SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1309

VIZUALIZACIJA SIMULACIJE DEFORMABILNIH TIJELA

Hana Ujčić

Zagreb, lipanj 2026.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1309

VIZUALIZACIJA SIMULACIJE DEFORMABILNIH TIJELA

Hana Ujčić

Zagreb, lipanj 2026.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. ožujka 2026.

DIPLOMSKI ZADATAK br. 1309

Pristupnica:	**Hana Ujčić (0036534022)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentorica:	prof. dr. sc. Željka Mihajlović

Zadatak: **Vizualizacija simulacije deformabilnih tijela**

Opis zadatka:

U suvremenoj računalnoj grafici simulacija elastičnih i plastičnih deformacija tijela predstavlja jedan od ključnih izazova, posebice kada je riječ o postizanju vizualnog realizma u interaktivnom okruženju. Simulacija mekih tijela i tkanina poseban su izazov. U radu je potrebno proučiti dinamiku temeljenu na poziciji PBD (engl. Position-Based Dynamics) kao robustan pristup simulaciji fizike. Posebno proučiti i noviju inačicu proširene dinamike temeljene na poziciji XPBD (engl. Extended Position-Based Dynamics) te usporediti ove metode. Razraditi proučene postupke, ostvariti implementaciju i okruženje pogodno za usporedbu postupaka. Posebno obratiti pažnju na definiranje realističnih ograničenja koja se temelje na energiji. Na nizu primjera ostvariti ispitivanje dobivenih rezultata. Analizirati i ocijeniti postignute rezultate. Diskutirati upotrebljivost rješenja kao i moguća proširenja. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 26. lipnja 2026.

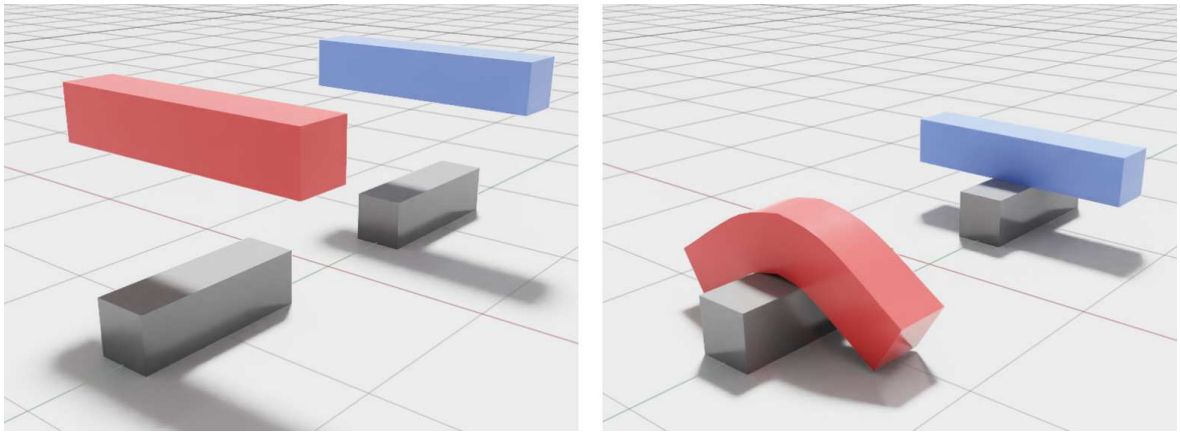
Hvala mentorici prof. dr. sc. Željki Mihajlović na svoj pomoći i strpljenju. Hvala mojim prijateljima i kolegama što su uvijek tu za mene i voljni slušati sve moje probleme. Posebno se zahvaljujem svojim roditeljima, sestri te cijeloj obitelji na podršci i savjetima tijekom cijelog studija.

So long, and thanks for all the fish!

Sadržaj

1. Uvod

Dinamika mekih ili elastičnih tijela predstavlja jedno od temeljnih područja fizikalno utemeljenih simulacija u računalnoj grafici. Za razliku od sustava krutih tijela, kod kojih objekti zadržavaju svoj oblik tijekom gibanja, simulacija mekih tijela modelira deformabilne materijale čija se geometrija mijenja pod djelovanjem vanjskih i unutarnjih sila (Slika 1.1.). Primjeri uključuju tkaninu, gumu, tkivo, užad, želatinozne materijale i deformabilne objekte poput jastuka ili mekanog terena. Realistična simulacija takvih materijala postala je iznimno važna u suvremenoj računalnoj grafici, videoigrama, virtualnoj stvarnosti, robotici i medicinskim simulacijama, gdje su vizualna uvjerljivost, fizička stabilnost i izvođenje u stvarnom vremenu ključni zahtjevi.



Slika 1.1. Razlika između deformabilnog i krutog tijela pri sudaru. Deformabilno tijelo (crveno) se mijenja pod utjecajem sila (sudar s drugim objektom), dok kruto tijelo (plavo) ne mijenja svoj oblik.

Rani pristupi simulaciji deformabilnih tijela temeljili su se na sustavima masa-opruga i metodama mehanike kontinuuma. Sustavi masa-opruga [1] bili su jednostavni za implementaciju i računalno učinkoviti, ali su bili numerički nestabilni i teško su održavali očuvanje volumena i krutosti materijala. Kasnije su razvijene preciznije metode poput metode konačnih elemenata (FEM) [2], koje omogućuju realističnije simulacije defor-

macija, ali uz značajno veće računalne zahtjeve.

S razvojem interaktivnih aplikacija i videoigara pojavila se potreba za metodama koje daju prednost stabilnosti i učinkovitosti izvođenja u stvarnom vremenu. Jedna od razvijenih metoda za rješavanje tog problema je dinamika temeljena na poziciji (Position-Based Dynamics – PBD) [3]. PBD ograničenja provodi izravno nad pozicijama čestica pomoću iterativnih korekcija, čime postiže visoku stabilnost i robusnost čak i pri većim vremenskim koracima.

Iako je PBD postao vrlo popularan u simulaciji tkanine, užadi i mekih tijela, jedna od većih mana je ovisnost krutosti tijela o parametrima simulacije poput vremenskog koraka, što otežava rukovanje i podešavanje simulacije. Zato je razvijena je proširena dinamika temeljena na poziciji (Extended Position-Based Dynamics – XPBD) [4], koja nadograđuje na osnovnu PBD metodu i rješava njene glavne probleme, dok još uvijek zadržava njenu stabilnost i robusnost.

Ovaj rad bavi se matematičkim osnovama, implementacijom i analizom metoda PBD i XPBD za simulaciju mekih tijela. Poseban naglasak stavljen je na formulaciju i izračun ograničenja, uključujući ograničenja udaljenosti, volumena i sudara, kao i na iterativne metode koje se koriste za njihovo rješavanje.

2. Postupci numeričke integracije

2.1. Newtonova mehanika

Simulacija gibanja deformabilnih tijela temelji se na Newtonovim zakonima gibanja. Za sustav čestica Newtonov drugi zakon može se zapisati u matričnom obliku kao:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}, \tag{2.1}$$

gdje je:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix},$$

vektor generaliziranih koordinata sustava, a $\ddot{\mathbf{x}}$ njegova druga derivacija, odnosno vektor akceleracija,

$$\mathbf{M} = \begin{bmatrix} m_1\mathbf{I} & & \\ & \ddots & \\ & & m_n\mathbf{I} \end{bmatrix},$$

je matrica masa, a $\mathbf{f}$ vektor sila koje djeluju na sustav.

2.1.1. Vanjske i unutarnje sile

Sile koje djeluju na sustav mogu se podijeliti na vanjske i unutarnje sile:

$$\mathbf{f} = \mathbf{f}_{ext} + \mathbf{f}_{int}.$$

Vanjske sile djeluju iz okoline na cijeli sustav. Primjeri vanjskih sila su gravitacija, sila vjetra i sila sudara.

Unutarnje sile nastaju zbog međudjelovanja čestica unutar sustava. Primjeri uključuju elastične sile opruga, sile savijanja i sile očuvanja volumena deformabilnog tijela.

Kada su poznate sve sile koje djeluju na sustav, iz Newtonovog zakona (2.1) može se izračunati akceleracija:

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{f}. \tag{2.2}$$

Za pojedinu česticu ovaj izraz odgovara poznatom obliku:

$$a = \frac{F}{m}. \tag{2.3}$$

2.2. Numerička integracija

Nakon izračuna akceleracija potrebno je numerički integrirati jednadžbe gibanja kako bi se dobile nove brzine i položaji čestica.

Općeniti oblik jednadžbe gibanja može se zapisati kao:

$$\ddot{\mathbf{x}} = f(\dot{\mathbf{x}}, \mathbf{x}, t), \tag{2.4}$$

gdje su $\ddot{\mathbf{x}}$ i $\dot{\mathbf{x}}$ druga i prva derivacija položaja $\mathbf{x}$ po vremenu.

Jednadžba drugog reda može se rastaviti na sustav jednadžbi prvog reda:

$$\dot{\mathbf{v}} = f(\mathbf{v}, \mathbf{x}, t), \tag{2.5}$$

$$\dot{\mathbf{x}} = \mathbf{v}, \tag{2.6}$$

gdje je $\mathbf{v} = \dot{\mathbf{x}}$ brzina čestice.

Cilj numeričke integracije jest aproksimirati vrijednosti položaja i brzine u diskretnim vremenskim trenucima:

$$\mathbf{x}(0), \mathbf{x}(\Delta t), \mathbf{x}(2\Delta t), ...$$

2.3. Eksplicitni Eulerov postupak

Najjednostavnija metoda numeričke integracije je eksplicitni Eulerov postupak. Diskretizacijom derivacija dobivamo:

$$\mathbf{a}_n = \frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} \qquad \Rightarrow \qquad \mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t \tag{2.7}$$

$$\mathbf{v}_n = \frac{\mathbf{x}_{n+1} - \mathbf{x}_n}{\Delta t} \qquad \Rightarrow \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t \tag{2.8}$$

Brzina i položaj u koraku $n + 1$ mogu se zapisati kao:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + g(t_n, \mathbf{x}_n, \mathbf{v}_n)\Delta t, \tag{2.9}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + f(t_n, \mathbf{v}_n)\Delta t. \tag{2.10}$$

Eksplicitni Eulerov postupak jednostavan je za implementaciju, ali nije bezuvjetno stabilan. Stabilnost metode ovisi o veličini vremenskog koraka Δt, koji za krute sustave mora biti vrlo malen.

2.4. Implicitni Eulerov postupak

Problem stabilnosti može se ublažiti korištenjem implicitnog Eulerovog postupka, u kojem se vrijednosti u koraku $n + 1$ koriste s obje strane jednadžbe:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + g(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{v}_{n+1})\Delta t, \tag{2.11}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + f(t_{n+1}, \mathbf{v}_{n+1})\Delta t. \tag{2.12}$$

Implicitni Eulerov postupak stabilan je i za velike vremenske korake, ali zahtijeva rješavanje sustava algebarskih jednadžbi u svakom koraku simulacije.

2.5. Poluimplicitni Eulerov postupak

Jednostavno poboljšanje eksplicitnog Eulerovog postupka jest poluimplicitni Eulerov postupak. U ovoj metodi nova brzina prvo se izračuna eksplicitno, a zatim koristi za računanje novog položaja:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + g(t_n, \mathbf{x}_n, \mathbf{v}_n)\Delta t, \tag{2.13}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + f(t_n, \mathbf{v}_{n+1})\Delta t. \tag{2.14}$$

Iako je metoda i dalje eksplicitna, pokazuje bolju numeričku stabilnost od standardnog eksplicitnog Eulerovog postupka uz zadržavanje male računalne složenosti.

3. Klasične metode simulacije deformabilnih tijela

Prije razvoja metoda temeljenih na poziciji, simulacija deformabilnih tijela uglavnom se temeljila na metodama koje proizlaze iz Newtonove i Lagrangeove mehanike. U tim metodama gibanje sustava određuje se integracijom sila i akceleracija kroz vrijeme.

Glavna ideja ovih pristupa jest opisati deformabilni objekt pomoću kontinuiranog ili diskretnog modela te zatim izračunati sile koje djeluju na objekt. Dobivene sile koriste se za računanje akceleracija primjenom Newtonovog zakona:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}$$

Ovisno o načinu modeliranja objekta i sila razlikuje se više pristupa.

3.1. Model opruga i masa

Model opruga i masa (*Mass-Spring Model*) jedan je od najjednostavnijih i najčešće korištenih pristupa za simulaciju deformabilnih tijela. Objekt se modelira kao skup čestica povezanih idealnim elastičnim oprugama.

Svaka čestica posjeduje masu i položaj, dok opruge definiraju elastične veze između susjednih čestica. Deformacija objekta nastaje rastezanjem ili kompresijom opruga.

Elastična sila opruge definira se Hookeovim zakonom:

$$\mathbf{F}_{opruge} = -k_s \mathbf{\Delta x} \tag{3.1}$$

gdje je k_s konstanta krutosti opruge, a $\mathbf{\Delta x}$ pomak iz ravnotežnog položaja.

Budući da bi idealna opruga oscilirala beskonačno dugo, u model se uvodi i sila prigušenja koja smanjuje oscilacije sustava:

$$\mathbf{F}_{prigusenja} = -b\mathbf{v} \tag{3.2}$$

gdje je b koeficijent prigušenja, a $\mathbf{v}$ relativna brzina čestica.

Za realističnu simulaciju potrebno je uključiti i vanjske sile poput gravitacije:

$$\mathbf{F}_g = m\mathbf{g} \tag{3.3}$$

gdje je m masa čestice i $\mathbf{g}$ ubrzanje sile teže

Ukupna sila koja djeluje na česticu jednaka je zbroju svih sila:

$$\mathbf{F}_{uk} = \mathbf{F}_g + \mathbf{F}_{opruge} + \mathbf{F}_{prigusenja} \tag{3.4}$$

Nakon izračuna ukupne sile akceleracija se dobiva primjenom Newtonovog zakona:

$$\mathbf{a} = \frac{\mathbf{F}_{uk}}{m}$$

Model opruga i masa jednostavan je za implementaciju i računalno učinkovit, zbog čega se dugo koristio u simulacijama tkanine, užadi i mekih tijela. Međutim, ponašanje sustava snažno ovisi o parametrima opruga i vremenskom koraku integracije, što može dovesti do numeričke nestabilnosti.

3.2. Metoda konačnih elemenata

Metoda konačnih elemenata (*Finite Element Method – FEM*) predstavlja deformabilni objekt kao kontinuirani volumen diskretiziran na konačan broj elemenata. Za razliku od modela opruga i masa, gdje su sile definirane lokalnim vezama između čestica, FEM opisuje deformaciju pomoću kontinuirane mehanike materijala.

Kontinuirano gibanje elastičnog tijela opisano je jednadžbom očuvanja količine gi-

banja:

$$\rho\ddot{\mathbf{x}} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} \tag{3.5}$$

gdje je ρ gustoća materijala, $\boldsymbol{\sigma}$ tenzor naprezanja i $\mathbf{f}$ vektor vanjskih sila.

Kako bi se jednadžba mogla riješiti numerički, kontinuirani volumen diskretizira se na mrežu konačnih elemenata. Položaj unutar elementa aproksimira se interpolacijom čvorova:

$$\mathbf{x}(m,t) = \sum_i \mathbf{x}_i(t) N_i(m) \tag{3.6}$$

gdje je $\mathbf{x}_i$ položaj i-tog čvora i $N_i(m)$ interpolacijska funkcija oblika (*shape function*).

Iz interpolirane funkcije položaja može se izračunati deformacija materijala. Polje deformacija opisuje se tenzorom deformacije:

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right) \tag{3.7}$$

gdje je $\mathbf{u}$ polje pomaka.

Naprezanje i deformacija povezani su konstitutivnim zakonima materijala. Za linearno elastične materijale koristi se Hookeov zakon:

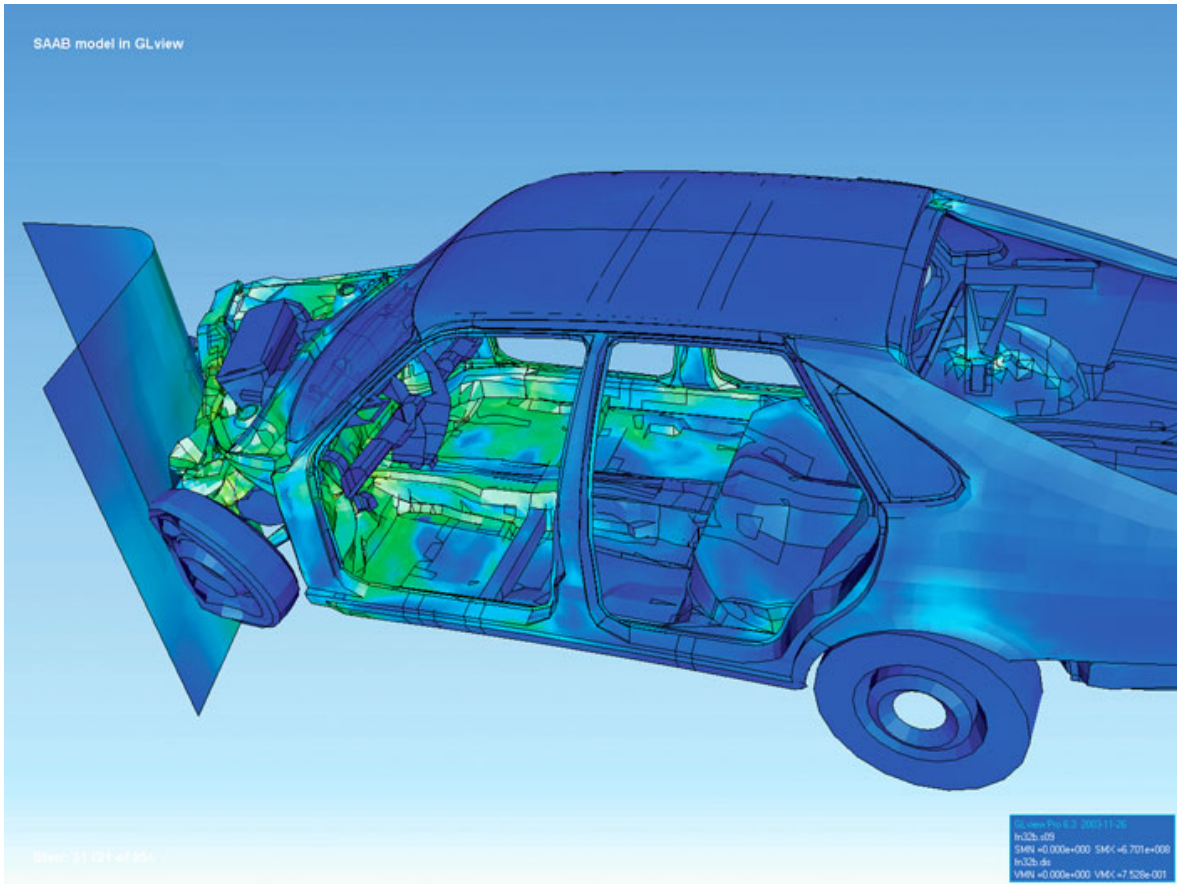
$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon} \tag{3.8}$$

gdje je $\mathbf{C}$ matrica elastičnosti materijala.

Ukupna energija deformacije tijela može se zapisati kao:

$$E = \int_V \boldsymbol{\varepsilon}\boldsymbol{\sigma}\, dV \tag{3.9}$$

Metoda konačnih elemenata omogućuje vrlo preciznu simulaciju deformabilnih tijela i široko se koristi u inženjerstvu i biomehanici. Nedostatak metode jest velika računalna složenost, posebno kod nelinearnih deformacija i velikih sustava.



Slika 3.1. Simulacija sudara auta koristeći FEM metodu.

4. Dinamika temeljena na poziciji

Dinamika temeljena na poziciji (eng. *Position Based Dynamics* – PBD) je popularna metoda za simulaciju deformabilnih tijela u stvarnom vremenu zbog svoje jednostavnosti, robusnosti i numeričke stabilnosti. Za razliku od prije navedenih metoda koje računaju sile i akceleracije kako bi simulirale gibanje tijela, metoda PBD radi direktno s pozicijama čestica. Takav pristup omogućuje stabilnije simulacije čak i pri većim vremenskim koracima, čime se smanjuje mogućnost pojave numeričkih nestabilnosti i prekoračenja (eng. *overshooting*) tijela tijekom simulacije. Zbog toga je metoda posebno pogodna za simulaciju mekih tijela u stvarnom vremenu, poput videoigara i interaktivnih aplikacija.

Prije objašnjavanja samog algoritma potrebno je definirati način reprezentacije deformabilnog objekta. PBD definira diskretni objekt pomoću skupa od N čestica i M ograničenja koja određuju na koji se način čestice smiju gibati.

Svaka čestica $i \in \{1, ..., N\}$ ima slijedeće atribute:

- pozicija $\mathbf{x}_i$
- brzina $\mathbf{v}_i$
- masa m_i

U praksi se često koristi inverzna masa $w_i = 1/m_i$ jer pojednostavljuje formulaciju korekcija ograničenja. Čestice koje predstavljaju statične ili fiksirane dijelove objekta imaju masu beskonačne vrijednosti, odnosno:

$$w_i = 0$$

Tijekom računanja nove vrijednosti pozicije se koristi procjena pozicije $\mathbf{p}_i$. Ova vrijednost služi kao međukorak pri računanju nove pozicije, te se ispravlja kako bi zadovo-

ljila sva ograničenja.

Ograničenje $j \in \{1, ..., M\}$ definirano je funkcijom C_j, koja djeluje na skup čestica $\{i_1, ..., i_{n_j}\}$ te parametrom krutosti

$$k_j \in [0, 1]$$

Formulacija i računanje ograničenja detaljnije će biti obrađene u potpoglavlju 4.1.

Algoritam PBD metode prikazan je sljedećim algoritmom:

Algorithm 1: Algoritam dinamike temeljene na poziciji

foreach *particle i* **do**
- $\mathbf{x}_i \leftarrow \mathbf{x}_i^0$;
- $\mathbf{v}_i \leftarrow \mathbf{v}_i^0$;
- $w_i \leftarrow 1/m_i$;

for *every timestep* Δt **do**
- **foreach** *particle i* **do**
  - $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \, w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$;
  - $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \, \mathbf{v}_i$;
- **for** $k \leftarrow 1$ **to** *solverIterations* **do**
  - **foreach** *constraint* $C_j \in \{C_1, ..., C_{M+M_{coll}}\}$ **do**
    - **if** *constraint is broken* **then**
      - compute $\Delta\mathbf{x}$ using Eq.4.3
      - $\mathbf{p}_i \leftarrow \mathbf{p}_i + \Delta\mathbf{x}$
- **foreach** *particle i* **do**
  - $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$;
  - $\mathbf{x}_i \leftarrow \mathbf{p}_i$;

Na početku algoritma svim česticama postavljaju se početne vrijednosti pozicija, brzina i masa. Iz geometrije objekta i odnosa između čestica generiraju se odgovarajuća ograničenja koja definiraju ponašanje sustava tijekom simulacije.

U svakom vremenskom koraku Δt prvo se računaju nove brzine i pozicije koristeći

poluimplicitni Eulerov postupak (formule 2.13).

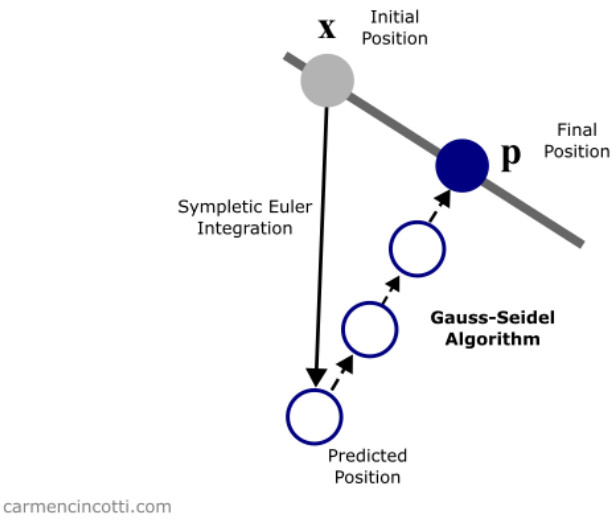
$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \, w_i \mathbf{f}_{ext}(\mathbf{x}_i).$$

gdje su $\mathbf{f}_{ext}$ vanjske sile poput gravitacije ili vanjskih impulsa.

$$\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \, \mathbf{v}_i.$$

Važno je naglasiti da se u ovom koraku ne računaju konačne pozicije čestica, već samo njihove predviđene vrijednosti. Takve pozicije još uvijek mogu kršiti neka od zadanih ograničenja sustava.

U sljedećem koraku algoritma se kreće u iterativno rješavanje ograničenja, gdje se ispravljaju predviđene pozicije tako da poštuju sva ograničenja. Za svako ograničenje provjerava se je li prekršeno te ako ograničenje nije zadovoljeno, pozicije čestica se korigiraju projekcijom na prostor dozvoljenih rješenja. Korekcije se računaju koristeći gradijent ograničenja i inverzne mase čestica, pri čemu čestice manje mase prolaze kroz veće korekcije. Postupak rješavanja ograničenja izvodi se iterativno korištenjem Gauss–Seidelovog rješavača (detaljnije u potpoglavlju 4.1.), gdje se nakon svake korekcije odmah ažuriraju procjene pozicija čestica (Slika 4.1.). Takav pristup omogućuje brzo približavanje rješenju uz relativno mali broj iteracija.



Slika 4.1. Jedan korak PBD algoritma. [5]

Rješavanjem jednog ograničenja je moguće narušiti drugo, pa se ovaj postupak ponavlja u petlji kako bi se osiguralo da sva ograničenja budu ispoštovana. Broj iteracija rješavača u petlji je definiram varijablom $solverIterations$.

Nakon završetka iterativnog rješavanja ograničenja računaju se nove brzine koristeći razliku između stare i korigirane pozicije:

$$\mathbf{v}_i \leftarrow \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t}, \tag{4.1}$$

te se kraju vremenskog koraka pozicije čestica ažuriraju na korigirane vrijednosti:

$$\mathbf{x}_i \leftarrow \mathbf{p}_i \tag{4.2}$$

Za razliku od klasičnih metoda temeljenih na silama, kod PBD metode brzine se ne koriste za direktno zadovoljavanje ograničenja, već se izvode iz već korigiranih pozicija. Upravo ta karakteristika omogućuje visoku stabilnost metode i čini PBD pogodnim za simulacije u stvarnom vremenu.

4.1. Računanje ograničenja

Ograničenja predstavljaju temeljni koncept metode PBD jer definiraju fizička pravila koja sustav mora zadovoljiti tijekom simulacije. Umjesto modeliranja gibanja pomoću sila i akceleracija, metode temeljene na poziciji koriste ograničenja kako bi direktno kontrolirale dozvoljene pozicije čestica. Na taj način moguće je modelirati različite fizičke pojave poput očuvanja udaljenosti između čestica, očuvanja volumena, savijanja materijala i detekcije sudara.

Svako ograničenje $j \in \{1, ..., M\}$ definirano je funkcijom ograničenja C_j koja djeluje na skup čestica $\{i_1, ..., i_{n_j}\}$ te parametrom krutosti

$$k_j \in [0, 1].$$

Funkcija ograničenja opisuje fizički uvjet koji sustav mora zadovoljiti. Ovisno o vrsti

ograničenja, funkcija može predstavljati udaljenost između čestica, volumen deformabilnog tijela, kut između bridova ili uvjet zabrane penetracije između objekata.

Matematički, ograničenja se mogu zapisati kao ograničenja jednakosti:

$$C_j(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_{n_j}}) = 0,$$

ili kao ograničenja nejednakosti:

$$C_j(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_{n_j}}) \geq 0.$$

Ograničenja jednakosti koriste se kada neka veličina mora biti strogo očuvana. Primjer takvog ograničenja je očuvanje udaljenosti između dviju čestica kod simulacije tkanine ili užadi. Ograničenja nejednakosti najčešće se koriste kod detekcije sudara, gdje je potrebno osigurati da objekti ne prodiru jedan kroz drugi.

Nakon izračuna predviđenih pozicija čestica provjerava se jesu li neka ograničenja narušena. Ako je ograničenje prekršeno, potrebno je izračunati korekcije pozicija čestica kako bi sustav ponovno zadovoljio zadani uvjet. Cilj rješavanja ograničenja jest pronaći korekciju pozicija $\Delta\mathbf{x}$ takvu da korigirane pozicije zadovoljavaju:

$$C(\mathbf{x} + \Delta\mathbf{x}) = 0.$$

Budući da je većina ograničenja nelinearna, funkcija ograničenja aproksimira se pomoću prvog člana Taylorovog reda:

$$C(\mathbf{x} + \Delta\mathbf{x}) \approx C(\mathbf{x}) + \nabla C(\mathbf{x}) \cdot \Delta\mathbf{x} = 0.$$

Gradijent funkcije ograničenja $\nabla C(\mathbf{x})$ predstavlja smjer najveće promjene funkcije ograničenja te definira smjer u kojem je potrebno korigirati pozicije čestica kako bi se smanjilo kršenje ograničenja.

Za pronalazak korekcije koristi se metoda Lagrangeovih multiplikatora. Korekcija

pozicije definira se izrazom:

$$\Delta \mathbf{x} = \lambda \mathbf{M}^{-1} \nabla C(\mathbf{x})^T \tag{4.3}$$

gdje je λ Lagrangeov multiplikator, $\mathbf{M}$ matrica masa sustava te $\nabla C(\mathbf{x})$ gradijent funkcije ograničenja.

Matrica masa određuje koliko pojedina čestica sudjeluje u korekciji. Čestice manje mase prolaze kroz veće korekcije, dok čestice beskonačne mase ostaju nepomične.

Ako se korekcija ograniči na smjer gradijenta ograničenja, za pojedinu česticu dobiva se:

$$\Delta \mathbf{x}_i = -\lambda \, w_i \, \nabla_{\mathbf{x}_i} C(\mathbf{x})^T \tag{4.4}$$

gdje je

$$w_i = \frac{1}{m_i}$$

inverzna masa čestice.

Uvrštavanjem izraza 4.4 u linearnu aproksimaciju ograničenja dobiva se izraz za Lagrangeov multiplikator:

$$\lambda = \frac{C(\mathbf{x})}{\sum_j w_j \left| \partial C(\mathbf{x}) / \partial \mathbf{x_j} \right|^2} \tag{4.5}$$

Općenitiji oblik može se zapisati pomoću matrice masa:

$$\lambda = \frac{C(\mathbf{x})}{\nabla C(\mathbf{x}) \mathbf{M}^{-1} \nabla C(\mathbf{x})^T} \tag{4.6}$$

Izračunati multiplikator koristi se za određivanje korekcija pozicija svih čestica koje sudjeluju u ograničenju. Sada se računaju korekcije pozicija $\Delta \mathbf{x}$ te se primjenjuje krutost k. Najjednostavniji način je samo pomnožiti krutost s korekcijom:

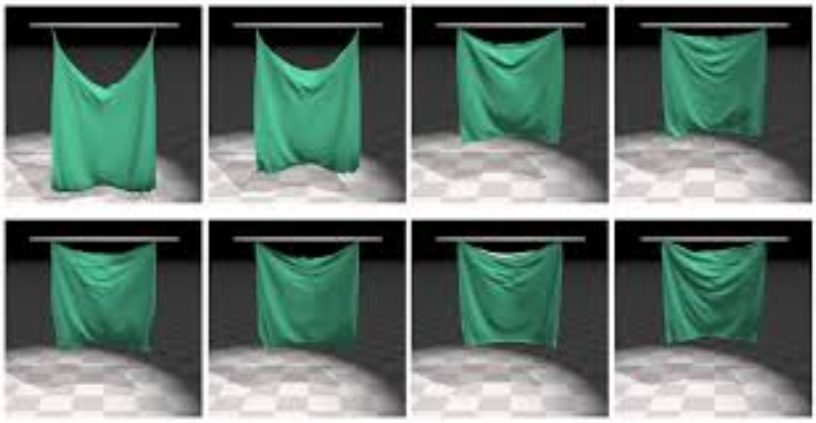
$$\Delta \mathbf{x} = k \Delta \mathbf{x}$$

te se s novim korekcijama računaju estimacije pozicija:

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \Delta\mathbf{x}$$

5. Proširena dinamika temeljena na poziciji

Jedna od glavnih mana PBD algoritma je njegova ovisnost o veličini vremenskog koraka i broju iteracija rješavača. U klasičnoj PBD formulaciji efektivna krutost ograničenja nije definirana isključivo parametrima materijala, već ovisi i o parametrima simulacije. U svakoj iteraciji ograničenja se ponovno projiciraju, odnosno pozicije čestica se dodatno korigiraju prema stanju u kojem je ograničenje zadovoljeno. Zbog toga veći broj iteracija znači da se ograničenja točnije zadovoljavaju, pa se objekt ponaša kruće. Primjerice, kod tkanine veći broj iteracija smanjuje rastezanje između čestica, što vizualno daje dojam krućeg materijala, što se može vidjeti na slici 5.1. Takva ovisnost otežava fizički konzistentno modeliranje materijala i podešavanje simulacije.



Slika 5.1. Viseća tkanina s 20, 40, 80 i 160 iteracija (s lijeva na desno). Gornji red: PBD, Donji red: XPBD. Krutost PBD-a ovisi o broju iteracija na nelinearan način, dok je ponašanje XPBD-a kvalitativno nepromijenjeno.

Proširena dinamika temeljena na poziciji (eng. *eXtended Position Based Dynamics* – XPBD) [4] proširuje klasični PBD uvođenjem podatnosti ograničenja (eng. *compliance*) i trajnih Lagrangeovih multiplikatora. Umjesto da se svaka iteracija promatra kao nezavisna korekcija pozicije, XPBD pamti prethodnu vrijednost Lagrangeovog multiplika-

tora λ tijekom vremenskog koraka. U svakoj sljedećoj iteraciji uzima se u obzir koliko je ograničenje već prethodno korigirano. Time se omogućuje formulacija ograničenja čija je krutost neovisna o vremenskom koraku i broju iteracija rješavača.

Podatnost α predstavlja recipročnu vrijednost krutosti:

$$\alpha = \frac{1}{k},$$

Za potpuno kruta ograničenja vrijedi:

$$\alpha = 0,$$

dok veće vrijednosti podatnosti omogućuju veće deformacije ograničenja.

Algoritam metode XPBD prikazan je sljedećim algoritmom:

Algorithm 2: Algoritam proširene dinamike temeljene na poziciji

foreach *particle i* **do**
- $\mathbf{x}_i \leftarrow \mathbf{x}_i^0$;
- $\mathbf{v}_i \leftarrow \mathbf{v}_i^0$;
- $w_i \leftarrow 1/m_i$;

for *every timestep* Δt **do**
- **foreach** *particle i* **do**
  - $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$;
  - $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$;
  - $\lambda_i \leftarrow 0$;
- **for** $k \leftarrow 1$ **to** *solverIterations* **do**
  - **foreach** *constraint* $C_j \in \{C_1, ..., C_{M+M_{coll}}\}$ **do**
    - **if** *constraint is broken* **then**
      - compute $\Delta\lambda$ using Eq.5.1
      - compute $\Delta\mathbf{x}$ using Eq.5.2
      - $\lambda_i \leftarrow \lambda_i + \Delta\lambda$
      - $\mathbf{p}_i \leftarrow \mathbf{p}_i + \Delta\mathbf{x}$
- **foreach** *particle i* **do**
  - $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$;
  - $\mathbf{x}_i \leftarrow \mathbf{p}_i$;

Na početku svakog vremenskog koraka računaju se nove brzine i predviđene pozicije čestica. Ovaj korak jednak je kao u PBD algoritmu i predstavlja gibanje čestica pod utjecajem vanjskih sila prije primjene ograničenja.

Glavna razlika između PBD i XPBD metode jest način računanja Lagrangeovog multiplikatora. Umjesto direktnog izračuna multiplikatora u svakoj iteraciji, XPBD uvodi akumulirani multiplikator koji se postupno ažurira tijekom rješavanja ograničenja.

Na početku svakog vremenskog koraka vrijednost multiplikatora postavlja se na:

$$\lambda_i = 0,$$

a u svakoj iteraciji rješavača računa se inkrement multiplikatora:

$$\Delta\lambda_i = \frac{-C(x_i) - \tilde{\alpha}\lambda_i}{\nabla C\, M^{-1}\, \nabla C^T + \tilde{\alpha}}. \tag{5.1}$$

Pri tome je:

$$\tilde{\alpha} = \frac{\alpha}{\Delta t^2},$$

Nakon izračuna inkrementa multiplikatora određuje se korekcija pozicija:

$$\Delta \mathbf{x}_i = M^{-1} \nabla C(x_i)^T \Delta\lambda \tag{5.2}$$

Vrijednosti multiplikatora i pozicija zatim se ažuriraju:

$$\lambda_{i+1} \leftarrow \lambda_i + \Delta\lambda_i,$$

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \Delta\mathbf{x}_i.$$

Postupak rješavanja ograničenja je sličan onom u PBD metodi, međutim, zbog uvođenja podatnosti i akumulacije multiplikatora, XPBD omogućuje stabilnije ponašanje vrlo krutih ograničenja čak i pri manjim brojevima iteracija.

Nakon završetka iterativnog rješavanja ažuriraju se brzine i pozicije čestica koristeći formule 4.1 i 4.2

Glavna prednost XPBD metode u odnosu na klasični PBD je mogućnost definiranja fizički smislenih parametara krutosti koji ostaju konzistentni neovisno o vremenskom koraku i broju iteracija rješavača. Time XPBD zadržava stabilnost i robusnost PBD metode, ali uz fizički konzistentnije ponašanje sustava.

6. Ograničenja

Ograničenja predstavljaju osnovni mehanizam kojim metode PBD i XPBD opisuju ponašanje simuliranih objekata. Različitim formulacijama ograničenja moguće je modelirati širok raspon fizičkih pojava i materijala, uključujući kruta tijela, meka tijela, tkanine, užad i fluide. Složenija ponašanja mogu se postići kombiniranjem više različitih vrsta ograničenja unutar istog sustava.

U ovom poglavlju bit će predstavljena matematička formulacija nekih od najčešćih ograničenja [6].

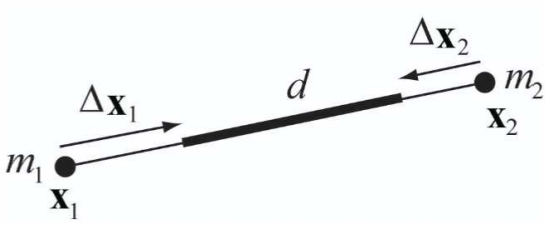
6.1. Ograničenje udaljenosti

Jedno od najčešće korištenih ograničenja je ograničenje udaljenosti (eng. *distance constraint*), čiji je cilj očuvati konstantnu udaljenost između dviju čestica.

Ograničenje udaljenosti definirano je izrazom:

$$C(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{x}_1 - \mathbf{x}_2|| - d, \tag{6.1}$$

gdje su $\mathbf{x}_1, \mathbf{x}_2$ pozicije čestica i d udaljenost između čestica u stanju mirovanja.



Slika 6.1. Vizualna reprezentacija ograničenja udaljenosti.

Ograničenje udaljenosti se može uzeti kao primjer za računanje ograničenja u meto-

dama PBD i XPBD. U nastavku je prikazan izračun korekcije $\Delta\mathbf{x}$ za ovo ograničenje [7]. Za izračun korekcija potrebno je odrediti gradijente funkcije ograničenja:

$$\frac{\partial C}{\partial \mathbf{x}_1} = \frac{\mathbf{x}_1 - \mathbf{x}_2}{||\mathbf{x}_1 - \mathbf{x}_2||},$$

te

$$\frac{\partial C}{\partial \mathbf{x}_2} = -\frac{\mathbf{x}_1 - \mathbf{x}_2}{||\mathbf{x}_1 - \mathbf{x}_2||}.$$

Za metodu PBD gradijent se uvrštava u formulu za računanje multiplikatora 4.5:

$$\lambda = -\frac{||\mathbf{x}_1 - \mathbf{x}_2|| - d}{w_1 + w_2} \tag{6.2}$$

te konačne korekcije pozicija čestica prema formuli 4.4 iznose:

$$\Delta\mathbf{x}_1 = -\frac{w_1}{w_1 + w_2}\left(||\mathbf{x}_1 - \mathbf{x}_2|| - d\right)\frac{\mathbf{x}_1 - \mathbf{x}_2}{||\mathbf{x}_1 - \mathbf{x}_2||} \tag{6.3}$$

$$\Delta\mathbf{x}_2 = \frac{w_2}{w_1 + w_2}\left(||\mathbf{x}_1 - \mathbf{x}_2|| - d\right)\frac{\mathbf{x}_1 - \mathbf{x}_2}{||\mathbf{x}_1 - \mathbf{x}_2||} \tag{6.4}$$

Za metodu XPBD $\Delta\lambda$ se računa koristeći formulu 5.1, a $\Delta\mathbf{x}$ formulom:

$$\Delta\mathbf{x}_1 = w_1\frac{\partial C}{\partial \mathbf{x}_1}\Delta\lambda \tag{6.5}$$

$$\Delta\mathbf{x}_2 = w_2\frac{\partial C}{\partial \mathbf{x}_2}\Delta\lambda \tag{6.6}$$

6.2. Ograničenje sudara s okolinom

Sudar čestica s okolinom predstavlja jedno od najvažnijih ograničenja u simulaciji deformabilnih tijela. U metodama PBD i XPBD sudari se najčešće modeliraju korištenjem ravnina kontakta koje sprječavaju penetraciju čestica kroz geometriju scene.

Ograničenje sudara s okolinom definirano je formulom:

$$C(\mathbf{x}) = \mathbf{n}^T\mathbf{x} - d_{rest} \tag{6.7}$$

gdje je: $\mathbf{x}$ pozicija čestice, $\mathbf{n}$ normala ravnine kontakta te d_{rest} minimalna udaljenost čestice od površine ravnine u stanju mirovanja. Ograničenje sudara je ograničenje nejednakosti pa je ono zadovoljeno kada vrijedi:

$$C(\mathbf{x}) \geq 0,$$

Ako čestica penetrira površinu objekta, vrijednost ograničenja postaje negativna te se pozicija čestice korigira u smjeru normale kako bi se uklonila penetracija.

6.3. Ograničenje sudara trokuta

Ograničenje sudara trokuta se najčešće koristi za sprječavanje međusobnog sudara površine tijela (eng. self-collision).

Neka je q vrh koji prodire kroz trokut definiran vrhovima $\mathbf{x}_1$, $\mathbf{x}_2$ i $\mathbf{x}_3$. Ograničenje sudara tada se definira kao udaljenost vrha od ravnine trokuta:

$$C(q, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (q - \mathbf{x}_1) \cdot \frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|} - h. \tag{6.8}$$

gdje je h minimalna udaljenost vrha q od površine trokuta. Izraz

$$\frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|}$$

predstavlja jediničnu normalu trokuta $\mathbf{n}$.

Ako vrh prodire kroz trokut, vrijednost ograničenja postaje negativna te je potrebno primijeniti korekciju pozicija kako bi se uklonila penetracija.

U slučaju kada vrh ulazi s druge strane trokuta, potrebno je koristiti normalu suprotnog smjera:

$$C(q, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (q - \mathbf{x}_1) \cdot \frac{(\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_2 - \mathbf{x}_1)}{\|(\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_2 - \mathbf{x}_1)\|} - h. \tag{6.9}$$

Na taj način ograničenje pravilno detektira penetraciju bez obzira s koje strane vrh

prilazi trokutu.

6.4. Ograničenje volumena

Kod simulacije zatvorenih deformabilnih tijela važno je očuvati volumen objekta tijekom deformacije. Bez volumenskih ograničenja meka tijela mogu se neprirodno urušavati ili gubiti volumen tijekom simulacije.

Jedan način za modelirati očuvanje volumena je simulirati konstantan tlak. Volumen zatvorene trokutaste mreže može se definirati pomoću svih trokuta mreže:

$$C(\mathbf{x}_1, ..., \mathbf{x}_N) = \left(\sum_{i=1}^{n_{triangles}} (\mathbf{x}_{t_1^i} \times \mathbf{x}_{t_2^i}) \cdot \mathbf{x}_{t_3^i} \right) - k_{pressure} V_0, \tag{6.10}$$

gdje je V_0 početni volumen objekta u stanju mirovanja, $k_{pressure}$ faktor unutarnjeg tlaka i t_1^i, t_2^i, t_3^i indeksi vrhova trokuta i.

Prvi dio izraza računa trenutni volumen zatvorene mreže, dok drugi dio predstavlja željeni volumen objekta. Ako trenutni volumen odstupa od željene vrijednosti, primjenjuju se korekcije pozicija vrhova kako bi se volumen ponovno očuvao.

Gradijent funkcije ograničenja za vrh i definiran je izrazom:

$$\nabla_{\mathbf{x}_i} C = \sum_{j:t_1^j=i} (\mathbf{x}_{t_2^j} \times \mathbf{x}_{t_3^j}) + \sum_{j:t_2^j=i} (\mathbf{x}_{t_3^j} \times \mathbf{x}_{t_1^j}) + \sum_{j:t_3^j=i} (\mathbf{x}_{t_1^j} \times \mathbf{x}_{t_2^j}). \tag{6.11}$$

Dobiveni gradijenti koriste se za računanje korekcija pozicija svih vrhova koji sudjeluju u ograničenju volumena.

6.5. Ograničenje fiksne točke

Ograničenje fiksne točke koristi se za čestice koje trebaju ostati fiksirane na jednoj poziciji. Takva ograničenja često se koriste kod simulacije tkanine, gdje su vrhovi tkanine pričvršćeni uz drugi objekt, npr. plašt pričvršćen za lika u videoigri.

Ograničenje fiksne točke može se definirati izrazom:

$$C(\mathbf{x}) = \mathbf{x} - \mathbf{x}_{fixed}, \tag{6.12}$$

gdje je $\mathbf{x}$ trenutna pozicija čestice, a $\mathbf{x}_{fixed}$ zadana fiksna pozicija.

Ograničenje je zadovoljeno kada se čestica $\mathbf{x}$ nalazi točno na definiranoj poziciji $\mathbf{x}_{fixed}$. U praksi se ovakva ograničenja često implementiraju postavljanjem inverzne mase čestice na nulu:

$$w = 0$$

čime njena masa postaje beskonačna, te čestica postaje nepomična i više ne sudjeluje u korekcijama pozicija tijekom simulacije.

6.6. Ograničenje savijanja

Kod simulacije tkanine nije dovoljno modelirati samo otpornost na rastezanje, već je potrebno uzeti u obzir i otpornost materijala na savijanje. Bez ograničenja savijanja tkanina se može neprirodno presavijati, čak i ako su udaljenosti između susjednih čestica očuvane.

Ograničenje savijanja definira se za svaki par susjednih trokuta koji dijele zajednički brid. Neka su zadana dva trokuta

$$(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2) \quad \text{i} \quad (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4),$$

gdje su $\mathbf{x}_1$ i $\mathbf{x}_2$ vrhovi zajedničkog brida, a $\mathbf{x}_3$ i $\mathbf{x}_4$ suprotni vrhovi trokuta.

Za takav par trokuta uvodi se bilateralno ograničenje savijanja:

$$C_{bend}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos\left(\frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|} \cdot \frac{(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1)}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1)\|}\right) - \phi_0 \tag{6.13}$$

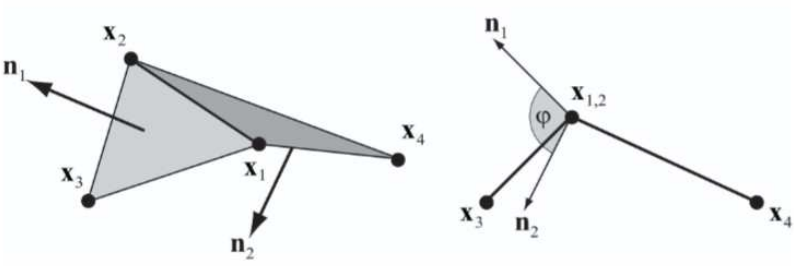
gdje je ϕ_0 početni dihedralni kut između dvaju trokuta u stanju mirovanja.

Ograničenje se može napisati i kao:

$$C_{bend}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \phi_0$$

gdje su $\mathbf{n}_1$ i $\mathbf{n}_2$ normale dvaju trokuta (Slika 6.2.).

Ograničenje je zadovoljeno kada trenutni kut između susjednih trokuta odgovara početnom kutu.



Slika 6.2. Ograničenje savijanja mjeri kut između normala trokuta $\mathbf{n}_1$ i $\mathbf{n}_2$.

6.7. Ograničenje izometrijskog savijanja

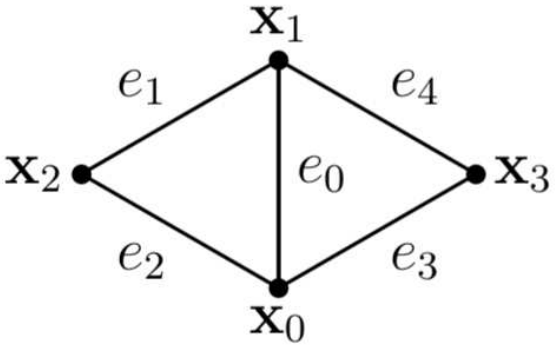
Ograničenje izometrijskog savijanja uvedeno je u radu [8] kao ograničenje namijenjeno simulaciji ne-rastezljivih površina. Ovakav pristup posebno je pogodan za simulaciju tkanina i odjevnih predmeta jer se mnogi tekstilni materijali mogu relativno lako savijati, ali se vrlo malo rastežu.

Za svaki unutarnji brid mreže definira se lokalna konfiguracija (engl. *stencil*) koja se sastoji od dva susjedna trokuta. Neka su

$$\mathbf{x}_s = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)^T$$

četiri vrha koja pripadaju toj konfiguraciji, pri čemu vrhovi $\mathbf{x}_0$ i $\mathbf{x}_1$ definiraju zajednički brid, dok su $\mathbf{x}_2$ i $\mathbf{x}_3$ suprotni vrhovi susjednih trokuta, kao na slici 6.3.

Iz geometrije konfiguracije definira se matrica



Slika 6.3. Lokalna konfiguracija ograničenja izometrijskog savijanja.

$$\mathbf{Q} = \frac{3}{A_0 + A_1}\mathbf{K}^T\mathbf{K},$$

gdje su A_0 i A_1 površine susjednih trokuta, a vektor

$$\mathbf{K} = (c_{01} + c_{04}, c_{02} + c_{03}, -c_{01} - c_{02}, -c_{03} - c_{04})$$

gdje je c_{jk} kotangenta $\cot \angle(\mathbf{e}_j, \mathbf{e}_k)$. Budući da matrica $\mathbf{Q}$ ovisi isključivo o geometriji u stanju mirovanja, može se unaprijed izračunati tijekom inicijalizacije simulacije.

Na temelju matrice $\mathbf{Q}$ definira se funkcija ograničenja:

$$C_{bend}(\mathbf{x}) = \frac{1}{2}\sum_{i,j}\mathbf{Q}_{ij}\mathbf{x}_i^T\mathbf{x}_j. \tag{6.14}$$

Ovo ograničenje predstavlja diskretnu aproksimaciju energije savijanja površine. Za razliku od jednostavnijih ograničenja savijanja koja se temelje na udaljenosti između vrhova, ograničenje izometrijskog savijanja uzima u obzir lokalnu geometriju površine te bolje opisuje promjene zakrivljenosti.

Glavna prednost ove formulacije jest njezina neovisnost o rastezanju površine. Budući da se temelji na pretpostavci očuvanja duljina bridova, ograničenje mjeri isključivo promjenu zakrivljenosti, a ne promjenu veličine trokuta. Zbog toga daje realističnije rezultate pri simulaciji tkanine i drugih tankih deformabilnih površina.

6.8. Ograničenje energije deformacije

Ograničenje energije deformacije (eng. *Strain Energy Constraint*) uvedeno je u radu [8] kao način integriranja modela mehanike kontinuuma u metode temeljene na poziciji. Za razliku od jednostavnijih ograničenja koja izravno kontroliraju udaljenosti, volumene ili kutove između čestica, ovo ograničenje opisuje deformaciju materijala pomoću energije pohranjene u deformiranom tijelu.

U mehanici kontinuuma deformacija tijela opisuje se funkcijom

$$\phi(\mathbf{X}) = \mathbf{X} + \mathbf{u} = \mathbf{x}, \tag{6.15}$$

gdje je $\mathbf{X}$ položaj točke u početnoj konfiguraciji, $\mathbf{x}$ njezin položaj nakon deformacije, a $\mathbf{u}$ polje pomaka.

Iz funkcije deformacije definira se gradijent deformacije

$$\mathbf{F} = \frac{\partial \phi(\mathbf{X})}{\partial \mathbf{X}}, \tag{6.16}$$

koji opisuje lokalnu promjenu oblika i volumena materijala. Na temelju gradijenta deformacije računa se Greenov tenzor deformacije

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right), \tag{6.17}$$

gdje je $\mathbf{I}$ jedinična matrica.

Veza između deformacije i naprezanja opisana je generaliziranim Hookeovim zakonom

$$\mathbf{S} = \mathbf{C}\boldsymbol{\varepsilon}, \tag{6.18}$$

gdje je $\mathbf{C}$ tenzor elastičnosti materijala. Za izotropne materijale elastična svojstva

najčešće se definiraju Youngovim modulom elastičnosti i Poissonovim omjerom.

Energija deformacije definira se pomoću gustoće energije deformacije

$$\Psi_s = \frac{1}{2}\boldsymbol{\varepsilon} : \mathbf{S} = \frac{1}{2}\,\mathrm{tr}\left(\boldsymbol{\varepsilon}^T\mathbf{S}\right) \tag{6.19}$$

gdje je $tr(\cdot)$ trag matrice.

Kod metoda PBD i XPBD energija deformacije koristi se za definiranje ograničenja oblika

$$C(\mathbf{x}) = E_s(\mathbf{x}), \tag{6.20}$$

pri čemu je E_s ukupna energija deformacije diskretiziranog elementa.

Za simulaciju tankih površina, poput tkanine, koristi se trokutasta mreža. Ograničenje pojedinog trokuta tada poprima oblik

$$C(\mathbf{x}) = E_s(\mathbf{x}) = A\Psi_s(\mathbf{F}), \tag{6.21}$$

gdje je A površina trokuta u početnoj konfiguraciji.

7. Implementacija

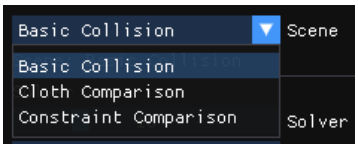
Kako bi se usporedile metode PBD i XPBD te ispitalo ponašanje različitih vrsta ograničenja, razvijen je vlastiti simulacijski sustav za simulaciju deformabilnih tijela. Sustav je implementiran u programskom jeziku C++, bez korištenja postojećih fizikalnih pogona, kako bi se omogućila potpuna kontrola nad implementacijom algoritama i ograničenja.

Za prikaz grafike korišten je OpenGL, dok se za upravljanje prozorom i unosom korisnika koristi biblioteka GLFW. Korisničko sučelje implementirano je pomoću biblioteke Dear ImGui, koja omogućuje jednostavno mijenjanje parametara tijekom izvođenja programa. Za rad s matricama i vektorima korištena je biblioteka Eigen, koja se koristi pri implementaciji matematičkih operacija potrebnih za simulaciju fizike.

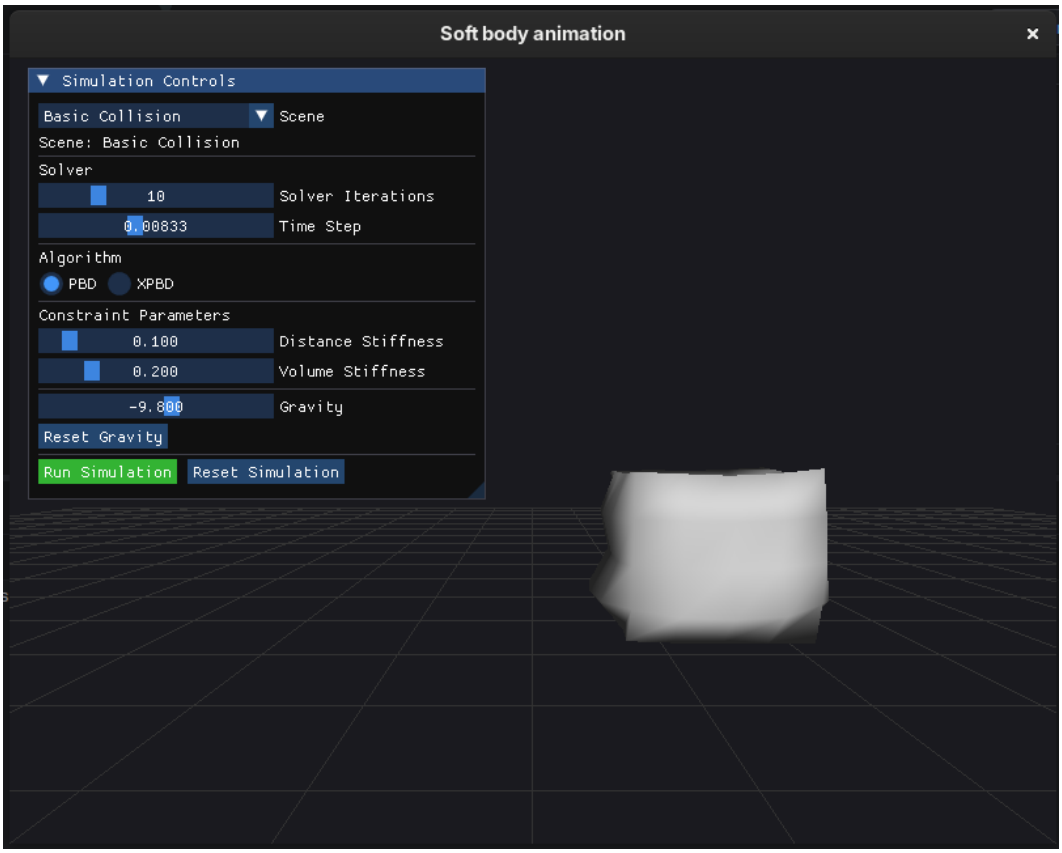
Tijekom razvoja simulacijskog sustava korišteno je nekoliko otvorenih repozitorija kao referenca za provjeru matematičkih formulacija, strukture programa i implementacijskih detalja. Posebno su korišteni projekti *Elasty* [9], *position-based-dynamics* [10] i *Feather* [11], koji su poslužili kao dodatni izvor informacija pri implementaciji metoda PBD i XPBD.

Aplikacija se sastoji od prikaza deformabilnog tijela i korisničkog sučelja koji omogućuje upravljanje simulacijom i promjenu njezinih parametara tijekom izvođenja programa. Primjer izgleda aplikacije je slika 7.2. Sučelje se sastoji od nekoliko skupina elemenata:

Scene je padajući izbornik koji omogućuje odabir između nekoliko različitih primjera koji se mogu izvršiti. Svaki primjer implementira početnu konfiguraciju objekta, korištena ograničenja te njihovo upravljanje korisničkim sučeljem. Pojedini primjeri i njihova namjena detaljnije su opisani u poglavlju 8.



Slika 7.1. Opcije u padajućem izborniku **Scene**



Slika 7.2. Izgled aplikacije

Parametri simulacije određuju način rada numeričkog rješavača:

- **Solver Iterations**: broj iteracija rješavača unutar jednog vremenskog koraka
- **Time Step**: veličina vremenskog koraka simulacije

Parametri deformabilnog tijela omogućuju podešavanje svojstava simuliranog objekta:

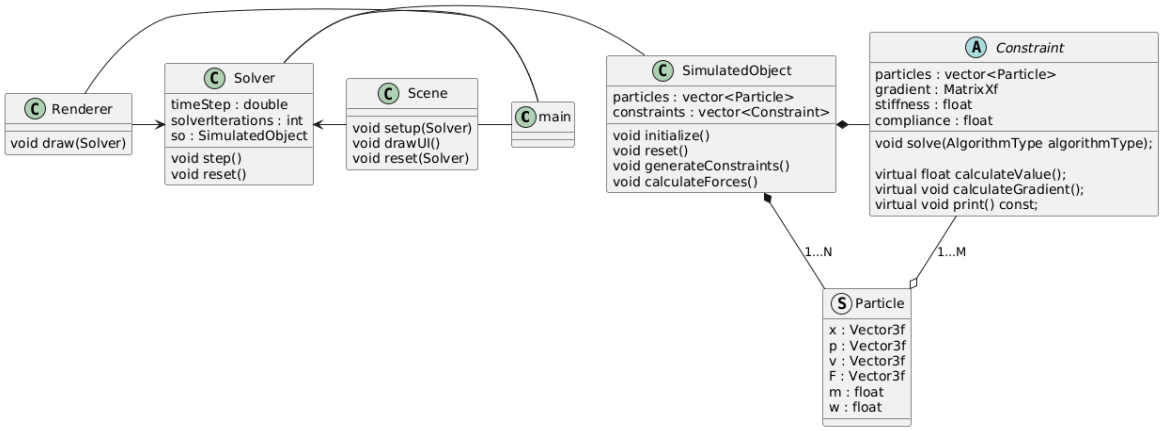
- **Algorithm**: odabir algoritma simulacije između metoda PBD i XPBD
- **Constraint Parameters**: ovisno o odabranoj metodi prikazuju se parametri pojedinih ograničenja. Kod metode PBD moguće je mijenjati krutost ograničenja, dok se kod metode XPBD podešava podatnost ograničenja
- **Gravity**: sila gravitacijskog ubrzanja koje djeluje na sve čestice sustava

- **Reset Gravity**: ponovno postavljanje sile na početnu vrijednost

Osim promjene parametara, korisničko sučelje omogućuje pauziranje i ponovno pokretanje simulacije gumbom **Run Simulation/Pause Simulation** te postavljanja objekta u početni položaj gumbom **Reset Simulation**.

7.1. Struktura programa

Struktura implementiranog sustava prikazana je UML dijagramom na slici 7.3.



Slika 7.3. UML dijagram strukture programa.

Klasa `Particle` predstavlja osnovnu jedinicu deformabilnog tijela. Svaka čestica sadrži trenutnu poziciju **x**, predviđenu poziciju **p**, brzinu **v**, rezultantnu silu **F**, masu m te inverznu masu w. Sve deformacije objekta modeliraju se promjenom položaja pojedinih čestica.

Ograničenja su implementirana pomoću apstraktne klase `Constraint`. Klasa sadrži skup čestica na koje djeluje, gradijent ograničenja te parametre krutosti i podatnosti koji se koriste u PBD i XPBD formulacijama. Funkcija `solve()` implementira postupak rješavanja ograničenja, dok izvedene klase implementiraju vlastite funkcije za izračun vrijednosti ograničenja i pripadajućih gradijenata. Iz klase `Constraint` izvedene su sve konkretne vrste ograničenja korištene u simulaciji.

Klasa `SimulatedObject` predstavlja deformabilni objekt sastavljen od skupa čestica i pripadajućih ograničenja. Jedan objekt sadrži proizvoljan broj čestica i ograničenja te pruža funkcije za inicijalizaciju objekta, ponovno postavljanje početnog stanja, generira-

nje ograničenja i izračun vanjskih sila. Geometrija objekta učitava se iz datoteka u OBJ formatu, nakon čega se generira odgovarajuća struktura čestica i ograničenja.

Numerički dio simulacije implementiran je u klasi `Solver`. Klasa sadrži simulirani objekt, veličinu vremenskog koraka te broj iteracija rješavača. Funkcija `step()` izvodi jedan korak simulacije, uključujući integraciju gibanja, predikciju pozicija, iterativno rješavanje ograničenja i ažuriranje brzina te je dana sljedećim kodom:

```
void step() {
    for (auto so : simObjects) {
        if (so->simulation.algorithmType == AlgorithmType::XPBD) {
            so->resetLambdaConstraints();
        }

        for (auto& p : so->particles) {
            p->v += timeStep * p->w * p->F;
            p->p = p->x + timeStep * p->v;
        }
    }

    for (auto so : simObjects) {
        for (int iter = 0; iter < solverIterations; ++iter) {
            so->projectConstraints();
        }

        for (auto& p : so->particles) {
            p->v = (p->p - p->x) / timeStep;
            p->x = p->p;
        }

        //velocity damping
        for (auto& p : so->particles) {
            p->v *= 0.98f;
        }
```

```
        }
}
```

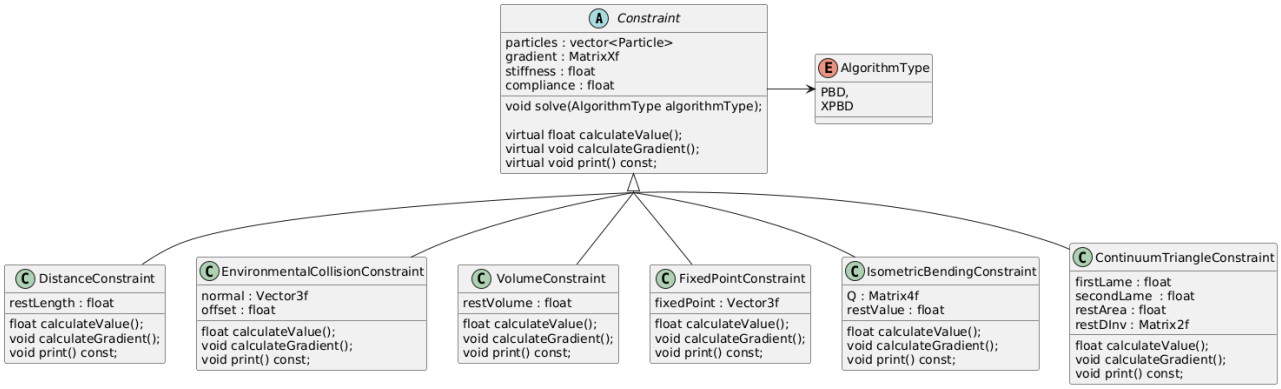
Za prikaz rezultata simulacije zadužena je klasa `Renderer`. Ova klasa koristi podatke dobivene iz rješavača te ih prikazuje pomoću OpenGL grafičkog sustava. Osim samih objekata, prikazuju se i pomoćni elementi poput koordinatne mreže i vizualizacije simulacijskog okruženja.

Klasa `Scene` predstavlja apstrakciju simulacijske scene. Svaka scena definira početnu konfiguraciju objekata, korisničko sučelje i postupak inicijalizacije simulacije. Takav pristup omogućuje jednostavno dodavanje novih eksperimenata bez izmjene ostatka sustava. Iz klase `Scene` izvedene su scene korištene za testiranje i usporedbu algoritama opisane u poglavlju 8.

Glavni program (`main`) odgovoran je za inicijalizaciju sustava, stvaranje odgovarajuće scene, pokretanje simulacijskog rješavača te upravljanje petljom izvođenja aplikacije. Tijekom izvođenja glavna petlja koordinira komunikaciju između scene, rješavača i sustava za iscrtavanje.

7.2. Klasa Constraint

Apstraktna klasa `Constraint` definira izgled i ponašanje ograničenja. Na slici 7.4. je nacrtan dijagram klasa ograničenja koje su implementirane u ovom radu.



Slika 7.4. Dijagram klasa ograničenja

Implementirana ograničenja su ograničenje udaljenosti (poglavlje 6.1.), ograničenje sudara s okolinom (poglavlje 6.2.), ograničenje volumena (poglavlje 6.4.), ograničenje

fiksne točke (poglavlje 6.5.), ograničenje izometrijskog savijanja (poglavlje 6.7.) te ograničenje energije deformacije (poglavlje 6.8.).

Klasa `Constraint` sama definira funkciju `solve`, a preostale funkcije se definiraju u naslijeđenim klasama. Funkcija `solve` odgovara računanju ograničenja opisanom u poglavlju 4.1., te je dana sljedećim kodom:

```
def solve(AlgorithmType algorithmType) {
    // constraint is satisfied
    float C = calculateValue();
    if (isSatisfied(C)) return;

    calculateGradient();

    if (algorithmType == AlgorithmType::PBD) {
        //compute lambda
        float denominator = 0.0f;
        for (unsigned int i = 0; i < particles.size(); i++) {
            denominator += particles[i]->w *
            gradient.col(i).dot(gradient.col(i));
        }
        if(denominator > 1e-6) {
            lambda = - stiffness * (C / denominator);
        }
        // update particle positions
        for (unsigned int i = 0; i < particles.size(); i++) {
            particles[i]->p += (lambda * particles[i]->w) *
            gradient.col(i);
        }
    } else if (algorithmType == AlgorithmType::XPBD) {
        // compute delta lambda
        float alphaTilde = compliance / (dt * dt);
        float numerator = -C - alphaTilde * lambda;
        float denominator = alphaTilde;
```

```
            for (unsigned int i = 0; i < particles.size(); i++) {
                denominator += particles[i]->w *
                gradient.col(i).dot(gradient.col(i));
            }

            float deltaLambda = 0.0;
            if (denominator < 1e-6) {
                deltaLambda = 0.0;
            } else {
                deltaLambda = numerator / denominator;
            }
            lambda += deltaLambda;

            // update particle positions
            for (unsigned int i = 0; i < particles.size(); i++) {
                particles[i]->p += (deltaLambda * particles[i]->w)
                * gradient.col(i);
            }
        }
    }
```

Funkcija `isSatisfied` provjerava ako je ograničenje zadovoljeno ovisno o tome je li ograničenje jednakosti ili nejednakosti. Naslijeđene klase same definiraju funkcije `calculateValue`, koja definira vrijednost funkcije ograničenja $C(\mathbf{x})$, te `calculateGradient`, koja računa gradijent $\nabla_{\mathbf{x}} C(\mathbf{x})$.

U nastavku je navedena implementacija funkcija `calculateValue` te `calculateGradient` za ograničenje udaljenosti (DistanceConstraint), koje je izvedeno u poglavlju 6.1. :

```
float calculateValue() {
    auto& p0 = particles[0];
    auto& p1 = particles[1];
    return (p0->p - p1->p).norm() - restLength;
```

```
}

void calculateGradient() {
    auto g1 = (particles[0]->p - particles[1]->p).normalized();
    auto g2 = -g1;

    gradient.col(0) = g1;
    gradient.col(1) = g2;
}
```

8. Rezultati

Cilj ovog poglavlja je analizirati ponašanje implementiranih metoda PBD i XPBD na nekoliko primjera simulacija te usporediti njihovu stabilnost, vizualnu kvalitetu i računalnu složenost.

8.1. Eksperimentalno okruženje

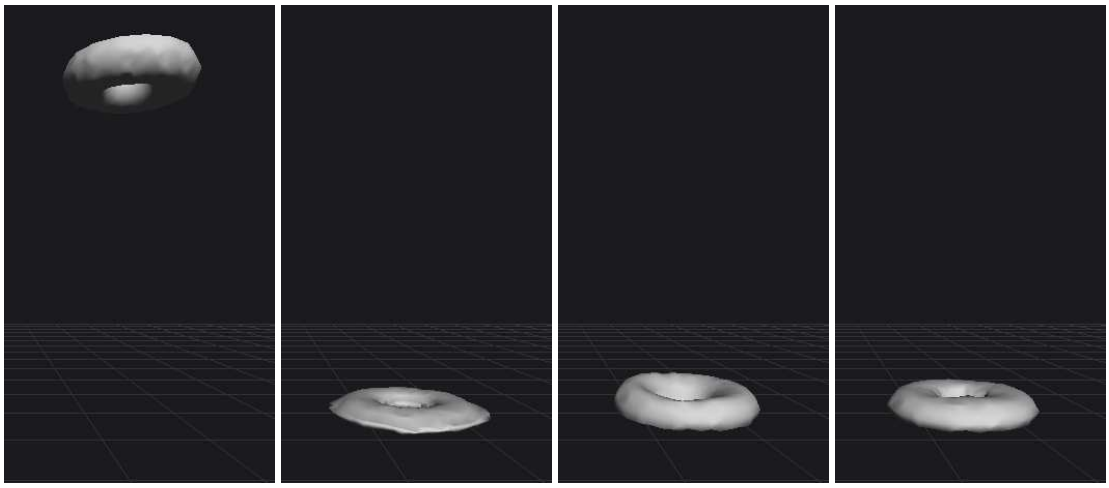
Svi eksperimenti provedeni su korištenjem simulacijskog sustava opisanog u prethodnom poglavlju. Za prikaz deformabilnih objekata korištene su trokutaste mreže, dok su simulacije izvođene korištenjem implementacija algoritama PBD i XPBD.

Mjerenja su provedena na prijenosnom računalu HP Laptop 15-da2xxx s procesorom Intel® Core™ i3-10110U, integriranom grafičkom karticom Intel® UHD Graphics (CML GT2), 8.0 GiB radne memorije i operacijskim sustavom Fedora Linux 43.

8.2. Primjer deformabilnog tijela sa sudarima

Prvi primjer koristi deformabilno tijelo koje pod utjecajem gravitacije pada na ravninu te ostvaruje sudare s okolinom. Objekt je modeliran pomoću ograničenja udaljenosti i volumena, dok se interakcija s okolinom ostvaruje korištenjem ograničenja sudara. Primjer je realiziran odabirom scene `Basic Collision`, a izgled aplikacije i korisničkog sučelja se može vidjeti na slici 7.2.

Cilj ovog primjera je provjera ispravnosti implementacije sudara i analiza stabilnosti simulacije tijekom kontakta s okolinom. Posebna pozornost posvećena je očuvanju volumena objekta i sprječavanju penetracije kroz podlogu.



Slika 8.1. Sudar objekta torus kroz vrijeme.

8.3. Primjer simulacije tkanine

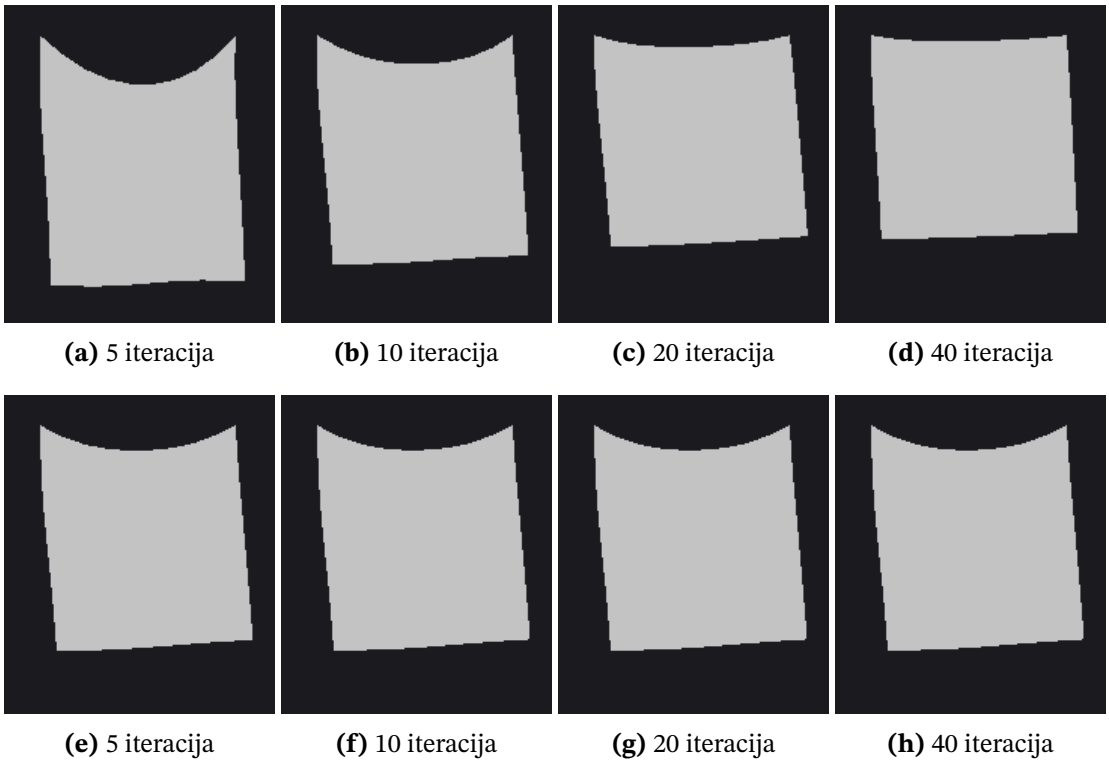
Drugi primjer temelji se na simulaciji tkanine modelirane kao pravilna mreža čestica. Čestice su povezane ograničenjima udaljenosti i savijanja, dok su rubni vrhovi učvršćeni pomoću ograničenja fiksne točke. Na slici 8.2. se može vidjeti primjer izvođenja aplikacije, koji se odabire izborom `Cloth Comparison` u izborniku **Scene**. Od **Constraint Parameters** se mogu podesiti krutost ograničenja udaljenosti i ograničenja izometrijskog savijanja, odnosno podatnost za metodu XPBD.

Slika 8.2. Primjer aplikacije sa opcijom `Cloth Comparison` u izborniku **Scene**.

Jedan od glavnih nedostataka metode PBD jest činjenica da efektivna krutost objekta ovisi o broju iteracija rješavača. Povećanjem broja iteracija ograničenja se preciznije zadovoljavaju, što rezultira krućim ponašanjem simuliranog materijala. Takvo ponašanje

otežava definiranje fizički konzistentnih parametara materijala jer promjena parametara simulacije mijenja i vizualni rezultat.

Kako bi se ispitao navedeni efekt, provedena je simulacija tkanine za različit broj iteracija rješavača. Parametri deformabilnog tijela kao krutost ili podatnost ograničenja nisu mijenjani, jedini mijenjani parametar je broj iteracija. Izvođenje je zaustavljeno nakon 40 koraka algoritma. Rezultati su prikazani na slici 8.3.



(a) 5 iteracija **(b)** 10 iteracija **(c)** 20 iteracija **(d)** 40 iteracija

(e) 5 iteracija **(f)** 10 iteracija **(g)** 20 iteracija **(h)** 40 iteracija

Slika 8.3. Usporedba simulacije tkanine za različit broj iteracija. **Prvi red**: metoda PBD, **Drugi red**: metoda XPBD.

Može se primijetiti da se kod metode PBD povećanjem broja iteracija smanjuje deformacija tkanine. Pri malom broju iteracija tkanina pokazuje veće rastezanje i mekše ponašanje, dok se povećanjem broja iteracija ponaša sve kruće. Drugim riječima, promjena broja iteracija mijenja efektivna svojstva simuliranog materijala.

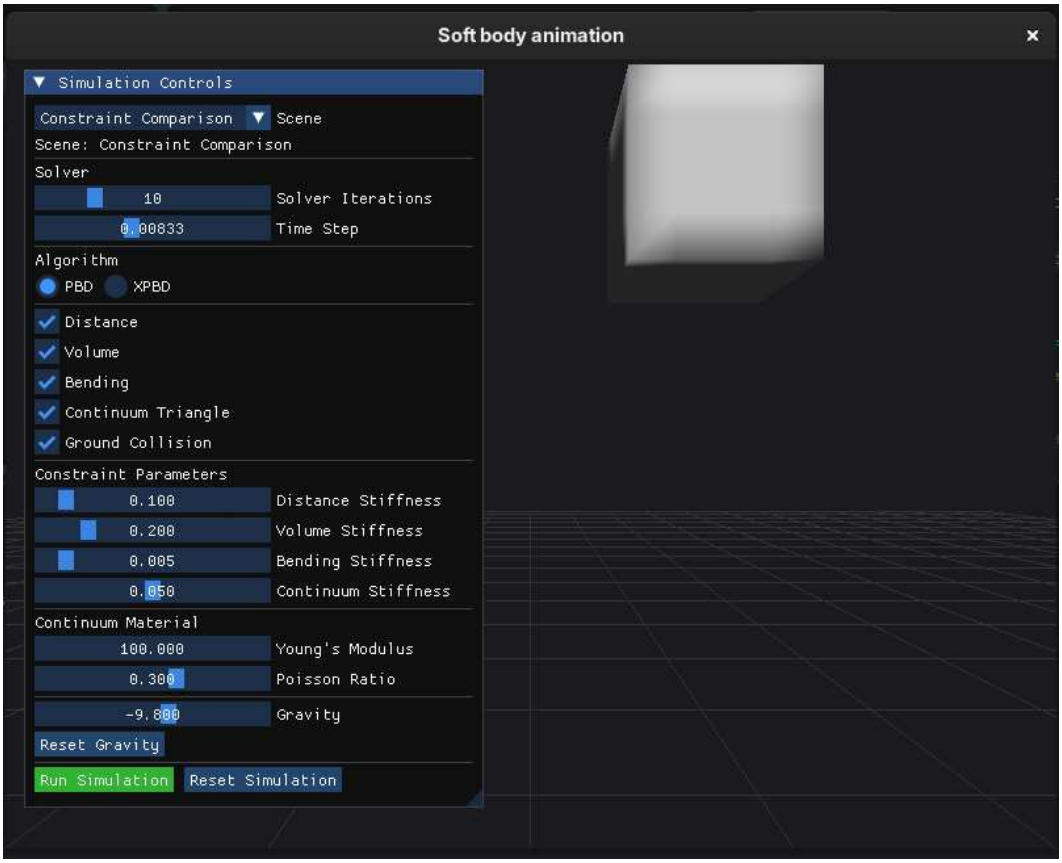
Kod metode XPBD takav efekt znatno je manje izražen. Uvođenjem podatnosti i akumulacije Lagrangeovih multiplikatora postiže se konzistentnije ponašanje ograničenja, neovisno o broju iteracija. Zbog toga se oblik tkanine mijenja znatno manje između pojedinih simulacija.

8.4. Primjer usporedbe ograničenja

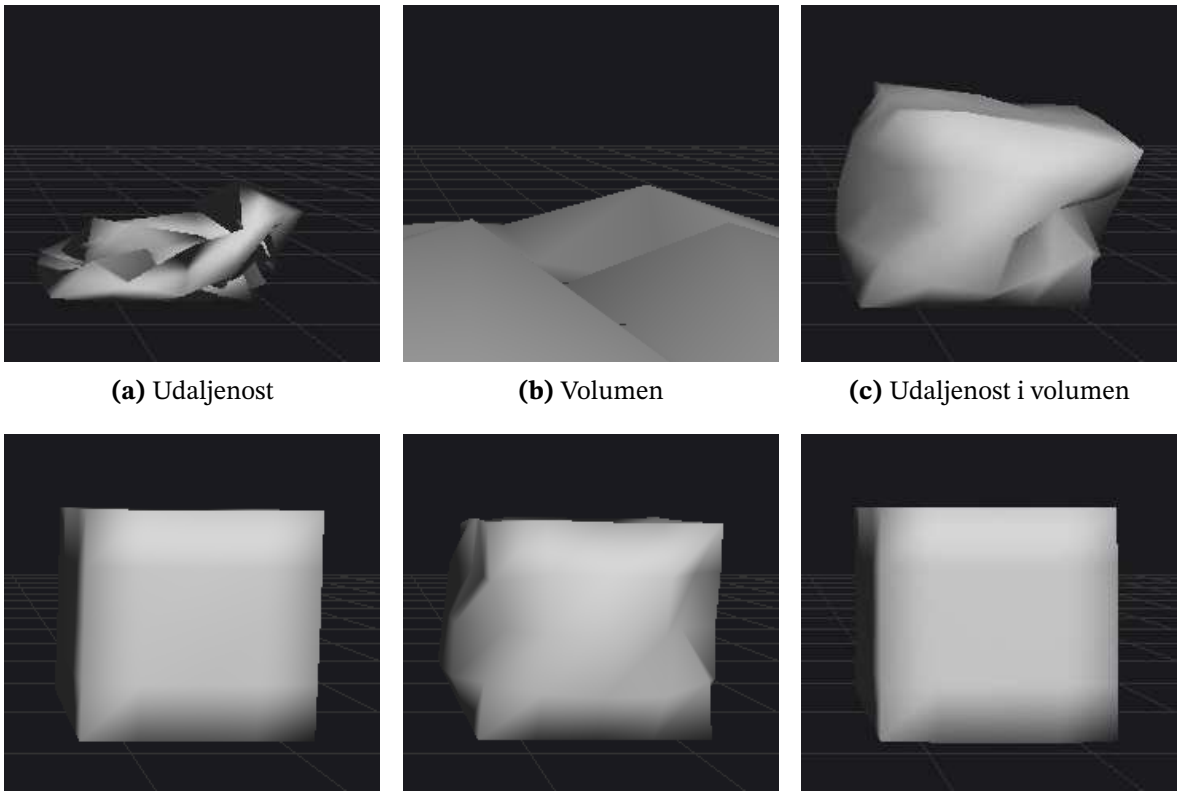
Treći primjer razvijen je s ciljem usporedbe različitih vrsta ograničenja korištenih u simulaciji deformabilnih tijela. Za isti objekt moguće je neovisno uključivati ili isključivati ograničenja udaljenosti, volumena, savijanja, energije deformacije i sudara.

Ovaj primjer omogućuje analizu utjecaja pojedinih ograničenja na ponašanje objekta te procjenu njihove prikladnosti za simulaciju različitih vrsta deformabilnih materijala.

Primjer aplikacije se nalazi na slici 8.4., odabirom `Constraint Comparison` u izborniku **Scene**.



Slika 8.4. Primjer aplikacije sa opcijom `Constraint Comparison` u izborniku **Scene**.



(a) Udaljenost **(b)** Volumen **(c)** Udaljenost i volumen

(d) Udaljenost, volumen i savijanje **(e)** Udaljenost, volumen i energija deformacije **(f)** Udaljenost, volumen, savijanje i energija deformacije

Slika 8.5. Usporedba različitih kombinacija ograničenja korištenih u trećem primjeru simulacije.

Slika 8.5. prikazuje utjecaj različitih kombinacija ograničenja na ponašanje deformabilnog objekta tijekom sudara s tlom. Svi primjeri su simulirani metodom PBD, te je krutost svakog ograničenja postavljena na nisku vrijednost kako bi se pokazao utjecaj pojedinog ograničenja. Simulacija je zaustavljena nakon 200 koraka algoritma, kada je objekt dosegao stanje približne ravnoteže i poprimio svoj konačni oblik.

Iz primjera **(a)** i **(b)** vidljivo je da korištenje samo jednog ograničenja nije dovoljno za očuvanje svih željenih svojstava materijala. Ograničenje udaljenosti sprječava preveliko rastezanje mreže, ali ne može očuvati volumen objekta, dok volumensko ograničenje samo po sebi ne sprječava značajne lokalne deformacije površine. Dodavanjem više ograničenja postiže se stabilnije i vizualno uvjerljivije ponašanje simuliranog tijela.

Usporedbom primjera **(c)**, **(d)** i **(e)** može se primijetiti da dodatna ograničenja postupno povećavaju otpornost objekta na deformacije te omogućuju bolje očuvanje izvornog oblika. Najmanje deformacije prisutne su u primjeru **(f)**, gdje su istovremeno korištena ograničenja udaljenosti, volumena, savijanja i energije deformacije. U tom slučaju

objekt nakon sudara zadržava oblik najsličniji početnoj konfiguraciji.

8.5. Performanse

Osim kvalitete simulacije, važan aspekt usporedbe predstavlja i računalna složenost algoritama. Za primjer 8.3. izmjereno je prosječno vrijeme izvođenja jednog simulacijskog koraka za različit broj iteracija rješavača.

Rezultati mjerenja prikazani su u tablici 8.1.

Tablica 8.1. Prosječno vrijeme izvođenja jednog simulacijskog koraka za različit broj iteracija.

	5 iteracija	10 iteracija	20 iteracija	40 iteracija
PBD	42.60 ms	75.04 ms	150.63 ms	306.95 ms
XPBD	43.05 ms	78.79 ms	163.30 ms	331.68 ms

Može se primijetiti da se vrijeme izvođenja približno linearno povećava s brojem iteracija rješavača. To je očekivano budući da se svaka iteracija sastoji od prolaska kroz sva ograničenja sustava.

Rezultati također pokazuju da XPBD ostvaruje nešto veće vrijeme izvođenja od metode PBD. Razlog tome je dodatni izračun korekcije Lagrangeovog multiplikatora i pohrana akumulirane vrijednosti multiplikatora za svako ograničenje. Međutim, razlika u vremenu izvođenja relativno je mala u usporedbi s dobivenim poboljšanjem konzistentnosti simulacije.

9. Zaključak

U ovom radu su proučene metode dinamike temeljene na poziciji (PBD) i proširene dinamike temeljene na poziciji (XPBD) za simulaciju deformabilnih tijela. Prikazana je teorijska podloga za obje metode, s posebnim naglaskom na formulaciju i rješavanje ograničenja, te su opisane najčešće korištene vrste ograničenja.

U sklopu rada razvijen je vlastiti simulacijski sustav u programskom jeziku C++ koji omogućuje simulaciju deformabilnih tijela korištenjem metoda PBD i XPBD te pruža mogućnost promjene parametara simulacije tijekom izvođenja programa. Implementirana su različita ograničenja i razvijeni primjeri simulacija koji omogućuju usporedbu ponašanja algoritama u različitim uvjetima.

Ostvareni simulacijski sustav uspješno demonstrira mogućnosti metoda PBD i XPBD za simulaciju deformabilnih tijela te potvrđuje prednosti XPBD pristupa pri modeliranju materijala čija svojstva trebaju ostati konzistentna neovisno o parametrima simulacije.

Moguća proširenja rada uključuju implementaciju dodatnih modela materijala, poput Neo-Hookeovog modela, implementaciju učinkovitijih algoritama za detekciju sudara između različitih objekata ili sa samim sobom te optimizaciju izvođenja korištenjem paralelnog programiranja ili grafičkih procesora.

Literatura

[1] S. F. Gibson i B. Mirtich, “A survey of deformable modeling in computer graphics”, Technical report, Mitsubishi Electric Research Laboratories, teh. izv., 1997.

[2] A. Nealen, M. Müller, R. Keiser, E. Boxerman, i M. Carlson, “Physically based deformable models in computer graphics”, u *Computer graphics forum*, sv. 25, br. 4. Wiley Online Library, 2006., str. 809–836, https://matthias-research.github.io/pages/publications/egstar2005.pdf.

[3] M. Müller, B. Heidelberger, M. Hennix, i J. Ratcliff, “Position based dynamics”, *Journal of Visual Communication and Image Representation*, sv. 18, br. 2, str. 109–118, 2007., https://matthias-research.github.io/pages/publications/posBasedDyn.pdf.

[4] M. Macklin, M. Müller, i N. Chentanez, “Xpbd: position-based simulation of compliant constrained dynamics”, u *Proceedings of the 9th International Conference on Motion in Games*, 2016., str. 49–54, https://matthias-research.github.io/pages/publications/XPBD.pdf.

[5] C. Cincotti. (2022., kolovoz) The pbd simulation loop. [Mrežno]. Adresa: https://carmencincotti.com/2022-08-01/the-pbd-simulation-loop/

[6] J. Bender, M. Müller, i M. Macklin, “A survey on position based dynamics, 2017”, u *Proceedings of the European Association for Computer Graphics: Tutorials*, 2017., str. 1–31, https://matthias-research.github.io/pages/publications/PBDTutorial2017-CourseNotes.pdf.

[7] Interactive Computer Graphics Group. (2025) Physics-based simulation. [Mrežno]. Adresa: https://interactivecomputergraphics.github.io/physics-simulation/

[8] J. Bender, D. Koschier, P. Charrier, i D. Weber, “Position-based simulation of continuous materials”, *Computers & Graphics*, sv. 44, str. 1–10, 2014., https://animation.rwth-aachen.de/media/papers/2014-CAG-PBER.pdf.

[9] Y. Koyama, “Elasty: A research-oriented elastic body simulation library”, https://github.com/yuki-koyama/elasty, 2025., gitHub repozitorij, pristupljeno 11.6.2026.

[10] Q.-M. Ton-That, “position-based-dynamics: Extended position-based dynamics implementation with soft body virtual cutting”, https://github.com/Q-Minh/position-based-dynamics, 2022., gitHub repozitorij, pristupljeno 11.6.2026.

[11] B. Paléologue, “Feather: Lightweight 3d c++ opengl engine with physics powered by xpbd”, https://github.com/BarthPaleologue/feather, 2024., gitHub repozitorij, pristupljeno 11.6.2026.

Izjava o korištenju umjetne inteligencije

Osvrt na korištenje umjetne inteligencije: Tijekom izrade ovog diplomskog rada korišten je alat ChatGPT, temeljen na velikom jezičnom modelu GPT-5.5. Alat je korišten za ispravak gramatičkih i pravopisnih pogrešaka, poboljšanje stila teksta, oblikovanje LaTeX zapisa te za prijevod pojedinih dijelova teksta između hrvatskog i engleskog jezika. Također je korišten za istraživanje područja rada te pri razumijevanju i objašnjavanju metoda korištenih u radu.

Izjava o korištenju umjetne inteligencije: Potvrđujem da sam tijekom izrade ovog diplomskog rada koristio umjetnu inteligenciju u skladu s Politikom primjerenog korištenja umjetne inteligencije na Fakultetu elektrotehnike i računarstva, te da umjetna inteligencija nije zamijenila moje kritičko razmišljanje niti moj doprinos.

Sažetak

Vizualizacija simulacije deformabilnih tijela

Hana Ujčić

Računalne simulacije danas se koriste u brojnim područjima, od videoigara i animacije do virtualne stvarnosti i inženjerskih primjena. Poseban izazov predstavlja simulacija objekata koji se mogu deformirati, poput tkanine, gume ili drugih mekih materijala, pri čemu je potrebno postići uvjerljivo ponašanje uz dovoljno brzo izvođenje simulacije.

U ovom radu proučene su metode za simulaciju deformabilnih tijela koje omogućuju izvođenje simulacija u stvarnom vremenu. Razvijen je simulacijski sustav koji omogućuje modeliranje različitih vrsta deformacija i interakcija između objekata te usporedbu dvaju pristupa simulaciji. Analiziran je utjecaj različitih parametara i matematičkih modela na ponašanje simuliranih objekata. Dobiveni rezultati pokazuju da suvremeni pristupi omogućuju realističnije i konzistentnije simulacije uz prihvatljiv računalni trošak.

Ključne riječi: računalna grafika; simulacija fizike; deformabilna tijela; simulacija u stvarnom vremenu

Abstract

Visualization of deformable objects simulation

Hana Ujčić

Computer simulations are widely used in areas such as video games, animation, virtual reality, and engineering applications. A particular challenge is the simulation of deformable objects, such as cloth, rubber, and other soft materials, where realistic behavior must be achieved while maintaining interactive performance.

This thesis investigates methods for real-time simulation of deformable bodies. A simulation system was developed to model different types of deformations and interactions between objects and to compare two simulation approaches. The influence of various parameters and mathematical models on the behavior of simulated objects was analyzed. The results show that modern simulation methods can provide more realistic and consistent behavior while maintaining an acceptable computational cost.

Keywords: computer graphics; physics simulation; deformable bodies; real-time simulation

Privitak A: The Code

Kod razvijenog simulacijskog sustava je moguće naći na adresi `https://github.com/UjcicHana/projekt_soft_bodies`.