SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 392

FIZIKALNO TEMELJEN MODEL LJUDSKE KOSE

Luka Samac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 392

FIZIKALNO TEMELJEN MODEL LJUDSKE KOSE

Luka Samac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 392

Pristupnik:	**Luka Samac (0036524776)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentorica:	prof. dr. sc. Željka Mihajlović

Zadatak: **Fizikalno temeljen model ljudske kose**

Opis zadatka:

Proučiti osnovne fizikalne komponente potrebne za razvoj modela ljudske kose. Razraditi matematički model za oblikovanje simulacije kose te pripadne vizualizacijske komponente za prikaz simulacije. Implementirati razrađeni model te ostvariti fizikalno temeljenu simulaciju uz pripadnu vizualizaciju. Načiniti testiranje na nizu primjera. Analizirati i ocijeniti ostvarene rezultate. Diskutirati upotrebljivost ostvarenih rezultata kao i moguća proširenja. Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Simulacija ljudske kose je čest, no zahtijevan problem u računalnoj grafici. Zbog računalne zahtjevnosti simulacije, korištenje fizikalnih modela ljudske kose bilo je ograničeno na filmsku industriju i znanstvene radove, dok je simulacija u stvarnom vremenu koristila pojednostavljene modele.

S poboljšanjem grafičkih kartica te rasta popularnosti videoigara, porasla je potreba za metodama fizikalnim simulacijama kose u stvarnom vremenu. Koristeći trenutne metode simulacije, moguće je postići razumnu kvalitetu simulacije kose bez gubitka performansi cjelokupnog programa. Na Sl. 0.1 Simulacija kose u Unreal Engine-u.

Sl. 0.1 Simulacija kose u Unreal Engine-u [9]

Prosječna ljudska glava ima preko 100 tisuća dlaka kose. Zbog toga, model ljudske kose predstavlja izazov u učinkovitom prikazu i simulaciji. U svrhu smanjenja računske složenosti, često se ne simuliraju sve dlake kose, nego se simulira dio modela, a ostatak se dobije interpolacijom.

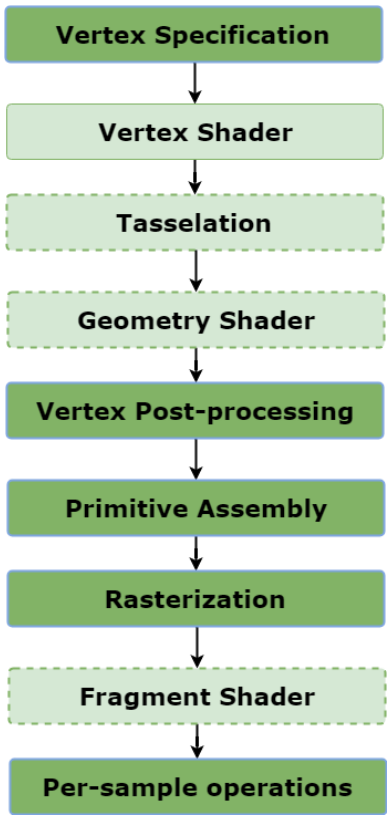
U računalnoj grafici, dlaka kose je često prikazana krivuljom koja ima određenu širinu. Pri tome se zanemaruje oblik valjka koje dlake kose imaju.

1. Grafički protočni sustav

Prije opisa samog modela ljudske kose, potrebno je razumjeti grafički protočni sustav te svrhu pojedinih dijelova protočnog sustava pri simulaciji kose.

Grafički protočni sustav je niz koraka koje OpenGL obavlja kako bi prikazao objekte. Ovaj proces se sastoje od 9 koraka, kao što je prikazano u Sl. 1.1 Grafički protočni sustav u OpenGL-u.1. Mnoge od ovih koraka je moguće programirati te je neke moguće i preskočiti. Koraci koji se ne mogu programirati su prikazani tamno zelenom bojom.

Za potrebe ovog rada, detaljnije će biti objašnjeni koraci koji se mogu programirati jer pisanje vlastitih sjenčara je bitan dio simulacije kose.



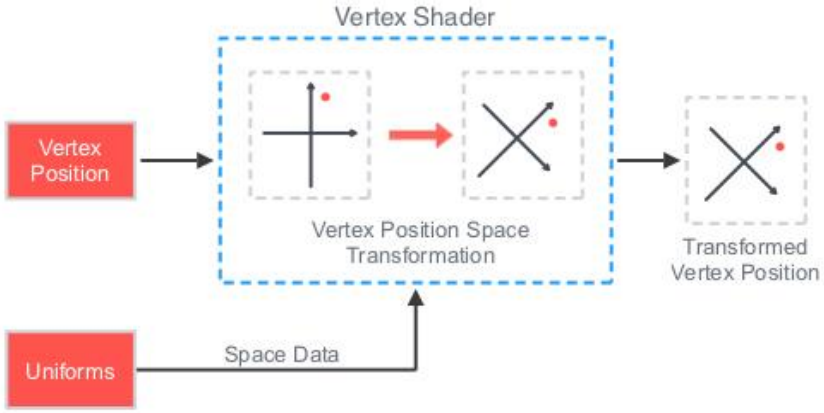
Sl. 1.1 Grafički protočni sustav u OpenGL-u [7]

1.1. Sjenčar vrhova

Sjenčar vrhova je program pisan u GLSL (OpenGL Shading Language) koji obavlja operacije nad vrhovima (operacije per-vertex). Ovaj program se izvodi jednom za svaki vrh koji grafička kartica obrađuje.

Glavna uloga sjenčara vrhova je da izračuna konačnu poziciju vrhova u sceni primjenom geometrijskih transformacija na pozicije vrhova s obzirom na položaj kamere. Ovaj proces je prikazan na Sl. 1.2 Sjenčar vrhova.

Nakon izračuna konačnih pozicija vrhova, atributi vrha se prosljeđuju dalje u protočni sustav.

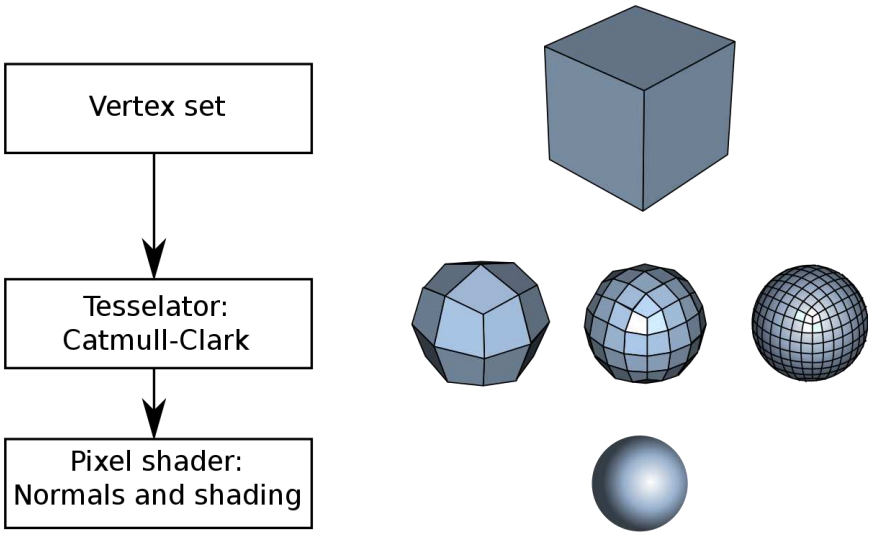


Sl. 1.2 Sjenčar vrhova

1.2. Sjenčar teselacije

Teselacija je neobavezan korak protočnog sustava. Sastoji se od dva sjenčara (Tessellation Control Shader i Tessellation Evaluation Shader) te fiksne funkcije teselatora između njih. Osnovna uloga teselacije je podjela trokuta u detaljnije mreže trokuta, prikazano na Sl. 1.3 Teselacija.

Pri simulaciji kose, teselacija će se koristiti za interpolaciju kose ovisno o gustoći kose. Fizikalna simulacije će se izvoditi samo na nekolicini dlaka, dok će se ostale dlake dobiti interpolacijom. Interpolacijom je moguće smanjiti količinu resursa potrebnih za izvođenje simulacije, bez velikog gubitka na kvaliteti simulacije.



Sl. 1.3 Teselacija primjenom Catmull-Clark metode

1.2.1. Sjenčar kontrole teselacije

Sjenčar kontrole teselacije je prvi sjenčar koji se izvodi u procesu teselacije. Sjenčar na ulazu dobije krpicu (engl. patch)“ – primitivu koja je skup vrhova bez implicirane topologije. Glavna uloga sjenčara kontrole teselacije je da pošalje razinu teselacije teselatoru te da proslijedi podatke o krpici sjenčaru evaluacije teselacije.

1.2.2. Sjenčar evaluacije teselacije

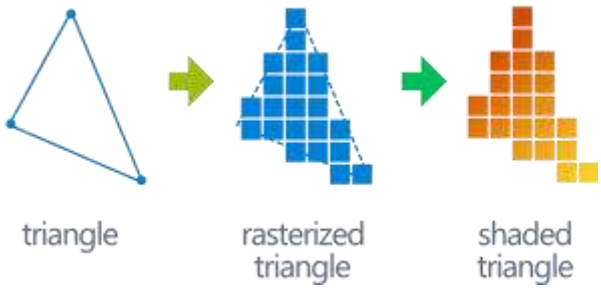
Sjenčar evaluacije teselacije je sjenčar koji se izvodi zadnji u procesu teselacije, poslije fiksne funkcije teselatora. Kod ovog sjenčara se izvodi barem jednom za svaki finalni vrh. Glavna uloga ovog sjenčara je da izračuna interpolirane pozicije svakog vrha te ostale potrebne atribute vrha. Podatci o vrhovima se prosljeđuju dalje u sljedeći korak protočnog sustava.

1.3. Sjenčar geometrije

Sjenčar geometrije je neobavezan korak u grafičkom protočnom sustavu. Ovaj sjenčar na ulazu dobije primitive te vraća nula ili više izlaznih primitiva. Izlazne primitive mogu bili različite od ulaznih primitiva. Svrha sječara geometrije u simulaciju kose je da pretvori linije dlake kose u pravokutnike koje gledaju prema kameri. Prednosti prikazivanja dlake kose kao skup pravokutnika naspram linija će biti raspravljene u sljedećim poglavljima.

1.4. Sjenčar fragmenata

Sjenčar fragmenata je program koji se izvodi nakon što su primitive rasterizirane. Za svaki piksel koji je pokriven primitivom, stvara se „fragment“. Sjenčar fragmenata na ulaz dobiva jedan fragment. Svrha sjenčara fragmenata je odrediti konačnu boju tog fragmenta, kao što je prikazano na Sl. 1.4 Sjenčanje fragmenata. U sjenčaru fragmenata je moguće računati osvjetljenje po pikselu te preslikavanje tekstura ovisno o *uv* koordinatama fragmenta u teksturi.



Sl. 1.4 Sjenčanje fragmenata

1.5. Sjenčar računanja

Sjenčar računanja (engl. compute shader) je poseban sjenčar jer za razliku od svih ostalih sjenčara, ne koristi se za prikazivanje objekata i crtanje trokuta i piksela. Ovaj sjenčar se koristi za računanje proizvoljnih podataka na grafičkoj kartici.

Prostor u kojem sjenčar radi je apstraktan te je na sjenčaru da definira značenje tog prostora. Najbitnije od svega, ovaj sjenčar nema ulaze koje korisnik može definirati te nema izlaza. Zbog toga, ako ovaj sjenčar želi uzeti neke vrijednosti kao ulaz, mora se sam pobrinuti da dohvati te podatke. Dohvat tih podataka može biti preko raznih sučelja koje OpenGL pruža, poput dohvata teksture ili učitavanje proizvoljne slike.

Unutar sjenčara računanja, odvijati će se fizikalna simulacija kose i vjetra jer je računanje podataka na grafičkoj kartici znatno brže nego na procesoru.

2. Prikazivanje kose

Ljudska kosa, osobito kada je duga, može imati komplicirane geometrijske strukture. Osoba može imati preko 100 tisuća dlaka kose. Zbog ovih razloga, prikazivanje te simulacija ljudske kose predstavlja izazov za aplikacije u stvarnom vremenu.

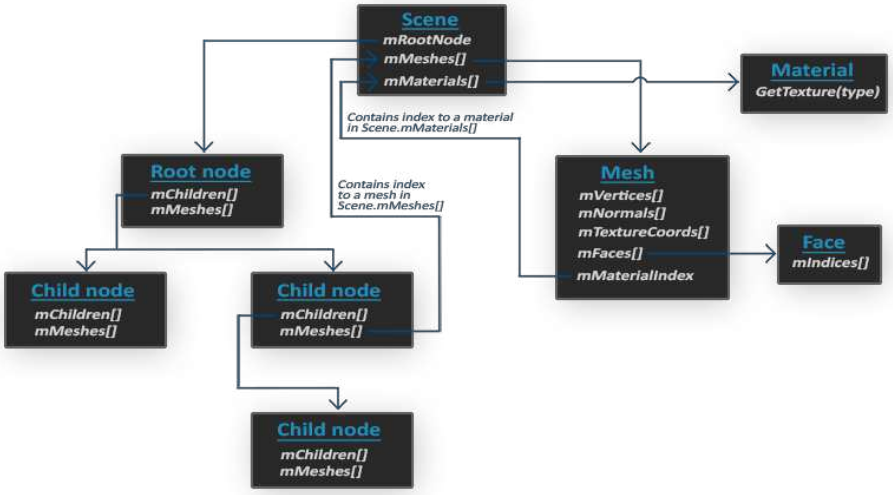
U ovom poglavlju, efikasno prikazivanje ljudske kose će biti opisano. Cilj ovog rada je prikazivanje i simulacija ljudske kose u stvarnom vremenu, no spomenut će se i druge metode koje nisu nužno namijenjene za aplikacije u stvarnom vremenu.

2.1. Učitavanje modela

OpenGL ne nudi gotove metode za učitavanje objekata iz datoteka. Iz toga razloga, potrebno je implementirati vlastite metode za učitavanje objekata koristeći C++ standardnu biblioteku ili koristiti neke od OpenGL biblioteka koje su namijenjene za učitavanje objekata.

Čest problem pri implementaciji vlastitih metoda za učitavanje objekata iz datoteka su razni formati zapisa objekata. Većinom ne postoji općenita struktura među formatima te svaki format zapisuje podatke na svoj način. Zbog ovog razloga, preporučeno je koristiti biblioteke za učitavanje objekata.

Assimp je OpenGL biblioteka koja je namijenjena učitavaju objekata. Ova biblioteka podržava razne formate te omogućava apstraktan pristup podatcima preko sučelja koje je neovisno o formatu zapisa. Jednostavna struktura Assimp biblioteke je prikazan na Sl. 2.1 Struktura Assimp-a.



Sl. 2.1 Struktura Assimp-a [7]

2.2. Prikazivanje dlaka kose

Na prvi pogled, prikazivanje dlake kose može izgledati jednostavno koristeći linije. OpegGL nudi primitivu GL_LINE_STRIP pomoću koje je moguć prikaz kose koji je jednostavan za implementaciju te grafičkoj kartici za prikaz.

Međutim, postoje nedostatci ovakvog prikaza. Širina linije je cijeli broj koji označava broj piksela na ekranu te je jedan poziv za iscrtavanje ograničen na jednu širinu linije. Zbog toga, kosa može izgledati pretanka kada je blizu kamere i preširoka kada je daleko od kamere. Drugi nedostatak je primjena teksture na linije. Linije podržavaju samo 1D teksture te nije moguće primijeniti teksturu po širini dlake kose.

Rješenje ovih problema je prikaz dlake kose kao niz pravokutnika koji gledaju prema kameri. Svaki pravokutnik je prikazan uz pomoć dva trokuta te OpenGL primitive GL_TRIANGLE_STRIP. Prednosti prikaza kose kao niz pravokutnika je svaka dlaka kose ima stvarnu duljinu u sceni. Moguće je i imati različite duljine dlake pri korijenu dlake i pri vrhu dlake. Osim toga, na pravokutnik je moguće primijeniti 2D teksture za razliku od linija. Nedostatak ovakvog prikaza je veća kompleksnost prikaza kose jer je grafičkog kartici teže prikazati niz trokuta nego jednu liniju.

Pretvorbu iz linije u pravokutnike je moguće napraviti u pomoć sjenčara geometrije.

2.2.1. Pisanje sjenčara geometrije

Pri pisanju sjenčara geometrije, potrebno je definirati ulazne i izlazne primitive. Kako na ulazu dobijemo linije, a na izlazu želimo pravokutnik, potreban je sljedeći kod u sjenčaru.

```
layout(lines) in;
layout(triangle_strip, max_vertices = 4) out;
```

Programski kod 2.1 - Ulaz i izlaz sjenčara geometrije

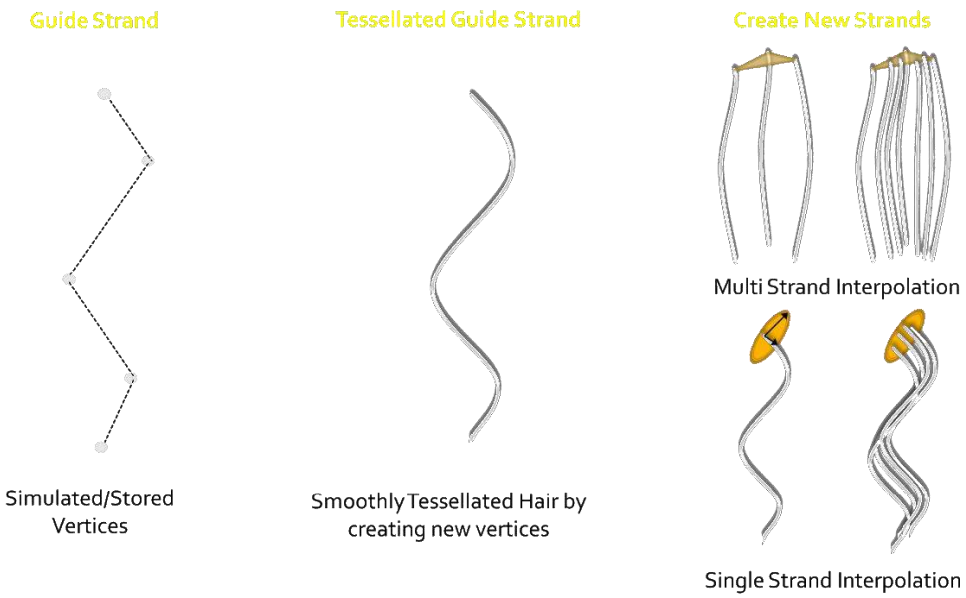
Za izračun samih pozicija, potrebna je tangenta dlake te vektor prema očištu. Tangentu je moguće izračunati uz pomoć pozicije vrhova linije. U programskom kodu 2.2 je prikazan pseudokod izračuna pozicija vrhova pravokutnika.

```
vec3 Tangent = normalize(vertexPosition[1] –
vertexPosition[0]);
vec3 sideVec = normalize(cross(eyeVec, tangent));
vec3 outPosition;
vec3 width0 = sideVec * 0.5 * vertexWidth[0];
vec3 width1 = sideVec * 0.5 * vertexWidth[1];
outPosition = vertexPosition[0] - width0;
EmitVertex();
outPosition = vertexPosition[0] + width0;
EmitVertex();
outPosition = vertexPosition[1] - width1;
EmitVertex();
outPosition = vertexPosition[1] + width1;
EmitVertex();
```

Programski kod 2.2 – Izračun pozicija vrhova pravokutnika

2.3. Interpolacija kose

Tipična ljudska glava sadrži oko 100 tisuća dlaka kose. Simulacija svih 100 tisuća dlaka kose je računalno zahtjevna. Osim same simulacije, ovisno o grafičkoj kartici, ponekad je brže prebaciti samo dio podataka o kosi te kreirati ostale dlake interpolacijom. Interpolacija dlaka kose je prikazana na Sl. 2.2 Interpolacija kose.

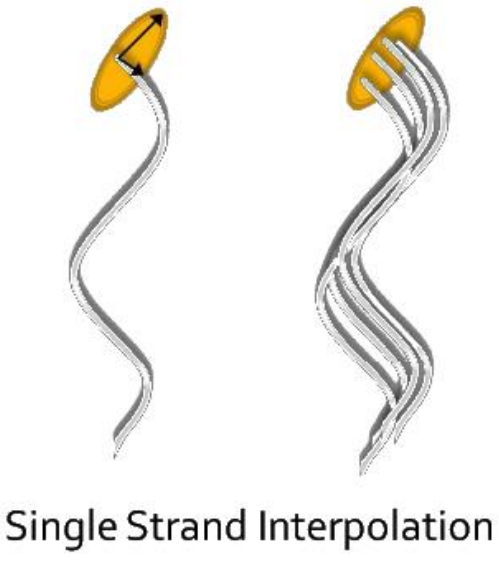


Sl. 2.2 Interpolacija kose [1]

Kosu je moguće interpolirati na dva načina. Moguće je interpolirati kosu iz jedne dlake ili pomoću više dlaka.

2.3.1. Interpolacija iz jedne dlake

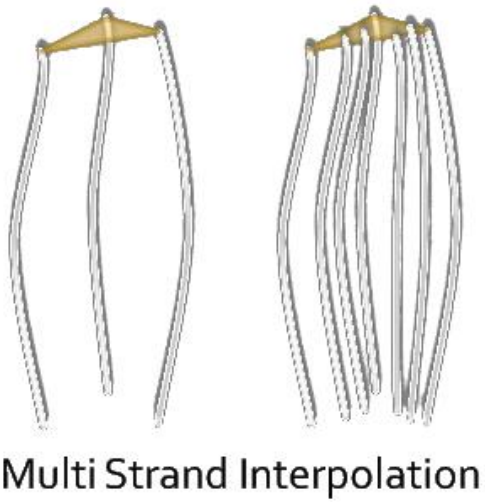
Za ovu vrstu interpolacije, dovoljno je znati poziciju vrhova samo jedne dlake. Ovisno o željenoj gustoći kose, potrebno je stvoriti nove dlake kose pomicanjem inicijalne dlake po x i z koordinati. Konačan rezultat ove interpolacije je prikazan na Sl. 2.3 Interpolacija iz jedne dlake.



Sl. 2.3 Interpolacija iz jedne dlake [1]

2.3.2. Interpolacija iz više dlaka

Ova metoda interpolacije stvara nove dlake kose pomoću pozicija više dlaka. Ovo je metoda koje je implementirana u radu te se koristi interpolacije između tri dlake. Rezultat ove interpolacije je prikazan na Sl. 2.4 Interpolacija iz više dlaka.



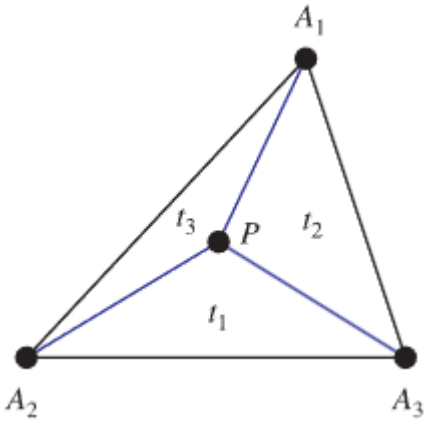
Sl. 2.4 Interpolacija iz više dlaka [1]

Interpolacije između tri dlake je postignuta uzimanjem nasumičnih baricentričnih koordinata t1 i t2. Zbroj svih baricentričnih koordinata iznosi 1 prema izrazu (1). Treću koordinatu je moguće izračunati uz pomoć prve dvije prema izrazu (2). Konačna pozicija vrha P je izračunata pomoću izraza (3), kao što je prikazano na Sl. 2.5 Baricentrične koordinate.

$$t_1 + t_2 + t_3 = 1 \tag{1}$$

$$t_3 = 1 - t_1 - t_2 \tag{2}$$

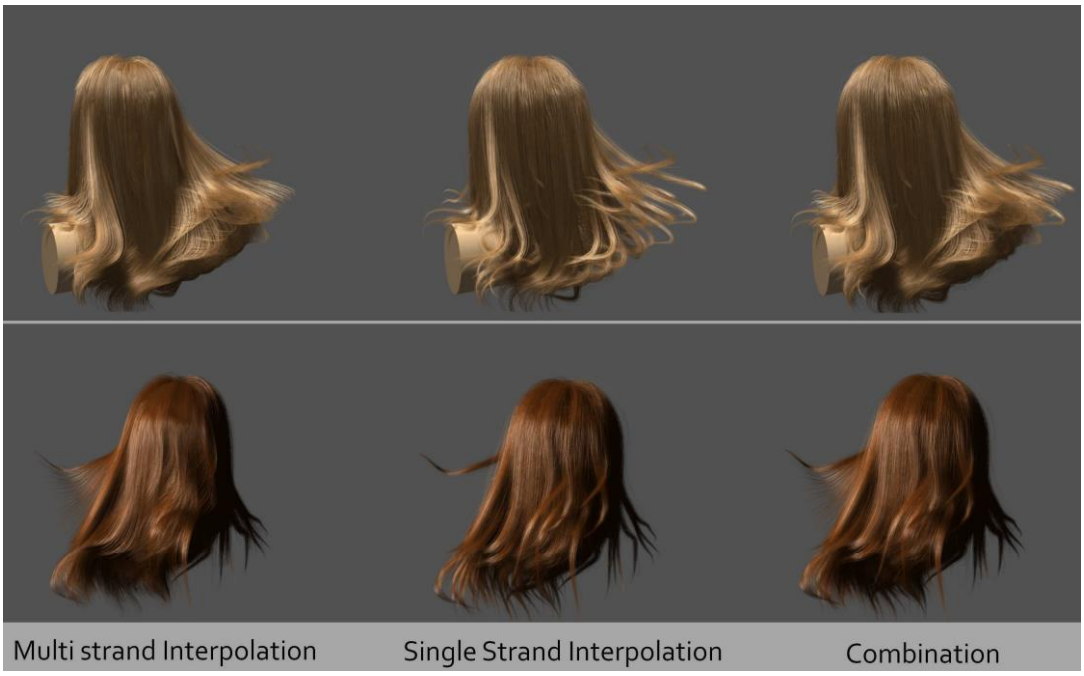
$$P = t_1 A_1 + t_2 A_2 + t_3 A_3 \tag{3}$$



Sl. 2.5 Baricentrične koordinate

2.3.3. Miješanje više vrsta interpolacije

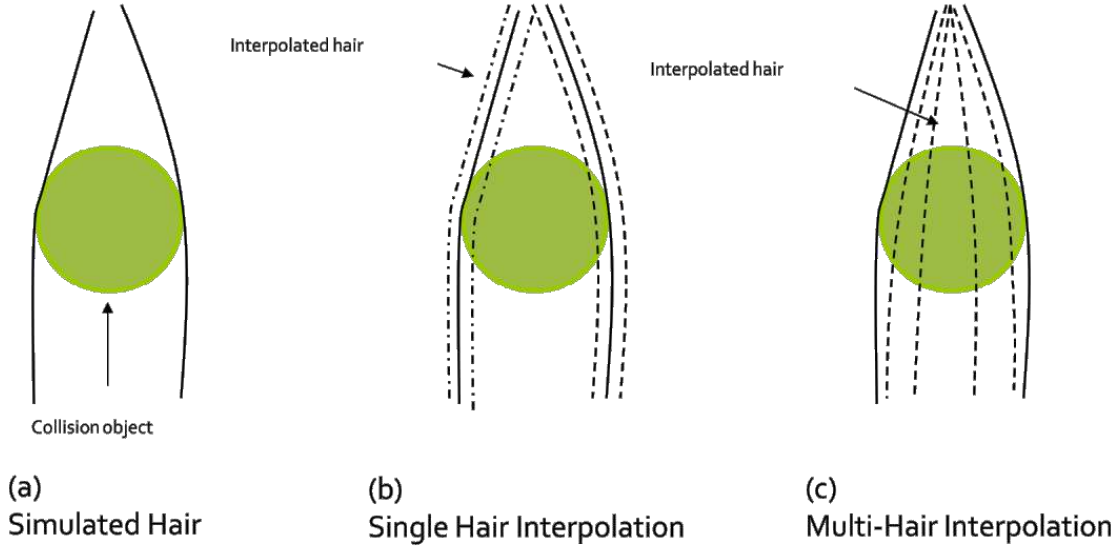
Iako navedene vrste interpolacije mogu dati dobre rezultate pri simulaciji kose, izvrsni rezultati se mogu postići miješanjem različitih vrsta interpolacije. Ponekad, određene vrste interpolacije mogu izgledati lošije za neke vrste kose. Na primjer, interpolacija iz jedne dlake grupira interpolirane dlake kose oko inicijalnih dlaka koje su proslijeđene teselatoru. Miješanjem raznih vrsta interpolacije se može smanjiti negativni utjecaji pojedinih interpolacija na ukupan izgled kose. Razlika između vrsta interpolacije oko jedne dlake, interpolacije između više dlaka te kombinacije obje interpolacije se mogu primijetiti na Sl. 2.6 .



Sl. 2.6 Razlika izgleda interpolacija [1]

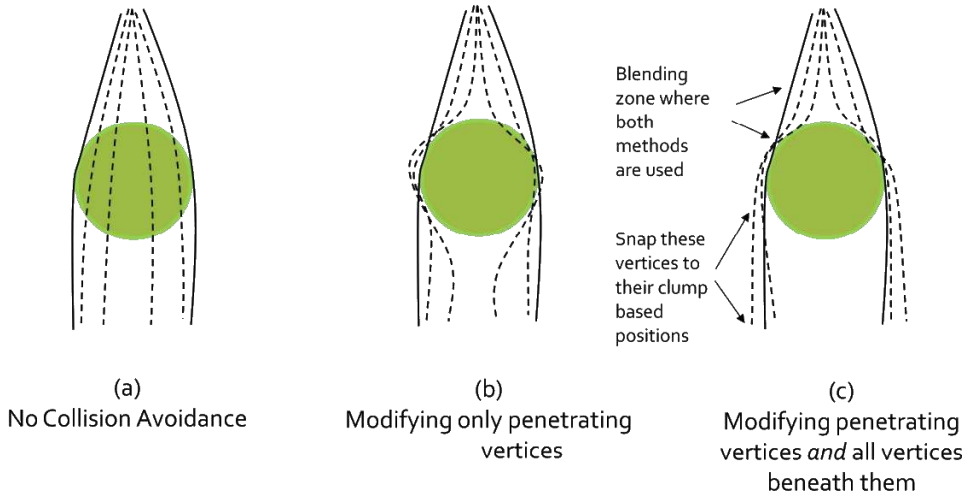
2.3.4. Interpolacija i sudari

U slučaju kada su drugi objekti u sceni, posebnu pažnju treba obratiti na vrstu interpolacije kose. Interpolirana kosa može biti stvorena unutar drugog objekta, što negativno utječe na izgled kose. Ovaj problem je posebice vidljiv pri interpolacije između više dlaka kose, kao što je prikazano na Sl. 2.7 Interpolacija bez detekcije sudara.



Sl. 2.7 Interpolacija bez detekcije sudara [1]

Čak i ako se primjeni detekcija sudara i svi vrhovi se pomaknu da nisu u sudaru, interpolacija između više dlaka kose ne daje zadovoljavajuće rezultate. Zato je potrebno za sve vrhove koji su u sudaru i za sve vrhove ispod njih u dlaci promijeniti vrstu interpolacije na interpolaciju iz jedne dlake, kao što je prikazano na Sl. 2.8 Interpolacija s detekcijom sudara.



Sl. 2.8 Interpolacija s detekcijom sudara [1]

2.3.5. Pisanje sjenčara teselacije

Za postupak teselacije, potrebno je napisati dva sjenčara, sjenčar kontrole teselacije i sjenčar evaluacije teselacije. Potrebne su dvije razine teselacije, jedna po duljini dlake koja će stvoriti nove vrhove i napraviti detaljniju dlaku kose te jedna razina koja će stvarati nove dlake kose.

2.3.5.1 Sjenčar kontrole teselacije

U ovom sjenčaru se određuje razina teselacije te varijable koje se prosljeđuju direktno sjenčaru evaluacije teselacije. Razine teselacije određuju se varijablom gl_TessLevelOuter[4]. Dvije pomoćne varijable su proslijeđene sljedećem sjenčaru, rootIndex koji određuje indeks dlake i segmentIndex koji određuje indeks segmenta unutar dlake. To se može postići Programskim kodom 2.2. Broj segmenata u jednoj dlaci, gustoća i faktor teselacije se mogu proslijediti kao uniformne varijable.

```
patch out int rootIndex;
patch out int segmentIndex;

void main()
{
      if(gl_InvocationID == 0) {
            rootIndex = gl_PrimitiveID / segmentsCount;
            segmentIndex = gl_PrimitiveID % segmentsCount;
```

```
            gl_TessLevelOuter[0] = density;
            gl_TessLevelOuter[1] = tesselationFactor;
    }
}
```

Programski kod 2.2 - Sjenčar kontrole teselacije

2.3.5.2 Sjenčar evaluacije teselacije

Prije opisa koda ovog sjenčara, potrebno je spomenuti da je za interpolaciju između tri dlake potreban generator slučajnih brojeva. GLSL ne nudi metodu koja generira slučajne brojeve te je potrebno koristiti vlastitu implementaciju. Funkcija koje je korištena u ovom radu kao generator pseudoslučajnih brojeva je funkcija koja se često koristi u GLSL-u te ima slična obilježja hash funkcijama. Za isti ulaz, ova funkcija će dati isti izlaz, ali za male pomaku u ulazu, izlaz će biti vrlo različit.

```
float rand(vec2 co)
{
      return fract(sin(dot(co.xy ,vec2(12.9898,78.233))) *
43758.5453);
}
```

Programski kod 2.3 - Pseudoslučajni generator brojeva u GLSL-u

Uz pomoć generatora pseudoslučajnih brojeva, moguće je odrediti baricentrične koordinate za interpolaciju. Ova funkcija je opisana u Programskom kodu 2.4. U funkciji rand(), brojevi 0.6 i -0.4 su uzeti kao bilo koja dva nasumična broja.

```
vec3 getBarycentricCoordinates()
{
      float u = rand(vec2(gl_TessCoord.y, 0.6));
      float v = rand(vec2(gl_TessCoord.y, -0.4));
      if(u + v > 1.0)
      {
            u = 1.0 - u;
            v = 1.0 - v;
      }
      return vec3(u, v, 1.0 - u - v);
}
```

Programski kod 2.4 - Određivanje baricentričnih koordinata

Sada kada su dobivene baricentrične koordinate, moguće je odrediti interpolirane pozicije segmenta dlake. Pozicija se određuje prema izrazu (3). Potrebno je znati indekse korijena od sve tri dlake te indeks segmenta. getVertexPosition() je pomoćna funkcija koja uz pomoć indeksa korijena i segmenta dobiva koordinate vrha segmenta dlake u sceni.

```
vec3 getInterpolatedPosition(ivec3 rootIndices, int
segmentIndex, vec3 barycentricCoords)
{
	vec3 position = vec3(0, 0, 0);
	position += getVertexPosition(rootIndices[0],
segmentIndex) * barycentricCoords[0];
	position += getVertexPosition(rootIndices[1],
segmentIndex) * barycentricCoords[1];
	position += getVertexPosition(rootIndices[2],
segmentIndex) * barycentricCoords[2];
	return position;
}
```

Programski kod 2.5 - Izračun interpolirane pozicije vrhova dlake

Izračunate su pozicije vrhova interpolirane dlake, no još uvijek je potrebno napraviti teselaciju po duljini dlake kako bi se dobila glatka dlaka kose. U tu svrhu, izračunati će se B-krivulja sa dobivenim interpoliranim vrhovima dlake kao kontrolnim točkama te će se teselacija izvršiti po B-krivulji. Periodički segment B-krivulje (k=3) računa se prema izrazu (4).

$$\overrightarrow{p_i}(t) = [t^3 \quad t^2 \quad t \quad 1]\frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \vec{r}_{i-1} \\ \vec{r}_i \\ \vec{r}_{i+1} \\ \vec{r}_{i-2} \end{bmatrix} \tag{4}$$

Za računanje B-krivulje, potrebne su pozicije prošlog segmenta, trenutnog segmenta te sljedeća dva segmenta. Izračun konačne pozicije vrhova interpolirane dlake će moguće postići Programskim Kodom 2.6.

```
vec3 r0 = getInterpolatedPosition(rootIndices, segmentIndex -
1, weights);
```

```
vec3 r1 = getInterpolatedPosition(rootIndices, segmentIndex,
weights);
vec3 r2 = getInterpolatedPosition(rootIndices, segmentIndex +
1, weights);
vec3 r3 = getInterpolatedPosition(rootIndices, segmentIndex +
2, weights);
float t = gl_TessCoord.x;
float t2 = t * t;
float t3 = t2 * t;
vec4 tVector = vec4(t3, t2, t, 1);
mat4 coeffMatrix;
coefficientMatrix[0] = vec4(-1, 3, -3, 1);
coefficientMatrix[1] = vec4(3, -6, 0, 4);
coefficientMatrix[2] = vec4(-3, 3, 3, 1);
coefficientMatrix[3] = vec4(1, 0, 0, 0);
vec4 bSpline = (tVector / 6.0) * coefficientMatrix;
out_pos = r0 * bSpline[0] + r1 * bSpline[1] + r2 * bSpline[2]
+ r3 * bSpline[3];
```

Programski kod 2.6 - Teselacija dlake koristeći B-krivulju

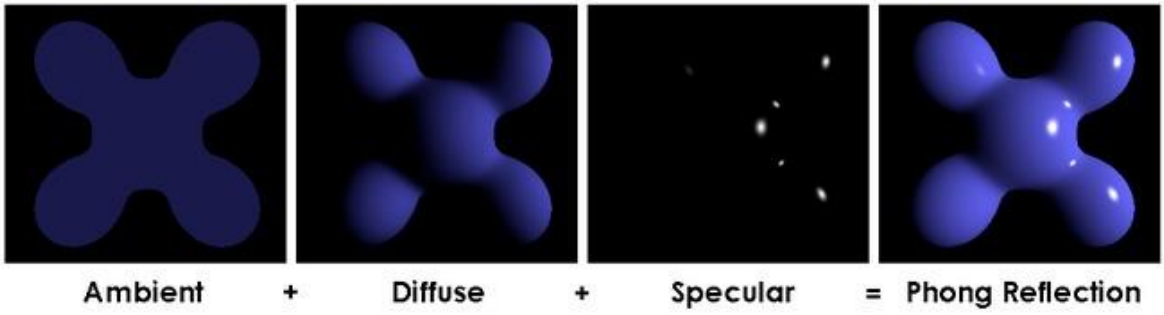
Još je jedino potrebno odrediti širinu segmenta dlake te tu vrijednost proslijediti sjenčaru geometrije. Širina može biti fiksna vrijednost, no moguće je i imati različitu širinu u korijenu dlake i vrhu dlake. To je moguće postići Programskim kodom 2.7. Širina korijena dlake i vrha dlake mogu biti proslijeđene sjenčaju kao uniformne varijable.

```
out_width = mix(rootWidth, tipWidth, clamp((segmentIndex +
gl_TessCoord.x) / segmentsCount, 0.0, 1.0));
```

Programski kod 2.7 - Izračun širine segmenta dlake

2.3.6. Model osvjetljavanja

Za osvjetljavanje kose, korišten je Phongov model osvjetljavanja. Phongov model osvjetljavanja se sastoji od tri komponente: ambijentom, difuznom i zrcalne. Ove komponente prikazane su na Sl. 2.9 Phongov model osvjetljavanja.



Sl. 2.9 Phongov model osvjetljavanja [8]

Intenzitet ambijente komponente je konstantan te ga nije potrebno računati.

Intenzitet difuzne komponente računa se prema izrazu (5) gdje je k_d koeficijent difuzne refleksije, I_P intenzitet točkastog izvora, N normala vrha dlake i L vektor prema izvoru svjetlosti.

$$I_d = k_d I_P (N * L) \tag{5}$$

Intenzitet zrcalne komponente računa se prema izrazu (5) gdje je k_s koeficijent zrcalne refleksije, I_P intenzitet točkastog izvora, E vektor prema očištu i R reflektirani vektor izvora svjetlosti oko normale objekta i n koji određuje prostornu razdiobu i vezan je uz materijal.

$$I_s = k_s I_P (E * R)^n \tag{6}$$

2.3.6.1 Sjenčar fragmenata

Svrha sjenčara fragmenata je izračunati sve tri komponente Phongovog modela osvjetljavanja te na temelju intenziteta svjetla i boje objekta odrediti konačnu boju fragmenta. Taj postupak je prikazan u Programskom kodu 2.8.

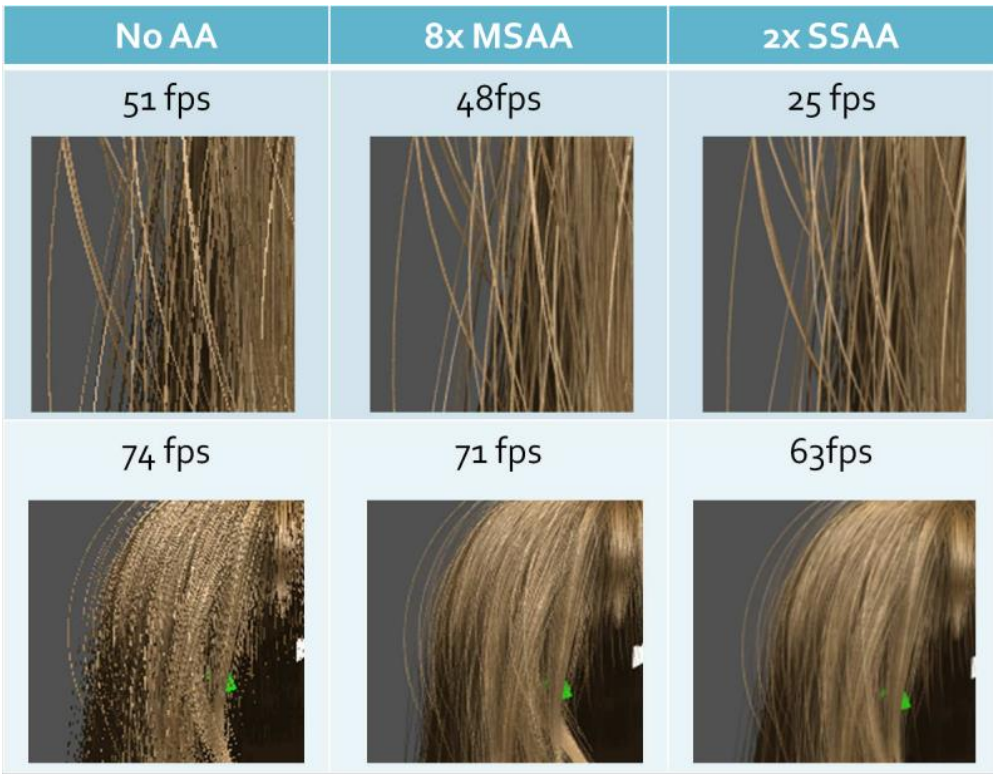
```
void main()
{
	vec3 ambient = ambientStrength * lightColor;
	vec3 lightVec = normalize(lightPos - in_pos);
	float diff = max(dot(in_normal, lightVec), 0.0);
	vec3 diffuse = diffuseStrength * diff * lightColor;
	vec3 eyeVec = normalize(in_pos - eyePos);
```

```
        vec3 reflectedVec = reflect(lightVec, in_normal);
        float spec = pow(max(dot(eyeVec, reflectVec), 0.0),
specularPower);
        vec3 specular = specularStrength * spec * lightColor;
        vec3 result = (ambient + diffuse + specular) *
objectColor;
        out_color = vec4(result, 1.0);
}
```

Programski kod 2.8 - Sjenčar fragmenata

2.3.7. Uklanjanje neželjenih učinaka diskretizacije (Anti-aliasing)

Pri uzorkovanju boje fragmenta, može doći do pojave alias učinaka. Alias ima negativni utjecaj na izgled kose, kao što je prikazano na Sl. 2.10 Usporedba metoda Anti-aliasing.



Sl. 2.10 Usporedba metoda Anti-aliasing metoda [1]

OpenGL ima ugrađenu podršku za Multisample Anti-aliasing (MSAA). Anti-aliasing je moguće uključiti Programskim kodom 2.9.

```
glEnable(GL_MULTISAMPLE);
glfwWindowHint(GLFW_SAMPLES, n);
```

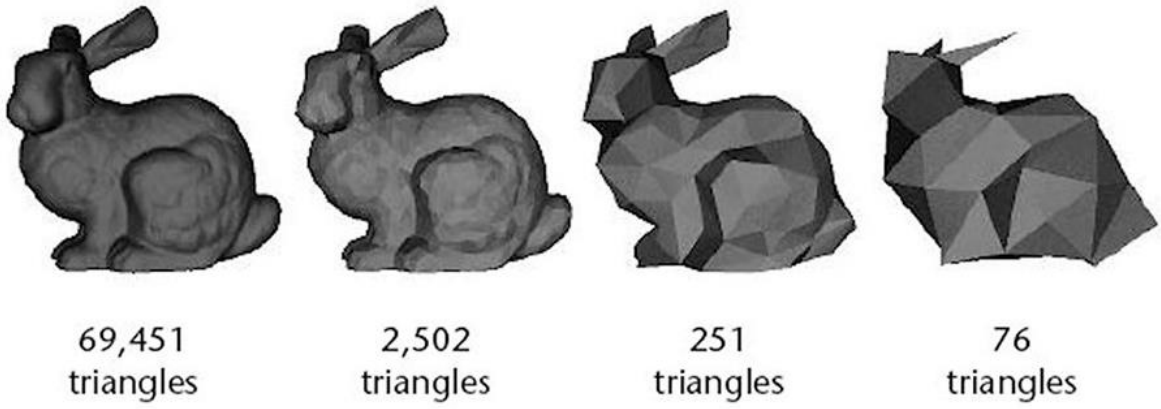
Programski kod 2.9 - Multisample Anti-aliasing

2.3.8. Dodatne mogućnosti

Ovisno o potrebama aplikacije, moguće je implementirati dodatne funkcionalnosti koje mogu poboljšati efikasnost ili kvalitetu prikaza kose.

2.3.8.1 Razina detalja

Mogućnost dinamičkog upravljanja kompleksnosti kose može biti jako bitno za aplikacije u stvarnom vremenu. Ovisno o udaljenosti od kamere, poželjno je postupno smanjenje kompleksnosti prikaza kose. Primjer raznih razina detalja je prikazan na Sl. 2.11 Razne razine detalja.



Sl. 2.11 Razne razine detalja

Ova funkcionalnost se može postići dinamičkim mijenjanjem razina teselacije u sjenčaru kontrole teselacije te promjenom širine kose pri stvaranju pravokutnika u sjenčaru geometrije. Udaljavanjem od kamere, potrebno je smanjiti razine teselacije te povećati širinu dlake kose.

2.3.8.2 Kovrčava kosa

Pri modeliranju kovrčave kose, potrebno je spremiti dodatne pomake od početne pozicije u dodatan spremnik ili kao teksturu. Ove pomake je moguće generirati proceduralno

ili ih je moguće eksplicitno izračunati iz stvarnih modela. Ovaj postupak je prikazan na Sl. 2.12 Kovrčava kosa.

U sjenčaru evaluacije teselacije, učitava se spremnik pomaka te se dodaje pomak pri interpolaciji dlaka kose.

Sl. 2.12 Kovrčava kosa [1]

2.3.8.3 Nasumične varijacije

Dodavanje nasumičnih varijacija može dati kosi realističniji izgled. Bez nasumičnih varijacija, kose može izgledati previše glatka i sintetična.

Ovaj efekt je moguće postići dodavanjem slučajnog pomaka na određeni dio interpoliranih dlaka kose. Slučajne pomake je moguće unaprijed izračunati i proslijediti sjenčarima.

3. Fizikalna simulacija kose

U ovom poglavlju, fizikalno ponašanje dlake kose će biti opisano. Simulacija se izvršava samo na dijelu dlaka kose, prije interpolacije i stvaranja novih dlaka u sjenčaru teselacije.

3.1. Integracija

Izračun nove pozicije te brzine dlake kose se računa pomoću Verlet integracije. Nova pozicija se računa prema izrazu (7).

$$X_{i+1} = X_i + v_i \Delta t + a_i \Delta t^2 \tag{7}$$

Brzina se računa prema izrazu (8).

$$v_{i+1} = \frac{X_{i+1} - X_i}{\Delta t} \tag{8}$$

Za razliku od Eulerovog postupka, Verletov postupak je stabilan za simulaciju sustava. Eulerov postupak zahtijeva mali korak integracije kako bi bio stabilan te akumulira grašku kroz simulaciju. Postoje druge metode koje su numerički stabilne za veći korak integracije poput implicitnog Eulerovog postupka, no prednost Verletovog postupka je nije potrebno spremati informacije o brzini. Izraz za izračun pozicije bez člana brzine prikazan je sljedećom formulom.

$$X_{i+1} = 2X_i - X_{i-1} + a_i \Delta t^2 \tag{9}$$

U ovom koraku simulacije, ne postoje ovisnosti između vrhova te je moguće primijeniti Verlet integraciju na sve vrhove paralelno.

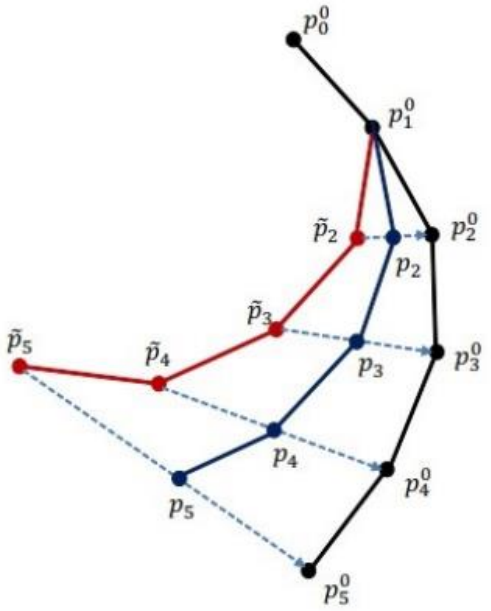
3.2. Fizikalna ograničenja

Fizikalna ograničenja drže dlaku kose u određenom položaju. Postoje tri vrste ograničenja koja su primijenjena na dlaku kose: globalno ograničenje, lokalno ograničenje te ograničenje duljine. Fizikalna ograničenja se primjenjuju nakon Verlet integracije.

3.2.1. Globalno ograničenje

Globalno ograničenje održava strukturu dlake u poziciji odmora, koja je unaprijed definirana. To postiže tako da gura vrhove dlake prema poziciji odmora, ovisno o koeficijentu globalnog ograničenja k_G, kao što je prikazano na Sl. 3.1 Globalno ograničenje. Koeficijent globalnog ograničenja mora biti broj između 0 i 1. Nova pozicija vrhova dobije se linearnom interpolacijom između trenutno pozicije vrha te poziciju vrha u odmoru kao u izrazu (10).

$$X_i = X_i + k_G(X_i^{rest} - X_i) \tag{10}$$

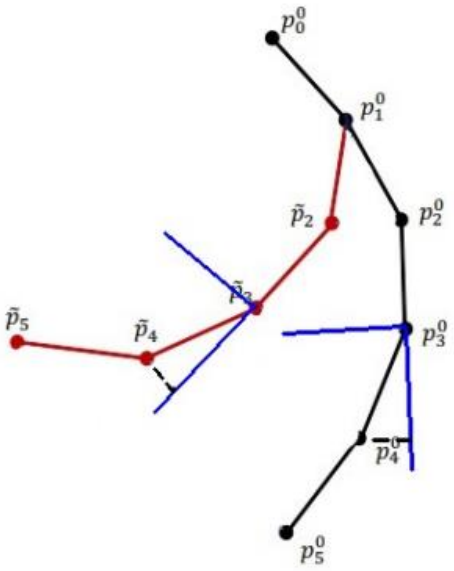


Sl. 3.1 Globalno ograničenje

3.2.2. Lokalno ograničenje

Lokalno ograničenje ispravlja deformaciju segmenta dlake ovisno o susjednim segmentima. Kao i globalno ograničenje, lokalno ograničenje pomaže održati strukturu kose. To postiže tako da gura vrhove segmenta da budu pod određenim kutom ovisno o prošlom segmentu, kao što je prikazano na . Nove pozicije segmenta se računaju prema izrazu (11)

$$X_{i-1} = X_{i-1} - \frac{1}{2} k_L (X_i^{rest} - X_i),$$
$$X_i = X_{i-1} + \frac{1}{2} k_L (X_i^{rest} - X_i) \tag{11}$$



Sl. 3.2 Lokalno ograničenje

Budući da ovo ograničenje mijenja pozicije više vrhova, potrebno ga je primijeniti više puta kako bi konvergiralo.

3.2.3. Ograničenje duljine

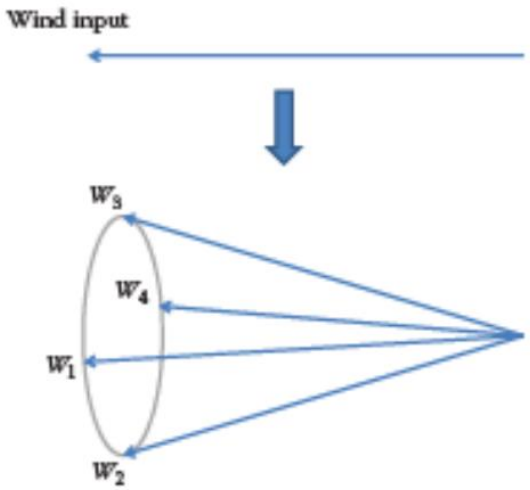
Ograničenje duljine osigurava nerastezljivost dlake. U svakom koraku iteracije, provjerava se duljina svakog segmenta te se uspoređuje sa inicijalnom duljinom. U slučaju da su duljine različite, oba vrha segmenta se pomaknu kako bi se održavala inicijalna duljina segmenta. Nove pozicije vrhova segmenta se računaju prema izrazu (12).

$$\Delta X_i = -\frac{1}{2} \Delta l * \vec{n},$$
$$\Delta X_{i+1} = \frac{1}{2} \Delta l * \vec{n} \tag{12}$$

Kao i kod lokalnog ograničenja, ovo ograničenje potrebno je primijeniti više puta kako bi konvergiralo jer mijenja pozicije više vrhova.

3.3. Simulacija vjetra

Simuliranjem vjetra unosimo nasumičnu promjenu u poziciji dlaka kose. Nasumičnost unosimo u simulaciju podjelom smjera vjetra na više vektora, kao što je prikazano na Sl. 3.3 Podjela smjera vjetra na više vektora.



Sl. 3.3 Podjela smjera vjetra na više vektora

Konačan smjer vjetra je dobiven interpolacijom između ovih vektora uz slučajne težine interpolacije. Magnituda vjetra može biti konstanta kroz cijelu simulaciju, no mijenjanje magnitude vjetra kroz simulaciju dalje bolje rezultate. Primjer funkcije magnitude je dan izrazom (13).

$$magnitude = sin^2(0.05 * frame) + 0.5 \tag{13}$$

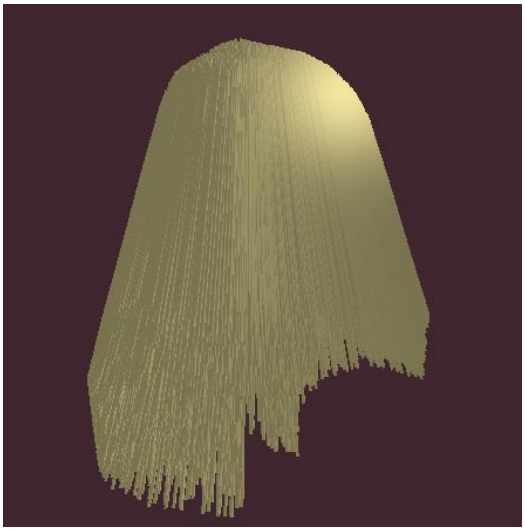
Sila vjetra se računa pomoću izraza (14) gdje je T tangenta segmenta dlake, a W konačan smjer vjetra.

$$F_W = magnitude * ((TxW)xT) \tag{14}$$

Nakon izračuna sile vjetra, iznos se dodaje u korak Verlet integracije kao jedna od vanjskih sila uz gravitaciju.

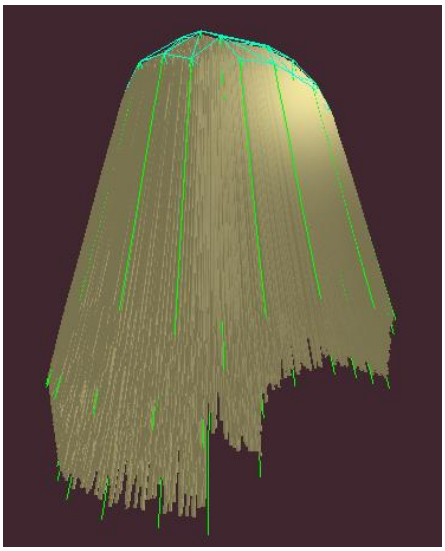
4. Rezultati

Postignut je prikaz i simulacija kose na jednostavnom module u OpenGL-u. Iz jednostavnog modela, dlake kose su interpolirane te je dobivena gusta kosa iz relativno malenog broja inicijalnih dlaka kose. Prikaz kose u stanju mirovanja je vidljiv na slici Sl. 4.1 Prikaz kose u stanju mirovanja.

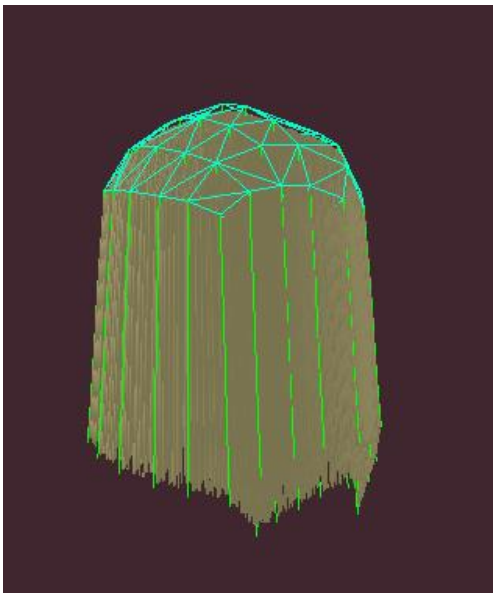


Sl. 4.1 Prikaz kose u stanju mirovanja

Simulaciji je dodana mogućnost prikaza inicijalnog modela kose. Kosa sa vizualizacijom inicijalnog modela je prikazana na slikama Sl. 4.2 Prikaz kose u stanju mirovanja sa uključenom vizualizacijom inicijalnog modela i Sl. 4.3 Prikaz kose iz drugog položaja sa uključenom vizualizacijom.

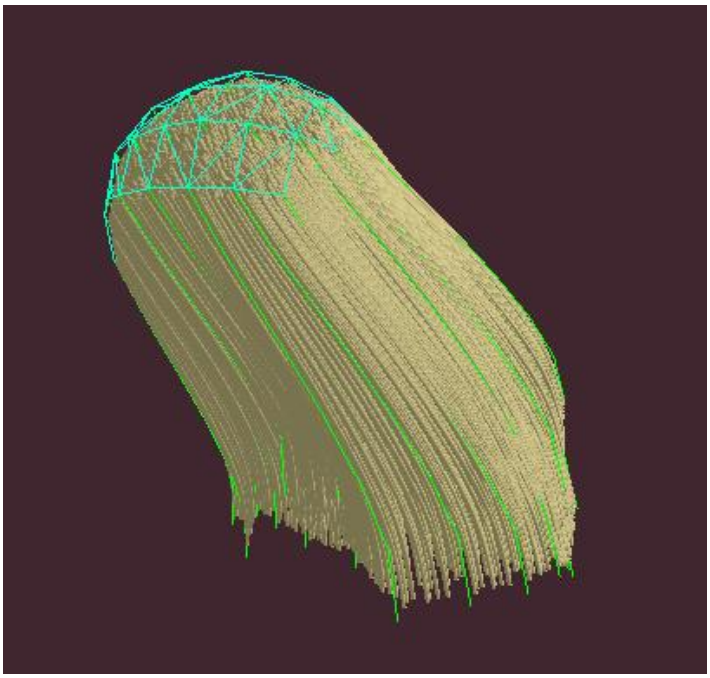


Sl. 4.2 Prikaz kose u stanju mirovanja sa uključenom vizualizacijom inicijalnog modela

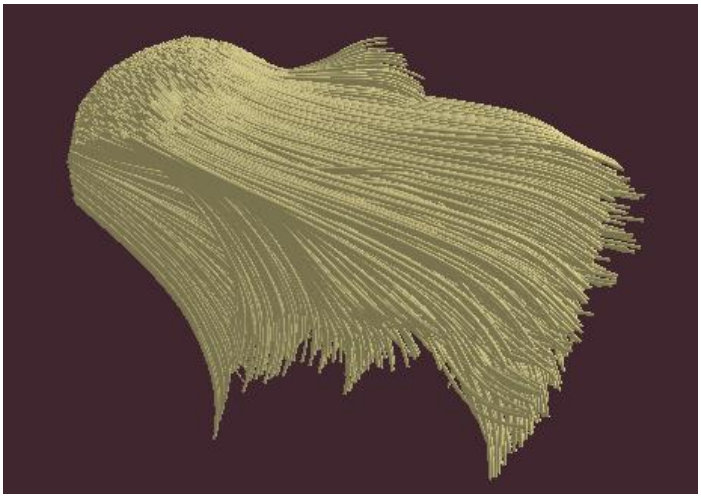


Sl. 4.3 Prikaz kose iz drugog položaja sa uključenom vizualizacijom

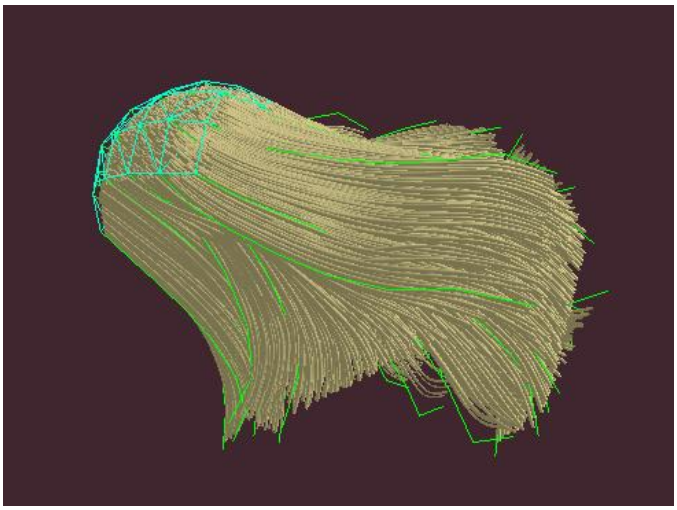
Osim simulacije gravitacije te fizikalnih ograničenja na dlake kose, simuliran je i vjetar. Kosa sa simulacijom slabog vjetra je prikazana na Sl. 4.4 Prikaz kose sa simulacijom slabog vjetra, dok kosa sa simulacijom jakog vjetra je prikazana na slikama Sl. 4.5 Prikaz kose sa simulacijom jakog vjetra i Sl. 4.6 Prikaz kose sa simulacijom jakog vjetra sa uključenom vizualizacijom.



Sl. 4.4 Prikaz kose sa simulacijom slabog vjetra



Sl. 4.5 Prikaz kose sa simulacijom jakog vjetra



Sl. 4.6 Prikaz kose sa simulacijom jakog vjetra sa uključenom vizualizacijom

Simulaciji je moguće promijeniti razne parametre, od snage vjetra do jačine fizikalnih ograničenja na dlake kose, kako bi se postignulo željeno ponašanje kose. Također, moguće je učitati složenije modele kose sa većim brojem dlaka te smanjiti razinu teselacije novih dlaka kose kako bi se preciznije računala simulacija. Međutim, povećanjem broja dlaka koje se simuliraju zahtijeva više računalnih resursa te je iz tog razloga odabran jednostavan model.

Trenutni model sadrži 87 dlaka kose koje se simuliraju, što nakon postupka teselacije sa faktorom 64 postaje 5568 dlaka kose koje se prikazuju na zaslon. Simulacija je pokrenuta na grafičkoj kartici AMD Radeon RX 590, procesoru AMD Ryzen 5 1500X te sa 16 GB RAM-a. U prosjeku, simulacija postiže 1120 okvira u sekundi (engl. frames per second).

5. Moguća poboljšanja

5.1. Korištenje drugih modela osvjetljavanja

Iako Phongov model osvjetljavanja daje dovoljno dobre rezultate, postoje mnogi napredniji algoritmi koji daju mnogo bolje rezultate. Primjer nekih od algoritama su Path tracing i Dual scattering, prikazani na Sl. 5.1 Usporedba metoda osvjetljavanja.

Sl. 5.1 Usporedba metoda osvjetljavanja [1]

5.2. Detekcija sudara sa ljudskom glavom

Detekcija sudara sa objektima proizvoljnog oblika u 3D nije trivijalna te može biti računski zahtjevna. Postoje mnoge metode detekcije i rezolucije sudara, no optimizacija tih metoda može biti izuzeto zahtjevna.

5.3. Detekcija sudara između dlaka kose

Detekcijom sudara između dlaka kose moguće je poboljšati izgled kose, dajući volumen i bolji oblik kosi. Problem kod detekcije sudara između dlaka kose je velik broj dlaka, što može znatno usporiti samu simulaciju. Osim toga, ovakvi sudari mogu uzrokovati nestabilnosti u kosi, što je posebice vidljivo kada je kosa u položaju odmora.

Zaključak

Prikazivanje ljudske kose je računski zahtjevno zbog velikog broja dlaka kose. OpenGL ne nudi vlastitu implementaciju za učitavanje modela te je potrebno koristiti vlastitu implementaciju ili neku od biblioteka. Dlake kose mogu biti prikazane kao linije ili niz pravokutnika. Prikaz pomoću pravokutnika nudi bolje upravljanje širinom dlake te omogućuje korištenje 2D tekstura. Zbog učinkovitosti simulacije i prikaza, samo se dio dlaka prenosi na grafičku karticu, a ostatak se stvara interpolacijom u sjenčaru teselacije. Moguće je interpolirati dlake kosu koristeći jednu ili više dlaka. Miješanjem vrsta interpolacije se mogu dobiti bolji rezultati. U slučaju sudara, potrebno je promijeniti vrstu interpolacije na interpolaciju iz jedne dlake za sve vrhove u sudaru i vrhove ispod njih. Teselacija po duljini dlake se računa uz pomoć B-krivulje. Za Phongov model osvjetljavanja, potrebno je izračunati ambijentnu, difuznu i zrcalnu komponentu unutar sjenčara fragmenata. OpenGL ima ugrađenu podršku za Multisample Anti-aliasing. Razina detalja se postiže mijenjanjem razine teselacije i širine dlake. Za sinulacije kovrčave kose, potrebno je spremiti dodatan spremnik pomaka. Moguće je postići realističniji izgled kose dodavanjem nasumičnih varijacija na dio dlaka kose. Pri fizikalnoj simulaciji kose, potrebno je prvi napraviti korak integracije te onda primijeniti fizikalna ograničenja. Fizikalna ograničenja održavaju oblik kose te osiguravaju nerastezljivost kose. Globalno ograničenje gura vrhove dlake kose prema poziciji odmora. Lokalno ograničenje ispravlja deformaciju segmenata kosu u odnosu na druge segmente. Ograničenje duljine osigurava nerastezljivost dlake kose. Simuliranjem vjetra unosimo nasumičnu promjenu pozicije dlaka kose. Nasumičnost vjetra postižemo interpolacijom sa slučajnim težinama.

Literatura

[1] Cem Yuksel, Sarah Tariq, Advanced Techniques in Real-time Hair Rendering and Simulation, SIGGRAPH 2010

[2] Luka Samac, *Simulacija krzna*, Diplomski projekt, Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2024.

[3] Dongsoo Han and Takahiro Harada. RealTime Hair Simulation with Efficient Hair Style Preservation. In VRIPHYS, edited by Jan Bender, Arjan Kuijper, Dieter W. Fellner, and Eric Guerin, pp. 4551. Aire-la-Ville, Switzerland: Eurographics Association, 2012.

[4] Petrovic L., Henne M., Anderson J.: Volumetric methods for simulation and rendering of hair. In Tech. rep., Pixar Animation Studios (2005)

[5] M. Müller, T.Y. Kim, N. Chentanez, Fast Simulation of Inextensible Hair and Fur, Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS, 2012

[6] Jerome Lengyel, Emil Praun, Adam Finkelstein and Hugues Hoppe. ”Real-Time Fur over Arbitrary Surfaces”. In proceedings of the 2001 symposium on Interactive 3D graphics, 2001.

[7] Joey de Vries, Learn OpenGL, Poveznica: https://learnopengl.com/; pristupljeno 18. lipnja 2024.

[8] Phong Shading, Poveznica: https://en.wikipedia.org/wiki/Phong_shading; pristupljeno 18. lipnja 2024.

[9] An early look at next-generation real-time hair and fur, Poveznica: https://www.unrealengine.com/en-US/tech-blog/an-early-look-at-next-generation-real-time-hair-and-fur; pristupljeno 18. lipnja 2024.

Sažetak

Fizikalno temeljen model ljudske kose

Sažetak

U ovom radu, prikazane su metode prikaza i simulacije ljudske kose. Implementacija je pisana u OpenGL-u. Opisane su metode prikaza, modeli osvjetljavanja, anti-aliasing, razina detalja, simulacija kovrčave kose te nasumična varijacija. Prikazane su dvije vrste interpolacije dlaka kose. Osim prikaza kose, objašnjena je i fizikalna simulacija kose. Korištena je Verlet integracija za izračun pozicija i brzine. Opisana su fizikalna ograničenja na model koja održavaju oblik kose. Vjetar je simuliran kako bi uveo nasumičnu promjenu pozicije dlaka kose. Priloženi su programski kodovi sjenčara geometrije, sjenčara teselacije te sjenčara fragmenata.

Ključne riječi: simulacija, fizikalna simulacija, ljudska kosa, kosa, OpenGL, GLSL, sjenčari, modeli osvjetljavanja, anti-aliasing, razina detalja, simulacija vjetra

Summary

Physically based model of human hair

Summary

In this thesis, human hair rendering and simulations methods are presented. The implementation was written in OpenGL. Rendering methods, shading methods, anti-aliasing, level of detail, curly hair simulation and random variation are described. Two types of hair interpolation are presented. Apart from rendering, physically based simulation is explained. To calculate speed and position, Verlet integration is used. Physical constraints which preserve the shape of the hair are described. Wind is simulated to bring random change to the position of the hair. Geometry shader code, Tessellation shader code and fragment shader code are shown.

Keywords: simulation, physically based simulation, human hair, hair, OpenGL, GLSL, shaders, shading, anti-aliasing, level of detail, wind simulation

Privitak

Programski kod je dostupan na poveznici**:**

https://github.com/Samcina/PhysicallyBasedHairSimulation