

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1850

**Deformacije objekta metodom  
konačnih elemenata**

Josip Sito

Zagreb, lipanj 2019.

Zagreb, 7. ožujka 2019.

## DIPLOMSKI ZADATAK br. 1850

Pristupnik: **Josip Sito (0036487715)**  
Studij: Računarstvo  
Profil: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Deformacije objekta metodom konačnih elemenata**

### Opis zadatka:

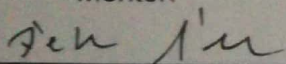
Proučiti metode koje se koriste za deformiranje objekata a posebice proučiti metodu koja se temelji na konačnim elementima. Proučiti metode koje se koriste za dijeljenje objekta na konačne elemente. Razraditi algoritam za dijeljenje objekta na konačne elemente i razraditi algoritme za deformiranje objekata temeljene na konačnim elementima. Prikazati rezultate. Diskutirati utjecaj različitih parametara. Razmotriti mogućnosti implementacije u stvarnom vremenu. Načiniti ocjenu rezultata i implementiranih algoritama.

Izraditi odgovarajući programski proizvod. Koristiti programsko razvojno okruženje Unity 3D te programski jezik C#. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

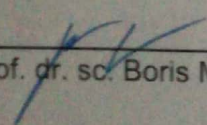
Zadatak uručen pristupniku: 15. ožujka 2019.

Rok za predaju rada: 28. lipnja 2019.

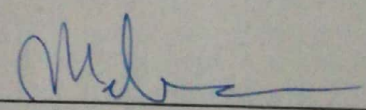
Mentor:

  
\_\_\_\_\_  
Prof. dr. sc. Željka Mihajlović

Djelovođa:

  
\_\_\_\_\_  
Izv. prof. dr. sc. Boris Milašinović

Predsjednik odbora za  
diplomski rad profila:

  
\_\_\_\_\_  
Izv. prof. dr. sc. Igor Mekterović



# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Model deformabilnog objekta</b>	<b>3</b>
<b>3. Metode izračuna deformacija</b>	<b>6</b>
3.1. Sustav masa i opruga . . . . .	9
3.2. Metoda konačnih elemenata . . . . .	10
3.2.1. Mehanika kontinuuma . . . . .	11
3.3. Uvođenje vremenske dimenzije . . . . .	16
3.3.1. Eksplicitni Euler . . . . .	16
3.3.2. Runge-Kutta integracija . . . . .	17
3.3.3. Verlet integracija . . . . .	18
3.3.4. Implicitna integracija . . . . .	19
<b>4. Diskretizacija</b>	<b>22</b>
4.1. Konvergencija mreže . . . . .	23
4.2. Delaunayeva tetraedarizacija . . . . .	23
4.3. Varijacijska triangulacija . . . . .	24
<b>5. Implementacija</b>	<b>30</b>
5.1. Triangulacija . . . . .	33
5.2. Metoda konačnih elemenata . . . . .	34
5.3. Detekcija kolizije . . . . .	36
5.4. Integracija vremena . . . . .	37
<b>6. Zaključak</b>	<b>39</b>
<b>Literatura</b>	<b>41</b>

# 1. Uvod

Tehnike za fizičku simulaciju u računalnoj grafici u zadnjih dvadeset godina su značajno unapredovale. Simulacije kompleksnih fenomena, fluida i čvrstih tijela sada mogu rezultirati animacijom koju je teško razlikovati od stvarnosti (Parker i O'Brien, 2009). Takav sustav je još uvijek značajno i aktivno područje istraživanja i spaja Newtonovu dinamiku, mehaniku kontinuuma, numeričku matematiku, diferencijalne izračune i računalnu grafiku u moćan alat za rješavanje problema koji je u konstantnom razvoju (Nealen et al., 2006).

U grafičkim aplikacijama najčešće je prioritet računalna efikasnost generiranja željenih ponašanja u odnosu na preciznost egzaktnih rezultata te sve dok simulacija izgleda realno, prihvatljivo je koristiti metode koje pojednostavljaju fizički sustav. Jedan takav pojednostavljeni sustav je sustav masa i opruga. Mana sustava masa i opruga je u tome što je kod njega teško prikazati bitna svojstva materijala u simulaciji kao što je odnos između vlačnog naprezanja i linijske vlačne deformacije, a metode koje prikazuju svojstva mehanike kontinuuma su značajno računalno skuplje i često se koriste u *offline* animacijama deformabilnog objekta. Deformabilni objekti korišteni su u širokom spektru grafičkih aplikacija od animacija tkanine, izraza lica, modela mekih tijela (engl. *soft bodies*) pa sve do kirurških treninga i virtualnog oblikovanja modela (engl. *sculpting*) (Müller et al., 2001).

Ubrzan rast računalne snage CPU-a i GPU-a u prethodnim godinama omogućio je simulacije fizičkih učinaka u stvarnom vremenu te je takva simulacija jedna od glavnih značajki video igara. Omogućavanje korisničke interakcije s fizičkim simulacijama stvara dodatni skup problema i izazova (Müller et al., 2008).

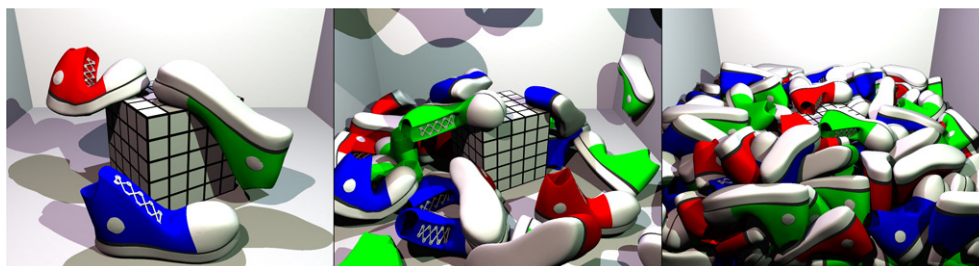
Jedna od metoda koja prikazuje bitna svojstva materijala te omogućuje fizičku simulaciju u stvarnom vremenu je metoda konačnih elemenata. Ona je u posljednje vrijeme postala neophodan alat u računalnoj grafici. Uglavnom se koristi za simulaciju deformabilnih objekata i fluida, a zahvaljujući čvrstom matematičkom temelju prona-

laze se novi načini primjene i u ostalim područjima gdje se problemi mogu predstaviti u obliku parcijalnih diferencijalnih jednažbi (Kaufmann, 2012).

U ovom radu izloženi su principi modela deformabilnog objekta, opisani su načini prema kojima se ostvaruje deformacija objekta pri čemu se detaljnije opisuje metoda konačnih elemenata i njezina primjena u računalnoj grafici. Prikazana je implementacija simulacije pomoću metode konačnih elemenata izrađena u Unity grafičkom alatu.

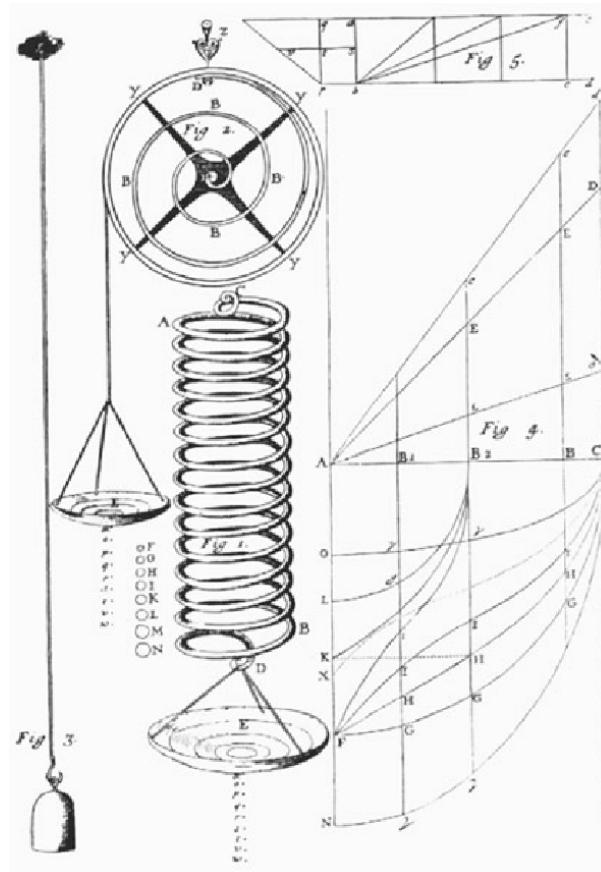
## 2. Model deformabilnog objekta

Za potrebe računalne simulacije koja prikazuje različite fenomene fizike, tijela su podijeljena u tri grupe: čvrsta tijela (engl. *rigid bodies*), meka tijela (engl. *soft bodies*) i tkanina (engl. *cloth*). Sa stajališta fizike, u stvarnom svijetu ne postoji klasifikacija navedenih tijela jer je svako tijelo u određenoj mjeri deformabilno. Ali iz algoritamske i simulacijske perspektive klasifikacija takvih tijela ima smisla. Npr. pretpostavka da objekti koji su napravljeni od kamena sadrže beskonačnu čvrstoću ili simuliranje tkanine kao 2D objekta umjesto 3D objekta pojednostavljuje rukovanje tih objekata, ubrzava računanje i smanjuje potrošnju memorije (Müller et al., 2008). U nastavku primarni fokus će biti na čvrstim tijelima (Slika 2.1).



Slika 2.1: Primjer simulacije i interakcije deformabilnih objekata

Elastičnost je svojstvo materijala da se uslijed deformacije uzrokovane djelovanjem vanjske sile vrati u svoj prvobitni oblik po prestanku djelovanja sile. Za mnoge elastične materijale vrijedi linearna funkcija omjera primijenjenih sila i naprezanja materijala. Naprezanje materijala je svojstvo da uslijed primjene sile na materijal on stvara protusilu. Klasični model linearne elastičnosti je savršena opruga. Savršenu oprugu, kao pojam je prvi put predstavio Robert Hooke 1675. godine (Slika 2.2). Linearan odnos primijenjene sile i naprezanja materijala opisan je Hookeovim zakonom. Sila potrebna da se opruga produži ili sažme računa se prema  $F = -kx$ , gdje je  $x$  pomak opruge iz ravnotežnog položaja uz faktor  $k$  koji je konstantan i zove se konstanta opruge. Mnoge elastične pojave aproksimiraju se Hookeovim zakonom.



**Slika 2.2:** Hookeov zakon

U fizici se plastičnost opisuje kao deformacija materijala koja se uslijed djelovanja vanjskih sila ne vraća u svoj početni oblik. Plastične deformacije primijećene su u većini materijala poput kamenja, minerala, metala itd. (Slika 2.3).

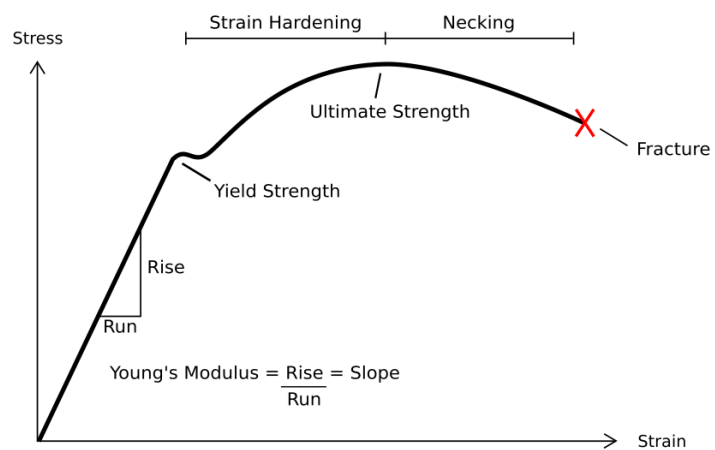
U prirodi se susrećemo s materijalima koji su elastoplastični. Za njih vrijedi da se uz primjenu određene količine vanjske sile na materijal oni ponašaju kao elastični, no uslijed dovoljno velikog povećanja sile počnu se ponašati kao plastični. Prijelaz iz elastičnosti u plastičnost opisan je pojmom koji se zove elastična granica. Daljnjim povećanjem sile na materijal dolazi do pucanja materijala (Slika 2.4).

Također postoje i materijali koji su pseudoelastični, odnosno superelastični. Takvi materijali se pod utjecajem neke vanjske sile ponašaju kao plastični predmeti, naizgled trajno mijenjajući svoj oblik. No, uslijed djelovanja nekog specifičnog vanjskog podražaja na materijal, poput topline ili električne struje, oni se vraćaju u svoj prvobitni položaj čime se ponašaju kao da su elastični (Vadlja, 2011).





**Slika 2.3:** Primjer deformiranih plastičnih objekata u obliku aluminijskih limenki

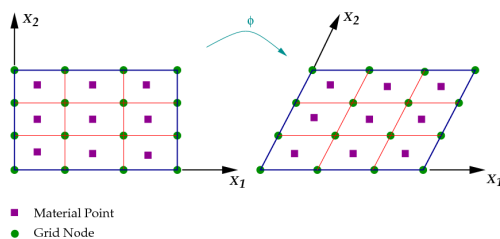


**Slika 2.4:** Dijagram odnosa između naprezanje i deformacije

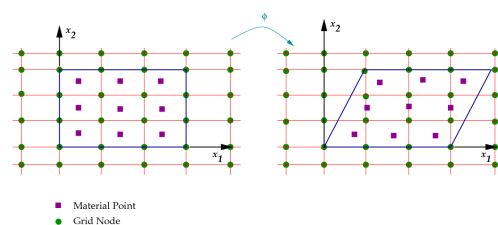
### 3. Metode izračuna deformacija

Kako bi se riješile određene parcijalne jednačbe razvijen je znatan broj različitih numeričkih metoda. S obzirom na opis materijala ili prostora jednačbi sve metode mogu se klasificirati kao Eulerove i Lagrangeove metode. Lagrangeove metode mogu se nadalje podijeliti na metode zasnovane na geometrijskoj mreži (engl. *mesh-based methods*) i bezmrežne metode (engl. *meshless methods*).

Eulerove i Lagrangeove metode razlikuju se po načinu na koji gledaju materijal. Eulerove metode evaluiraju svojstva materijala u stacionarnim točkama u prostoru i računaju kako se ta svojstva mijenjaju kroz vrijeme, dok Lagrangeove metode prate kretanje koordinata materijala. Zbog toga što je geometrijska mreža u Eulerovim metodama fiksna u prostoru, rješavanje jednačbi je brzo i stabilno. Također Eulerove metode mogu podnijeti ekstremne deformacije bez mijenjanja diskretizacije, dok Lagrangeove metode moraju prilagođavati diskretizaciju kako bi izbjegli numeričke probleme. Unatoč tome, Lagrangeove metode sadrže nekoliko prednosti. Pošto je geometrijska mreža (engl. *mesh*) povezana s materijalom koji se kreće, praćenje je vrlo jednostavno i egzaktno. Nadalje, objekt ili fluid je eksplicitno određen preko geometrijske mreže, dok kod Eulerovih metoda granica mora na neki način biti praćena. Lagrangeove metode nisu ograničene na određeno područje u prostoru, za razliku od Eulerovih metoda gdje geometrijska mreža određuje domenu simulacije (Slika 3.1 i Slika 3.2).

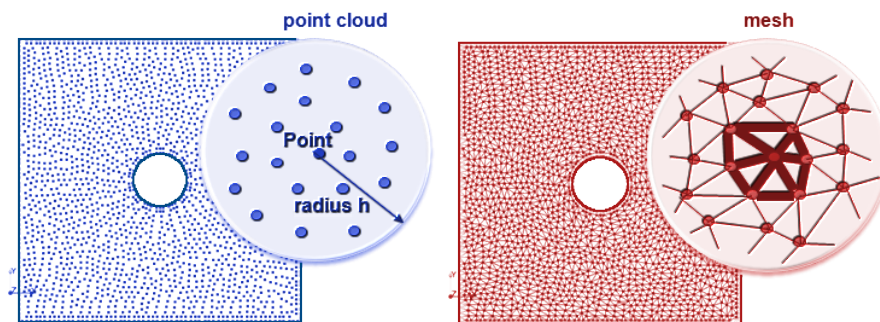


Slika 3.1: Lagrangeova metoda



Slika 3.2: Eulerova metoda

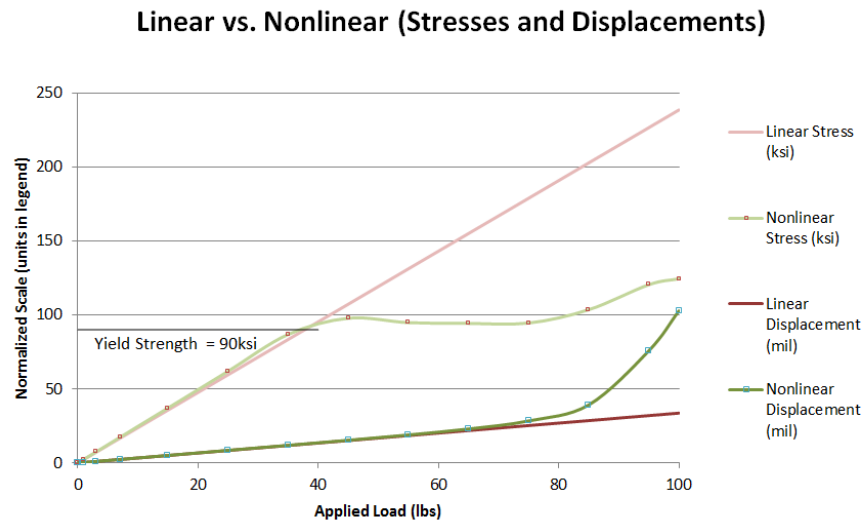
Lagrangeove metode zasnovane na geometrijskoj mreži dijele kontinuum u diskretne elemente koji su zajedno povezani. Takav način nije uvijek pogodan jer je topologija mreže fiksna i zbog toga se teško prilagođava fizičkim promjenama u kontinuumu. Velike deformacije uzrokuju izobličenja mreže koja rezultiraju ozbiljnim problemima u vezi stabilnosti i preciznosti. Topološke promjene zahtijevaju kompleksne operacije izmjena u geometrijskoj mreži koje uvode različite numeričke pogreške. Bezmrežne metode rješavaju navedene probleme tako što ne spremaju povezanost geometrijske mreže, nego aproksimiraju ili interpoliraju svojstva materijala iz interpoliranih točaka korištenjem bezmrežnih oblikovnih funkcija. Interpolirane točke, nazvane česticama, kreću se zajedno s materijalom. Prostorna diskretizacija s volumetrijskim česticama može efikasno i stabilno biti prilagođena tijekom vremena u skladu s promjenama u simulaciji. Dobivena fleksibilnost dolazi uz cijenu većeg računalnog troška potrebnog za računanje bezmrežnih oblikovnih funkcija (Keiser, 2006). U ovom radu detaljnije su objašnjeni sustavi masa i opruga i metoda konačnih elemenata koje pripadaju Lagrangeovim metodama zasnovanim na geometrijskoj mreži.



**Slika 3.3:** Primjer Lagrangeove metode zasnovane na geometrijskoj mreži i bezmrežne metode

Metoda konačnih elemenata jedna je od najraširenijih tehnika u računarskoj znanosti. Metoda dijeli objekt na elemente konačne veličine. Povezujući elemente, polje deformacije je interpolirano kroz cijeli objekt. Na taj način, umjesto rješavanja kontinuiranog vektorskog polja, moraju se izračunati deformacije u diskretnim točkama ili čvorovima, a diferencijalne jednadžbe u čvorovima ponašaju se kao skup linearnih algebarskih jednadžbi (Müller et al., 2001). Određene diferencijalne jednadžbe su nelinearne, a sustavi koji rješavaju nelinearne jednadžbe su prespori za primjenu u stvarnom vremenu. Moguće je raditi s linearnim aproksimacijama, pogotovo ako su deformacije male kao kod analize zgrada (Slika 3.4). Međutim, u slučaju deformabilnih objekata koji se slobodno kreću, mogu se stvoriti neželjeni artefakti (Slika 3.5).

Jedan od načina kako simulacije zasnovane na metodi konačnih elemenata učiniti dovoljno brzim za primjenu u aplikacijama koje se odvijaju u stvarnom vremenu je da se deformacija odvoji u linearni i rotacijski dio.



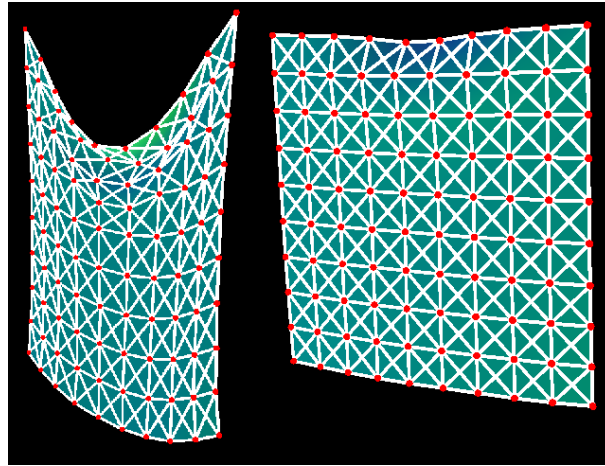
**Slika 3.4:** Dijagram naprežanja i deformacije koji prikazuje odnos između linearnog i nelinearnog načina



**Slika 3.5:** Pojava neželjenih artifakta kod modela psa, na lijevoj slici je glava povećana pri rotaciji, dok je desna slika ispravna

U određenim aplikacijama, objekti su prikazani kao sustav masa i opruga umjesto mreže zasnovanoj na metodi konačnih elemenata. Sustav mase i opruga intuitivniji je i lakši za programirati nego metoda konačnih elemenata s manom da takav sustav ne može prikazivati određena bitna realna svojstva materijala. Ti nedostaci, u određenim scenarijima, predstavljaju značajan problem. Unatoč tome simulacija tkanine najčešće se odvija pomoću sustava masa i opruga (Slika 3.6). Sustav masa i opruga s lakoćom može rukovati s dvodimenzionalnom strukturom tkanine, dok je kod metode konačnih

elemenata potrebno uvesti posebne kompleksne elemente koji omogućuju savijanje tkanine. Takvo rješenje je nepotrebno, kompleksno i presporo za sustave koji rade u stvarnom vremenu (Müller et al., 2008).



Slika 3.6: Tkanina ostvaren na temelju sustava mase i opruga

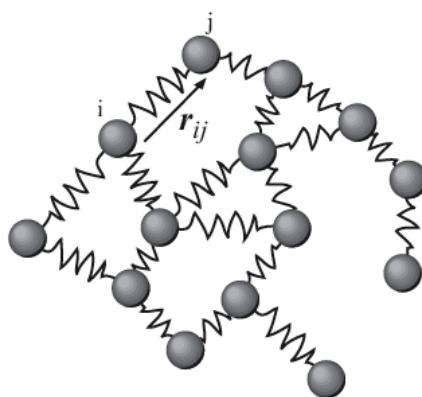
### 3.1. Sustav masa i opruga

Jedan od jednostavnijih načina simuliranja čvrstog objekta je u obliku sustava mase i opruga (Slika 3.7). Sustav masa i opruga sastoji se od skupa točaka koje sadrže mase i povezani su s oprugama. Jednostavnost ovakvog sustava dolazi uz nekoliko nedostataka:

- Ponašanje objekta ovisi o mreži opruga na temelju kojih je objekt povezan
- Ponekad je teško definirati konstante opruge kako bi se dobilo željeno ponašanje.
- Sustav masa i opruga ne može obuhvatiti volumetrijske učinke kao očuvanje volumena ili sprječavanja volumnih inverzija.

Unatoč tome, za određene aplikacije ovi nedostaci nisu bitni. U tim slučajevima, sustav masa i opruga je najbolji odabir jer su brz i lagan način za implementirati deformaciju (Müller et al., 2008).

Sustav masa i opruga sastoji se od skupa čvorova s masom, položajima  $x_i$  i brzinama  $v_i$ , gdje je  $i \in 1 \dots N$ . Ti čvorovi su povezani sa skupom opruga  $S$ , gdje su  $i$  i  $j$



**Slika 3.7:** Sustav masa i opruga

indeksi susjednih čvorova,  $l_0$  inicijalna duljina,  $k_s$  konstanta opruge i  $k_d$  je koeficijent prigušenja. Sile opruge koje djeluju na susjedne čvorove su

$$f_i = f^s(x_i, x_j) = k_s \frac{x_j - x_i}{|x_j - x_i|} (|x_j - x_i| - l_0), \quad (3.1)$$

$$f_j = f^s(x_j, x_i) = -f(x_i, x_j) = -f_i. \quad (3.2)$$

A sile prigušenja su

$$f_i = f^d(x_i, v_i, x_j, v_j) = k_d (v_j - v_i) \frac{x_j - x_i}{|x_j - x_i|}, \quad (3.3)$$

$$f_j = f^d(x_j, v_j, x_i, v_i) = -f_i. \quad (3.4)$$

Zbrajanjem tih sila dobijemo ukupnu silu:

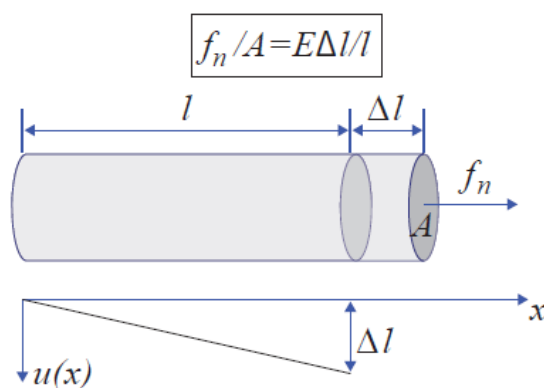
$$f(x_i, v_i, x_j, v_j) = f^s(x_i, x_j) + f^d(x_i, v_i, x_j, v_j). \quad (3.5)$$

## 3.2. Metoda konačnih elemenata

Kao što je prije navedeno, sustav masa i opruga ne može obuhvatiti volumetrijske učinke te ponašanje ovisi o strukturi geometrijske mreže. Kako bi popravili te probleme potrebno je deformirano tijelo promatrati kao kontinuirani volumen. Prema tome, u ovom poglavlju iznošeni su neki osnovni principi mehanike kontinuuma.

### 3.2.1. Mehanika kontinuuma

U računarskoj znanosti, deformabilni objekti su često predstavljeni kao kontinuirani objekti. Kako bi se opisalo željeno ponašanje primarna su tri svojstva: pomak, naprezanje i deformacija. U jednodimenzionalnom sustavu ta svojstva imaju intuitivne reprezentacije. Za primjer definirajmo šipku poprečnog presjeka površine  $A$ . Kada je sila  $f_n$  primijenjena u smjeru okomitom na poprečni presjek, šipka početne duljine  $l$  produljava se za  $\Delta l$  (Slika 3.8).



Slika 3.8: Deformirana šipka

Ta su svojstva povezana Hookeovim zakonom:

$$\frac{f_n}{A} = E \frac{\Delta l}{l}. \quad (3.6)$$

Konstanta  $E$  je Youngov modulus. Za metal  $E$  je reda  $10^{11} N/m^2$  dok je za gumu između  $10^7$  i  $10^8 N/m^2$ . Te jednadžbe ukazuju na to da što je sila veća, to će relativna elongacija  $\Delta l/l$  biti veća. Također, veličina sile po jedinici površine koja je potrebna da bi se dobila određena relativna elongacija povećava se s većim  $E$  što znači da Youngov modulus opisuje čvrstoću šipke. Hookeov zakon može se napisati u obliku

$$\sigma = E\varepsilon. \quad (3.7)$$

gdje je  $\sigma = f_n/A$  primijenjeno naprezanje, a  $\varepsilon = \Delta l/l$  rezultirajuća deformacija. Naprezanje se izražava u sili po jedinici površine, a deformacija nema mjernu jedinicu. Ovo svojstvo također vrijedi i u 3 dimenzije (Müller et al., 2008).

Parcijalne diferencijalne jednađbe koje definiraju objekt dane su prema

$$\rho \ddot{\mathbf{x}} = \nabla \cdot \boldsymbol{\sigma} + f, \quad (3.8)$$

gdje je  $\rho$  gustoća materijala, a  $f$  je vanjska sila koja može biti sila gravitacije ili sila stvorena sudarom tijela. Operator divergencije pretvara tenzor naprezanja u vektor veličine 3 koji predstavlja unutarnju silu iz infinitezimalnog volumena (Jednađba 3.9).

$$\nabla \cdot \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx,x} + \sigma_{xy,y} + \sigma_{xz,z} \\ \sigma_{yx,x} + \sigma_{yy,y} + \sigma_{yz,z} \\ \sigma_{zx,x} + \sigma_{zy,y} + \sigma_{zz,z} \end{bmatrix}, \quad (3.9)$$

Metoda konačnih elemenata koristi se za pretvaranje parcijalnih diferencijalnih u skup algebarskih jednađbi koje se zatim numerički rješavaju. Deformabilni objekt definiran je sa svojim početnim nedeformiranim oblikom (engl. *rest shape*) i sa skupom parametara materijala koji definiraju kako će se objekt deformirati pri utjecaju sile. Ako se inicijalni oblik predstavi kao kontinuirano povezani podskup  $M \in \mathbb{R}$ , onda koordinate  $\mathbf{m} \in M$  su točke objekta nazvane koordinatama materijala.

Kada je određena sila primijenjena, objekt se deformira i točka iz izvorne lokacije  $\mathbf{m}$  prelazi na novu lokaciju  $\mathbf{x}(\mathbf{m})$ . Pošto su nove lokacije definirane za sve točke materijala  $\mathbf{m}$ ,  $\mathbf{x}$  predstavlja vektorsko polje. Deformacija se također može specificirati kao vektor polja pomaka  $\mathbf{u}(\mathbf{m}) = \mathbf{x}(\mathbf{m}) - \mathbf{m}$ . Iz  $\mathbf{u}(\mathbf{m})$  računa se elastična deformacija  $\boldsymbol{\varepsilon}$  (u 1D slučaju  $\boldsymbol{\varepsilon}$  je jednostavno  $\Delta l/l$ ). Popularni odabiri za izračunavanje deformacije su

$$\boldsymbol{\varepsilon}_G = \frac{1}{2}(\nabla \mathbf{u} + [\nabla \mathbf{u}]^T + [\nabla \mathbf{u}^T] \nabla \mathbf{u}) \quad (3.10)$$

$$\boldsymbol{\varepsilon}_C = \frac{1}{2}(\nabla \mathbf{u} + [\nabla \mathbf{u}]^T) \quad (3.11)$$

gdje je  $\boldsymbol{\varepsilon}_G \in \mathbb{R}^{3 \times 3}$  Greenov nelinearni tenzor naprezanja, a  $\boldsymbol{\varepsilon}_C \in \mathbb{R}^{3 \times 3}$  je Cauchyev linearni tenzor naprezanja. Gradient polja pomaka je matrica 3.12 gdje indeks nakon zareza predstavlja prostornu derivaciju (Nealen et al., 2006).



$$\nabla \mathbf{u} = \begin{bmatrix} u_{,x} & u_{,y} & u_{,z} \\ v_{,x} & v_{,y} & v_{,z} \\ w_{,x} & w_{,y} & w_{,z} \end{bmatrix} \quad (3.12)$$

Domena  $M$  je diskretizirana u konačni broj elemenata. Umjesto rješavanja prostorno kontinuirane funkcije  $x(m, t)$ , rješava se diskretni skup nepoznatih položaja  $x_i(t)$  čvorova geometrijske mreže. Prvo, funkcija  $x(m, t)$  je aproksimirana koristeći vrijednosti čvorova

$$\tilde{x}(m, t) = \sum_i x_i(t) b_i(m), \quad (3.13)$$

gdje su  $b_i()$  bazne funkcije fiksnih čvorova koji su jednaki jedan kod čvora  $i$  i nula kod svih ostalih čvorova. Pri zamjeni  $x(m, t)$  s aproksimacijom  $\tilde{x}(m, t)$ , infinitezimalni prostor pretraživanja mogućih rješenja smanjen je u konačno dimenzionalni podprostor. Aproksimacija će stvoriti devijaciju pri zamjeni s parcijalnim diferencijalnim jednadžbama.

U računalnoj grafici često se koristi jednostavan oblik metode konačnih elemenata za simulaciju deformacije objekta, nazvana eksplicitna metoda konačnih elemenata, koja je laka za razumjeti i implementirati. Bitno je da se eksplicitna metoda konačnih elemenata ne zamijeni sa standardom metodom konačnih elemenata koja je eksplicitno integrirana. Eksplicitna metoda konačnih elemenata može biti integrirana ili eksplicitno ili implicitno.

U eksplicitnom načinu konačnim elementima, i mase i vanjska i unutarnja sila su sadržani u čvorovima. Čvorovi u geometrijskoj mreži označeni su kao točke u sustavu masa i opruga, dok se svaki element ponaša kao opruga spajajući susjedne čvorove. Za zadani položaj točaka elementa i baznu funkciju, kontinuirana deformacija polja  $u(m)$  unutar elementa može se računati prema 3.13. Iz  $u(m)$ , polje deformacije  $\varepsilon(m)$  i polje naprezanja  $\sigma(m)$  se mogu računati. Energija deformacije elementa dana je pomoću

$$E = \int_V \varepsilon(m) \cdot \sigma(m) dm, \quad (3.14)$$

gdje točka predstavlja skalarni umnožak svake komponente dva tenzora. Sile se zatim mogu računati kao derivacije energije. Općenito, odnos između sila u čvorovima

i položaja čvorova je nelinearan. Kada se uvede linearizacija, veza između elementa  $e$  koji spaja  $n_e$  čvorove može se jednostavno izraziti kao

$$f_e = K_e u_e, \quad (3.15)$$

gdje  $f_e \in \mathbb{R}^{3n_e}$  sadrži silu čvora  $n_e$  i  $u_e \in \mathbb{R}^{3n_e}$  sadrži pomak čvora  $n_e$ . Matrica  $K_e \in \mathbb{R}^{3n_e \times 3n_e}$  se zove matrica čvrstoće (engl. *stiffnes matrix*) elementa. Zbog toga što se elastične sile koje dolaze od susjednih elemenata zbrajaju u čvorovima, matrica čvrstoće  $K \in \mathbb{R}^{3n \times 3n}$  za cijelu geometrijsku mrežu može biti formirana spajanjem matrica čvrstoće svih elemenata

$$K = \sum_e K_e. \quad (3.16)$$

U ovoj sumi matrica čvrstoće je proširena popunjavajući nule na položajima gdje čvorovi nisu spojeni s određenim elementom. Koristeći linearizirane elastične sile, linearna jednadžba gibanja za cijelu geometrijsku mrežu postaje ( $u = x - x_0$ )

$$M\ddot{u} + D\dot{u} + Ku = f_{ext}, \quad (3.17)$$

gdje je  $M \in \mathbb{R}^{n \times n}$  matricu mase,  $D \in \mathbb{R}^{n \times n}$  matrica prigušenja i  $f_{ext} \in \mathbb{R}^n$  vanjska sile. Često su matrice  $M$  i  $D$  prikazane kao dijagonalne matrice. U tom slučaju,  $M$  sadrži samo mase točaka geometrijske mreže na svojoj dijagonali. Vektori  $x$  i  $x_0$  sadrže trenutni i početni položaj čvorova.

Metoda konačnih elemenata stvara linearni sustav algebarskih jednadžbi ako je primijenjena na linearne parcijalne diferencijalne jednadžbe. Ako je korištena linearna mjera za deformaciju i ako je korišten Hookeov zakon za izotropne materijale koji je supstituiran u 3.8, onda Laméove linearne parcijalne jednadžbe glase:

$$\rho\ddot{x} = \mu\Delta u + (\lambda + \mu)\nabla(\nabla \cdot u), \quad (3.18)$$

gdje su  $\lambda$  i  $\mu$  Laméove konstante materijala i one se mogu izravno izračunati preko Youngovog modulusa i Poissonovog koeficijenta (Müller et al., 2001).

U 3D slučaju često se koristi izračun naprežanja za izotropne materijale:

$$\sigma = 2\mu\varepsilon + \lambda\text{Trace}(\varepsilon)\mathbf{I}, \quad (3.19)$$

Tako su (Parker i O'Brien, 2009) koristili elemente u obliku tetraedara s linearnom baznom funkcijom. Svaki element definiran je s četiri čvora koji sadrže početne položaje  $u_1, u_2, u_3$  i  $u_4$ . Neka je  $D_u$  matrica dimenzija  $3 \times 3$  sa stupcima  $u_2 - u_1, u_3 - u_1$  i  $u_4 - u_1$ , a bazna matrica je zatim  $\beta = D_u^{-1}$ . Svaki čvor ima svoj trenutni položaj i trenutnu brzinu označenu kao  $x_i$  i  $v_i$ . Matrice  $D_x$  i  $D_v$  definirane su na isti način kao i matrica  $D_u$  samo što one sadrže  $x_i$  ili  $v_i$  umjesto  $u_i$ . Gradijent deformacije  $F$  i njegova vremenska derivacija  $G$  označene su kao

$$F = \frac{\partial x}{\partial u} = D_x \beta, \quad (3.20)$$

$$G = \frac{\partial v}{\partial u} = D_v \beta. \quad (3.21)$$

Cauchyev infinitezimalni tenzor deformacije,  $\varepsilon = 1/2(F + F^T) - I$  jeftin je za računanje direktno iz  $F$ , ponaša se linearno s deformacijom i njegov Jakobijan je konstantan u odnosu na  $x_i$ . Sva ta svojstva su poželjna, ali  $\varepsilon$  nije invarijantan u odnosu na rotaciju pa na taj način stvara nepoželjne artefakte kada je primijenjena velika deformacija. Jedan od načina rješavanja navedenog problema je da se koriste korotacijske metode gdje se koriste polarne dekompozicije tako da je  $F = QA$  gdje je  $Q$  ortonormirana, a  $A$  je simetrična matrica. Prema tome, kako degenerirani ili invertirani elementi ne bi stvorili nestabilnost u aplikaciji, provodi se  $QR$  dekompozicija matrice.

Kada imamo izračunat  $Q$  možemo ga faktorizirati s gradijentom deformacije zamjenjujući  $F$  s  $\tilde{F} = Q^T F$  te se deformacija može izračunati kao  $\tilde{\varepsilon} = 1/2(\tilde{F} + \tilde{F}^T) - I$ . Koristeći linearni izotropni konstitutivni model naprezanje se računa prema  $\sigma = \lambda\text{Tr}(\tilde{\varepsilon})I + 2\mu\tilde{\varepsilon}$ . Elastična sila koji vrši element na čvoru  $i$  je  $f_i = Q\sigma n_i$ , gdje je  $n_i$  vanjska normala za trokut koji je suprotan čvoru  $i$ .

Nadalje, koristi se aditivni plastični model koji odvaja ukupnu deformaciju u plastične i elastične komponente tako da je  $\tilde{\varepsilon} = \varepsilon^P + \varepsilon^E$ . Ukupna deformacija  $\tilde{\varepsilon}$  računa se prema definiranom pravilu i na početku  $\varepsilon^E$  zauzima cijelu vrijednost  $\tilde{\varepsilon}$  jer je  $\varepsilon^P = 0$ . Kada  $\|\varepsilon^E\|$  (Frobeniusova norma) prijeđe granicu materijala  $y$ ,  $\varepsilon^P$  se ažurira na temelju

$$\varepsilon^P := \varepsilon^P + c \frac{\|\varepsilon^E\| - y}{\|\varepsilon^E\|} \varepsilon^E, \quad (3.22)$$

gdje je  $c$  parametar materijala koji predstavlja stopu puzanja. Ako  $\|\varepsilon^E\|$  prelazi maksimalnu plastičnu deformaciju materijala  $n$ , onda se ta deformacija postavlja kao  $\|\varepsilon^P\| := \varepsilon^P n / \|\varepsilon^E\|$

### 3.3. Uvođenje vremenske dimenzije

Metode za vremensku integraciju vrednuju se prema dva kriterija: stabilnosti i preciznosti. Preciznost se mjeri konvergencijom prema vezi s vremenskim korakom  $\Delta t$ . U području animacija temeljenih na fizici u računalnoj grafici, stabilnost je često puno bitnija od preciznosti (Nealen et al., 2006).

#### 3.3.1. Eksplicitni Euler

Newtonov drugi zakon gibanja  $f = m\ddot{x}$  primaran je za dobivanje definiranih sila u simulacijskom algoritmu što znači da se akceleracija dobije kao  $\ddot{x} = f/m$ , gdje je  $\ddot{x}$  druga derivacija položaja u odnosu na vrijeme. Navedena formula koristi se kako bi se izračunala ubrzanja čvorova ovisno o silama koje djeluju na njih. Kao prvi korak stvaranju simulacije, diferencijalna jednačnja drugog reda odvaja se u dvije jednačnje prvog reda

$$\dot{v} = f(x, v)/m, \quad (3.23)$$

$$\dot{x} = v. \quad (3.24)$$

Analitička rješavanje ovih jednačnji su

$$v(t) = v_0 + \int_{t_0}^t f(t)/m dt, \quad (3.25)$$

$$x(t) = x_0 + \int_{t_0}^t v(t) dt. \quad (3.26)$$

Krećući od inicijalnih uvjeta  $v(t_0) = v_0$  i  $x(t_0) = x_0$ , integral sumira infinitezimalne promjene do trenutka  $t$ . Simulacije je istovjetna s računanjem  $x(t)$  i  $v(t)$  od

trenutka  $t_0$  te se stoga riječi simulacija i integracija vremena često koriste kao sinonimi. Najjednostavniji način da se jednažbe riješe numerički je da se aproksimiraju derivacije s konačnim razlikama:

$$\dot{v} = \frac{v^{t+1} - v^t}{\Delta t} + O(\Delta t^2), \quad (3.27)$$

$$\dot{x} = \frac{x^{t+1} - x^t}{\Delta t} + O(\Delta t^2), \quad (3.28)$$

gdje je  $\Delta t$  diskretan vremenski korak, a  $t$  je broj slike u simulaciji. Substitucijom aproksimacija s 3.23 i 3.24 dobivaju se dvije jednostavne formule

$$v^{t+1} = v^t + \Delta t f(x^t, v^t)/m, \quad (3.29)$$

$$x^{t+1} = x^t + \Delta t v^t. \quad (3.30)$$

Ovo je eksplicitna Eulerova integracijska shema. Eksplicitna je jer se vrijednosti za sljedeći vremenski korak mogu direktno izračunati preko vrijednosti u trenutnom vremenskom koraku koristeći eksplicitne formule. Jedan način poboljšavanja ove metode je da se varijabla  $v^t$  zamijeni s varijablom  $v^{t+1}$  na desnoj strani jednažbe 3.30. Tako poboljšana formula pripada semi-implicitnim metodama.

Eksplicitna Eulerova metode jedna je od najjednostavnijih integracijskih metoda, ali je zato ona samo stabilna za male vremenske korake. Glavni razlog za nestabilnost je da se Eulerova metoda kreće slijepo u budućnost. Primjer je situacija u sustavu masa i opruga kada je opruga blago nategnuta i kada se spojeni čvorovi počnu kretati jedan prema drugome. S velikim vremenski korakom, čvorovi će prijeći položaj ravnoteže te u sljedećem koraku sile postaju još veće što rezultira eksponencijalnom povećanju energija i na kraju "eksploziji".

Jedan od načina unaprjeđivanja situacije je da se koriste preciznije integracijske metode. Popularni odabiri su Runge-Kutta integratori drugog i četvrtog reda. Navedene metode računaju sile više puta unutar jednog vremenskog koraka kako bi reducirale navedeni problem.

### 3.3.2. Runge-Kutta integracija

Runge-Kutta drugog reda zamjenjuje 3.29 i 3.30 s

$$a_1 = v^t, \quad a_2 = f(x^t, v^t)/m, \quad (3.31)$$

$$b_1 = v^t + \frac{\Delta t}{2}a_2, \quad b_2 = f(x^t + \frac{\Delta t}{2}a_1, v^t + \frac{\Delta t}{2}a_2)/m, \quad (3.32)$$

$$x^{t+1} = x^t + \Delta t b_1, \quad v^{t+1} = v^t + \Delta t b_2, \quad (3.33)$$

kako bi se iz položaja  $x^t$  i brzine  $v^t$  u trenutnom vremenskom koraku dobili  $x^{t+1}$  i  $v^{t+1}$  u sljedećem vremenskom koraku. Prema navedenim formulama sile se moraju evaluirati dva puta tijekom svakog vremenskog koraka što znači da za jedan korak Runge-Kutta traje koliko i dva Eulerova koraka. Međutim, Runge-Kutta je drugog reda u odnosu na preciznost prvog reda Eulerove metode. To znači da kada bi se vremenski korak polovio, koristeći Euleru, dobila bi se polovica pogreške u odnosu na jedne četvrtinu pogreške za Runge-Kutta slučaj. Postoji i Runge-Kutta integracija koja je navedena u nastavku:

$$a_1 = v^t, \quad a_2 = f(x^t, v^t)/m, \quad (3.34)$$

$$b_1 = v^t + \frac{\Delta t}{2}a_2, \quad b_2 = f(x^t + \frac{\Delta t}{2}a_1, v^t + \frac{\Delta t}{2}a_2)/m, \quad (3.35)$$

$$c_1 = v^t + \frac{\Delta t}{2}b_2, \quad c_2 = f(x^t + \frac{\Delta t}{2}b_1, v^t + \frac{\Delta t}{2}b_2)/m, \quad (3.36)$$

$$d_1 = v^t + \Delta t c_2, \quad d_2 = f(x^t + \Delta t c_1, v^t + \Delta t c_2)/m, \quad (3.37)$$

$$x^{t+1} = x^t + \frac{\Delta t}{6}(a_1 + 2b_1 + 2c_1 + d_1), \quad v^{t+1} = v^t + \frac{\Delta t}{6}(a_2 + 2b_2 + 2c_2 + d_2), \quad (3.38)$$

Ova metoda je jedna do najpopularnijih tehnika u računarskoj znanosti. Sile se moraju evaluirati četiri puta što je nagrađeno s preciznošću četvrtog reda. Prepolavljanjem vremenskog koraka rezultira jednom šesnaestinom pogreške.

### 3.3.3. Verlet integracija

Način na koji je Runge-Kutta korištena da unaprijedi stabilnost i preciznost je da evaluira sile više puta tijekom jednog vremenskog koraka. Drugi način je da se vrijednosti računate u prošlosti iskoriste kako bi se povećala razina aproksimacije derivacija iz 3.27 i 3.28. Jedna od popularnijih i jednostavnijih integracija koja se koristi takvim načinom je Verletova integracija. Glavna ideja je da se čuvaju pozicije u trenutku  $t - \Delta t$

u dodatnoj varijabli stanja i da se ta informacija iskoristi za preciznije predviđanje. Taylorov red položaja u dva vremenska smjera iznosi

$$x(t + \Delta t) = x(t) + \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2 + \frac{1}{6}\dddot{x}(t)\Delta t^3 + O(\Delta t^4), \quad (3.39)$$

$$x(t - \Delta t) = x(t) - \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2 - \frac{1}{6}\dddot{x}(t)\Delta t^3 + O(\Delta t^4). \quad (3.40)$$

Zbrajajući te dvije jednačbe dobivamo

$$x(t + \Delta t) = 2x(t) - x(t - \Delta t) + \ddot{x}(t)\Delta t^2 + O(\Delta t^4) \quad (3.41)$$

$$= x(t) + [x(t) - x(t - \Delta t)] + f(t)\Delta t^2/m + O(\Delta t^4). \quad (3.42)$$

Linearni i kubni izrazi se pokrate što završnu formulu čini integracijsko shemom četvrtog reda. Brzina se definira kao  $v(t) = [x(t) - x(t - \Delta t)]/\Delta t$ . Primjenjujuću ovu ideju u diskretnom vremenu, dobivaju se sljedeće formule:

$$x^{t+1} = x^t + v^t\Delta t + f(x^t)\Delta t^2/m, \quad (3.43)$$

$$v^{t+1} = (x^{t+1} - x^t)/\Delta t, \quad (3.44)$$

### 3.3.4. Implicitna integracija

Integracijske metode koje su do sada objašnjene samo su uvjetno stabilni što znači da postoji određeni raspon za vremenski korak  $\Delta t$  za koji je simulacija stabilna. Ovaj raspon, u većini je slučajeva, ovisan o čvrstoći opruge. Što je opruga čvršća to je manji vremenski korak potreban za stabilnost simulacije. U simulacijama koje rade u stvarnom vremenu (npr. video igrama), od bitne je važnosti da je integracija bezuvjetno stabilna što znači da mora biti stabilna u svim okolnostima i za određeni vremenski korak. Jedan način ostvarivanja takvog scenarija je preko implicitnih integracijskih metoda.

Najpopularnija implicitna integracija korištena u računalnoj grafici je implicitni Euler. Eksplisitnog Eulera potrebno je neznatno promijeniti da bi ga se učinilo implicitnim:

$$v^{t+1} = v^t + \Delta t f(x^{t+1})/m, \quad (3.45)$$

$$x^{t+1} = x^t + \Delta t v^{t+1}. \quad (3.46)$$

Iz 3.45 i 3.46 izbačena je sila trenja kako bi suma svih sila bila ovisna samo o položajima. Sila trenja stabilizira sustav tako da ona može biti dodana preko eksplicitnog načina poslije implicitnog rješenja.

Glavna promjena, za razliku od eksplicitnog načina, je da se položaji i brzine u novom vremenskom koraku koriste na desnoj strani. Nije više moguće direktno i eksplicitno evaluirati te dvije jednadžbe. Umjesto toga, imamo dvije implicitne jednadžbe koje formiraju nelinearan algebarski sustav s položajima i brzinama sljedećeg vremenskog koraka kao nepoznanice.

Kao prvi korak pri rješavanju tih jednadžbi potrebno je kombinirati položaje, brzine i sile svih čvorova u dva vektora i matricu sile

$$x = [x_1^T, \dots, x_n^T]^T \quad (3.47)$$

$$v = [v_1^T, \dots, v_n^T]^T \quad (3.48)$$

$$f(x) = [f_1(x_1, \dots, x_n)^T, \dots, f_n(x_1, \dots, x_n)^T]^T. \quad (3.49)$$

Neka je dana dijagonalna matrica mase  $M \in \mathbb{R}^{3n \times 3n}$  s vrijednostima  $m_1, m_1, m_1, m_2, m_2, m_2, \dots, m_n, m_n, m_n$  na dijagonali. Dane definicije vode do sljedećeg sustava jednadžbi:

$$Mv^{t+1} = Mv^t + \Delta t f(x^{t+1}), \quad (3.50)$$

$$x^{t+1} = x + \Delta t v^{t+1}. \quad (3.51)$$

Substituirajući 3.50 i 3.51 dobiva se jedinstveni sustav za nepoznate brzine  $v^{t+1}$  za sljedeći vremenski korak

$$Mv^{t+1} = Mv^t + \Delta t f(x^t + \Delta t v^{t+1}). \quad (3.52)$$

Sustav je nelinearan jer su sile u položajima nelinearne. Generalni način na koji se rješava navedeni sustav jednadžbi je pomoću Newton-Raphsonove metode. Ta metoda linearizira trenutno stanje i počinje na pretpostavci za nepoznati  $v^{t+1}$  te iterativno

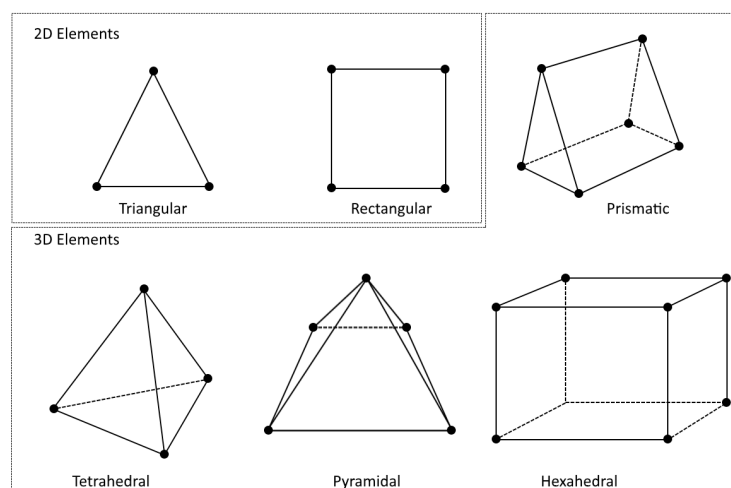


poboljšava rješenje sve dok vrijednost pogreške ne padne ispod određene granice. U pravilu, za aplikacije koje se izvode u stvarnom vremenu, linearizacija koja se izvodi više puta u jednom vremenskom koraku je preskupa. Zbog toga se jednom linearizira, a trenutna brzina  $v^t$  koristi se kao pretpostavka za  $v^{t+1}$  (Müller et al., 2008)).

## 4. Diskretizacija

U računalnoj grafici, dizajneri modeliraju 3D objekt kao geometrijsku mrežu koja predstavlja njegovu površinu. Ta mreža često nije dovoljno prikladna za direktnu simulaciju jer ona ne predstavlja unutrašnjost objekta te može sadržavati loše oblikovane trokute, dvostruke vrhove ili trokute koji se sijeku. Općenito je bolje koristiti dva prikaza određenog objekta, jedan za vizualizaciju i jedan za simulaciju, a ponekad je dodan i treći prikaz mreže za detekciju kolizije. Ako vizualni i simulacijski prikaz nisu isti, potrebno je dodati mehanizam kako bi se vizualna mreža pomakla zajedno s fizičkom mrežom (Müller et al., 2008).

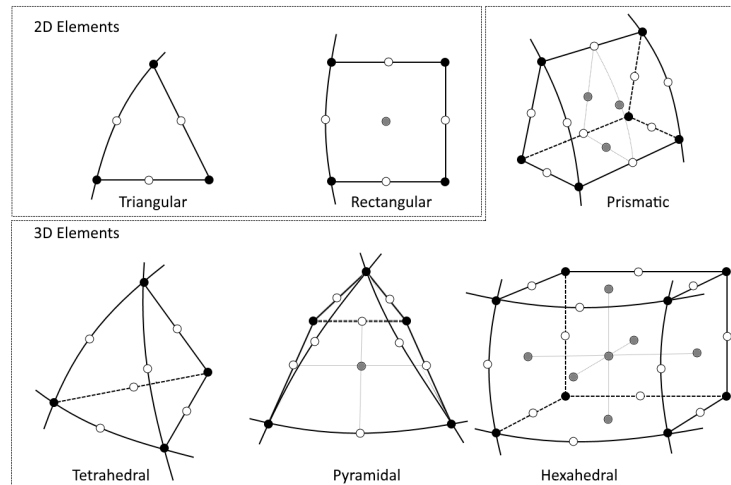
Kao što je ranije navedeno, potrebno je definirati baznu funkciju za konačne elemente te se ta funkcija može definirati na puno načina. Za linearne funkcije u 2D i 3D, najčešći elementi koji se koriste su ilustrirani na slici 4.1. Bazne funkcije su izražene kao funkcije položaja čvorova ( $x$  i  $y$  u 2D i  $x, y, z$  u 3D).



Slika 4.1: Najčešći elementi koji se koriste

Odgovarajući elementi drugog reda prikazani su na slici 4.2. Rubovi i površine koje gledaju u smjeru granice domene su često zakrivljeni, dok su rubovi i površine koje su

usmjerene prema unutarnjem dijelu domene prikazane kao linije ili ravne površine.



Slika 4.2: Elementi drugog reda

## 4.1. Konvergencija mreže

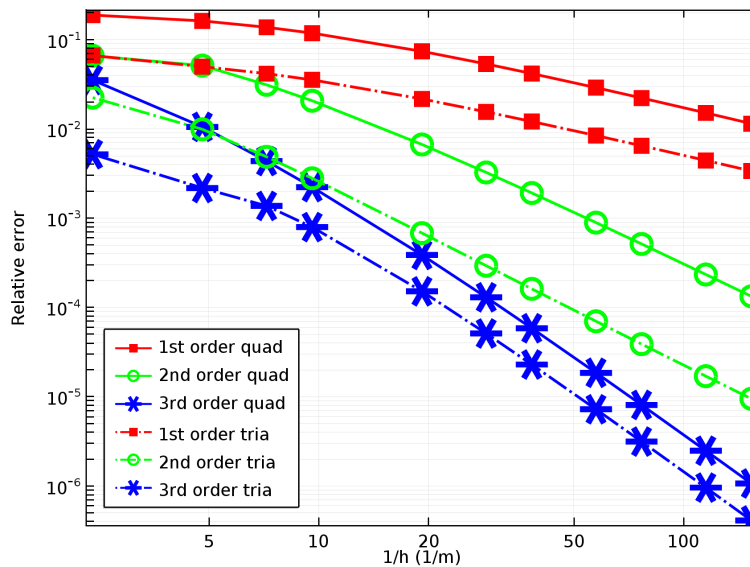
Konvergencija mreže jednostavna je metoda koja uspoređuje aproksimirana rješenja dobivena iz različitih mreža. Aproksimacijska pogreška izvedena iz mreže može biti direktno evaluirana kao

$$e = u - u_h, \quad (4.1)$$

gdje je  $u$  rješenje numeričkih jednadžbi, a  $u_h$  egzaktno rješenje matematičkog modela. U praksi, računanje aproksimacije vrlo glatke mreže može biti dosta zahtjevno. Slika 4.3 prikazuje da se relativna pogreška smanjuje kako se veličina elementa ( $h$ ) smanjuje za sve elemente i krivulja konvergencije postaje strmija kako red baznih funkcija (red elemenata) postaje veći. Međutim, broj nepoznanica u numeričkom modelu raste zajedno s redom elementa za zadanu veličinu. To znači da kada se poveća red elementa, dobiva se veća preciznost u obliku povećanog vremena računanja. Jedna od alternativa je da se stvori glatka mreža pomoću elemenata nižeg reda (Comsol, 2016).

## 4.2. Delaunayeva tetraedarizacija

Popularan odabir elemenata u računalnoj grafici je u obliku trokutastih elemenata za 2D, a tetraedara za 3D simulaciju. Kako bi se stvorili takvi elementi potrebno je prvo



**Slika 4.3:** Relativna pogreška u odnosu na veličinu elementa ( $h$ ) za sve elemente

ispuniti vizualizacijsku mrežu dodatnim točkama. Tradicionalan način provjere je li određena točka unutar mreže je stvaranjem zrake iz određene točke i brojanjem koliko je ta zraka površina trokuta presjekla. Ako je taj broj neparan, onda je točka unutar mreže. Zatim se pokreće Delanauyeva tetraedarizacija (Delaunayeva tringulacija u 2D) na temelju dodanih točaka. Trokute koji su nastali izvan mreže brišemo (Algoritam 1). Ovaj algoritam se na isti način koristi i u dvije dimenzije, samo što su tetraedri zamijenjeni s trokutima, a sfera s krugom (Müller et al., 2008).

Delaunayeva triangulacija ima svojstvo da za skup diskretnih točaka ne postoji točka koja je unutar opisane kružnice nekog trokuta. Graf inverzan Delunayevoj triangulaciji je Voronoiev dijagram pri čemu su vrhovi trokuta u triangulaciji zadane točke za Voronoiev dijagram. Voronoiev dijagram dijeli prostor u ćelije (engl. *Voronoi cells*) gdje je svaka točka u određenoj ćeliji bliža zadanoj točki (Slika 4.4 i Slika 4.5).

Voronoiev dijagram također ima široki spektar mogućnosti te su tako u (Schvartzman i Otaduy, 2014) implementirali lomljenje objekata pomoću Voronoievih dijagrama.

### 4.3. Varijacijska triangulacija

Neovisno o tipu elementa koji je odabran, preciznost i stabilnost numeričkog računanja uvelike ovisio kvalitetnoj geometrijskoj mreži koja se sastoji samo od dobro oblikovanih, nedegenerativnih elemenata, a jedan nepovoljan element može biti dovoljan kako

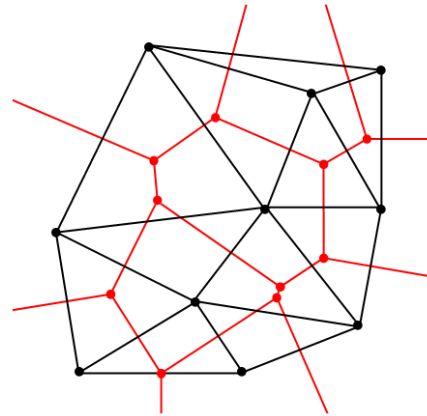
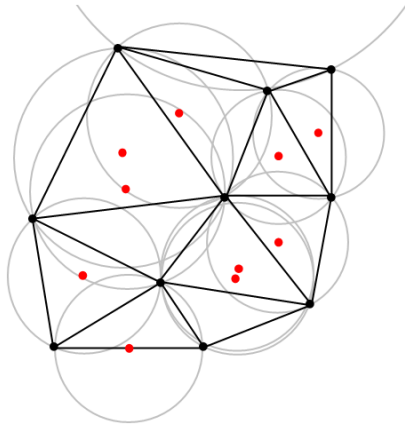
---

**Algorithm 1** Delaunayeva tetraedarizacija

---

```
1: for all points  $p_i$  do  
2:   clear face list  $l$   
3:   for all tetrahedra  $t_j$  the circumsphere of which contains  $p_i$  do  
4:     for all faces  $f_k$  of  $t_j$  do  
5:       if  $l$  contains  $f_k$  then  
6:         remove  $f_k$  from  $l$   
7:       else  
8:         add  $f_k$  to  $l$   
9:       end if  
10:    end for  
11:    delete  $t_j$   
12:  end for  
13:  for all faces  $f_k$  in  $l$  do  
14:    create a tetrahedron from  $f_k$  to  $p_i$   
15:  end for  
16: end for  
17: remove all tetrahedra that contain vertices of the large tetrahedron created in  
    step(1)
```

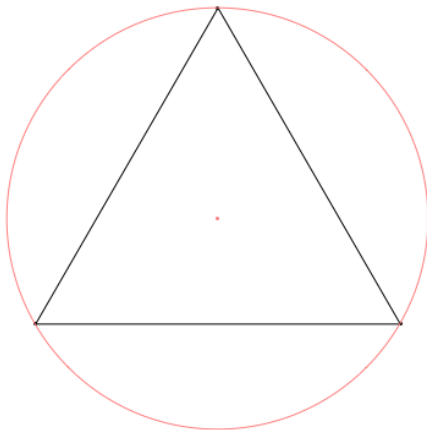
---



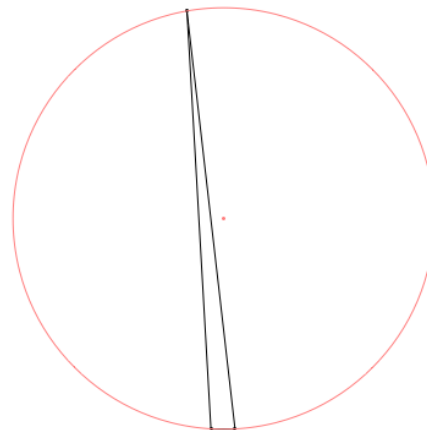
**Slika 4.4:** Prikazana Delaunayeva triangulacija zajedno sa središtima opisanih kružnica **Slika 4.5:** Odnos između Voronoievog dijagrama i Delaunayeve triangulacije

bi "uništio" numerički uvjet. Najčešće je poželjno izotropno umrežavanje gdje je potrebno generirati mrežu koja se sastoji od tetraedara koji su isti, odnosno imaju iste duljine bridova.

Jedan od načina mjerenja kvalitete trokuta u 2D je preko omjera radijusa opisane kružnice trokuta i najkraćeg brida. Manja vrijednost omjera označuje veću kvalitetu trokuta, što se intuitivno čini razumnim jer dugački, "mršavi" trokuti imaju veći radijus opisane kružnice i jedan jako kratki brid (Slika 4.6 i Slika 4.7).



**Slika 4.6:** Trokut dobre kvalitete



**Slika 4.7:** Trokut loše kvalitete

Desni jednakostraničan trokut sadrži omjer vrijednosti  $\sim 0.5$  i najbolji je mogući trokut kada se navedena mjera uzima u obzir. Vrijednost omjera desnog trokuta je  $\sim 8.7$  i središte opisane kružnice takvog trokuta nalazi se izvan njega.

Varijacijski algoritmi popravljaju geometrijsku mrežu tijekom nekoliko optimi-

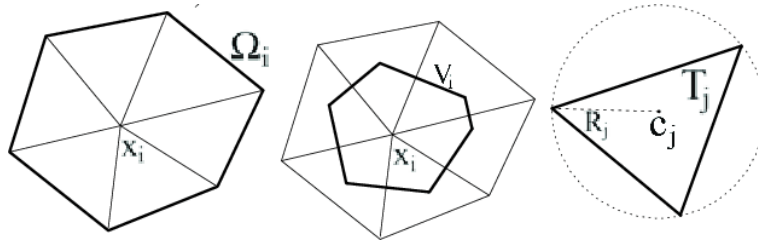
zacijskih koraka koji pokušavaju minimizirati globalnu funkciju energije (Macklin, 2012). U (Alliez et al., 2005) varijacijski način ostvaruje se preko optimizacije povezanosti i optimizacije položaja točaka.

Optimizacija povezanosti je lako ostvariva: za zadani skup položaja točaka  $x_i$ , njegova Delaunayeva triangulacija je optimalna povezanost koja minimizira energiju  $E_{ODT}$ . Prema tome, za svaku iteraciju računa se Delaunayeva povezanost, tako osiguravajući optimalnu povezanost.

Optimizacija položaja točaka ostvaruje se za geometrijsku mrežu  $M$  s točkama  $x_i$  i  $E_{ODT}$  može napisati kao:

$$E_{ODT} = \frac{1}{4} \sum_i x_i^2 |\Omega_i| - \int_M x^2 dx, \quad (4.2)$$

gdje je  $|\Omega_i|$  mjera (volumen u 3D) susjedstva 1-prstena točke  $x_i$  (Slika 4.8).



**Slika 4.8:** Nomenklatura: Lijevo:  $\Omega_i$  označujemo kao 1-prsten točke  $x_i$ . Sredina:  $V_i$  je Voronoieva ćelija točke  $x_i$ . Desno: Središte opisane kružnice trokuta  $T_j$  je označeno kao  $c_j$ , dok je radijus označen kao  $R_j$

Derivacija navedene kvadratne energije u  $x_i$  vodi do sljedećeg optimalnog položaja  $x_i^*$  unutarnje točke  $x_i$  u svojem 1-prstenu.

$$x_i^* = x_i - \frac{1}{2|\Omega_i|} \sum_{T_j \in \Omega_i} \left( \nabla_{x_i} |T_j| \left[ \sum_{x_k \in T_j} \|x_i - x_k\|^2 \right] \right). \quad (4.3)$$

Kako bi poboljšali razumijevanje prednosti ovog varijacijskog načina navedena optimizacija se može evaluirati u geometrijskom izrazu:

$$x_i^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| c_j, \quad (4.4)$$

gdje je  $c_j$  središte opisane kružnice tetraedra  $T_j$ . Prethodni izraz pokazuje da, premda se svaka točka pomiče prema prosječnoj vrijednosti, optimalni smještaj ponajviše ovisi o lokalnoj distribuciji. Na primjer, ako su 1-prsten susjedi na zajedničkoj

sferi, onda će optimalan položaj biti središte te sfere što znači da su optimalni položaji ovisni samo o 1-prsten susjedstvu, a ne o trenutnoj lokaciji. Prema navedenim pravilima za poboljšanje Delaunayeve triangulacije može se izraziti sljedeći algoritam:

---

**Algorithm 2** Delaunayeva tetraedarizacija

---

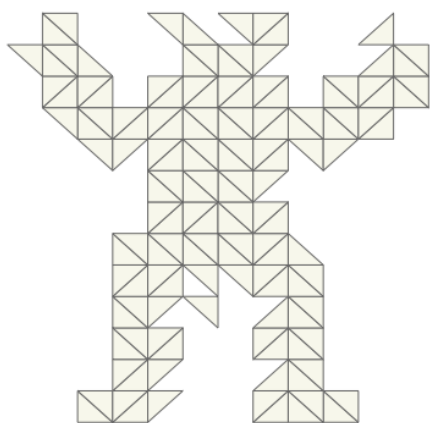
- 1: Generate a set of uniformly distributed points interior to the shape P
  - 2: Generate a set of points on the boundary of the shape B
  - 3: Generate a Delaunay triangulation of P
  - 4: Optimize boundary points by moving them to the average of their neighbours in B
  - 5: Optimize interior points by moving them to the centroid of their Voronoi cell (area weighted average of connected triangle circumcenters)
  - 6: Unless stopping criteria met, go to 3.
  - 7: Remove boundary sliver triangles
- 

Slika 4.9 prikazuje objekt na temelju kojeg se želi generirati geometrijska mreža koja sadrži trokute kao konačne elemente. Preko Delaunayeve triangulacije dobije se mreža prikazana na slici 4.10, dok je na slici 4.11 prikazan mreža nakon sedme iteracije varijacijskog načina triangulacije. Na temelju navedenih slika vidljiva je prednost varijacijskog pristupa. Mreža prikazana na 4.11 kvalitetnije obavija definirani model te smanjuje mogućnost za nastanka degeneriranih trokuta.

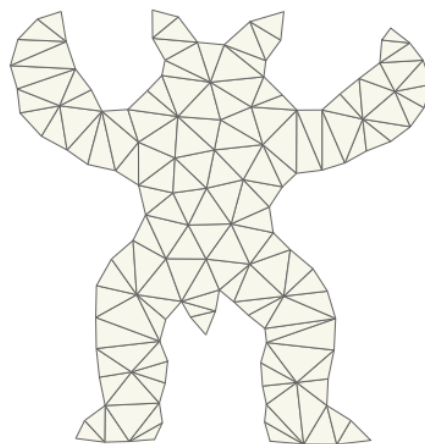




**Slika 4.9:** Objekt na temelju kojeg se želi generirati geometrijska mreža



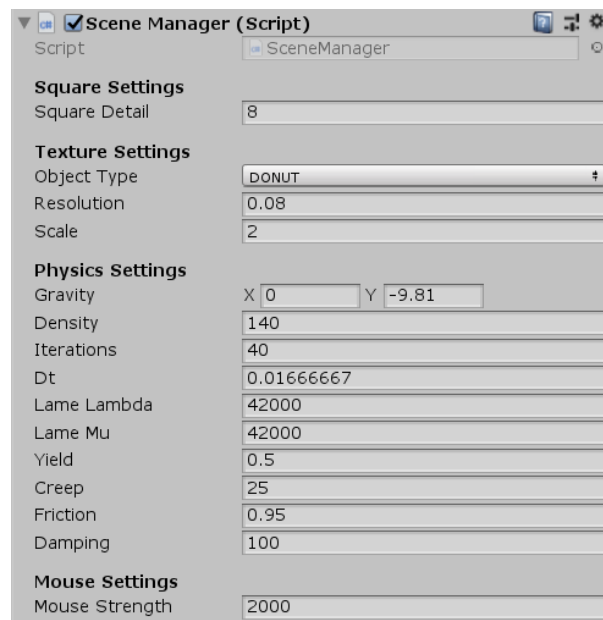
**Slika 4.10:** Početna Delaunayeva triangulacija



**Slika 4.11:** Varijacijska triangulacija nakon sedme iteracije

## 5. Implementacija

Kao primjer napisana je aplikacija koja prikazuje 2D simulaciju deformacije. Aplikacija je napisana u Unity grafičkom alatu, a za programski jezik korišten je C#. Klikom na objekt *SceneManager* u *Inspektoru* prikazana je skripta *Scene Manager* koja sadrži različite vrijednosti za podešavanje simulacije (Slika 5.1).



Slika 5.1: Skripta Scene Manager

**Square Settings:** Postavke za generiranjem objekta u obliku kvadra

**Square Detail:** razina detalja kvadra, za cijeli broj  $n$  generirat će se  $n^2$  točaka uniformno unutar kvadra

**Texture Settings:** Postavke za generiranjem objekta

**Object Type:** Vrsta objekta koji će se stvoriti, u izborniku su sljedeće opcije ponuđene: krafna (engl. *donut*) (Slika 5.2), oklopnik (engl. *armadillo*)

(Slika 5.3), zec (engl. *bunny*) (Slika 5.4), trokut (engl. *triangle*) (Slika 5.6), kvadar (engl. *square*) (Slika 5.5)

**Resolution:** Broj točaka koji se stvara unutar teksture, što je vrijednost veća, to će se manje točaka stvoriti

**Scale:** Veličina objekta

**Physics Settings:** Postavke za fiziku

**Gravity:** Gravitacija

**Density:** Gustoća objekta

**Iterations:** Broj iteracija računanja sila unutar jednog vremenskog koraka kako bi se simulacija stabilizirala

**Dt:** Vremenski razmak

**Lame Lambda:** Lambda konstanta za linearni izotropni model

**Lame Mu:** Mu konstanta za linearni izotropni model

**Yield:** Granica između plastičnosti i elastičnosti

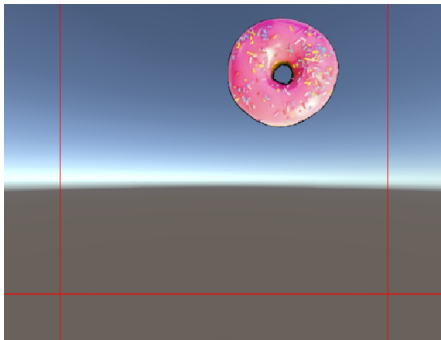
**Creep:** Stopa puzanja

**Friction:** Koeficijent trenja

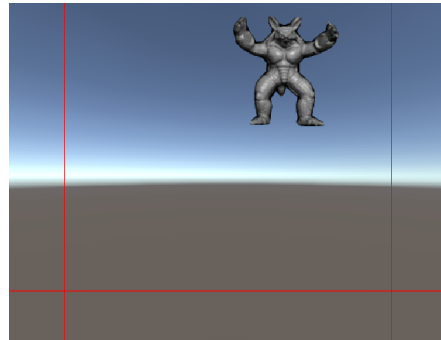
**Damping:** Koeficijent prigušenja

**Mouse Settings:** Postavke za pokazivač miša

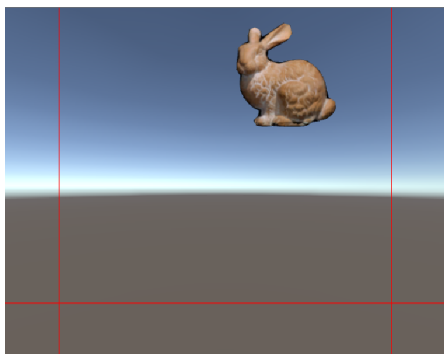
**Mouse Strength:** Jačina sile u smjeru pokazivača miša



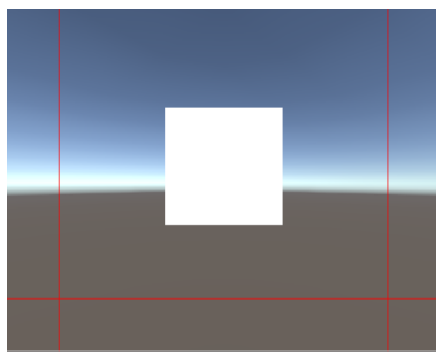
Slika 5.2: Model krafne



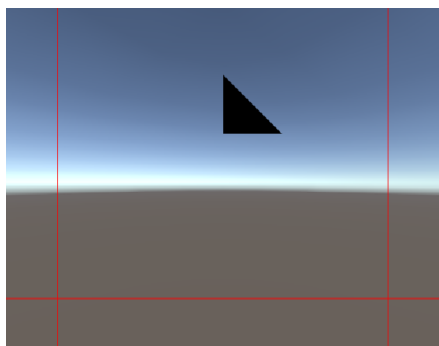
Slika 5.3: Model oklopnika



**Slika 5.4:** Model zeca

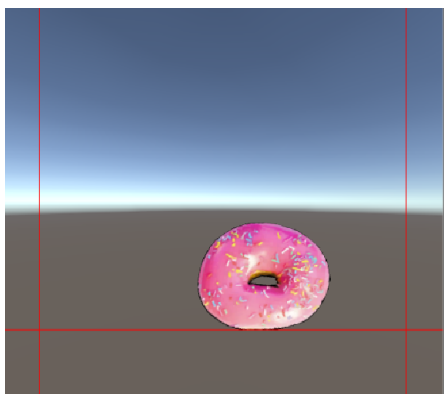


**Slika 5.5:** Model kvadra

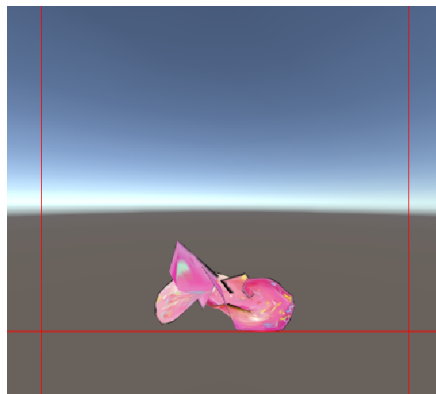


**Slika 5.6:** Model trokuta

Simulacija se odvija na temelju zadanih parametara. Potrebno je biti oprezan s vrijednostima parametara jer simulacija za određene vrijednosti može lako postati nestabilna. Npr. za veću čvrstoću tijela i veći broj točaka potrebno je povećati broj iteracija. Slika 5.8 prikazuje nestabilnu simulaciju uz vrijednosti parametara  $\text{Lame Lambda: } 45000$ ,  $\text{Lame Mu: } 45000$ ,  $\text{Iterations: } 10$ , dok Slika 5.7 prikazuje simulaciju koja je stabilizirana povećavanjem broja iteracija na 40. Međutim u prvom slučaju vrijednost FPS-a je 60, dok je u drugom slučaju 17 zbog veće količine računanja.



**Slika 5.7:** Stabilna simulacija



**Slika 5.8:** Nestabilna simulacija

Crvene crte označuju područje kolizije s objektima odnosno na neki način predstavljaju "zidove" preko kojih objekt ne smije proći (Slika 5.7). Interakcija se odvija tako da se lijevim klikom miša dodaje sila u smjeru pokazivača miša.

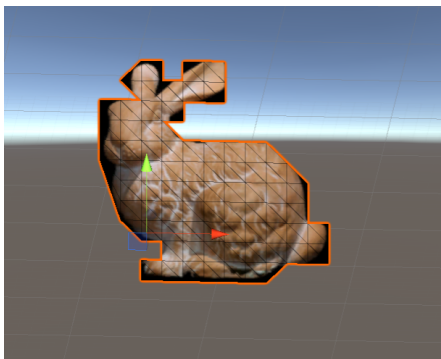
## 5.1. Triangulacija

Triangulacija je odrađena pomoću Delaunayeve metode. Za objekte krafne, oklopnika i zeca točke se generiraju na temelju zadane teksture tako što će se točke postaviti na mjesta piksela koji nije crn. Za objekte kvadra i trokuta točke su unaprijed zadane (Isječak 5.1).

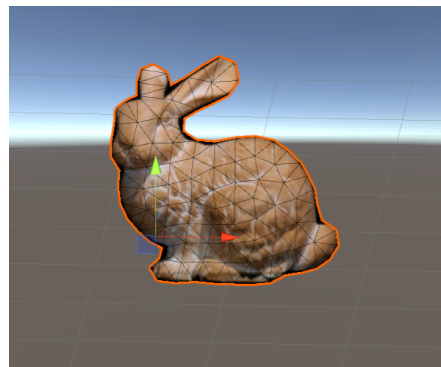
**Isječak 5.1:** Funkcija koja generira mrežu za trokut

```
1 public Mesh CreateTriangle ()
2 {
3     Mesh mesh = new Mesh ();
4     Vector3 [] vertices = new Vector3 [] { new Vector2 (0, 1), new
5         Vector2 (1, 0), new Vector2 (0, 0) };
6     int [] triangles = new int [] { 0, 1, 2 };
7     mesh.vertices = vertices;
8     mesh.triangles = triangles;
9     return mesh;
10 }
```

Kod za Delaunayevu triangulaciju istovjetan je s algoritmom 1 samo što je u ovom slučaju prilagođen za 2D. Kako bi se popravila topologija geometrijske mreže implementiran je varijacijski pristup stvaranju triangulacije (Slika 5.9 i Slika 5.10).



**Slika 5.9:** Delaunayeva triangulacija



**Slika 5.10:** Varijacijska triangulacija

## 5.2. Metoda konačnih elemenata

Na temelju stvorenih trokuta stvaraju se liste čvorova i elemenata koji se dalje koriste za računanje sila. Isječak 5.2 prikazuje klasu koja predstavlja pojedini čvor. Ona sadrži attribute kao što su sila, brzina, početni položaj, trenutni položaj, masa čvora i indeks točke u geometrijskoj mreži objekta, dok isječak 5.3 prikazuje pojedini element koji sadrži tri cjelobrojne varijable  $a$ ,  $b$ ,  $c$  koje označuju indeks točke u geometrijskoj mreži, baznu  $2 \times 2$  matricu odnosno inverz početnih konfiguracija  $mInvDm$ , plastičnu deformaciju  $mEp$  i vanjske normale bridova  $mB$ .

**Isječak 5.2:** Node klasa koja predstavlja čvor

```
1
2 public class Node {
3     public Vector2 Force;
4     public Vector2 Velocity;
5     public Vector2 InitialPosition;
6     public Vector2 Position
7     public float Mass;
8
9     private int index;
10
11     ...
12 }
```

**Isječak 5.3:** Element klasa koja predstavlja element

```
1 public class FemElement {
2     public int a;
3     public int b;
4     public int c;
5
6     public Matrix2x2 mInvDm;
7     public Matrix2x2 mEp = new Matrix2x2 ();
8     public Vector2 [] mB;
9
10    ...
11 }
```

Računanje sila za element je implementirano na temelju (Parker i O'Brien, 2009). Sljedeći isječak prikazuje funkciju koja računa sile za određeni element.

**Isječak 5.4:** Računanje sila za pojedini element

```
1 private void CalculateFEMForce(FemElement elem, Node[] nodes)
```

```

2      {
3          Node node0 = nodes[elem.a];
4          Node node1 = nodes[elem.b];
5          Node node2 = nodes[elem.c];
6
7          Vector3 pos0 = node0.Position;
8          Vector3 pos1 = node1.Position;
9          Vector3 pos2 = node2.Position;
10
11         Vector3[] dx = new Vector3[] { pos0, pos1, pos2 };
12         Matrix2x2 f = CalculateDeformation(dx, elem.mInvDm);
13         Matrix2x2 q = MathUtil.QRDecomposition(f);
14         // strain
15         Matrix2x2 e = CalcCauchyStrainTensor(q.Transpose * f);
16
17         // update plastic strain
18         float ef = FrobeniusNorm(e);
19
20         if (ef > Yield)
21             elem.mEp += e * dt * Creep;
22
23         const float epmax = 0.6f;
24         if (ef > epmax)
25             elem.mEp *= epmax / ef;
26
27         // adjust strain
28         e -= elem.mEp;
29
30         Matrix2x2 stress = CalculateStressTensor(e, LaméLambda,
31             LaméMu);
32         // damping forces
33         Vector3[] dv = new Vector3[]
34         {
35             node0.Velocity,
36             node1.Velocity,
37             node2.Velocity
38         };
39         Matrix2x2 dfdt = CalculateDeformation(dv, elem.mInvDm);
40         Matrix2x2 dedt = CalcCauchyStrainTensorDt(q.Transpose * dfdt
41             );
42         Matrix2x2 dsdt = CalculateStressTensor(dedt, Damping,
43             Damping);

```

```

42         Matrix2x2 p = stress + dsdt;
43
44         Vector3 f1 = q * p * elem.mB[0];
45         Vector3 f2 = q * p * elem.mB[1];
46         Vector3 f3 = q * p * elem.mB[2];
47
48         nodes[elem.a].AddForce(-f1 / 3.0 f);
49         nodes[elem.b].AddForce(-f2 / 3.0 f);
50         nodes[elem.c].AddForce(-f3 / 3.0 f);
51     }

```

Na liniji 12 odvija se računanje deformacije za baznu matricu  $mInvDm$  i trenutne položaje čvorova u elementu trokuta. Metoda *CalculateDeformation* za svaku točku u trokutu preko vektora položaja  $u_1, u_2, u_3$  stvara  $2 \times 2$  matricu  $D_x$  tako da su stupci redom  $u_2 - u_1, u_3 - u_1$  te zatim koristi formulu  $F = D_x \cdot mInvDm$  kako bi se dobio gradijent deformacije.

Računa se QR dekompozicija dobivenog gradijenta kako bi se uklonili nepovoljni artefakti (Linija 13). Novi gradijent se dobiva preko  $\tilde{F} = Q^T F$  te se on koristi kako bi kako bi se izračunala Cauchyeva linearna deformacija  $\tilde{\varepsilon} = 1/2(\tilde{F} + \tilde{F}^T) - I$  (metoda *CalcCauchyStrainTensor*, linija 15). Ako vrijednost deformacije (Frobeniusova norma) prelazi određeni prag (engl. *Yield*), plastična deformacije se ažurira te ako plastična deformacija prelazi maksimalnu vrijednost  $epmax$ , onda se plastična deformacija osvježava prema  $\varepsilon^P = \varepsilon^P n / \|\varepsilon^P\|$  (Linija 20 - 25). Elastična deformacija  $\varepsilon^E$  zauzima cjelokupno mjesto deformacije  $\tilde{\varepsilon}$  pa se zato na liniji 28 računa  $\tilde{\varepsilon} = \tilde{\varepsilon} - \varepsilon^P$ . Isti proces se provodi i za računanje deformacija prigušenja (linija 30 - 40). Na kraju se deformacije zbrajaju (linija 42) te se sile računaju prema  $f_i = Q \cdot p \cdot mB[i]$  (linija 42 - 50).

### 5.3. Detekcija kolizije

Za svrhu testiranja dodana je jednostavna detekcija kolizije koja prema zadanim vektorima određuje ravnine preko kojih objekt ne smije proći. Algoritam detekcije kolizije djeluje na način da vraća točke unutar granica ako one izlazi izvan njih i ažurira vektor brzine (Isječak 5.5).

**Isječak 5.5:** Metoda za detekciju kolizije

```

1     private void CollidePlanes ()
2     {
3         for (int i = 0; i < nodes.Length; ++i)

```



```

4      {
5          for (int p = 0; p < planes.Length; ++p)
6          {
7              Vector2 n = planes[p];
8              float d = Vector2.Dot(nodes[i].Position, n) + planes
              [p].z;
9
10             if (d < 0.0f)
11             {
12                 nodes[i].Position -= d * n;
13
14                 float rv = Vector2.Dot(nodes[i].Velocity, n);
15
16                 if (rv < 0.0f)
17                 {
18                     Vector2 nv = -rv * n;
19
20                     Vector3 tv = (nodes[i].Velocity + nv) *
                     Friction;
21
22                     // update velocity
23                     nodes[i].Velocity = tv;
24                 }
25             }
26         }
27     }
28 }

```

Vektori ravnina za detekciju kolizije iznose [1, 0, 2.8], [0, 1, 2.8], [-1, 0, 2.8] gdje prvi i drugi član u vektoru određuju smjer vektora, a zadnja varijabla proširuje granice za određenu vrijednost. Linije 7 - 10 provjeravaju nalazi li se točka unutar granica. Ako je točka izvan granica, onda joj se postavlja novi položaj unutar granica (Linija 12). Linije 14 i 16 provjeravaju smjer vektora brzine. Ako je točka izvan granica i ako smjer vektora brzine "gleda" van granica, onda se vektor brzine ažurira tako da se smanjuje s vektorom suprotnog smjera (Linije 18 - 23).

## 5.4. Integracija vremena

Za integraciju vremena implementirana je semi-implicitna Eulerova metoda. Odabrana je iz razloga jer je često korištena u sustavima u stvarnom vremenu, laka je i intuitivna za programirati te ne zahtijeva puno računanja.

### Isječak 5.6: Semi-implicitna Eulerova integracija vremena

```
1  public Node CalculateSemiImplicitEuler(Node node, float dt)
2  {
3      node.Velocity += (node.Force / node.Mass) * dt;
4      node.Position += node.Velocity * dt;
5      node.Force = Vector3.zero;
6      return node;
7  }
```

Isječak 5.6 računa brzinu prema drugom Newtonovom zakonu na temelju ukupne sile i koristi novu brzinu za izračun novoga položaja.

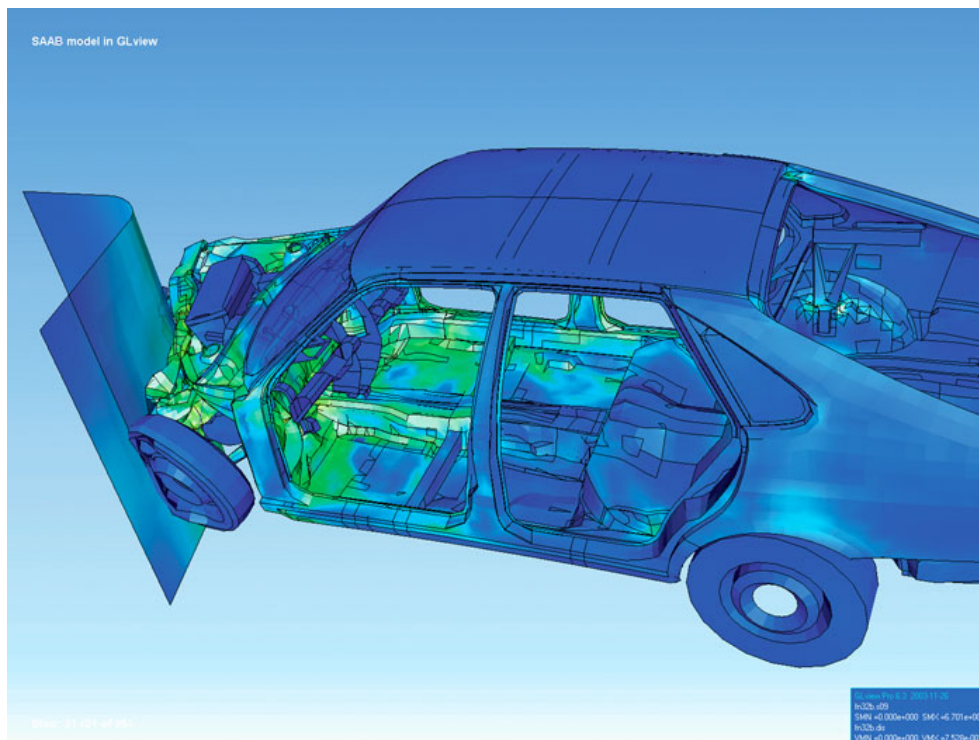
## 6. Zaključak

Iako postoji znatan broj različitih metoda i načina za ostvarivanje simulacije deformacije objekata, samo mali skup metoda ostvaruje rezultate prigodne za sustave u stvarnom vremenu pošto takvi sustavi zahtijevaju povećanu potrošnju resursa u odnosu na *offline* sustave. Prema tome, za sustave koji rade u stvarnom vremenu bitnija je stabilnost od preciznosti. Najčešće se koriste različiti trikovi i aproksimacije koji se fokusiraju na vizualan prikaz fizičkog izgleda deformacije, a ne na egzaktno fizički točno rješenje. Takvim načinom postiže se smanjena količina računanja i dobiva se na brzini izvođenja sustava kako bi simulacija tekla bez zastajkivanja.

Jedan od jednostavnijih i intuitivnijih načina izvođenja deformacija je pomoću sustava masa i opruga. Nedostatak takvog sustava je u tome što nije u mogućnosti prikazati određena bitna svojstva materijala te se zbog toga najčešće koristi za simulaciju tkanine i kose. Metoda konačnih elemenata proširuje sustav masa i opruga jer ona umjesto elementa opruge prihvaća i druge različite vrste elemenata ovisno o potrebnoj primjeni. Na taj način može prikazati realističniji model deformabilnog objekta. Iako je računski zahtjevnija koriste se različite verzije metode konačnih elemenata koje minimiziraju količinu računanja. Primjena metode konačnih elemenata je širokog opsega, od automobilske, aeronautičke, biomehaničke industrije pa sve do industrije video igara (Slika 6.1). Tako su (Parker i O'Brien, 2009) pokazali da se metoda konačnih elemenata može koristiti u okruženju video igre "Star Wars: The Force Unleashed".

Za potrebe rada implementirana je 2D simulacija deformacije pomoću metode konačnih elemenata u stvarnom vremenu. Korištena je Delaunayeva triangulacija za diskretizaciju objekta i razvijena je jednostavna detekcija kolizije. Potrebno je pažljivo rukovati s parametrima simulacije jer svaki od parametara utječe na stabilnost aplikacije. Također s većim brojem točaka i većom čvrstoćom potrebno je povećati broj iteracija ili smanjiti vremenski razmak što donosi povećanu količinu računanja. Daljnja poboljšanja u implementaciji mogu biti uvođenje loma, detekcija kolizije s ostalim objektima, proširenje na 3D itd.

Područje simulacije fizičkih svojstava u računalnoj grafici predstavlja interdisciplinarno polje koje se aktivno istražuje i u kojem uvijek ima prostora za poboljšanje i optimizaciju.



**Slika 6.1:** Vizualizacija deformacije auta koristeći metodu konačnih elemenata

# LITERATURA

- Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, i Mathieu Desbrun. Variational tetrahedral meshing. U *ACM SIGGRAPH 2005 Courses*, stranica 10. ACM, 2005.
- Comsol. The finite element method (fem), 2016. URL <https://www.comsol.com/multiphysics/finite-element-method>.
- Peter Kaufmann. *Discontinuous Galerkin FEM in Computer Graphics*. Doktorska disertacija, ETH Zurich, 2012.
- Richard Keiser. *Meshless Lagrangian methods for physics-based animations of solids and fluids*. Doktorska disertacija, ETH Zurich, 2006.
- Miles Macklin. 2d fem, 2012. URL <http://blog.mmacklin.com/2012/06/27/2d-fem/>.
- Matthias Müller, Leonard McMillan, Julie Dorsey, i Robert Jagnow. Real-time simulation of deformation and fracture of stiff materials. U *Computer Animation and Simulation 2001*, stranice 113–124. Springer, 2001.
- Matthias Müller, Jos Stam, Doug James, i Nils Thürey. Real time physics: class notes. U *ACM SIGGRAPH 2008 classes*, stranica 88. ACM, 2008.
- Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, i Mark Carlson. Physically based deformable models in computer graphics. U *Computer graphics forum*, svezak 25, stranice 809–836. Wiley Online Library, 2006.
- Eric G Parker i James F O’Brien. Real-time deformation and fracture in a game environment. U *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, stranice 165–175. ACM, 2009.
- Sara C Schwartzman i Miguel A Otaduy. Fracture animation based on high-dimensional voronoi diagrams. U *Proceedings of the 18th meeting of the ACM*

*SIGGRAPH Symposium on Interactive 3D Graphics and Games*, stranice 15–22.  
ACM, 2014.

M Vadjla. Modeliranje i simulacija deformabilnih objekata. *Zagreb: Sveučilište u Zagrebu, Fakultet elektronike i računarstva*, 2011.

## Deformacije objekta metodom konačnih elemenata

### Sažetak

U ovom radu izložene su metode za deformiranjem objekata, a posebice su proučene metode koje se temelje na konačnim elementima. Proučene su metode koje se koriste za diskretizaciju objekta na konačne elemente pri čemu je istaknuta Delaunayeva triangulacija i Varijacijska triangulacija. Objasnjen je fizički model deformabilnog objekta, a od metoda za izračun deformacija opisan je sustav masa i opruga i metoda konačnih elemenata. Proučene su različite metode vremenske integracije. Implementirana je 2D simulacija deformacije u stvarnom vremenu u Unity grafičkom alatu i C# programskom jeziku. U aplikaciji je deformacija ostvarena pomoću metoda konačnih elemenata koji su generirani preko Delaunayeve triangulacije i Varijacijske metode, a za vremensku integraciju je korišten semi-implicitni Euler. Također, implementirana je jednostavna detekcija kolizije. Prikazani su rezultati simulacije za određene vrijednosti parametara te je objašnjeno na koji način neki parametri utječu na stabilnost.

**Ključne riječi:** računalna grafika, deformacija, Unity, elastičnost, plastičnost, Delaunayeva triangulacija, Hookeov zakon, semi-implicitni Euler, simulacija

## **Object deformation based on the finite element method**

### **Abstract**

This paper presents methods for object deformation, especially methods based on finite elements. Methods used to discretize the object to finite elements were studied, highlighting Delaunay's triangulation and Variational triangulation. The physical model for a deformable object is explained, and from the methods for calculating the deformation, the mass and spring system and the finite element method is presented. Different methods for time integration have been studied. A 2D real-time deformation simulation was implemented in Unity and C# programming language. In the application, deformation was accomplished by the finite element method that was generated through Delaunay triangulation and variational method. Semi-implicit Euler was used for time integration. Also, a simple collision detection has been implemented. The simulation results for certain parameters values are shown and it is explained how some parameters affect stability.

**Keywords:** computer graphics, deformation, Unity, elasticity, plasticity, Delaunay triangulation, Hooke's law, semi-implicit Euler, simulation