törmäyskyselykirjastolla. Ohjelmassa on toteutettu yksinkertainen visualisointi Microsoft DirectX 9.0 SDK:n [22] avulla.

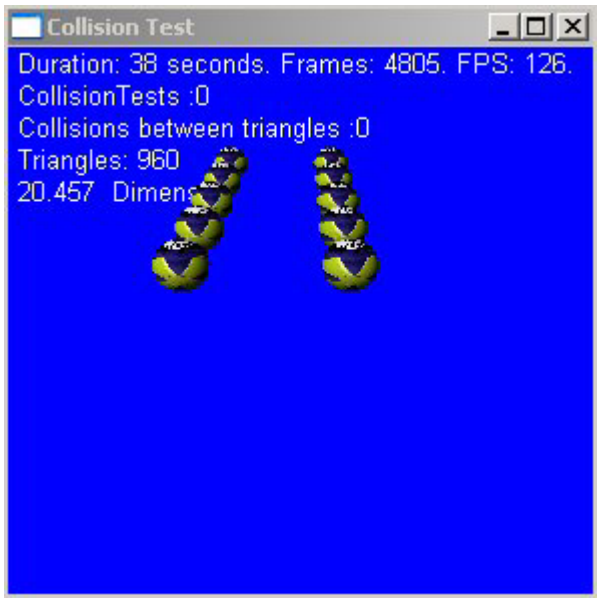
Testin suorittamiseksi ohjelmaan ladataan *n* kohdetta, joita liikutetaan mielivaltaisesti minuutin ajan. Ensin ohjelma suoritetaan ilman törmäyskyselyitä, jolloin mitataan keskimääräinen ruudunpäivitysnopeus. Tämän jälkeen ohjelma suoritetaan siten, että ruudunpäivitysnopeuden lisäksi lasketaan törmäyskyselyjen ja havaittujen primitiivien leikkausten lukumäärät. Jokainen testi suoritetaan kolme kertaa.

Ohjelmassa käytetään kahta erityyppistä mallia, jotka ovat kuutio ja pallo. Mallien kolmioiden lukumäärät ilmoitetaan testituloksien yhteydessä. Ensimmäinen testi suoritetaan siten, että siihen ladataan kymmenen mallia. Toisessa testissä käytetään kahtakymmentä mallia. Kuvissa 38 ja 39 on kuvakaappaukset kuution ja pallon testeistä kymmenen mallin tapauksessa.

Testit suoritetaan tietokoneella, jossa on AMD 2000+-suoritin, Geforce3-näytönohjain sekä 512MB muistia.



Kuva 38: Törmäystesti kuutioilla



Kuva 39: Törmäystesti palloilla

9.1. Testitulokset

Ensimmäisessä testissä käytetään kymmentä kuutiota, joissa jokaisessa on kaksitoista kolmiota. Testitulokset on ilmoitettu taulukossa 2. Taulukon 2 testinumero 0 tarkoittaa testiä ilman törmäyskyselyitä. Testit 1-3 ovat normaaleita testituloksia ja testinumero 4 tarkoittaa testien 1-3 keskiarvoja. Tämä merkintätapa on voimassa myös taulukoissa 3, 4 ja 5. Taulukoiden

viimeisessä sarakkeessa olevan ruudunpäivityksen yksikkö on animaatioruutujen lukumäärä sekunnissa.

Testinumero	Törmäystestit	Törmäykset	Ruudunpäivitys
0	0	0	99.3
1	62338	10534	95.6
2	27544	4432	96.6
3	66022	10198	96.3
4	51968	8388	96.2

Taulukko 2: Testitulokset kymmenelle kuutiolle, joissa 12 kolmioita.

Toisessa testissä käytetään kahtakymmentä kuutiota. Testitulokset ovat taulukossa 3.

Testinumero	Törmäystestit	Törmäykset	Ruudunpäivitys
0	0	0	99.2
1	105396	17900	87.1
2	125183	21598	87.2
3	153924	26523	87.1
4	128168	22007	87.1

Taulukko 3: Testitulokset kahdellekymmenelle kuutiolle, joissa 12 kolmiota.

Kolmannessa testissä käytetään kymmentä palloa. Testitulokset ovat taulukossa 4.

Testinumero	Törmäystestit	Törmäykset	Ruudunpäivitys
0	0	0	98.6
1	1382463	74985	89.7
2	933768	46940	91.4
3	555881	30859	93.2
4	957371	50928	91.4

Taulukko 4: Testitulokset kymmenelle pallolle, joissa 960 kolmiota.

Neljännessä testissä käytetään kahtakymmentä palloa. Testitulokset ovat taulukossa 5.

Testinumero	Törmäystestit	Törmäykset	Ruudunpäivitys
0	0	0	97.9
1	1689925	77454	69.0
2	1754957	87339	66.6
3	881769	337713	69.9
4	1442217	167502	68.5

Taulukko 5: Testitulokset kahdellekymmenelle pallolle, joissa 960 kolmiota.

Ensimmäisessä testissä törmäyskyselyiden aiheuttama ruudunpäivityksen pieneneminen on ainoastaan 3.1%, mutta neljännessä testissä vastaava luku on jo 30%. Tosin animaatiossa olevien kolmioiden lukumäärä on neljännessä testissä 160-kertainen ensimmäiseen testiin verrattuna.

Tämän pienen testin perusteella on helppo huomata, että nopeakin törmäysalgoritmi alkaa hidastaa animaatiota ratkaisevasti, jos kohteiden ja niissä olevien kolmioiden lukumäärä kasvaa suureksi. Tässä käytetyn menetelmän ensimmäisen vaiheen aikavaatimus on $O(n+k)$, missä n on animaatiossa mukana olevien kohteiden lukumäärä ja k on mahdollisesti törmäävien kohteiden lukumäärä. Jokaista k:n arvoa vastaa kaksi kohdetta, joille suoritetaan tarkka törmäystarkastelu niiden primitiivitasoille asti. Tämän vaiheen aikavaatimuksen ala- ja ylärajat esitettiin kohdissa 8.8 ja 8.9.

10. Lopuksi

Tässä tutkielmassa on tarkasteltu kolmioverkkoihin perustuvien kohteiden välisien törmäysten havaitsemista tietokoneanimaatiossa. Valitsin kolmioverkkotarkastelun siksi, että se on tällä hetkellä yleisin tapa esittää kohteita tietokonegrafiikassa.

Luvussa 9 kuvattu yksinkertainen testi tuottaa ainoastaan animaatiossa leikanneiden kolmioiden lukumäärät. Tällaisella tiedolla ei todellisessa animaatiossa ole kuitenkaan käyttöä. Pienellä muutostyöllä testissä voitaisiin luoda lista, joka sisältäisi ne pisteet, tai viivasegmentit, joissa kohteet törmäävät toisiinsa. Tätä listaa voitaisiin käyttää edelleen kohteiden välisessä fysiikkamallinnuksessa. Yleensä törmäysten etsinnässä riittää kuitenkin ensimmäisen törmäyskohdan löytäminen ja kohteiden liikkeiden muuttaminen tämän tiedon mukaan.

Animaatiosysteemi ei siis ole valmis törmäysten havaitsemisen jälkeen, vaan se vaatii kunnollisen fysiikkamallinnuksen, joka suoritetaan löydettyjen

törmäyskohtien perusteella. Näin ollen kohteiden liiketilojen muutoksien tutkiminen törmäyskohtien perusteella olisi luonnollinen jatko tälle tutkielmalle.

Verkko-osoitteesta Rapid [26] löytyy paljon tietoa ja valmiita törmäystarkastelukirjastoja asiasta kiinnostuneille. Näiden kirjastojen käytössä suurin vaikeus on kohteiden monikulmioverkon kulmapisteiden lataaminen käyttöä varten. Liitteessä 1 on esitetty yksi tapa kulmapisteiden lataamiseksi. Liitteessä 1 kuvatussa koodissa oletetaan, että malli on ladattu tiedostosta LPD3DDDXMESH-tyyppiseen osoittimeen. Tämän jälkeen mallissa olevat kulmapisteet saadaan käyttöön LPDIRECT3DVERTEXBUFFER9- ja LPDIRECT3DINDEXBUFFER9-rajapintojen avulla.

Viiteluettelo

[1] E. Angel, *Interactive Computer Graphics a Top-Down Approach With OpenGL*. Addison-Wesley 2002, third edition.

[2] J. Arenberg, Ray/triangle intersection with barycentric coordinates. In: *Ray Tracing news*, edited by Eric Haines, Vol. 1, No. **11**, November 4 1988.

[3] D. Badouel. An efficient ray-polygon intersection. In: *Graphics Gems*, edited by A. Glassner, Academic Press Inc, Boston 390-393, 1990.

[4] C. Barber, D. Dobkin, H. Huhdanpää, The quickhull algorithm for convex hull. Geometry Center Technical Report **GCG53**.

[5] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer-Verlag 1997.

[6] G. Van Den Bergen, Efficient collision detection of complex deformable models using AABB trees. Department of Mathematics and Computing Science Eindhoven, University of Technology.

[7] G. Bradshaw, C. O'Sullivan, Sphere-tree construction using medial axis approximation. *ACM SIGGRAPH*, Symposium of Computer Animation SCA 2002.

[8] J. Cohen, M. Lin, D. Manocha, M. Ponamgi, I-Collide: An interactive and exact collision detection system for large scale environments. Department of Computer Science, UNC Chapel Hill.

[9] Discreet 3D Studio Max. http://www.discreet.com/products/3dsmax/. Tarkastettu 4.8.2003.

[10] F. Dunn, I. Parberry, *3D Math Primer for Graphics and Game Development*. Wordware Publishing 2002.

[11] A. Fournier, D. Montuno. Triangulating simple polygons and eqvivalent problems. *ACM Transactions on Graphics*, 3:153-174, 1984.

[12] S.Gottschalk, Collision queries using oriented bounding boxes. Department of Computer Science UNC Chapel Hill, 2000.

[13] S.Gottschalk, M. Lin, D. Manocha. OBBTree a hierarchial structure for rapid interference detection. *Computer Graphics (Siggraph 96 Proceedings)*, pp. 171-180, August, 1996.

[14] E. Haines, Point in polygon strategies. In: *Graphics Gems IV*, edited by Paul S. Heckbert, AP Professional 1994, pp.24-46.

[15] E. Haines T. Möller, *Real-Time Rendering*. A K Peters Ltd, 1 edition 1999.

[16] P. Hubbard, Approximating polyhedra with spheres for time critical collision detection. *ACM Transactions on Graphics*, pp. 179-210, 1996.

[17] T. Hudson, M. Lin, J. Cohen, S. Gottschalk, D. Manocha, V-COLLIDE: Accelerated collision detection for VRLM. *Proceedings of VRLM 97*, Monterey, California, February 1997.

[18] Jacobi-Davidson method, http://www.cs.utk.edu/~dongarra/etemplates/node340.html. Tarkastettu 15.8.2003.

[19] J. Klosowski, M. Held, J Mitchell, H. Sowizral, K. Zikan, Efficient collision detection using bounding volume hierarchies of k-DOPs. Submitted for publication *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[20] E. Larsen, S. Gottschalk, M. Lin, D. Manocha, Fast proximity queries with swept sphere volumes. **TR99-018**. Department of Computer Science, UNC Chapel Hill.

[21] M. Lin, Efficient collision detection for animation and robotics, Ph.D thesis. University of California, Berkeley, 1993.

[22] Microsoft DirectX 9.0 SDK. http://msdn.microsoft.com/directx. Tarkastettu 16.8.2003,

[23] B. Mirtich, V-Clip: Fast and robust polyhedral collision detection. ACM *Transactions on Graphics*, pp. 177-208, July 1998.

[24] T. Möller, A fast triangle-triangle intersection test. *Journal of Graphics Tools* pp. 25-30, 1997.

[25] T. Möller, B. Trumbore, Fast, minimum storage ray/triangle Intersection. *Journal of Graphics Tools* pp. 21-28 1997.

[26] RAPID, Robust and Accurate Polygon Interference Detection. http://www.cs.unc.edu/geom/~collide/index.shtml. Tarkastettu 16.8.2003.

[27] M. Slater, A. Steed, Y. Chrysanthou, *Computer Graphics and Virtual Environments*. Addison-Wesley 2002.

[28] A. Watt, *3D Computer Graphics*. Addison-Wesley 2000, third edition.

Liite 1

```
struct VERTEX
{
 D3DXVECTOR3 p;
 D3DXVECTOR3 n;
 FLOAT tu;
 FLOAT tv;
};
#define D3DFVF_VERTEX (D3DFVF_XYZ|D3DFVF_NORMAL|D3DFVF_TEX1)
struct TRIANGLE
{
 D3DXVECTOR3 tri[3];
};
D3DXVECTOR3  v0,v1,v2;
LPD3DXMESH              pMesh;
LPDIRECT3DVERTEXBUFFER9 pVB;
LPDIRECT3DINDEXBUFFER9 pIB;
WORD*      pIndices;
VERTEX*    pVertices;
DWORD      dwNumFaces;
DWORD      dwNumVertices;
RAPID_model *pRmodel;

/*
Tähän   verkon   lataaminen   levyltä.   Löytyy   DirectX   SDK:n
tutoriaalista.
Ladataan pMesh osoittimeen.
*/

  pMesh->GetVertexBuffer(&pVB);
  pMesh->GetIndexBuffer(&pIB);

  pVB->Lock( 0,0,(void**)&pVertices, 0 );
  pIB->Lock(0,0,(void**)&pIndices, 0 );
  triangles =new TRIANGLE[dwNumFaces];
  TRIANGLE temp;
  pRmodel->BeginModel(); //Mallin lataaminen alkaa.
```

```
 for(DWORD i=0; i<dwNumFaces; i++)
   {
      v0 = pVertices[pIndices[3*i+0]].p;
      v1 = pVertices[pIndices[3*i+1]].p;
      v2 = pVertices[pIndices[3*i+2]].p;
 temp.tri[0]=v0;
 temp.tri[1]=v1;
 temp.tri[2]=v2;
 triangles[i]=temp;

                      vert1[0]=(double)v0.x;
                      vert1[1]=(double)v0.y;
                      vert1[2]=(double)v0.z;
                      vert2[0]=(double)v1.x;
                      vert2[1]=(double)v1.y;
                      vert2[2]=(double)v1.z;
                      vert3[0]=(double)v2.x;
                      vert3[1]=(double)v2.y;
                      vert3[2]=(double)v2.z;

      pRmodel->AddTri(vert1,vert2,vert3,id);//Kolmio malliin.
       }

       sprintf(buftris, "%d", numtriangles);
       pVB->Unlock();
       pIB->Unlock();
       pVB->Release();
       pIB->Release();

pRmodel->EndModel(); //Mallin lataaminen suoritettu.
  }
```