DYNAMIIKAN SIMULOINTI 3D-ANIMAATIOSSA

LAHDEN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto
Opinnäytetyö
14.5.2007
Ville Laine

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma

LAINE, VILLE: Dynamiikan simulointi 3d-animaatiossa

Teknisen visualisoinnin opinnäytetyö, 50 sivua

Kevät 2007

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee fysiikanmallinnuksen hyödyntämistä kolmiulotteisen tietokoneanimaation toteutuksessa.

Tietokoneanimaatio soveltuu hyvin sellaisten tapahtumien visualisointiin, joiden kuvaaminen olisi muilla keinoin hankalaa. Yleisin tekniikka tietokoneanimaation toteuttamiseksi on määrittää liikkeiden äärirajoja kuvaavat avainkehykset aikajanalle. Avainkehysten perusteella suorittaa tietokone näiden rajojen puitteissa tapahtuvan liikkeen. Tällainen key frame -tekniikalla toteutettu animaatio perustuu aina animaattorin näkemykseen liikkeestä.

Mikäli kuitenkin lopputulokselta odotetaan tarkkuutta ja todenmukaisuutta, ei käsin toteutettu animaatio useinkaan täytä vaatimuksia. Animointi voidaan toteuttaa dynamiikkasimulaation avulla. Tällöin animointiin tarvitaan fysiikkamoottori. Fysiikkamoottori on tietokoneohjelma, jonka algoritmien perusteella tietokoneen suoritin laskee oikeanlaiset liikkeet mallinnetuille kappaleille niiden fysikaalisten ominaisuuksien perusteella. Myös paljon animoitavia kappaleita käsittävien kokonaisuuksien hallitseminen helpottuu, koska jokaista kappaletta ei tarvitse animoida erikseen.

Fysiikkamoottorin tehtävät jakautuvat törmäystarkasteluun, systeemin päivittämiseen sekä muutosten välittämiseen rajapinnalle. Törmäystarkastelussa tutkitaan simulaatioon kuuluvien kappaleiden pintoja ja tutkitaan, milloin kappaleet osuvat toisiinsa. Kun kappaleiden pintojen välillä havaitaan törmäys, lasketaan annettujen fysikaalisten ominaisuuksien perusteella kappaleiden tilassa tapahtuvat muutokset. Tämän jälkeen välitetään systeemissä tapahtuneet muutokset käytettävälle rajapinnalle, jolloin animaation tapauksessa muodostetaan kappaleiden avainkehykset aikajanalle.

Koska suurin osa dynamiikkasimulaation vaatimasta laskentakapasiteetista käytetään törmäystarkasteluun, on tämän työvaiheen yksinkertaistamisella suuri merkitys koko prosessin sujuvuuden kannalta. Törmäystarkastelun helpottamiseksi voidaan tarkasteltavia muotoja pelkistää ja rajata tarkastelun ulkopuolelle epätodennäköiset törmäysparit.

Lopullisen animaation tarkkuuden ratkaisee fysiikkamoottorin suorittamien systeemin päivitysten määrä aikayksikköä kohden. Mitä suurempia ovat animoitavien kappaleiden nopeudet, sitä useammin on systeemi päivitettävä.

Avainsanat: animaatio, fysiikkamoottori, törmäystarkastelu, dynamiikan simulointi, Reactor

Lahti University of Applied Sciences
Faculty of Technology

LAINE, VILLE: Dynamics simulation in 3d-animation.

Bachelor's thesis in Visualization Engineering, 50 pages

Spring 2007

ABSTRACT

This thesis deals with the use of dynamics simulation in the 3d animation process.

Computer animation is a useful way to visualize events that could be difficult to describe by other methods. In computer animation extreme positions are set for each object and the motions between these points are produced automatically. The most common way to set extreme positions of motion is to define the specific animation frames on the timeline. This technique is called key frame animation and it is based on the animator's idea of correct movement.

Sometimes more realism is required from the motion. Adding dynamics simulation to the animation makes it possible to create accurate and realistic motion based on the physical laws. Dynamics simulation is carried out by a physics engine, which calculates movements of the objects based on physical properties defined for them. This is a relatively quick and simple way to create accurate lifelike animation. It also helps to manage large scenes with a lot of animated objects because there is no need to carry out keyframing procedures for every single object.

A physics engine has three main tasks. They are collision detection, updating the system and reporting updates to the user. In collision detection the physics engine observes movements of objects and detects if any of the objects are colliding. When collision is detected between objects, the system is updated. That means that the right movements for objects are calculated after collision. After this, updates are transmitted to the interface and they are displayed to the user.

Collision detection takes most of the time used in dynamics simulation. To make the whole process as smooth as possible, simplified models can be used as the geometry of the object. Unnecessary collision pairs can be left out of the simulation. It means that collision detection is not carried out between objects that are not likely to be colliding.

The accuracy of the final animation is defined by how many times the system is updated during a specific time period. If update frequency is too low, objects may interpenetrate each other on collisions. Another thing that has an influence on accuracy is collision tolerance. If collision tolerance is too high, visible spaces will occur between objects when surfaces should be in contact. However, lowering collision tolerance will make simulation heavier to compute.

Keywords: animation, physics engine, collision detection, dynamics simulation, Reactor

SISÄLLYS

1 JOHDANTO

Kolmiulotteisen tietokoneanimaation käyttö erilaisiin tarkoituksiin on nykyisin hyvin arkipäiväistä. Animaation käyttö tuotteiden markkinoinnissa, havainnollistamisvälineenä, sekä erilaisissa viihdetarkoituksissa on koko ajan yleisempää. Tietotekniikan kehittymisen mukanaan tuoma laskentatehojen kasvu mahdollistaa yhä todenmukaisempien suurta laskentakapasiteettia vaativien 3d-animmaatioiden laajamittaisen käytön.

Toteutettaessa tietokoneanimaatiota on ensin mallinnettava näyttämö siihen kuuluvine kohteineen, valaistuksineen sekä materiaaleineen ja lopuksi animoitava näyttämön tapahtumat. Lisäksi saattaa laajojen kokonaisuuksien hallitseminen muodostua ongelmalliseksi, mikäli joudutaan animoimaan suuria määriä kappaleita. Animaation toteuttaminen voi siis olla hyvinkin paljon aikaa vievä kokonaisuus.

Animaatioprosessin nopeuttamiseksi on joitakin työvaiheita voitava automatisoida. Fysiikan mallinnuksen avulla voidaan toteuttaa ohjelmallisesti animaatiota, joka perustuu kappaleille määriteltäviin ominaisuuksiin.

Tässä opinnäytetyössä tutkitaan fysiikan mallinnuksen eli dynamiikkasimulaation hyödyntämistä 3d-animaaton toteuttamisessa. Tarkoituksena on selvittää fysiikan mallinnuksen periaate ja fysiikan mallinnusta hyödyntävän animaation toteuttamisprosessi sekä se, mitä etua kyseisellä menetelmällä saavutetaan. Työssä tullaan käyttämään 3ds Max 8 –mallinnusohjelmaa sekä siihen kuuluvaa Reactor-fysiikkatyökalua.

Aluksi tehdään katsaus tietokoneanimaatioon yleisellä tasolla, määritellään mitä on dynamiikka-animaatio, sekä selvitetään fysiikan mallinnuksen periaate. Seuraavaksi tutkitaan 3ds Maxin ja Reactorin avulla dynamiikkasimulaation toteuttamista ja sitä, millaisia vaatimuksia se asettaa kappaleiden mallintamiselle. Tutkitaan myös, miten määritellään simulaatioon kuuluvien kappaleiden ominaisuudet ja kappaleiden välillä vaikuttavat voimat sekä, miten liikkeitä rajoitetaan. Lopuksi selvitetään dynamiikkasimulaatioon perustuvan animaation toteuttaminen edellä mainitun ohjelman avulla sekä, käytetyn animointimenetelmän edut ja puutteet.

2 ANIMAATIO

2.1 Käsite

Animaatio-termi pohjautuu latinan kielen sielua tarkoittavaan sanaan anima. Animaatio tarkoittaa elollistamista tai eläväksi tekemistä. (Mäkelä & Tiainen 1993, 54.) Animaatio perustuu jälkikuvaksi kutsuttuun näköhavaintoon, jossa silmän verkkokalvo reagoi pienellä viiveellä siihen kohdistuviin valoärsykkeisiin, jolloin valoaistimus havaitaan verkkokalvolla hetken aikaa vielä ärsykkeen jälkeen. Näin ollen nopeassa tahdissa peräkkäin esitetyt yksittäiset kuvat silmät ja aivot tulkitsevat tasaisesti liikkuvaksi kuvaksi. Jotta liikkuvan kuvan aistimus saataisiin aikaiseksi, on kuvia esitettävä vähintään 12 kappaletta sekunnissa. Tällainen kuva näyttää vielä jonkin verran nykivältä, ja standardiksi onkin muodostunut 24 kuvaa sekunnissa. Ylitettäessä 70 kuvaa sekunnissa ei ihmissilmä huomaa minkäänlaista parannusta kuvan laadussa.

Ensimmäiset elokuvat olivat animaatioita. Ranskalainen Emile Reynaud esitti 1890-luvulla animaatioita kehittelemällään laitteella, jossa peräkkäisiä kuvia heijastettiin pyöritettävältä rullalta ja myöhemmin filmiltä valkokankaalle (Lehtinen 2007). Filmiprojektorilla esitetyt piirrosanimaatiot alkoivat yleistyä 1910-luvulla. Alun perin animaatio, kuten muukin elokuva, oli suunnattu puhtaasti viihdetarkoituksiin. Pian kuitenkin huomattiin animaation soveltuvan erinomaisesti havainnollistamiskeinoksi. Animaatio on usein helpoin tai jopa ainoa tapa visualisoida jokin tapahtuma tai tapahtumasarja.

2.2 Tietokoneavusteisen suunnittelun historiaa

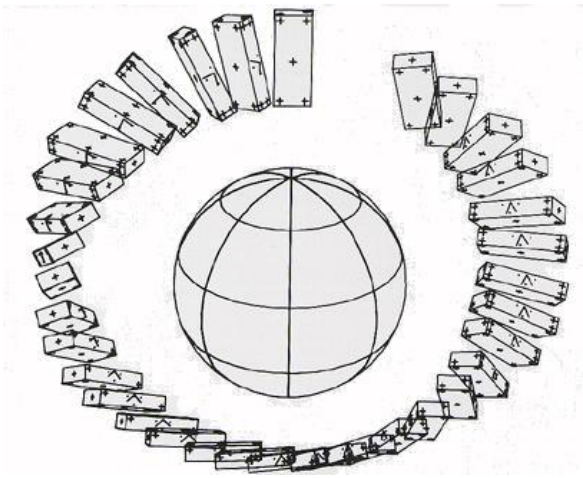
Ensimmäinen tietokoneavusteiseen suunnitteluun tarkoitetut ohjelmat otettiin käyttöön 1950- ja 1960-lukujen vaihteessa. Tuolloin alan kehityksestä vastasivat suuryritykset, joilla oli käytössään riittävän tehokkaita tietokoneita.

Ensimmäisenä varteenotettavana CAD-ohjelmana pidetään yleisesti DAC-1-nimistä järjestelmää, jonka kehitystyön IBM ja General Motors aloittivat vuonna 1959 (Carlson 2003a). Ohjelmaa käytettiin autojen suunnittelussa, ja se mahdollisti kolmiulotteisten mallien tarkastelun eri suunnista. Vielä tässä vaiheessa malli syötettiin tietokoneelle numeerisina parametreina.
Seuraava merkittävä innovaatio olikin Ivan Sutherlandin vuonna 1961 kehittämä ensimmäinen interaktiivinen piirto-ohjelma nimeltään Sketchpad (Carlson 2003b). Siinä voitiin eräänlaisella valokynällä piir-

tää kuvaputkinäytölle kuvioita, jotka tallennettiin tietokoneen muistiin (kuva 1). Kyseistä ohjelmaa pidetään prototyyppinä kaikille nykyisinkin käytössä oleville graafisille käyttöliittymille.

Kuva 1. Ivan Sutherland kehittämänsä Sketchpad-ohjelman parissa vuonna 1963. (kuva: Carlson 2003a.)

Merkittävä askel tietokoneanimaation saralla otettiin vuonna 1961, kun Bell Telephone Laboratory yhtiössä toteutettiin animaatio, joka kuvasi satelliittia Maan kiertoradalla (kuva 2)(Carlson 2003c).

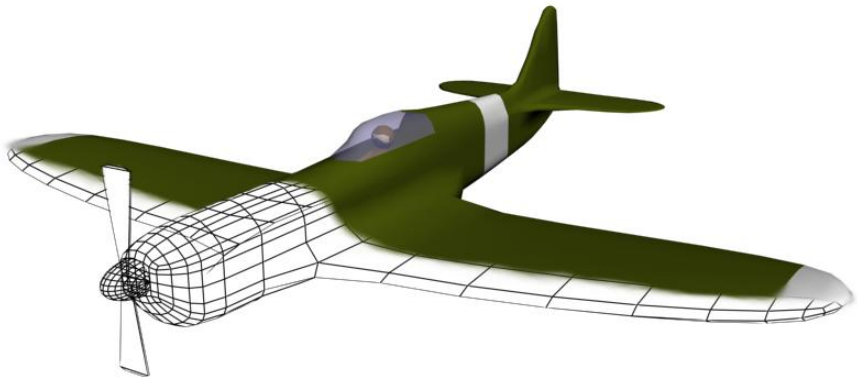


Kuva 2. Tunnetuin ensimmäisistä tietokoneella toteutetuista animaatioista oli nimeltään *A two gyro gravity gradient altitude control system.* Animaatiolla demonstroitiin satelliitin asennon hallintaa Maan kiertoradalla. (kuva: Carlson 2003b.)

Tutkijat alkoivat laajemmaltikin käyttää tietokoneanimaatiota tutkimustensa havainnollistamiseen. Yhä tehokkaampien tietokoneiden tullessa markkinoille 1960-luvun aikana kiinnostuivat suuret teollisuusyritykset tietokoneanimaation mahdollisuuksista suunnittelu- ja tutkimusvälineenä. Mikroprosessorin keksiminen 1971 mahdollisti tehokkaammat tietokoneet ja siten myös näyttävämmän kolmiulotteisen tietokonegrafiikan.

2.3 3d-animaatio

3D-grafiikka eli kolmiulotteinen grafiikka on tietokonegrafiikkaa, joka on mallinnettu kolmen tilaulottuvuuden suhteen. Kolmiulotteinen grafiikka on tyypillisesti vektorigrafiikkaa, jossa muodot koostuvat monikulmioista (kuva 3). Perusmuotona on kolmio, joka on yksinkertaisin muoto, jolle voidaan määrittää pinta. (Franklin 2007.)



Kuva 3. Kolmiulotteinen grafiikka muodostuu monikulmioiden päälle muodostetuista pinnoista. (Laine 2007.)

Kuvitteellista ääretöntä tilaa, jossa kappaleet esimerkiksi mallinnusohjelmassa ovat, kutsutaan näyttämöksi (engl. scene). Kolmiulotteisen grafiikan automaattista piirtoa sanotaan *3D-renderöinniksi*. Renderöintitekniikoita on monenlaisia, mutta tämän työn kannalta niitä ei ole aiheellista tutkia kovin syvällisesti. Lyhyesti voidaan kuitenkin mainita, että reaaliaikaisessa animaatiossa on piirtoalgoritmien oltava yksinkertaisia normaalisti käytössä olevien tietokoneiden rajallisen laskentaka-

pasiteetin vuoksi. Pyrittäessä korkeatasoiseen lopputulokseen voidaan yhden animaatioruudun renderöintiin käyttää jopa tunteja, riippuen näyttämön monimutkaisuudesta ja käytettävissä olevasta laskentatehosta. Nykyiset 3d-mallinnusohjelmat mahdollistavat myös animaation luomisen, joten tavanomaisella kotitietokoneellakin on mahdollista tuottaa varsin näyttäviä animaatioita.

2.4 Animointitekniikat

2.4.1 Key frame -animaatio

Kuten aiemmin todettiin, perustuu tietokoneanimaatio nopeassa tahdissa näytettäviin yksittäisiin kuviin, jotka tulkitaan yhtenäiseksi liikkeeksi. Kuvien esitysnopeus ilmaisee, kuinka monta kuvaa esitetään yhden sekunnin aikana. Tietokoneanimaation esitysnopeus voidaan määrittää halutuksi, mutta standardiksi on muodostunut elokuva- ja televisiomaailmassa 24 kuvaa sekunnissa.

Animaatio-ohjelmissa on aikajana, jonka muodostavat peräkkäin olevat yksittäiset ruudut eli kehykset. Animointi tapahtuu aikajanalle sijoitettavien avainkehysten (eng. key frame) avulla. Animoitavan kappaleen liikkeelle määritetään avainkehyksiä, joiden välillä tapahtuvan liikkeen ohjelma toteuttaa.

Esimerkiksi paikallaan pomppivaa palloa varten tulee määrittää liikkeen ääripäät, siis ylin ja alin kohta. Tällä tavalla toteutettuna liike ei kuitenkaan ole vielä kovin todenmukainen, koska pallo liikkuu edestakaisin vakionopeudella. Pallon nopeus kuitenkin hidastuu, ennen kuin se lähtee putoamaan alaspäin kiihdyttäen vauhtiaan. Nopeuden muutos voidaan toteuttaa lisäämällä avainruutuja.

Tämä key frame -tekniikaksi kutsuttu tapa on kuitenkin työläs ja aikaa vievä, joten animaatio-ohjelmissa on työkaluja, joiden avulla avainruutujen välillä tapahtuvien liikkeiden nopeutta ja määrää voidaan säätää helpommin. Esimerkiksi 3ds Max -ohjelmassa tämä tapahtuu Track View -näkymässä, jossa liikettä säädetään käyrien avulla sekä voidaan tarvittaessa lisätä avainruutuja. Tällä tekniikalla tapahtuva animointi perustuu ainoastaan animaattorin näkemykseen sekä tämän kykyyn jäljitellä todenmukaista liikettä.

2.4.2 Dynamiikka-animaatio

Mahdollisimman realistiseen tulokseen päästään käyttämällä animaatiossa dynamiikkaa. Tämä tarkoittaa, että liikkeen animointi tapahtuu ohjelmallisesti ja perustuu mallinnetuille kappaleille annettuihin fysikaalisiin ominaisuuksiin. Tällä tavalla on mahdollista toteuttaa erittäin todenmukaista liikettä yksinkertaisesti ja nopeasti.

Fysikaalisia ominaisuuksia omaavien kappaleiden muodostamaa kokonaisuutta kutsutaan simulaatioksi. Kaikkien näyttämöllä olevien kappaleiden ei tarvitse kuulua simulaatioon, vaan myös key frame -animaatiota voidaan käyttää samanaikaisesti.

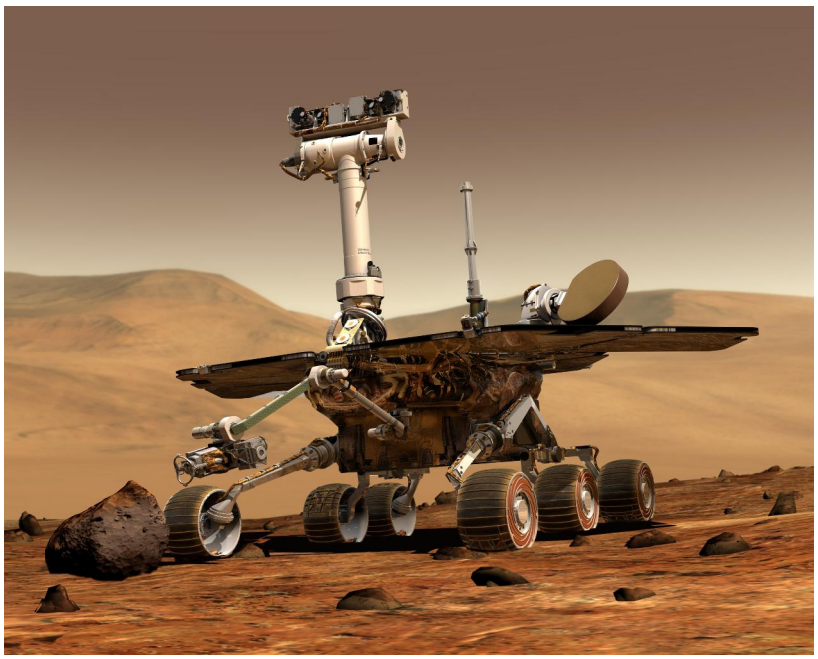
Todenmukaisen liikkeen lisäksi dynamiikka 3d-animaatiossa mahdollistaa suuria määriä kappaleita sisältävien näyttämöiden hallitsemisen. Jokaisen kappaleen liikkeitä ei tarvitse määritellä erikseen, vaan kappaleille määritetään fyysiset ominaisuudet ja annetaan tietokoneen suorittaa animointi.

Lopuksi tietokone asettaa automaattisesti jokaisen kappaleen liikkeitä kuvaavat avainkehykset aikajanalle. Tällä tavalla saavutetaan huomattava ajan säästö, sekä voidaan toteuttaa animaatioita, joiden tekeminen key frame -menetelmällä ei olisi järkevää valtavan työmäärän takia.

2.5 Soveltamiskohteet

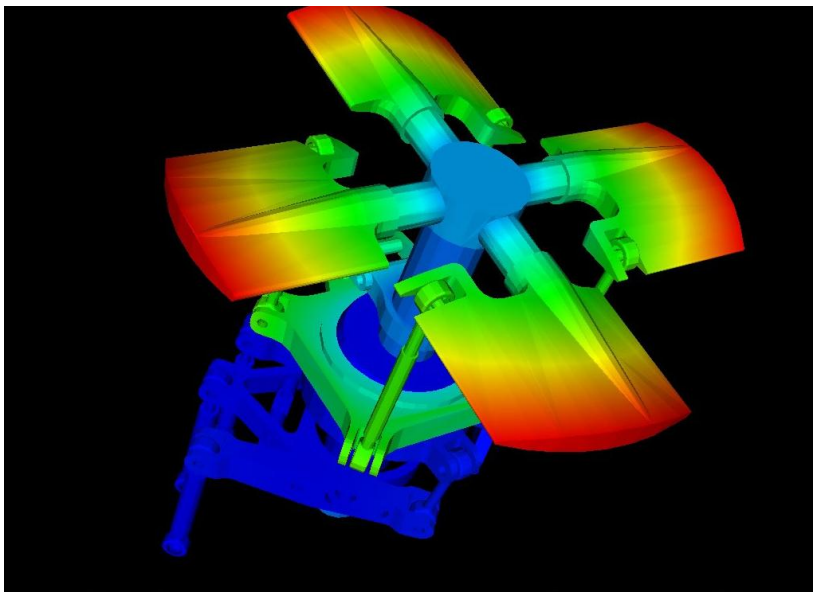
Fysiikkamallinnusta voidaan animaatiossa hyödyntää hyvin laajasti. Kolmiulotteisen tietokonegrafiikan kehityksen alkuvaiheissa on dynaaminen animaatio ollut merkittävässä osassa. Suurimpana motivaationa 3d-grafiikan kehittämiselle alun perin oli tutkijoiden ja insinöörien pyrkimys hyödyntää tietokoneanimaatioita tutkimuksissaan. Alkeellisenkin vektorigrafiikan avulla oli mahdollista toteuttaa 3d-animaatiota, jotka sopivat erinomaisesti havainnollistamaan tapahtumia.

3d-animaation käyttö juuri havainnollistamiseen onkin koko ajan suositumpaa. Dynaamisen animaation avulla voidaan simuloida ja rekonstruoida tapahtumia, joiden kuvaaminen muuten olisi toteutettavissa vaikeasti tai se olisi jopa mahdotonta (kuva 4). Atomitasolla tapahtuvien reaktioiden tai antiikin maanjäristyksen visualisointi on mahdollista vain animaation avulla.



Kuva 4. Tietokoneanimaation avulla voidaan visualisoida muuten hankalasti kuvattavia tapahtumia. (Caltech/JPL 2007.)

Erilaisten tapahtumasarjojen kulun selvittäminen onnistuu dynaamisen animaation avulla, jolloin erilaisten muuttujien vaikutukset lopputilanteeseen on todennettavissa. Esimerkiksi onnettomuustutkinnassa voidaan tehtyjen laskelmien tuloksia tarkastella animaation avulla. Paitsi havaintovälineenä, voidaan animaatiota käyttää myös suunnitteluvaiheessa (kuva 5).



Kuva 5. Fysiikan mallinnusta voidaan hyödyntää esimerkiksi koneiden suunnittelussa. (MSC.Software 2005.)

Erilaisia rakenteita ja rakennuksia suunniteltaessa voidaan tutkia esimerkiksi millaisia voimia kyseiset rakenteet kestävät. Tällaisten simulaatioiden toteuttaminen vaatii kuitenkin valtaisan määrän laskentaa, koska fysiikkamallinnuksen on oltava erittäin tarkkaa luotettavan lopputuloksen aikaansaamiseksi. Näin vaativaan käyttöön soveltuvat ohjelmat eivät luonnollisestikaan ole normaalisti markkinoilta saatavia tuotteita, vaan usein niitä käyttävän yhtiön tai laitoksen juuri tähän tehtävään kehittämiä ohjelmia.

Viihdekäytössä tullaan toimeen huomattavasti kevyemmällä fysiikkamallinnuksella, koska pelkästään uskottavalta näyttävä animaatio riittää. Dynaaminen animaatio soveltuu hyvin elokuvateollisuuden käyttöön, koska se mahdollistaa todenmukaisten erikoistehosteiden toteuttamisen suhteellisen helposti.

On myös muistettava, että 3d-animaatiossa käytettävä dynamiikka ei ole sidottua maapallolla vaikuttaviin fyysisiin voimiin. Simulaation parametreja muokkaamalla voidaan aikaansaada vaikkapa Kuussa vallitsevat olosuhteet.

Dynamiikkaa hyödyntävän 3d-animaation hyöty- ja viihdekäytön välimuotona voidaan pitää dokumenttielokuvia, joissa tällaisten animaatioiden käyttö on koko ajan yleisempää. Lähinnä viihde- ja havainnollistamiskäyttöön soveltuvien animaatioiden toteuttaminen on mahdollista yleisesti kaupan olevilla 3d-mallinnusohjelmilla sekä kotitietokoneilla.

3 VOIMAT JA NIIDEN VAIKUTUKSET

3.1 Mekaniikka

Koska dynamiikka-animaatio perustuu kappaleiden välisiin vuorovaikutuksiin sekä kappaleisiin vaikuttaviin voimiin, tarvitaan fysiikan mallinnusta. Jotta ymmärrettäisiin dynamiikkaa hyödyntävän animaation ja fysiikan mallinnuksen periaatteita, on hyvä hieman tutkia, mitkä lainalaisuudet ympäröivässä maailmassa vaikuttavat.

Mekaniikka käsittelee kappaleiden liikkeitä sekä näiden liikkeiden syitä. Mekaniikan perustana toimivat fyysikko Isaac Newtonin 1687 julkaisemat mekaniikan peruslait, eli Newtonin lait. Dynamiikka on mekaniikan haara, joka tutkii kappaleiden välisistä vuorovaikutuksista aiheutuvia voimia ja niiden vaikutusta kappaleiden liikkeeseen. Dynamiikka, jota usein myös kinetiikaksi kutsutaan, edellyttää kinematiikkaa. Kinematiikka puolestaan ei ota huomioon liikkeen syitä. (Salmi 1996, 11.)

Kinematiikassa tutkitaan kappaleen liikettä. Kappaletta kuvataan partikkelilla eli massapisteellä. Se ei ole todellinen hiukkanen, vaan pelkkä matemaattinen malli, jonka avulla voimien vaikutuksia voidaan tutkia. Sekaannusten välttämiseksi lienee syytä korostaa, että tässä yhteydessä termillä partikkeli ei ole mitään tekemistä 3d-animaatiossa esimerkiksi efektien luomiseen käytettävien partikkelien kanssa.

3.2 Kinematiikka

Kinematiikan peruskäsite on keskivauhti, joka saadaan jakamalla kappaleen kulkema matka ajalla.

$$v_k = \frac{\Delta s}{\Delta t}$$

Koska luonnossa tasaisella, muuttumattomalla nopeudella tapahtuvaa liikettä harvoin esiintyy, on kappaleella kiihtyvyys. Keskikiihtyvyys saadaan jakamalla nopeudenmuutos siihen kuluneella ajalla.

$$a_k = \frac{v - v_0}{t}$$

3.3 Dynamiikka

Newtonin I laki eli jatkuvuuden laki määrittelee, että lepotilassa oleva kappale pyrkii jatkamaan lepotilaansa ja liiketilassa oleva pyrkii jatkamaan liiketilaansa. Tätä massan liiketilan muutoksia vastustavaa ominaisuutta kutsutaan massan hitaudeksi. (Ahoranta & Helakorpi 1999, 47.)

Newtonin II lain eli liikelain mukaan kokonaisvoima F antaa kappaleelle, jonka massa on m kiihtyvyyden a.

$$\overline{F} = m\overline{a}$$

Kaikkiin kappaleisiin vaikuttaa vähintäänkin painovoima. Koska vapaassa putoamisliikkeessä on kysymyksessä alaspäin suuntautunut kiihtyvyys, painovoiman suuruus saadaan seuraavasta lausekkeesta massan ja putoamiskiihtyvyyden tulona

$$G = mg \text{ .}$$

Putoamiskiihtyvyyden likiarvona käytetään arvoa 9,81 m/s^2 .

Tasolla olevaan kappaleeseen vaikuttaa myös liikettä vastustava kitkavoima. Tasolla liukuvaan kappaleeseen vaikuttaa liukukitka. Kitkavoiman suuruus lasketaan yhtälöstä

$$F_\mu = \mu N \text{ .}$$

Lausekkeessa μ on kitkakerroin ja N normaalivoima, joka on pintaa vastaan kohtisuora.

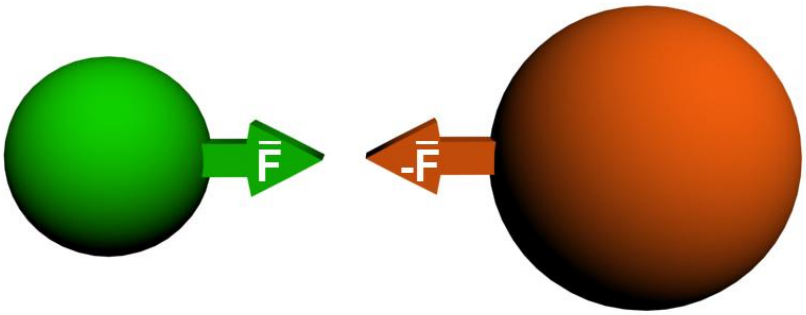
Kitkakertoimet ovat kokeellisesti todettuja kahden pinnaan välisiä arvoja, jotka ilmoitetaan kirjallisuudessa likiarvoina. Koska kitkavoima johtuu pintojen epätasaisuuksista, vaikuttaa kitkavoiman suuruuteen myös pintojen kovuus. Vierivään kappaleeseen vaikuttaa liukukitkaa pienempi vierimiskitka. Johtuen edellä kuvatusta massan hitaudesta, tarvitaan levossa olevaan kappaleeseen vaikuttavan lepokitkan kumoamiseksi suurempi voima kuin kappaleen liikkeen ylläpitämiseen. Kitkavoima on tason suuntainen ja päinvastainen kappaletta liikuttavalle voimalle. (Ahoranta & Helakorpi 1999, 57.)

Kaikkien kappaleiden liikettä vastustaa myös väliaineen vastus. Väliaineen vastus riippuu kappaleen nopeudesta ja muodosta sekä väliai-

neesta. Väliaineen vastus on suoraan verrannollinen kappaleen nopeuteen. Erityisesti suurilla nopeuksilla vastusvoima on suhteessa nopeuden neliöön. (Salonen 1998, 56 - 57.)

Newtonin III laki eli voiman ja vastavoiman laki on törmäystarkastelun kannalta kaikkein keskeisin asia. Kahden kappaleen vaikuttaessa toisiinsa ovat niiden väliset voimat massoista riippumatta yhtä suuria, mutta vastakkaissuuntaisia (kuva 6). Tämä tarkoittaa sitä, että myös törmäystilanteessa kappaleet vaikuttavat toisiinsa yhtä suurilla voimilla. Törmäys tilanteessa voiman vaikutusta kuvaa voiman impulssi I, joka saadaan voiman F ja sen vaikutusajan Δt tulona yhtälöstä

$$\bar{I} = \bar{F}\Delta t .$$



Kuva 6. Voima F ja sen vastavoima –F. (Laine 2007.)

Kun tiedetään kappaleen massa ja nopeus, saadaan niiden tulona liikemäärä p.

$$\bar{p} = m\bar{v}$$

Yhdistämällä keskikiihtyvyyden ja Newtonin II liikelain yhtälöt voiman impulssin yhtälöön saadaan kappaleeseen vaikuttavan ulkoisen voiman impulssi, joka on yhtä suuri kuin kappaleen liikemäärän muutos.

$$\bar{I} = m\bar{v} - m\bar{v}_0$$

Törmäyksessä kokonaisliikemäärä säilyy, koska molemmat kappaleet saavat yhtä suuren impulssin, kuten aikaisemmin todettiin. Toinen kappale menettää liikemääräänsä yhtä paljon kuin toisen liikemäärä li-

sääntyy. Liikemäärän säilymisen lain mukaisesti systeemin kokonaisliikemäärä säilyy, jos systeemiin ei vaikuta ulkoisia voimia.

3.4 Jäykän kappaleen dynamiikka

Jäykän kappaleen dynamiikka eroaa edellä kuvatusta partikkelin dynamiikasta. Se tutkii pyörivän kappaleen liiketilan muutoksia aiheuttavia voimia (Hautala & Peltonen 2002, 62). Koska jäykän kappaleen käyttäytymisen kuvaamiseen ei riitä yksittäinen partikkeli, sitä voidaan kuvata useamman partikkelin systeemiksi. Jäykkä kappale ei muuta törmäyksessä muotoaan, joten partikkelien etäisyydet pysyvät samoina. Partikkelisysteemissä voidaan määrittää kappaleen painopiste, toisin kuin yksittäisen partikkelin tapauksessa. Jäykän kappaleen avulla ei voida määrittää kappaleen sisäisiä jännitteitä, joten sen avulla ei voida tutkia materiaaliominaisuuksia. (Salonen 1998, 53.)

4 DYNAMIIKKA-ANIMAATION PERIAATE

4.1 Fysiikkamoottori

Dynamiikkaa hyödyntävän animaation toteuttamiseksi on käytettävä ohjelmaa, joka pystyy mallintamaan fysiikkaa. Fysiikan mallintamisen suorittaa ohjelmaan integroitu erityinen fysiikkamoottori. Fysiikkamoottori on sovellus, jonka algoritmien mukaisesti tietokoneen suoritin laskee animaatiossa tapahtuvat liikkeet kappaleille annettujen ominaisuuksien perusteella.

Fysiikkamoottorin ominaisuudet määräytyvät sen käyttötarkoituksen mukaan. Pelikäytössä fysiikan mallinnuksen on oltava reaaliaikaista, joten joitakin toimintoja joudutaan yksinkertaistamaan. Siksi pelikäyttöön tarkoitetun fysiikkamoottorin tarkkuus ei voi saavuttaa samaa tasoa, kuin animointiin tarkoitetun.

Fysiikan mallintaminen voidaan suorittaa myös erillisessä fysiikkaprosessorissa kuormittamatta tietokoneen varsinaista suoritinta. Tästä on hyötyä etenkin pelikäytössä, jolloin tietokoneella saattaa olla useita samanaikaisia tehtäviä.

Fysiikkamoottori tarkkailee simulaatioon kuuluvien kappaleiden tilaa ja päivittää niissä tapahtuvat muutokset, kuten törmäykset ja kappaleiden liikkeet törmäysten jälkeen. Fysiikkamoottorin toiminta jakautuu törmäystarkasteluun, systeemin päivittämiseen ja muutosten välittämiseen rajapinnalle (Autodesk 2001, 81). Nämä vaiheet toistuvat samassa järjestyksessä käyttäjän määrittelemällä nopeudella, niin kauan kuin fysiikkamoottori on käynnissä.

4.2 Törmäystarkastelu

Olennaisin fysiikkamoottorin tehtävä on törmäyksen tunnistaminen. Törmäyksen tunnistamisessa tutkitaan milloin animoitavien kappaleiden pinnat osuvat toisiinsa. Tutkittava kappale jaetaan pienempiin elementteihin, joiden pintoja tarkkaillaan (Autodesk 2001, 77). Yksityiskohtaisesti mallinnettu kappale muodostuu suuresta määrästä polygoneja eli pintapaloja. Kappaleiden lähestyessä toisiaan on niiden kaikkia polygoneja tarkkailtava.

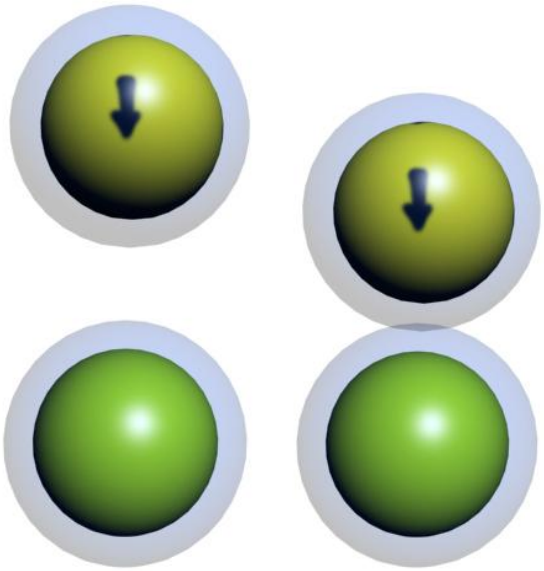
Animoitavalle kappaleelle voidaan määrittää sen todellisesta muodosta riippumaton geometria, jota käytetään pinnan määrittämiseen fysiikan mallinnuksessa.

Kun havaitaan pintojen osuminen toisiinsa, suoritetaan kaikille törmäykseen liittyville polygoneille laskutoimitukset. Mitä suuremmasta määrästä polygoneja mallinnettu kappale muodostuu, sitä suurempi on suoritettavien laskutoimitusten määrä.

Koska tällainen ohjelmallisesti toteutettu animaatio vaatii paljon laskutoimituksia, vaaditaan myös käytössä olevalta laitteistolta suorituskykyä. Törmäystarkasteluun käytetään yli 90 prosenttia simulaation tarvitsemasta laskenta-ajasta (Autodesk 2001, 86). On kuitenkin olemassa erilaisia tapoja, joilla fysiikkamoottorin työskentelyä voidaan oleellisesti helpottaa.

Simulaatioon on turhaa ottaa mukaan sellaisia kappaleita, joiden tilassa ei tapahdu muutoksia. Suuret kokonaisuudet kannattaa jakaa pienempiin osiin, joille fysiikan simulointi suoritetaan. Näin vältytään suorittamasta vaativaa simulointia sellaisille kohteille, joissa ei tapahtuisi mitään muutoksia.

Laskentatehokkuuden maksimoimiseksi voidaan törmäyksille määrittää törmäystoleranssi, jonka puitteissa tunnistaminen tapahtuu (kuva 7). Tämä tarkoittaa, että vasta pintojen tullessa riittävälle etäisyydelle toisistaan, kuitenkaan vielä koskettamatta, ryhdytään suorittamaan tarkempia laskutoimituksia odottavaa törmäystä silmällä pitäen.



Kuva 7. Törmäys tulkitaan tapahtuneeksi, kun kappaleiden törmäystoleranssi alueet leikkaavat. (Laine 2007.)

4.3 Systeemin päivittäminen

Fysiikkamoottorin havaittua törmäyksen kappaleiden välillä on vuorossa systeemin päivitys. Kappaleiden fysikaalisiin ominaisuuksiin sekä simulaatiossa vallitseviin olosuhteisiin perustuen kappaleille lasketaan sopiva liike. Liikkeiden laskemiseen fysiikkamoottori käyttää ainoastaan kappaleiden fysikaalisista ominaisuuksista annettuja tietoja. Mallinnetun kappaleen todellisella geometrialla ei siis ole merkitystä fysiikkamoottorin kannalta. (Autodesk 2001, 81.)

Luonnollisesti kappaleiden määrä vaikuttaa olennaisesti laskutoimitusten määrään, joten suorituskykyä voidaan optimoida jättämällä simulaation ulkopuolelle kappaleet, jotka eivät vaikuta törmäykseen.

On myös tärkeätä määrittää, kuinka usein systeemiä päivitetään eli kuinka monta laskutoimitusta suoritetaan aikayksikköä kohden. Simulaation tarkkuutta voidaan parantaa suorittamalla systeemin päivitys animaation esitysnopeutta tiheämmin. Jos toteutettavan animaation esitysnopeus on 25 ruutua sekunnissa, simulaation tarkkuus paranee merkittävästi, mikäli systeemin päivitys suoritetaan vaikkapa neljä kertaa jokaista ruutua kohden, yhteensä siis sata kertaa sekunnissa. Simulaation tarkkuutta käsitellään tarkemmin hieman jäljempänä.

4.4 Muutosten välittäminen rajapinnalle

Fysiikkamoottorin suoritettua kappaleen liikkeen laskemisen täytyy päivittää myöskin ohjelman käyttäjälle tuleva informaatio. Fysiikkamoottori päivittää muutokset käytettävälle rajapinnalle, jolloin mallinnusohjelman aikajanalle muodostetaan tarvittavat avainkehykset.

Fysiikkamoottorin tekemälle animaatiolle voidaan tämän jälkeen suorittaa samat muokkaustoimenpiteet, kuin perinteisesti käsin toteutetulle key frame -animaatiollekin.

5 3DS MAX REACTOR

5.1 Yleistä Reactorista

Fysiikkamoottoreita on saatavilla sekä ilmaisia että kaupallisia. Suurten yritysten ja tutkimuslaitosten käytössä on myös kyseisten organisaatioiden omiin käyttötarkoituksiinsa räätälöimiä sovelluksia.

Tässä opinnäytetyössä dynaamista animaatiota tutkitaan käyttäen apuna 3ds Max -ohjelmaan sisältyvää Reactor-fysiikkatyökalua. Alunperin Reactor oli erillinen 3ds Maxiin liitettävä laajennusosa, mutta on jo muutaman ohjelmistoversion ajan sisältynyt siihen kiinteänä osana. Reactorin on kehittänyt irlantilainen Havok.com, Inc. -yhtiö, joka kuuluu johtaviin kehittäjiin dynaamisen simulaation saralla. Fysiikan mallinnukseen käytetään siis Havok-fysiikkamoottoria, josta on olemassa eri käyttötarkoituksiin soveltuvia versioita. Sitä käytetään laajasti myös tietokone- ja konsolipeleissä sekä elokuvateollisuudessa. (Havok 2006.)

5.2 Simulaation asetukset

5.2.1 Yleiset asetukset

Mahdollisimman todenmukaisen animaation aikaansaamiseksi on simuloitavassa ympäristössä vaikuttavien fysikaalisten voimien merkitys kriittinen. Tästä syystä kyseisten asetusten määrittäminen on tärkeimpiä työvaiheita fysiikkamallinnusta hyödyntävän animaation toteuttamisessa.

Simuloitavassa ympäristössä, kuten myöskin reaalimaailmassa, fysiikan perustana on painovoima. Jokaiseen kappaleeseen maan päällä kohdistuu suoraan alaspäin vaikuttava painovoima. Simulaatiossa voidaan painovoima määrittää kaikkien kolmen akselin suhteen vektorisuureina, jolloin voimien yhteisvaikutuksesta aikaansaadaan vinosti kappaleisiin kohdistuva jatkuva voima. On muistettava, että kyseisen voiman suure on kiihtyvyys, joten sitä ei voida käyttää kappaleiden liikuttamiseen tasaisella nopeudella.

Mikäli halutaan simuloida olosuhteita, jotka poikkeavat maanpinnalla vaikuttavista, voidaan painovoiman suuruus vapaasti määrittää halutun suuruiseksi. Voiman suunta ilmaistaan kyseisen akselin suuntaisena, joten voiman suunta voidaan muuttaa vaihtamalla sen etumerkkiä.

5.2.2 Mittakaava

Simulaation ja mallinnetun näyttämön on oltava samassa mittakaavassa eli mittayksiköiden tulee olla yhteneväiset. Fysiikkamoottori suorittaa tehtävänsä simulaatiolle määritettyjen mittayksiköiden mukaisesti, joten mittakaavojen poikkeaminen toisistaan vaikuttaa voimakkaasti kappaleiden käyttäytymiseen.

Oikean mittayksikön valitsemiseksi on hyvä hieman tutkia tietokoneen tapaa käsitellä lukuja. Tietokoneissa lukuja käsitellään binäärijärjestelmässä, jolloin niiden esittämiseen tarvitaan suuri määrä bittejä, joita kuitenkin on käytössä vain rajallisesti. Tästä syystä reaalilukuja käsitellään tietokoneissa liukulukuina, jolloin lukuja joudutaan pyöristämään tai katkaisemaan, joten myös niiden tarkkuus heikkenee. (Hillis 1999, 46; Hotokka 2004.)

Fysiikkaa mallinnettaessa laskutoimitukset ovat monimutkaisia ja saattavat sisältää suuren määrän välituloksia. Pyöristysvirheet välituloksissa kertautuvat ja siitä johtuen voidaan saada lopputuloksena täysin virheellistä liikettä. Simulaatiossa tulisikin välttää hyvin suuria ja toisaalta paljon desimaaleja sisältäviä lukuja, joissa pyöristysvirheet ovat merkittävämpiä. Optimaalisin arvo laskennan kannalta olisi yksi, joten sivunpituudeltaan metrin kokoisen laatikon liikkeitä simuloitaessa päästään parempaan tarkkuuteen käytettäessä mittayksikköinä metrejä, kuin vaikkapa senttimetrejä (Autodesk 2005a).

Kuten aiemmin todettiin, määrittää törmäystoleranssi millä etäisyydellä kappaleiden tulkitaan osuvan toisiinsa. Suuri toleranssi helpottaa tietokoneen prosessorin työtaakkaa, mutta voi aikaansaada lopullisessa animaatiossa kappaleiden välissä näkyvän raon. Kappaleiden pinnat eivät siis todellisuudessa kosketa toisiaan.

Nyrkkisääntönä tulisi törmäystoleranssi pitää suurempana kuin 1/40 simulaation mittakaavan yksikköä (Autodesk 2005b). Mikäli simuloitavat kappaleet ovat kuitenkin hyvin pieniä, voidaan simulaation mittakaavaa ja painovoiman suuruutta muuttamalla aikaan saada tarpeeksi pieni törmäystoleranssi. Tällöin voi kuitenkin esiintyä laskentatarkkuuteen liittyviä ongelmia, jos käytettävät mitat tai painot ovat kovin pitkiä lukuja.

5.2.3 Liiketilan määrittäminen

Kappaleen liikkeen hidastuessa ja lähestyttäessä lepotilaa saatetaan törmätä fysiikkamallinnuksessa ongelmalliseen tilanteeseen. Liikkeiden hidastuessa myös liikemäärät ovat hyvin pieniä. Kappaleen liikkeet voivat olla merkityksettömän pieniä, joten niiden laskemiseen kulutetaan turhaan laskentatehoa. Esimerkiksi näin voi tapahtua jousen päässä roikkuvan punnuksen tai vapaasti heiluvan heilurin kohdalla, jolloin nopeus ei saavutakaan nollaa, vaan muuttuu sen läheisyydessä päinvastaiseksi. Tilanne saatetaan havaita siten, että punnuksen tai heilurin liike hidastuu ensin normaalisti, mutta alkaakin jälleen kiihtyä.

Edellä kuvattu tilanne voidaan estää määrittämällä pienin liikemäärä, joka kappaleella voi olla ennen kuin se tulkitaan pysähtyneeksi. Samalla tavalla vältytään myös tuhlaamasta laskentatehoa liikkumattomiin kappaleisiin.

Pysähtyneiden kappaleiden määrittämiseksi voidaan asettaa arvot sekä pitkälle, että lyhyelle taajuudelle tai molemmille. Lyhyt taajuus määrittelee matkan, jonka kappaleen on vähintään liikuttava simulaation aikayksikön aikana ennen kuin se tulkitaan pysähtyneeksi. Pitkällä taajuudella kappaleen siirtymää tarkkaillaan harvemmin kuin lyhyellä taajuudella. Tällöin voidaan määrittää pienempi liike, joka sallitaan kappaleelle, vaikka sen varsinainen liike olisikin jo pysähtynyt. Kyseinen kappale voi siis värähdellä vielä törmäyksen jälkeenkin. Kun kappale tunnistetaan liikkumattomaksi, se poistetaan automaattisesti simulaatiosta, eli sille ei suoriteta törmäystarkastelua.

5.2.4 Törmäysparit

Törmäysparien määrittämisellä voidaan vaikuttaa merkittävästi suoritettavien laskutoimitusten määrään. Mikäli kappaleiden sijaintien perusteella tiedetään, mitkä kappaleet voivat simulaation aikana törmätä toisiinsa, voidaan törmäystarkastelu rajata suoritettavaksi ainoastaan näiden kappaleiden välillä. Oletuksena kaikkien kappaleiden välillä suoritetaan törmäystarkastelu. Jos kaksi kappaletta sijaitsee kaukana toisistaan tai muusta syystä ei voi törmätä toisiinsa, voidaan kyseinen törmäyspari jättää pois simulaatiosta. Paljon kappaleita käsittävässä simulaatiossa saavutetaan tällä tavalla merkittävä etu, koska törmäystarkastelussa ei varauduta sellaisiin törmäyksiin, joita ei missään tapauksessa tule tapahtumaan.

Törmäysparien määrittämisellä voidaan myös mahdollistaa toisissaan kiinni olevien kappaleiden käyttö simulaatiossa. Jos kappaleet ovat törmäystoleranssin kannalta liian lähellä toisiaan, tulkitaan kappaleiden lävistävän toisensa. Ongelma voidaan korjata kytkemällä kappaleiden välinen törmäystarkastelu pois päältä. (Autodesk 2005c.)

Kaikista jäykkien kappaleiden törmäyksistä voidaan ottaa talteen informaatiota. Törmäysinformaatiosta saadaan selville tietoja, kuten törmäävät kappaleet sijainteineen sekä nopeudet törmäyshetkellä (kuva 8). Tätä informaatiota voidaan käyttää apuna ajoitettaessa erilaisia käsin animoitavia tapahtumia, kuten partikkeliefektejä. (Autodesk 2005d.)

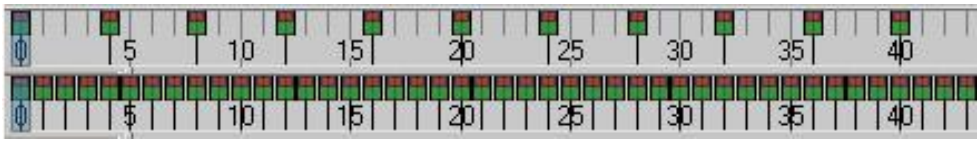
Ticks	Frame	Object A	Object B	Point	Normal	Speed	Phantom
18592	116	Ragdoll ...	mönkijä	(-9.968m, -9.968m...	(0.9, 0.1, -0.3)	0.043m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.496m, -9.496m...	(0.3, -0.8, -0.5)	0.019m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.563m, -9.563m...	(0.1, -1.0, -0.2)	0.075m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.505m, -9.505m...	(0.3, -0.0, 1.0)	0.007m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.513m, -9.513m...	(-0.6, 0.6, -0.5)	0.007m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.473m, -9.473m...	(0.6, -0.6, 0.6)	0.004m /s	Not Phantom
18593	116	ap02	alusta	(-11.821m, -11.82...	(-0.0, -0.0, 1.0)	0.082m /s	Not Phantom
18593	116	alusta	ap03	(-11.102m, -11.10...	(-0.0, -0.0, -1.0)	0.018m /s	Not Phantom
18593	116	ap04	alusta	(-9.759m, -9.759m...	(0.0, 0.0, 1.0)	0.055m /s	Not Phantom
18593	116	ap04	alusta	(-9.779m, -9.779m...	(0.0, 0.0, 1.0)	0.056m /s	Not Phantom
18593	116	alusta	moPyör...	(-9.686m, -9.686m...	(0.0, 0.0, -1.0)	0.014m /s	Not Phantom
18593	116	alusta	moPyör...	(-8.453m, -8.453m...	(0.0, 0.0, -1.0)	0.026m /s	Not Phantom
18593	116	Ragdoll ...	tanko	(-10.327m, -10.32...	(-0.0, -0.2, 1.0)	0.19m /s	Not Phantom
18593	116	Ragdoll ...	tanko	(-10.317m, -10.31...	(0.0, -0.2, 1.0)	0.117m /s	Not Phantom
18593	116	Ragdoll ...	tanko	(-10.331m, -10.33...	(-0.0, -0.2, 1.0)	0.221m /s	Not Phantom
18593	116	Ragdoll ...	tanko	(-10.327m, -10.32...	(-0.2, -0.1, 1.0)	0.189m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.331m, -10.33...	(-0.0, -0.2, 1.0)	0.218m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.327m, -10.32...	(-0.2, -0.1, 1.0)	0.186m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.314m, -10.31...	(0.2, 0.9, 0.4)	0.041m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.334m, -10.33...	(-0.8, 0.6, 0.3)	0.078m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.331m, -10.33...	(-0.7, 0.6, 0.3)	0.067m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.317m, -10.31...	(0.6, -0.1, 0.8)	0.095m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.317m, -10.31...	(0.7, 0.0, -0.8)	0.087m /s	Not Phantom
18592	116	Ragdoll ...	tanko	(-10.32m, -10.32m...	(-0.8, 0.6, -0.0)	0.012m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.968m, -9.968m...	(0.9, 0.1, -0.3)	0.043m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.496m, -9.496m...	(0.3, -0.8, -0.5)	0.019m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.563m, -9.563m...	(0.1, -1.0, -0.2)	0.075m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.505m, -9.505m...	(0.3, -0.0, 1.0)	0.007m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.513m, -9.513m...	(-0.6, 0.6, -0.5)	0.007m /s	Not Phantom
18592	116	Ragdoll ...	mönkijä	(-9.473m, -9.473m...	(0.6, -0.6, 0.6)	0.004m /s	Not Phantom
18593	116	ap02	alusta	(-11.821m, -11.82...	(-0.0, -0.0, 1.0)	0.082m /s	Not Phantom
18593	116	alusta	ap03	(-11.102m, -11.10...	(-0.0, -0.0, -1.0)	0.018m /s	Not Phantom

Kuva 8. Ote törmäysinformaatiosta. (Laine 2007.)

5.2.5 Simulaation tarkkuus

Simulaation esikatselu- ja animaatioasetuksista olennaisimmat ovat simulaation alkamis- ja päättymisajankohtien sekä tarkkuuden asetukset. Simulaation alkamiselle ja päättymiselle voidaan määrittää omat avainruudut, mikäli näyttämöllä on valmiina muuta animoitua liikettä tai muusta syystä halutaan, että fysiikan mallinnus ei ole käytössä koko aikajanalla näkyvää aikaa.

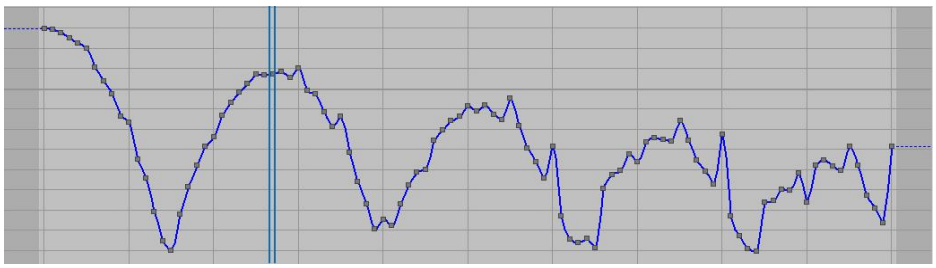
Simulaation tarkkuuden määrittää aikayksikköä kohden tehtävien systeemin päivitysten määrä. On määriteltävä, kuinka monta kehystä animaatio etenee yhden simulaation askeleen aikana ja kuinka monta kertaa fysiikkamoottori päivittää systeemin tuona aikana. Jokaisen simulaation askeleen jälkeen muodostetaan aikajanalle avainkehys (kuva 9). Koska liike on avainkehysten välillä aina suoraviivaista, on liikkeen tarkkuus suoraan verrannollinen simulaation askelten määrään.



Kuva 9. Alemmalla aikajanalla simulaatio etenee askeleen jokaisen avainkehyksen aikana, ylemmällä simulaatio etenee neljän kehyksen välein. (Laine 2007.)

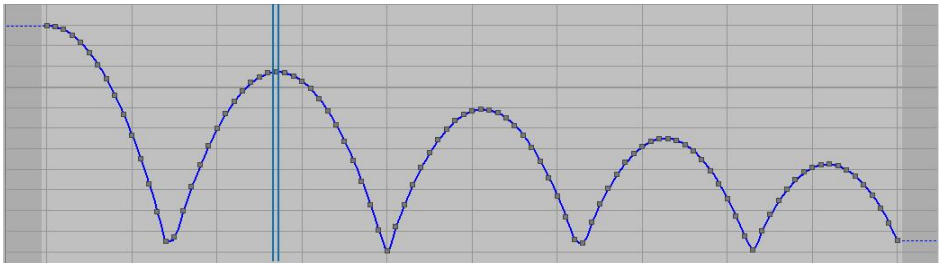
Usein ei kuitenkaan päästä riittävään tarkkuuteen, jos simulaatio etenee yhden askeleen avainkehystä kohden, joten systeemin päivitystiheyttä tulee nostaa. Tämä tapahtuu lisäämällä simulaation askeleen aikana tapahtuvien päivitysten määrää.

Simulaation tarkkuusmääritysten havainnollistamiseksi tarkastellaan pomppivan pallon pystysuuntaisen liikkeen kuvaajaa (kuva 10). Simulaatio etenee viiden avainkehyksen aikana yhden askeleen. Vastaavasti yhden askeleen aikana suoritetaan systeemin päivitys kymmenen kertaa. Animaation edetessä viiden avainkehyksen verran suorittaa fysiikkamoottori tänä aikana systeemin päivityksen yhteensä kymmenen kertaa. Kuvaaja on varsin epämääräinen, joten liike on epätarkkaa, paikoin jopa väärän suuntaista.



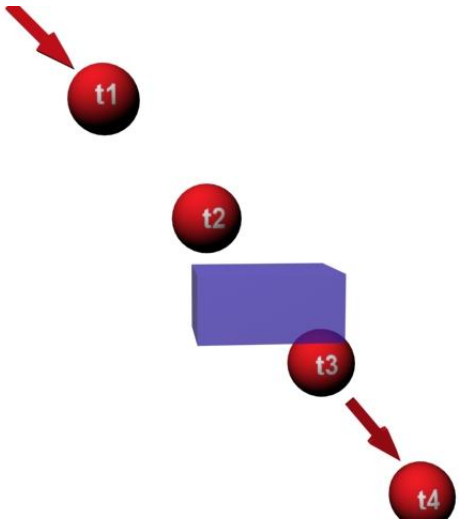
Kuva 10. Pomppivan pallon kuvaaja kun systeemi päivitetään kaksi kertaa yhtä avainkehystä kohden. (Laine 2007.)

Kun simulaatio etenee yhden askeleen jokaisen avainkehyksen aikana ja yhden askeleen aikana päivitetään systeemi kaksikymmentä kertaa, päästään huomattavasti parempaan tarkkuuteen (kuva 11). Systeemi päivitetään nyt sata kerta viiden avainkehyksen aikana.



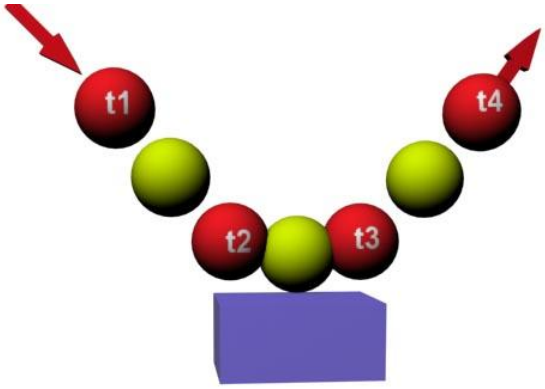
Kuva 11. Pomppivan pallon kuvaaja kun systeemi päivitetään kaksikymmentä kertaa yhtä avainkehystä kohden. (Laine 2007.)

Myös simulaatiossa esiintyvät nopeudet määrittävät, kuinka usein systeemin päivitys on syytä suorittaa. Mikäli kappale liikkuu niin nopeasti, että se lävistää toisen kappaleen pinnaan systeemin päivitysten välillä (kuva 12), ei kappaleiden välistä törmäystä havaita (Autodesk 2001, 83).



Kuva 12. Systeemin päivitys punaisten pallojen kohdalla, jolloin törmäystä ei havaita. (Laine 2007.)

Edellä mainittu aiheuttaa virheitä lopullisessa animaatiossa, koska kappaleet lävistävät toisensa ja saattavat jäädä kiinni toisiinsa. Kiinni tarttuminen johtuu siitä, että pintojen välillä havaitaan törmäys vasta siinä vaiheessa, kun toinen kappale on tulossa ulos toisen kappaleen sisältä. Tilanne korjataan lisäämällä päivitysten määrää aikayksikköä kohden (kuva 13).



Kuva 13. Systeemin päivitys suoritetaan myös keltaisten pallojen kohdalla, jolloin pallo muuttaa suuntaansa osuessaan laatikkoon. (Laine 2007.)

Lisättäessä fysiikkamoottorin työkierrosten määrää lisääntyy laskutoimitusten määrä, joten myös aiemmin kuvattujen liukulukulaskennassa esiintyvien laskuvirheiden todennäköisyys kasvaa. Simulaation tarkkuuden parantaminen kasvattamalla päivitystiheyttä saattaa johtaa tietyn rajan jälkeen siihen, että tarkkuus todellisuudessa heikkenee.

Mikäli kappaleiden liikkeet ovat hyvin suoraviivaisia, kuten yksittäinen putoava pallo, systeemiä ei ole tarpeen päivittää kovin usein. Toisaalta paljon kappaleita sisältävä simulaatio on päivitettävä huomattavasti tiuhempaan tahtiin, koska kappaleiden välisiä törmäyksiä tapahtuu useammin.

Systeemin päivitystiheydelle ei voida antaa yleispätevää ohjearvoa, vaan se riippuu eri tekijöistä, kuten törmäävien kappaleiden määrästä, niiden muodonmuutoksista sekä liikkeiden nopeudesta ja monimutkaisuudesta.

Aikaskaalauksella voidaan vaikuttaa animoitavan liikkeen nopeuteen (Autodesk 2005e). Aikaskaalaus ei vaikuta animaation esitysnopeuteen, vaan hidastaa simulaatiota vähentämällä aikayksikköä kohti tehtävien systeemin päivitysten määrää. Tällä tavalla voidaan tuottaa hidastettua tai nopeutettua animaatiota ilman videoeditointiohjelmassa tapahtuvaa jälkikäsittelyä. On huomioitava, että aikaskaalaus vaikuttaa vain fysiikkamoottorin muodostamaan liikkeeseen, joten näyttämöllä oleviin käsin animoituihin kappaleisiin sillä ei ole vaikutusta.

5.3 Kappaleen ominaisuudet

5.3.1 Fysikaaliset ominaisuudet

Simulaation yleisten asetusten määrittäessä vallitsevat voimat, täytyy mallinnetuille kappaleille antaa fysikaalisia ominaisuuksia, jotta ne reagoisivat näiden vallitsevien voimien mukaisesti.

Minkä tahansa kappaleen ominaisuuksista merkittävin fysiikan kannalta on sen massa. Mikäli kappaleen massaa ei tunneta, on sen käyttäytymistä mahdotonta tutkia matemaattisesti.

Kuten aiemmin todettiin, fysiikkamoottori suorittaa toimenpiteensä ainoastaan perustuen kappaleelle annettuihin fysikaalisiin ominaisuuksiin. Esimerkkinä voidaan ajatella näyttämöllä olevaa laatikkoa, johon vaikuttaa sivuttainen voima, kuten tuuli. Jos laatikolle ei ole määritelty massaa, ei fysiikkamoottori voi laskea tuulen eikä myöskään painovoiman vaikutusta siihen. Laatikko pysyy paikallaan. Sen sijaan törmäystarkastelu voidaan laatikolle suorittaa vaikka sillä massaa ei olisikaan. Tällöin toisen, massallisen, kappaleen osuessa laatikkoon törmäävä kappale käyttäytyy, kuten se käyttäytyisi törmätessään kiinteään esteeseen, mutta laatikko ei edelleenkään liiku.

Kappaleen pinnalle voidaan antaa kitkaa kuvaava arvo, jolla määritellään, miten kappale liukuu suhteessa pintaan, johon se on kosketuksessa. Koska todellisuudessa kitkakerroin ilmaisee kahden pinnan välistä keskinäistä kitkaa, on tällainen yhdelle pinnalle annettu arvo lähinnä viitteellinen. Myöskään lepo- ja liikekitkaa ei voida erikseen määrittää. Lisäksi kappaleen massa vaikuttaa oleellisesti kitkavoiman suuruuteen, kuitenkaan mallinnetun kappaleen massan ja kitkakertoimen arvojen muuttamisella ei ole vaikutusta toisiinsa.

Koska aineidenväliset kitkakertoimet on määritetty kokeellisesti ja kitkan suuruuteen vaikuttaa suuri määrä muuttujia, annetaan kirjallisuudessa kitkakertoimille usein pelkkä arvoalue. Kahden animoitavan kappaleen välisen kitkakertoimen suuruusluokka saadaan lähelle oikeaa jos tiedetään materiaalien välinen kitkakerroin luonnossa ja asetetaan se molempien kitka-arvoksi.

Jäykkä muotonsa säilyttävä kappale ei jousta törmäyksessä, mutta sille pystytään antamaan matemaattinen arvo, jonka avulla törmäystarkastelussa voidaan simuloida kappaleen elastisuutta. Näin voidaan säädellä kappaleiden kimpoamista niiden osuessa toisiinsa.

Luonnossa elastiset kertoimet, kuten kimmokerroin, liukukerroin ja puristuvuuskerroin, ilmaisevat materiaalin herkkyyttä muodon muutoksille (Hautala & Peltonen 2002, 144 - 150). Jäykkä kappale ei kuitenkaan muuta muotoaan, joten mallinnetun kappaleen elastisuudelle annettu arvo ei vastaa luonnossa esiintyviä materiaalien elastisia kertoimia. Fysiikkamoottori yhdistää törmäävien kappaleiden elastisuudelle annetut arvot, kuten se tekee myös kitkavoiman kohdalla. Haluttaessa määrittää tarkasti törmäyksessä tapahtuva kimmoke on molemmille kappaleille annettava sama arvo.

5.3.2 Kappaleen aktiivisuus

Kappaleiden aktiivisuutta simulaatiossa voidaan säädellä. Sopivan tason kappaleen aktiivisuudelle määrää kappaleen asema simulaation alkaessa sekä siihen törmäävät kappaleet. Jos kappale määritetään passiiviseksi, ottaa fysiikkamoottori sen ominaisuudet huomion vasta kun toisen kappaleen havaitaan törmäävän siihen. Tällä tavalla säästetään huomattavasti laskentatehoa, koska tällaisen kappaleen käyttäytymistä tutkitaan vain, mikäli se joutuu vuorovaikutukseen toisen kappaleen kanssa.

Jos kappaleen törmäystarkastelu kytketään pois päältä, vaikuttavat siihen kaikki simulaatiossa määritetyt voimat, mutta se ei reagoi muihin kappaleisiin törmäystilanteessa. Kappale putoaa painovoiman mukaisesti, ja siihen vaikuttavat muutkin ulkoiset voimat, kuten tuuli ja kytkennät kappaleisiin, joilla on törmäystarkastelu.

Mikäli kappale määritellään peräänantamattomaksi, siihen ei vaikuta mitkään ulkoiset voimat, mutta kappaleiden väliset kytkennät toimivat. Tällaiselle kappaleelle tehdään törmäystarkastelu, mutta se pysyy liikkumattomana sille määrätyssä paikassa törmäyksistä tai muista voimista riippumatta. Käsin animoidut kappaleet, jotka halutaan sisällyttää simulaatioon, on määritettävä tällaisiksi peräänantamattomiksi kappaleiksi.

Haamukappale käyttäytyy animaatiossa kuten kappale, jolle ei suoriteta törmäystarkastelua, mutta fysiikkamoottori kuitenkin laskee törmäykset kappaleelle. Tiedot törmäyksistä otetaan talteen ja tätä törmäysinformaatiota voidaan käyttää hyödyksi esimerkiksi muiden animoitavien tapahtumien ajoittamisessa.

5.3.3 Simuloitu geometria

Kuten on jo todettu, käyttää fysiikkamoottori kappaleen käsittelemiseen todellisten geometristen ääriviivojen sijasta kappaleelle annettuja matemaattisia arvoja. Näillä arvoilla ilmoitetaan kappaleen törmäystarkastelussa tutkittavat ääriviivat eli kappaleen simuloitu geometria.

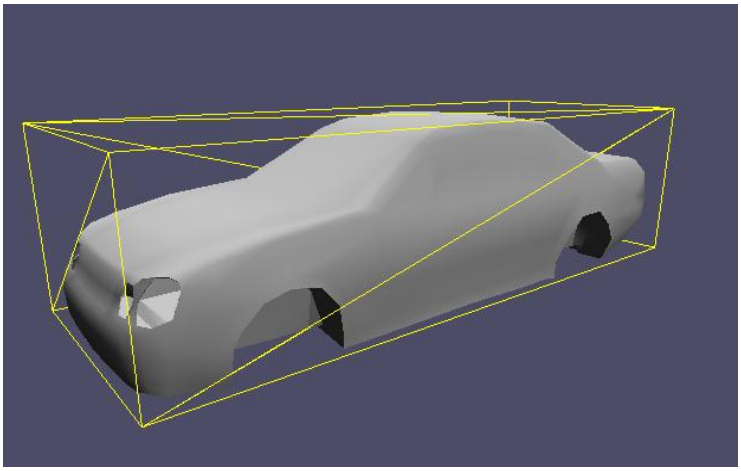
Törmäystarkastelussa käytetty geometria vaikuttaa ratkaisevasti simulaation tarkkuuteen. Tarkalla paljon tahoja käsittävällä geometrialla päästään suureen tarkkuuteen, mutta laskennan lisääntyessä prosessi hidastuu ja järjestelmä muuttuu epästabiilimmaksi.

Simulaatiossa kappaleet jaetaan geometrialtaan kuperiin ja koveriin. Kuperan kappaleen sisälle on mahdollista muodostaa kuviteltu suora mistä tahansa pisteestä toiseen ilman, että suora käy geometrian ulkopuolella. Koveralle geometrialle tehdään törmäystarkastelu myös sisäpinnalle. Tästä syystä on kappaleet, joiden sisälle asetetaan muita kappaleita määriteltävä koveriksi. Kappaleen geometriana olisi pyrittävä käyttämään kuperaa muotoa nopeamman simuloitavuuden takia. (Autodesk 2005f.)

Mikäli simuloitu geometria määritellään koveraksi, on mahdollista, että kappaleet tarttuvat törmäystilanteessa kiinni toisiinsa luvussa 5.2.5

mainitulla tavalla. Tällöin törmäys havaitaan kappaleiden välillä vasta siinä tilanteessa kun kappaleet ovat lävistäneet toisensa ja ovat irtoamassa. Kuperan geometrian ollessa kyseessä ei edellä mainittua pääse tapahtumaan, koska kappaleille ei suoriteta törmäystarkastelua sisältä päin.

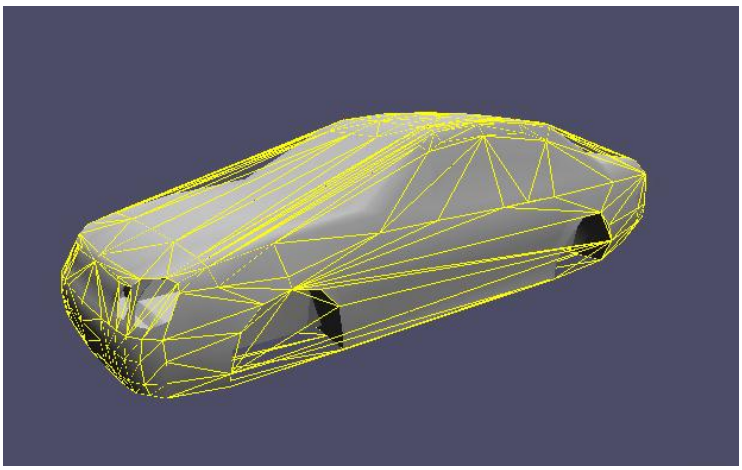
Simuloidun geometrian määrittelemiseksi on varsin laajat mahdollisuudet. Törmäystarkastelun kannalta helpoin geometria on suorakulmainen särmiö, jonka sisään käsiteltävä kappale juuri mahtuu (kuva 14). Kappaletta käsitellään särmiönä, joten törmäystarkastelu tehdään ainoastaan kuudelle sivulle. Myös pyöreä pallo, jonka sisälle kappale mahtuu, voidaan ottaa käytettäväksi geometriaksi. Tämä vaihtoehto käsittää jo huomattavasti enemmän tahoja kuin särmiö, joten laskutoimitusten määrä kasvaa edelliseen verrattuna.



Kuva 14. Särmiön ja pallon muotoiset simuloidut geometriat soveltuvat hyvin yksinkertaisille tai pienille kappaleille, jolloin epätarkkuuksia on vaikea havaita lopullisessa animaatiossa. (Laine 2007.)

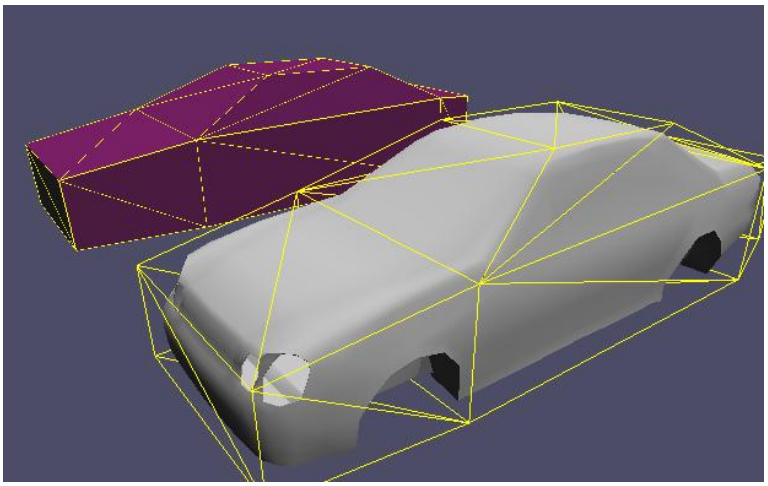
Molemmat edellä mainitut tavat soveltuvat parhaiten muodoltaan hyvin lähellä näitä perusprimitiivejä oleville tai kooltaan pienille kappaleille. Etenkin särmiötä käyttämällä saavutetaan etua, mikäli simulaatio sisältää paljon pieniä kappaleita, joiden käyttäytymisen suhteen ei ole kovin korkeita vaatimuksia.

Jos animaatiossa on tarkoitus kuvata kappaleita läheltä tai ne ovat muodoiltaan monimutkaisia, täytyy simuloitu geometria määritellä paremmin pinnanmuotoja myötäileväksi. Tämä voidaan toteuttaa muodostamalla eräänlainen kappaleen muotoja seuraava kuori. Kupera verkkokuori muodostaa Mesh-verkon, joka piirtyy seuraten kappaleen ääriviivoja (kuva 15). Koska kuori on kupera, se ei kuitenkaan mukaudu pinnassa oleviin painanteisiin tai reikiin.



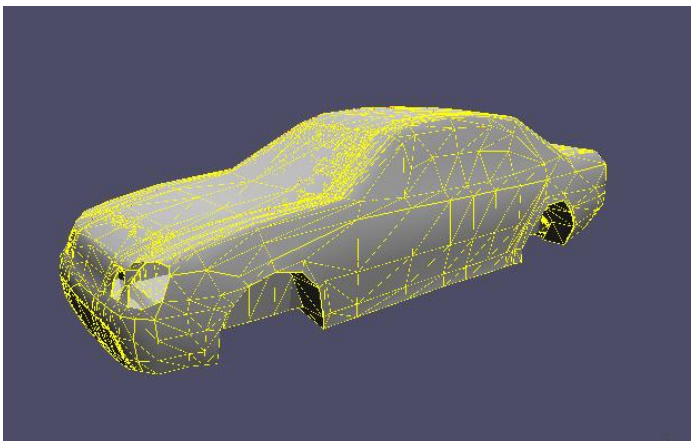
Kuva 15. Kupera verkkokuori. (Laine 2007.)

Verkkokuoreksi voidaan valita myös toisen kappaleen Mesh-verkko (kuva 16). Näin voidaan vähemmän pintapaloja käsittävän kappaleen pinnanmuotoa käyttää samanmuotoisen mutta yksityiskohtaisemman kappaleen geometriana.



Kuva 16. Simuloituna geometriana voidaan käyttää takana näkyvän yksinkertaisemman kappaleen mesh-verkkoa. (Laine 2007.)

Kovera verkkokuori myötäilee tarkasti kappaleen pinnanmuotoja tunkeutuen myös painanteisiin ja reikiin, joten se on törmäystarkastelun kannalta kaikkein tarkin vaihtoehto (kuva 17). Tutkittavien pintojen määrää vastaa kuitenkin kappaleen pintapalojen määrää, joten varsinkin monimutkaiset muodot voivat olla laskennan kannalta hyvin raskaita.



Kuva 17. Tarkasti pinnanmuotoa noudattava kovera verkko. (Laine 2007.)

Myös koverana verkkokuorena voidaan käyttää yksinkertaisemman kappaleen pintaa, jolloin laskentatyön määrä vähenee, mutta päästään monimutkaistenkin muotojen osalta parempaan tarkkuuteen. Koverat geometriat ovat huomattavasti hitaampia käsitellä, joten ensisijaisesti tulisi simuloinnissa käyttää kuperia geometrioita.

Kappaleen muodon lisäksi on simuloitua geometriaa valittaessa huomioitava myös tapahtuvien törmäysten voimakkuus. Kappaleet, joiden simuloitu geometria on kupera, saattavat suurilla nopeuksilla läpäistä toiset kappaleet.

5.4 Dynamiikat

Simulaatioon kuuluvat kappaleet kuuluvat erilaisiin kokoelmiin (engl. collection), jotka määräävät kappaleen dynamiikan, eli sen ominaisuudet, jotka määräävät kappaleen käyttäytymisen sen joutuessa vuorovaikutuksiin toisten kappaleiden kanssa. Esimerkiksi taipuisaa kangasta kuvaava kappale kuuluu eri kokoelmaan kuin kova tiiliskivi.

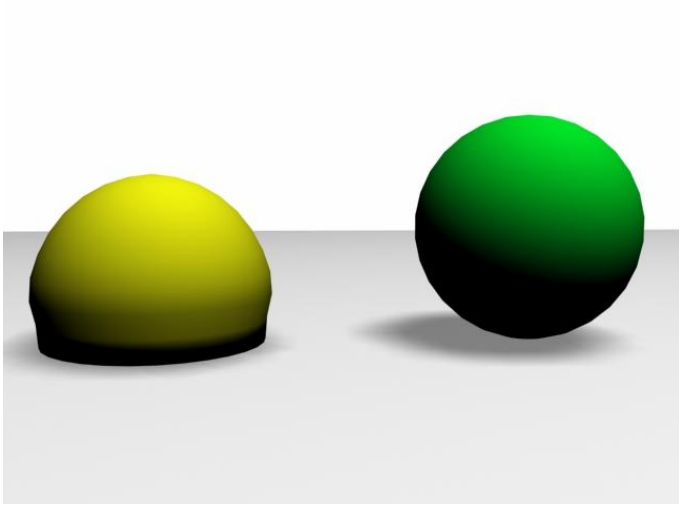
Jokainen mallinnettu kappale, jolla halutaan olevan fysikaalisia ominaisuuksia, määritetään johonkin kokoelmaan. Tähän kokoelmaan kuuluvat kaikki samanlaisen dynamiikan omaavat kappaleet. Samaan dynamiikkaan kuuluvien kappaleiden ei kuitenkaan tarvitse olla ominaisuuksiltaan samanlaisia, vaan esimerkiksi jokaisen paino tai tiheys voidaan määritellä erikseen. Monimutkaisen simulaation tapauksessa voidaan käyttää useampia samanlaisia dynamiikkakokoelmia, jolloin kappaleiden ryhmittely voidaan toteuttaa selkeämmin. Yksi kappale ei kuitenkaan voi kuulua kuin yhteen dynamiikkaan. (Murdock 2005, 986.)

Mikäli kappaleen halutaan sisältyvän simulaatioon ja siten olevan vuorovaikutuksessa muihin kappaleisiin, on sen siis kuuluttava johonkin kokoelmaan.

3ds Max Reactorin kokoelmatyypit

- Rigid Body on tarkoitettu jäykille, muotonsa säilyttäville kappaleille. Esimerkiksi talon seinä voisi olla tällainen jäykkä kappale, joka ei muuta muotoaan vaikkapa pallon osuessa siihen. Jos kuitenkin halutaan seinän sortuvan osumasta, voidaan se koostaa pienemmistä osasista, jotka kuuluvat samaan Rigid Body -kokoelmaan.

- Soft Body -dynamiikkaan kuuluvat kappaleet voivat muuttaa törmäyksessä muotoaan. Tällainen pehmeä kappale käyttäytyy törmäyksessä ja sen jälkeen eri tavalla kuin jäykkä kappale (kuva 18). Sillä on erilaiset liukuominaisuudet, ja se voi muuttaa muotoaan toisen kapalleen osuessa siihen. Tällaisen kappaleen käsittely vaatii fysiikkamoottorilta enemmän laskemista pinnanmuodon muutoksista johtuen. Lisäksi kappale muodostuu tavallisesti suuremmasta määrästä pintapaloja, joten törmäystarkastelu on suoritettava tarkemmin.



Kuva 18. Kaksi samanlaista kappaletta käyttäytyy törmäyksessä eritavalla. Vasemmalla oleva pallo kuuluu *Soft Body* -kokoelmaan ja oikealla oleva *Rigid Body* –kokoelmaan. (Laine 2007.)

- Rope-dynamiikan avulla voidaan simuloida vapaasti muotoiltavia pitkiä ketjumaisia kappaleita, kuten erilaisia naruja tai köysiä. Kappaleen taipumisjäykkyyttä tai pitkittäistä venymistä voidaan säätää. Näin voidaan simuloida esimerkiksi joustavan kuminauhan liikettä. Myös paksuus voidaan määrittää erilaiseksi eri kohdissa kappaletta. Tällöin taipumis- tai venymisominaisuudet ovat erilaisia kappaleen eri osissa.

- Cloth-dynamiikka on Rope-dynamiikan kaltainen ominaisuuksiltaan, mutta soveltuu kappaleille, jotka ovat pinta-alaltaan laajempia. Tällä dynamiikalla toteutettavia ovat esimerkiksi tuulessa liehuva lippu tai jonkin esineen päälle laskeutuva kangas.

- Deforming Mesh -dynamiikan omaavaa kappaletta voidaan muokata erilaisten voimien avulla. Se voi esimerkiksi taipua tuulessa.

Koska kappaleen dynamiikka määrää, kuinka törmäystarkastelu sekä tapahtumien päivittäminen tehdään, on tärkeää miettiä, mitä dynamiikkoja on järkevää käyttää. Jos verrattaessa jonkin kappaleen käyttäytymistä sekä Rigid Body- että Soft Body -dynamiikoilla ja kappaleen käyttäytymisessä ei ole silmin havaittavaa eroa, ei ole järkevää käyttää enemmän laskentaa edellyttävää Soft Bodya. Ennen monimutkaisen simulaation mallintamista kannattaakin tehdä yksittäisillä kappaleilla testejä, joiden avulla voidaan määritellä kappaleille sopivat ominaisuudet. Luonnollisesti on myös syytä tutkia, mitkä tapahtumat on edes tarpeellista suorittaa dynaamisen animaation avulla.

5.5 Kappaleiden väliset voimat

Eri dynamiikkojen lisäksi fysiikan mallinnusohjelmassa on määritettävissä erilaisia voimia, jotka vaikuttavat kappaleiden väillä. Reactorissa näitä voimia saadaan aikaiseksi erityisten Reactor-objektien avulla. Nämä objektit eivät ole näkyvissä lopullisessa animaatiossa, vaan ne ainoastaan määrittelevät simulaatioon kuuluvien kappaleiden välisiä vaikutuksia. Reactor-objektit reagoivat ominaisuuksiensa mukaan eri tavalla näyttämöllä oleviin kappaleisiin sekä niiden aiheuttamiin voimiin. (Murdock 2005, 990.)

<u>Reactor-objektit</u>

- Plane on näyttämölle lisättävä näkymätön kiinteä pinta, johon osuessaan kappale voi ottaa kimmokkeen tai muuttaa suuntaansa.

- Spring-objektilla kaksi kappaletta voidaan kytkeä toisiinsa ja määrittää, kuinka voimakkaasti ne vetävät toisiaan puoleensa tai työntävät poispäin. Näin kappale saadaan seuraamaan toista kappaletta tai työntämään sitä edellään. Spring-objektille voidaan määrittää jäykkyyden ja vaimennuskyvyn lisäksi pituus lepotilassa. Lisäksi voidaan myös valita, toimiiko jousi vain puristus- vai vapautumissuuntaan.

- Linear Dashpot vaimentaa lineaarista liikettä kahden kappaleen välillä. Se ei työnnä kappaleita kauemmas toisistaan kuten Spring-objekti.

- Angular Dashpot toimii kuten edellä, mutta rajoittaa säteittäistä voimaa kappaleiden välillä. Esimerkiksi pyörivä kappale saa myös toisen kappaleen pyörimään.

- Motor-obektin avulla kappaleelle voidaan antaa voima, joka pyörittää sitä akselinsa ympäri.

- Wind-objekti on näyttämöllä vaikuttava lineaarinen voima, kuten nimensä mukaisesti tuuli. Wind-objekti on animoitavissa, joten sen ominaisuuksia, kuten suuntaa ja voimakkuutta, voidaan muuttaa simulaation aikana.

- Toy Car on yksinkertainen auto, jossa on pyörivät pyörät, jotka antavat sille suoraviivaisen liikkeen.

- Fracture-objekti on useammasta kappaleesta koostuva rakennelma, joka voi hajota osiin. Tällainen kokonaisuus voi esimerkiksi olla yksittäisistä tiilistä koostuva muuri, joka hajoaa jonkin kappaleen osuessa siihen. Tällöin jokainen tiili sijoitetaan Fracture-objektiin. Rakenteen rikkoutumiseen tarvittava voima määritellään yksittäisten kappaleiden sijaan koko rakennelmalle. Käyttäjä voi myös itse määrittää ajankohdan, jolloin rakennelma hajoaa.

- Water muodostaa pinnan, joka käyttäytyy nestepinnan tavoin, joutuessaan kontaktiin toisen kappaleen kanssa. Vaikka nesteen tilavuutta ei todellisuudessa mallinneta, voidaan pinnan parametreja muokkaamalla jäljitellä nesteen ominaisuuksia, kuten kykyä kelluttaa tiheydeltään riittävän pientä esinettä.

5.6 Liikeiden rajoittaminen

Constraining- eli rajoitinobjekteja käytetään Reactorissa rajoittamaan kappaleiden liikkeitä. Niiden avulla voidaan vaikuttaa siihen, miten kappaleet käyttäytyvät joutuessaan vuorovaikutuksiin muiden kanssa. Rajoitinobjektien avulla voidaan kiinnittää kappaleita toisiinsa ja määrittää minkä suuntainen liike kappaleiden välillä sallitaan, sekä millaisten voimien vaikutuksesta tämä liitos rikkoutuu. (Murdock 2005, 1000.)
Esimerkkinä liitettäessä ovi ovenkarmiin määritellään, kuinka paljon ja missä suunnassa karmiinsa nähden ovi voi aueta. Määriteltäessä, kuinka helposti liitos rikkoutuu, ovi saadaan putoamaan saranoiltaan sen paiskautuessa riittävän voimakkaasti tai jonkin esineen osuessa siihen riittävällä voimalla. Rajoittimien rajoittamat kappaleet voidaan määrittää pää- ja aliobjekteiksi. Tällöin aliobjekti seuraa pääobjektia rajoittimen määräämällä tavalla.

Rajoittimien avulla voidaan aikaansaada varsin monipuolisesti liitoksia kappaleiden välille. Useista kappaleista koostuvat kokonaisuudet sisältävät paljon kappaleiden välisiä rajoittimia, jotka vaikuttavat näiden kappaleiden välityksellä myös toisiinsa. Lisättäessä rajoitinten määrää lisääntyy muuttujien määrä ja siten kokonaisuuden hallitseminen hankaloituu. Tästä syystä kovin monimutkaisten rakenteiden muodostaminen rajoitinobjekteja käyttäen on hankalaa tai jopa mahdotonta. Monimutkaisissa kokonaisuuksissa apuna voidaan käyttää myös edellisessä kappaleessa mainittua Fracture-objektia, jolloin rajoittimien määrä pysyy pienempänä.

Rajoitinobjektit

Constraint Solver on ratkaisija, jonka useimmat rajoittimet vaativat toimiakseen. Sen alle kootaan kaikki ne ratkaisijat, jotka vaikuttavat tietyn kokoelman kappaleisiin. Constraint solverille on siis ilmoitettava myös mihin kokoelmaan kyseiset kappaleet kuuluvat, koska simulaatio voi sisältää useita kokoelmia. Constraint solveria ei tarvita yksinkertaisten ratkaisijoiden yhteydessä. Yksinkertaisia ratkaisijoita ovat Spring, Linear Dashpot ja Angular Dashpot.

- Rag Doll -rajoitin määrittelee ihmiskehon nivelten liikeradat. Liitettäessä se erityiseen humanoidi-objektiin, voidaan sen avulla simuloida ihmiskehon törmäyksiä siten, että raajojen liikkeet ovat luonnolliset. Rajoitin liittää automaattisesti ruumiinosat toisiinsa. Jokaisen nivelen liikkeiden rajoja voidaan säätää erikseen. On huomattava, että muista rajoitin-objekteista poiketen humanoidi-objekti ja siihen liitettävä Rag Doll -rajoitin saadaan käyttöön suorittamalla

rctRagdollScript.ms-skripti Utilities-paneelin MAXScript-kohdassa. Rag Doll -rajoitin voidaan liittää myös 3ds Maxin Biped luurankoihin, jolloin minkä tahansa hahmon animointi tällä tavalla on helppoa.

- Hinge sallii kappaleiden välillä tietyn akselin suuntaisen kiertoliikkeen, jolle voidaan määrittää liikkeen ääriarvojen lisäksi vastus, joustavuus sekä kestävyys eli voima, jolla kappaleiden välinen linkitys rikkoutuu.

- Point-Point kytkee kaksi kappaletta toisiinsa. Liitoksen tyyppi voidaan valita kolmesta vaihtoehdosta: normaali, joka käyttäytyy kuten näkymätön naru, jolloin perussäädöt, kuten säteittäinen ja aksiaalinen poikkeama sekä liitoksen kestävyys, ovat mahdollisia; rajoitettu tyyppi, jossa rajat on asetettavissa eri akseleiden suuntaisesti astelukuina; kolmantena vaihtoehtona jäykkä jousi, joka voi työntää kappaleita myös toisistaan poispäin.

- Prismatic ei salli kiertoliikettä, vaan ainoastaan yhden akselin suuntaisen liikkeen kahden kappaleen välillä.

- Car-Wheel on pyöräobjekti, joka synnyttää ajoneuvoa eteenpäin vievän lineaarisen voiman. Pyörän jousto-ominaisuuksia, pyörimis- ja etenemisnopeutta sekä kestävyyttä voidaan säätää.

- Point-Path kytkee kappaleen seuraamaan valmista polkua. Ainoastaan liitoksen jäykkyys, tiukkuus ja kestävyys ovat säädettävissä.

6 TEKNIIKAN SOVELTAMINEN

6.1 Esimerkin kuvaus

Koska fysiikan mallinnusta hyödyntävä 3d-animaatio perustuu muun muassa kappaleiden massoihin ja nopeuksiin, soveltuu se hyvin erilaisten törmäystilanteiden kuvaamiseen. Esimerkkinä toteutetaan 3d-animaationa törmäystilanne, jossa mönkijä, eli maastonelipyörä, törmää paikallaan olevaan henkilöautoon. Lisäksi mönkijän kuljettaja putoaa törmäyksen voimasta pois ajoneuvonsa päältä.

Esimerkillä pyritään havainnollistamaan fysiikan mallinnusta hyödyntävän animaation toteuttamisprosessia, sekä antamaan kuva niistä erityisvaatimuksista, joita mainittu animointitekniikka asettaa. Esimerkissä tullaan käsittelemään Rigid Body -dynamiikkaa käyttäviä muotoaan muuttamattomia kappaleita.

6.2 Mallintaminen

6.2.1 Näyttämö

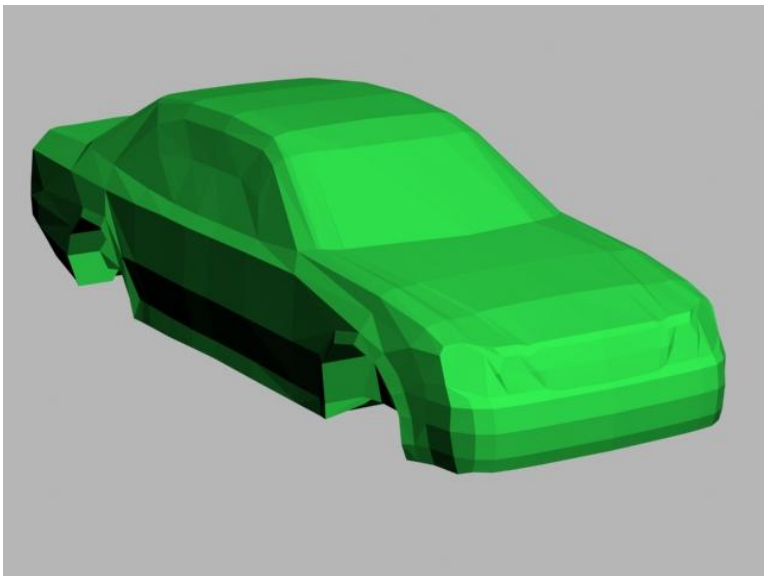
Kuten jo aiemmin on todettu, on dynamiikkasimulaation asetukset ensiarvoisen tärkeässä asemassa onnistuneen lopputuloksen kannalta. Ensimmäisenä tulee ratkaista mittakaava, jossa animaatio toteutetaan. Oikean mittakaavan ratkaisee simulaatioon kuuluvien kappaleiden koko, sekä käytettävät nopeudet. Tässä tapauksessa törmäyksen kannalta tärkeimmät kappaleet ovat törmäykseen osallistuvat henkilöauto ja maastomönkijä. Kyseisten ajoneuvojen luonnollinen kokoluokka huomioon ottaen, on sopiva mittayksikkö metri, jolloin pysytellään mahdollisimman lähellä simuloinnin kannalta edullisinta arvoa yksi. Näin ollen muodostuu henkilöauton pituudeksi noin 4,8 metriä ja mönkijän pituudeksi noin 2,2 metriä.

Dynamiikan simuloiminen asettaa näyttämöllä olevien kappaleiden mallintamiselle erityisiä vaatimuksia verrattuna key frame -tekniikalla toteutettuun animaatioon. Kappaleiden geometrialta edellytetään toimivuutta, koska liikkeet perustuvat fysiikan lakeihin. Key frame -tekniikalla saadaan kappale esimerkiksi kimpoamaan johonkin pintaan osuessaan mihin suuntaan tahansa. Dynamiikkasimulaatiossa sen sijaan kimmokkeen suunnan määrää törmäävien kappaleiden pintojen suunnat ja nopeudet. Ennen mallintamisen aloittamista on syytä miettiä, miten kappaleet todennäköisesti simulaatiossa käyttäytyvät, ja huomioida myös mahdollisuus muuttaa geometriaa tarvittaessa.

Törmäys tapahtuu tasaisella pinnalla, joka muodostetaan näyttämölle yksinkertaisesta suorakulmaisesta särmiöstä. Särmiön korkeudella ei ole merkitystä, mutta sen on pinta-alaltaan oltava niin suuri, että törmäävän ajoneuvon kiihdyttäminen tarvittavaan nopeuteen onnistuu särmiön pinnalla. Koska tarvittavaa kiihdytysmatkaa ei vielä tiedetä, voidaan alustan kokoa helposti muokata myöhemmin.

6.2.2 Ajoneuvot

Henkilöauton korista tehdään ensin karkea malli, jonka mesh-verkko käsittää 1350 pintaa (kuva 19). Tätä muotoa tullaan käyttämään törmäystarkastelussa simuloituna geometriana tarkemmalle mallille, joten se tallennetaan myöhempää käyttöä varten. Kori tehdään polygoni mallinnuksella, jolloin on oltava erittäin huolellinen liitettäessä kärkipisteitä toisiinsa. Mikäli geometriasta ei tule täysin suljettu, saattaa törmäystarkastelussa ilmetä ongelmia, koska kappaleella ei ole tilavuutta. Mesh-verkossa olevien aukkojen löytäminen voi olla myöhemmin vaikeata.



Kuva 19. Karkea malli auton korista. (Laine 2007.)

Tarkemman auton korin työstämistä jatketaan lisäämällä mesh-verkoon Mesh Smooth -määrite, jolloin korin linjat saadaan sulavammiksi. Tätä 6600 pintaa käsittävää mallia käytetään auton lopullisena korina. Yksityiskohdat, kuten ajovalot ja ikkunat, lisätään pintamateriaaleina. Auton pyörät lisätään omille paikoilleen erillisinä lieriön muotoisina kappaleina ja yksityiskohdat myös niihin pintamateriaalina. Yk-

sityiskohtien lisääminen pintamateriaalien avulla mahdollistaa karkeampien mesh-verkkojen käytön, jolloin simulointi voidaan suorittaa nopeammin (kuva 20). Henkilöauton malli koostuu yhteensä viidestä kappaleesta; korista sekä neljästä pyörästä, jotka liitetään koriin myöhemmin.

Kuva 20. Yksityiskohdat toteutetaan pintamateriaalina. (Laine 2007.)

Mönkijän mallintaminen ei eroa juurikaan auton korin mallintamisesta. Kuitenkin mönkijän monimutkaisemman muodon sekä siihen liitettävän ihmishahmon takia on muutamia seikkoja otettava huomioon.

Mönkijään liitetyt tavaratelineet, ohjaustanko ja iskunvaimentimet ovat erillisiä kappaleita, jotka on liitetty osaksi mesh-verkkoa (kuva 21). Nämä elementit voidaan myöhemmin irrottaa erillisiksi kappaleiksi, mikäli niitä tarvitsee siirtää tai muokata. Tämä mahdollisuus on huomioitava, koska ajoneuvon päälle sijoitettavan ihmishahmon käyttäytymistä törmäystilanteessa ei voida tarkasti arvioida. Mikäli jokin edellä mainituista osista estää hahmon liikkeen törmäyksessä, voidaan tilannetta korjata tekemällä geometriaan muutoksia.

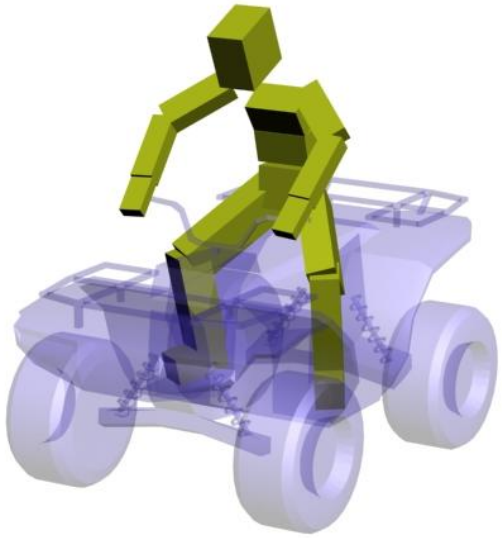
Kuva 21. Mönkijän malli. (Laine 2007.)

Myös mönkijälle tehdään simuloitua geometriaa varten karkea malli. Tämä malli peittää allensa tavaratelineet, jotta kuljettajan paikalle asetettava ihmishahmo ei takertuisi niihin (kuva 22). Pyörät toteutetaan samalla tavalla kuin henkilöautoonkin.

Kuva 22. Simuloituna geometriana käytettävä muoto peittää yksityiskohtaisemman mallin. (Laine 2007.)

6.2.3 Hahmo

Mönkijän kuljettajaksi luodaan Rag Doll -skriptin avulla ihmishahmoa kuvaava humanoidiobjekti, pituudeltaan 170 senttimetriä. Hahmo sijoitetaan oikealle paikalleen ja säädetään ruumiinjäsenet haluttuihin asentoihin. Ruumiinosien sääteleminen on suoritettava ennen rajoittimien asettamista kyseisille kappaleille (kuva 23). Rajoitin-objektien avulla kytkettyjä kappaleita ei tulisi liikutella toisiinsa nähden rajoittimen asettamisen jälkeen. Muutoin kappaleet hakeutuvat välittömästi simulaation alkaessa niihin asemiin, joissa ne olivat rajoitinta asetettaessa ja kappaleet käyttäytyvät odottamattomasti.



Kuva 23. Hahmo asetellaan haluttuun asentoon. (Laine 2007.)

6.3 Simulaatio

6.3.1 Ominaisuuksien määrittäminen

Ryhdyttäessä toteuttamaan dynamiikkasimulaatiota on ensimmäiseksi hyvä määritellä näyttämöllä vallitsevat olosuhteet. Esimerkissä on tarkoituksena kuvata maapallolla valitsevia olosuhteita, joten painovoimaksi asetetaan fysiikassa yleisellä tasolla käytettävä likiarvo -9,81 m/s. Koska mallintamisessa on käytetty mittayksikkönä metriä, on varmistuttava, että metri näyttämöllä ja simulaatiossa on yhtä pitkä. Tällä tavalla varmistetaan, että todellinen ja simulaation mittakaava ovat yhteneväiset, jolloin nopeudet ja voimat muodostuvat oikean suuruisiksi. Käytettäessä mittayksikkönä metriä voidaan törmäystoleranssiksi asettaa pienimmillään neljä millimetriä. Törmäävien kappaleiden kokoluokka huomioon ottaen voidaan simuloinnin nopeuttamiseksi kat-

soa suuremmankin toleranssin riittävän tässä tapauksessa. Törmäystoleranssiksi määritetään alustavasti yksi senttimetri.

Kappaleille annettavat massat ratkaisevat niiden käyttäytymisen simulaatiossa. Mallinnetun henkilöauton kokoisille autoille ilmoittavat valmistajat omapainoiksi keskimääriin noin 1600 kilogrammaa. Mallinnettu auto koostuu viidestä eri osasta, joten korin painoksi määritetään 1500 kilogrammaa ja kunkin pyörän painoksi 20 kilogrammaa. Pyörien kitka-arvoksi annetaan 0,8, joka ilmoitetaan lepokitkaksi kumin ja kuivan asfaltin välillä (Kervinen & Smolander 2003).

Mönkijän pyörien arvioidaan painavan hieman auton pyöriä enemmän. Yhden pyörän painoksi määritetään 25 kilogrammaa ja mönkijän painoksi 200 kilogrammaa, jolloin yhteispainoksi koko ajoneuvolle tulee 300 kilogrammaa. Kitkavoimaksi asetetaan sama 0,8 kuin auton pyörille. Koska kitkavoima vaikuttaa kahden pinnan välillä, on sama kitkakerroin asetettava myös alustakappaleelle, johon pyörät koskettavat.

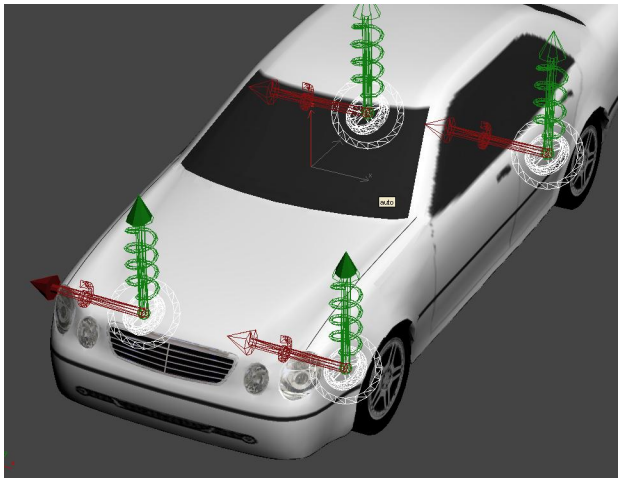
Mönkijän päälle asetettu hahmo koostuu seitsemästätoista osasta. RagDoll-skriptin luodessa kyseisen humanoidiobjektin, on jokaisen ruumiinosan paino oletusarvoisesti kymmenen kilogrammaa. Koska näillä arvoilla hahmon painoksi muodostuisi epärealistiset 170 kilogrammaa, muutetaan jokaisen ruumiinosan painoksi viisi kilogrammaa.

6.3.2 Ajoneuvojen rajoittimet

Useista kappaleista koostuvien kokonaisuuksien, kuten tässä tapauksessa ajoneuvojen ja ihmishahmon toiminnallisuuden märittäminen on tärkeimpiä työvaiheita. Toiminnallisuudella tarkoitetaan sitä, kuinka kappaleet liitetään kokonaisuuksiksi siten, että ne toimisivat halutulla tavalla simulaatiossa. Lopullinen toimivuus varmistetaan vasta tekemällä kokeita, koska monet virheet paljastuvat vasta simulaation aikana.

Tässä esimerkissä kaikki kappaleet ovat jäykkiä ja muotoaan muuttamattomia, joten dynamiikka kokoelmista tulee kysymykseen ainoastaan Rigid Body -kokoelma. Näiden kokoelmien lopullisen määrän ratkaisee kokonaisuuksien toiminnallisuus. Ensimmäiseen Rigid Body -kokoelmaan määrätään yhden kokonaisuuden muodostavat autonkori ja pyörät. Valittaessa tiettyihin kokoelmiin sisällytettäviä kappaleita tulee huomioida, että rajoittimilla tai muilla keinoin toisiinsa kiinnitettävien kappaleiden on kuuluttava samaan kokoelmaan.

Pyörien liittämiseksi auton koriin on eri vaihtoehtoja. Yksinkertaisin tapa Reactor-objekteista on Toy Car -ratkaisija, jolloin ajoneuvoa voitaisiin liikuttaa määrittämällä pyörille kulmanopeus. Pyörien halutaan kuitenkin pyörivän vain siinä tapauksessa, että törmäystilanteessa jokin voimaa saa auton liikkumaan. Tästä syystä paras tapa pyörien liittämiseksi on Car Wheel -rajoitin. Pyörille määritetään tämän rajoittimen avulla pyörimisakseli, asento koriin nähden, sekä jousitusliikkeen suunta (kuva 24). Pyörimisliikettä määrittelevät arvot asetetaan nollaksi, joten pyörään ei kohdistu minkäänlaisia jarruttavia tai kiihdyttäviä voimia, joten pyörä käyttäytyy simulaatiossa halutulla tavalla. Car Wheel -rajoittimet vaativat toimiakseen Constraint Solver -ratkaisijan, jollainen niille lopuksi muodostetaan.



Kuva 24. Pyörät liitettynä koriin Car Wheel –rajoittimilla. (Laine 2007.)

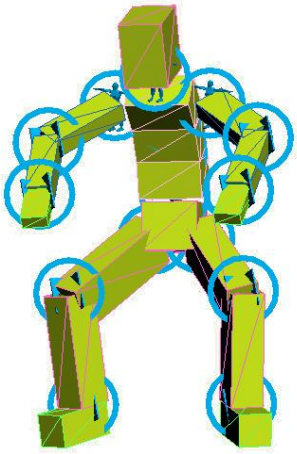
Mönkijän pyörät liitetään samalla tavalla käyttäen Car Wheel -rajoitinta. Mönkijän on tarkoitus liikkua, joten on ratkaistava miten tämä liike saadaan aikaiseksi. Myös Car wheel -rajoittimelle voidaan antaa kulmanopeus, jolla pyörää pyöritetään ja mahdollisesti kiihdytetään. Mikäli kuitenkin halutaan määrittää nopeus törmäyshetkellä tarkemmin, on ajoneuvon liike toteutettava muulla tavoin. Koska pyörät pyörivät vapaasti, saadaan ajoneuvo liikkeelle key frame -tekniikalla. Mallinnetaan ja käsin animoidaan kappale, joka määritellään simulaatiossa peräänantamattomaksi, ja sijoitetaan se alkuasemaltaan mönkijän taakse. Näin kappale työntää ajoneuvon liikkeelle ja ajoneuvon nopeutta voidaan helposti säädellä.

Mönkijän halutaan törmäävän autoon 40 kilometrin tuntinopeudella eli 11,111 m/s. Tässä animaatiossa käytetään nopeutena 30 kuvaa sekunnissa, joten työntävä kappale määritellään kulkemaan törmäystä edeltävien kolmenkymmenen ruudun aikana 11,111 metriä.

6.3.3 Hahmon rajoittimet

Ihmishahmon ruumiinosien liittämiseen käytetään Rag Doll -rajoitinta, joka automaattisesti kytkee osat toisiinsa ja muodostaa Rigid Body -kokoelman, johon ruumiinosat kuuluvat. Rajoitin ei pyri säilyttämään muotoaan vaan nimensä mukaisesti käyttäytyy räsynuken tavoin. Hahmon on kuitenkin tarkoitus säilyttää asentonsa törmäykseen asti, joten ruumiin osat sisällytetään Fracture-ratkaisijaan. Tämän ratkaisijan sisällä olevat kappaleet säilyttävät keskinäiset asemansa niin kauan kuin halutaan, joten asento saadaan rikkoutumaan törmäyshetkellä.

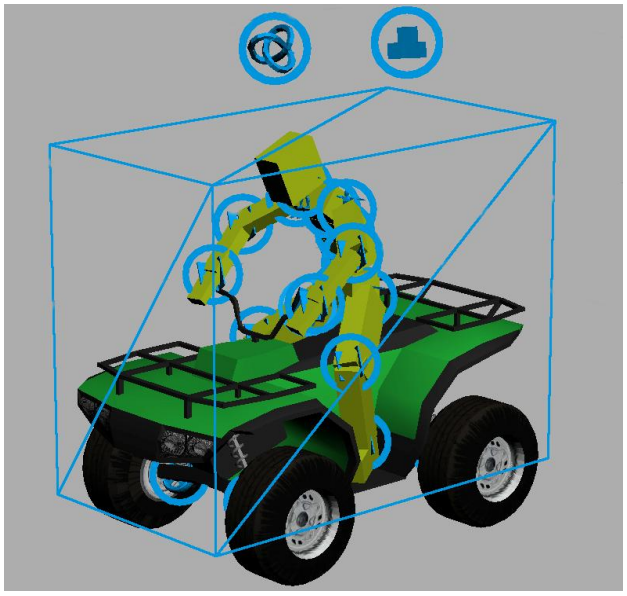
Rag Doll -rajoitin on käytännössä useasta yksittäisestä rajoittimesta koostuva kokonaisuus, jossa liikkeet on määritelty valmiiksi vastaamaan ihmiskehon nivelten liikkeitä (kuva 25). On tärkeätä, että hahmo asetellaan haluttuun asentoon ennen Rag Doll -rajoittimen käyttöönottoa. Mikäli hahmon asentoa muutetaan rajoittimen asettamisen jälkeen, saattaa hahmo käyttäytyä simulaatiossa virheellisesti. Virhe ilmenee, mikäli asento lukitaan paikoilleen edellä kuvatusti Fracture-ratkaisijan avulla, jolloin hahmo alkaa pyöriä akselinsa ympäri simulaation aikana.



Kuva 25. Ruumiinosat liitetty toisiinsa Rag Doll –rajoittimen avulla. (Laine 2007.)

Sekä Mönkijä pyörineen, että ihmishahmon jäsenet on liitetty omiksi kokonaisuuksiksi. Jotta hahmo pysyisi ajoneuvon kyydissä simulaation alkaessa on edellä mainitut kokonaisuudet yhdistettävä. Tämä tapahtuu lisäämällä mönkijä Fracture-ratkaisijaan hahmon kanssa. Pyörät on kuitenkin jätettävä ulkopuolelle, koska kyseinen ratkaisija ei salli niiden pyörimistä. Kaikkien Fracture-ratkaisijaan kuuluvien kappaleiden on kuuluttava samaan dynamiikkakokoelmaan, joten mönkijä pyörineen

sijoitetaan siihen Rigid Body -kokoelmaan, johon hahmon ruumiinosat kuuluvat. (kuva 26.)



Kuva 26. Fracture-ratkaisija, Constraint Solver ja Rigib Body -kokoelma. (Laine 2007.)

6.4 Simulaation testaus ja korjaukset

6.4.1 Alkutilanne

Simulaatioon kuuluvat kappaleet on mallinnettu, niille on asetettu tarvittavat rajoittimet sekä määritelty kappaleiden fysikaaliset ominaisuudet. Testataan simulaation toimivuutta, jolloin voidaan tehdä tarvittavia muutoksia halutun lopputuloksen aikaansaamiseksi. Ensimmäiset kokeilut suoritetaan esikatseluikkunassa, jolloin simulaation etenemistä voidaan tarkkailla reaaliajassa.

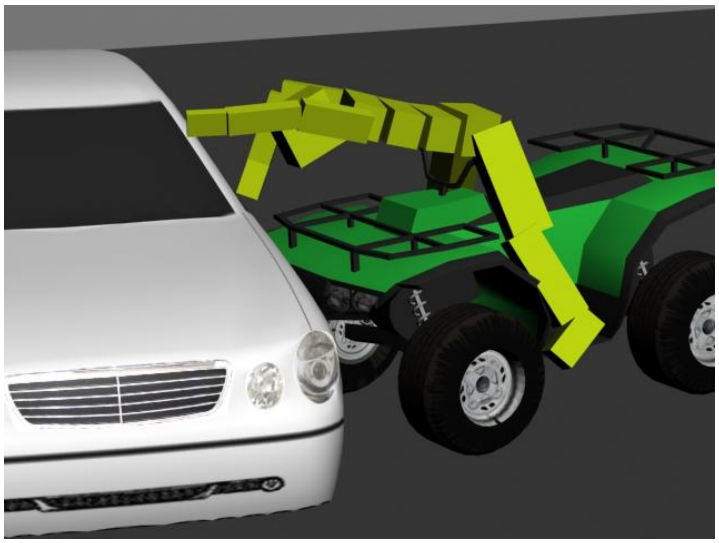
Ensimmäisenä havaintona huomataan, kuinka mönkijän kiihtyessä se alkaa kiertyä sivuun aiottuun kulkusuuntaansa nähden. Koska virhe selvästikin liittyy mönkijälle vauhtia antavaan kappaleeseen, tehdään muutama testi sijoittaen kappale eri korkeuksille mönkijään nähden. Huomataan, että mikäli työntävä kappale on mönkijän painopisteeseen nähden korkealla, kääntyy mönkijä sivulle. Mikäli työntävä kappale on painopisteeseen nähden alhaalla, nousee ajoneuvo takapyöriensä varaan. Mönkijän painopisteen vaikutuksen minimoimiseksi siirretään työntökohtaa eteenpäin. Mönkijän simuloituna geometriana käytettävän kappaleen pohjaan tehdään vauhtia antavaa kappaletta varten

uloke. Tämä uloke sijoittuu hieman etupyörien takapuolelle, jolloin mönkijän painopiste jää sen taakse. Testattaessa havaitaan korjauksen onnistuneen ja ajoneuvo käyttäytyy halutulla tavalla. Ajoneuvojen pyörien kiinnityksiä määrittävät rajoittimet todetaan toimiviksi. Mönkijä kulkeutuu törmäykseen asti halutulla tavalla ja myös auto käyttäytyy halutusti törmäystilanteessa.

6.4.2 Törmäystilanne

Ihmishahmo on saatava putoamaan pois mönkijän päältä törmäyksessä, joten hahmon jäsenet ja mönkijän rungon sisältävän Fracture-objektin hajoamisajankohta tulee määrittää. Koska mönkijälle vauhtia antava kappale pysähtyy ruutuun 80, joka on viimeinen ruutu ennen törmäystä, määritellään Fracture-rakenne hajoamaan kyseisessä avainkehyksessä. Erilaisia hajoamisajankohtia voidaan testata hajottamalla rakenne mainittua aikaisemmin tai myöhemmin. Sulavin ja todenmukaisin liike saavutetaan ajoittamalla Fracture-objektin hajoaminen mahdollisimman lähelle törmäysajankohtaa.

Seuraavana havaintona huomataan hahmon takertuvan kiinni mönkijän ohjaustankoon (kuva 27). Tämä ongelma korostuu erityisesti suoritettaessa simulaatio korkeilla päivitys taajuuksilla. Suoritettaessa systeemin päivitys alle 20 kertaa kuvaa kohden ei käytetyillä suurilla nopeuksilla havaita hahmon vartalon osumaa ohjaustankoon. Hahmo kulkee tangon lävitse törmäyksessä.



Kuva 27. Ohjaustanko estää hahmon vapaan liikkeen. (Laine 2007.)

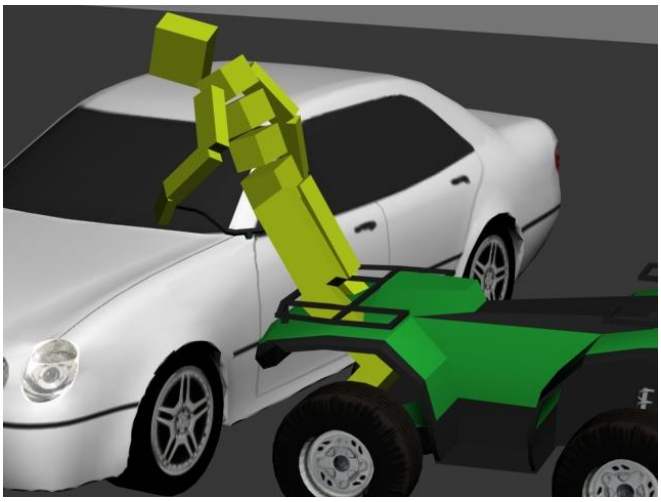
Tilanne voidaan korjata irrottamalla ohjaustanko muusta mönkijän geometrista. Koska mönkijän mallinnusvaiheessa ohjaustanko liitettiin mönkijään erillisenä elementtinä, voidaan se nyt helposti irrottaa erilliseksi kappaleeksi. Tämä kappale lisätään Fracture-objektiin sekä oikeaan Rigid Body -kokoelmaan. Ohjaustangon massaksi tulee irrotettaessa koko mönkijälle määritetty 200 kilogrammaa, joten tangon paino on määritettävä uudelleen. Koska simulaatiossa ratkaisee ainoastaan simuloitu geometria, on myös mönkijän simuloituna geometriana käytettävästä kappaleesta poistettava ohjaustanko.

Jälleen suoritetaan testaus, jolloin voidaan todeta, että ohjaustanko ei enää estä hahmon liikettä törmäyksessä, vaan se irtoaa muusta ajoneuvosta.

6.4.3 Tarkkuuden määrittäminen

Kun näyttämöllä olevien kappaleiden toiminnallisuus on varmistettu, valitaan simulaation tarkkuus lopputuloksen kannalta sopivimmaksi. Oletuksena muodostaa fysiikkamoottori kappaleille avainruudut jokaiseen animaation kehykseen. Systeemi päivitetään kymmenen kertaa avainruutua kohden. Animaation esitysnopeudeksi on tässä esimerkissä valittu 30 kuvaa sekunnissa, jolloin aikayksikköä kohden suoritetaan systeemin päivityksiä enemmän kuin vaikkapa 25:llä kuvalla sekunnissa. Koska törmäysnopeus on varsin suuri, on päivitystiheyden oltava myös suuri, jotta vältyttäisiin toisensa lävistäviltä kappaleilta.

Testattaessa simulaatiota erilaisilla päivitysnopeuksia havaitaan, että päivitettäessä systeemi alle 50 kertaa avainruutua kohden, lävistää hahmon ruumiinjäsenet selvästi mönkijän (kuva 28). Päivitystiheyden tulisi siis olla tätä arvoa suurempi. Kuitenkin suoritettaessa yli 30 päivitystä avainruutua kohden jää hahmo jaloistaan kiinni ajoneuvoonsa. Tapahtumien selvittämiseksi tarkemmin annetaan fysiikkamoottorin laskea kappaleille avainkehykset aikajanalle, jolloin tapahtumia voidaan tutkia ruutu kerrallaan. Huomataan, että hahmon asennon säilyttävä Fracture-kokonaisuus hajoaa korkeilla päivitystiheyksillä liian aikaisin törmäykseen nähden, jolloin hahmo menettää liike-energiaansa ennen törmäystä. Törmäysajankohtaa hienosäädetään aikaisemmaksi siirtämällä autoa muutaman senttimetrin verran mönkijän kulkusuuntaan päin.

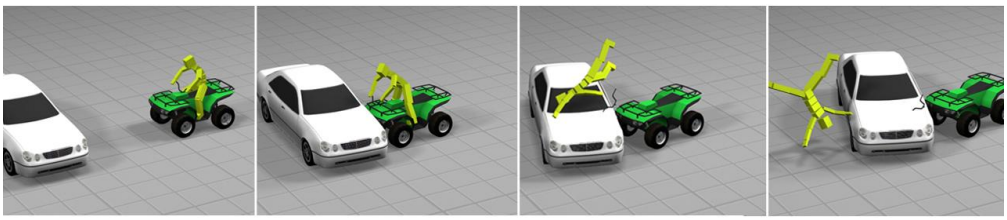


Kuva 28. Pienillä systeeminpäivitys määrillä nopeasti liikkuvat kappaleet lävistävät toisensa. (Laine 2007.)

Auton siirtämisen jälkeen suoritettavassa testissä havaitaan hahmon jatkavan liikettään sulavasti törmäyksen jälkeen. Lopulliseksi systeeminpäivitystiheydeksi määritetään sata kertaa avainkehystä kohden, jolloin toisiaan lävistäviä kappaleita ei ole havaittavissa.

6.5 Viimeistely ja renderöinti

Kun testaamalla havaitut virheet on korjattu ja animaation kokonaispituus sekä simulaation kesto määritelty, voidaan antaa fysiikkamoottorin laskea lopulliset avainkehykset. Näyttämö viimeistellään lopullisen animaation renderöintiä varten määrittämällä valaistus, kamerat, sekä tuomalla näyttämölle mahdolliset simulaatioon kuulumattomat kohteet. Lopuksi renderöintiasetuksista määritellään resoluutio sekä tiedostomuoto koodekkeineen ja valitaan renderöintiohjelma, joka toteuttaa lopullisen animaation (kuva 29).



Kuva 29. Kun liikkeeseen ollaan tyytyväisiä voidaan renderöidä lopullinen animaatio. (Laine 2007.)

7 YHTEENVETO

Fysiikkamoottorin avulla toteutettu liike soveltuu parhaiten vapaasti liikkuvien kappaleiden animointiin. Erilaiset putoavat ja lentävät kappaleet saadaan käyttäytymään painovoiman vaikutuksesta täysin todenmukaisesti. Myöskin esimerkkianimaation kaltaisten törmäystilanteiden kuvaamiseen menetelmä sopii hyvin, koska mallinnettujen kappaleiden liikkeet törmäyksen jälkeen olisi muuten vaikea toteuttaa uskottavasti. Tämä käy hyvin ilmi esimerkkianimaatiossa, jossa monesta yksittäisestä ruumiinosasta koostuvan ihmishahmon liikkeiden toteuttaminen olisi hankalaa. Kaikkien liikkeiden vaikutuksia muihin ruumiinosiin olisi mahdotonta täydellisesti hahmottaa.

Dynamiikkasimulaation käyttö mahdollistaa uskottavan liikkeen toteuttamisen 3d-animaatiossa. Mahdollisimman todenmukaisen lopputuloksen saavuttaminen edellyttää perusteellisesti rakennettua simulaatiota. Animaation suurempi tarkkuus liikkeen osalta edellyttää myös suurempaa laskentatehokkuutta. Siksi lopputulos on usein kompromissi todenmukaisuuden, käytettävissä olevan ajan ja käytössä olevan laitteiston tehokkuuden välillä.

Käsin suoritettava key frame -animaatio perustuu yleensä animaattorin käsitykseen oikeasta liikkeestä kyseisessä tapauksessa. Dynamiikkasimulaation avulla voidaan toteuttaa fysiikan lainalaisuuksiin perustuvaa liikettä, jolloin saadaan aikaiseksi realistista liikettä varsin nopeasti. Vaikka vaatimukset liikkeen todenmukaisuuden suhteen eivät olisikaan kovin korkealla, saavutetaan kyseisellä menetelmällä merkittävä ajansäästö, mikäli animoitavia kappaleita on suuri määrä, jolloin niiden säätäminen yksitellen olisi vaivalloista.

Esimerkkianimaation tapaisen törmäystilanteen toteuttaminen key frame -tekniikalla, ei olisi millään tavalla järkevää, koska todenmukaisten liikkeiden selvittämiseksi jouduttaisiin suorittamaan valtaisa määrä laskutoimituksia. Esimerkin suhteellisen lyhyen ja yksinkertaisen simulaation aikana tunnistettiin törmäysinformaation mukaan 432235 törmäystä. Dynamiikka-animaatio soveltuukin hyvin tilanteisiin, joissa täytyy tarkasti selvittää jonkin tapahtumasarjan kulku. Tällaista animointitekniikkaa voidaankin soveltaa esimerkiksi onnettomuustutkinnassa. Tieteellisissä tarkoituksissa tosin vaaditaan huomattavasti suurempaa tarkkuutta simulaation osalta.

Puutteena voitaisiin mainita työssä käytetyn ohjelmiston kykenemättömyys simuloida todellisia materiaalien ominaisuuksia. Esimerkiksi materiaalin lujuuden mallintaminen toisi laajasti uusia mahdollisuuksia

animointiin. On kuitenkin huomioitava 3ds Max Reactorin käyttötarkoitus, joka on lähinnä viihdekäyttöön suunnatun sisällön tuottaminen. Materiaaliominaisuuksien mallintaminen vaatisi huomattavan paljon laskentaa, joten se edellyttäisi myös tehokasta laitteistoa. Tieteelliseen tutkimukseen ja kehitystyöhön suunnatuista sovelluksista toki edellä mainittukin ominaisuus löytyy. Tietotekniikan kehittymisen myötä onkin mielenkiintoista jäädä odottamaan, miten laajemmalle käyttäjäkunnalle tarkoitettujen sovellusten fysiikan mallinnusominasuudet kehittyvät.

LÄHTEET

Julkaistut materiaalit

Ahoranta, J & Helakorpi, S. 1999. Uusi Omega: Ammatillinen fysiikka. Porvoo: WSOY.

Hautala, M & Peltonen, H. 2002. Insinöörin (AMK) Fysiikka osa I. 6. uudistettu painos. Lahti: Lahden Teho-Opetus Oy.

Hillis, D. 1999. Miten tietokone toimii. Porvoo: WSOY.

Kervinen, M & Smolander, J. 2003. MAOL-taulukot. 1.-4. uudistettu painos. Helsinki: Otava.

Murdock, K. 2005. 3ds max 7 Bible. Hoboken, USA: Wiley Publishing.

Mäkelä, P. & Tiainen, J. 1993. Tietojätti. Jyväskylä: Gummerus.

Salmi, T. 1996. Dynamiikka 1: Kinematiikka. 4. painos. Tampere: Pressus Oy.

Salonen, E-M. 1998. Dynamiikka 1. Helsinki: Otatieto Oy.

Elektroniset materiaalit

Autodesk. 2001. Reactor for 3DS Max 4 Manual [pdf-dokumentti]. [viitattu 22.2.2007].

Autodesk. 2005a. 3ds Max 8 User Reference > Introducing Dynamics Simulation [käyttöohjetiedosto]. [viitattu 16.2.2007].

Autodesk. 2005b. 3ds Max 8 User Reference > World Rollout [käyttöohjetiedosto]. [viitattu 16.2.2007].

Autodesk. 2005c. 3ds Max 8 User Reference > Collisions Rollout [käyttöohjetiedosto]. [viitattu 17.2.2007].

Autodesk. 2005d. 3ds Max 8 User Reference > Storing and Accessing Collisions [käyttöohjetiedosto]. [viitattu 17.2.2007].

Autodesk. 2005e. 3ds Max 8 User Reference > Preview and Animation Rollout [käyttöohjetiedosto]. [viitattu 19.2.2007].

Autodesk. 2005f. 3ds Max 8 User Reference > Rigid Body Properties [käyttöohjetiedosto]. [viitattu 17.2.2007].

Carlson, W. 2003a. A Critical History of Computer Graphics and Animation - Bell labs [verkkojulkaisu]. The Ohio State University [viitattu 14.2.2007]. Saatavissa:
http://accad.osu.edu/~waynec/history/tree/bell.html

Carlson, W. 2003b. A Critical History of Computer Graphics and Animation - Section 10: CAD/CAM/CADD/CAE. [verkkojulkaisu]. The Ohio State University [viitattu 14.2.2007]. Saatavissa:
http://accad.osu.edu/~waynec/history/lesson10.html

Carlson, W. 2003c. A Critical History of Computer Graphics and Animation - Section 3: The computer graphics industry evolves. [verkkojulkaisu]. The Ohio State University [viitattu 14.2.2007]. Saatavissa:
http://accad.osu.edu/~waynec/history/lesson3.html

Franklin, C. 2007. How 3-D Graphics Work - How to Make It Look Like the Real Thing [verkkojulkaisu] [viitattu 14.2.2007]. Saatavissa:
http://computer.howstuffworks.com/3dgraphics3.htm

Havok. 2006. Havok Product Family [verkkojulkaisu]. Havok.com Inc. 1999-2006 [viitattu 8.1.2007]. Saatavissa:
http://www.havok.com/content/view/72/57/

Hotokka, P. 2004. Liukulukulaskenta [verkkojulkaisu] [viitattu 14.2.2007]. Saatavissa:
http://www.cc.jyu.fi/~pejuhoto/opiskelu/liukuluvut.pdf

Lehtinen, J. 2007. 115 vuotta animaatioelokuvaa. [verkkojulkaisu]. [viitattu 17.2.2007]. Saatavissa:
http://www.kvaak.fi/naytajuttu.php?articleID=767

Kuvalähteet

Caltech/JPL. 2007. [verkkojulkaisu]. [viitattu 18.2.2007]. Saatavissa:
http://mars.jpl.nasa.gov/gallery/spacecraft/hires/phoenix_hr.jpg

Carlson, W. 2003a. A Critical History of Computer Graphics and Animation [verkkojulkaisu]. The Ohio State University [viitattu 20.2.2007]. Saatavissa: http://accad.osu.edu/~waynec/history/images/pages/ivan-sutherland_jpg.htm

Carlson, W. 2003b. A Critical History of Computer Graphics and Animation [verkkojulkaisu]. The Ohio State University [viitattu 20.2.2007]. Saatavissa:
http://accad.osu.edu/~waynec/history/tree/images/zajac.JPG

Laine, V. 2007. Lahden ammattikorkeakoulu.

MSC.Software. 2005. MCS.Adams model [verkkojulkaisu]. CEI [viitattu 20.2.2007]. Saatavissa:
http://legacy.ensight.com/aug2005/featured_image.html

LIITTEET

CD-ROM -levy, jossa lopullinen animaatio ja opinnäytetyö pdf-dokumenttina.