SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 2460

# POSTUPCI IZRADE ELASTIČNO DEFORMABILNIH OBJEKATA

Damjan Križaić

Zagreb, lipanj 2012.

# SADRŽAJ

# 1. Uvod

U računalnoj animaciji se od njezinih začetaka koriste modeli krutih tijela (eng. *rigid body*). S vremenom se javila želja i potreba za sve realnijim računalnim animacijama. Nažalost, stvaranje realističnih animacija pomoću modela krutih tijela nije praktično i predstavlja pravi izazov jer se svodi na premještanje vrhova modela između kojih nema nikakvih dinamičkih procesa. Razvojem računarske znanosti je došlo do razvoja modela mekih tijela (eng. *soft body*) koji su olakšali računalnu animaciju i učinili ju realističnijom.

Meka tijela su računalni modeli nad kojima je omogućena simulacija djelovanja sila i samim time deformiranje njihovih površina. Mogu se opisivati svojstvima plastičnosti i elastičnosti, temperaturom, masom itd. Rezultati dobiveni simulacijom ponašanja mekih tijela stvaraju dojam ponašanja koje bi to tijelo imalo u stvarnosti. Ona se primjenjuju uglavnom u računalnoj animaciji, industriji video igara i medicini.

Jedan podskup mekih tijela su elastična tijela koja od svojstava mekih tijela koriste elastičnost ali ne i plastičnost. U ovom radu fokus će biti na simulaciji elastično deformabilnih objekata. Elastično deformabilni objekti su objekti koji se uslijed deformacije uzrokovane djelovanjem vanjske sile, nakon prestanka djelovanja te sile vraćaju u svoj prvotni oblik. Imaju stalni volumen i opiru se deformaciji. Neki primjeri iz stvarnog svijeta za elastična tijela su gumena lopta, opruga i elastin.

# 2. Fizikalne osnove

Da bi pomoću računala modelirali elastično tijelo i simulirali sile koje djeluju na to tijelo, potrebno nam je određeno znanje fizike i matematike.

Najprije se odredi fizikalni model koji će odrediti kako će se simulirati meki objekt. Zatim se odaberu strukture podataka koje će na zadovoljavajući način opisivati fizikalni model i oblik tijela nad kojim će se vršiti simulacija. Sam model mekog objekta nastaje povezivanjem strukture podataka koja opisuje oblik tijela sa strukturom podataka koja opisuje fizikalni model. Nad njime se zatim vrše izračuni jednadžbi koje vrijede za taj model. Time se simulira ponašanje mekog objekta.

## 2.1. Elastičnost

O tijelima s elastičnim svojstvima je potrebno znati Hookeov zakon koji govori da je deformacija tijela proporcionalna primijenjenoj sili. Kada utjecaj sile nestane tijelo se vraća u svoj prvobitni oblik. Zakon je otkrio engleski fizičar Robert Hooke 1676. godine. Hookeov zakon je opisan formulom:

$$F = -k * \Delta x$$

gdje je:

$F$ - elastična sila opruge

$k$ - konstanta opruge

$\Delta x$ - pomak u smjeru suprotnom od djelovanja elastične sile

Hookeov zakon se koristi samo za aproksimacije elastičnih pojava a ne i za savršen opis jer realne elastične pojave su mnogo složenije za opisati.

## 2.2. Gravitacija

Od sila koje djeluju na elastično deformabilno tijelo najjednostavnija je gravitacijska sila:

$$F_g = m * g$$

gdje je:

$F_g$ - sila gravitacije

$m$ - masa čestice

$g$ - gravitacijska konstanta

## 2.3. Drugi Newtonov zakon

Ostale sile ovise o odabiru fizikalnog modela.

Utjecaj sila se računa preko 2. Newtonovog zakona:

$$\vec{F_i} = m_i * \frac{\partial^2 \vec{r_i}}{\partial t^2}$$

koji se na kraju numerički integrira. Ali prije toga se ta diferencijalna jednadžba drugog reda svodi na diferencijalnu jednadžbu prvog reda radi lakše implementacije i primjene Eulerove metode integracije:

$$\frac{\overrightarrow{F_i}}{m_i} = \frac{\partial^2 \vec{r_i}}{\partial t^2}$$

Preko:

$$\overrightarrow{v_i} = \frac{\partial \vec{r_i}}{\partial t}$$

Dobiva se:

$$\frac{\overrightarrow{F_i}}{m_i} = \frac{\partial \overrightarrow{v_i}}{\partial t}$$

## 2.4. Eulerova metoda integracije

Za potrebe izračuna diferencijalnih jednadžbi koristit će se Eulerova metoda integracije. Metoda se koristi da bi se parcijalne diferencijalne jednadžbe pretvorile u skup algebarskih jednadžbi koje se zatim rješavaju numerički. Time se izbjegava integriranje koje je računski vrlo zahtjevno i mnogo složenije od numeričkog rješavanja algebarskih jednadžbi. Osim brzine, nusprodukt metode je i manja preciznost pa se računanje položaja čestica kroz vrijeme svodi na aproksimacije. Činjenica da aproksimira i zaokružuje rezultat joj je ujedno i glavni nedostatak zbog čega može doći do pogrešaka prilikom simulacije. Eulerova metoda je jednostavnija i brža od ostalih numeričkih metoda.

Da bi se simulirale deformacije elastičnih tijela, potrebne su vrijednosti prostornih i vremenskih koordinata čestica tijela.

Brzina čestice se računa kao:

$$\dot{v} = a$$

$$v_{n+1} = v_n + a_n * \Delta t$$

Akceleracija je poznata preko sila koje utječu na česticu i preko mase same čestice.

Zatim se položaj čestice računa kao:

$$\dot{x} = v$$

$$x_{n+1} = x_n + v_{n+1} * \Delta t$$

Preko navedenih postupaka se saznaje novi položaj svake čestice pod utjecajem sile.

$\Delta t$ je vrijeme proteklo između dvije iteracije računanja djelovanja sila na česticu.

To vrijeme može biti veće ili manje a odabirom njezine vrijednosti izravno se utječe na kriterije točnosti i stabilnosti.

Nestabilnost simulacije se javlja npr. kad je $\Delta t$ prevelik pa se sile a samim time i položaji čestica neprecizno izračunaju. To se može ispraviti na način da se ograniči maksimalna sila koja može djelovati na česticu ili pak maksimalna brzina koju čestica može imati. U ovome radu je kasnije kod programske implementacije korišteno ograničavanje maksimalnog tlaka koji djeluje na pojedine dijelove modela.

Još jedan način stabilizacije modela je uvođenje sila prigušenja koje ispravljaju greške Eulerove metode integracije.

Opisana metoda numeričke integracije nije jedina. Postoje još i Runge-Kuttine metode i Verletova metoda. Navedene metode se razlikuju po složenosti, preciznosti, brzini, primjeni i drugim svojstvima.

# 3. Metode za izradu elastično deformabilnih objekata

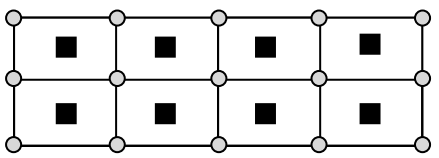
Postoji više metoda za izradu elastično deformabilnih objekata a najpoznatije su metoda konačnih elemenata i sustav masa i opruga. Sustav masa i opruga je jednostavnija i brža metoda ali daje slabije rezultate dok je metoda konačnih elemenata komplicirana i spora ali daje jako dobre rezultate simulacije. Uz opis navedene dvije metode, opisat će se i programski implementirati modificirana inačica sustava masa i opruga - tlačni model u simulaciji mekih tijela.

Od ostalih metoda za simulaciju mekih tijela valja još spomenuti metodu konačnih volumena, metodu konačnih razlika, metodu konačnih granica i metode koje pamte originalni oblik tijela te nastoje nakon svake deformacije vratiti oblik tijela u početno stanje.

## 3.1. Metoda konačnih elemenata

Konačni elementi su konačni broj podskupova dobiveni diskretizacijom kontinuirane površine tijela kojeg se želi deformirati. Diskretizacijom površine se zapravo dobije mreža čvorova, a to su točke u kojima su elementi međusobno povezani. Na mrežu se zatim primjenjuju parcijalne diferencijalne jednadžbe koje djeluju na konačne elemente preko čvorova.

Konačni produkt izrade diskretiziranog modela je globalni sustav jednadžbi koje kao nepoznanice imaju vrijednosti čvorova konačnih elemenata. Taj globalni sustav jednadžbi se može zapisati u jednu veliku matricu što pogoduje primjeni u području računalne grafike jer se operacije nad matricama mogu paralelizirati i time ubrzati izvođenje simulacije na računalu.




**Slika 1. Apstrakcija konačnih elemenata**

## 3.2. Sustav masa i opruga

Sustav masa i opruga je najjednostavniji i najintuitivniji od svih modela izrade deformabilnih objekata. Lako se implementira i računski je vrlo efikasan. Kao što ime govori, objekti su opisani česticama koje su s okolnim česticama povezane oprugama. Svaka čestica ima masu, položaj u koordinatnoj mreži, svoju brzinu i akceleraciju. Na svaku česticu djeluje gravitacijska sila i sile opruga kojima je povezana. Čestice zajedno s oprugama čine unutrašnji volumen tijela. Model mekog objekta u ovom slučaju ovisi o čvrstoći i stabilnosti sustava čestica i opruga, što pak ovisi o razmještaju i razlučivosti mreže čestica i opruga.

Sustav masa i opruga se iz temelja opisuje diskretnim modelom. Time se u začetku razlikuje od metode konačnih elemenata kod koje se na objekt gleda kao na beskonačni skup točaka koji se zatim diskretizira.

Sila opruge koja djeluje nad česticom se određuje pomoću svojstava opruga koje su spojene s česticom i položajem susjednih čestica. Opruge se obično modeliraju tako da su linearno elastične, a sila opruge koja djeluje na dvije čestice izražava se kao:

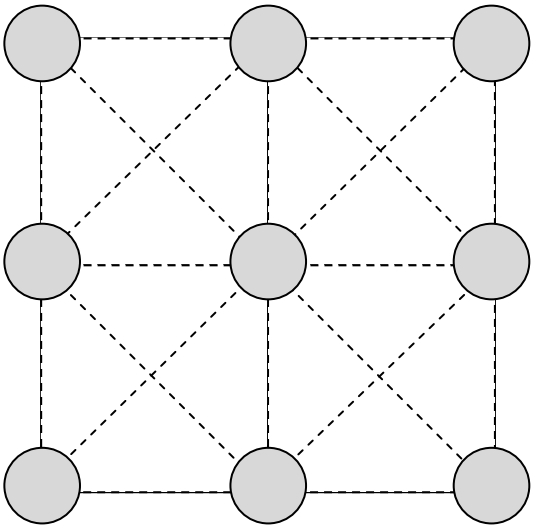
$$\vec{F_i} = k_s * \left(\left|\overrightarrow{x_{ij}}\right| - l_{ij}\right) * \frac{\overrightarrow{x_{ij}}}{\left|\overrightarrow{x_{ij}}\right|}$$

gdje je:

$\left|\overrightarrow{x_{ij}}\right|$ - razlika između vektora položaja čestica *i* i *j*

$l_{ij}$ - duljina opruge u položaju mirovanja

$k_s$ - konstanta opruge



**Slika 2. Apstrakcija čestica spojenih oprugama**

## 3.3. Tlačni model u simulaciji mekih tijela

Za izradu elastično deformabilnih objekata za ovaj rad je korištena modificirana inačica sustava masa i opruga. Ona se razlikuje od standardnog modela po tome što je u izračun sila dodana pritisna sila idelanog plina čiji utjecaj se koristi na način kao da ispunjava volumen tijela kojeg se elastično deformira. Taj pristup daje pritisnu silu koja se dodatno uvažava kod izračuna utjecala sila na čestice objekta. Glavna prednost modela je što je sustav masa i opruga pojednostavljen do te mjere da su potrebni samo vrhovi i bridovi koji čine površinu objekta a ne i njegovu unutrašnjost.

Cijeli model opisuju dvije glavne jednadžbe koje se koriste uz postojeći model sustava masa i opruga. Za sustav masa i opruga se zna da se pomoću 2. Newtonovog zakona računaju sile koje djeluju na objekt. Osim gravitacijske sile i sila opruga, potrebno je još izračunati pritisnu silu. Tlak je sila koja djeluje na neku površinu i u smjeru vektora normale te površine. U 2D slučaju je to sila koja djeluje na brid u smjeru vektora normale brida.

Pritisna sila se računa kao:

$$\overrightarrow{F_p} = \vec{P} * A$$

gdje je:

*A* - površina

$\vec{P}$ - tlak

Tlak se računa kao:

$$\vec{P} = P * \hat{n}$$

gdje je:

*P* - vrijednost tlaka

$\hat{n}$ - vektor normale na površinu na koju tlak djeluje

Treba izračunati vrijednost tlaka u tijelu. Implementacija se bazira na zakonima termodinamike. Točnije, na termodinamičkim svojstvima idealnog plina. Jednadžba stanja klasičnog termodinamičkog idealnog plina glasi:

$$P * V = n * R * T$$

gdje je:

*P* - tlak

*V* - volumen tijela

*n* - broj čestica

*R* - plinska konstanta

*T* - temperatura plina

Iz te jednadžbe slijedi da se tlak idealnog plina u tijelu računa kao:

$$P = \frac{n * R * T}{V}$$

Za jednostavniju implementaciju uzima se da je *T* konstanta. Iz toga slijedi da pri simulaciji treba izračunati još samo volumen, tj. u 2D slučaju površinu elastičnog tijela.

Površina tijela se računa preko Gaussovog teorema.

$$V = \sum_{i}^{broj\ opruga} \frac{1}{2} * |x_1 - x_2| * n_x * dl$$

gdje je:

*V* - volumen tijela

$|x_1 - x_2|$ - apsolutna vrijednost udaljenosti x-komponenata čestica zajedničke opruge

$n_x$ - vektor normale x komponente

$dl$ - duljina opruge

Kada se znaju sve vrijednosti, svakoj čestici se akumuliraju utjecaji sila na nju i pomakne se prema izračunima Eulerove metode integracije.

# 4. Programska implementacija

Simulacija tlačnim modelom je za ovaj rad implementirana pomoću programskog jezika C++ i grafičkog programskog sučelja OpenGL. Korišteni algoritam vrijedi i za sustav masa i opruga, s razlikom da ovdje korišteni ima dodanu 3. silu - pritisnu silu.

ALGORITAM:

1) Za sve vrhove:
    1. Izračunaj i akumuliraj utjecaj gravitacijske sile i sile opruge
2) Izračunaj volumen tijela
3) Za sve bridove:
    1. Izračunaj pritisnu silu na brid
    2. Izračunaj vrijednost tlaka
    3. Pomnoži s normalom brida
    4. Akumuliraj utjecaj pritisne sile na vrh
4) Integriraj jednadžbu kretanja
5) Pomakni vrhove

## 4.1. Strukture podataka i izvorni kod

Osnovu simulacije čini model sastavljen od čestica i opruga. U nastavku su opisane klase koje opisuju model. Čestice mase i opruge su objekti koji se spremaju u klase vektora iz biblioteke <vector>.

```
class Point2d
{
public:
	double x, y;
	double vx, vy;
	double fx, fy;

	...
};
```

Svaka čestica pamti svoju poziciju u koordinatnoj mreži scene te brzinu i ukupnu silu koja djeluje na nju u smjeru *x* i *y* osi.

```
class Spring
{
public:
	int i, j;
	float length;
	float nx, ny;

	...
};
```

Opruga sprema indekse čestica koje povezuje. Indeksi *i* i *j* pokazuju na čestice preko vektorskog polja *points*. Opruga također pamti svoju početnu duljinu i vektor smjera normale. Početna duljina se koristi pri izračunu sile opruge.

Sama animacija se izvodi na način da OpenGL funkcija glutTimerFunc iterativno poziva funkcije koje djeluju nad česticama i oprugama. Najprije se izračuna utjecaj svih sila na model a zatim se dijelovi modela pomiču prema izračunima Eulerove metode integracije. U nastavku su opisani izvorni kodovi za izračun sila i za Eulerovu integraciju.

```
const double FINAL_PRESSURE = 100.00;
float Pressure = 0.0;

...

computeForces();
eulerIntegrate();

if(Pressure < FINAL_PRESSURE)
{
	Pressure += FINAL_PRESSURE / 100.0f;
}
```

Isječak koda uvjeta (Pressure < FINAL_PRESSURE) povećava tlak unutar tijela koji može pasti kao posljedica deformacija. Time se osigura nastojanje tijela da se vrati u početni oblik.

Sljedeći isječak koda računa gravitacijsku silu i silu kojom korisnik djeluje klikom miša. Sila koja se stvara mišem djeluje u vektoru smjera od lokacije pokazivača miša do najbližeg vrha modela.

```
const double MASS = 1.0;
const double GY = 10.0;

...

for(i = 0; i < points.size(); i++)
{
	points[i].fx = 0.0;
	points[i].fy = MASS * GY * (Pressure - FINAL_PRESSURE >= 0);

	if(i == closestPointIndex)
	{
		if(mouseDown)
		{
			x1 = points[i].x;
			y1 = points[i].y;
			x2 = mouseX;
			y2 = mouseY;

			r12d = sqrt((x1 - x2) * (x1 - x2) +
				(y1 - y2) * (y1 - y2));

			f = r12d +
			(points[i].vx * (x1 - x2) +
			points[i].y * (y1 - y2)) / r12d;

			Fx = ((x1 - x2) / r12d) * f;
			Fy = ((y1 - y2) / r12d) * f;

			points[i].fx -= Fx;
			points[i].fy -= Fy;
		}
	}
}
```

Konstante MASS i GY su odabrane proizvoljno. Uvjet (Pressure - FINAL_PRESSURE >= 0) osigurava da sila gravitacije ne djeluje na čestice na koje tlak djeluje jače od zadane gornje granice tlaka.

Utjecaj sile opruge se računa po ranije opisanim formulama. Na kraju se za svaku oprugu, koja je ujedno i brid, izračunaju normale koja se kasnije koriste za izračun volumena tijela i za izračun pritisne sile unutarnjeg tlaka tijela.

```
const double KS = 1600.0;
const double KD = 40.0;

...

for(i = 0; i < springs.size(); i++)
{
	x1 = points[springs[i].i].x;
	x2 = points[springs[i].j].x;
	y1 = points[springs[i].i].y;
	y2 = points[springs[i].j].y;

	r12d = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));

	if(r12d != 0)
	{
		vx12 = points[springs[i].i].vx - points[springs[i].j].vx;
		vy12 = points[springs[i].i].vy - points[springs[i].j].vy;

		f = (r12d - springs[i].length) * KS +
		(vx12 * (x1 - x2) + vy12 * (y1 - y2)) * KD / r12d;

		Fx = ((x1 - x2) / r12d) * f;
		Fy = ((y1 - y2) / r12d) * f;

		points[springs[i].i].fx -= Fx;
		points[springs[i].i].fy -= Fy;

		points[springs[i].j].fx += Fx;
		points[springs[i].j].fy += Fy;
	}
	springs[i].ny = -(x1 - x2) / r12d;
	springs[i].nx = (y1 - y2) / r12d;
}
```

Konstante KS i KD su odabrane proizvoljno.

Dio koda zadužen za izračun pritisne sile najprije izračuna volumen tijela pomoću kojeg se za svaku česticu računa najprije tlak a potom i sila kojom taj tlak djeluje na česticu.

```
float Pressure = 0.0;

...

for(i = 0; i < points.size(); i++)
{
	x1 = points[springs[i].i].x;
	x2 = points[springs[i].j].x;
	y1 = points[springs[i].i].y;
	y2 = points[springs[i].j].y;

	r12d = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));

	volume += 0.5 * fabs(x1 - x2) * fabs(springs[i].nx) * (r12d);
}

for(i = 0; i < points.size(); i++)
{
	x1 = points[springs[i].i].x;
	x2 = points[springs[i].j].x;
	y1 = points[springs[i].i].y;
	y2 = points[springs[i].j].y;

	r12d = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));

	pressurev = r12d * Pressure * (1.0f / volume) * 1000;

	points[springs[i].i].fx += springs[i].nx * pressurev;
	points[springs[i].i].fy += springs[i].ny * pressurev;
	points[springs[i].j].fx += springs[i].nx * pressurev;
	points[springs[i].j].fy += springs[i].ny * pressurev;
}
```

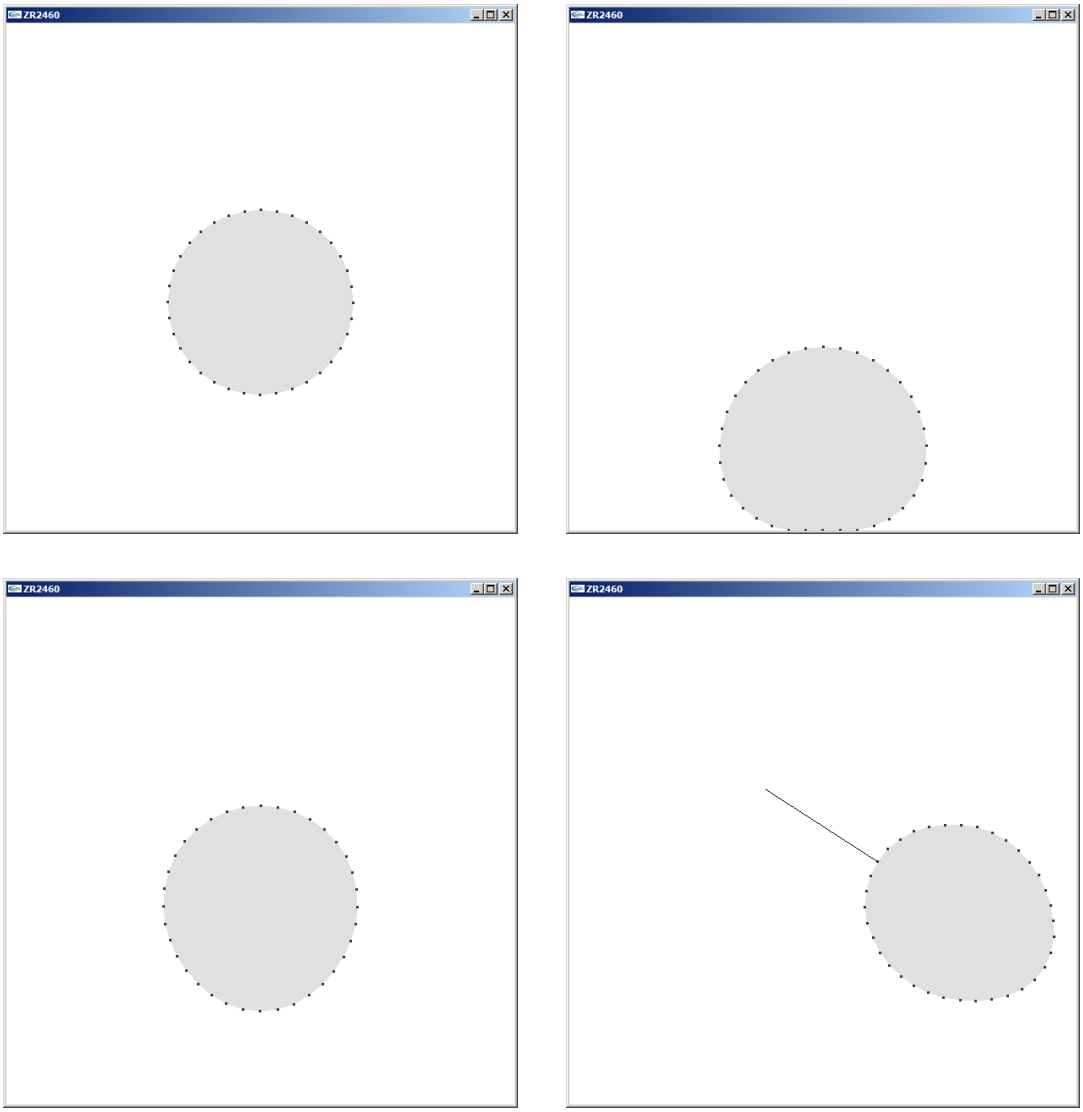
Eulerov integrator računa pomak čestica prema ranije opisanim algoritmima i formulama.

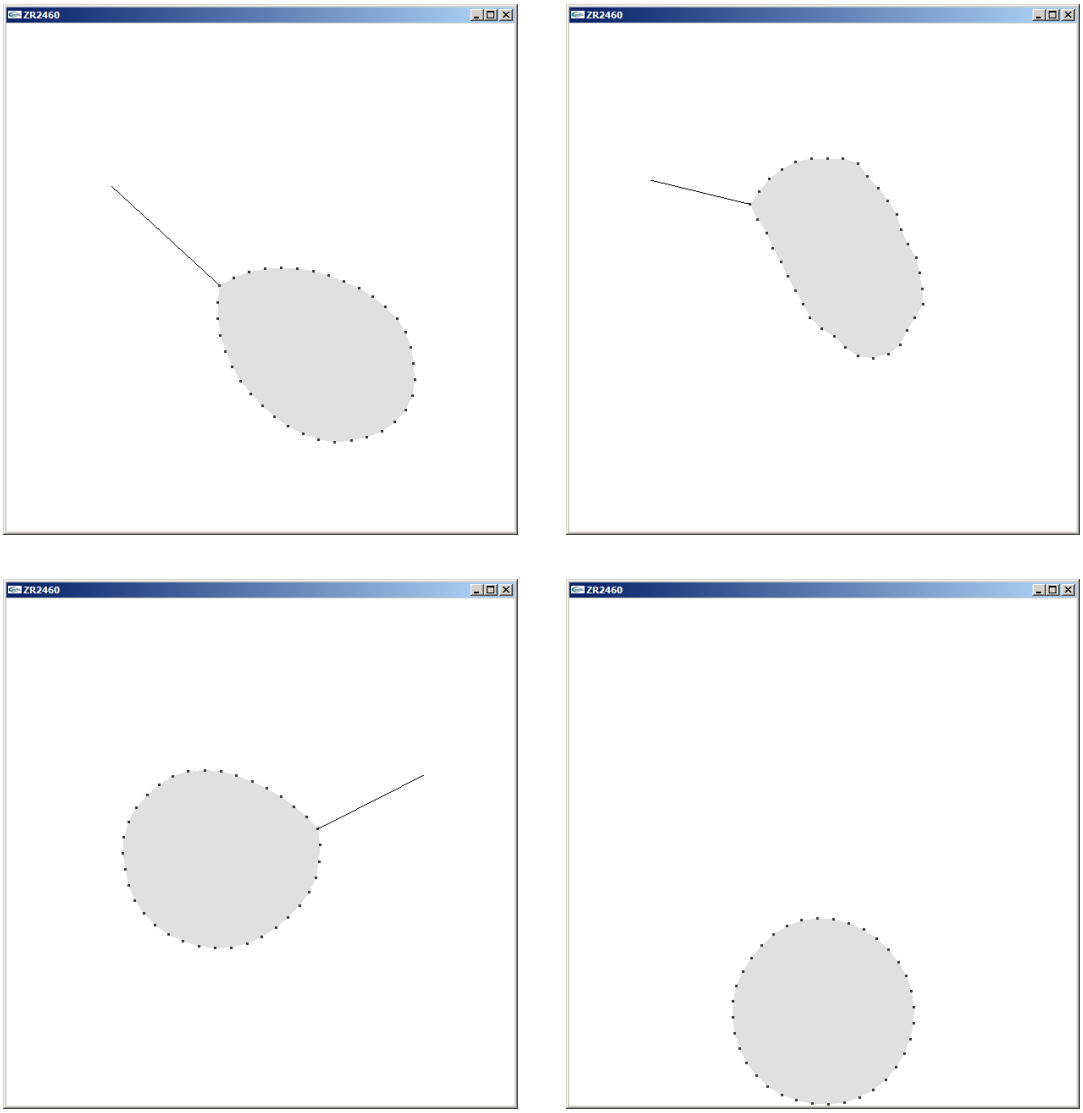
```
const double DT = 0.001;
const double MASS = 1.0;

...

void eulerIntegrate()
{
	int i;
	float dry, drx;

	for(i = 0; i < points.size(); i++)
	{
		points[i].vx += (points[i].fx / MASS) * DT;
		drx = points[i].vx * DT;

		...

		points[i].x += drx;


		points[i].vy += (points[i].fy) * DT;
		dry = points[i].vy * DT;

		...

		points[i].y += dry;
	}
}
```

Vrijednost DT je odabrana proizvoljno.

## 4.2. Primjer simulacije

Sljedeće slike prikazuju deformaciju elastičnog tijela kroz vrijeme. Najprije gravitacijska sila povuče tijelo do dna od kojeg se tijelo zatim odbije. Nakon toga se djeluje na tijelo silama proizvedenim mišem od kojih se tijelo značajno deformira. Posljednja slika prikazuje oblik tijela što se nakon nekog vremena vratilo u početni oblik.






**Slika 3. Simulacija deformacije elastičnog tijela temeljena na tlačnom modelu**

Model ima 36 vrhova.

Ostale konstante imaju sljedeće vrijednosti:

FINAL_PRESSURE = 100.00

KS = 1600.0

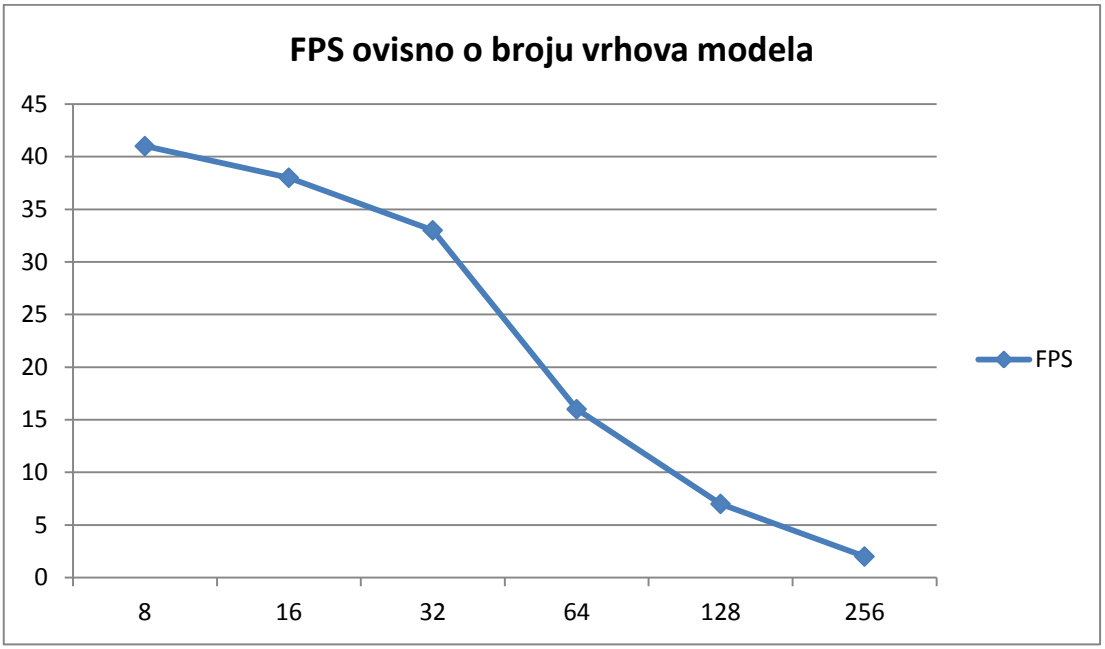
KD = 40.0

BALL_RADIUS = 100

## 4.3. Utjecaj različitih parametara na ostvarene rezultate

Glavni pokazatelj brzine simulacije u realnom vremenu je broj prikazanih sličica u sekundi (eng. FPS - *frames per second)*. Cilj svake aplikacije koja generira računalne animacije u realnom vremenu je održavati taj broj čim većim. Kao donja granica fluidnosti animacije se uzima 25-30 FPS-a.

Simulacija je nekoliko puta pokrenuta s različitim parametrima modela i algoritma. Podešavanjem parametara ustanovljeno je da samo broj vrhova modela ima značajan utjecaj na njezine performanse. Ostali parametri poput konstante opruge, konačnog tlaka i mase čestica ili nemaju ili imaju zanemariv utjecaj na broj prikaza sličica u sekundi.




**Slika 4. Utjecaj broja vrhova modela na FPS**

Iz dobivenih rezultata se vidi da je simulacija izrazito neoptimizirana. Četrdesetak sličica po sekundi za samo 8 vrhova su neprihvatljive performanse. Iz grafa je vidljivo da već sa 64 vrhova u sceni simulacija počinje „trzati“.

Kada bi se na temelju izvornog koda napravio grafički pokretač (eng. *engine*) i pomoću njega neka video igra, igra bi već sa pedesetak vrhova u sceni bila neigriva. Bez dvojbe slijedi zaključak da ima mjesta poboljšanjima.

Testna konfiguracija je bila**:**

CPU: AMD Turion 64 X2 Mobile TL-58, Tyler 65nm Technology

RAM: 2,00 GB Dual-Channel DDR2 @ 315MHz (5-5-5-15)

GPU: 256MB ATI Mobility Radeon HD 2400 XT

Za mjerenje prikaza sličica u sekundi korištena je besplatna verzija programa *Fraps*.

# 5. Zaključak

Kroz rad je učinjen kratki presjek kroz područje računalne grafike i animacije koje se bavi simulacijom mekih tijela. Rad iznova ukazuje na korisnost i primjenu dostignuća sa područja matematike i fizike.

Tlačni model u simulaciji mekih tijela se ističe kao pokazatelj inovacija i napretka u polju računalne grafike i animacije. Također, njegovom simulacijom se daje do znanja važnost optimizacije programskih rješenja.

Iz svega se zaključuje da je računalna grafika neprestano u razvoju i da budućnost nosi načine za stvaranje još realističnije i impresivnije virtualne stvarnosti.

# 6. Literatura

[1] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M., „Physically Based Deformable Models in Computer Graphics“, EuroGraphics 2005, Dublin, Republika Irska, 2005.

[2] Maciej Matyka, Mark Ollila, „Pressure Model of Soft Body Simulation“, University of Wrocław, Poljska, Linköping University, Švedska, 2003.

[3] Maciej Matyka, „How To Implement a Pressure Soft Body Model“, 2004.

[4] Witkin, A. and Baraff, D., "An Introduction to Physically Based Modeling", SIGGRAPH Course Notes, 1993.

[5] Terzopoulos D., Platt J., Barr A., Fleischer K.: "Elastically deformable models", SIGGRAPH ’87, 1987.

[6] Wikipedia, Elasticity (physics), http://en.wikipedia.org/wiki/Elasticity_(physics), 06. svibnja 2012.

[7] Wikipedia, Soft body dynamics, http://en.wikipedia.org/wiki/Soft_body_dynamics, 08. svibnja 2012.

[8] Wikipedia, Finite element method, http://en.wikipedia.org/wiki/Finite_element_method, 08. svibnja 2012.

[9] Wikipedia, Finite element method in structural mechanics, http://en.wikipedia.org/wiki/Finite_element_method_in_structural_mechanics, 08. svibnja 2012.

# Sažetak


## POSTUPCI IZRADE ELASTIČNO DEFORMABILNIH OBJEKATA

Ovaj rad ukratko opisuje neke od postupaka izrade elastično deformabilnih objekata. Konkretno, to su metoda konačnih elemenata i sustav masa i opruga. Njihovom usporedbom pojašnjava njihove prednosti i nedostatke. Predstavljena je i modificirana inačica sustava masa i opruga - tlačni model u simulaciji mekih tijela.

Rad objašnjava potrebne fizikalne osnove za ostvarenje modela sposobnih za simulaciju mekih tijela potpuno elastičnih svojstava. Također navodi neke matematičke alate potrebne za izračunavanje jednadžbi koje se koriste u svrhu simulacije.

Od opisanih metoda programski je implementirana metoda temeljena na tlačnom modelu. Korišten je programski jezik C++ i grafičko programsko sučelje OpenGL. Sam algoritam je objašnjen uz pomoć izvornog koda programa.

Model je odsimuliran i njegove performanse su analizirane.



**Ključne riječi:** meko tijelo**,** elastično deformabilni objekti, tlačni model, C++, OpenGL

## Abstract

### ELASTICALLY DEFORMABLE BODY MODELING METHODS

This paper briefly describes some of the procedures for creating elastically deformable objects. Specifically, the finite element method and the mass-spring system. Their advantages and disadvantages are explained by their comparison. A modified version of the mass-spring system is presented - the pressure model for soft body simulation.

The paper explains the necessary knowledge of physics needed for the construction of models capable of simulating elastic soft body mechanics. It also specifies some mathematical tools needed to calculate the equations used for simulation.

The pressure model was chosen to be implemented in C++ and OpenGL. The algorithm itself is explained with the help of source code.

The model was simulated and its performance analyzed.



**Keywords:** soft body**,** elastically deformable body, pressure model, C++, OpenGL