Kuura Jänkälä

Pehmytkappaleiden simulointi reaaliajassa

kandidaatintutkielma

18. heinäkuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Kuura Jänkälä

Yhteystiedot: kumajank@jyu.fi

Työn nimi: Pehmytkappaleiden simulointi reaaliajassa

Title in English: Simulation of soft-body objects in realtime

Työ: Kandidaatintutkielma

Sivumäärä: 19+0

Tiivistelmä: Kirjallisuuskatsaus dynaamisesti simulaatiossa muovautuvien kappaleiden mallinnukseen. Katsaus keskittyy reaaliaikaisiin simulaatioihin.

Avainsanat: pehmytkappale, simulointi, mallinnus (3D), deformaatio

Abstract: Review of existing literature about modelling dynamically deformable objects. Focus of the text is on real-time solutions.

Keywords: soft-body, simulation, modeling (3D), deformation

Kuviot

Sisällys

1 Johdanto

Yleisesti simulaatioilla pyritään luomaan jokin tapahtuma, ilman, että tapahtumaa täytyy reaalimaailmassa järjestää. Tarkemmin fysiikkasimulaatioilla tavoitteena on usein joidenkin kappaleiden tai materiaalien tilan, sijainnin, tai muodon muuttumisen esittäminen. Eritysesti muodon ja sijainnin esittäminen on luontevaa tietokoneella.

Kuitenkaan laskentatehoa ei ole loputtomiin, ja mitä enemmän mallin vaaditaan vastaavan todellisuutta, sitä hitaampaa simulaation laskeminen on. Tästä hitaudesta seuraa se, että erityisesti reaaliaikaisia tapahtumia mallintavan laskentatavan on tehtävä kompromisseja taustalla tapahtuvan laskennan tarkkuudesta. Käyttäjän on siis ensisijaisesti pystyttävä seuraamaan tapahtumia reaaliajassa, ja simulaation sisäisen fysiikkalaskennan todenmukaisuus rajoittuu tämän mukaan. Esimerkkinä Unity-pelimoottorin (2024, rigidbody2d.drag) sisältämä ilmanvastus on lineaarinen, vaikka toisen asteen yhtälöä käyttämällä saataisiin realistisempia tuloksia.

Samaan tapaan laskentatehoa voidaan säästää käyttämällä simulaation sisältämistä kappaleista mahdollisimman yksinkertaisia toteutuksia. Helpointa on käyttää muovautumattomia kappaleita (rigidbody, rigid-body). Ne ovat siis kappaleita, jotka voivat vaikuttaa ympäristöönsä törmäämällä toisiinsa, mutta eivät muuta omaa muotoaan simulaation sisäisten voimien seurauksena. PhysX-fysiikkalaskurin dokumentaatio (2024, manual/geometry) jakaa rigidbodyt kahdenlaisiin. Alkeismuodot (primitives) ovat kappaleita, joiden muoto on ennalta annettu, kuten kuutio tai pallo. Lisäksi 3d-mallia seuraava kappale (mesh) auttaa monimutkaisten kappaleiden, kuten maaston, simuloinnissa.

Puutteita on vielä, sillä moni todellisuuden ilmiö vaatii kappaleen itsensä muodon muuttumista. Zhang, Zhong ja Gu (2018) mainitsevat, että lääketieteen vaatimassa leikkaussimuloinnissa on olennaista, että voidaan todenmukaisesti ja reaaliaikaisesti mallintaa pehmytkudosta. Tähän on kehitetty pehmytkappaleet. Pehmytkappale pystyy olennaisesti myös muuttamaan omaa muotoaan kun siihen kohdistuu muusta ympäristöstä vaikutus. Vaikkapa pintajännityksen koossa pitämä vesipisara voi olla järkevä mallintaa pehmytkappaleena.

1.1 Pehmytkappale

Yleisemmin englanninkielisillä sanoilla softbody, soft-body, tai deformable object, tunnettu pehmytkappale on tietokonemalli jollekin kappaleelle, jonka muodon oletetaan muuttuvan ulkoisten vaikutteiden, kuten törmäysvoimien tai paineen seurauksena. Esimerkkejä ovat pesusieni, vesitippa, sekä pehmytkudos. Pehmytkappaleelle kiinnostavia käyttötarkoituksia ovat muunmuassa peli- tai muu viihdegrafiikka, erilaiset opetussimulaatiot, sekä reaalimaailman muuntaminen koneen luettavaksi.

1.2 Sisältöesittely

Dynaamisesti muovautuvat kappaleet täyttävät täydellisen fysiikkalaskennan ja pelimäisen, kovilla kappaleilla toteutetun ympäristön välin. Aiheen ulkopuolelle jäävät kaasujen ja nesteiden simuloinnit (fluid dynamics). Tässä katsauksessa keskitytään erityisesti reaaliaikaisen simulaation haasteisiin, ja siihen kykenevien toteutustapojen esittelyyn. Katsauksessa esitellään ratkaistavia ongelmia pehmytkappaleiden simuloinnissa, tapoja määritellä simuloitava kappale, ja tilanteita, joissa on jokin määrittelytapa on erityisen kiinnostava.

2 Haasteita pehmytkappaleiden simuloinnissa

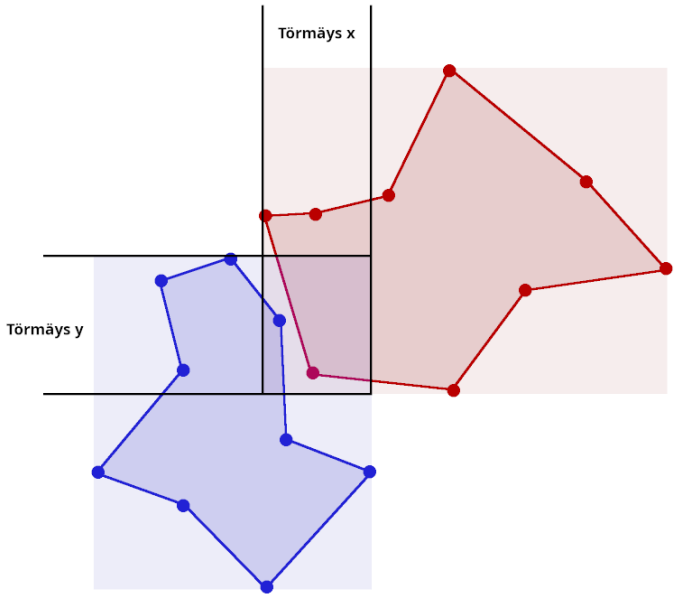
Erityisesti reaaliaikaan tapahtuvissa simulaatioissa korostuu tärkeys sille, että simulaation vaatimat laskutoimitukset minimoidaan. Sekä jousitettua massaa kehittäneet Luo ja Ip (2003) että Muototavoite-mallia tutkivat Müller ym. (2005) huomoivat omien menetelmiensä hyödyksi laskentanopeuden. Simulaatioympäristöissä, kuten Unityssä (2024, MonoBehaviour.FixedUpdate), tämä laskentatehon vaatimus johtuu siitä, että ympäristö päivittyy simulaatioaskeleissa (game tick, simulation step, frame).

Simulaatiossa tapahtuvien muutosten laskenta pyritään siis suorittamaan yhden tällaisen askeleen aikana. Tavoiteltu askelmäärä sekunnissa riippuu käyttötarkoituksesta, ja yleisesti pelisovelluksista tuttu päivitysnopeus on 60 askelta/s. Jos riittävään nopeuteen ei pystytä, simulaation tavoite olla uskottava menetetään. Siksi onkin jo mallinnusmenetelmän valinnassa hylättävä tarkkaan fysiikan ilmiöihin perustuvat simulaatiotavat. Toteutustavasta riippumatta voidaan myös helpottaa laskentaa vähentämällä simuloitavien kappaleiden määrää, tai yksinkertaistaa niiden muotoa. Nämä eivät kuitenkaan ole usein järkeviä, sillä ne myös rajaavat simulaation käyttömahdollisuuksia ja uskottavuutta. Müller ym. (2005) simuloivat jopa satoja kappaleita reaaliajassa.

2.1 Törmäysten tunnistus

Sen lisäksi, että todenmukaisuudesta on osittain luovuttava, on dynaamisesti muovautuvien kappaleiden toteuttamisessa myös muita laskentatehoa vaativia pulmia. Kuten Ganovelli, Dingliana ja O'Sullivan (2000) huomauttavat, ettei monia olemassaolevia tapoja tunnistaa törmäykset simulaatiossa ole suunniteltu käsittelemään kappaleiden muodon muutoksia. Tutkimuksessaan he kehittävät tapaa vähentää vaadittavaa laskentatehoa rajaamalla törmäysten käsittelyä etäisyyden perusteella. Tutkimuksen perustana on, että kaukana toisistaan oleville kappaleille ei ole järkevää verrata tarkkoja yksityiskohtia, sillä voidaan olettaa, että ne eivät ole tarpeeksi lähellä vaikuttaakseen toisiinsa. Heidän kehittämänsä toteutustapa käyttää ulkoisimman törmäyksen tunnistamiseen rajalaatikkoa (bounding box), jonka sisällä kappale on. Kun kahden kappaleen rajalaatikot törmäävät, tiedetään, että niiden yksittäisillä kulmilla

on järkevää tehdä törmäyksentunnistusta.



Kuvio 1. Kaksiulotteisessa avaruudessa sijaitsevien kappaleiden ympärille piirretyt rajalaatikot. Vaikka rajalaatikot törmäävät, laatikoiden sisällä olevat todelliset kappaleet eivät. Kuvio piirretty Ganovelli, Dingliana ja O'Sullivan (2000) tekemän kolmiulotteisen kuvion pohjalta.

Pelkkä törmäyksien mahdollisuuden ennustaminen ei kuintenkaan riitä. Teschner ym. (2004) tutkivat tapoja tehdä tarkempaa muodon mukaista törmäysten käsittelyä. Tekstissään he mainitsevat tunnistamansa keskeiset vaikeudet tätä ongelmaa ratkaistessa.

- Muotoaan muuttava kappale saattaa törmätä itsensä kanssa, kuten köyden solmussa tai kankaan kasautuessa itsensä päälle mytyksi.
- Simulaatiosijantidatan ennakoiminen on vaikeaa, ja datan käsittelyyn olemassaolevat tietorakenteet eivät ole aina soveltuvia pehmytkappaleille.
- Pehmytkappaleiden uskottava törmäystapahtuma vaatii enemmän tietoja kuin muovautumaton esine. Teschner ym. (2004) mainitsevat erityisesti törmäyksen syvyyden.
- Laskentanopeus. Ymmärrettävästi, jotta käyttäjälle voidaan tuottaa hyöty pehmytkappaleen simuloinnista, siihen kohdistuvan vaikuttamisen on tapahduttava riittävän nopeasti.

Myös tarkkaa törmäystunnistusta voidaan tehdä rajalaatikoita käyttämällä. Teschner ym.

2004 kertovat kappaleen törmäyksien käsittelystä rajaamalla sen eri osia omiin kupliinsa (bounding volume). Tästä teknologiasta käytetään lyhennettä BVH, bounding-volume hierarchies. BVH:n alkuperäinen tavoite on ollut kovien kappaleiden (rigidbody) törmäysten tunnistus siten, että keskimääräinen vastausaika törmäystarkastukseen on mahdollisimman lyhyt. BVH:ssa kappaleen ympärille muodostetaan sen ulkoreunoja rajaavia, muodoltaan yksinkertaisempia törmäyskappaleita (collider), joka vähentää vaadittavaa laskentatehoa törmäysten tunnistamiseen. Mitä enemmän näitä yksinkertaisia kappaleita jaetaan kappaleen eri osien ympärille, sen tarkemmin voidaan tunnistaa todellinen muoto.

Pehmytkappaleen BVH:n muodostamisessa on lisähaasteena se, että kappaleen rajaama alue päivittyy mahdollisesti joka simulaatioaskeleella. Samoin kuin koville kappaleille, pehmytkappaleen alustava BVH muodostetaan simulaation aluksi. Yleisin tapa pitää kappaleen rajoja yllä simulaation aikana tehdään muutoksilla olemassaolevaan tietorakenteeseen, sillä uuden rakenteen luominen on hidasta. BVH:n lisähyöty on, että kappaleen sisäiset törmäykset on mahdollista käsitellä samalla tavalla kuin ulkoiset. Kuitenkin on varottava liiallisia törmäyksiä, sillä toistensa lähelle muodostuneet rajalaatikot voivat olla päällekäiset kappaleen ollessa levossa. Teknologia myös sujuvasti tukee jatkuvaa törmäyskäsittelyä (continuous collision), joka poikkeaa normaalista siten, että aika-askelten välissä tapahtunut siirtymä myös huomioidaan. Tämä siksi, että rajalaatikko voidaan rakentaa sisältämään kappaleen nykyinen ja edellinen sijainti. Erityisesty ohuet materiaalit, kuten paperi ja kangas, jotka saattaisivat muuten kulkea toisen kappaleen läpi ja aiheuttaa epäuskottavia simulaatioita, on järkevää mallintaa jatkuvalla törmäyskäsittelijällä, kuten tekee (Teschner ym. 2004).

Kappaleen törmäyskäsittelyyn voidaan myös käyttää todennäköisyyksiin perustuvia menetelmiä. Teschner ym. (2004) perustelevat, että reaaliajassa käyttäjälle näkyvä palaute on saavutettavissa todennäköisyyksiä käyttämällä, ja ihminen ei tunnista merkittävää eroa uskottavan ja todenmukaisen välillä. BVH:n päälle kehitetty todennäköisyysmenetelmä pyrkii arvioimaan törmäävien rajalaatikoiden sisältämän pistejoukon törmäyksiä. Se on yhteensopiva useanlaisten rajalaatikoiden kanssa. Toinen vaihtoehto on sattumanvarainen törmäävien osien etsintä. Tässä todenmukaisuudesta luovutaan alkuvaiheessa, kun ei tarkasti seurata koko kappaleen kaikkia pisteitä. Kuitenkin riittävän kattava peitto (sampling) kappaleen koko pinnalta mahdollistaa toimivan törmäysten havaitsemisen, ja törmäystilanteen havaitsemisen

jälkeen voidaan laajentaa pistemäärää, joiden törmäyksiä tarkkaillaan (Teschner ym. 2004). Alustavan törmäystunnistuksen laskentanopeus on tämän menetelmän merkittävä hyöty.

Ympäristön pilkkominen (spatial subdivision) on tukiteknologia törmäyskäsittelyyn, joka rajaa muutokset käsiteltäviksi 3D-ympäristön pienemmissä osissa. Tämä vaatii, että koko ympäristö tukee pilkkomista, ja on käsiteltävissä soveltuvana tietorakenteena (Teschner ym. 2004). Ympäristö voidaan myös muuntaa kuvaksi, josta sitten tunnistetaan törmäykset. Koska tämä ei vaadi kappaleen rajaamista laatikoihin, tai muuta valmistelua, Teschner ym. (2004) mukaan menetelmä on erityisen hyvin soveltuva pehmytkappaleiden törmäysten havaitsemiseen. Menetelmän heikkous on, että palautettu törmäystieto on epätarkkaa. Se siis saattaa vaatia tueksi jälkikäsittelyä, jos halutaan tietää esimerkiksi törmäyssyvyys.

2.2 Sovelluskohtaisia pulmia

Lloyd ym. (2008) tuovat esiin toisen haasteen, kun pyritään luomaan paitsi uskottavaa, myös todenmukaista pehmytkappaletta simulaatiossa. Koska kappaleiden mallinnuksessa on ensisijaisesti keskitytty laskennan helppouteen ja nopeuteen, on vaikea määrittää sopivat parametrit, joilla saavutetaan jonkin reaalimaailman materiaalin ominaisuudet. He keskittyvät yleisesti käytettyyn jousitettuna massana mallinnettuun kappaleeseen, mutta kyseinen ongelma on myös ilmeinen muissa nopeissa metodeissa, kuten muototavoitteessa.

Muodonmuutoksesta seuraava vastus on myös haaste, jonka ratkaiseminen on käyttötapauskohtaisesti jopa ehdoton. Zhangin, Zhongin ja Gun (2018) tekemässä tutkimuksessa esitellään olennainen kysymys: miten leikkaussimulaattorissa harjoittelevalle käyttäjälle voidaan antaa tuntopalaute (haptic feedback) kun hän vaikuttaa simulaatiokappaleisiin? Ongelma on edelleen avoin. Lisäksi he mainitsevat tavoiteltavat simulaation päivitysnopeudet. Pelkkä graafinen palaute, jossa kappaleen muodon muutos näkyy käyttäjälle, on saavutettava 30Hz/30fps tarkkuudella, mutta tuntopalautteen päivitysnopeus voi olla jopa 1000Hz.

Lisäksi pehmytkappale-mallia käytettäessä teknologioissa, joissa tavoite on käsitellä jotakin todellista esinettä, esiintyy omat ongelmansa. Robottikäden on tunnistettava kappaleessa tapahtuvat muodon muutokset, jotta se pystyy manipuloimaan kappaletta onnistuneesti. Antonova ym. (2021) pyrkivät vastaamaan kysymykseen siitä, miten tekoälylle on järkevää opet-

taa kappaleiden käsittelyä. Heidän simulaatioympäristönsä käyttää pehmytkappaleita opetustyökaluna. Tämän tukena on koneelle oleellisesti mahdollisuus muuntaa reaalimaailman esineet tietokoneymmärrettäviksi malleiksi. Schulman ym. (2013) ovat kehittäneet ongelman ratkaisuksi algoritmin, joka ennustaa kappaleiden muotoa kuvasta ja muuntaa sen tietokonemalliksi.

Stabiilius on ominaisuus, joka korostuu pelisovelluksissa. Pehmytkappaleen on pystyttävä uskottavasti reagoimaan tilanteisiin, jotka eivät vastaa todellisuutta. Stabiilius on menetelmästä riippuva, ja vaikuttaa siihen, miten kappale reagoi suuriin törmäysvoimiin. Epästabiili kappale voi törmäyksessä taipua siten, ettei se palaakkaan törmäyksen jälkeen alkuperäiseen muotoonsa. Esimerkiksi rajoittamaton jousitettu massa on epästabiili. Müller ym. (2005) kehittämä muototavoite-kappale on stabiili.

Toinen hyödyllinen ominaisuus peliympäristöissä on ohjattavuus. Müller ym. (2005) sanovat, että pelinkehittäjän tulisi itse pystyä määrittämään dynaamisen muovautumisen voimakkuus. Ohjattavuus on myös toteutusmenetelmän tasolla ratkaistava pulma. Sovellukset, jotka ovat jo valmiiksi erikoistuneet tiettyyn käyttötarkoitukseen, voivat sisällyttää sopivat parametrit toteutukseensa.

Lisäksi menetelmää valitessa on huomioitava yksityiskohtaisempia erikoistilanteita tapauskohtaisesti. Esimerkkinä kysymys siitä, miten laaja-alainen voima, kuten tuuli, vaikuttaa kappaleeseen. Yksinkertaisimmillaan se voi vaikuttaa suoraan kappaleen keskipistettä siirtämällä, mutta osa toteutustavoista ei tätä mahdollista. Jousitetussa massassa ei ole sopivaa keskipistettä, esimerkiksi Luon ja Ipin (2003) määrittämässä massassa painovoima vaikuttaa joko osaan tai kaikkiin kappaleen pisteisiin tai solmuihin (node).

3 Keskeiset mallinnustavat

Pehmytkappaleiden mallinnukseen on muutamia keskeisiä menetelmiä. Reaaliaikaisia menetelmiä voidaan luokitella vain kappaleen sisäisiä voimia laskeviksi, sekä geometriseen muotoon vertaaviksi malleiksi. Lisäksi on muita, ei yhtä hyvin reaaliaikaan soveltuvia menetelmiä, jotka perustuvat vielä todenmukaisempaan mallinnukseen. Jianping, Feng ja Hock Soon (2016) jakavat luvuittain geometriset, jousitettuun massaan tai muuten pisteparveen pohjautuvat, sekä fysiikan ilmiöihin pohjautuvat menetelmät. On myös näiden päälle rakentuvia erikoistuvia menetelmiä, kuten heidän tutkimansa malli, mikä kykenisi taipumaan epäsymmetrisesti eri suuntiin. Esimerkiksi puu taipuu eri tavoin sen syiden suuntaan, kuin syiden poikki. Tässä luvussa tarkemmin selitän jousitetun massan sekä geometriseen mallin.

3.1 Jousitettu massa

Määritelmältään yksinkertainen, yleisesti käytössä ollut tapa laskea pehmytkappale on mass-spring system, eli jousitettu massa (Jianping, Feng ja Hock Soon 2016). Jousitettu massa pitää muotonsa laskemalla kappaleen sisällä vaikuttavia jousivoimia. Verrattaessa aikaisempiin, enemmän todellisuutta vastaaviin menetelmiin, jousitetun massan vaatima laskenta on nopeampaa (Luo ja Ip 2003).

Jousitetussa massassa kappaleen kulmista tai pisteistä (node) mallinnetaan jousia muihin kulmiin. Luo ja Ip (2003) kertovat oman toteutuksensa kappaleen sisäisten voimien laskemisesta. He päivittävät simulaatioaskeleittain jokaiselle kappaleen pisteelle uuden sijainnin ja kiihtyvyyden. Tämä lasketaan yhdistämällä pisteen kokema painovoima, jousivoimat, ja paine. Paine ei ole jousitetulle massalle ehdoton, vaan kyseisen toteutuksen lisäominaisuus.

Jousitettu massa ei ole itsessään stabiili. Kuten Müller ym. (2005) mainitsevat, ei-haluttujen muodonmuutosten rajoittamista ei huomioida useissa olemassaolevissa toteutuksissa. Lisäksi kappaleen keskipisteen puuttuessa kaikkien manipulaatioiden on tapahduttava vaikuttamalla yksittäisiin pisteisiin. Jousitettu massa on edelleen hyödyllinen toteutuksen helppouden takia, ja soveltuu hyvin esimerkiksi 2-ulotteisiin ympäristöihin, joissa kappaleet voivat olla muodoltaan yksinkertaisia, ja aikavaativuus ei siksi kasva liikaa.

3.2 Geometriapohjainen mallinnus

Toinen, varsinkin peliympäristöihin erityisesti soveltuva menetelmä on sijaintiin pohjautuva laskenta. PBD:ksi lyhennety Position Based Dynamics, eli suoraan kappaleen kulmien sijaintia ohjaava muodon mallinnus on pohja, joka mahdollistaa nopean, reaaliajassa tapahtuvan kappaleiden muotoutumisen. Müller ym. (2007) kertoo PBD:n tuoman hyödyn olevan, että simulaatioaskeleen sisällä on suoritettava vähemmän laskutoimituksia. Tällä toteutuksella vältytään laskemasta kappaleen sisällä kasautuvia voimia, kuten tehdään jousitetussa massassa.

Geometrisen mallin päälle rakentuvat toteutukset tuovat myös muita hyötyjä. Sen avulla voidaan helpommin välttää epävakausongelmat, sillä kappaleen tavoiteltu paluumuoto voidaan pitää muistissa. Lisäksi mahdollisuus sujuvasti hallita kappaleen muotoa ja sijaintia simulaatioympäristön ulkopuolelta on kiinnostava (Müller ym. 2007).

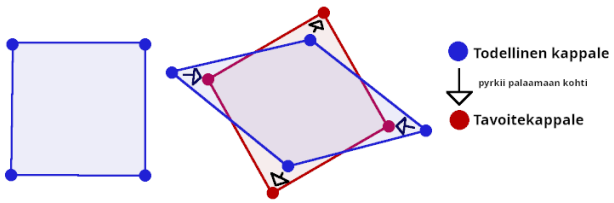
Jianping, Feng ja Hock Soon (2016) listaavat PBD:n laskentatapahtumat seuraavasti:

1. Ensin lasketaan ulkoisten voimien vaikutus, ja tämän avulla ennustetaan kappaleen paikka seuraavaa simulaatioaskelta varten.
2. Ennustettuihin sijainteihin vaikuttaa jokin kappaleen sisäinen rajaus, kuten sen tavoitemuoto.
3. Kappaleen pisteiden nopeudet päivitetään lasketuilla tiedoilla.

Muototavoite-kappale myös minimoi sisäisten voimien laskennan. Müller ym. (2007) mainitsevat, että Müller ym. (2005) tekstissä esitelty laskentatapa tulisi myös tunnistaa sijaintipohjaiseksi, sillä se käyttää vain yhtä kulman sijaintiin vaikuttavaa sisäistä voimaa, eikä siis vaadi sen ratkaisemiseen ylimääräistä laskua. ("They only treat one specialized global constraint and, therefore, do not need a position solver.")

Muototavoite laskee kappaleen todellisen sijainnin poikkeamista tavoitesijaintiin verrattuna. Sen sijaan, että kappaleen kulmille laskettaisiin niihin kohdistuva voima, ja yhdistettäisiin olemassaolevaan liikkeeseen, tämä toteutustapa käyttää geometrisia rajoitteita ja sijaintieroa tavoite- ja todellisten pisteiden välillä. Muototavoitte-menetelmässä kappaleella on siis aina kaksi esitystä. Sen tavoitetila, sekä sen todellinen tila. Tavoitetila on kappaleen lepomuoto,

ja todellinen muoto pyrkii joka simulaatioaskeleella palaamaan lepomuotoonsa. Kuten PBD, tämä toteutus on vakaa riippumatta simulaatiovoimien suuruudesta (Müller ym. 2005).



Kuvio 2. muototavoitte-kappaleen lepotila (vasemmalla) sekä taittunut tila (oikealla), piirretty Müller ym. (2005) mallin mukaan

4 Käyttötapauksia

Peliympäristöissä muototavoite on selkeästi kiinnostavin menetelmä, sillä se on laskentanopeudeltaan kilpailukykyinen, pystyy simuloimaan kappaleita stabiilisti, sekä mahdollistaa simulaation ulkopuolisen muovaamisen, jossa vaikutetaan suoraan tavoitekappaleeseen. Müller ym. (2005) erityisesti huomioivatkin kehittämänsä menetelmän soveltuvuuden tällaiseen ympäristöön. Jianping, Feng ja Hock Soon (2016) mainitsevat, että menetelmä kykenee myös käsittelemään pisteiden asentoa (orientation), mutta ei sen pidemmälle harkitse, missä tämä kyky voisi olla hyödyksi.

Lisäksi Jianping, Feng ja Hock Soon (2016) havaitsevat, että muototavoitteen, ja muutenkin sijaintipohjaisen PBD:n ongelma on, että se ei sujuvasti muunnu mallintamaan todellisia materiaaleja tai ilmiöitä, sillä kappaleen muovautuminen riippuu muunmuassa simulaatioaskeleiden pituudesta. Kirjassaan he pyrkivät kehittämään simulointitavan, joka kykenisi mallintamaan epäsymmetrisesti muovautuvia materiaaleja. Vastaavasti Lloyd ym. (2008) etsivät tapaa yhdistää hyväksyttävän nopea jousitettu massa mallintamaan tunnettujen materiaalien ominaisuuksia. Myös Zhang, Zhong ja Gu (2018) etsivät yhdistelmää reaaliaikaisen ja todenmukaisen välillä, mutta heidän tuloksensa osoittavat, että kyseisessä käyttötapauksessa tämä ongelma on vielä ratkaisematta. Heidän mukaansa vaikka leikkaussimulaattoreissa on käytetty useita menetelmiä, niin lääketieteellisesti riittävän tarkkaa mallinnusta ei ole vielä saavutettu.

Robotiikassa pehmytkappale itsessään ei välttämättä ole kiinnostava. Sitä on jatkosovellettava, kuten Laux, Singh ja Fabisch (2023) ja Antonova ym. (2021) käyttävät pehmytkappaletta opetustyökaluna. Schulman ym. (2013) tuo esiin toisen tapauksen, jossa koneelle on pystyttävä muodostamaan pehmytkappalemalli reaalimaailman esineestä. Tässä käyttötapauksessa Schulman ym. (2013) käyttävät jousitettua massaa, sillä heidän valitsemansa simulaatioympäristö valmiiksi tukee sitä. He kuitenkin ehdottavat, että valinta ei ole ehdoton, vaan muutkin mallinnustavat voivat soveltua tähän. Voidaan silti nähdä, että jousitetun massan toteuttamishelppous tarkoittaa, että se on järkevä vaihtoehto tukemaan muita teknologioita. Lisäksi, koska jousitetussa massassa jokaiseen kappaleen pisteeseen kohdistuvia voimia muokataan, se soveltuu tilanteisiin, joissa ulkoinen voima kuten paine-ero vaikuttaa muotoon.

Kun mallinnettavassa ympäristössä on ennustettamattomissa olevia muutoksia, useimmiten käyttäjän toimesta, on pehmytkappaleita selkeä soveltaa muovautuvien materiaalien simulointiin. Kuitenkaan näin ei aina ole, joten on huomioitava tilanteet, joissa dynaamista muovautumista ei ole järkevää simuloida. 3D-mallinnustyökalu Blenderissä (2024, armatures/introduction) sisäisen luurangon (armature) ja lopulta näkyviin jäävän täyden mallin voi yhdistää painotetusti siten, että mallin on mahdollista taipua ja venyä luiden ympärillä. Keskeisenä käyttötarkoituksena ovat pelihahmojen eleet ja ilmeet. Ilmeen suora muokkaaminen mahdollistaa uskottavamman lopputuloksen. Kovat materiaalit, jotka saattavat lommoutua tai murtua, ovat harvinaisemmin tarvittu tapaus, jossa muovautumiset on järkevämpi tehdä vaihtamalla kappaleen koko geometria törmäyksen seurauksena. Tällaisessa tilanteessa kappaleelle ennen muodonuutosta kuulunut törmäysmuoto (rigidbody collider) voidaan poistaa, lommoutuminen laskea, ja lisätä täysin uusi lommoutumista vastaava kova kappale (rigidbody) simulaatioon. Unitystä (2024, GameObject.AddComponent) löytyy vastaava toiminto , joka mahdollistaa uusien törmäyskappaleiden (collider) lisäämisen.

5 Johtopäätöksiä

Pehmytkappaleiden mallinnuksessa on vielä paljon avointa, ja sekä nopeammat laskentamenetelmät, että edelleen kehittyvä tietokoneiden laskentateho mahdollistaa yhä monimuotoisempia käyttötarkoituksia. Kuten Ganovelli, Dingliana ja O'Sullivan (2000) tuo esille, uudet simulaatioympäristöt voivat mahdollistaa tehokkaamman törmäyskäsittelyn. Tämän myötä nykyisin avoimet ongelmat, kuten lääketieteelliseen käyttöön toivotut simulaattorit, mahdollistuvat. Tutkittavaa on lisäksi edelleen eri mallinnusmenetelmien yhdistämisessä reaalimaailman ilmiöihin. Peliympäristöissä kiinnostavaa on myös simuloitavien kappaleiden määrän lisääminen, enemmän on enemmän. Mahdollinen esimerkki pehmytkappaleiden tulevaisuudesta löytyy verratessa Noita-peliin, ja sen onnistuneeseen putoavan hiekan simulointiin.

Lähteet

2024. https://docs.unity3d.com/ScriptReference.

2024. https://gameworksdocs.nvidia.com/PhysX/4.1/documentation.

2024. https://docs.blender.org/manual.

Antonova, Rika, Peiyang Shi, Hang Yin, Zehang Weng ja Danica Kragic Jensfelt. 2021. "Dynamic Environments with Deformable Objects". Teoksessa *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=WcY35wjmCBA.

Ganovelli, Fabio, John Dingliana ja Carol O'Sullivan. 2000. "BucketTree: Improving Collision Detection Between Deformable Objects". https://api.semanticscholar.org/CorpusID:900409.

Jianping, Cai, Lin Feng ja Seah Hock Soon. 2016. *Graphical Simulation of Deformable Models*. Springer International Publishing. https://doi.org/10.1007/978-3-319-51031-6.

Laux, Melvin, Chandandeep Singh ja Alexander Fabisch. 2023. "Grasping 3D Deformable Objects via Reinforcement Learning: A Benchmark and Evaluation". https://api.semanticscholar.org/CorpusID:260976880.

Lloyd, Bryn A., S. Kirac, Gábor Székely ja Matthias Harders. 2008. "Identification of Dynamic Mass Spring Parameters for Deformable Body Simulation". Teoksessa *Eurographics*. https://api.semanticscholar.org/CorpusID:2413854.

Luo, Haoxin ja Brandon Ip. 2003. "Real Time Simulation of Soft Objects Using Mass-Spring System". https://api.semanticscholar.org/CorpusID:84833313.

Müller, Matthias, Bruno Heidelberger, Marcus Hennix ja John Ratcliff. 2007. "Position based dynamics". *Journal of Visual Communication and Image Representation* 18 (2): 109–118. ISSN: 1047-3203. https://doi.org/https://doi.org/10.1016/j.jvcir.2007.01.005.

Müller, Matthias, Bruno Heidelberger, Matthias Teschner ja Markus Gross. 2005. “Meshless deformations based on shape matching”. *ACM Trans. Graph.* (New York, NY, USA) 24, numero 3 (heinäkuu): 471–478. ISSN: 0730-0301. https://doi.org/10.1145/1073204.1073216.

Schulman, John, Alex Lee, Jonathan Ho ja Pieter Abbeel. 2013. “Tracking deformable objects with point clouds”. Teoksessa *2013 IEEE International Conference on Robotics and Automation,* 1130–1137. https://doi.org/10.1109/ICRA.2013.6630714.

Teschner, M., S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani ym. 2004. “Collision Detection for Deformable Objects”. Teoksessa *Eurographics 2004 - STARs.* Eurographics Association. https://doi.org/10.2312/egst.20041028.

Zhang, Jinao, Yongmin Zhong ja Chengfan Gu. 2018. “Deformable Models for Surgical Simulation: A Survey”. *IEEE Reviews in Biomedical Engineering* 11:143–164. https://doi.org/10.1109/RBME.2017.2773521.