Proceduralno generiranje i prikaz krzna u stvarnom vremenu

Cafuk, Borna

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 286

PROCEDURALNO GENERIRANJE I PRIKAZ KRZNA U STVARNOM VREMENU

Borna Cafuk

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 286

PROCEDURALNO GENERIRANJE I PRIKAZ KRZNA U STVARNOM VREMENU

Borna Cafuk

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 286

Pristupnik:	**Borna Cafuk (0036513396)**
Studij:	Računarstvo
Profil:	Računalno inženjerstvo
Mentorica:	prof. dr. sc. Željka Mihajlović

Zadatak: **Proceduralno generiranje i prikaz krzna u stvarnom vremenu**

Opis zadatka:

Proučiti proceduralne tehnike generiranja krzna. Posebice obratiti pažnju na tehnike prikaza generiranih objekata u stvarnom vremenu. Razraditi i implementirati proceduralno generiranje krzna uz implementaciju prikaza u stvarnom vremenu. Razmotriti različite tehnike u postupku ostvarivanja prikaza. Načiniti programsku implementaciju koja omogućuje analizu i usporedbu načinjenih postupaka. Na različitim primjerima prikazati ostvarene rezultate. Načiniti ocjenu rezultata i implementiranih algoritama. Izraditi odgovarajući programski proizvod. Koristiti programski jezik C++ i grafičko programsko sučelje OpenGL. Rezultate rada učinit dostupne putem weba. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 9. veljače 2024.

Sadržaj

Uvod

Kosa i krzno sadrže mnogo sitnih detalja i time predstavljaju problem za prikaz u računalnoj grafici. Iako modeliranje svake pojedine vlasi omogućuje realistične simulacije i prikaze, potrebni izračuni takav pristup čine neprikladnim za prikazivanje u stvarnom vremenu. Ovakve izračunski skupe tehnike rijetko nalaze primjene izvan animacijskih studija. Igre i ostale primjene koje zahtijevaju interaktivnost se oslanjaju na tehnike prikladne za stvarno vrijeme.

U sklopu ovog rada izrađeno je programsko rješenje koje preko proizvoljnog 3D modela prikazuje krzno u stvarnom vremenu. Prvi dio rada daje pregled alata korištenih u izradi rješenja. U drugom dijelu bit će opisani i uspoređeni korišteni pristupi za prikaz krzna. Treći dio ukratko opisuje tehniku korištenu za generiranje uzorka krzna. Četvrti dio opisuje izrađeno programsko rješenje. Zaključno, bit će razmotrena korist opisanih pristupa i njihova upotreba u računalnoj grafici.

1. Korišteni alati i tehnologije

1.1. Programski jezik C++

C++ je programski jezik visoke razine koji se ustalio za razvoj grafičkih programa i video igara koji zahtijevaju visoke performanse. Izvorno je za javnu upotrebu objavljen 1985. godine.[1] Ovaj programski jezik i njegovu standardnu biblioteku normirala je Međunarodna organizacija za normiranje (ISO) i svake tri godine izdaje normu s novom inačicom jezika.[2, str. 3] Sljedeće izdanje je planirano za izdanje u 2026. godini.[2] Programsko rješenje razvijeno za ovaj rad koristi inačicu jezika iz 2020., koja se uobičajeno zove C++20. Jezik se najčešće prevodi u strojni jezik specifičan za određenu platformu i koji obično nije prenosiv na druge operacijske sustave i računalne arhitekture.

Kod korištenja C++-a, od programera se očekuje da vodi računa o zauzimanju i oslobađanju memorije, kao i o upravljanju drugim resursima, npr. opisnicima datoteka, dretvama ili teksturama na grafičkom procesoru. Kroz idiom poznat kao RAII (engl. *resource acquisition is initialization*, stjecanje resursa je inicijalizacija), upravljanje resursa je usko vezano uz životni tijek objekata. Time se u usporedbi s radom u programskom jeziku kao što je C smanjuje misaono opterećenje programera pri radu s resursima. U C-u, kod koji „počisti" korištene resurse mora eksplicitno biti pozvan pri izlasku iz funkcija, s posebnom pozornošću na to da se pozove i kad se dogodi greška. Za razliku od C-a, ukljanjanje varijable sa stoga ili njena dealokacija s gomile u C++-u bi trebala u destruktoru automatski osloboditi i sve resurse koje posjeduje. Iako je normom u inačici C++11 definirano kako bi implementacija jezika mogla podržavati sakupljanje smeća, niti jedan prevoditelj podršku nije implementirao i ta mogućnost je uklonjena u inačici C++23.[3]

C++ podržava polimorfizam tijekom prevođenja (engl. *compile-time polymorphism*)

kroz upotrebu predložaka (engl. *templates*). Predlošci kao argumente mogu imati tipove podataka, što se npr. koristi za definiciju generičkih struktura podataka. Prevoditelj monomorfizira razrede i funkcije definirane predlošcima, pri čemu generira zaseban kod za svaku specijalizaciju predloška.

1.2. Programsko sučelje za grafiku OpenGL

OpenGL je programsko sučelje (API) za računalnu grafiku. OpenGL specifikacija je slobodno dostupna, razvija ju Khronos grupa, konzorcij preko 150 tvrtki.[4] Prva inačica OpenGL specifikacije je objavljena 1992. kao alternativa IRIS GL-u, koji nije bio slobodno dostupan.[5]

Programsko rješenje razvijeno u sklopu ovog rada koristi OpenGL inačicu 4.5, čija je specifikacija objavljena 2017. godine.[6] Ova inačica je odabrana jer je uvela direktan pristup stanju (DSA, engl. *direct state access* u jezgreni profil OpenGL-a. DSA olakšava rad s objektima OpenGL-a jer dozvoljava dohvat i izmjenu stanja većine objekata bez da se objekt mora vezati na kontekst.[6, dio I.3.3.153] Vezanje objekta mijenja globalni kontekst OpenGL-a, čime se otežava rad s objektima OpenGL-a na enkapsuliran, objektno orijentiran način. Bez DSA, programer mora voditi računa o stanju OpenGL-a i sjetiti se relevantne dijelove globalnog stanja postaviti na poznate vrijednosti prije nego izvodi operacije koje ovise o stanju.

Programski kôd 1.1. pokazuje izmjene globalnog stanja potrebne da bi se teksturi promijenio način filtriranja i da bi ju se vezalo na teksturnu jedinicu (engl. *texture unit*) za iscrtavanje. Retci 4 i 14 sadrže pozive funkcija koji mijenjaju globalno stanje na način koji s DSA ne bi bio potreban. Programski kôd 1.2. prikazuje iste operacije obavljene korištenjem DSA. Izmjene globalnog stanja svedene su na minimum. Kako se sampler uniform varijable sjenčara referiraju na teksture korištenjem brojčanih oznaka teksturnih jedinica, teksture i dalje trebaju biti vezane na teksturne jedinice, ali se to može učiniti bez mijenjanja aktivne teksturne jedinice.

```
void changeFilteringMode() {
  // ...
  // Binds the texture to the active texture unit!
  glBindTexture(GL_TEXTURE_2D, texture);

  // Requires the texture to be bound to the active texture unit.
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  // ...
}

void render() {
  // ...
  // This changes which texture unit is active!
  glActiveTexture(GL_TEXTURE0);

  // Requires the desired texture unit to be active.
  glBindTexture(GL_TEXTURE_2D, texture);
  // ...
}
```

Programski kôd 1.1.: Primjer koda koji ne koristi DSA.

```
void changeFilteringMode() {
  // ...
  // Sets the filtering mode without any binding.
  glTextureParameteri(texture, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  // ...
}

void render() {
  // ...
  // Binds the texture to a texture unit
  // without changing the active unit.
  glBindTextureUnit(0, texture);
  // ...
}
```

Programski kôd 1.2.: Primjer koda koji koristi DSA.

1.3. Biblioteka SDL 2

Simple DirectMedia Layer (SDL) je biblioteka za razvoj grafičkih aplikacija za razne platforme. SDL predstavlja apstrakcijski sloj koji omogućava upravljanje prozorima, unosom s miša i tipkovnice, zvuk, itd. na jednak način neovisno o operacijskom sustavu. Vrijedno je spomena da se igre tvrtke Valve, ali i brojnih drugih autora, uvelike oslanjaju na inačicu 2 ove biblioteke. Biblioteka je izrađena u programskom jeziku C i dostupna pod zlib licencom.[7]

Iako SDL 2 sadrži vlastitu implementaciju jednostavnog crtanja pravokutnika i slika, također podržava i izradu prozora za korištenje s OpenGL-om, Vulkanom ili Metalom. Također postoje dodatne biblioteke koje SDL-u 2 dodaju funkcionalnosti mrežne komunikacije, prikaza TrueType pisama, te miješanja zvuka.

1.4. Biblioteka Dear ImGui

Dear ImGui je biblioteka za izradu grafičkih korisničkih sučelja. Izrađena je u C++-u i dostupna pod MIT licencom.[8]

Osnovni dio biblioteke je neovisan o platformi i grafičkom API-ju. Rukovanje ulaznim podacima i iscrtavanje su izdvojeni u komponente specifične za pojedinu platformu i grafički API. Programsko rješenje razvijeno za ovaj rad koristi komponentu za rukovanje korisničkim unosom pomoću SDL-a 2, kao i komponentu za iscrtavanje korištenjem inačica OpenGL-a od 3 naviše.

1.5. Biblioteka Open Asset Import Library

Open Asset Import Library (assimp) je biblioteka za uvoz raznih formata datoteka koje se koriste u 3D grafici. Biblioteka je implementirana u C-u i C++-u, te je dostupna pod izmijenjenom BSD licencom.[9]

1.6. Biblioteke stb

Stb je skupina manjih biblioteka. Implementirane su u C-u i dostupne su kao kod u javnoj domeni ili pod MIT licencom.[10] Jedna od biblioteka vrijednih spomenuti je

stb_image, koji se koristi za učitavanje slika.

1.7. Biblioteka OpenGL Mathematics

Biblioteka OpenGL Mathematics čini dostupnima funkcije za rad s objektima kao što su vektori, matrice i kvaternioni. Biblioteka je izrađena u C++-u i dostupna pod The Happy Bunny License ili MIT licencom.[11] Prvenstveno je namijenjena za korištenje s OpenGL-om i sadrži funkcije koje odgovaraju onima dostupnima u GLSL-u.

2. Prikaz krzna u stvarnom vremenu

Modeliranje svake pojedine dlake krzna bi preplavilo grafički sustav i time je neprikladno za prikaz u stvarnom vremenu. Zbog toga postoje pristupi koji korištenjem pojednostavljene geometrije i tekstura stvaraju iluziju krzna, a koji su dovoljno brzi da se mogu koristiti za prikaz u stvarnom vremenu.

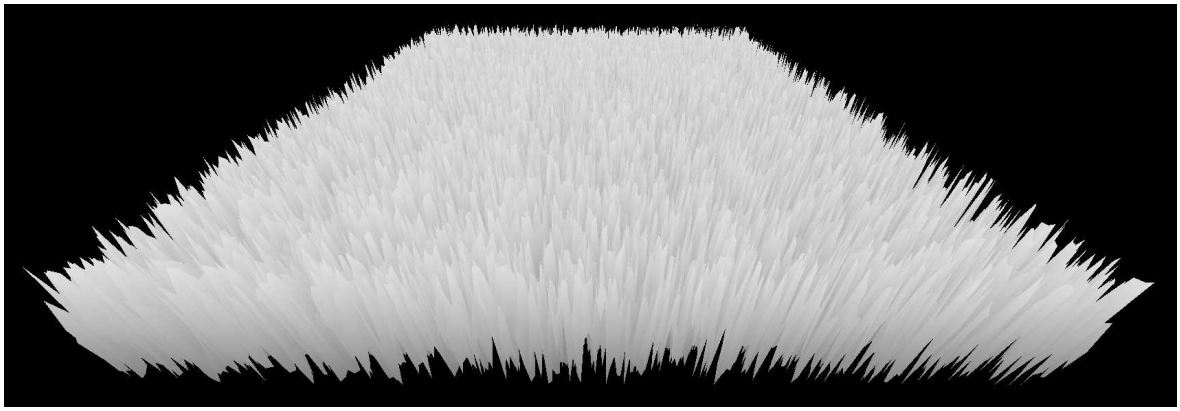
2.1. „Zgužvani" poligoni

Tehnika modeliranja ,,zgužvanim" poligonima[12] krzno modelira pomicanjem točaka modela od njegove površine za slučajnu udaljenost. Točke pomaknute za veću udaljenost predstavljaju dulje dlake, dok one pomaknute za manju udaljenost predstavljaju kraće dlake i prostor između dlaka. Slučajnim pomicanjem se aproksimira kaotičan izgled krznate površine. Primjer korištenja ove tehnike se može vidjeti na slici 2.1.

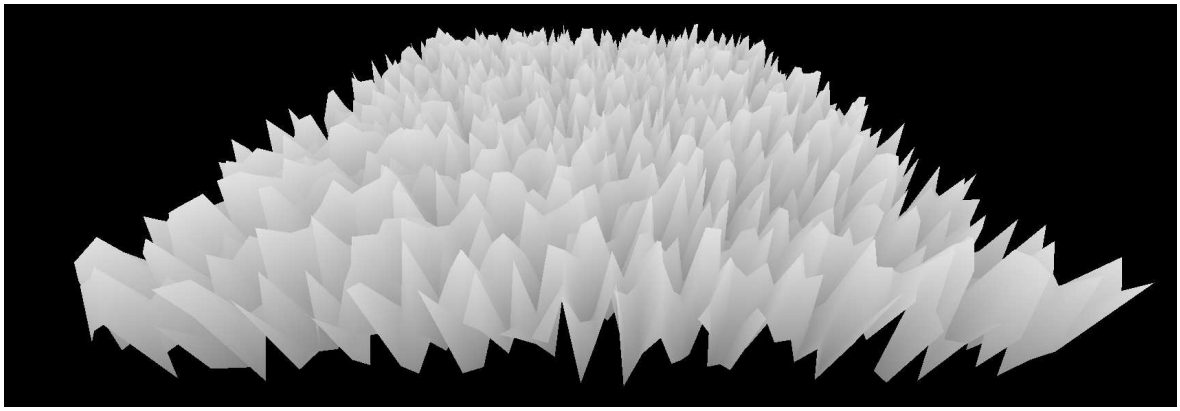
Ova je tehnika prikladna za implementaciju u sjenčaru vrhova: vrhovi se za slučajnu udaljenost pomiču duž normale ili nekog drugog vektora koji određuje smjer rasta dlaka. Nedostatak ove tehnike je što zahtijeva model s velikim brojem vrhova. Model koji nije dovoljno gusto pokriven vrhovima, poput onog na slici 2.1.b, neće korištenjem ove tehnike poprimiti izgled prekrivenosti tankim dlakama. Kod sjenčanja geometrije nastale ,,gužvanjem" važno je sjenčati na temelju normala izvornog modela, a ne ,,zgužvanog" jer će se zbog velikog broja sitnih detalja pojaviti šum i *aliasing* artefakti kao na slici 2.1.c

2.2. Teksturiranje ljuskama

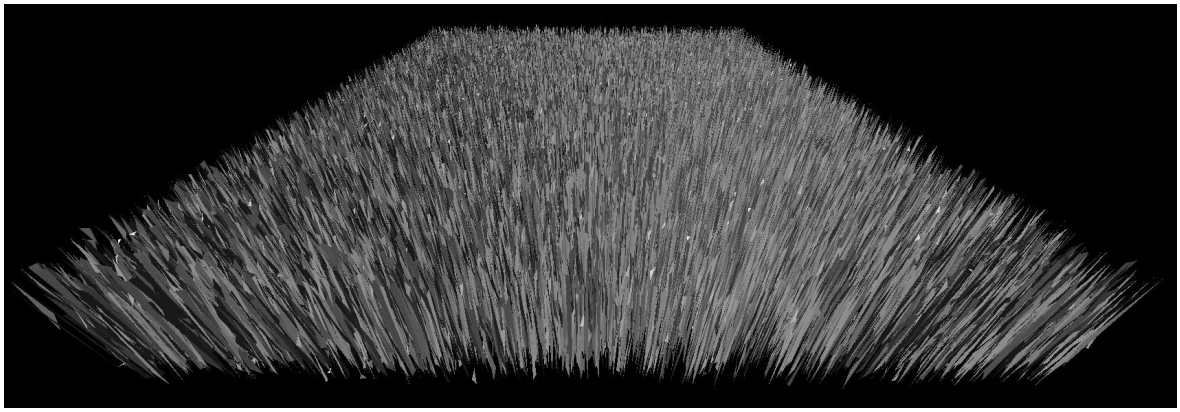
Tehnika teksturiranja ljuskama (engl. *shell texturing*) iznad modela dodaje paralelne plohe teksturirane djelomično prozirnom teksturom (slika 2.3.).[13] Ljuske se teksturiraju presjecima krzna. Što je ljuska udaljenija od modela, to je veći dio njene teks-



(a) Kvadrat gusto pokriven vrhovima.

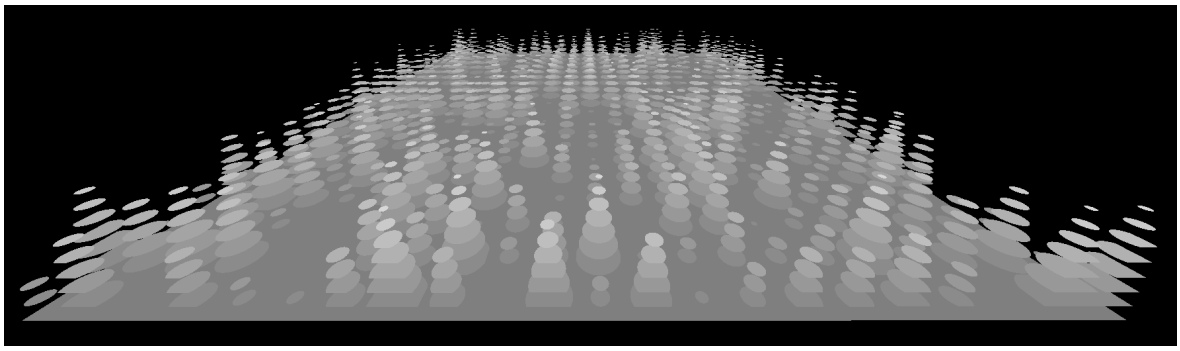


(b) Kvadrat rijetko pokriven vrhovima.

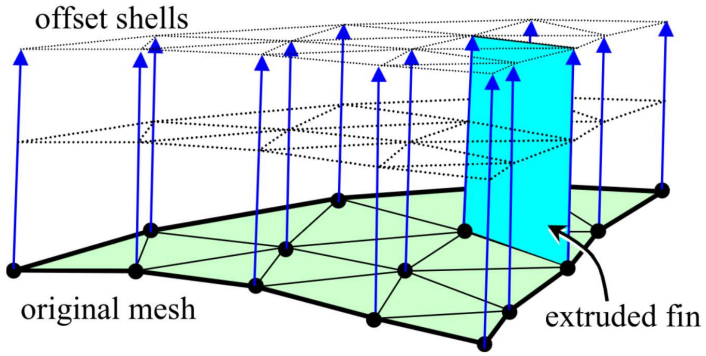


(c) Kvadrat gusto pokriven vrhovima i sjenčan po normalama nakon „gužvanja."

Slika 2.1. Kvadrat prekriven krznom modeliranim „zgužvanim" poligonima.



Slika 2.2. Kvadrat teksturiran ljuskama. Postavljen je velik promjer dlaka kako bi se jasno vidjele pojedine ljuske.



Slika 2.3. Geometrija za teksturiranje ljuskama.[13, str. 229] *Shells* su ljuske, dok je *fin* umetak.

ture proziran jer takav presjek sadrži manji broj dlaka i presječene su bliže svojem kraju, gdje su tanje. To se može vidjeti na slici 2.2. U sklopu ovog rada razmatrane su dvije implementacije teksturiranja ljuskama: implementacija u sjenčaru, te implementacija unaprijed generiranim volumnim teksturama.

Kako su ljuske paralelne s originalnim modelom, na rubovima modela će promatrač moći vidjeti praznine između ljusaka. Kako bi se taj problem ispravio, mogu se uz ljuske koristiti umetci (engl. *fins*), poligoni koji se postavljaju iznad bridova modela (slika 2.3.). Kako bi se sakrili umetci koji za neko gledište nisu na rubovima modela, prozirnost umetaka se smanjuje kako se normala umetka udaljava od smjera prema gledištu.[13] Umetci se mogu dinamički generirati za model korištenjem sjenčara geometrije, tako da se iznad svakog brida generiraju dva dodatna trokuta koji zajedno čine četverokut.

2.2.1. Teksturiranje ljuskama u sjenčaru

Tekstura ljusaka se može proizvesti proceduralno u sjenčaru fragmenata na temelju teksturnih koordinata. Prostor teksturnih koordinata može se podijeliti na ćelije i za svaku

ćeliju korištenjem šuma (npr. Perlinovog šuma) odrediti visina dlake koja se u njoj nalazi. Zatim se za svaki fragment ljuske određuje iznad koje se ćelije nalazi i na kojoj visini, ovisno o čemu se odredi hoće li biti proziran ili neproziran. Okrugli oblik dlake i njeno sužavanje prema vrhu mogu se postići odbacivanjem fragmenata ovisno o udaljenosti od središta ćelije.

Prednost ovakvog pristupa je u tome što nije potrebno unaprijed pripremiti volumnu teksturu za ljuske. Međutim, ovim je pristupom teže prikazati savinute ili kovrčave dlake.

2.2.2. Teksturiranje ljuskama s unaprijed generiranim teksturama

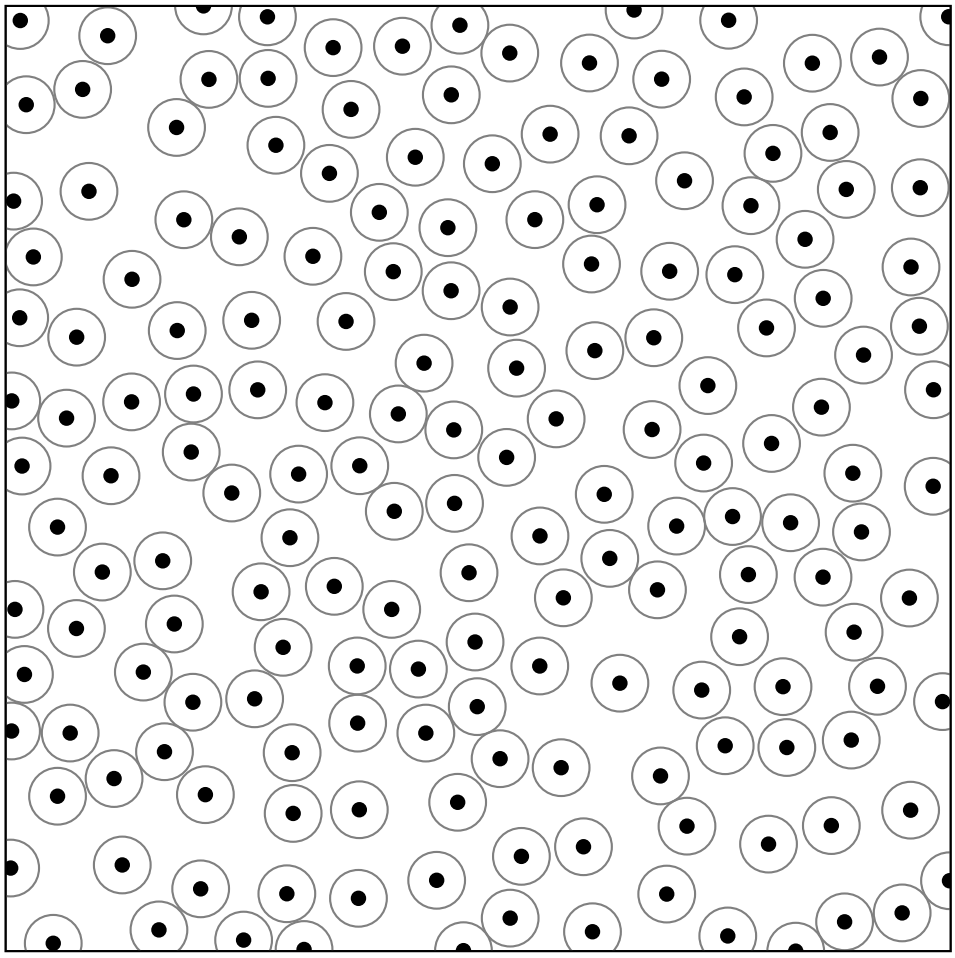
Za ljuske se može koristiti i unaprijed pripremljena volumna tekstura. Jedan način za dobiti takvu teksturu je izrada modela koji predstavlja dio krzna i iscrtavanje dijelova tog modela u teksture.

U ovom dijelu bit će opisan način na koji razvijeno programsko rješenje proceduralno generira teksture potrebne za ovaj pristup.

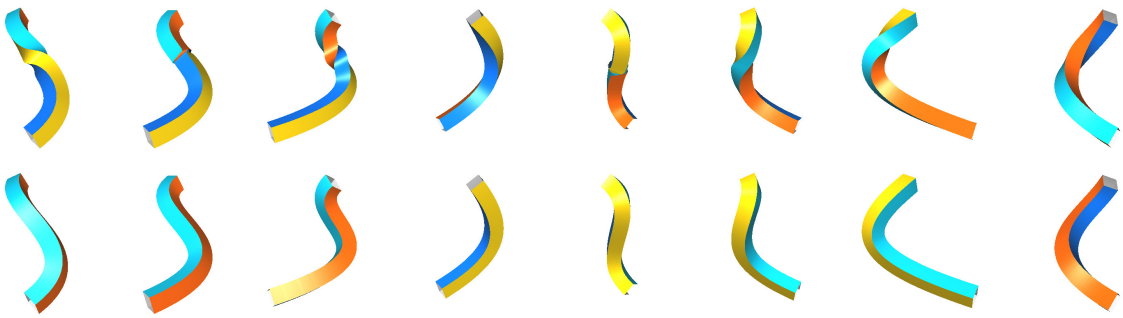
Prvo se na jediničnom kvadratu raspoređuju korijeni dlaka. Dlake se mogu rasporediti korištenjem plavog šuma, na primjer postupkom uzorkovanja Poissonovim diskom. To je postupak koji pokušava n dimenzionalan prostor ravnomjerno popuniti točkama bez stvaranja područja s nakupinama točaka ili prazninama, pritom osiguravajući da točke neće biti postavljene preblizu jedna drugoj. U sklopu ovog rada je za to korišten Bridsonov algoritam[14].

Slika 2.4. prikazuje primjer izvršavanja Bridsonovog algoritma na kvadratnom dijelu ravnine. Crni obrub predstavlja područje u kojem su točke raspoređene. Crne točke su postavljene točke. Bridsonov algoritam kao parametar prima minimalnu udaljenost između dvije točke. Sive kružnice oko točaka imaju promjer jednak toj minimalnoj udaljenosti. Valja primijetiti da se nikoje dvije kružnice ne sijeku, što znači da je algoritam osigurao da raspored točaka poštuje zadanu minimalnu udaljenost.

Nakon što se na jediničnom kvadratu odrede lokacije korijenja dlaka, potrebno je izgraditi geometriju dlaka. Putanje dlaka se modeliraju pomoću krivulja. Svaka dlaka



Slika 2.4. Primjer izvršavanja Bridsonovog algoritma na dijelu ravnine.



Slika 2.5. Plohe nastale pomicanjem kvadrata duž putanje: u gornjem retku korištenjem Frenetovog trobrida i u drugom retku korištenjem trobrida minimalne rotacije.[15, str. 4]

ima slučajno određenu duljinu i početnu rotaciju. Program pri generiranju geometrije dlaka krivulju ravnomjerno uzorkuje i za svaki uzorak u normalnoj ravnini konstruira obruč vrhova. Radijus obruča se s udaljavanjem od korijena dlake smanjuje kako bi se dlaka prema svojem vrhu stanjivala. Obruči susjednih uzoraka se povezuju u plohu. Dodatno, zbog metode korištene da se iz modela krzna dobiju teksture, za svaki obruč vrhova se u model dodaje i mnogokut s tim vrhovima. Metoda dvostrukog zrcaljenja[15] se koristi kako bi se minimizirala rotacija između uzastopnih obruča. Slika 2.5. pokazuje prednosti korištenja metode dvostrukog zrcaljenja naspram Frenetovog trobrida.

Ovisno o parametrima krivulje, dlake mogu poprimiti razne oblike, od potpuno ravnih (slika A.1.), preko blago zakrivljenih (slika A.2.) do kovrčavih (slika A.3.).

Potrebno je iz modela izraditi volumnu teksturu za ljuske. U sklopu ovog rada, korišten je OpenGL kako bi se model iscrtao u slojeve teksture. Za to je isti model iscrtan po jednom za svaki sloj teksture, ali s ravninama odsijecanja postavljenima tako da se svaki put obuhvati novi dio modela. Na primjer, ako se iscrtavaju tri sloja, u prvi će biti iscrtana samo donja trećina modela, u drugi samo srednja trećina, a u treći samo gornja trećina.

Za teksturu umetka se također iscrtava dio modela, ali se ne iscrtavaju presjeci duž vertikalne osi, već samo jedan duž jedne od horizontalnih osi.

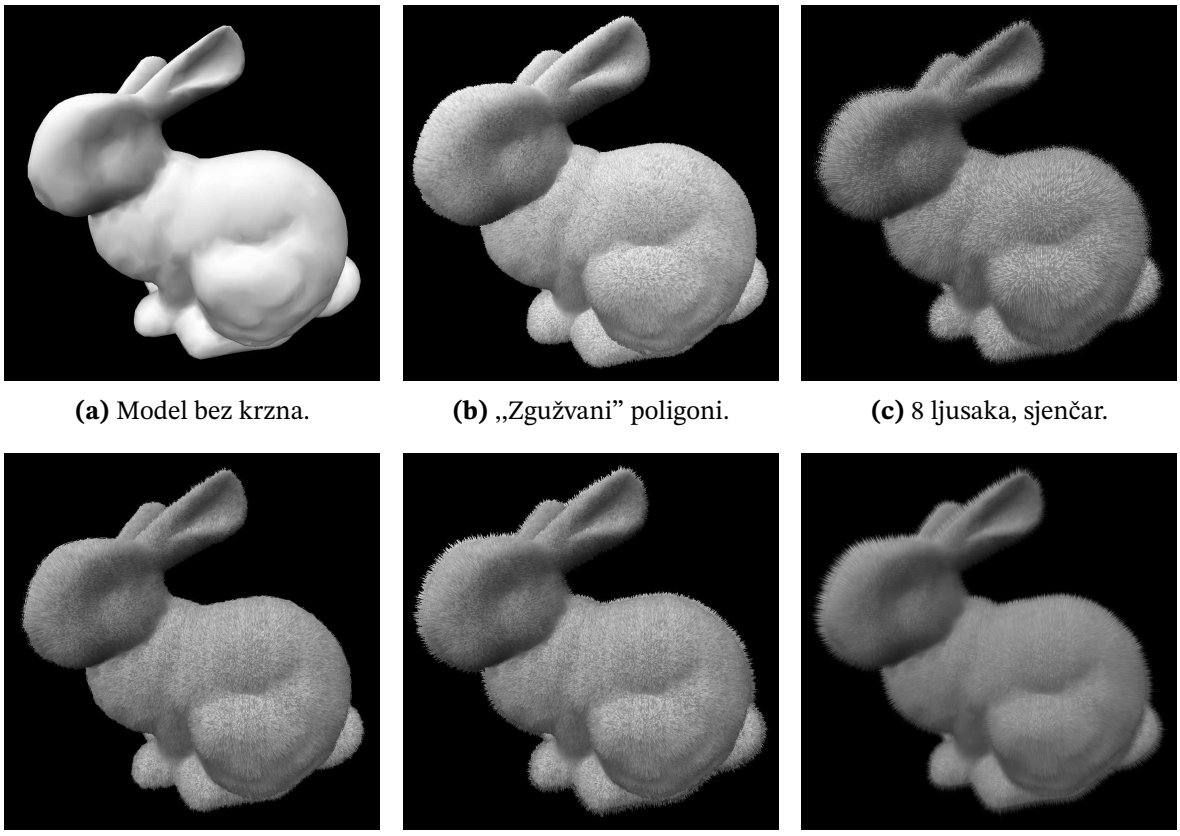
Nastale teksture su vidljive u privitku A

2.3. Usporedba pristupa

Slike 2.6. i 2.7. prikazuju isti model zeca na koji je različitim pristupima dodano krzno.

Modeliranje „zgužvanim" poligonima je jednostavno za implementirati, ali zahtijeva model gusto pokriven vrhovima. Za dobiti podjednaku gustoću dlaka koja je moguća s teksturiranjem ljuskama, model treba imati puno veći broj vrhova.

Implementacija teksturiranja ljuskama u sjenčaru za jednostavnije oblike dlaka može postići rezultate slične onima kod primjene volumnih tekstura. Jedan problem koji je potrebno riješiti kod korištenja sjenčara jest raspoređivanje dlaka po površini modela. Zbog jednostavnosti implementacije je za ovaj rad korišten pristup koji prostor dijeli na



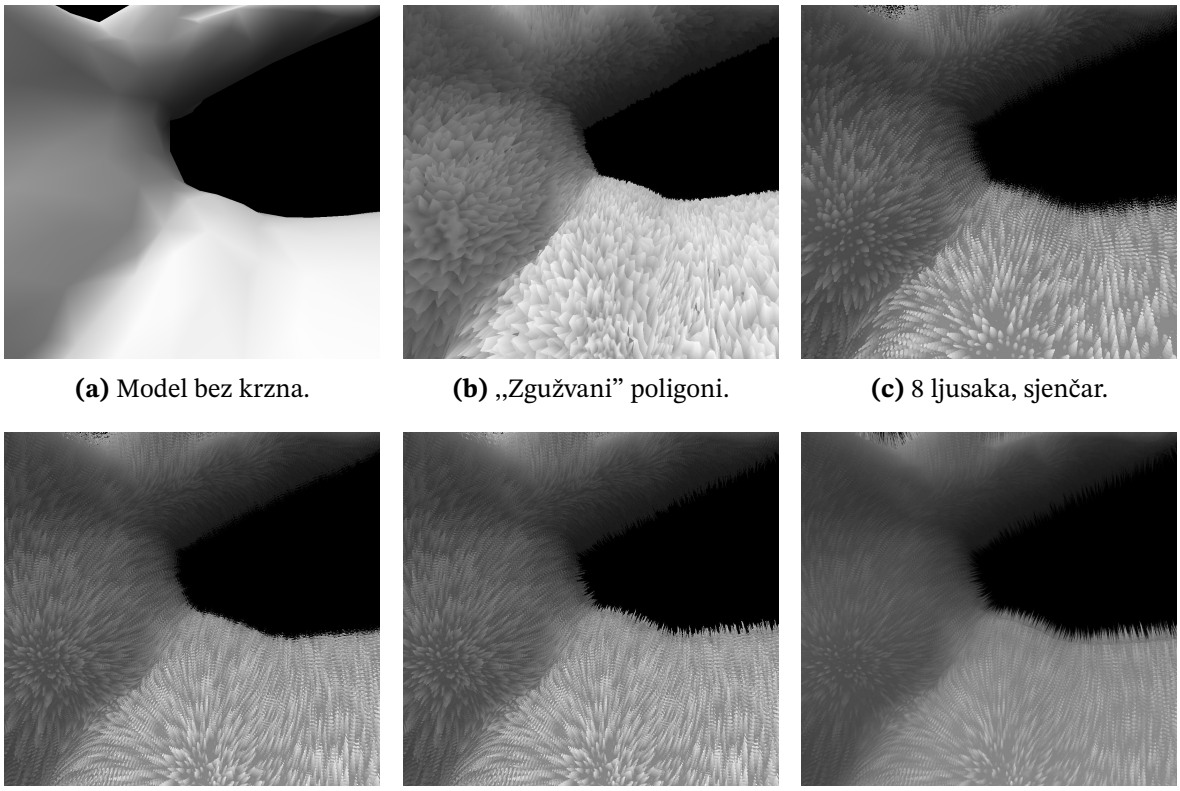
(a) Model bez krzna. **(b)** „Zgužvani" poligoni. **(c)** 8 ljusaka, sjenčar.

(d) 8 ljusaka, volumna tekstura, bez alfa miješanja. **(e)** 8 ljusaka i umetci, volumna tekstura, bez alfa miješanja. **(f)** 8 ljusaka i umetci, volumna tekstura, s alfa miješanjem.

Slika 2.6. Demonstracija različitih pristupa crtanju krzna na modelu zeca.

voksele, te korijene dlaka postavlja na temelju mreže voksela. Pod određenim kutevima gledanja se mogu vidjeti nizovi dlaka koje leže na istom pravcu, što narušava izgled životinjskog krzna, ali može biti prikladno za npr. tekstile poput ručnika i deka. Potencijalno rješenje za problem rasporeda dlaka jest izrada ponavljajuće teksture koja sadrži Voronojev dijagram. Ako se koriste modeli u kojima vrhovi imaju pridodijeljene teksturne koordinate, one se mogu koristiti za raspoređivanje dlaka umjesto pozicije vrha u prostoru za raspoređivanje dlaka. Kod korištenja pozicija vrhova treba obratiti pozornost na korištenje pozicije prije eventualnih transformacija radi animacije, pozicioniranja i sl.

Teksturiranje ljuskama s volumnim teksturama daje uvjerljive rezultate. U usporebi s generiranjem teksture u sjenčaru, korištenjem unaprijed pripremljenih volumnih tekstura je puno lakše prikazati složenije oblike dlaka, npr. kovrčave. Nedostatak ovog pristupa je što se unaprijed treba izgenerirati po jednu volumnu teksturu za svaki tip krzna koji se želi prikazati. Slično kao u prethodnom pristupu, ovdje je potrebno riješiti problem preslikavanja volumne teksture na ljuske. Za rad je korištena triplanarna pro-



(a) Model bez krzna. **(b)** „Zgužvani" poligoni. **(c)** 8 ljusaka, sjenčar.

(d) 8 ljusaka, volumna tekstura, bez alfa miješanja. **(e)** 8 ljusaka i umetci, volumna tekstura, bez alfa miješanja. **(f)** 8 ljusaka i umetci, volumna tekstura, alfa miješanje.

Slika 2.7. Demonstracija različitih pristupa crtanju krzna na modelu zeca, bliži pogled.

jekcija. Na mjestima gdje projekcija prelazi iz jednog smjera u drugi će doći do naglih skokova u teksturnim koordinatama, no taj je prijelaz prikriven detaljima u krznu. Jedan pristup za prekrivanje modela teksturom su tzv. *lapped textures*[16], no za njih je model potrebno unaprijed pripremiti.

Korištenje alfa miješanja kako bi se na rubovima modela postigla poluprozirnost daje bolje rezultate, ali sa sobom nosi probleme koji se i inače asociraju s poluprozirnošću. Korištenjem spremnika dubine se neprozirni objekti mogu iscrtavati na zaslon bilo kojim redoslijedom, ali se takav pristup ne može koristiti za poluprozirnim objekte. Ako scena sadrži više od jednog poluprozirnog objekta, potrebno ih je crtati fiksnim redoslijedom, po silaznoj udaljenosti od kamere. Isto vrijedi i ako postoji samo jedan poluproziran objekt koji je nekonveksan, dijelovi koji se prekrivaju se moraju crtati istim redoslijedom. Zbog toga alfa miješanje nije jednostavno za korištenje u sceni koja sadrži više od jednog poluprozirnog objekta.

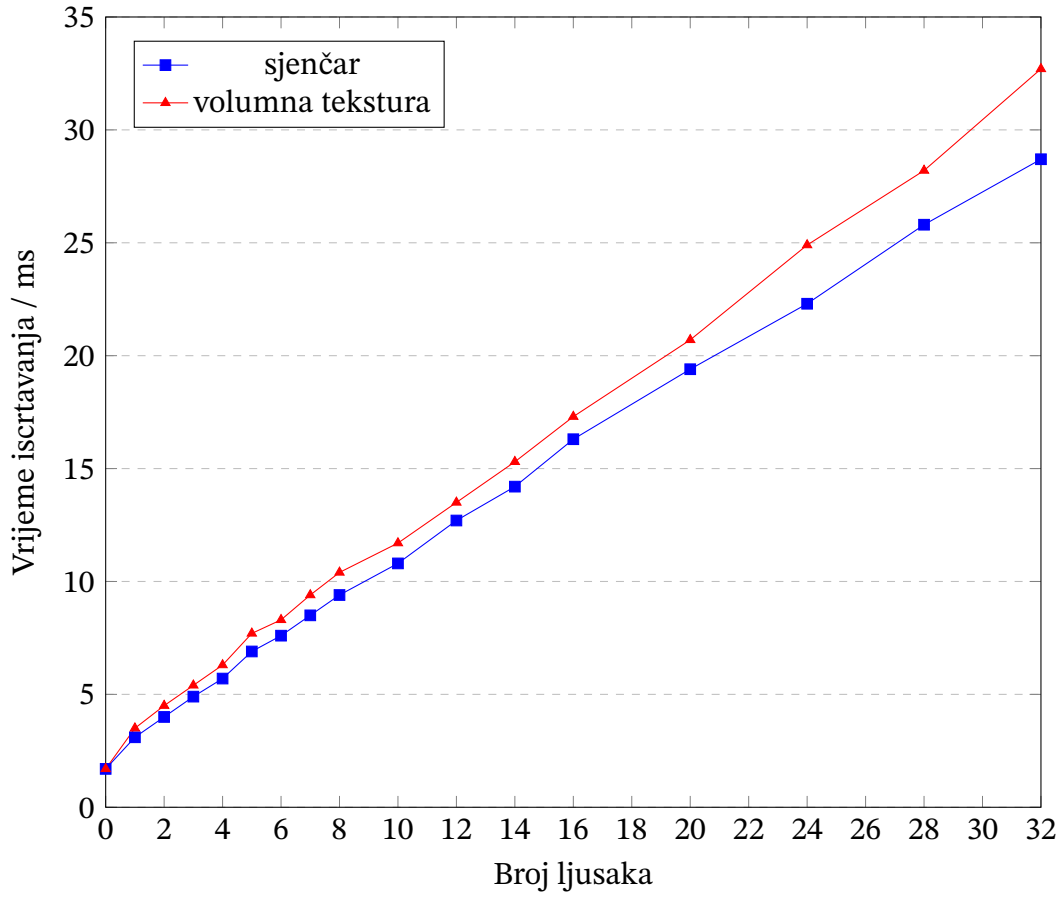
Tablica 2.1. sadrži vrijeme potrebno za iscrtavanje modela kod svakog od pristupa.

Pristup	Slika 2.6.	Slika 2.7.
(a)	1.3	2.4
(b)	19.1	19.2
(c)	8.7	10.6
(d)	9.8	15.5
(e)	12.0	17.2
(f)	15.4	19.6

Tablica 2.1. Vrijeme iscrtavanja primjera iz slika, u ms.

„Zgužvani" model je višestruko sporiji zbog velikog broja vrhova u modelu. Za svaku ljusku se model ponovno iscrtava, samo s vrhovima pomaknutima duž normale. Prema tome, očekivano je da će vrijeme potrebno za iscrtati model rasti linearno u odnosu na broj ljusaka. Ta je veza vidljiva iz slike 2.8. Umetci se ovdje generiraju sjenčarima geometrije, koji na dobrom dijelu grafičkih procesora loše utječu na performanse[17].

Mjerenja su provedena na računalu s AMD Ryzen 5 2500U procesorom (2 GHz) s integriranom Radeon Vega 8 grafikom, na operacijskom sustavu Arch Linux s AMDGPU upravljačkim programom za grafiku.



Slika 2.8. Graf vremena iscrtavanja krzna teksturiranjem ljusaka u ovisnosti o broju ljusaka.

3. Proceduralno generiranje uzorka krzna

Računalna grafika često koristi šum za generiranje tekstura. Za to se često koristi Perlinov šum. Perlinov šum je izvorno n-dimenzionalan prostor popločavao n-kockama. To je kao rezultat imalo vidljive artefakte duž koordinatnih osi. Usto algoritam interpolira između vrijednosti određenih vrhovima n-kocaka, koje imaju 2^n vrhova, zbog čega algoritam ima eksponencijalnu asimptotsku vremensku složenost $O(2^n)$ u broju dimenzija domene.

Perlin je kasnije razvio simpleks šum (engl. *simplex noise*), koji funkcionira na sličnom principu kao i izvorni Perlinov šum. Međutim, simpleks šum n-dimenzionalan prostor popločava n-simpleksima. Time se izbjegavaju artefakti prisutni u izvornom šumu. Kako n-simpleks ima $n+1$ vrhova, složenost algoritma više nije određena brojem vrhova, već je mnogo povoljnija kvadratna u broju dimenzija, $O(n^2)$.[18] Na slici 3.1.a je vidljiv trodimenzionalni simpleks šum primijenjen na model zeca.

Daljnjom obradom šuma mogu se dobiti novi šumovi koji s raznim primjenama u proceduralnom generiranju tekstura i geometrije, na primjer fraktalni šum ili šum koji generira hrptove (engl. *ridge noise*). Korištenjem step funkcije se jedan dio modela može obojati jednom bojom, a drugi dio drugom (slika 3.1.b). Glatki prijelazi između boja mogu se dobiti korištenjem smoothstep funkcije (slika 3.1.c).

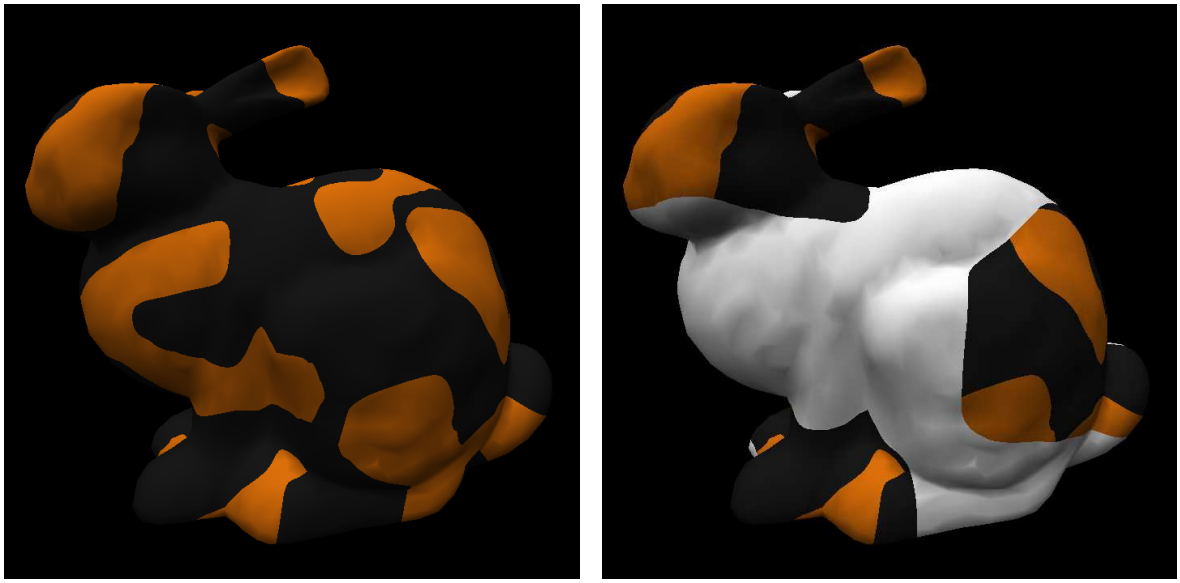
Kod nekih životinja kao što su mačke se u krznu može pojaviti crno-narančasti uzorak kornjačevine (*tortoiseshell*). On se može modelirati korištenjem jedne funkcije šuma sa step funkcijom (slika 3.2.a). Kada uz uzorak kornjačevine mačka ima i velik dio krzna bijele boje, takav se trobojan uzorak naziva *calico*. To se može modelirati još jednom primjenom šuma sa step funkcijom, koji će kad je vrijednost šuma ispod nekog praga preuzeti boju iz prethodnog koraka, a u suprotnom postaviti bijelu (slika 3.2.b).

(a) Običan simpleks šum.

(b) Simpleks šum obrađen step fukcijom.

(c) Simpleks šum obrađen smoothstep funkcijom.

Slika 3.1. Model zeca teksturiran simpleks šumom, bez sjenčanja.



(a) Uzorak kornjačevine.

(b) *Calico* uzorak.

Slika 3.2. Model zeca teksturiran uzorcima, sjenčano.

4. Opis razvijene aplikacije

Razvijena je desktop aplikacija (slika 4.1.) koja u prozoru prikazuje 3D model i dva pod-prozora s postavkama. Aplikacija kao parametar komandnog retka prima putanju do 3D modela koji će biti prikazan. Model mora biti u OBJ formatu. Izvršna datoteka mora biti pokrenuta u direktoriju u kojem je prisutan assets direktorij sa sjenčarima.

Pritiskom miša na dio prozora u kojem je prikazan model pokazivač miša nestaje i pomicanjem miša se može rotirati kamera. Korištenjem tipki W, A, S i D se kamera može pomicati naprijed, lijevo, nazad i desno, respektivno. Tipkama razmak i lijevi shift se kamera može pomicati gore i dolje. Pritisak tipke F izvor dodatne sile postavlja na poziciju kamere. Na pritisak tipke escape se pokazivač miša opet pojavljuje i aplikacija izlazi iz načina rotacije kamere.

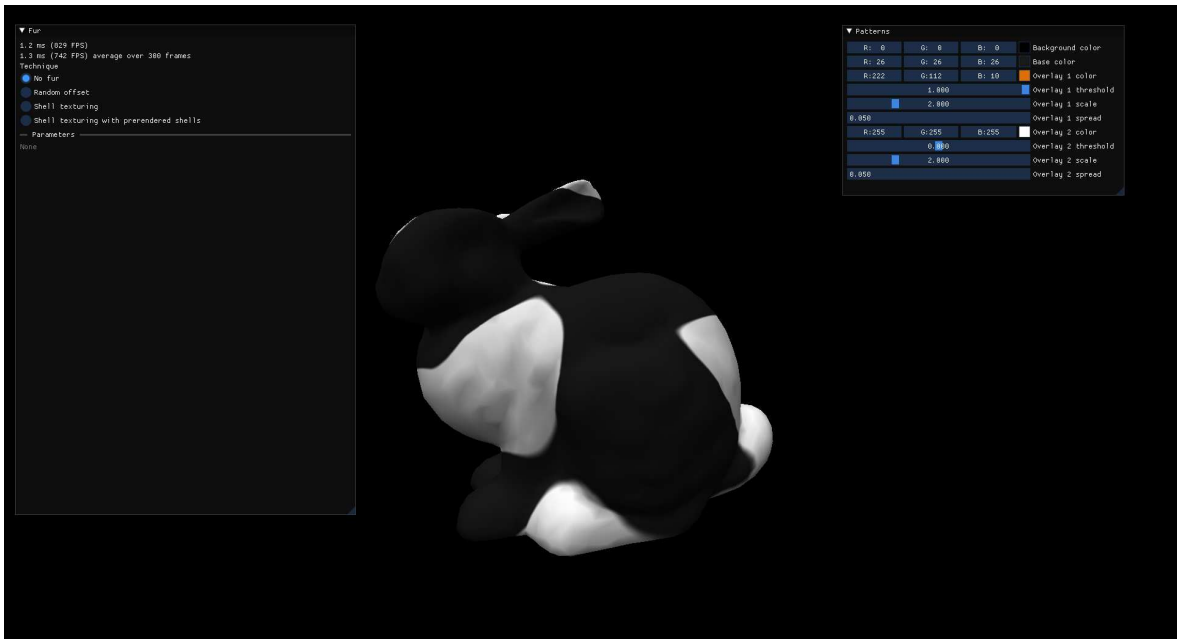
4.1. Postavke u prozoru za upravljanje prikazom krzna

Prozor za upravljanje prikazom krzna (slika 4.2.) sadrži postavke koje mijenjaju način na koji je prikazano krzno.

Na vrhu se prikazuje vrijeme potrebno za crtanje jednog *framea* u milisekundama i koliko sličica u sekundi može pri toj brzini biti prikazano. Ispod toga se nalaze isti podaci ali za prosjek posljednjih 300 *frameova*.

Četiri opcije su dostupne za izbor tehnike (*technique*): bez prikaza krzna (*no fur*), „zgužvani poligoni” (*random offset*), teksturiranje ljuskama u sjenčaru (*shell texturing*) i teksturiranje ljuskama s volumnom teksturom (*shell texturing with prerendered shells*). Ovisno o izabranoj tehnici će se relevantne postavke prikazati ili sakriti.

Fur length određuje maksimalnu duljinu krzna. *Global force* utječe na smjer krzna,



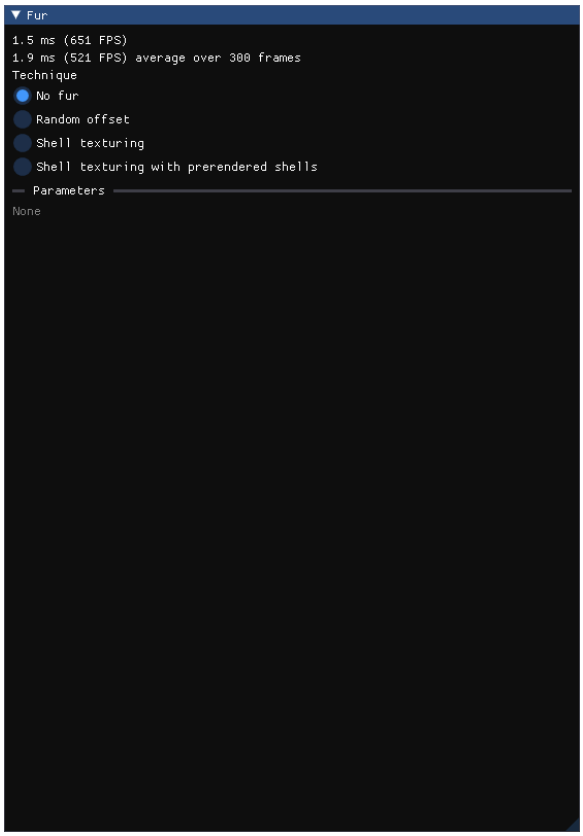
Slika 4.1. Slika izrađene aplikacije

na primjer za simulaciju savijanja dlaka pod utjecajem gravitacije. *Local force location* određuje odakle će djelovati dodatna sila na model, na primjer za simulaciju puhanja ili usisavanja (slika 4.3.). *Local force* određuje snagu dodatne sile na model.

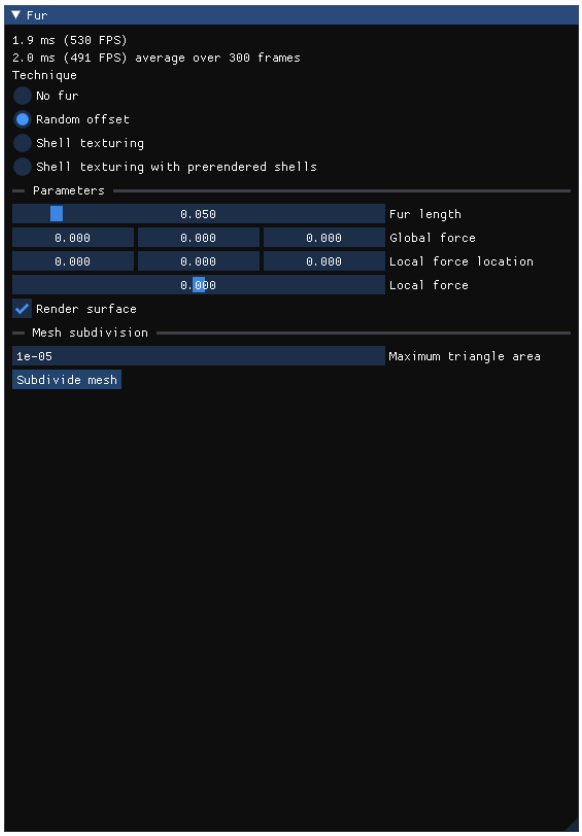
Render surface omogućava, uz „zgužvani model," prikaz i nezgužvanog. Time se mogu sakriti eventualne rupe u „zgužvanom" modelu. Kako učitani model možda neće imati dovoljan broj vrhova, u aplikaciji je implementirana mogućnost dodavanja vrhova podjelom postojećih trokuta na manje dijelove. Pritiskom na tipku *Subdivide mesh* se taj proces započinje. Algoritam usitnjava trokute čija je površina veća od one zadane u opciji *Maximum triangle area* i to ponavlja dokle god takvih trokuta ima.

Shell count određuje broj ljusaka u prikazu. *Shell voxel scale* određuje broj voksela po jedinici duljine za pristup teksturiranja ljuskama kroz sjenčar. *Shell texture scale* skalira teksturu ljusaka za pristup koji koristi volumnu teksturu. *Fin texture scale* skalira teksturu umetaka. *Alpha threshold* određuje alfa vrijednost ispod koje će poluprozirni pikseli biti odbačeni. *Alpha blending* uključuje alfa miješanje. *Render fins* omogućava prikaz umetaka.

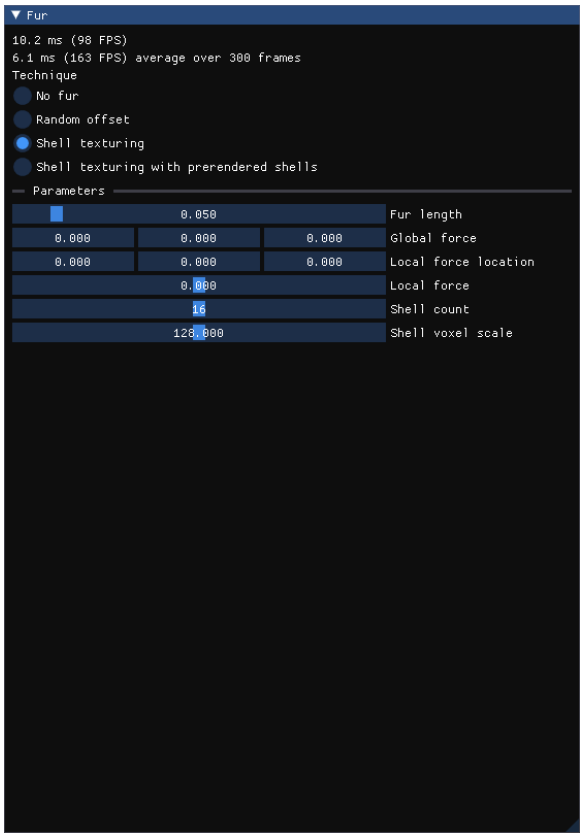
Shell texture size određuje širinu i visinu teksture ljusaka. *Shell texture layers* određuje broj slojeva u teksturi ljusaka. Ovaj bi parametar trebao biti jednak broju ljusaka u prikazu. *Fin texture width* i *Fin texture height* određuju veličinu teksture umetaka.



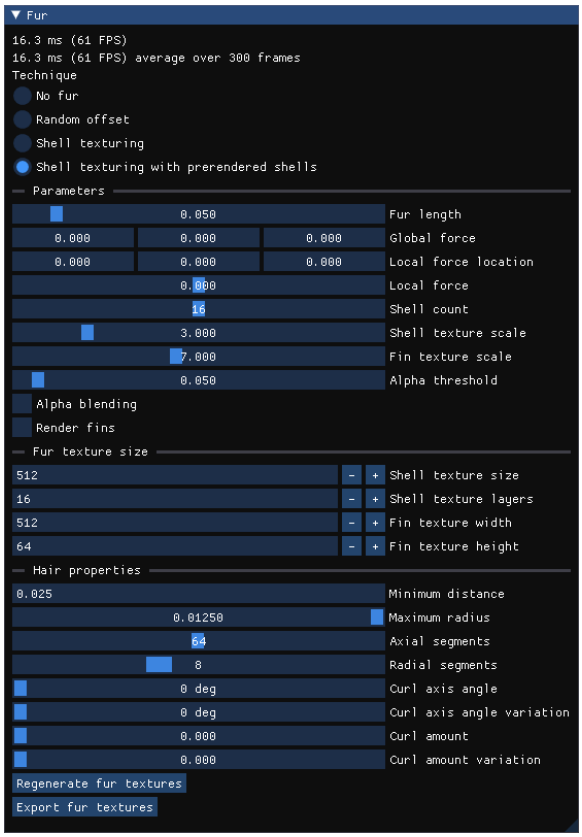
(a) Prikaz krzna ugašen.



(b) Prikaz „zgužvanim” poligonima.

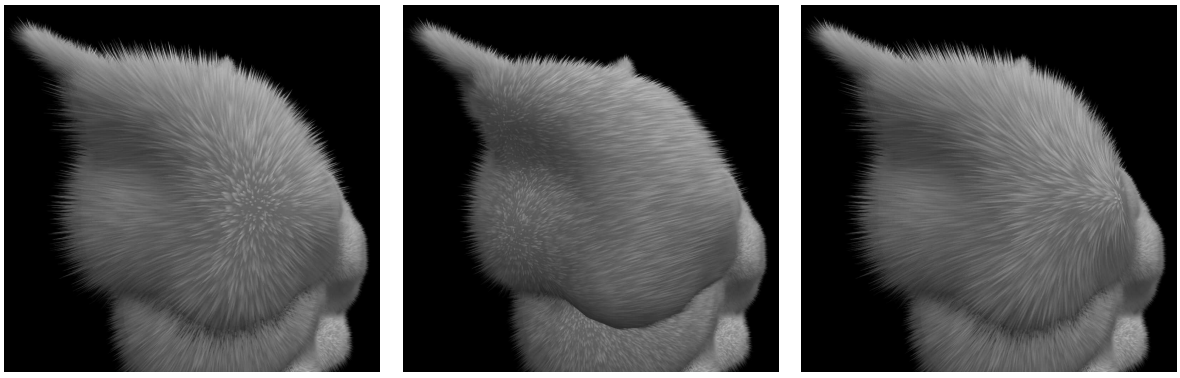


(c) Prikaz ljuskama u sjenčaru.



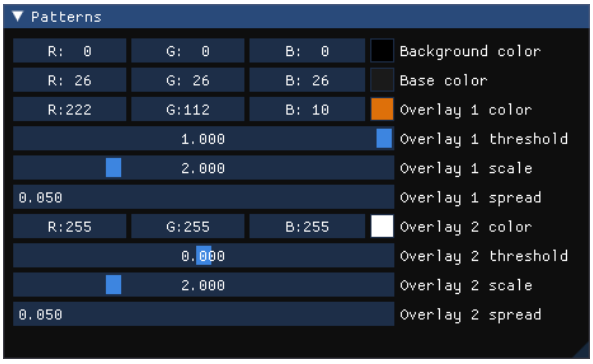
(d) Prikaz ljuskama s volumnim teksturama.

Slika 4.2. Prozor za upravljanje prikazom krzna.



(a) Bez primjene dodatne sile. **(b)** Simulacija usisavanja. **(c)** Simulacija puhanja.

Slika 4.3. Utjecaj dodatne sile na model.



Slika 4.4. Prozor za upravljanje uzorkom.

Minimum distance određuje minimalnu udaljenost između dlaka kod izvršavanja Bridsonovog algoritma za generiranje modela krzna. *Maximum radius* određuje maksimalni polumjer dlake u modelu krzna. *Axial segments* određuje broj segmenata dlake u modelu krzna duž njene duljine. *Radial segments* određuje koji n-terokut će biti korišten za presjek dlake u modelu krzna. *Curl axis angle* određuje kut između normale i osi oko koje se dlake uvijaju. *Curl axis angle variation* određuje maksimalno slučajno odstupanje tog kuta. *Curl amount* određuje koliko se jako dlake uvijaju. *Curl amount variation* određuje maksimalno slučajno odstupanje količine uvijanja. *Regenerate fur textures* pokreće proceduralno generiranje modela krzna i tekstura iz njega. *Export fur textures* zadnje generirane teksture pohranjuje na disk kao PNG datoteke.

4.2. Postavke u prozoru za upravljanje uzorkom

Prozor za upravljanje uzorkom (slika 4.4.) sadrži postavke koje mijenjaju uzorak krzna.

Base color, *Overlay 1 color* i *Overlay 2 color* određuju boje pojedinih dijelova krzna.

Overlay 1 threshold i *Overlay 2 threshold* određuju prag u izlazu funkcije šuma za pojedine dijelove krzna. *Overlay 1 scale* i *Overlay 2 scale* skaliraju šum kojim se generira uzorak. *Overlay 1 spread* i *Overlay 2 spread* određuju koliko će nagao biti prijelaz između boja.

4.3. Prevođenje iz izvornog koda

Za izgradnju su potrebni:

- CMake, verzija 3.21 ili novija,
- Make,
- prevoditelj za C++ koji podržava C++20,
- verzija 5.2.5 biblioteke Assimp,
- verzija 1.89.6 biblioteke Dear ImGui,
- verzija 2.2.0 biblioteke GLEW,
- verzija 0.9.9.8 biblioteke GLM,
- verzija 2.26.5 biblioteke SDL 2,
- biblioteka stb i
- podrška grafičkog procesora za OpenGL 4.5.

Pokretanjem naredbe `cmake <put>` će se u trenutnom direktoriju izraditi Makefile, gdje `<put>` predstavlja putanju do direktorija u kojem se nalazi CMakeLists.txt datoteka projekta. Zatim se pokretanjem naredbe `make` izvorni kod prevodi i generira se izvršna datoteka programa.

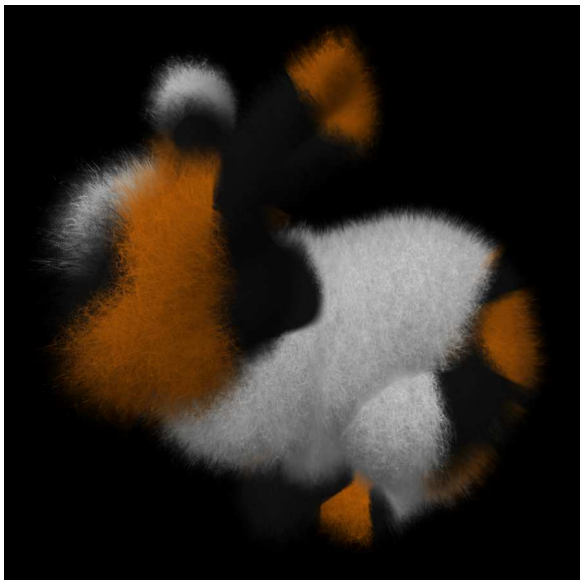
Zaključak

Prikaz krzna i sličnih materijala predstavlja zanimljiv problem za grafiku u stvarnom vremenu. U razvoju igara je pogotovo važno koristiti tehnike koje su računski dovoljno jednostavne da ne narušavaju performanse.

U ovome radu opisane tehnike su primijenjive za prikaz ne samo životinjskog krzna (slika 4.5.a), već i umjetnih predmeta (slika 4.5.b) i niske trave (slika 4.5.c). Upravljanjem parametrima se mogu postići različiti izgledi krzna. To ove metode čini ne samo prikladnima u aspektu performansi, već i fleksibilnima u pogledu realizacije umjetničke vizije.

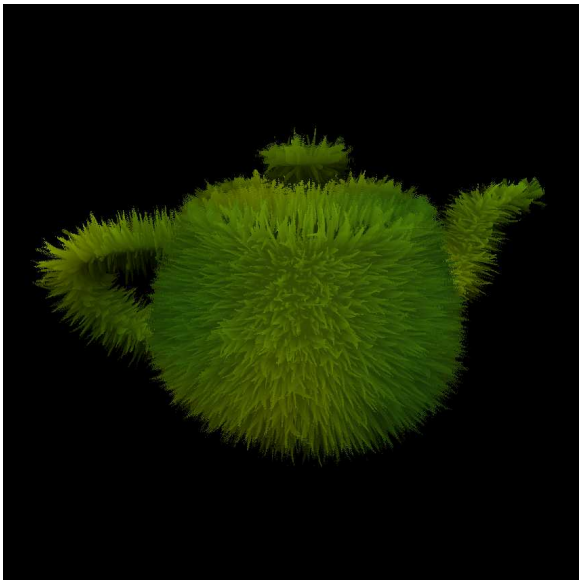
Razvijena aplikacija krzno postavlja na proizvoljan 3D model. Dodatnom pripremom modela bi se mogla postići još veća kontrola nad krznom. Na primjer, vrhovi modela bi mogli sadržavati parametre koji opisuju duljinu krzna i smjer rasta. U ovom su radu opisane samo osnove tehnika, one se mogu po potrebi proširivati radi postizanja željenih efekata.

Iako su opisane tehnike već stare nekoliko desetljeća, još uvijek su u upotrebi i kao takve su korisno znanje za bavljenje računalnom grafikom, posebice u stvarnom vremenu.



(a) Čupavi *calico* zec.

(b) Plišani medvjed.



(c) Travnati čajnik.

Slika 4.5. Prikazi modela u izrađenoj aplikaciji.

Literatura

[1] M. Bhave i S. Patekar, “Object oriented programming with C++”, https://www.oreilly.com/library/view/object-oriented-programming/9789332503663/xhtml/head-0045.xhtml, 2012., mrežno; stranica posjećena: veljača 2024.

[2] H. Sutter, “C++ IS schedule, 5. izdanje”, https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p1000r5.pdf, ISO/IEC JTC1/SC22/WG21 - Odbor za norme C++-a, dokument P1000R5, 2023., mrežno; stranica posjećena: veljača 2024.

[3] J. Bastien i A. Meredith, “Removing Garbage Collection Support, 2. izdanje”, https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2021/p2186r2.html, ISO/IEC JTC1/SC22/WG21 - Odbor za norme C++-a, prijedlog P2186R2, 2021., mrežno; stranica posjećena: veljača 2024.

[4] “About The Khronos Group”, https://www.khronos.org/about/, The Khronos Group Inc., 2014., mrežno; stranica posjećena: veljača 2024.

[5] Suradnici na OpenGL wiki, “History of OpenGL”, https://www.khronos.org/opengl/wiki/History_of_OpenGL, 2022., mrežno; stranica posjećena: veljača 2024.

[6] M. Segal i K. Akeley, *The OpenGL Graphics System: A Specification (Version 4.5 (Core Profile))*, https://registry.khronos.org/OpenGL/specs/gl/glspec45.core.pdf, The Khronos Group Inc., 2017., mrežno; stranica posjećena: veljača 2024.

[7] Suradnici na SDL-u, “About SDL”, https://www.libsdl.org/index.php, mrežno; stranica posjećena: veljača 2024.

[8] Suradnici na Dear ImGuiju, “Dear ImGui”, https://github.com/ocornut/imgui, 2023., mrežno; stranica posjećena: veljača 2024.

[9] Suradnici na Open Asset Import Libraryju, “Open asset import library : Documentation & faq”, https://assimp.sourceforge.net/FAQ.html, mrežno; stranica posjećena: veljača 2024.

[10] Suradnici na stb-u, “stb single-file public domain libraries for C/C++”, https://github.com/nothings/stb, mrežno; stranica posjećena: veljača 2024.

[11] Suradnici na OpenGL Mathematics, “OpenGL mathematics (GLM)”, https://github.com/g-truc/glm, mrežno; stranica posjećena: veljača 2024.

[12] Ž. Mihajlović, “Predavanja iz računalne grafike: 7. prikaz detalja površine”, https://www.zemris.fer.hr/predmeti/ra/predavanja/7_bump.pdf, mrežno; stranica posjećena: veljača 2024.

[13] J. Lengyel, E. Praun, A. Finkelstein, i H. Hoppe, “Real-time fur over arbitrary surfaces”, u *Proceedings of the 2001 symposium on Interactive 3D graphics*, 2001., str. 227–232.

[14] R. Bridson, “Fast poisson disk sampling in arbitrary dimensions.” *SIGGRAPH sketches*, sv. 10, br. 1, str. 1, 2007.

[15] W. Wang, B. Jüttler, D. Zheng, i Y. Liu, “Computation of rotation minimizing frames”, *ACM Transactions on Graphics*, sv. 27, br. 1, str. 1–18, 2008.

[16] E. Praun, A. Finkelstein, i H. Hoppe, “Lapped textures”, u *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000., str. 465–470.

[17] J. Barczak, “Why geometry shaders are slow (unless you’re Intel)”, http://www.joshbarczak.com/blog/?p=667, 2015., mrežno; stranica posjećena: veljača 2024.

[18] S. Gustavson, “Simplex noise demystified”, Linköping University, izvještaj, 2005.

Sažetak

Proceduralno generiranje i prikaz krzna u stvarnom vremenu

Borna Cafuk

Rad predstavlja proceduralne tehnike za prikaz krzna u računalnoj grafici u stvarnom vremenu. Posebno je naglašena metoda teksturiranja ljuskama. Opisane su dvije varijante te metode: jedna koja generira teksturu u sjenčaru i druga koja koristi unaprijed generiranu volumnu teksturu. Prikazan je proceduralan postupak generiranja volumne teksture krzna. Također je opisana i metoda za generiranje jednostavnog uzorka krzna korištenjem šuma. Rad opisuje razvijenu aplikaciju za prikaz krzna na proizvoljnom trodimenzionalnom modelu.

Ključne riječi: prikaz krzna; prikaz u stvarnom vremenu; volumne teksture; teksturiranje ljuskama

Abstract

Procedural generation and real-time fur rendering

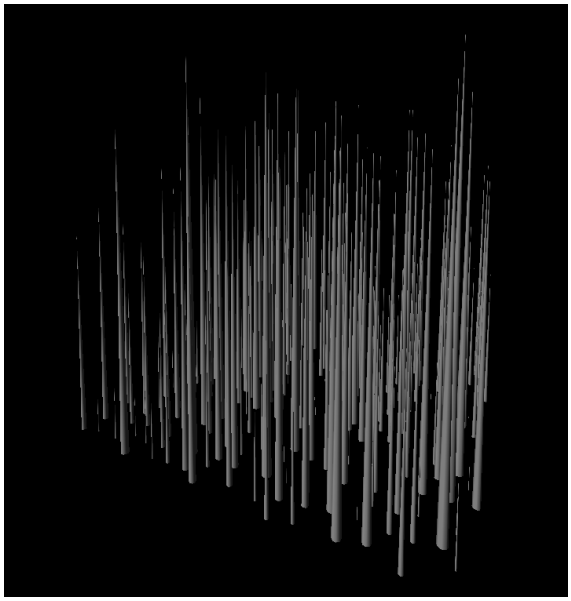
Borna Cafuk

The thesis presents procedural techniques for rendering fur in real-time computer graphics. Special attention is paid to the shell texturing method. Two variants of the method are described: one generates the texture in a shader and the other uses a volume texture generated ahead of time. A procedural process for generating the fur volume texture is shown. Also described is a method for generating a simple fur pattern using noise. The thesis describes the developed application for displaying fur on an arbitrary three-dimensional model.

Keywords: fur rendering; real-time rendering; volume textures; shell texturing

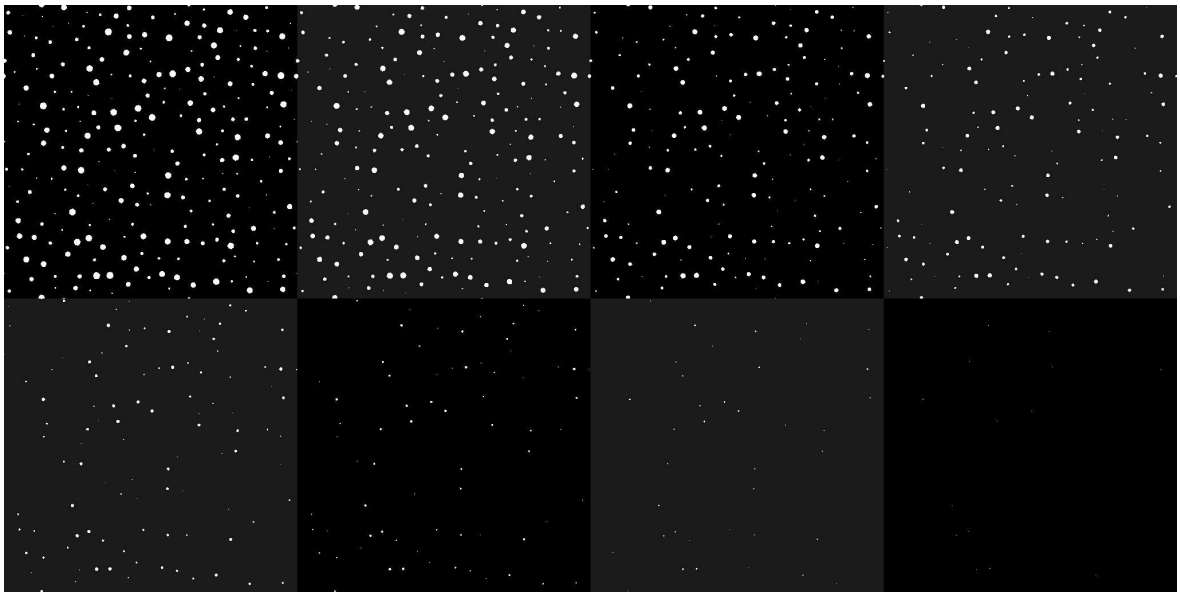
Privitak A: Primjeri modela krzna

Ovaj se dodatak sastoji od slika proceduralno generiranih modela krzna i iz njih dobivenih tekstura. Na prvoj je podslici prikazan sam model krzna. Na drugoj je podslici tekstura umetka. Treća podslika prikazuje slojeve volumne teksture ljusaka.



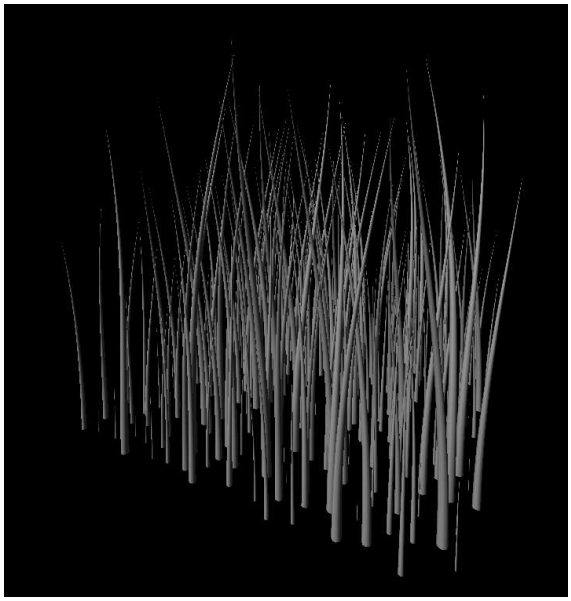
(a) Model krzna.

(b) Tekstura umetka.



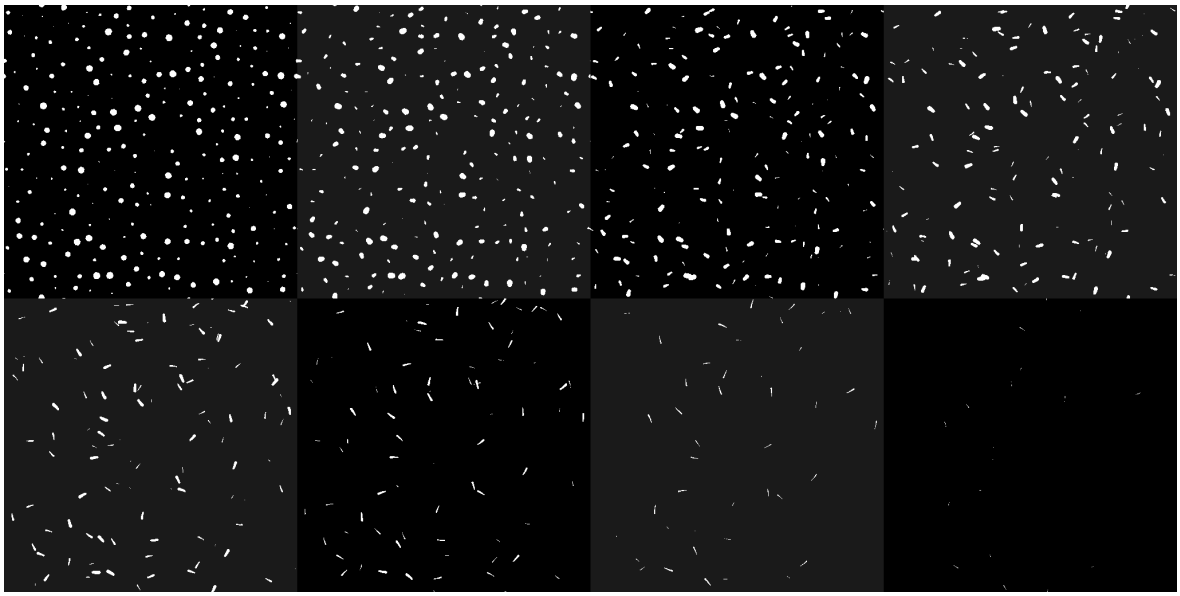
(c) Slojevi volumne teksture ljusaka. Tekstura se sastoji od osam slojeva.

Slika A.1. Model potpuno ravnog krzna i iz njega dobivene teksture.



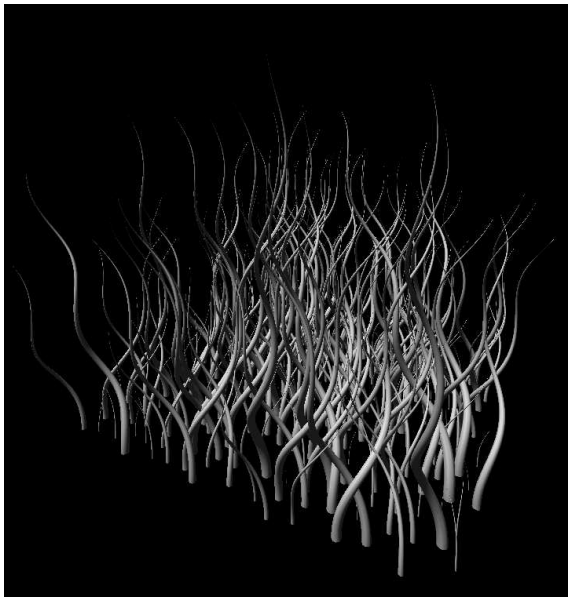
(a) Model krzna.

(b) Tekstura umetka.



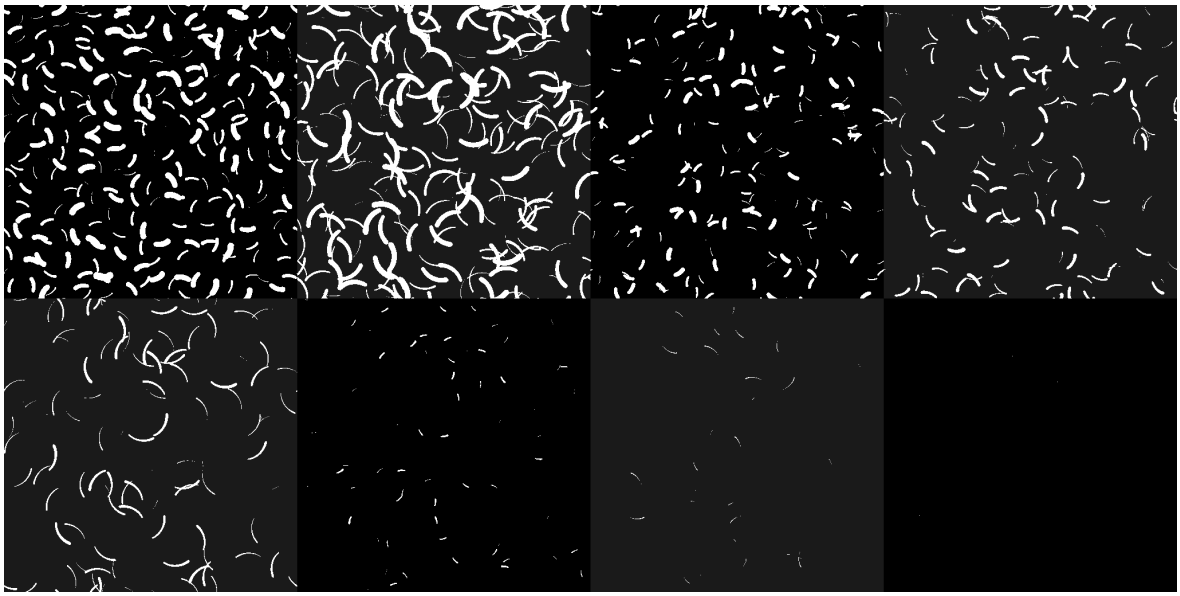
(c) Slojevi volumne teksture ljusaka. Tekstura se sastoji od osam slojeva.

Slika A.2. Model blago zakrivljenog krzna i iz njega dobivene teksture.



(a) Model krzna.

(b) Tekstura umetka.



(c) Slojevi volumne teksture ljusaka. Tekstura se sastoji od osam slojeva.

Slika A.3. Model kovrčavog krzna i iz njega dobivene teksture.