Simulacija tkanine

Jansky, Marin

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Simulacija tkanine

Rijeka, rujan 2024.

Marin Jansky
0069082562

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Simulacija tkanine

Mentor: prof. dr. sc. Jerko Škifić

Rijeka, rujan 2024.

Marin Jansky
0069082562

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 14. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Platformski nezavisno programiranje**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Marin Jansky (0069082562)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Programsko inženjerstvo

Zadatak: **Simulacija tkanine / Cloth simulation**

Opis zadatka:

Izraditi računalni program dinamičkog procesa savijanja tkanine pod utjecajem vanjskih sila modelom opruga i čestica. Voditi računa o geometrijskim i fizikalnim veličinama koje definiraju osobine simuliranog fleksibilnog objekta. Analizirati utjecaj diskretizacije i fizikalnih osobina objekta na konačni rezultat simulacije. Računalni program treba biti platformski nezavisan, vodeći računa o ograničenjima odabranih platformi.
Polje znanstvenog područja: Računarstvo
Grana znanstvenog područja: Programsko inženjerstvo

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

M Jansky

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Izv. prof. dr. sc. Jerko Škifić

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2024.

_____

Marin Jansky

Zahvala

Zahvaljujem se mentoru na dostupnosti, podršci i stručnim savjetima za vrijeme pisanja ovog diplomskog rada. Nadalje, zahvaljujem se svojoj obitelji na podršci tijekom studiranja te se posebno želim zahvaliti svojoj djevojci koja mi je bila podrška i velika motivacija pri pisanju ovog rada.

Sadržaj

Popis slika

Poglavlje 1

Uvod

Simulacija tkanine pripada području računalne grafike pod nazivom dinamika mekih tijela (eng. soft-body dynamics) koje se temelji na vizualno uvjerljivoj fizičkoj simulaciji gibanja tijela koja imaju sposobnost deformacije. Sama činjenica da se tijelo može deformirati pod utjecajem sila uvodi velik broj problema koji ne postoje kod dinamike krutih tijela što smanjuje računalnu izvedivost i zahtijeva znatno veću količinu računalnih resursa za zadovoljivu izvedbu u realnom vremenu.

Prilikom modeliranja tkanine postoje 2 glavna modela: model temeljen na sili i model temeljen na poziciji. Model temeljen na sili koristi sustav čestica određene mase povezan oprugama i po potrebi prigušivačima (eng. dampers) koji su paralelno s oprugama spojeni s česticama. Model temeljen na poziciji sastoji se od čestica s ograničenjima; umjesto opruga se postavljaju ograničenja udaljenosti između povezanih čestica koje se tijekom postupka rješavanja pomiču kako bi zadovoljila zadana ograničenja. Model masa i čestica ima temelj u fizici pri čemu se koristi drugi Newtonov zakon za definiranje jednadžbi gibanja što za implementaciju zahtijeva rješavanje sustava običnih diferencijalnih jednadžbi. Model temeljen na poziciji je pojednostavljenje sustava koji se ne temelji na zakonima fizike te sam po sebi neće biti fizički točan, ali budući da je priča o računalnoj grafici taj aspekt nije toliko bitan koliko je bitno da gibanje tkanine izgleda vizualno zadovoljavajuće te bude relativno brzo za izračunati za primjene u stvarnom vremenu.

Unatoč određenim poželjnim kvalitetama pozicijskog modela u usporedbi s mo-

delom masa i opruga, kroz ovaj rad bit će objašnjena implementacija interaktivnog simulatora tkanine temeljenog na modelu masa i opruga uz analizu rezultata implementacije. Navedena implementacija realizirana je s pomoću programskog jezika C++ (po standardu C++20) te korištenjem OpenGL-a kao sučelja za programiranje grafičkih aplikacija. Sve dodatne knjižnice i njihove uloge bit će navedene i opisane u sljedećem poglavlju.

Poglavlje 2

Korištene knjižnice i pokretanje aplikacije

Projekt u ovom radu koristi CMake kao platformski nezavisno rješenje za stvaranje potrebnih datoteka za kompilaciju C i C++ projekata. Prije kompilacije potrebno je pokrenuti CMake koji će stvoriti sve potrebne datoteke za kompilaciju projekta i svih potrebnih knjižnica.

Postupak za kompilaciju na *nix operacijskim sustavima (Linux, FreeBSD, itd.):

1. otvorite terminal
2. navigirajte do direktorija koji sadrži *CMakeLists.txt* datoteku i direktorij *deps*
3. utipkajte sljedeće naredbe

```
$ mkdir build
$ cd build
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make -j 4
$ cp -r ../ClothSimulator/models ../ClothSimulator/
    scenes ../ClothSimulator/shaders ClothSimulator
$ cd ClothSimulator
```

Unutar trenutnog direktorija moguće je sada pokrenuti izvršnu datoteku *ClothSim*.

2.1 OpenGL

OpenGL je platformski nezavisno sučelje za programiranje aplikacija (eng. application programming interface, API). Sučelje koje pruža služi za komunikaciju s grafičkim procesorom (eng. graphics processing unit, GPU) najčešće za iscrtavanje 2D i 3D vektorske grafike, ali novije verzije OpenGL standarda omogućuju i korištenje grafičkog procesora za ne-grafičke namjene kao što su strojno učenje, kopanje kriptovaluta i razne simulacije.

U ovom radu konkretno je u primjeni OpenGL verzija 4.5 koja nije najnovija, ali zadovoljava sve potrebe i kompatibilna je s nešto starijim grafičkim hardverom od najnovije verzije.

2.2 GLEW

GLEW (OpenGL Extension Wrangler) je platformski nezavisna knjižnica otvorenog koda koja učitava i omogućava korištenje proširenja za OpenGL. Budući da OpenGL nije klasična knjižnica funkcija, već samo sučelje, dostupnost određenih funkcionalnosti i proširenja može varirati ovisno o platformi, korištenom hardveru i programskoj podršci.

OpenGL sam po sebi pruža pristup samo onim funkcionalnostima za koje je garantirano da su podržane na određenoj platformi. Sve ostale funkcionalnosti se smatraju neobaveznim dodacima (proširenjima) i ne moraju nužno biti podržani na ciljanoj platformi. Moguće je pri kompiliranju koda povezati i proširenja, ali ako ciljana platforma ne podržava korišteno proširenje program se neće pokrenuti. GLEW u tom slučaju pomaže u prenosivosti na druge platforme i omogućuje implementaciju alternativnih rješenja u slučaju da željeno proširenje nije dostupno bez rušenja programa.

2.3 GLFW

GLFW knjižnica namijenjena je za upotrebu zajedno sa OpenGL, OpenGL ES ili Vulkan sučeljima. Otvorenog je koda i platformski nezavisna te pruža funkcije za stvaranje i upravljanje prozorima, grafičkim kontekstima, primanjem događaja od operacijskog sustava i ulaznih podataka od npr. tipkovnice i miša. Apstrahira specifičnosti operacijskog sustava kroz jednostavno sučelje te na taj način postiže platformsku nezavisnost.

2.4 GLM

GLM pruža velik broj matematičkih funkcija namijenjenih za grafičke aplikacije u čijoj domeni prevladava linearna algebra. Funkcije su dizajnirane prema OpenGL Shading Language (GLSL) specifikaciji te se velik broj funkcija iz GLSL-a može koristiti i u C++ izvornom kodu i obrnuto zahvaljujući ovoj knjižnici. U ovom radu GLM knjižnica koristi se gotovo isključivo unutar domene grafike te su najkorištenije funkcije za računanje transformacijskih matrica i jednostavne operacije nad vektorima. Za samu simulaciju koristi se nešto naprednija knjižnica, više o njoj kasnije.

2.5 Dear ImGui

Dear ImGui omogućava iznimno jednostavno stvaranje grafičkih korisničkih sučelja (eng. graphical user interface, GUI). Otvorenog je koda i podržava sva popularna sučelja za programiranje grafičkih aplikacija uključujući OpenGL, Vulkan, DirectX i druge. Isto tako podržava i popularne multimedijske platforme kao što su GLFW, SDL, Win32, Android i druge. Dear ImGui vrlo je bitan dio ovog rada jer je omogućio mnogo veću brzinu iteracije pri razvoju programskog rješenja te promjenu simulacijskih i grafičkih parametara tijekom izvođenja programa dajući veću razinu interaktivnosti nego što bi inače bilo moguće.

2.6 Eigen

Eigen pokreće cijelu simulaciju unutar ovog rada, od računanja sila do detekcije i rješavanja sudara. Sadrži velik broj algoritama i struktura podataka potrebnih za linearnu algebru (vektori, matrice, numerički solveri, algoritmi za dekompoziciju i sl.). Knjižnica je otvorenog koda i besplatna za bilo koju primjenu. Programsko sučelje je jako intuitivno i jednostavno za koristiti te nudi ono što je bitno za ovaj rad, a to su odlične performanse.

2.7 Nlohmann JSON

Nlohmann JSON knjižnica omogućava serijalizaciju i deserijalizaciju podataka u/iz JSON formata. JSON je moderan i iznimno popularan format za serijalizaciju strukturiranih podataka u čovjeku čitljivom obliku. Koristi se često u mrežnim aplikacijama za prijenos podataka preko HTTP protokola i za konfiguraciju. U ovom radu scenariji su zapisani u JSON formatu te se ova knjižnica koristi za učitavanje istih.

2.8 Tiny Obj Loader

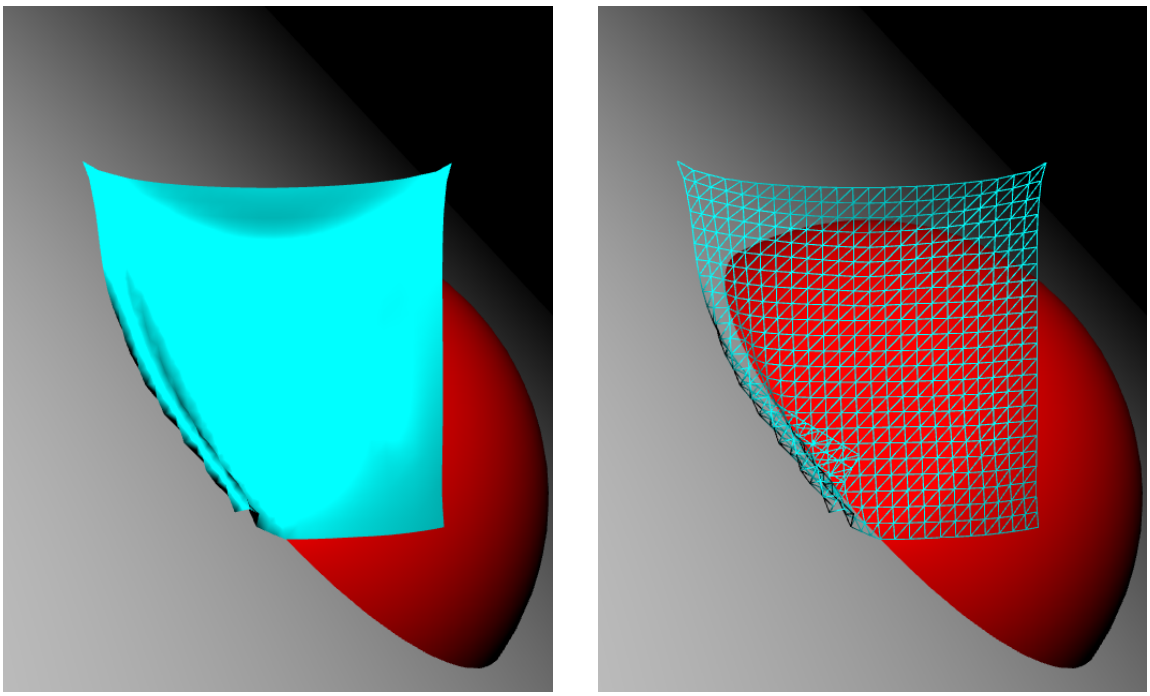
Kao što ime predlaže Tiny Obj Loader je jednostavna knjižnica čija je jedina zadaća učitavanje i parsiranje 3D podataka iz datoteka u Wavefront OBJ formatu. Kako je većina 3D modela u sklopu ovog rada proceduralno generirano, koristi se samo za učitavanje modela za lakšu orijentaciju u prostoru (3D gizmo).

Poglavlje 3

Prikaz tkanine

Na području računalne grafike geometrija je obično izražena korištenjem trokuta. Bilo koji oblik može se više ili manje vjerno prikazati diskretizacijom u trokute, pri čemu naravno veći broj trokuta omogućuje bolju aproksimaciju oblika. Pravokutnik, kvadar i ostali oblici koji ne sadrže zakrivljene plohe mogu se savršeno vjerno prikazati podjelom na trokute. Kod zakrivljenih ploha takav postupak postaje kompromis između kvalitete i količine potrebnih računalnih resursa za njihovo učitavanje i obradu, iz tog razloga razina podjele tkanine na trokute u ovom radu će biti jedan od upravljivih parametara simulacije.

Trokut se sastoji od tri vrha povezanih bridovima. U kontekstu simulacije za vrh će se koristiti termin čestica, ali oba termina predstavljaju istu stvar u kontekstu ovog rada. Na slici 3.1 prikazana je usporedba ispunjenog prikaza i prikaza samo bridova trokuta od kojih se tkanina sastoji. Neistreniranom oku bi se moglo činiti da je tkanina kontinuirana, ali na desnoj slici može se jasno vidjeti kako to nije slučaj. U ovom konkretnom slučaju tkanina se sastoji od 576 čestica (24 horizontalno i vertikalno) što nije puno, ali već izgleda lijepo. Idealan broj čestica bi bio reda veličine 10^6 ili čak i više, ali to bi zahtijevalo računalo iznimne snage, a i pri tom broju bi svaka dodatna čestica sve manje pridonosila krajnjoj kvaliteti prikaza pa je potrebno naći prihvatljiv kompromis.



(a) ispunjen prikaz tkanine

(b) prikaz tkanine kao mreža povezanih trokuta

Slika 3.1 Usporedba ispunjenog prikaza i prikaza trokuta

3.1 Čestice

Svaka čestica tkanine ima dvije veličine koje ju opisuju unutar trodimenzionalnog prostora, a to su pozicija x i brzina v, svaka izražena vektorom s tri komponente ($x, v \in \mathbb{R}^3$). Te dvije veličine zajedno zvat će se *stanje čestice*, ono će se konstantno mijenjati tijekom simulacije po pravilima modela. Čestice još imaju dodatne vektorske veličine kao što su normale i teksturne koordinate, ali te veličine nisu ključne za simulaciju nego sudjeluju u grafičkom prikazu i za razliku od pozicije i brzine mogu se odrediti na temelju stanja ili indeksa čestice.

Sve pozicije i brzine čestica pohranjene su unutar dugih stupčastih vektora X i

V gdje x_i i v_i predstavljaju poziciju i brzinu i-te čestice:

$$X = \begin{bmatrix} x_{0_x} \\ x_{0_y} \\ x_{0_z} \\ \vdots \\ x_{n-1_x} \\ x_{n-1_y} \\ x_{n-1_z} \end{bmatrix}, V = \begin{bmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \\ \vdots \\ v_{n-1_x} \\ v_{n-1_y} \\ v_{n-1_z} \end{bmatrix} \tag{3.1}$$

Nakon što su sve veličine pohranjene u obliku dugih vektora potrebno je nekako zapisati koje čestice (i njihove pripadajuće veličine) pripadaju kojem trokutu. Jedan jednostavan način (a istovremeno i onaj koji OpenGL koristi za iscrtavanje trokuta) je niz uređenih trojki indeksa čestica. Ključna dio je da trojke moraju biti uređene jer OpenGL uzima redoslijed vrhova kako bi odredio ako je ploha trokuta definiranog danim vrhovima prednja ili stražnja. Više detalja o ovome se može naći u kasnijem poglavlju.

3.2 Vremenska diskretizacija

Računalo kao i svi digitalni sustavi ima ograničenje da može obrađivati samo diskretne vrijednosti u diskretnim vremenskim koracima. Nediskretne vrijednosti bi zahtijevale beskonačnu količinu memorije za pohranu dok bi kontinuirana obrada podataka zahtijevala beskonačnu brzinu procesora. Budući da to (barem trenutno) nije fizički izvedivo potrebno je odlučiti do koje granice nam je preciznost potrebna i napraviti kompromis.

U računalnoj grafici i videu iluzija pokreta dobiva se dovoljno brzom promjenom prikazane slike. Što je dovoljno brzo? U filmskoj sceni broj koji je široko prihvaćen je 24 slike u sekundi dok tipične interaktivne aplikacije kao što su video igre tu vrijednost podižu na 30, 60 ili čak preko 120 slika u sekundi. Taj broj se zove *framerate*, a mjerna jedinica je s^{-1} ili popularnije *frames per second* (*FPS*). Prilikom mjerenja performansi algoritama i dijelova programa zbog svoje linearne prirode češće

se koriste sekunde (ili milisekunde) pri čemu bi analogna veličina framerateu bila *frame time*, odnosno trajanje slike što je zapravo recipročno framerate vrijednosti.

Pri simulaciji tkanine nekad je potrebno napraviti više simulacijskih koraka prije prikazivanja rezultata što znači da za svaku prikazanu sliku (frame) simuliramo više manjih vremenskih koraka gibanja tkanine. Više o tome u poglavlju 5.

Poglavlje 4

Crtanje tkanine

Prije početka rada na simulaciji potrebno je prvo postaviti temelje, a to je prikazivanje tkanine na ekranu kako bi se mogle vidjeti gibanje tkanine u prostoru? Prvi korak za ostvarenje tog cilja neka bude stvaranja mreže trokuta. Radi jednostavnosti, u ovom radu tkanina će biti kvadratnog oblika s pretpostavkom da je jednak broj čestica u horizontalnom i vertikalnom smjeru te da su na početku sve susjedne čestice međusobno jednako udaljene (gustoća čestica je konstantna). Definirajmo broj čestica u horizontalnom smjeru oznakom u i broj čestica u vertikalnom smjeru oznakom v. Iz prijašnje tvrdnje da su na početku simulacije sve susjedne čestice međusobno jednako udaljene može se zaključiti da $u = v$. Zamijenimo to jednom veličinom koju ćemo zvati rezolucija tkanine r tako da vrijedi $r = u = v$. Sad kad su definirane neke veličine mogu se definirati algoritmi za stvaranje vrhova, njihovih atributa i povezivanja vrhova u trokute.

4.1 Pravilna kvadratna rešetka

Stvaranje pravilne kvadratne rešetke jednostavan je problem, potrebno je još definirati početne uvjete i veličinu rešetke. Kao početni oblik uzmimo kvadrat jedinične površine sa središtem u ishodištu koji leži u XY ravnini. Iz podatka o jediničnoj površini slijedi da je duljina stranice tog kvadrata jednaka 1, odnosno $\Delta x = \Delta y = 1$. Označimo s $\vec{x_0'}$ poziciju krajnjeg gornjeg lijevog vrha i s $\vec{x'_{n-1}}$ poziciju krajnjeg donjeg

desnog vrha. Da bi središte kvadrata bilo u ishodištu očito je da $\vec{x'_0} = (\frac{-\Delta x}{2}, \frac{\Delta y}{2}, 0)$ i $\vec{x'_{n-1}} = (\frac{\Delta x}{2}, \frac{-\Delta y}{2}, 0)$. Tvrdnju je moguće provjeriti zbrajanjem vektora i dijeljenjem svake komponente s 2, to rezultira središtu kvadrata i u ovom slučaju to je nul-vektor koji predstavlja ishodište koordinatnog sustava.

Nakon određivanja 2 suprotna vrha kvadrata ostalo je još samo odrediti sve vrhove linearnom interpolacijom. Budući da se radi o dvodimenzionalnoj strukturi uvest će se dvodimenzionalni način indeksiranja čestica pri čemu $\vec{x_{i,j}} = \vec{x_{ir+j}}$ zbog kraćeg zapisa. Za pravilnu kvadratnu rešetku vrijedi sljedeće:

$$\forall 0 \leq i, j < r : \vec{x_{i,j}} = w \left(\begin{bmatrix} (1 - \frac{j}{r-1})\vec{x'_{0x}} + \frac{j}{r-1}\vec{x'_{n-1x}} \\ (1 - \frac{i}{r-1})\vec{x'_{0y}} + \frac{i}{r-1}\vec{x'_{n-1y}} \\ 0 \end{bmatrix} \right) \tag{4.1}$$

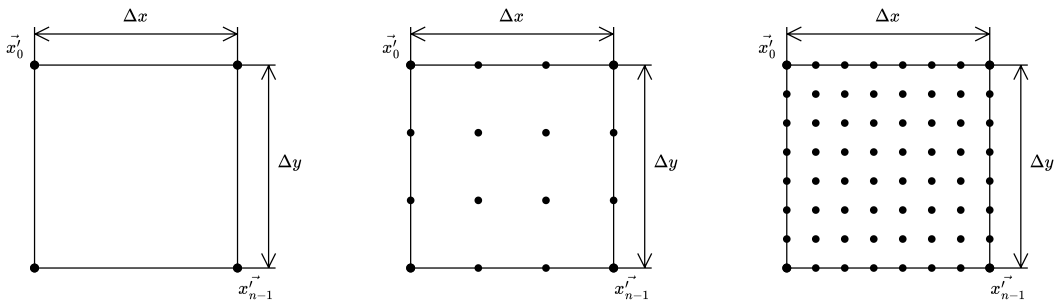
U izrazu iznad funkcija w predstavlja Euklidsku transformaciju uz neobavezno uniformno skaliranje koja omogućuje promjenu početnog stanja tkanine. Bez transformacijske funkcije tkanina bi uvijek započinjala u istoj poziciji i orijentaciji što bi otežalo analizu ponašanja tkanine u različitim uvjetima. Funkcija w bi se za ovakav tip primjene definirala kao matrično množenje transformacijske matrice i nezavisne varijable, npr:

$$w(x) = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x}_x \\ \vec{x}_y \\ \vec{x}_z \\ 1 \end{bmatrix} \tag{4.2}$$

Homogene koordinate nam omogućuju da sve transformacije, uključujući i translaciju možemo predstaviti u matričnom obliku. Homogene koordinate dobijemo dodavanjem jedne komponente vektoru i jednog reda i stupca matrici što možemo vidjeti u izrazu iznad. Pojednostavljeno, ako pomnožimo homogene koordinate skalarom različitim od nule, te koordinate će predstavljati istu točku u prostoru, matematički izraženo:

$$\begin{bmatrix} \lambda a \\ \lambda b \\ \lambda c \\ \lambda \end{bmatrix} \iff \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \lambda \neq 0 \tag{4.3}$$

Budući da transformacija koju predstavlja w nije projektivnog tipa, četvrta komponenta vektora nakon transformacije bit će $\vec{x'}_w = 1$ što znači da ju možemo samo izostaviti i uzeti prve 3 komponente bez ikakve promjene. Slika 4.1 prikazuje kako kvadratna mreža izgleda za različite vrijednosti r uz uvjet da je funkcija w transformacija identiteta, tj. $w(x) = x$.



(a) $r = 2$ (b) $r = 4$ (c) $r = 8$

Slika 4.1 Usporedba kvadratnih rešetki s različitim vrijednostima r

4.2 Normale

Normale su vektori jedinične duljine koji su okomiti na pripadajuću plohu. Potrebne su kako bi mogle izračunati veličine koje ovise o orijentaciji plohe, kao što su osvjetljenje i utjecaj sile vjetra. Kao što je već spomenuto, normale se definiraju za plohe, međutim u korištenom modelu ploha sama po sebi nije eksplicitno definirana. Trokut se može gledati kao ravna ploha, što i je, ali kako onda prikazati zakrivljenu površinu? Tkanina, a ni drugi 3D modeli ne mogu biti sastavljeni od prevelikog broja trokuta jer će postati prezahtjevni za aplikacije koje rade u realnom vremenu. Rješenje leži u tome da se normale, umjesto svakom trokutu, odrede svakom vrhu. Bez poznavanja modernog grafičkog hardvera to nema smisla jer su potrebne barem tri točke kako bi se definirala ploha. Moderni grafički procesori ovdje uskaču kako bi riješili problem zakrivljenih ploha tako što tijekom procesa rasterizacije (pretvaranje

u piksele) trokuta linearno interpoliraju atribute vrhova koji pripadaju tom trokutu što uključuje i normale ako su definirane.

Računanje normale vrhova jednog trokuta nije kompliciranо. Ako su zadane pozicije vrhova trokuta $\vec{v_0}$, $\vec{v_1}$ i $\vec{v_2}$ korištenjem svojstva vektorskog produkta dobije se sljedeće:

$$\vec{n'} = (\vec{v_1} - \vec{v_0}) \times (\vec{v_2} - \vec{v_0}) \tag{4.4}$$

Iznad je spomenuto da su normale vektori jedinične duljine, međutim vektor $\vec{n'}$ skoro nikad neće biti jedinične duljine te da bi rezultat bio isprava potrebno ga je potrebno normalizirati:

$$\vec{n} = \frac{\vec{n'}}{\|\vec{n'}\|} = \frac{\vec{n'}}{\sqrt{\vec{n'} \cdot \vec{n'}}} = \hat{n} \tag{4.5}$$

Od sada svi jedinični vektori bit će označeni kapicom kao i vektor $\hat{n}$ s desne strane prethodne jednadžbe.

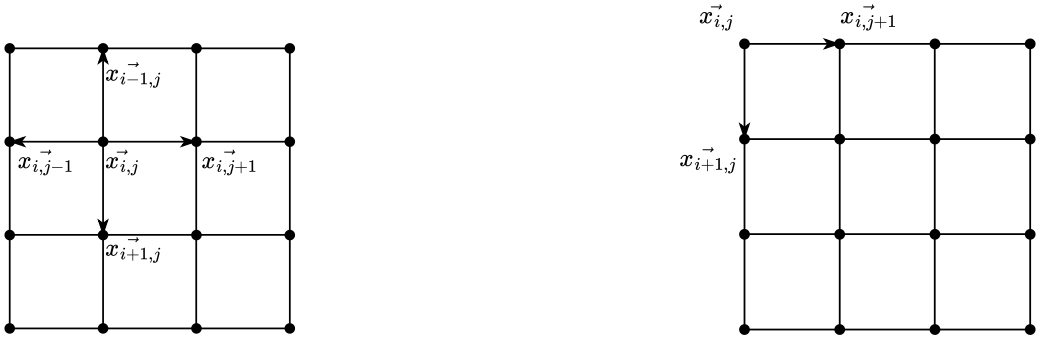
Kada se radi o mreži trokuta pristup je nešto drugačiji, ali glavna ideja je ista. Kako više trokuta u mreži može dijeliti neke vrhove, treba uzeti u obzir i susjedne vrhove kojima je promatrani vrh jedan od vrhova trokuta kojeg tvore. Drugim riječima, moramo uzeti u obzir pozicije susjednih vrhova pri računanju normala kako bi mogli dobiti efekt zakrivljene plohe. površine. Zbog brzine računanja, svaka čestica će uzeti u obzir svoja 2 vertikalna i 2 horizontalna susjeda te efektivno usrednjiti izračunate normale. Definirajmo funkciju s kojom se računaju susjedne normale određenog vrha:

$$\begin{aligned} s(i,j) = {} & (x_{\vec{i-1},j} - x_{\vec{i,j}}) \times (x_{i,\vec{j-1}} - x_{\vec{i,j}}) + (x_{i,\vec{j+1}} - x_{\vec{i,j}}) \times (x_{\vec{i-1},j} - x_{\vec{i,j}}) \\ & + (x_{\vec{i+1},j} - x_{\vec{i,j}}) \times (x_{i,\vec{j+1}} - x_{\vec{i,j}}) + (x_{i,\vec{j-1}} - x_{\vec{i,j}}) \times (x_{\vec{i+1},j} - x_{\vec{i,j}}) \end{aligned} \tag{4.6}$$

Posebnu pozornost moramo posvetiti rubnim slučajevima kad su nezavisne varijable funkcije i i j na rubu tkanine, odnosno kad $i < 0$, $i \geq r$, $j < 0$ ili $j \geq r$. U tom slučaju vektorske produkte gdje bilo koji broj susjeda ne postoji potrebno je izbaciti iz izračuna. Slika 4.2 prikazuje 2 ključna tipa slučaja koja utječu na evaluacije funkcije s. Budući da na slici 4.2b točka $x_{\vec{i,j}}$ ima samo 2 susjeda moguće je evaluirati samo jedan vektorski produkt te u tom slučaju vrijedi

$$s(i,j) = (x_{\vec{i+1},j} - x_{\vec{i,j}}) \times (x_{i,\vec{j+1}} - x_{\vec{i,j}}) \tag{4.7}$$

dok u slučaju kao na slici 4.2a vrijedi način računanja kao po jednadžbi 4.6. Postoji i slučaj gdje točka nije u kutu, već na rubu tkanine. Tada će tkanina imati još jednu susjednu točku te funkcija s će sadržavati još jedan vektorski produkt.



(a) promatrana točka u unutrašnjosti tkanine

(b) promatrana točka u kutu tkanine

Slika 4.2 Normalan i rubni slučaj pozicije točke $\vec{x_{i,j}}$

Nakon evaluacije funkcije s potrebno je samo normalizirati rezultat i tražena normala je dobivena:

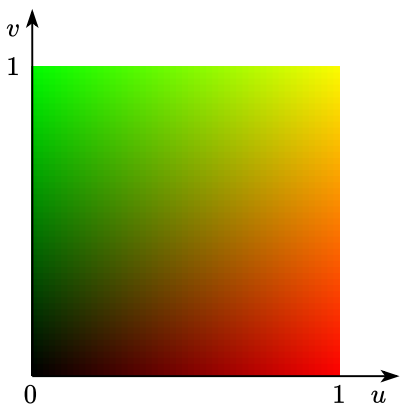
$$\vec{n_{i,j}} = \frac{s(i,j)}{\|s(i,j)\|} \tag{4.8}$$

4.3 Koordinate teksture

Koordinate teksture su generalno dvodimenzionalna vektorska veličina koja određuje kako se uzorkuju teksture pri sjenčanju trokuta. U najosnovnijem obliku za neki pravokutnik izgledat će kao gradijent po x i y-osima odvojeno. Česti naziv za koordinate teksture je *UV koordinate* pa će se taj naziv koristiti nadalje. Uglavnom vrijednosti UV koordinata variraju u intervalu $[0, 1]$ no mogu primiti i druge vrijednosti u specifičnim slučajevima. Kod prikaza vektorskih veličina u grafici najčešće se koriste boje: crvena, zelena i plava za x, y i z-os pojedinačno. Vrijednosti manje ili jednake 0 predstavljaju izuzetak određene boje dok vrijednosti veće ili jednake 1

predstavljaju maksimalni udio te boje, sve ostale vrijednosti se predstavljaju tamnije nijanse te boje. U slučaju UV koordinate u predstavlja x komponentu vektora, a v predstavlja y komponentu vektora.

Slika 4.3a prikazuje dva odvojena gradijenta UV koordinata, kako gledamo prema desno slika postaje sve više crvena, isto tako i prema gore slika postaje više zelena. Gornji desni kut, gdje obje koordinate imaju maksimalnu vrijednost, je žute boje koju dobijemo kombiniranjem crvene i zelene boje. Desni dio slike (4.3b) prikazuje kako možemo koristiti koordinate teksture za uzorkovanje neke slike (teksture).



(a) UV gradijent (b) uzorkovanje korištenjem UV koordinata

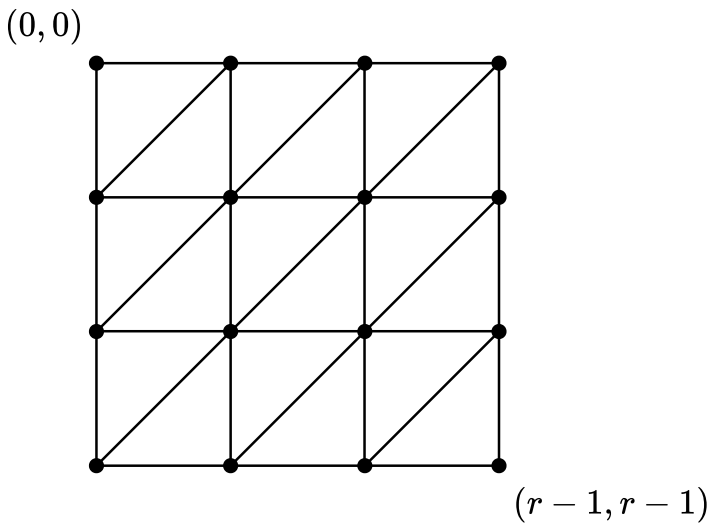
Slika 4.3 Prikaz UV koordinata i uzorkovanje OpenGL logo teksture

UV koordinate $\vec{uv}$ za vrhove pravilne kvadratne rešetke su trivijalne za odrediti po sljedećoj formuli:

$$\vec{uv}_{i,j} = \begin{bmatrix} \frac{j}{r-1} \\ 1 - \frac{i}{r-1} \end{bmatrix} \tag{4.9}$$

4.4 Indeksi

Sada kada su definirani svi potrebni atributi vrhova potrebno je odrediti od kojih vrhova se sastoji svaki trokut unutar mreže kako bi mogli iscrtati tkaninu (trenutno samo kvadrat), a to se može tako da generiramo niz indeksa vrhova. Svaki trokut zahtijeva tri cjelobrojna indeksa koji predstavlja svaki 3 vrha od kojih se sastoji. Zasad imamo kvadratnu rešetku koja se sastoji od malih kvadratića susjednih skupina od po četiri točke. Te kvadratiće je potrebno triangulirati, odnosno razlomiti ih u trokute. Najjednostavniji način triangulacije kvadrata je „dodati" kvadratu dijagonalu tj. prepoloviti ga po dijagonali kao što je prikazano na slici 4.4.



Slika 4.4 Triangulacija kvadratne rešetke

Iz slike se također točno može vidjeti koji indeksi bi pripadali kojem trokutu npr. gornji lijevi trokut to bila indeksi $(0, 0)$, $(0, 1)$ i $(1, 0)$. Budući da OpenGL koristi jednodimenzionalno indeksiranje jer ne poznaje koncept višedimenzionalnih polja potrebno je sve dvodimenzionalne indekse pretvoriti u jednodimenzionalne korištenjem relacije $idx = ir + j$. Nakon pretvorbe dobijemo indekse 0, 1 i r. Definirat ćemo

odmah algoritam za generiranje indeksa sljedećim blokom pseudokoda napisan u C stilu:

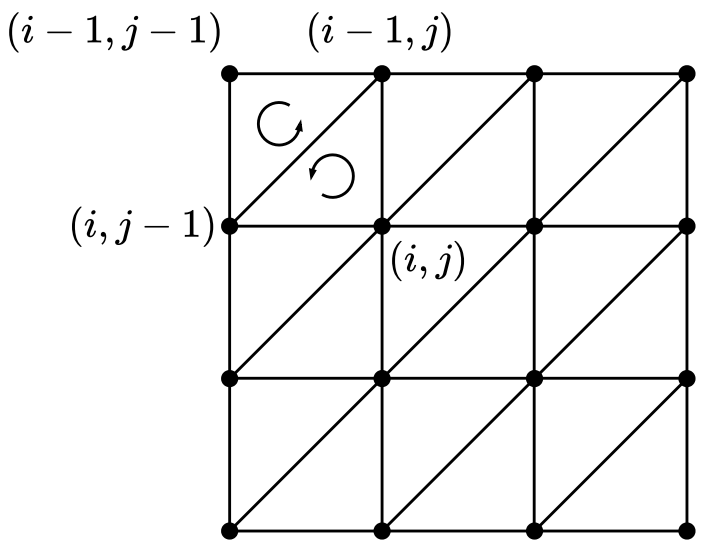
```
idx = Array();
for (i = 1; i < r; i++)
{
    PrevRow = (i - 1) * r;
    CurRow = i * r;
    for (j = 1; j < r; j++)
    {
        PrevColumn = j - 1;
        idx.Append(CurRow + PrevColumn);
        idx.Append(PrevRow + j);
        idx.Append(PrevRow + PrevColumn);

        idx.Append(CurRow + PrevColumn);
        idx.Append(CurRow + j);
        idx.Append(PrevRow + j);
    }
}
```

Analizirajmo prva tri indeksa koja algoritam generira u dvodimenzionalnom zapisu: $(1, 0)$, $(0, 1)$ i $(0, 0)$. Redoslijed je drugačiji od gore navedenog i postoji dobar razlog zašto je redoslijed takav. OpenGL redoslijed vrhova smatra kao dodatnim podatkom, a to je orijentacija plohe trokuta koja može biti prema promatraču ili od promatrača. Ako pogledamo rešetku na slici 4.5 vidimo kako su indeksi vrhova svakog trokuta definirani u smjeru obrnuto od kazaljke na satu. Takav redoslijed govori OpenGL-u da su ti trokuti orijentirani prema promatraču.

Također je moguće primijetiti kako se pri računanju normala vrhova isto pazi na redoslijed operanada vektorskog množenja jer vektorski produkt je upravo način na koji OpenGL određuje ako je trokut vidljiv ili ne. Vektorski produkt je antikomutativna operacija što znači da će zamjena operanada rezultirati suprotnom vrijednosti, matematički izraženo $\vec{a} \times \vec{b} = -(\vec{b} \times \vec{a})$.

Primijetimo isto da rotacijom tkanine za 180° oko x ili y-osi kako su ti isti vrhovi sad odjednom orijentirani u smjeru kazaljke na satu što je obrnuto od originalnog smjera i to je točno jer sad promatramo stražnju stranu tkanine. OpenGL pruža informaciju ako je prednja ili stražnja strana trokuta okrenuta prema promatraču pa se to može iskoristiti za razne efekte. OpenGL isto tako omogućuje promjenu orijentacije koja predstavlja prednju stranu trokuta, ali inicijalna vrijednost je u smjeru obrnuto od kazaljke na satu (matematički pozitivan smjer).



Slika 4.5 Orijentacija indeksa vrhova trokuta

4.5 Shaderi

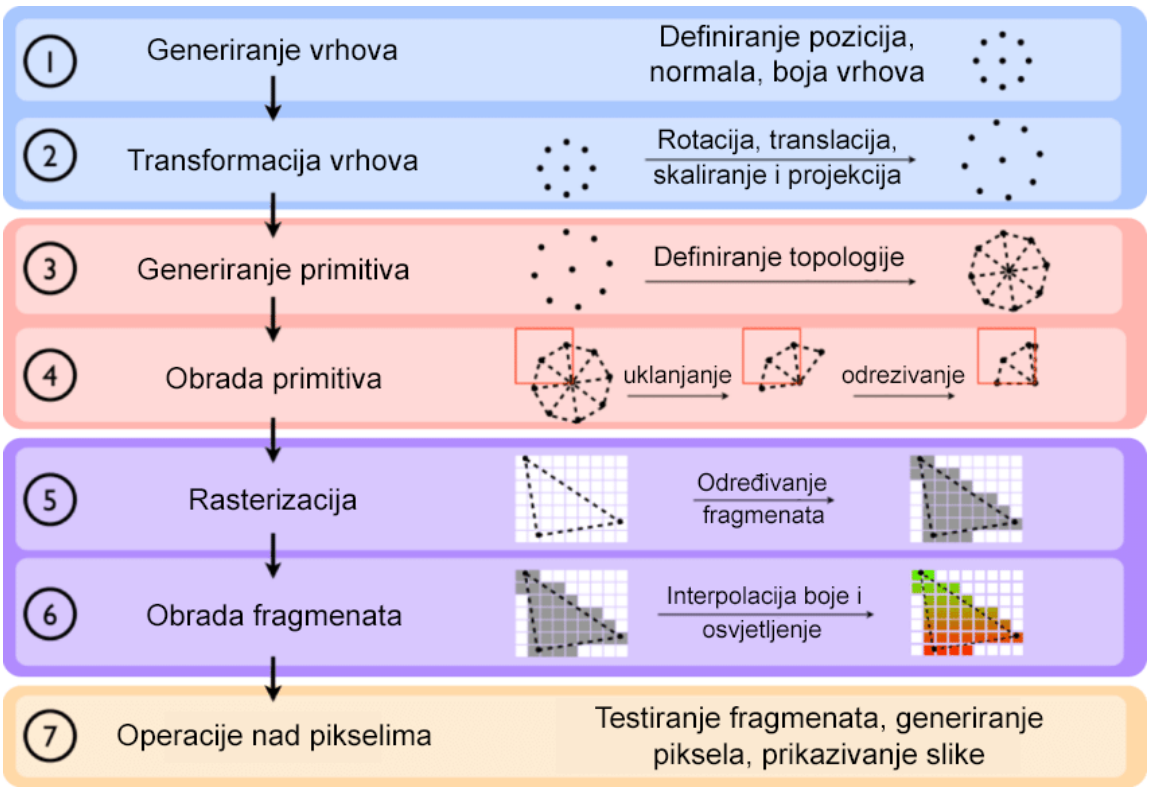
Nakon svog ovog posla napokon je moguće reći OpenGL-u da nacrta nešto na ekran. Aplikacija se pokrene i pojavi se crni ekran. Poprilično nezanimljivo, zar ne? Svi ovi atributi vrhova što smo računali u ovom poglavlju su samo hrpa brojeva, OpenGL ne zna što s njima, ne zna kako interpretirati sve te podatke. Kako bi mogli reći OpenGL-u što da radi sa svim tim podacima moramo napisati tzv. *shadere*.

Shaderi su mali programi koji se vrte na grafičkom procesoru i glavna zadaća im je transformacija geometrije i sjenčanje piksela. Moderni grafički procesori podržavaju oko 8 različitih vrsti shadera od kojih svaka vrsta ima svoju specifičnu domenu i zadaću, ali u ovom radu ograničit ćemo se na minimalan skup vrsti shadera potrebnih za grafički prikaz na ekran, a to su *vertex shader* i *fragment shader*.

Ime vertex shader nam već puno govori o tome što on radi. Pokreće se za svaki vrh (eng. vertex) koji predamo OpenGL-u preko neke od naredbi za crtanje. Ima pristup

svim atributima vrha (pozicija, normale, UV koordinate) za koji je pokrenut te izvršava transformaciju pozicije i ostalih atributa nakon čega ih prosljeđuje sljedećem tipu shadera u nizu (u našem slučaju to je fragment shader jer nijedan drugi nije definiran).

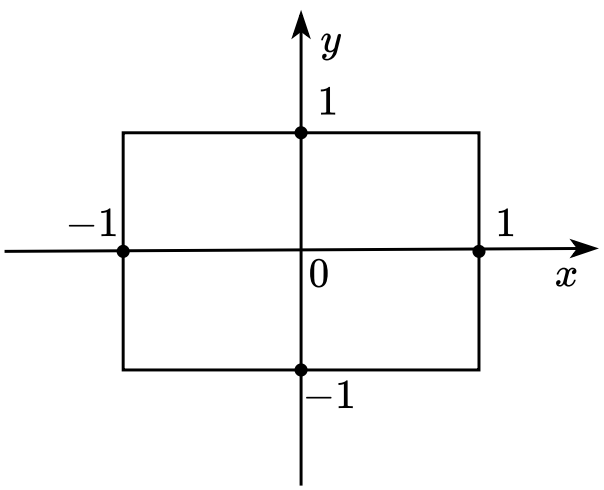
Fragment shader (poznat kao pixel shader) pokreće se za svaki rasterizirani fragment (budući piksel) te je odgovoran za sjenčanje, uzorkovanje tekstura, dodjeljivanje boje pikselu i sl. Fragment shader je onaj koji se izvršava velik broj puta i stoga je potrebno voditi računa da je što jednostavniji i efikasniji za izvršiti. Cijeli pregled grafičkog protočnog sustava (eng. graphics pipeline) možemo vidjeti na slici 4.6.[1]



Slika 4.6 Dijagram grafičkog protočnog sustava

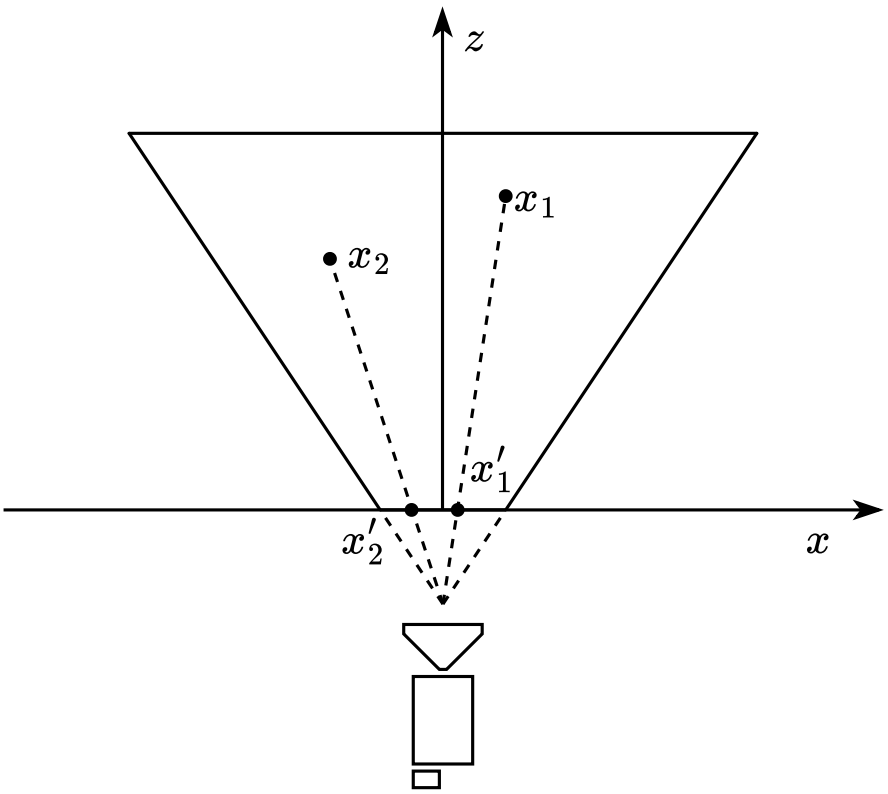
4.5.1 Vertex shader

Cilj je implementirati vertex shader koji će transformirati trodimenzionalnu poziciju vrha u dvodimenzionalne koordinate za prikaz na ekranu i dubinu kako se trokuti koji su iza već iscrtanih trokuta ne bi prikazali. Za postizanje navedenog rezultata treba pretvoriti vektor $\vec{x}$ u $\vec{x'}$ tako da svaka komponenta vektora bude u intervalu $[-1, 1]$. Ta kocka se također zove kanonski volumen pogleda ili *normalized device coordinates*. Slika 4.7 prikazuje presjek kanonskog volumena pogleda xy-ravninom koji se preslikava u koordinate na zaslonu. Sve što je izvan tog prostora (uključujući i po z-osi) bit će „odrezano" te se neće prikazati.



Slika 4.7 Presjek kanonskog volumena pogleda koji se preslikava na zaslon

Sad kad je poznato što treba napraviti treba smisliti kako. Kako mapirati koordinate vrha tako da budu unutar intervala $[-1, 1]$ tako da budu vidljive na ekranu. Potrebno je uzeti u obzir činjenicu da objekti koji su dalje od promatrača izgledaju manji nego oni koji su bliže. Treba nam tzv. perspektivna projekcija kao na slici 4.8.



Slika 4.8 Perspektivna projekcija u dvije dimenzije

Ova projekcija preslikava točku x_1 u x_1' i točku x_2 u x_2' na način da što je točka udaljenija od projekcijske ravnine koja je paralelna s x-osi to je njena slika bliža sredini krnje piramide pogleda (eng. view frustum). Izvod transformacijske matrice za perspektivnu projekciju je poduži pa će umjesto njega biti predstavljen krajnji oblik projekcijske matrice (eng. projection matrix) $\boldsymbol{P}$:

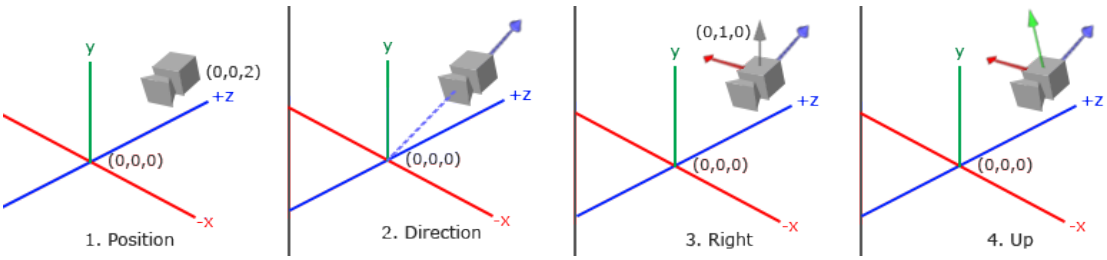
$$\boldsymbol{P} = \begin{bmatrix} \frac{cot(\frac{\theta}{2})}{\alpha} & 0 & 0 & 0 \\ 0 & cot(\frac{\theta}{2}) & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{4.10}$$

Iz jednadžbe 4.10 vidi se da su za konstrukciju matrice $\boldsymbol{P}$ potrebna sljedeća četiri parametra:

- θ - vertikalni kut vidnog polja
- α - omjer širine i visine zaslona (w/h)
- n - udaljenost projekcijske ravnine od kamere (pogleda, promatrača)
- f - krajnja udaljenost odrezivanja

Perspektivna transformacija je tehnički dovoljna za ispravan prikaz trodimenzionalnog virtualnog svijeta, samo što nas ograničava na jedan jedini kut gledanja. Bez dodatne transformacije ne možemo se gibati i okretati po svijetu i vidjeti objekte iz neke druge perspektive, zapeli smo u ishodištu gledajući niz negativnu z-os. Rješenje leži u matrici pogleda (eng. view matrix) V koja predstavlja Euklidsku transformaciju (rotacija i translacija).

Uzmimo u obzir poziciju kamere (pogleda) $\vec{c}$ i točku u koju želimo gledati $\vec{l}$. Vektor smjera pogleda definiran je kao $\vec{d} = \vec{l} - \vec{c}$. Oko tog vektora potrebno je konstruirati cijeli koordinatni sustav, zasad imamo prednji vektor, još je potrebno definirati desni i gornji vektor. Svaki od tri vektora mora biti okomit sa svakim drugim i jedinične duljine. Vektorski produkt zvuči nešto što bi riješilo problem, ali nemamo drugi referentni vektor. Ili ipak imamo? Ako pretpostavimo da nećemo naginjati pogled lijevo i desno možemo uzeti bazni vektor y-osi $\hat{j} = (0, 1, 0)$ kao referentni vektor i preko pravila desne ruke odrediti desni vektor $\vec{r} = \hat{d} \times \hat{j}$. Može se činiti da je već sve gotovo, ali nije definiran gornji vektor, referentni vektor $\hat{j}$ ne možemo koristiti kao gornji vektor jer ako npr. spustimo pogled prema dolje vektor više neće biti poravnat s vektorom $\hat{j}$ i to će uzrokovati deformacije pogleda. Ponovnim korištenjem pravila desne ruke dobijemo $\hat{u} = \hat{r} \times \hat{d}$. Jedna mala dorada je potrebna kako bi postigli ispravnost sastavljenog koordinatnog sustava. Spomenuto je pravilo desne ruke i kako bez transformacije pogleda gledamo niz negativnu z-os, analogno tome koordinatni sustav kamere mora imati vektor koji je usmjeren u suprotnom smjeru od točke u koju gledamo pa dobijemo stražnji vektor $\hat{b} = -\hat{d}$ i to je gotov koordinatni sustav, referenca na slici 4.9 [2].



Slika 4.9 Sastavljanje koordinatnog sustava pogleda

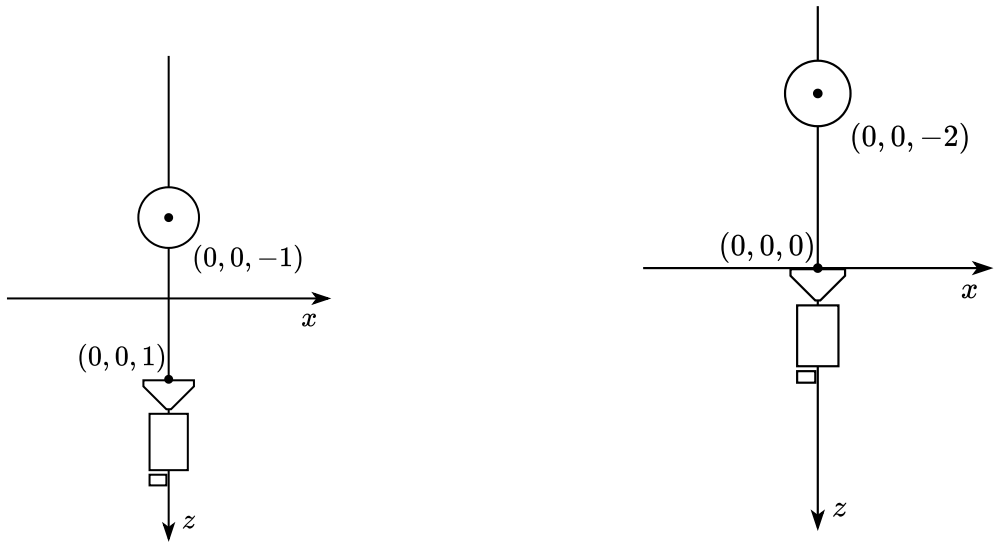
Matrično zapisano bi izgledalo ovako:

$$\boldsymbol{R} = \begin{bmatrix} \hat{r}_x & \hat{u}_x & \hat{b}_x \\ \hat{r}_y & \hat{u}_y & \hat{b}_y \\ \hat{r}_z & \hat{u}_z & \hat{b}_z \end{bmatrix} \tag{4.11}$$

Činjenica da kamera kao takva ne postoji kao objekt u virtualnom svijetu nego je samo apstrahirana kroz matricu pogleda znači da ne možemo rotirati kameru nego moramo rotirati svijet dok se pravimo da je kamera u ishodištu i gleda niz negativnu z-os. Kad okrećemo glavu u stvarnom životu sve izgleda kao da se okreće u suprotnom smjeru, isti koncept treba primijeniti i u OpenGL-u. Matrica $\boldsymbol{R}$ je tzv. ortonormirana matrica što znači da su sva tri bazna vektora (stupca) jedinične duljine i međusobno okomita. Kako se svijet rotira suprotno od rotacije kamere, matricu $\boldsymbol{R}$ treba invertirati kako bi se dobila suprotna rotacija, iznimno korisno svojstvo ortonormiranih matrica je da je operacija inverzije jednaka operaciji transponiranja:

$$\boldsymbol{R^{-1}} = \boldsymbol{R^T} = \begin{bmatrix} \hat{r}_x & \hat{r}_y & \hat{r}_z \\ \hat{u}_x & \hat{u}_y & \hat{u}_z \\ \hat{b}_x & \hat{b}_y & \hat{b}_z \end{bmatrix} \tag{4.12}$$

Riješen je problem orijentacije kamere, još ostaje za riješiti problem pozicije kamere. Pozicija kamere može se gledati na sličnom principu kao rotacija samo jednostavnije, ako se kamera pomakne npr. za jednu jedinicu po negativnoj z-os, cijeli svijet se zapravo pomakne jednu jedinicu po pozitivnoj z-osi. Vizualno je lakše dočarati što se događa, a to prikazuje slika 4.10. Udaljenost kamere od kružnice jednaka je u oba slučaja. Translacijom svijeta za $-\vec{c}$ kamera dolazi u ishodište koordinatnog sustava i sve udaljenosti u odnosu na kameru ostaju iste.



(a) scena u globalnom koordinatnom sustavu

(b) scena u lokalnom koordinatnom sustavu kamere

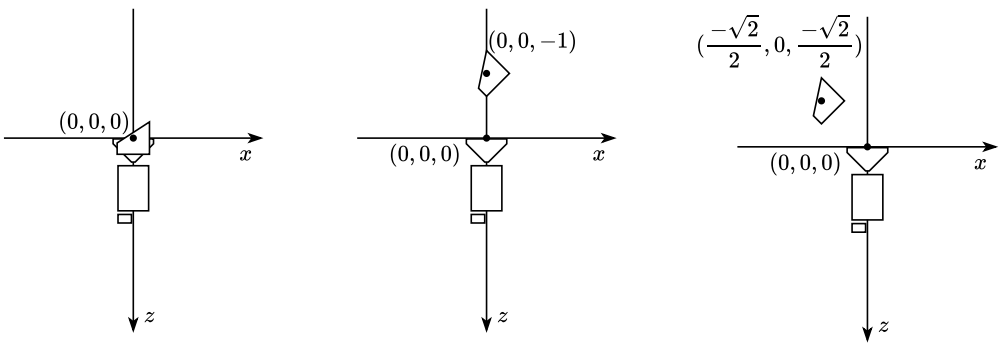
Slika 4.10 Pozicioniranje kamere

Kombinacijom matrice $\boldsymbol{R^T}$ i translacije za $-\vec{c}$ dobijemo konačnu matricu pogleda s tim da treba paziti na redoslijed množenja matrica jer matrično množenje nije komutativno i potrebno je paziti kojim redoslijedom se primjenjuju transformacije. Kako translacija nije linearna transformacija potrebno je proširiti matricu $\boldsymbol{R^T}$ na 4 x 4 matricu $\boldsymbol{R'^T}$ i sastaviti transformacijsku matricu $\boldsymbol{T}$:

$$\boldsymbol{R'^T} = \begin{bmatrix} \hat{r}_x & \hat{r}_y & \hat{r}_z & 0 \\ \hat{u}_x & \hat{u}_y & \hat{u}_z & 0 \\ \hat{b}_x & \hat{b}_y & \hat{b}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{T} = \begin{bmatrix} 0 & 0 & 0 & -\vec{c}_x \\ 0 & 0 & 0 & -\vec{c}_y \\ 0 & 0 & 0 & -\vec{c}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

Ispravan redoslijed kombinacije matrica $\boldsymbol{R'^T}$ i $\boldsymbol{T}$ može se logički zaključiti jednos-

tavnim primjerom. Uzmimo da je pozicija kamere $\vec{c} = (0, 0, 1)$, objekt ima pozicijski vektor $\vec{x} = (0, 0, 0)$ i kamera je zarotirana za 45° stupnjeva u smjeru kazaljke na satu. Primjenom prvo rotacije pa translacije dobijemo pogled prema centru objekta, dok primjenom prvo translacije pa rotacije objekt je s lijeve strane vidnog polja. Ilustracija primjera na slici 4.11 prikazuje kako se bi scena izgledala s različitim matricama pogleda $\boldsymbol{V}$.



(a) $\boldsymbol{V} = \boldsymbol{I}$ (b) $\boldsymbol{V} = \boldsymbol{T}\boldsymbol{R}'^{\boldsymbol{T}}$ (c) $\boldsymbol{V} = \boldsymbol{R}'^{\boldsymbol{T}}\boldsymbol{T}$

Slika 4.11 Izgled scene s primijenjenim različitim matricama pogleda

Slika 4.11c jedina prikazuje ispravnu matricu pogleda jer je moguće vidjeti samo jednu stranu objekta i još se nalazi na lijevoj strani vidnog polja. Time smo zaključili da se prvo radi translacija pa tek onda rotacija te stoga vrijedi:

$$\boldsymbol{V} = \boldsymbol{R}'^{\boldsymbol{T}}\boldsymbol{T} \tag{4.14}$$

Sve je definirano te je preostalo samo implementirati vertex shader sa svim navedenim konceptima, transformiranu poziciju $\vec{x}' = \boldsymbol{P}\boldsymbol{V}\vec{x}$ ćemo dodijeliti GLSL varijabli *gl_Position* i proslijediti sve ostale atribute vrha dalje u fragment shader čime je zaključen vertex shader.

4.5.2 Fragment shader

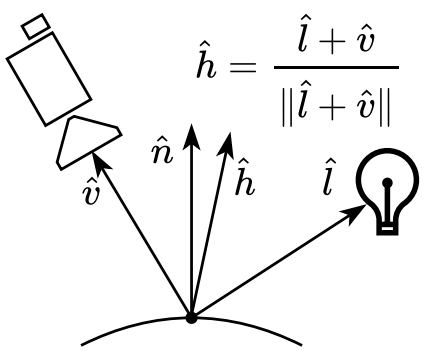
Unutar fragment shadera će se računati krajnja boja svakog piksela korištenjem interpoliranih podataka dobivenih od vertex shadera koji će biti prikazan na ekranu nakon crtanja. Koristit će se pojednostavljeni Blinn-Phong model sjenčanja[3] čiji je potpun oblik s raznim parametrima zadan sljedećom jednadžbom:

$$I_x = k_a i_a + k_d(\hat{l} \cdot \hat{n}) i_d + k_s \left(\hat{n} \cdot \frac{\hat{l} + \hat{v}}{\|\hat{l} + \hat{v}\|} \right)^{\alpha} i_s \tag{4.15}$$

Malo ćemo pojednostaviti model i maknuti članove koji nam nisu potrebni. Eliminirat ćemo ambijentalno osvjetljenje kompletno (k_a i i_a), nakon toga ćemo pretpostaviti da vrijedi $i_d = i_s$ što predstavlja intenzitet svjetla za različite komponente modela budući da se koristi isti izvor svjetla. Uz to ćemo još uzeti da $k_d = k_s = 1$ i time se dobije pojednostavljeni model.

$$I_x = i \left[\hat{l} \cdot \hat{n} + \hat{n} \cdot \left(\frac{\hat{l} + \hat{v}}{\|\hat{l} + \hat{v}\|} \right)^{\alpha} \right] \tag{4.16}$$

Ovo je već puno lakše za zamisliti, ali i dalje ne govori puno pa će biti popraćeno vizualnim objašnjenjem na slici 4.12.



Slika 4.12 Vektorske veličine kod Blinn-Phong modela sjenčanja

Sve vektorske veličine koje sudjeluju u definiciji intenziteta osvjetljenja jedinične su duljine jer nam je od interesa isključivo kut između njih. Vrijednost skalarnog produkta iznosi $\vec{a} \cdot \vec{b} = \|a\|\|b\|cos(\theta)$ te ako su oba vektora jedinične duljine s desne strane ostaje samo $cos(\theta)$ - kut koji vektori zatvaraju.

Površine su najjače osvjetljenje kad zrake padaju okomito na površinu, odnosno kad su zrake paralelne s normalom površine $\hat{n}$. Upravo je opisana difuzna komponenta osvjetljenja, komponenta koja ne ovisi o položaju promatrača već grubo modelira raspršenje zraka svjetlosti podjednako u svim smjerovima. Druga važna komponenta osvjetljenja je tzv. zrcalna (eng. specular) komponenta koja se na glatkim površinama primijeti kao svjetlost visokog intenziteta kao posljedica izravne refleksije od površine. Zrcalna komponenta ima najveći intenzitet kad se vektor na pola puta između $\hat{v}$ i $\hat{l}$ nazvan $\hat{h}$ poklapa s normalom površine, tada se zraka savršeno reflektira od površine izravno prema promatraču.

Intenzitet svjetlosti izvora svjetla i nije bio označen kao vektor zbog mogućnosti kompleksnijeg definiranja boje i intenziteta, međutim u klasičnoj računalnoj grafici to je zapravo skalarna veličina s tri komponente koja opisuje udio svake od tri temeljne boje: crvene, zelene i plave, tim redoslijedom. Dosad je bilo korišteno indeksiranje za komponente vektora simbolima x, y i z; za boje je prikladnije koristiti r, g i b, ali oboje je tehnički ispravno te će biti naizmjenično korišteno, ovisno o kontekstu.

Došao je trenutak za provjeriti ako se sav ovaj mukotrpan posao isplatio. Shaderima ćemo poslati obje transformacijske matrice, poziciju kamere i svjetla te boju tkanine i pokrenuti program. Ako je sve ispravno odrađeno moći će se vidjeti nešto kao na slici 4.13.

Slika 4.13 Komad tkanine nacrtan korištenjem OpenGL-a

Napokon imamo nešto nacrtano na ekranu, ne radi ništa, ali bar lijepo izgleda. Ovime je zaključeno poglavlje o crtanju tkanine te u sljedećem poglavlju bit će objašnjeni postupci kako napraviti da ovaj kvadrat oživi i ponaša se kao uvjerljiv komad tkanine.

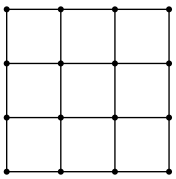
Poglavlje 5

Simulacija tkanine

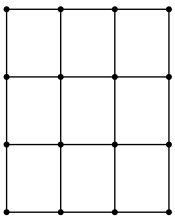
Do ove točke smo zadovoljavali sve preduvjete za grafički prikaz tkanine i sad kad se napokon nešto vidi na ekranu možemo početi raditi na tome da tkanina počne pratiti zakone fizike te se gibati po svijetu, rastezati, savijati i smicati. Već je spomenuto u ranijem poglavlju kako će se koristiti model masa i opruga, sada samo treba razraditi detalje i kako će se taj model inkorporirati u već postavljene temelje.

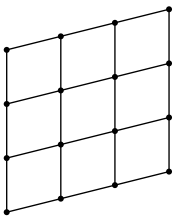
5.1 Model masa i opruga

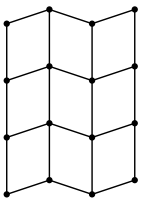
Model masa i opruga prirodno dolazi iz činjenice da atomi i subatomske čestice iziskuju privlačne i odbojne sile međusobno te uz to imaju masu koja je možda jako mala, ali na toj skali vrlo značajna. Opruge tu dolaze u priču jer, kao i prava tkanina, omogućuju lokalizirano rastezanje. Čestice su međusobno povezane prividnim oprugama koje pokušavaju spriječiti tkaninu da se širi, kompresira ili smiče previše od ravnotežnog položaja. Ono što je jako slabo ograničeno je savijanje. Praktički svaki komad tkanine u pravom životu može se relativno lako savinuti kako bi zauzimao manje prostora u jednoj ili dvije dimenzije nauštrb zauzimanja više prostora u trećoj dimenziji. Navedene promjene oblika prikazane su na slici 5.1.



(a) početni položaj







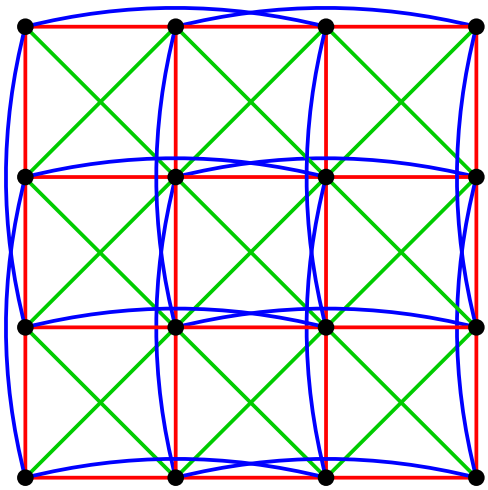
(b) rastezanje (c) smicanje (d) savijanje

Slika 5.1 Vrste promjena oblika tkanine

5.1.1 Postavljanje opruga

Prava tkanina se opire svakoj promjeni oblika, kako bi ograničili koliko je moguće promijeniti oblik na svaki od navedenih načina potrebno je čestice tkanine povezati oprugama. Budući da se opruge opiru promjeni duljine u odnosu na ravnotežni položaj, pravi su način pružanja otpora deformaciji. Povezat ćemo svaku česticu sa svakom susjednom i još neobavezno možemo povezati svaku česticu sa svakom drugom horizontalno i vertikalno. Slika 5.2 prikazuje 4 x 4 rešetku čestica povezanih oprugama. Opruge su obojene ovisno o tome kojoj vrsti promjene oblika se opiru:

- crvena - rastezanje
- zelena - smicanje
- plava - savijanje



Slika 5.2 Sustav čestica povezanih trima vrstama opruga

Potrebno je napraviti jasnu distinkciju između različitih vrsta opruga na način da svakoj od tri vrste dodijelimo nezavisnu vrijednost konstante opruge k. Vrijednosti konstanti bitno utječu na izgled i ponašanje tkanine te za najrealističniju predodžbu otpor na rastezanje bi trebao biti najveći, popraćen otporom na smicanje te na kraju otporom na savijanje. Zbog načina povezivanja nije moguće kompletno odvojiti utjecaje različitih vrsti opruga na otpor druge vrste. Iz slike 5.1 može se zaključiti da će i zelene i plave opruge itekako utjecati na otpor na rastezanje, no njihov utjecaj će biti puno manji zbog manjih konstanti te u slučaju zelenih opruga i kuta pod kojim su povezani s česticama. [4]

5.1.2 Određivanje mase

Svaka točka unutar modela mora imati definiranu pozitivnu masu, masa je koncentrirana unutar infinitezimalno malog volumena te se takav tip mase naziva točkasta masa. Postoji nekoliko pristupa za određivanje mase:

- konstanta - svaka čestica ima masu jednaku m
- površinska gustoća - definiran je parametar γ s mjernom jedinicom $[kgm^{-2}]$ koji predstavlja masu jedinične površine
- stupanj povezanosti - svaka čestica ima masu proporcionalnu broju susjednih čestica

Neovisno o metodi korisno je tijekom generiranja geometrije tkanine odrediti matricu mase $\boldsymbol{M}$ za svaku česticu. Ta matrica je 3 x 3 dijagonalna matrica koja uzduž cijele dijagonale sadrži masu čestice:

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{5.1}$$

U ovom radu korišten je prvi pristup zbog visokog otpora na nestabilnost i jednostavnosti. Drugi pristupi definitivno se mogu koristiti, ali treba imati na umu da će se prije javiti nestabilnost pri povećavanju rezolucije sustava. Više o problemu nestabilnosti će biti rečeno u kasnijem poglavlju.

5.1.3 Određivanje sile

Stvarni fizički sustavi teže stabilnosti, to postižu minimiziranjem energije sustava. Uzmimo za primjer hidroelektranu: voda iz rijeke pada s visoke točke na generatorsku turbinu pri čemu gubi gravitacijsku potencijalnu energiju. Opruge, gumice i slični predmeti predstavljaju drugi oblik potencijalne energije - elastičnu potencijalnu energiju. Elastična potencijalna energija opruge ima najmanju vrijednost u ravnotežnom položaju te raste kako se opruga rasteže. Potencijalne energije rezultiraju konzervativnim silama. Kod konzervativne sile vrijedi da rad pri pomicanju čestica ne ovisi o putu kojim se čestica gibala već isključivo o razlici početnog i krajnjeg

položaja. Drugim riječima ako promatranje počne dok je čestica u nekoj poziciji i pređe određeni put te se vrati u početnu poziciju smatrat će se da konzervativna sila nije napravila nikakav rad. Konzervativna sila F djeluje kako bi smanjila potencijalnu energiju čestice što znači da ako promjena položaja čestice uzrokuje povećanje energije sila mora djelovati u suprotnom smjeru. Odnosno:

$$F(x) = -\frac{dE(x)}{dx} \tag{5.2}$$

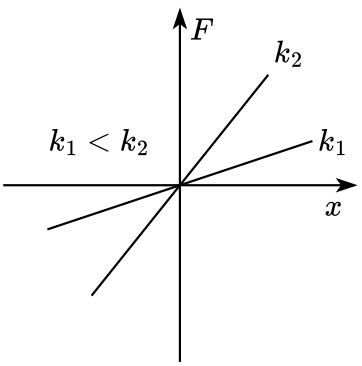
Elastična potencijalna energija opruge konstante k i pomaka od ravnotežnog položaja x iznosi:

$$E(x) = \frac{x^2}{2} \tag{5.3}$$

Deriviranjem jednadžbe 5.3 dobivamo Hookeov zakon:

$$F(x) = -kx \tag{5.4}$$

Takve opruge nazivaju se linearne, odnosno Hookeove opruge (slika 5.3).



Slika 5.3 Odnos sile i pomaka od ravnotežnog položaja za dvije različite Hookeove opruge

Za gravitaciju znamo da gravitacijska potencijalna energija E_g iznosi $E_g = mgh$. U našem virtualnom svijetu visina je predstavljena y komponentom vektora pozicije

$\vec{x}$. Vektor gravitacijske potencijalne energije tada iznosi:

$$\vec{E_g}(x) = mg \begin{bmatrix} 0 \\ \vec{x}_y \\ 0 \end{bmatrix} \tag{5.5}$$

Iz toga dobijemo:

$$\vec{F_g}(x) = -\frac{\partial E_g(x)}{\partial x} = -\begin{bmatrix} \frac{\partial \vec{E_g}}{\partial x_x} \\ \frac{\partial \vec{E_g}}{\partial x_y} \\ \frac{\partial \vec{E_g}}{\partial x_z} \end{bmatrix} = -\begin{bmatrix} 0 \\ mg \\ 0 \end{bmatrix} \tag{5.6}$$

Jednadžba 5.4 nam govori kako se ponaša u odnosu na pomak opruge od ravnotežnog položaja u jednoj dimenziji dok se u našem slučaju radi o opruzi koja ima mase na oba kraja te rastezanje može uzrokovati bilo koja od te dvije mase. Ako promjenu položaja x zamijenimo s razlikom udaljenosti između čestica i i j i duljinom opruge u ravnotežnom položaju L_0 dobijemo sljedeći izraz za elastičnu potencijalnu energiju:

$$E_{ij}(\vec{x_i}, \vec{x_j}) = \frac{1}{2}k(\|\vec{x_i} - \vec{x_j}\| - L_0)^2 \tag{5.7}$$

Sila koja djeluje na česticu može se dobiti derivacijom funkcije elastične potencijalne energije po poziciji te čestice te za česticu i iznosi:

$$\begin{aligned} F_i(\vec{x_i}, \vec{x_j}) &= -\frac{\partial E_{ij}(\vec{x_i}, \vec{x_j})}{\partial \vec{x_i}} \\ &= -k(\|\vec{x_i} - \vec{x_j}\| - L_0)\frac{\partial \|\vec{x_i} - \vec{x_j}\|}{\partial \vec{x_i}} \\ &= -k(\|\vec{x_i} - \vec{x_j}\| - L_0)\frac{\vec{x_i} - \vec{x_j}}{\|\vec{x_i} - \vec{x_j}\|} \end{aligned} \tag{5.8}$$

Intuitivno se može zaključiti da sila koja djeluje na česticu j mora biti suprotnog smjera od sile koja djeluje na česticu i jer su povezane istom oprugom, ali izvest ćemo ju kako bi dokazali ispravnost intuicije:

$$\begin{aligned} -F_j(\vec{x_i}, \vec{x_j}) = F_i(\vec{x_i}, \vec{x_j}) &= \frac{\partial E_{ij}(\vec{x_i}, \vec{x_j})}{\partial \vec{x_j}} \\ &= k(\|\vec{x_i} - \vec{x_j}\| - L_0)\frac{\vec{x_i} - \vec{x_j}}{\|\vec{x_i} - \vec{x_j}\|} \end{aligned} \tag{5.9}$$

Postizanje stabilnosti simulacije može biti jako problematično pri korištenju modela masa i opruga, jedan od najlakših načina za postizanje stabilnosti je dodavanje prigušenja u sustav. Bez prisutnog prigušenja sustav može ući u oscilatorni ciklus koji u najboljem slučaju ostaje isti, a u najgorem slučaju raste u magnitudi što dovodi do eksplozije tkanine u beskonačnost i kompletan raspad simulacije. Jednostavan način za dodati prigušenje je uvesti silu koja djeluje u smjeru suprotnom od smjera kretanja čestice. Između čestice i i j povezane oprugom možemo definirati silu prigušenja na sljedeći način:

$$F_{di}(\vec{v_i}, \vec{v_j}) = -k_d(\vec{v_i} - \vec{v_j}) = -F_{dj}(\vec{v_i}, \vec{v_j}) \tag{5.10}$$

Koeficijent k_d određuje razinu prigušenja, njegova vrijednost je obično u intervalu $[0, 1]$ te za visoke vrijednosti može učiniti da se tkanina giba kao da je unutar nekog gustog medija kao što je tekućina. Kao nekakvo pravilo dobro je odabrati vrijednost k_d između 0.15 i 0.25 kako bi se postigla veća razina stabilnosti bez gubitka realističnog ponašanja tkanine.

5.2 Eksplicitna integracija

Gotovo svatko tko je u nekom trenutku radio u području razvoja videoigara je u nekom trenutku implementirao gibanje krutih tijela korištenjem eksplicitne integracije. Ova metoda integracije je vrlo intuitivna i lagana za shvatiti i implementirati te je dovoljna za primjenu u većini simulacija dinamike krutih tijela za aplikacije koje rade u realnom vremenu. Unatoč tome što se iznimno često koristi i vrlo je efikasno i efektivno rješenje, ima svoje probleme koji će se baš pokazati problematični za simulaciju tkanine. Ti problemi će biti demonstrirani i objašnjeni u ovom poglavlju.

Već smo definirali kako je minimalno stanje čestice potrebno za vršiti simulaciju pozicija $\vec{x}$ i brzina $\vec{v}$ koje su trodimenzionalne vektorske veličine koje se mijenjaju kroz vrijeme. Početna vrijednost pozicije za česticu i zadana je kvadratnom rešetkom objašnjenom u potpoglavlju 4.1, dok je svake šestice jednaka nul-vektoru, odnosno nema početnu brzinu. Na promjenu pozicije čestice utječe brzina čestice, ali što utječe na brzinu čestice? Odgovor na to pitanje je sila, odnosno više njih kao što

su definirane u potpoglavlju iznad. Budući da se brzina definira kao promjena pozicije kroz vrijeme, a akceleracija promjena brzine kroz vrijeme možemo to zapisati matematički na sljedeći način:

$$\begin{aligned} \vec{v}(t) &= \frac{d\vec{x}(t)}{dt} \\ \vec{a}(t) &= \frac{d\vec{v}(t)}{dt} \end{aligned} \tag{5.11}$$

Iz drugog Newtonovog zakona gibanja znamo da je sila promjena količine gibanja u vremenu te iz toga možemo dobiti izraz za akceleraciju u vremenu t:

$$\begin{aligned} \vec{p}(t) &= m\vec{v}(t) \\ \vec{F}(t) &= \frac{d\vec{p}(t)}{dt} = \frac{d(m\vec{v}(t))}{dt} \\ &= m\frac{d\vec{v}(t)}{dt} = m\vec{a}(t) \implies \vec{a}(t) = \frac{\vec{F}(t)}{m} \end{aligned} \tag{5.12}$$

Sada imamo sve alate potrebne za implementaciju eksplicitne integracije, ali ih je potrebno prilagoditi prije nego što ih možemo početi koristiti. Na početku rada spomenuto je kako kada radimo bilo kakvu simulaciju na računalu moramo diskretizirati sve. Derivacije definiraju promjenu vrijednosti funkcije za infinitezimalno malu promjenu u argumentu po kojem se vrši derivacija. Računalo ne može obraditi podatke u tako malom vremenskom intervalu te možemo jedino aproksimirati promjenu u vremenu uvođenjem konačne varijable Δt pri čemu izrazi u jednadžbi 5.11 počinju podsjećati na definiciju derivacije:

$$\begin{aligned} \vec{v}(t) &\approx \frac{\vec{x}(t+\Delta t) - \vec{x}(t)}{\Delta t} \\ \vec{a}(t) &\approx \frac{\vec{v}(t+\Delta t) - \vec{v}(t)}{\Delta t} \end{aligned} \tag{5.13}$$

Sređivanjem izraza dobijemo vrijednosti pozicije i brzine u idućem vremenskom koraku:

$$\begin{aligned} \vec{x}(t+\Delta t) &\approx \vec{x}(t) + \Delta t\vec{v}(t) \\ \vec{v}(t+\Delta t) &\approx \vec{v}(t) + \Delta t\vec{a}(t) \end{aligned} \tag{5.14}$$

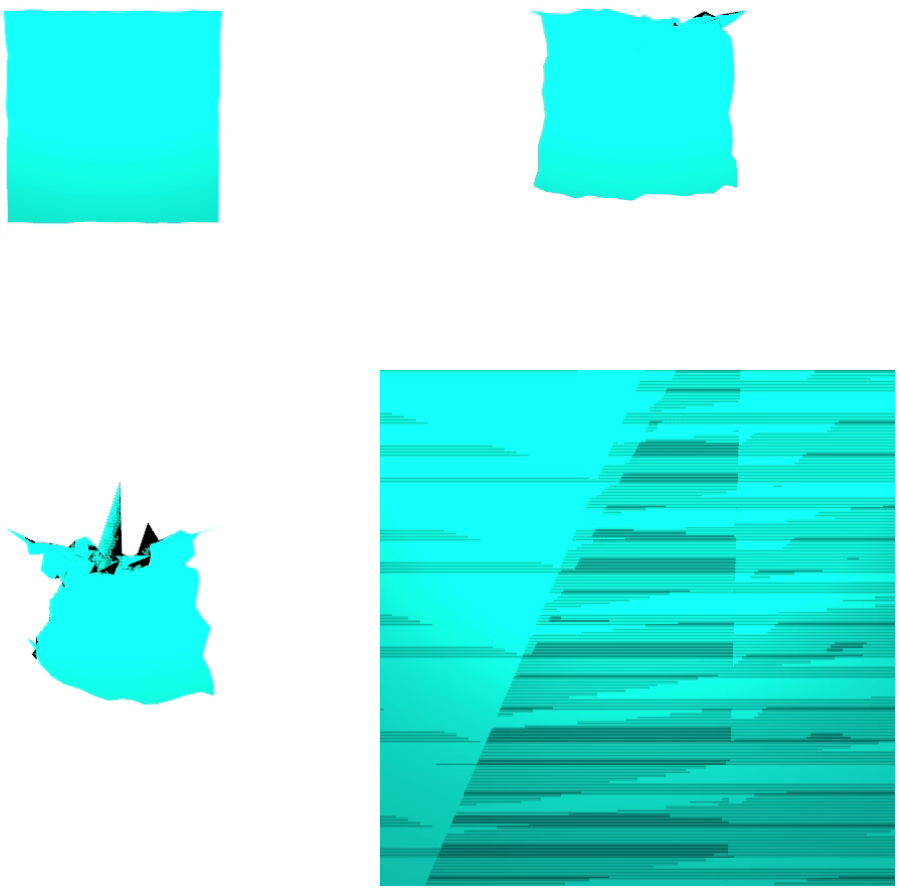
Konačnu akceleraciju čestice u vremenu t možemo dobiti tek nakon što se odrede sve sile koje djeluju na česticu u vremenskom trenutku t. Dvije su vrste sila koje mogu djelovati na česticu:

- Unutarnje (interne) sile koje dolaze iz samog sustava, odnosno one čiji su uzrok opruge. Takve sile nastaju zbog promjena položaja čestica tkanine kao što su rastezanje, smicanje i savijanje.
- Vanjske (eksterne) sile koje dolaze izvan sustava kao što su gravitacija, otpor medija, sudari, vjetar i sl. Ovaj oblik sila ne ovisi o odnosu čestica tkanine, ali može ovisiti u položaju tkanine npr. vjetar i otpor medija.

Kada spojimo jednadžbe 5.12 i 5.14 te ih prebacimo u vremenski diskretizirani oblik gdje indeks n predstavlja broj simulacijskog koraka od početka simulacije dobijemo sljedeće:

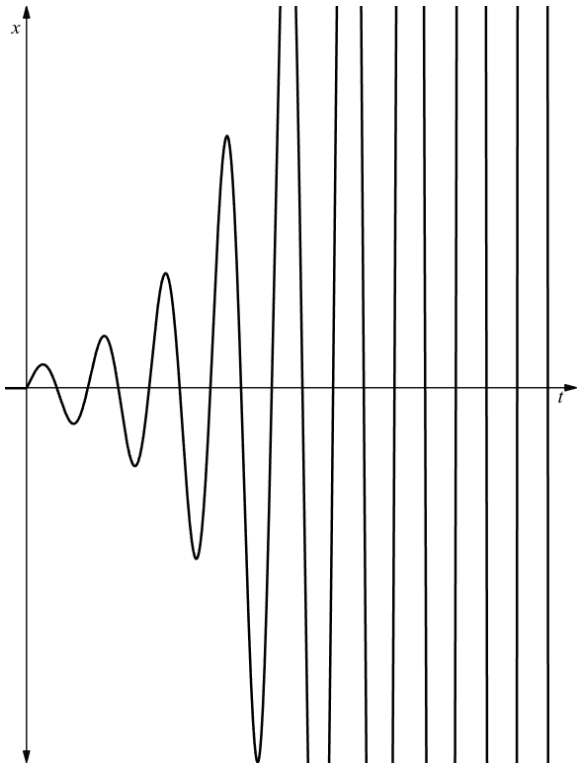
$$\begin{aligned} \vec{x}_{n+1} &= \vec{x}_n + \Delta t \vec{v}_n \\ \vec{v}_{n+1} &= \vec{v}_n + \Delta t \frac{\vec{F}_n}{m} \end{aligned} \tag{5.15}$$

Pokušajmo implementirati ovaj model i vidjeti kako se ponaša. Kao parametre simulacije pretpostavit ćemo konstante opruga $75\frac{N}{m}$ za opruge koje se opiru rastezanju, $22\frac{N}{m}$ za opruge koje se opiru smicanju te $8\frac{N}{m}$ za opruge koje se opiru savijanju. Pretpostavit ćemo postojanje gravitacije, koeficijent otpora medija $k_d = 0.25$ te konstantnu vrijednost vremenskog koraka $\Delta t = \frac{1s}{60}$ kako bi postigli da se nakon 60 frameova pri 60 FPS tkanina simulira kao da je protekla jedna sekunda - prihvatljiva vrijednost za videoigre. Ubrzo nakon pokretanja simulacije tkanina se otme kontroli i dogodi se situacija slična kao na slici 5.4.



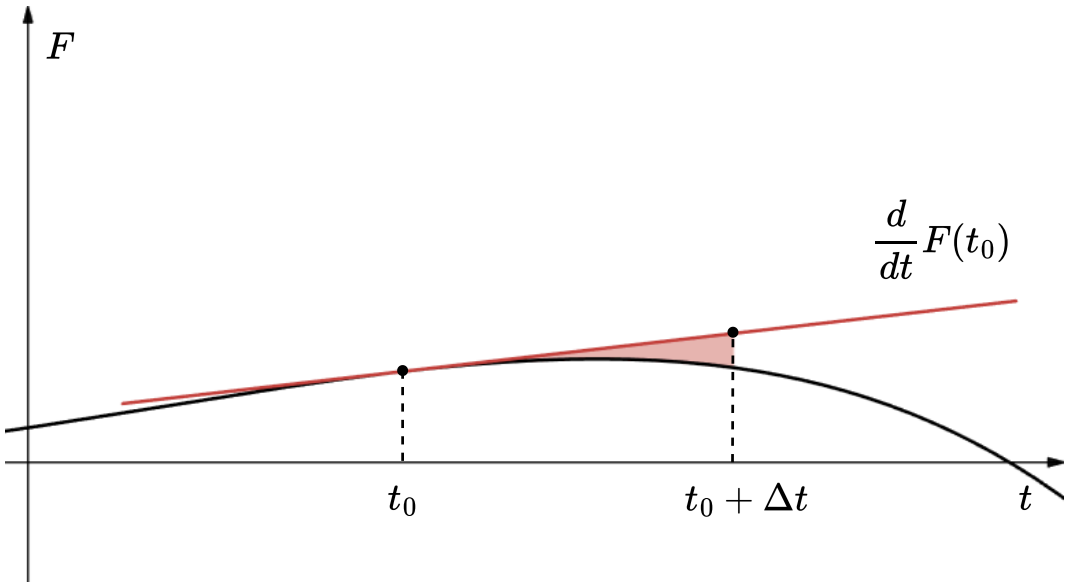
Slika 5.4 Prikaz numeričke nestabilnosti pri eksplicitnoj integraciji

Upravo smo svjedočili numeričkoj nestabilnosti algoritma implicitne integracije. Prva slika prikazuje tkaninu neposredno nakon početka simulacije te progresivno veće razine nestabilnosti sve do četvrte slike gdje je tkanina jedan simulacijski korak od nestajanja u beskonačnost. Tkanina je zbog naglih promjena sila počela oscilirati uz samopobudu rezultirajući konstantno povećavajućoj amplitudi oscilacije koja je dovela do eksplozije pozicija čestica u beskonačnost. Pojednostavljeni prikaz takve oscilacije može se vidjeti na slici 5.5.



Slika 5.5 Grafički prikaz oscilacije s eksponencijalno rastućom amplitudom

Detaljnije objašnjeno, oscilacije uzrokovane naglom promjenom sile i većim vremenskim korakom Δt pojavljuju se zbog precjenjivanja utjecaja sile na brzinu čestice, odnosno dolazi do prevelike promjene količine gibanja u simulacijskom koraku. Ta velika promjena može izazvati naglo veliku promjenu sile u suprotnom smjeru koja će potencijalno opet biti precijenjena i uzrokovati još veću promjenu sile opet suprotnom smjeru pri čemu će čestica konstantno oscilirati s povećavajućom amplitudom. Slika 5.6 prikazuje integracijsku pogrešku uzrokovanu većim vremenskim korakom Δt gdje je crveno osjenčano područje pogreška integracije koja precjenjuje promjenu količine gibanja čestice.



Slika 5.6 Integracijska pogreška zbog vremenskog koraka Δt

Pitanje je kako spriječiti ili barem umanjiti problem? Spriječiti nije moguće nažalost, koliko god se trudili eksplicitna integracija će promjenom parametara kad-tad postati nestabilna. Možemo pokušati umanjiti problem smanjivanjem vremenskog koraka Δt. Smanjivanjem Δt za faktor od 10 dobijemo $\Delta t = \frac{1s}{600}$ te ako sad pokušamo pokrenuti simulaciju, izgledat će usporeno. Razlog tome je što se simulacijski koraci i dalje izvode jednom po frameu, odnosno 60 puta u sekundi i kad uzmemo to u obzir očito je zašto: $60s^{-1}\Delta t = 60s^{-1}\frac{1s}{600} = 0.1$. Simulacija se izvršava pri desetini brzine! Solucija leži u izvršavanju više simulacijskih koraka prije prikazivanja tkanine, točan broj koraka koji treba izvršiti recipročan je upravo izračunatoj vrijednosti, a to za ove konkretne vrijednosti iznosi 10.

Uvođenjem navedene promjene tkanina više ne eksplodira unutar prve sekunde simulacije, ali ovisno o snazi računala moguće je primijetiti kako je simulacija opet usporena. Ispada da kada je potrebno izvršiti 10 puta više simulacijskih koraka s manjim vremenskim korakom da računalo mora napraviti 10 puta više posla što može negativno utjecati na performanse ako je procesor slabijih karakteristika performansi. Mijenjanjem drugih parametara će se promijeniti ponašanje tkanine, ali ako nije mo-

guće postići stabilnost smanjenjem vremenskog koraka zbog gubitka interaktivnosti i rada u realnom vremenu onda valja razmotriti i te mogućnosti.

Tri su faktora sustava koja utječu na razinu numeričke nestabilnosti: koeficijenti opruga, koeficijent prigušenja i masa čestica. Za koeficijente opruga nije teško vidjeti kako utječu na razinu nestabilnosti jer njihovim povećavanjem se povećava magnituda sile koja će djelovati na česticu za isti pomak. Ako vremenski korak nije dovoljno mali te velike promjene će brzo dovesti do velikih oscilacija koje će se samo nastaviti povećavati s vremenom. No ako previše smanjimo koeficijente opruga dobit ćemo tkaninu koja je jako opuštena i više ne izgleda kao tkanina. Koeficijent prigušenja može spriječiti nastanak mikrooscilacija koje se mogu brzo oteti kontroli te zato uvijek mora biti prisutan, ali ne i pretjeran visoke vrijednosti jer će se tkanina ponašati kao da je unutar gustog medija što u sklopu ovog rada nije željeno. Zadnji parametar je masa čestica čiji se utjecaj može vidjeti iz jednadžbe 5.12. Veća masa čestica smanjit će magnitudu akceleracije pa će teže doći do oscilacija, ali povećavanjem mase čestica poništava se utjecaj koeficijenata opruga pa tkanina može izgledati opušteno s povećanjem mase.

Implementacijom eksplicitne integracije otkrili smo kako je vrlo teško postići zadovoljavajuću razinu stabilnosti simulacije uz minimalni utjecaj na ponašanje tkanine. Eksplicitna integracija zahtijeva vrlo malene vremenske korake pri simulaciji te posljedično velik broj simulacijskih koraka za postići numerički stabilne rezultate te je teško primjenjiva u interaktivnim aplikacijama koje rade u realnom vremenu. Metoda koju koristi završna verzija rada bit će objašnjena u idućem potpoglavlju i omogućit će korisniku interaktivnu manipulaciju česticama tkanine uz puno manje vjerojatnosti uvođenja nestabilnosti.

5.3 Implicitna integracija

Glavni ciljevi pri simulaciji tkanine u kontekstu računalne grafike su postizanje vizualno zadovoljavajućih rezultata uz efikasnu uporabu računalnih resursa. To podrazumijeva simulaciju koja se neće lomiti i koja neće imati problema s nestabilnosti te razuman broj simulacijskih koraka u nekom vremenskom intervalu. Implicitna inte-

gracija je alat koji će pomoći ostvariti veću razinu stabilnosti za isti broj simulacijskih koraka.

Uzmimo da smo trenutno u vremenskom koraku t_n i želimo izračunati promjenu pozicije $\Delta\vec{x}$ i brzine $\Delta\vec{v}$ na sljedeći način :

$$\begin{aligned} \Delta\vec{x} &= \Delta t(\vec{v_n} + \Delta\vec{v}) \\ \Delta\vec{v} &= \Delta t(\frac{1}{m}F(\vec{x_n} + \Delta\vec{x}, \vec{v_n} + \Delta\vec{v})) \end{aligned} \tag{5.16}$$

Glavna razlika u odnosu na eksplicitnu integraciju je korištenje vrijednosti funkcije sile iz sljedećeg vremenskog koraka. Činjenica da se pri eksplicitnoj integraciji koristila trenutna vrijednost sile je razlog zašto je toliko jednostavnija za implementirati i efikasna za izračunati. Kod implicitne metode potrebno je odrediti derivacije funkcije sile te preko derivacija odrediti silu u sljedećem vremenskom koraku. Tek nakon toga možemo dobiti promjenu brzine čestice potrebnu za napredak simulacije. U jednadžbi 5.16 definirane su promjene pozicije i brzine te preko njih možemo zapisati poziciju i brzinu u idućem vremenskom koraku kao:

$$\begin{aligned} \vec{x_{n+1}} &= \vec{x_n} + \Delta t\vec{v_{n+1}} \\ \vec{v_{n+1}} &= \vec{v_n} + \Delta t\frac{1}{m}F(\vec{x_{n+1}}, \vec{v_{n+1}}) \end{aligned} \tag{5.17}$$

Pojavljuje se problem što funkcija sile F neće biti linearna te ju je prije rješavanja potrebno linearizirati. Linearizacijom sustava dobivamo jednostavnu aproksimaciju sustava što je prihvatljivo za domenu računalne grafike jer kao nagradu za smanjenu točnost dobijemo efikasniji algoritam te veću vjerojatnost da neće narušiti interaktivnost. Ako uzmemo Taylor polinom prvog stupnja funkcije sile dobijemo sljedeću linearnu funkciju:

$$F(\vec{x_{n+1}}, \vec{v_{n+1}}) \approx F(\vec{x_n}, \vec{v_n}) + \frac{\partial F}{\partial\vec{x}}\Delta\vec{x} + \frac{\partial F}{\partial\vec{v}}\Delta\vec{v} \tag{5.18}$$

Uvrštavanjem jednadžbe iznad u jednadžbu 5.16 dobijemo izraz za $\Delta\vec{v}$

$$\Delta\vec{v} = \Delta t\frac{1}{m}\left(F(\vec{x_n}, \vec{v_n}) + \frac{\partial F}{\partial\vec{x}}\Delta t(\vec{v_n} + \Delta\vec{v}) + \frac{\partial F}{\partial\vec{v}}\Delta\vec{v}\right) \tag{5.19}$$

Nepoznanica $\Delta\vec{v}$ nalazi se na obje strane jednadžbe, prema tome se ova metoda zove implicitna metoda. Nakon sređivanja izraza na način da je nepoznanica s lijeve

strane dobijemo linearni matrični sustav [5]:

$$\left(\boldsymbol{I} - \Delta t\frac{1}{m}\frac{\partial F}{\partial \vec{v}} - \Delta t^2\frac{1}{m}\frac{\partial F}{\partial \vec{x}}\right)\Delta\vec{v} = \Delta t\frac{1}{m}\left(F(\vec{x_n}, \vec{v_n}) + \Delta t\frac{\partial F}{\partial \vec{x}}\vec{v_n}\right) \tag{5.20}$$

Dobiven je matrični sustav oblika $\boldsymbol{A}x = b$ gdje je x naša tražena promjena brzine $\Delta\vec{v}$. Matrični sustav ne možemo riješiti samo dijeljenjem s $\boldsymbol{A}$ jer ne postoji matrično dijeljenje. Možemo birati način na koji ćemo riješiti sustav, za 3x3 matrične sustave možemo izravno pronaći inverz matrice $\boldsymbol{A}$ te njime pomnožiti obje strane matrice čime dobijemo sljedeći izraz $\boldsymbol{A^{-1}A}\Delta\vec{v} = \boldsymbol{A^{-1}}b \implies \Delta\vec{v} = \boldsymbol{A^{-1}}b$. Za veće matrice postoje efikasniji algoritmi koji faktoriziraju matricu $\boldsymbol{A}$ prije rješavanja i iterativni algoritmi. U ovom radu se koristi metoda QR kompozicije za rješavanje sustava.

Definiran je sustav, ali nije ga moguće početi rješavati bez određivanja derivacija internih sila u ovisnosti o promjeni pozicije i brzine čestica. Koristit ćemo skraćeni zapis za razliku pozicija čestica i i j $\vec{x_{ij}} = \vec{x_i} - \vec{x_j}$ da bi izraz koji slijedi bio nešto čitljiviji. Identitet koji je bitan za izvod derivacije sile je sljedeći:

$$\begin{aligned}\frac{\partial \hat{x}}{\partial \vec{x}} &= \frac{\partial(\frac{\vec{x}}{\|\vec{x}\|})}{\partial \vec{x}} \\ &= \frac{\boldsymbol{I} - \hat{x}\hat{x}^T}{\|\vec{x}\|}\end{aligned} \tag{5.21}$$

Korištenjem navedenog identiteta za funkciju $F(\vec{x_i}, \vec{x_j}) = -k(\|\vec{x_{ij}}\| - L_0)\hat{x_{ij}}$ dobijemo njezinu derivaciju po $\vec{x_i}$

$$\begin{aligned}\frac{\partial F(\vec{x_i}, \vec{x_j})}{\partial \vec{x_i}} &= -k\left((\|\vec{x_{ij}}\| - L_0)\frac{\partial \hat{x_{ij}}}{\partial \vec{x_i}} + \hat{x_{ij}}\frac{\partial(\|\vec{x_{ij}}\| - L_0)}{\partial \vec{x_i}}\right) \\ &= -k\left[\left(1 - \frac{L_0}{\|\vec{x_{ij}}\|}\right)\left(\boldsymbol{I} - \hat{x_{ij}}\hat{x_{ij}}^T\right) + \hat{x_{ij}}\hat{x_{ij}}^T\right]\end{aligned} \tag{5.22}$$

Derivacijom po $\vec{x_j}$ kao i pri derivaciji elastične potencijalne energije dobijemo negiranu matricu:

$$\frac{\partial F(\vec{x_i}, \vec{x_j})}{\partial \vec{x_i}} = -\frac{\partial F(\vec{x_i}, \vec{x_j})}{\partial \vec{x_j}} \tag{5.23}$$

Komponenta funkcije F koja ovisi o brzini čestice nije dosad bila spomenuta ni objašnjena, razlog tome je što ta komponenta predstavlja prigušenje, odnosno otpor medija te je sama po sebi vrlo jednostavna i prilikom derivacije po poziciji čestice

nestala bi iz jednadžbe jer se smatra konstantom za odabranu varijablu derivacije. Ispravan puni oblik funkcije F bio bi:

$$F(\vec{x_i}, \vec{x_j}, \vec{v_i}, \vec{v_j}) = -k(\|\vec{x_i} - \vec{x_j}\| - L_0)\frac{\vec{x_i} - \vec{x_j}}{\|\vec{x_i} - \vec{x_j}\|} - k_d(\vec{v_i} - \vec{v_j}) \tag{5.24}$$

Derivacijom funkcije F po brzini čestice gubi se prva komponenta u potpunosti te ostaje samo komponenta prigušenja:

$$\frac{\partial F(\vec{v_i}, \vec{v_j})}{\partial \vec{v_i}} = -k_d \boldsymbol{I} \tag{5.25}$$

Prateći uzorak deriviranih funkcija dosad, promjenom varijable derivacije mijenja se samo predznak:

$$-\frac{\partial F(\vec{v_i}, \vec{v_j})}{\partial \vec{v_i}} = \frac{\partial F(\vec{v_i}, \vec{v_j})}{\partial \vec{v_j}} = k_d \boldsymbol{I} \tag{5.26}$$

Stuyck u svojoj knjizi[4] navodi kako cijeli sustav treba predstaviti kao potencijalno ogromnu $3n \times 3n$ matricu koja se većinski sastoji od nula, tzv. rijetka matrica (eng. sparse matrix). Pokušaj implementacije takvog sustava rezultirao je lošim performansama pri relativno niskim rezolucijama tkanine ($r = 16$) što nije prihvatljivo za sustav koji radi u realnom vremenu. Aplikacija ima podršku za maksimalnu rezoluciju $r = 64$ koju ako uzmemo u obzir matrica sustava bit će dimenzija $3r^2 \times 3r^2 = 12288 \times 12288$. Ako za $r = 16$ postoje problemi s performansama, 16 puta više čestica nema smisla ni pokušavati simulirati.

Bolje performanse moguće je postići pojednostavljenjem sustava na način da svaku česticu gledamo individualno umjesto kombiniranja u globalnu matricu sustava. Matrice $\frac{\partial F(\vec{x_n}, \vec{v_n})}{\partial \vec{x}}$ i $\frac{\partial F(\vec{x_n}, \vec{v_n})}{\partial \vec{v}}$ će biti definirane zasebno za svaku česticu. Svako pojednostavljenje sa sobom nosi negativne posljedice, negativna posljedica ovog pojednostavljenja je gubitak utjecaja pomaka jedne od čestica na silu koju integriramo. Budući da pomak obje čestice utječe na iznos sile svake čestice znači da je potrebno 4 derivacije za svaku od matrica $\frac{\partial F(\vec{x_n}, \vec{v_n})}{\partial \vec{x}}$ i $\frac{\partial F(\vec{x_n}, \vec{v_n})}{\partial \vec{v}}$. Ovim pojednostavljenjem smanjujemo taj broj na 2 jer gledamo samo kako će pomak čestice utjecati na iznos sile na tu istu česticu. Efektivno smo prešli s $O(n^2)$ na $O(n)$ vremensku i memorijsku složenost algoritma te time dobili da s rastućim brojem čestica n algoritam postaje sve efikasniji. Izgubili smo dio stabilnosti, ali dobili preko dva reda veličine

brži algoritam što iznimno pomaže našem cilju interaktivne simulacije u realnom vremenu.

Dio izgubljene stabilnosti možemo dobiti određenom razinom „varanja". Što bi se moglo uopće smatrati varanjem? Pod varanjem se ovdje podrazumijeva izravno iziskivanje sile i pomicanje čestica bez prolaženja kroz integracijski korak. Sile nastale sudarima su jedan primjer tih sila, ali uz to dolazi i dodatno sprječavanje rastezanja tkanine. Izbjegavanjem integracije tih takvih sila postižemo manje promjene sile koje su se pokazale problematične ako su prevelike ako se mijenjaju prebrzo u odnosu na vremenski korak Δt.

5.4 Detekcija sudara

Detekcija i rezolucija sudara je nešto što često zna stvarati probleme kod simulacije krutih tijela, u našem slučaju to je dosta pojednostavljeno jer koristimo infinitezimalno male čestice. U ovom potpoglavlju bit će pokriveni sudari samo sa sferama i ravninama, sličan koncept može se relativno jednostavno prilagoditi za bilo koju mrežu trokuta pri čemu je najteži dio za prilagoditi detekcija samog sudara.

5.4.1 Sfera

Pretpostavimo da u svijetu postoji sfera sa središtem u $\vec{c}$ i polumjerom r (radi smanjenja konfuzije, ovdje se neće spominjati rezolucija tkanine). Znamo da svaka točka koja je od središta udaljena manje od r nalazi se unutar sfere, ako se nalazi unutar sfere moramo riješiti sudar tako da točku u pitanju pomaknemo izvan sfere. Najlakši način odabrati u kojem smjeru pomaknuti točku je uzduž normale sfere koja se može odrediti sljedećom jednadžbom:

$$\hat{n} = \frac{\vec{x} - \vec{c}}{\|\vec{x} - \vec{c}\|} \tag{5.27}$$

Ako je uvjet da je točka unutar sfere ostvaren točku možemo pomaknuti vrlo jednostavno

$$\vec{x'} = \vec{x} + \hat{n}(r - \|\vec{x} - \vec{c}\|) \tag{5.28}$$

5.4.2 Ravnina

Za ravninu zadanu normalom $\hat{n}$ i točkom koja leži u ravnini $\vec{p}$ potrebno je odrediti slobodni član d:

$$d = -\hat{n} \cdot \vec{p} \tag{5.29}$$

Slobodni član predstavlja najkraću udaljenost ravnine od ishodišta u smjeru $-\hat{n}$ što može pomoći odgonetnuti udaljenost točke $\vec{x}$ od ravnine i s koje strane ravnine je točka

$$l = d - \vec{x} \cdot \hat{n} \tag{5.30}$$

U slučaju da je udaljenost od ravnine l negativna potrebno je pomaknuti točku x u smjeru suprotnom od normale za l kako više ne bi sudjelovala u sudaru:

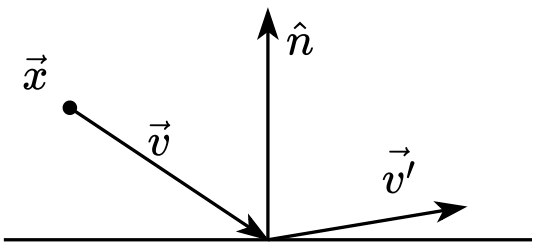
$$\vec{x'} = \vec{x} - l\hat{n} \tag{5.31}$$

5.4.3 Problem brzine

Rezolucija sudara još nije gotova jer u rubnim slučajevima tkanina može neočekivano eksplodirati. Unatoč tome što tkanina više ne sudjeluje u sudaru potencijalno i dalje ima brzinu u smjeru sudara te će idući simulacijski korak opet uzrokovati sudar i ako čestici interne sile ne promjene brzinu u suprotnom smjeru simulacija može postati nestabilna. Uvest ćemo transformaciju brzine koja će eliminirati dio ili cijelu komponentu brzine paralelnu normali površine sudara, na taj način čestica koja se gibala pod velikim kutom u odnosu na normalu će nastaviti kliziti. Vizualni prikaz transformacije nalazi se na slici 5.7 iz koje možemo vidjeti kako brzina zadržava komponentu paralelnu površini dok gubi dio komponente paralelnu normali. Ako uzmemo da je koeficijent apsorpcije komponente brzine paralelne normali $k_a \in [-1, 1]$ nastaje sljedeći izraz za brzinu čestice nakon sudara:

$$\vec{v'} = \vec{v} + k_a(\vec{v} \cdot \hat{n})\vec{v} \tag{5.32}$$

Veći koeficijent apsorpcije k_a znači da se tkanina neće jako odbijati pri sudaru što ima smisla za tekstil, međutim simulator može simulirati i npr. gumu koja se može deformirati jače se odbija prilikom sudara od tkanine pa bi za takvu primjenu imalo smisla koristiti negativnu vrijednost za k_a.



Slika 5.7 Transformacija brzine pri sudaru

5.5 Vjetar

Za vjetar se može reći da je klasična sila što se tiče simulacije tkanine. Gledati tkaninu kako vijori na vjetru i stvara valove na površini izgleda predobro da bi prošlo ispod radara. Za razliku od ostalih vanjskih sila iznos sile vjetra na česticu tkanine jedini još nije bio postavljen. Najviše zato što je pri implementaciji simulacije bitnija gravitacija buduće da je sveprisutna i cijelo vrijeme uzrokuje deformaciju tkanine bez koje bi vidjeli samo nepomični kvadrat. Vjetar će biti modeliran kao sila koja djeluje jednakom magnitudom i jednakim smjerom u svakom dijelu svijeta. Osim o jačini i smjeru, utjecaj sile vjetra na česticu ovisi i o njenoj orijentaciji. U stvarnosti vjetar ne može djelovati na infinitezimalnu česticu nego na neku površinu, ali budući da u korištenom modelu svaka čestica ima odgovarajuću orijentaciju vezanu za orijentacije trokuta čiji je dio koristit ćemo tu orijentaciju, odnosno normalu $\hat{n}$.

Ako imamo smjer vjetra $\hat{w}$, jačinu vjetra i_w i normalu čestice $\hat{n}$ dobijemo izraz sile koju vjetar iziskuje na česticu tkanine:

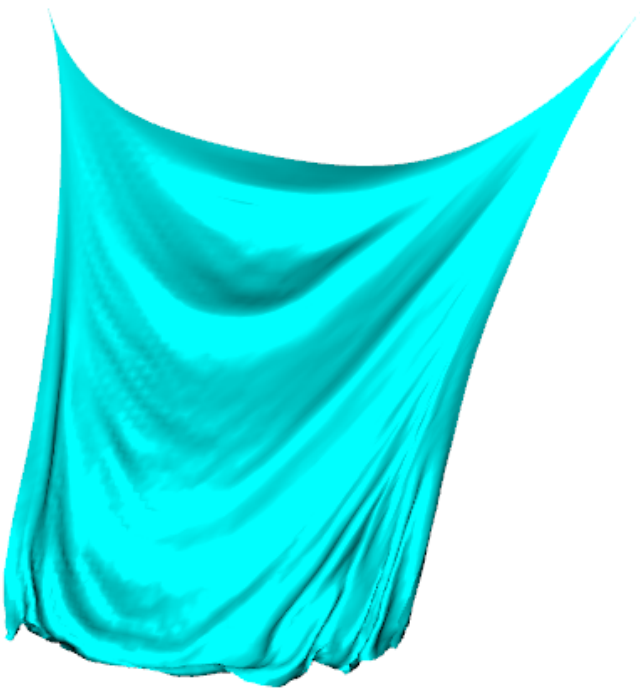
$$F_w(\hat{n}) = -i_w \hat{w}(\hat{w} \cdot \hat{n}) \tag{5.33}$$

Na prvi pogled čini se ispravno, ako namjestimo da vjetar puše od iza kamere prema tkanini dobit ćemo očekivani rezultat jer skalrni produkt $\hat{w} \cdot \hat{n}$ će biti negativan i onda se opet negira kako bi dobili isti smjer kao i $\hat{w}$. Međutim ako odlučimo obrnuti smjer vjetra tkanina će se pomicati u smjeru suprotnom od smjera vjetra jer će gore

navedeni skalarni produkt postati pozitivan te će se efektivno smjer vjetra obrnuti dva puta i dobit ćemo identičan rezultat. Potreban ispravak je uzimanje apsolutne vrijednosti skalarnog produkta $\hat{w} \cdot \hat{n}$ zbog čega je moguće ukloniti negaciju te konačni izraz za silu vjetra izgleda ovako:

$$F_w(\hat{n}) = i_w \hat{w} |\hat{w} \cdot \hat{n}| \tag{5.34}$$

Dodavanje vjetra značajno je povećalo vizualnu kvalitetu simulacije kao što se može vidjeti na slici 5.8.



Slika 5.8 Savijanje tkanine u valovima pod utjecajem vjetra

Poglavlje 6

Analiza rezultata i performanse

S gotovom i poprilično stabilnom simulacijom tkanine moguće je vršiti analizu ponašanja tkanine u određenim situacijama te u ovisnosti o fizičkim veličinama kao što su masa čestica, konstante opruga i snaga vjetra. Pri svakoj analizi razmatrat će se tkanine rezolucije $r = 64$, što je ujedno i maksimalna rezolucija koja se može odabrati unutar aplikacije, radi prikaza najveće razine kvalitete kroz slike.

Osim uz pomoć grafičkog korisničkog sučelja (eng. graphical user interface, GUI) moguće se kretati po virtualnom svijetu korištenjem dodatnih kontrola. Pritiskom lijeve tipke na mišu izvan GUI-a moguće je rotirati se (orbitirati) oko ciljane točke pogleda (*View target* polje unutar GUI-a) pomicanjem miša. Pritiskom desne tipke na mišu izvan GUI-a te pomicanjem miša moguće je približavati ili udaljavati se od ciljane točke pogleda. Držanjem tipke *Shift* i pritiskom lijeve tipke miša u blizini neke od čestica tkanine moguće je „uhvatiti" česticu te pomicanjem miša pomicati ju po ravnini paralelnoj s projekcijskom ravninom. Zadnja mogućnost je fiksiranje neke čestice u prostoru držanjem tipke *Alt* i pritiskom lijeve tipke miša u blizini čestice tkanine. Navedene kontrole omogućuju veću razinu interaktivnosti te dodavanje vanjskih sila koje nisu predvidive i fizički striktno definirane.

Za početak će se razmatrati osnovno ponašanje tkanine mijenjanjem raznih fizičkih parametara unutar okoline bez ikakvih drugih objekata tako da ne bude nikakvih sudara. Na slici 6.1 može se vidjeti kako tkanina izgleda u ravnotežnom položaju za različite vrijednosti mase čestica m.

(a) $m = 0.5$ (b) $m = 1$ (c) $m = 1.5$

Slika 6.1 Tkanina u ravnotežnom položaju za različite vrijednosti mase čestica m

Rezultat je kao i očekivan, kako raste masa čestica raste i gravitacijska sila koju opruge ne mogu savladati bez povećanja njihovih konstanti te je rezultat tkanina koja je sve više razvučena. Potvrdu navedene tvrdnje moguće je vidjeti na slici 6.2.

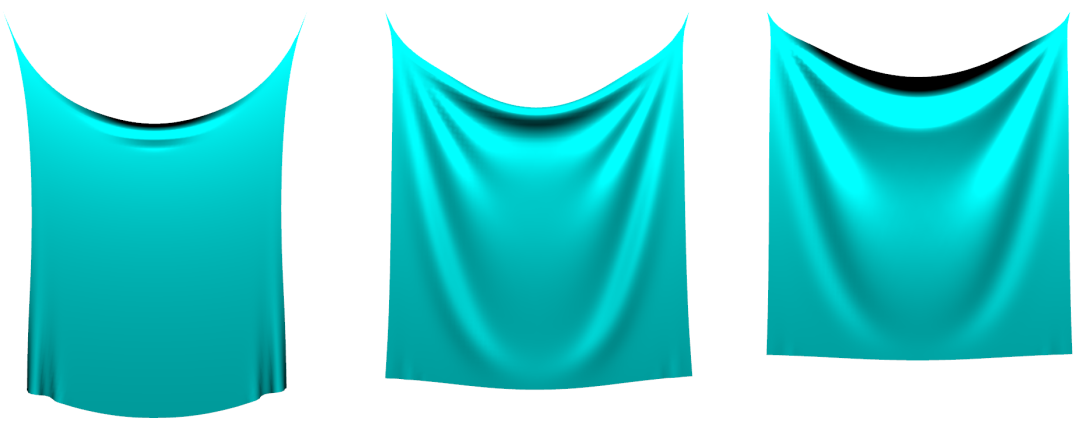
(a) $m = 0.5$ (b) $m = 1$ (c) $m = 1.5$

Slika 6.2 Djelovanje sile na čestice tkanine u ravnotežnom položaju za različite vrijednosti mase čestica m

Boje na slici iznad predstavljaju magnitudu sile na određenu česticu, pri određivanju boje uzimaju se najmanja i najveća magnituda sile među svim česticama što

se transformira u raspon boja od zelene do crvene gdje zelena predstavlja najmanju magnitudu sile, a crvena najveću. Boje između kao što su žuta i narančasta predstavljaju neku magnitudu sile između najmanje i najveće pri čemu narančasta predstavlja jaču magnitudu od žute. Budući da su krajnja gornja lijeva i desna čestica fiksirane (rezultirajuće sile koje djeluju na njih su jednake 0) one moraju „nositi" ostatak tkanine zbog čega su sile najviše magnitude u okolici tih točaka dok je ostatak tkanine praktički u ekvilibriju.

Rastezanje tkanine moguće je smanjiti povećanjem konstanti opruga, za tkaninu mase čestica $m = 1$ uzet ćemo nekoliko različitih vrijednosti za konstante svakog tipa opruga. Rezultat promjena nalazi se na slici 6.3.

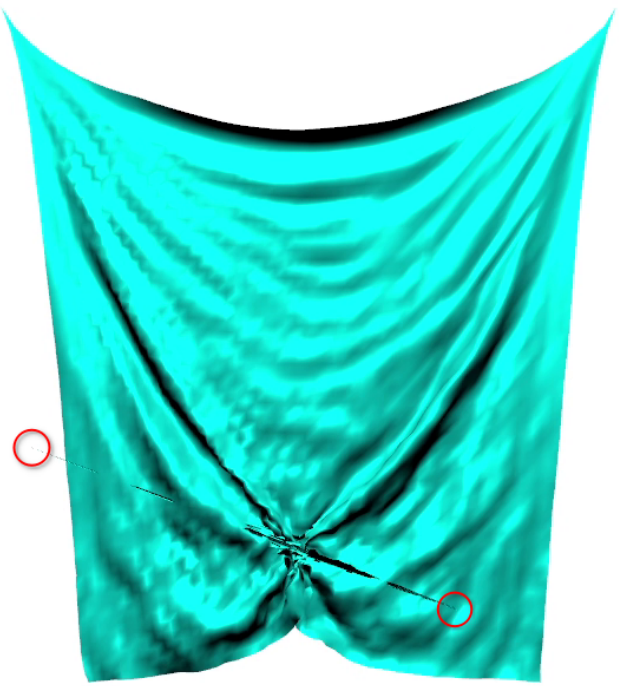


(a) $k_m = 2, k_s = 1, k_b = 0.5$ (b) $k_m = 50, k_s = 20, k_b = 8$ (c) $k_m = 200, k_s = 75, k_b = 25$

Slika 6.3 Tkanina u ravnotežnom položaju za različite vrijednosti konstanti opruga

U lijevoj slici gdje su najniže konstante opruga tkaninu na okupu drži ograničenje duljine opruga koje je tu zbog postizanja veće razine stabilnosti simulacije. Opruge same po sebi su preslabe za tako veliku masu čestica te bez ograničenja duljine tkanina bi se iznimno jako rastegnula dok se sila zbog velike udaljenosti između čestica ne bi dovoljno povećala da zaustavi rastezanje. U druge dvije slike vidi se kako to više nije slučaj, tkanina je značajno manje rastegnuta i zbog većih sila koje iziskuju pojavljuju se bore u smjeru od sredine prema fiksiranim točkama. Pažljivim promatranjem tkanine s parametrima kao na desnoj slici mogu se primijetiti mikrooscilacije na po-

vršini tkanine koje su rezultat visokih konstanti opruga. Te mikrooscilacije drže se pod kontrolom jer postoji prigušenje i magnituda sila nije dovoljno velika da bi uzrokovala rast amplituda tih oscilacija. Međutim kad su opruge tako napete, smanjenje mase ili hvatanje i brzo pomicanje čestica može vrlo lako uvesti lokalnu nestabilnost koja se može dalje proširiti na ostatak tkanine ili čak uzrokovati eksploziju tkanine u beskonačnost (koju uglavnom uzrokuje premala masa čestica). Na slici 6.4 prikazano je kako izgleda kad dođe do lokalne nestabilnosti tkanine.



Slika 6.4 Lokalna nestabilnost tkanine

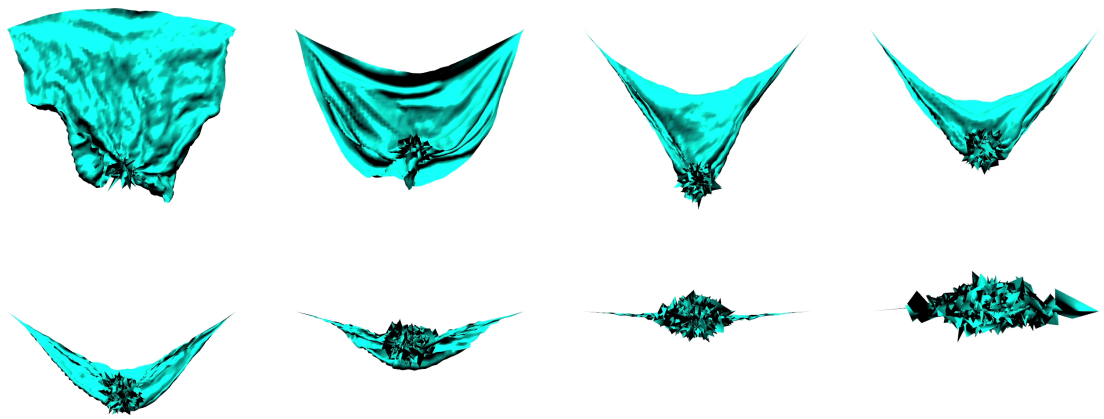
Označeno crvenim kružićima su čestice koje osciliraju oko ravnotežnog položaja koji se nalazi oko jako zgužvanog dijela tkanine otprilike na pola puta između te dvije čestice. Lokalna nestabilnost nastaje najčešće prilikom ručnog hvatanja i pomicanja čestice tkanine daleko od njenog ravnotežnog položaja te naknadnog otpuštanja iste. Budući da je čestica daleko od svog ravnotežnog položaja, čim se otpusti velike sile će početi djelovati na nju kako bi ju dovele u ravnotežni položaj te je moguće da pri tim velikim silama počne oscilirati zbog velikih promjena pozicije u jednom simulacij-

skom koraku koji uzrokuju naglo mijenjanje smjerova sile te oscilaciju. Te oscilacije mogu uzrokovati povećanje magnitude sile na susjedne čestice što može dovesti neku od tih susjednih čestica do oscilacije nakon čega će obje čestice oscilirati oko novog ravnotežnog položaja koji će biti u blizini vrlo zgužvanog dijela tkanine nastalog zbog navedenih oscilacija. Postoji nekoliko načina na koje je moguće eliminirati lokalnu nestabilnost:

- povećanjem mase čestica m
- smanjenjem konstanti opruga k_m, k_s, k_b
- povećanjem konstante prigušenja k_d
- smanjenjem vremenskog koraka Δt
- hvatanjem jedne od oscilirajućih čestica i pomicanje u blizinu ravnotežnog položaja

Prva tri načina su jednostavna za realizirati, no mijenjaju fizičke karakteristike te izgled tkanine pa možda nisu najbolje rješenje. Smanjenje vremenskog koraka poželjna je opcija ako je postignuta zadovoljavajuća razina vizualne kvalitete te nije problem žrtvovati nešto performansi za veću razinu stabilnosti. Zadnji način može biti vrlo težak zbog velike brzine oscilacije čestica, a i za razliku od ostalih načina ne sprječava ponovno pojavljanje lokalne nestabilnosti.

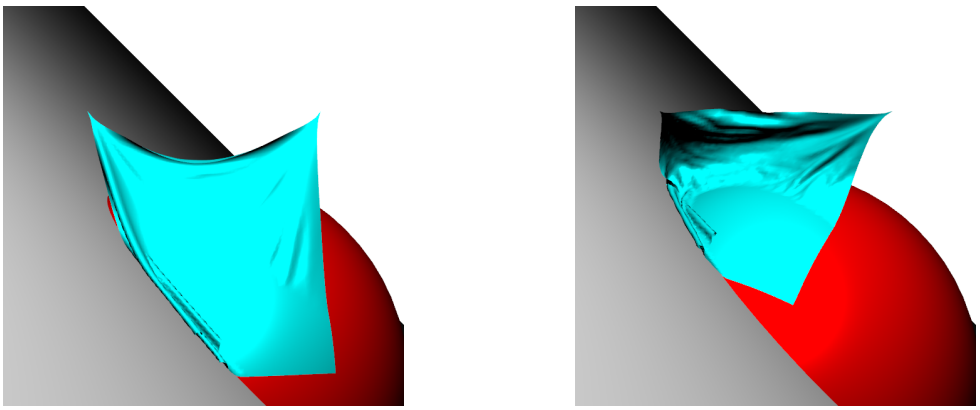
Ako su parametri namješteni tako da je tkanina na rubu stabilnosti (mala masa čestica, visoke konstante opruga, velik vremenski korak), veća promjena pozicije čestice može uzrokovati nastanak lokalne nestabilnosti koja se zbog niske otpornosti na nestabilnost može postepeno proširiti tako da obuhvati cijelu tkaninu pri čemu nastaje globalna nestabilnost. Ta globalna nestabilnost može uzrokovati eksploziju tkanine u beskonačnost, ali može ostati samo kako kaotična beskonačna oscilacija nefiksiranih čestica tkanine, ovisi o parametrima. Slika 6.5 pokazuje postepeno nastajanje globalne nestabilnosti tkanine.



Slika 6.5 Postepeno nastajanje globalne nestabilnosti iz lokalne nestabilnosti tkanine

6.1 Scenarij 1

Dosad je tkanina bila promatrana u izolaciji, odnosno nije bilo drugih objekata u virtualnom svijetu s kojima bi tkanina mogla imati bilo kakve interakcije. Ovaj scenarij vrlo je sličan dosadašnjem, samo što je dodana jedna kugla i kosina. Nakon učitavanja scenarija *sphere_rest.json* vidi se kako tkanina pada na kuglu i lagano klizi dok se ne izravna. Nakon što se tkanina izravna i dođe do ravnotežnog položaja moguće je povećati snagu vjetra i vidjeti kako kugla ne dozvoljava vjetru da lako pomiče tkaninu od ravnotežnog položaja te da ju prebaci na drugu stranu kugle što fizički ima smisla i očekivano je. Na slici 6.6 vidi se tkanina u ravnotežnom položaju pod laganim utjecajem vjetra te što se dogodi kad iz ravnotežnog položaja povećamo snagu vjetra preko skoro peterostruko.



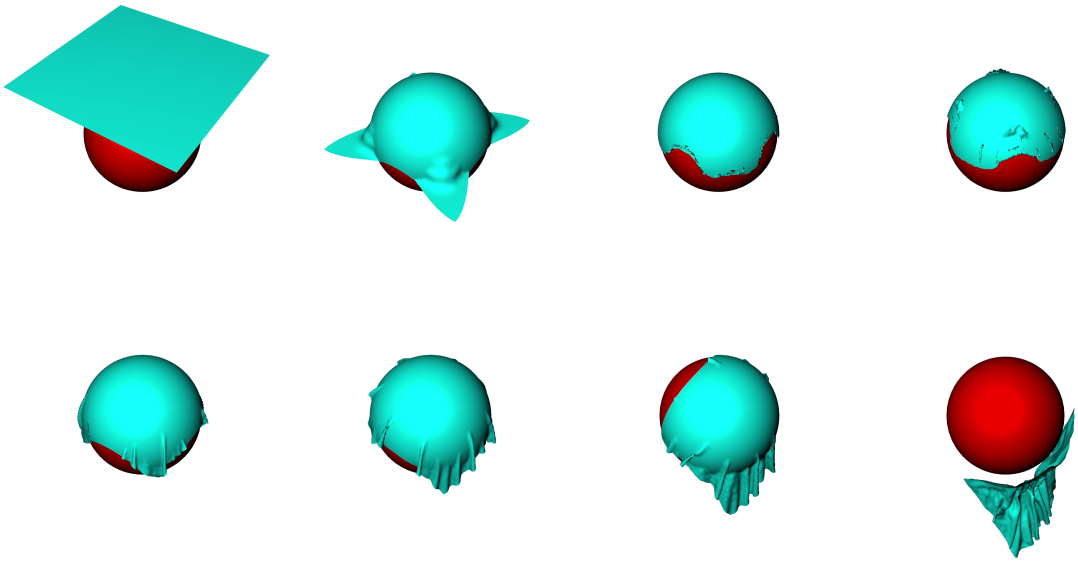
(a) $i_w = 4$ (b) $i_w = 19.5$

Slika 6.6 Tkanina pod utjecajem slabog i jakog vjetra

Na lijevoj slici tkanina je naslonjena na kuglu i kosinu što se može vidjeti iz blago poprimanja oblika kugle na donje desne strane i nakošenosti na donjoj lijevoj strani. Na desnoj slici bolje se vidi interakcija s kuglom zbog jačeg vjetra koji gura tkaninu prema kugli, također kao i kod prijašnjeg prikaza utjecaja vjetra može se vidjeti stvaranje valova na tkanini. Dok je tkanina pod utjecajem vjetra također je lakše moguće uočiti inerciju pri većoj masi čestica kao sporije pomicanje rubova tkanine koji nisu povezani s drugim česticama u svim smjerovima. Jedna bitna stvar za spomenuti je kako donji lijevi dio tkanine ne izgleda vizualno ispravno, razlog tome je što u sklopu ovog rada nije implementirano detektiranje i rješavanje sudara tkanine sa samom sobom pa umjesto da se tkanina savija i slaže u slojevima jedan na drugom čestice mogu prolaziti kroz površinu tkanine.

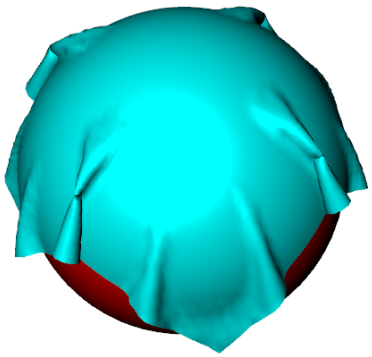
6.2 Scenarij 2

U ovom scenariju (*sphere_fall.json*) nijedna čestica tkanine nije fiksirana u prostoru, već kao što ime predlaže slobodno pada na kuglu u sredini. Slika 6.7 prikazuje slobodni pad tkanine na kuglu, zadržavanje neko vrijeme na kugli te lagano klizanje s iste i pad u beskonačnost.



Slika 6.7 Pad tkanine na kuglu i klizanje s kugle

Kako tkanina dolazi u kontakt s kuglom vrhovi tkanine i dalje ubrzavaju pri padu dok ih čestice tkanine koje su u kontaktu s kuglom ne povuku prema gore. To je uzrokovano niskim konstantama opruga, povećavanjem konstanti opruga smanjuje se magnituda skoka vrhova, ali to ostavlja vizualno nezadovoljavajuće rezultate kao što se može vidjeti na slici 6.8. Tkanina se neprirodno savija i neke čestice prolaze kroz površinu tkanine što bi se moglo ispraviti rješavanjem sudara tkanine sa samom sobom ili povećanjem radijusa kugle. Osim toga ako pogledamo ostatak animacije, izgleda kao što bi se moglo očekivati. Unatoč deterministočnosti simulacije, jedan vrh uvijek uspije povući tkaninu na način da krene kliziti s kugle u nekom smjeru koji nije svaki put isti. Prilikom klizanja tkanina se počinje gužvati po rubovima i dijelovima koji nisu u kontaktu s kuglom. Nakon što nijedna čestica tkanine nije više u kontaktu s kuglom zadnji dio tkanine koji je bio u kontaktu s kuglom i dalje ima dio horizontalne komponente brzine te se počinje savijati prema donjem dijelu.



Slika 6.8 Neprirodno savijanje tkanine uzrokovano visokim konstantama opruga

Osim problema prolaženja čestica kroz površinu tkanine i savijanja pri visokim konstantama opruga tkanina izgleda vizualno zadovoljavajuće i relativno realistično. Uz primjenu visokih konstanti opruga i minimalne mase koju je moguće postaviti kroz GUI sadržava potpunu stabilnost u slučaju da korisnik ne pomiče čestice daleko od ravnotežnog položaja.

6.3 Performanse

Pri izradi programskog rješenja velika količina vremena uložena je u poboljšanje performansi simulacije. Od eksplicitne integracije koja je zahtijevala vrlo male vremenske korake i svejedno nije bila dovoljno stabilna do implementacije implicitne integracije koja je mnogo stabilnija, ali nosi sa sobom puno kompleksniju matematičku formulaciju. Povećanje vremenskog koraka Δt uz korištenje implicitne integracije omogućilo je vraćanje jednog dijela performansi, ali na kraju povećana kompleksnost algoritma poništava to što je dobiveno povećanjem vremenskog koraka.

Performanse koje su potrebne za rad u realnom vremenu postignute su paralelizacijom obrade podataka korištenjem više jezgri procesora. Budući da suvremeni

procesori imaju uglavnom četiri ili više jezgri, takvo rješenje je imalo najviše smisla. Aplikaciji je dozvoljeno korištenje maksimalno 8 dretvi (jezgri procesora) pri čemu se dobije veliko ubrzanje svakog simulacijskog koraka. Budući da algoritam za računanje sila i derivacija ima određenu količinu dijeljene memorije, zahtijeva sinkronizaciju dretvi te se empirijski pokazalo da dodjeljivanje više dretvi za izvršavanje algoritma ga neće nužno ubrzati, a u najgorem slučaju može ga i usporiti.

Mjerenje performansi vršeno je na računalu s AMD Ryzen 9 7950X procesorom ograničenim na nominalni radni takt od 4.5 GHz, 128 GB DDR5 radne memorije s radnim taktom od 6 GHz te Nvidia GeForce RTX 4090 grafičkom karticom. Računalo pokreće Arch distribuciju Linux operacijskog sustava s verzijom jezgre (eng. kernel) 6.10.6. Program je kompajliran koristeći kompajler *clang* verzije 18.1.8 uz maksimalnu razinu optimizacije i prilagođen Zen 4 arhitekturi procesora (zastavice *-O3* i *-march=native*).

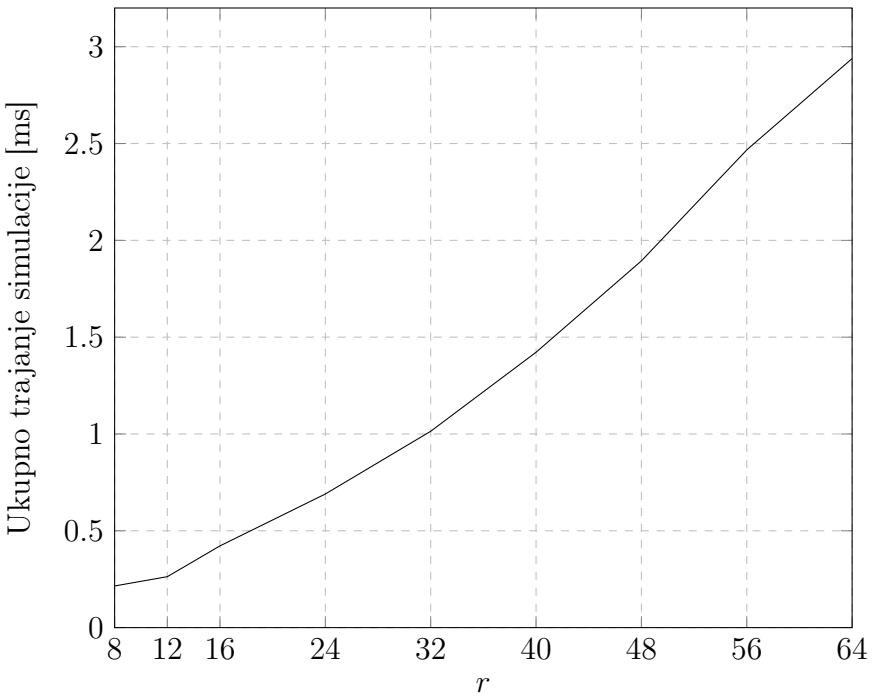
Sljedeća tablica sadrži prosječna trajanja određenih dijelova simulacijskog algoritma za odgovarajuću rezoluciju tkanine r. Vremena su izražena u milisekundama te grupirana za sve simulacijske korake koji se dogode za svaki prikazani frame, odnosno prosjek trajanja svih dijelova simulacije koji čine trenutni frame.

r	**Ukupno**	**Sile**	**Integracija**	**Sudari**	**Ograničenje širenja**	**Normale**
8	0.215	0.094	0.102	0.003	0.010	0.006
12	0.263	0.115	0.110	0.005	0.020	0.013
16	0.422	0.202	0.148	0.008	0.038	0.025
24	0.690	0.341	0.195	0.016	0.087	0.051
32	1.014	0.477	0.268	0.026	0.157	0.086
40	1.422	0.645	0.362	0.042	0.241	0.132
48	1.894	0.829	0.471	0.058	0.342	0.193
56	2.467	1.063	0.593	0.082	0.475	0.254
64	2.939	1.169	0.691	0.109	0.636	0.333

Tablica 6.1 Performanse simulacije

Na testnom računalu korišten je ekrana frekvencije osvježavanja 165 Hz što znači da svaki frame mora biti spreman za prikaz unutar manje od 6.06 ms. Iz tablice se vidi da za $r = 64$ trajanje simulacije je manje od pola toga vremena što znači da je

moguće dodatno povećati rezoluciju tkanine za postizanje veće vizualne kvalitete ili smanjiti vremenski korak Δt kako bi se postigla veća razina stabilnosti simulacije.



Slika 6.9 Graf ukupnog trajanja simulacije u odnosu na r

Na slici 6.9 prikazana je ovisnost ukupnog trajanja simulacije o rezoluciji tkanine. Slika međutim prikazuje neočekivanu anomaliju u podacima, a to je da ukupno trajanje simulacije raste otprilike linearno s povećanjem rezolucije tkanine što logički nema smisla budući da broj čestica tkanine raste s kvadratom rezolucije. Ako pogledamo zadnja tri stupca s vremenima u tablici 6.1 možemo primjetiti tu kvadratnu relaciju s rezolucijom tkanine npr. trajanje izračuna normala za $r = 64$ je 4 puta duže od onog za $r = 32$ jer $(\frac{64}{32})^2 = 4$. Iznimke tog pravila su trajanje integracije i izračuna sila, a jedini očiti razlog zašto se takvo nešto događa je korištenje paralelizacije za ta dva koraka u odnosu na ostale. Pretpostavka je da neke dretve završe posao značajno ranije od drugih te se time ne realizira potpuni potencijal svih dretvi. Jedan način na koji se može povećati iskoristivost hardvera je povećanje rezolucije tkanine što znači da će dretve imati više posla pa ako neke dretve ranije završe bit će manji udio vremena čekanja da ostale dretve završe i time više vremena biti aktivne.

Poglavlje 7

Zaključak

Cilj ovog diplomskog rada bio je realizirati simulaciju tkanine koristeći model masa i opruga te prikazati rezultate simulacije kroz interaktivnu simulaciju trodimenzionalnog virtualnog svijeta.

Objašnjen je cijeli postupak prikaza tkanine kao objekt u trodimenzionalnom svijetu. Obrađeno je sve od reprezentacije tkanine kao mreže trokuta, definiranja površine tkanine, transformacije koordinatnih sustava koji omogućuju prikazivanje tkanine na dvodimenzionalnom ekranu do procesa osvjetljenja i sjenčanja.

Prvo je predložen način simulacije korištenjem eksplicitne integracije čija je velika prednost iznimno jednostavna i efikasna implementacija. Međutim, zbog načina na koji funkcionira uvodi se nepostojeća energija u sustav što uzrokuje nestabilnost simulacije koja se manifestira kao oscilacije s povećavajućom amplitudom te završava eksplozijom tkanine u beskonačnost. Zbog navedenog problema bilo je potrebno smanjiti simulacijski korak do razine pri kojoj se gubi korist jednostavne i efikasne implementacije.

Krajnja implementacija koristi implicitnu integraciju kao način napredovanja simulacije kroz vrijeme. Implicitna integracija zahtijeva značajno kompleksniji algoritam kako bi funkcionirala, ali zbog svojstva pesimizacije energije sustava ostaje stabilna uz puno zahtjevnije parametre simulacije kao što su masa čestica, konstante opruga i vremenski korak. Problem s performansama, kojeg uzrokuje kompleksnost implicitne integracije, riješen je korištenjem više dretvi, odnosno fizičkih jezgri pro-

cesora pri izvršavanju algoritma.

Analizirano je kako se ponaša tkanina kada se mijenjaju određeni parametri simulacije kao što su masa čestica i konstante opruga te kako se ponaša u dva scenarija gdje ima interakcije s drugim objektima kao što su kugla i ravnina. Uz određene iznimke u drugom scenariju, tkanina se ponašala na vizualno zadovoljavajuć i većinskim dijelom uvjerljiv način.

Glavno područje gdje bi se simulacija mogla poboljšati je detekcija i rješavanje sudara tkanine sa samom sobom. Značajno bi se povećala vizualna kvaliteta i realizam kad bi se tkanina savijala i tvorila slojeve umjesto urušavanja u sebe kao da nema volumen. Unatoč tome interakcija tkanine sa samom sobom nije bila implementirana unutar ovog rada zbog velike razine kompleksnosti, utjecaja na stabilnost simulacije i potencijalno velikog udara na performanse simulacije koje su se predstavile kao problem i bez ove značajke.

Bibliografija

[1] N. Koliha, C. Janßen, and T. Rung, "Towards online visualization and interactive monitoring of real-time cfd simulations on commodity hardware," *Computation*, vol. 3, pp. 444–478, 09 2015.

[2] OpenGL tutorial. LearnOpenGL. , s Interneta, https://learnopengl.com

[3] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *SIGGRAPH Comput. Graph.*, vol. 11, no. 2, srpanj 1977. , s Interneta, https://doi.org/10.1145/965141.563893

[4] T. Stuyck, *Cloth Simulation for Computer Graphics*, ser. Synthesis Lectures on Visual Computing. Morgan & Claypool Publishers, 2018. , s Interneta, https://doi.org/10.2200/S00867ED1V01Y201807VCP032

[5] D. Baraff and A. P. Witkin, "Large steps in cloth simulation," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998. , s Interneta, https://api.semanticscholar.org/CorpusID:326544

[6] CMake dokumentacija. , s Interneta, https://cmake.org/documentation

[7] OpenGL dokumentacija. Khronos. , s Interneta, https://registry.khronos.org/OpenGL-Refpages/gl4

[8] GLFW dokumentacija. , s Interneta, https://www.glfw.org/documentation

[9] GLM dokumentacija. , s Interneta, https://glm.g-truc.net/0.9.9/api/index.html

[10] Dear ImGui online priručnik. , s Interneta, https://pthom.github.io/imgui_manual_online/manual/imgui_manual.html

[11] Eigen dokumentacija. , s Interneta, https://eigen.tuxfamily.org/dox

[12] Nlohmann JSON dokumentacija. , s Interneta, https://json.nlohmann.me/api/basic_json

Sažetak

Ovaj rad obrađuje simulaciju tkanine u području računalne grafike s naglaskom na vizualnu uvjerljivost umjesto fizičke točnosti. Realizirana je simulacija tkanine temeljena na modelu masa i opruga te implicitnoj integraciji koja za prikaz trodimenzionalnog virtualnog prostora koristi OpenGL. Rad objašnjava sve potrebne koncepte za prikaz trodimenzionalne grafike korištenjem modernog grafičkog protočnog sustava te ideje korištene za realizaciju simulacije sustava masa i opruga.

Ključne riječi **— simulacija tkanine, animacija, računalna grafika, OpenGL, model masa i opruga, implicitna integracija**

Abstract

This thesis deals with cloth simulation in the field of computer graphics while emphasizing visual quality over physical accuracy. A cloth simulation was realized based on the mass spring model and implicit integration which uses OpenGL for rendering of the three-dimensional virtual space. The thesis explains all the necessary concepts needed for rendering three-dimensional graphics using the modern graphics pipeline and ideas used for the realization of a mass spring simulation system.

Keywords **— cloth simulation, animation, computer graphics, OpenGL, mass spring model, implicit integration**