SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI ZADATAK br. 1703

Generiranje armature za trodimenzijske modele objekata

Josip Tomić

Zagreb, lipanj 2018

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 9. ožujka 2018.

DIPLOMSKI ZADATAK br. 1703

Pristupnik: **Josip Tomić (0036479683)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Generiranje armature za trodimenzijske modele objekata**

Opis zadatka:

Proučiti načine izrade armature objekata zadanih trodimenzijskim poligonalnim mrežama te načine povezivanja armature s objektima. Proučiti mogućnosti automatizacije procesa definiranja armature. Za proizvoljne trodimenzijske modele razraditi aplikaciju koja će omogućiti automatiziranje postupka izrade armature te računalni prikaz skeletnih animacija. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama.
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.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Sadržaj

Uvod

Računalna animacija veoma je zastupljena u današnje vrijeme. S njome se možemo susresti na mnogo mjesta, od specijalnih efekata u filmskoj industriji, kompleksnih kretnji u video igrama do specijaliziranih programskih rješenja u proizvodnji. Bilo da se radi fotorealističnim ili jednostavnijim modelima, proces animiranja zna biti dugačak i mukotrpan. Jedan od koraka izrada animacije je kreiranje armature za model koji se animira. Iako ovaj korak ne mora trajati dugo, zbog svoje prirode i potrebnih ponavljajućih akcija mogao bi se generalizirati i automatizirati.

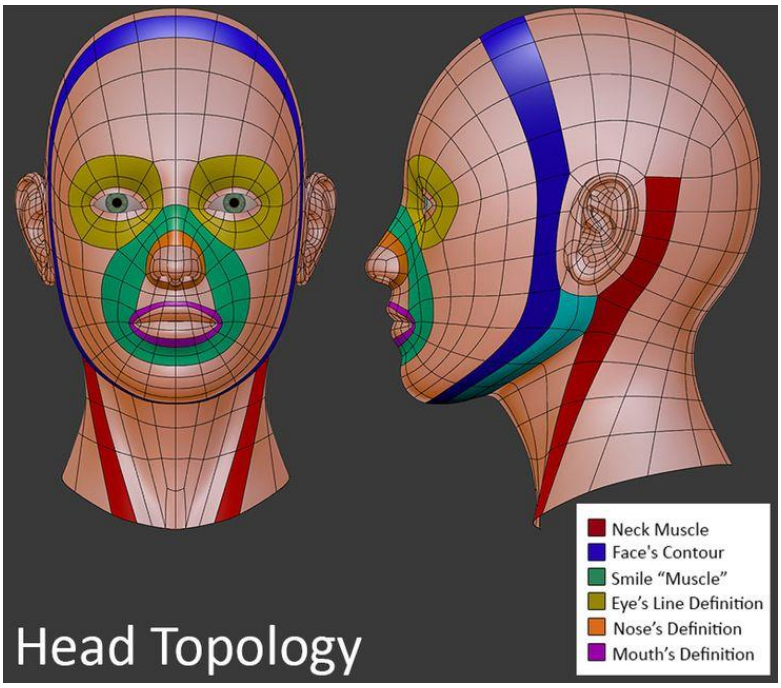
Ovaj rad opisuje jedan mogući pristup proceduralnom generiranju armature za gotovi trodimenzionalni model. Kreirani proizvod omogućiti će korisniku brzo i jednostavno kreiranje željene armature te utjecanje na način kreiranja pomoću definiranih parametara. Rješenje je izrađeno u programskom alatu Blender.

Model i armatura

Trodimenzionalni model je struktura koji određeni objekt iz stvarnoga svijeta opisuje u trodimenzionalnom prostoru na računalu. Postoji par različitih načina opisivanja modela, no najčešći je korištenjem mreže točaka i poligona (eng. *mesh*).

Obzirom na količinu točaka ili poligona modela često se koriste pojmovi rezolucija ili kompleksnost modela. Modeli manje rezolucije (manje točaka) koriste se kod igara, kod animiranih likova i u slučajevima kada je scena kompleksna i sadrži mnogo objekata. Modeli veće rezolucije koriste se uglavnom za statičke objekte s mnoštvom detalja te za 3D printanje.

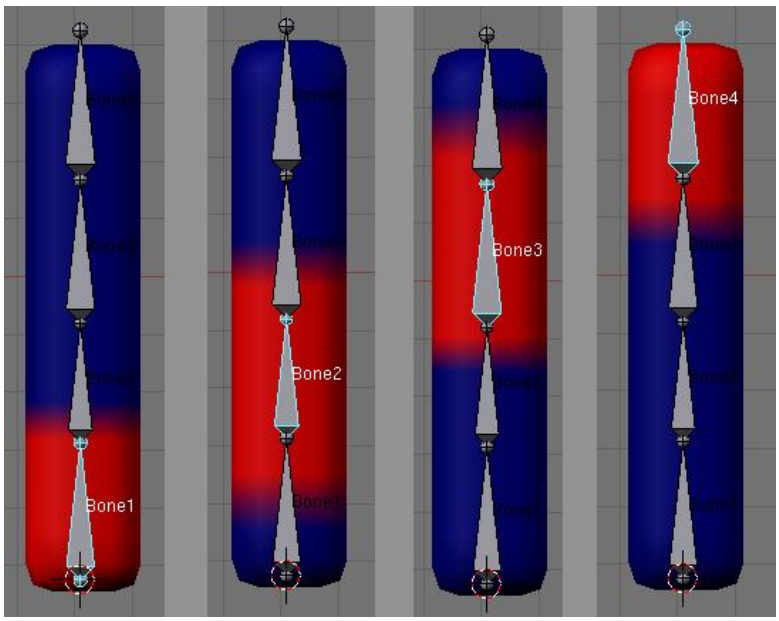
Još jedan pojam koji se veže uz kvalitetu mreže je topologija mreže poligona [1], a najviše ga koriste 3D modeleri. Topologija modela obuhvaća način na koji su točke posložene u mreži poligona. Dobra topologija modela je ona koja ima optimalan broj točaka po dijelovima modela, sadrži petlje poligona na korisnim mjestima te su joj točke generalno poslagane kako bi što bolje omogućile savijanje pojedinih dijelova kod animacije. Pod optimalan broj točaka smatra se da su točke raspoređene ovisno o razini detalja određenog područja te da jednostavniji i ravniji dijelovi modela ne sadrže mnoštvo nepotrebnih vrhova. Što je topologija modela bolja, to će i bilo koja akcija koja procesuira model biti uspješnija i točnija. Primjer dobre topologije na modelu glave možemo vidjeti na slici 1.



Slika 1. Topologija glave

Armatura ili kostur je pomoćna struktura koja se koristi za animiranje 3D modela. Armatura se sastoji od mnoštva kostiju koje su posložene u stablastu strukturu. Kost armature je jednostavna struktura koja ima svoj početak i kraj, koji se nekad zovu glava i rep. Ako na slici 2. izoliramo najdonju kost glava bio bila širi dio kosti pri dnu, a rep tanji dio sa suprotne strane. Početak i kraj svake kosti predstavljaju zglobove kostura. Stablasta struktura armature je ujedno i hijerarhijska, te sve kosti imaju svog roditelja. Kosti se u toj hijerarhiji mogu rotirati samo oko početne točke, tj. početna točka je fiksna i istovjetna završnoj točci roditelja, a završna točka je slobodna za kretanje. Kost je moguće i rotirati oko osi u koju je usmjerena (eng. *roll*).

Svrha armature je animiranje gotovog modela. To se izvodi tako što se armatura pridodjeli modelu, te svaka kost utječe na određeni dio modela nekom jačinom. Rotacijom kosti pomiču se sve točke modela na koje ta točka utječe, i to onoliko koliko je jak utjecaj kosti. Jačina utjecaja određuje se brojem između 0 i 1, a kod vizualizacije se za različite jačine koriste različite boje (slika 2). Slučajeve preklapanja područja na koju kost djeluje različiti programi rješavaju na razne načine. Blender će korisniku dopustiti da istu točku dodijeli više kostiju s najvećom težinom, no pri deformaciji će se utjecaj računati kao da svaka od njih utječe jednakim dijelom (npr. 0.25 ako se radi o 4 kosti). Autodeskova Maya korisniku uopće ne dopušta da se dovede u takvu situaciju te pri povećanju utjecaja jedne kosti na točku smanjuje utjecaj svih ostalih kostiju.

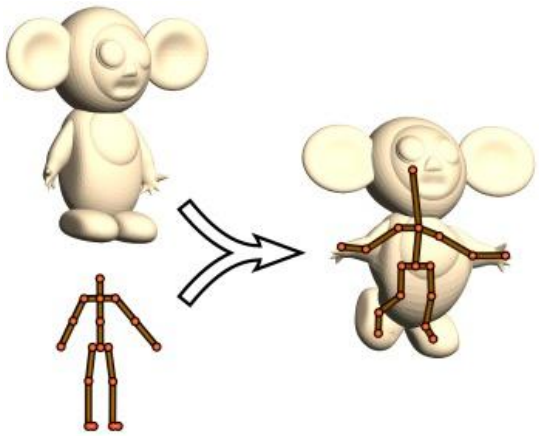


Slika 2. Kosti i njihove težine

Postojeća rješenja

Kako je riječ o problemu s kojim se ljudi često susreće, logično je da postoje i gotova rješenja za taj problem.

Ilya Baran i Jovan Popović s MIT-a razvili su sustav „Pinocchio“ [2] za automatsko generiranje animacija za modele. Njihov sustav se oslanja na gotov kostur koji se onda pokušava prilagodit danom modelu (slika 3). Koriste gotov kostur kako bi nakon prilagodbe mogli odmah koristiti sve predefinirane animacije za taj kostur poput hodanja, skakanja i trčanja. Algoritam kojim prilagođuju kostur je mnogo kompliciraniji od algoritma obrađenog u ovome radu. U svojoj osnovi, algoritam kroz tri koraka stvara graf unutar modela koji sliči na moguću strukturu kostura modela. Zatim pokušava predefinirani kostur prilagoditi tome grafu tražeći minimum funkcije kazne. Funkcija kazne definirana je predodređenim skupom pravila, a jačina određenog pravila podešena je strojnim učenjem. Algoritam je testiran na 16 bipedalnih modela te je za samo 3 dao nešto lošije rezultate. Spram Pinocchia, algoritam ovog rada nije toliko robusan te ima manje mogućnosti, no primjenjiviji je na veći broj modela, pogotovo na one s neuobičajenom raspodjelom kostiju.



Slika 3. Sustav Pinocchio

Uz Pinocchio, postoje i komercijalna rješenja poput Auto Character Setup [3] te Mixamo [4]. Takvi alati uglavnom ciljaju humanoidne bipedalne modele te nude velik spektar gotovih animacija takvih armatura. Algoritam kojim kreiraju armature nije poznat, no vrlo je vjerojatno da ide u istom smjeru kao onaj tima s MIT-a. Problem s ovakvim programima je njihova specijaliziranost na određen tip modela. Ako lik za kojeg želimo stvoriti armaturu odudara od kalupa za koji alat radi, potrebno je mnogo ručnog podešavanja kako bi dobili željeni rezultat.

Algoritam generiranja

Način generiranja armature trebao bi imati dva koraka. Prvi korak je dobivanje okvirne slike gdje bi se kosti i zglobovi trebali nalaziti, a drugi generiranje kostura iz takve slike. Poželjno bi bilo da prvi korak završi s modelom koji ima točke i rubove gdje bi se nalazili zglobovi i kosti. Takav model je trivijalno prebaciti u kostur, bitno je samo paziti na orijentaciju kostiju. Prilikom generiranja kostura potrebno je i pripaziti na kompleksnost rezultata te da se njegova kompleksnost može definirati određenim parametrima. Algoritam bi trebao raditi dobro za skup modela različitog oblika no kvalitetne i dobro razmještene geometrije mreže.

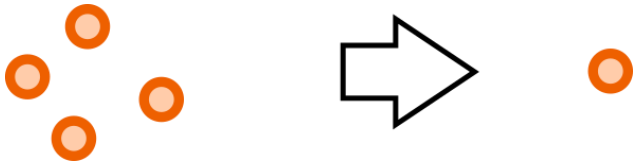
Implementacija

Odabir alata

Za implementaciju rješenja korišten je programski alat za 3D modeliranje Blender. Blender je besplatan program otvorenog koda koji omogućava modeliranje, izradu armatura, teksturiranje, uređivanje videa i sl. Blender je izabran kao okruženje za izradu rješenja zbog svojih jednostavnih no moćnih mogućnosti izrade skripti i proširenja. Prilikom izrade proširenja razvijaču na raspolaganju stoji sav asortiman gotovih rješenja za pojedine probleme koje su već implementirane. Također, korisniku će uporaba rješenja za generiranje armature biti jednostavnija budući da ne mora mijenjati programe nakon izrade modela i prije izrade animacija. Uz to, korisnik ima i mogućnost izvesti rezultat kao jedan od podržanih formata poput *.fbx*. Skripte i proširenja za Blender se pišu u programskom jeziku Python.

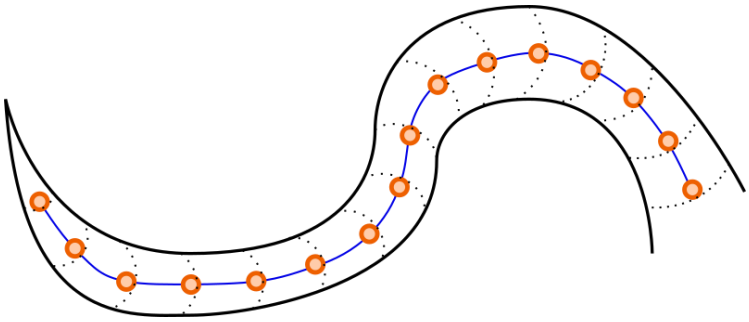
Podjela algoritma i kreiranje točaka

Prvi korak generiranja armature kod implementacije je podijeljen na dva dijela. U prvom dijelu se iz mreže modela traže točke koje bi bile povoljne za pozicije zglobova kostiju. Kako tih točaka kod kompleksnih modela može biti puno, vrši se pojednostavljivanje u vidu stapanja točaka koje se nalaze relativno blizu (slika 4). Udaljenost na kojoj se točke stapaju može se definirati parametrom skripte.



Slika 4. Stapanje točaka

Traženje takvih točaka riješeno je pomoću petlji rubova. Petlja rubova je skup rubova poredanih za redom u jednom smjeru. Takve petlje mogu biti otvorene ili zatvorene, te bi kod dobro definiranog modela trebale uredno pratiti geometriju tijela. Algoritam će tražiti takve petlje po modelu te pamtiti centre svih zatvorenih petlji. Dosta takvih centara se može naći u istoj točki, pa sve takve duple centre zanemarujemo. Pozitivna strana korištenja takve metode je da se kod modela s „udovima“ centri poslože kroz sredinu uda te tako čine lanac točaka koji je dosta sličan onome lancu kostiju koji bi trebao prolaziti kroz takav ud. Na slici 5. možemo uočiti tu pojavu: točkastim linijama označene su petlje rubova, a narančastim krugovima centri takvih petlji. Povučemo li liniju kroz te centre (plava crta) uočavamo da lijepo prati centar modeliranog kraka. Slično, no s nešto manje točaka bi kroz krak trebao prolaziti i lanac kostiju koji bi deformirao taj dio modela.



Slika 5. Lanac centara petlji rubova

Nakon generiranja točaka potrebno je te točke spojiti rubovima. Kako prethodni korak vraća samo nesortiranu listu točaka, potrebno je iz te liste kvalitetno odabrati početnu točku. Ako bi za početnu točku jednostavno odabrali prvu u listi, algoritam bi mogao početi iz nezgodne pozicije što bi otežalo logiku spajanja bridovima. Stoga je za početnu točku

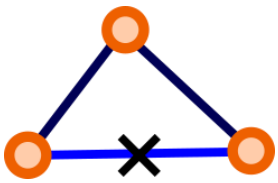
odabrana ona koja je najbliža središtu (eng. *origin*) modela. U Blenderu je središte modela označeno narančastom točkom. Kako korisnik ima mogućnost micati središte modela po želji tako može i utjecati na tok algoritma. Ako rezultat bude loš pomicanjem središta može ga pokušati poboljšati.

Spajanje bridova

Algoritam spajanja točaka radi na slijedeći način: za trenutnu točku se odaberu optimalni kandidati za spajanje, kandidati se sortiraju po udaljenosti te se na kraju točka spoji s nekoliko kandidata koju su otprilike na sličnoj udaljenosti. Odabir optimalnih kandidata svodi se na filtriranje skupa svih točaka bez trenutne. Prolazi se kroz nekoliko razina filtriranja kako bi konačan skup što više odgovarao optimalnim točkama za spajanje.

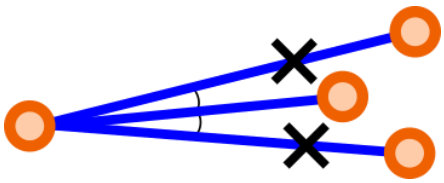
Prvi filter je trivijalan i izbacuje sve točke s kojima je trenutna već spojena. Time izbjegavamo mogućnost da nam se potkradu dupli bridovi koji bi mogli narušiti postupak generiranja kostiju iz bridova.

Drugi filter izbacuje sve točke koje bi, ako se spoje s trenutnom, napravile petlju u strukturi. Taj filter je potreban jer je armatura u većini slučajeva stablasta struktura. Armature mogu imati petlje, no u rijetkim slučajevima koje bi bilo teško prepoznati ovakvim algoritmom. Takva situacija prikazana je na slici 6, gdje su tamno plavom bojom označeni postojeći bridovi, a svjetlijom novi brid koji se ne bi smio pojaviti.



Slika 6. Petlja u strukturi

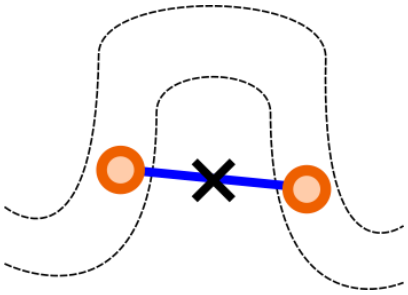
Treći filter utječe na točke koje su od trenutne orijentirane u istom smjeru. Prva i najbliža točka se ostavlja, a sve ostale se zanemaruju (slika 7). Time sprječavamo prijevremeno spajanje s točkama u podlancima armature. Orijentaciju računamo kao kut između bridova s vrhom u trenutnoj točki te minimalan kut ostavljamo kao parametar algoritma.



Slika 7. Ignoriranje točaka isto orijentacije

Četvrti filter će maknuti točke čijim spajanjem bi nastao predug brid. Ocjena koliko je dug predugačak brid je ostavljena korisniku kao parametar filtera. Izbacivanjem predugih bridova sprječavamo da nam se krajevi lanaca spoje dalje na neku drugu točku od ponuđenih.

Zadnji filter će izbaciti točke koje će stvoriti brid koji presijeca geometriju izvorišnog modela (slika 8). Ovaj filter je bitan jer omogućuje praćenje strukture i toka mreže modela, te će spriječiti rubne slučajeve kod kojih su točke dosta blizu no ne smiju se spojiti. Provjera da li potencijalni rub siječe geometriju nije trivijalna, pa se za nju implementacija oslanja na BVHTree [5] strukturu Blenderovog API-a. BVHTree je pomoćna struktura koja je optimizirana za pretraživanje blizine (eng. *proximity search*) te bacanje zraka (eng. *ray casting*). Za potrebe filtera algoritam baca zraku iz trenutne točke do točke kandidata, te kandidata izbacuje ako se zraka sudari sa geometrijom.



Slika 8. Zabranjeni rub koji presijeca model

Kao što je već navedeno, nakon filtriranja se uzimaju točke na podjednakoj udaljenosti od trenutne. Taj uvjet je također ponuđen na izbor korisniku za podešavanje, a implementiran je tako da se uzme najbliža točka te se za sve ostale gleda jesu li udaljene manje od određenog faktora.

Ovaj proces se pokreće za svaku točku zglobova. Poredak po kojem se točke procesiraju definiran je redom slijedećih točaka. Na početku se u redu nalazi samo početna točka, a tijekom algoritma se u red dodaju s kojima se trenutna točka spojila. Time se postiže lančano spajanje točaka koje podsjeća na razgranavanje stabla i pomaže procjeni kako će se algoritam ponašati u kojem trenutku. Ako se neka točka u svojoj iteraciji nije spojila

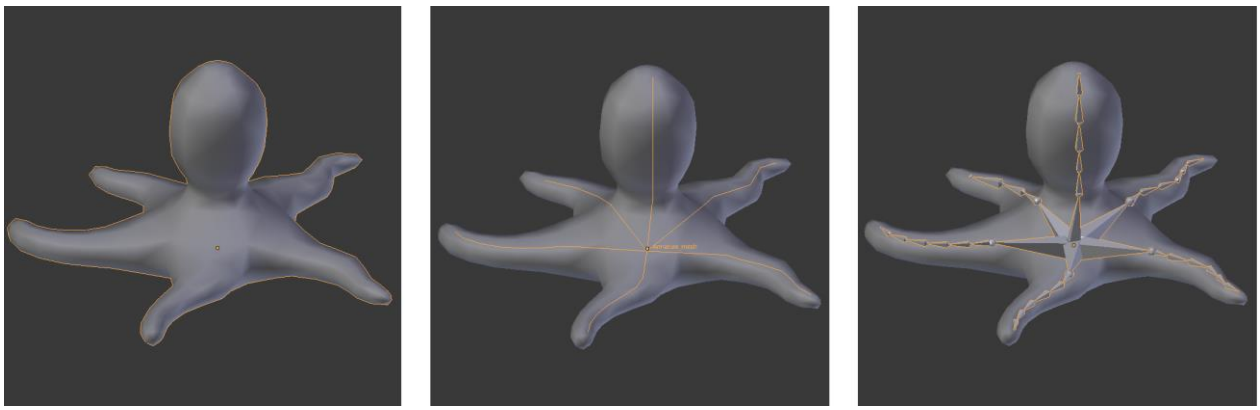
s nijednom drugom (što može biti slučaj kod krajeva udova) onda će za slijedeću točku uzeti jedna od još neposjećenih točaka.

Kako algoritam ne radi savršeno, odlučeno je odvojiti dva glavna koraka i prilikom pokretanja kroz sučelje. Na kraju prvog dijela korisnik će dobiti stablastu strukturu sastavljenu od točaka i bridova.
Ako se u takvoj strukturi dogodi situacija da je nastao loš brid ili se točka nije slijepila s nekom bližom, korisnik može napraviti te sitne promjene ručno te nakon prepravaka nastaviti na drugi korak generiranja kostura.

Pretvorba u kostur

Drugi dio stvaranja kostura je puno trivijalniji nego prvi. Generiranje kostiju izvodi se tako da se za svaku točku stablastog modela uzmu njeni bridovi te se po bridovima izvuku (eng. *extrude*) kosti. Najveća dilema u ovome dijelu je od koje točke krenuti. Tako je, slično kao kod prvog dijela, odlučeno da se kreće od točke najbliže središtu te se kosti orijentiraju od te točke prema ostalim susjedima. Prema tome, korisnik može utjecati kako će mu kosti biti orijentirane pomicanjem središta modela između koraka algoritma. Ovaj dio algoritma implementiran je rekurzivno, pazeći pritom da se hijerarhija roditelja i djeteta dobro postavi, te da se kraj kosti roditelja i početak kosti djeteta nalaze u istoj točci.

Konkretan primjer rada možemo vidjeti na slici 9. Model za koji se generira armatura je jednostavan model hobotnice, a krajnji rezultat je dosta sličan kosturu kojeg bi korisnik izradio ručno.



Slika 9. Faze rada algoritma

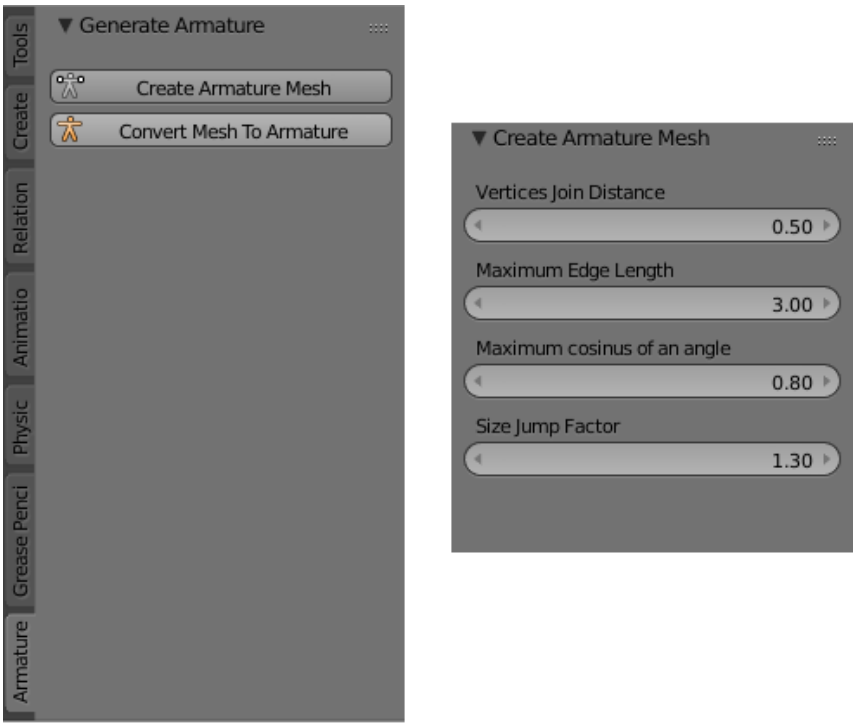
Grafičko sučelje

Zbog širokog spektra mogućnosti koje Blender omogućuje razvijačima korištenje implementiranog algoritma je jednostavno i brzo. Algoritam je napisan kao proširenje za Blender u obliku jedne Python skripte. Uz konkretan kod koji se odnosi na rad algoritma, u skripti se nalaze i definicije grafičkog korisničkog sučelja za interakciju i pokretanje. Instaliranje ovakvog proširenja svodi se klik gumba „Install Add-on From File..“ u postavkama, odabir skripte te označavanje kvačice koja aktivira proširenje.

Aktivacijom proširenja na traku s alatima kod 3D prikaza dodaje se kartica Armature. Klikom na karticu otvara se panel s dva gumba. Prvi gumb naziva „Create Armature Mesh“ pokreće prvi dio generiranja te stvara mrežu kostura. Klikom će se stvaranje mreže pokrenuti, a u traci će se pojaviti novi panel s parametrima algoritma. Mijenjanjem parametara automatski će se ponovo pokrenuti generiranje mreže. Ovisno o kompleksnosti modela, takav prikaz promjena u živo može uvelike olakšati traženje optimalnih parametara, ili onih parametara za koje je jednostavno ručno popraviti rezultat.

Klikom na drugi gumb iz selektirane mreže točaka stvara se gotova armatura. Kako ovaj korak nije parametriziran, neće se otvoriti dodatan panel.

Navedene elemente grafičkog korisničkog sučelja možemo vidjeti na slici 10.

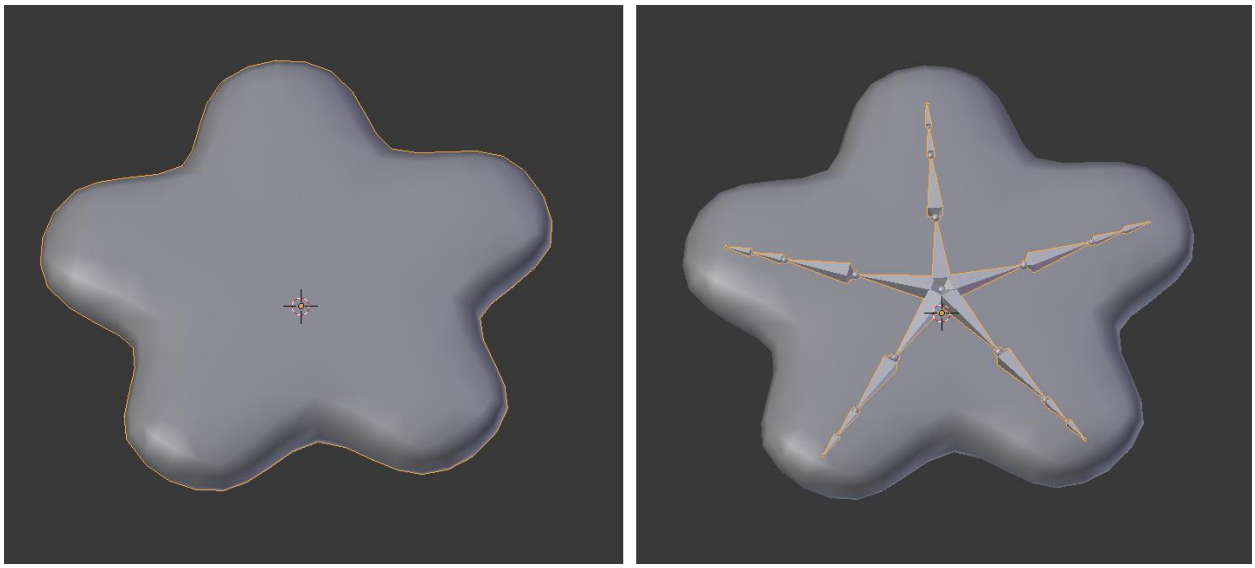


Slika 10. Grafičko korisničko sučelje za pokretanje algoritma

Ocjena rezultata

Općeniti rezultat

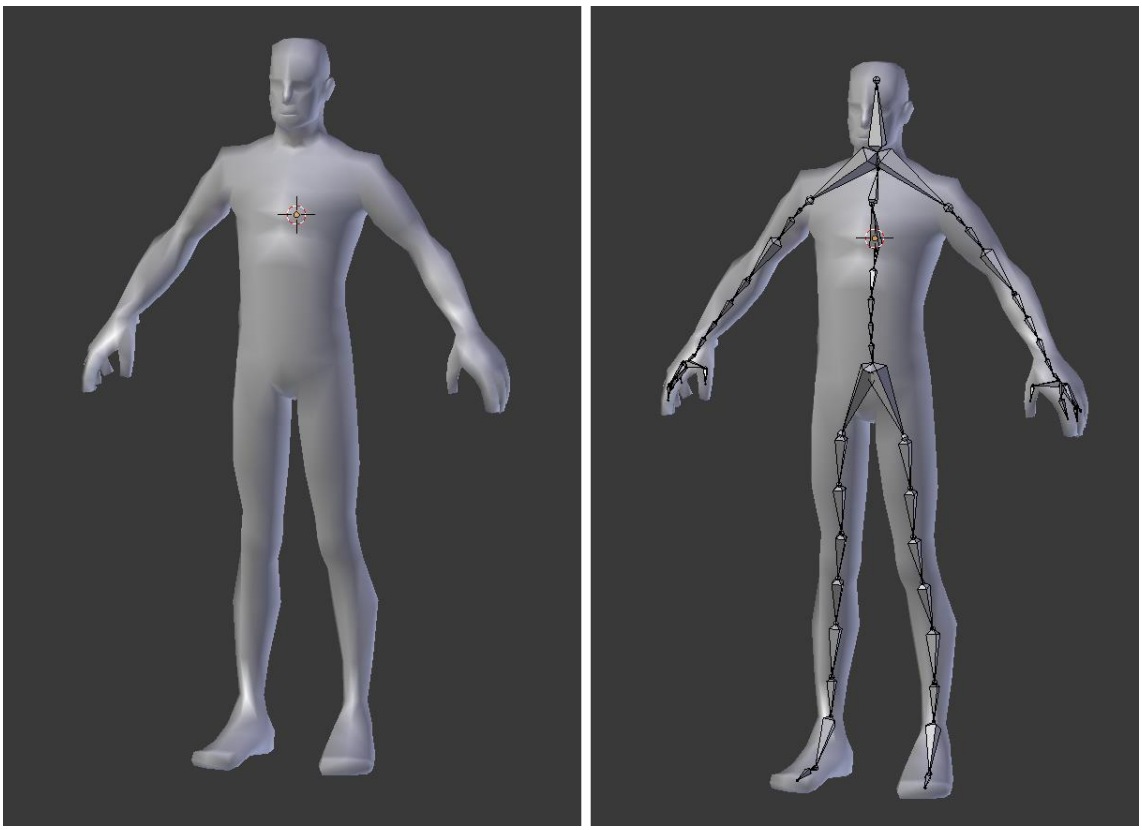
Algoritam je isproban na manjem skupu jednostavnih modela različitih oblika. Za takve jednostavnije slučajeve program radi dosta dobro, te je uz manje prepravke moguće dobiti kvalitetnu armaturu za model. Primjer ispravnog rada za jednostavan model možemo vidjeti na slici 11. Korišten model zvijezde opisan je gustom no jednostavnom mrežom koja ima mnoštvo petlji rubova. Centri velikog broja petlji smjeste se u centar zvijezde, a ostali opisuju krakove. Takva geometrija veoma pogoduje algoritmu pa je zato i rezultat uredan i bez grešaka.



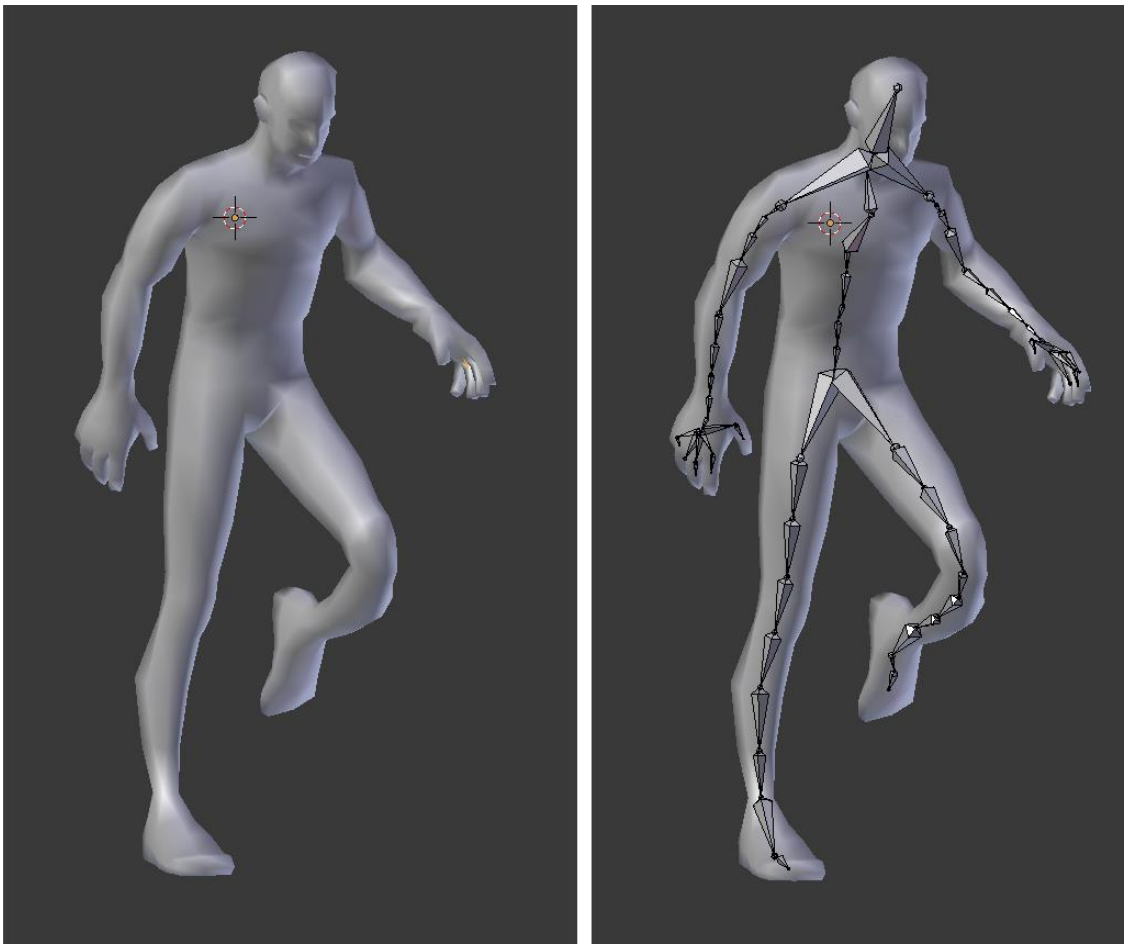
Slika 11. Rezultat algoritma na modelu zvijezde

Usporedba istog modela u različitim pozicijama

Ako je algoritam uspješan u određenoj poziciji modela, trebao bi biti i za sve slične pozicije. Razlog tome je zato što se algoritam trudi pratiti geometriju bez obzira na položaj, te za istu geometriju (isti broj i raspored točaka) daje slične ili jednake rezultate. Primjer toga možemo vidjeti na modelu čovjeka na slikama 12 i 13.



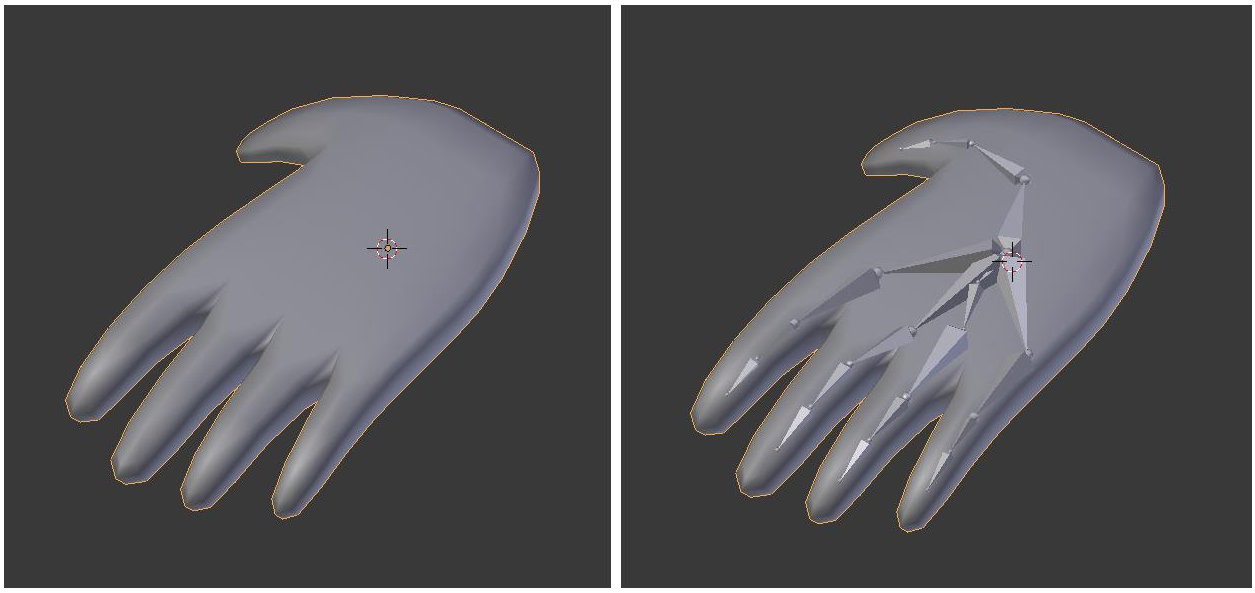
Slika 12.Rezultat algoritma na modelu čovjeka u uspravnoj poziciji



Slika 13. Rezultat algoritma na modelu čovjeka u uspravnoj poziciji

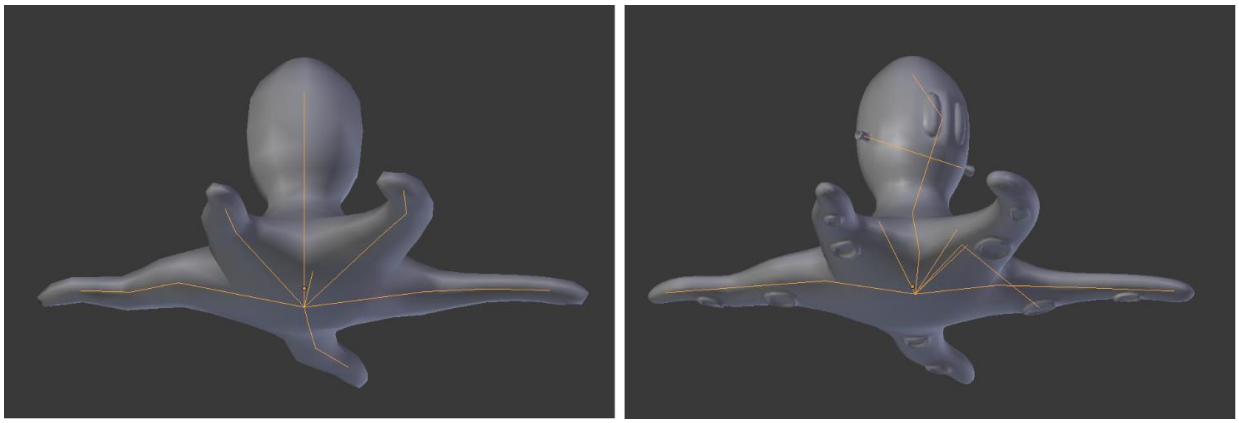
Rezultati na dijelu modela

U slučaju da algoritam ne radi na cjelovitom modelu, ili daje loše rezultate na određenim manjim dijelovima, opcija je odvojiti takve dijelove u posebne modele, izvršiti algoritam, te ih ponovo spojiti u cjelinu. Na slici 14 prikazan je takav pristup, te je algoritam uspješno izveden da šaci koja je samo djelić cijelog modela.



Slika 14. Rad algoritma na dijelu modela

Problemi s povećanjem detalja



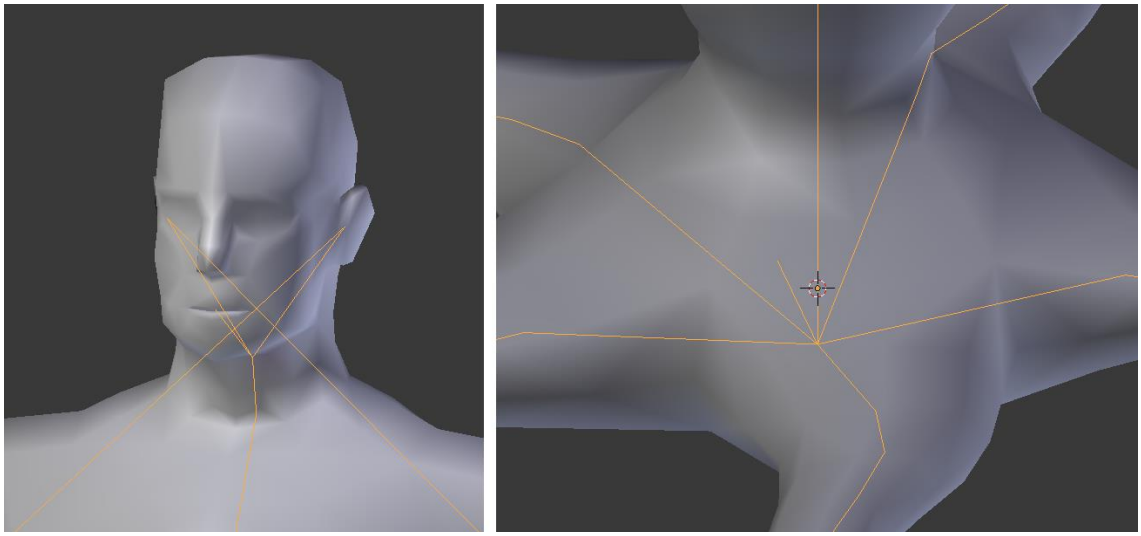
Slika 15. Razlika u uspješnosti algoritma kod različite razine detalja

Kao što možemo vidjeti na slici 15, dodavanjem detalja na jednostavan model točnost algoritma može se veoma pogoršati. Dodavanje detalja može dovesti do nestajanja potpunih petlji rubova čime se gube točke i bridovi koji mogu biti temeljni za određen

lanac kostiju. Također, dodavanjem detalja mogu nastati nove petlje rubova koje nisu važne za strukturu armature i tako bi potpuno narušile valjanost rezultata.

Greške pri radu

Postupak ručnog traženja optimalnih parametara u situacijama gdje je očekivani kostur kompleksan zna biti veoma iscrpljujuće i kontraproduktivno. Odabir krivih parametara može dovesti do loše generirane mreže kostura s raznim artefaktima. Neki od takvih artefakata su nepotrebni bridovi, krivo spojene točke zglobova te nepotpuno stablo kostura. U slučaju da je grešaka malo, jednostavniji pristup od traženja boljih parametara je ručno otklanjanje. Pošto se intervencije izvode na mreži točaka i bridova, takvi sitni popravci ne bi trebali potrošiti puno vremena. Primjer takvih grešaka koje je lakše otkloniti ručno možemo vidjeti na slici 16.



Slika 16. Greška kod spajanja točaka (lijevo), brid viška (desno)

Za određene vrste modela algoritam radi jako loše ili uopće ne radi. Najveća mana je u tome što nemaju svi modeli potpune petlje rubova tamo gdje bi bilo dobro optimalno staviti kosti. Primjeri takvih modela su životinje sa više udova koji počinju na bliskim mjestima (poput pauka). Kako bi za njih ovaj pristup proradio potrebno bi bilo dodati još potpunih petlji na strateškim pozicijama, no time se gubi na kvaliteti modela i fleksibilnosti animiranja nakon stvaranja kostura.

Ako je ciljni model podijeljen u više zatvorenih cjelina, generiranje pomoćne mreže će rezultirati nespojenim točkama te izgubljenim bridovima. U tom slučaju potrebna je intervencija korisnika da ručno spoji točke i bridove. Ovisno o kompleksnosti, takav proces može biti zamoran i dugotrajan. Izvor ovoga problema leži u filteru za uklanjanje kandidata koji bi stvorili brid što sječe geometriju modela. U ovom slučaju rješenje može biti slično kao u primjeru s dlanom, tj. razdvajanje zatvorenih cjelina objekta u posebne objekte.

Gledano po koracima, prvi korak koji generira mrežu kostura je dio gdje se događaju sve greške. Drugi korak koji stvara armaturu iz mreže će uvijek raditi dobro ako je mreža ispravna. Segregacija koraka omogućuje lakšu nadogradnju postupka, te se po želji prvi dio može potpuno zamijeniti s nekim drugim algoritmom, ili ručnim kreiranjem mreže.

Moguća poboljšanja

Algoritam se temelji na pretpostavci da će centri zatvorenih petlji bridova dati dobru osnovicu za potencijalni položaj kostiju armature. Iako to radi dobro za dugačke, valjkaste strukture poput nogu, ruku i krakova, problem se dogodi na mjestima gdje se ti udovi spajaju s ostatkom tijela i gdje petlje neće biti potpune. Jedan pokušaj poboljšanja algoritma koji ne mijenja temeljnu ideju bilo bi uzimanje u razmatranje onih petlji bridova koje nisu potpune, no čine odlično strateško mjesto za položaj kostiju. Traženje takvih petlji trebalo bi biti uvjetovano parametrom, recimo minimalnim brojem točaka koje petlja mora sadržavati.

Poboljšanje dijela spajanja točaka u mrežu trebalo bi ići u smjeru definiranja novih i boljih filtera kandidata za spajanje. Bolji filteri trebali bi uzimati u obzir povezanost i blizinu petlji od kojih točke nastaju kako bi smanjili mogućnost pogreške pri povezivanju. Uz to, velik problem je što filtri u trenutnoj implementaciji djeluju jednako na cijelom skupu točaka. Bolja implementacija bi mogla pokušati pametno skalirati parametre algoritma s obzirom na trenutnu točku i njene najbliže susjede i potencijalne kandidate. Time bi se spriječilo nestajanje pojedinih bitnih bridova, kao i stapanje/nestajanje korisnih točaka koje se nalaze blizu.

Proces generiranja se možda može poboljšati i zamjenom cijelog prvog dijela generiranja mreže kostura. Jedan od mogućih alternativnih pristupa je pokušati geometriju modela „uvlačiti“ prema unutrašnjosti dok ne konvergiraju prema stablastoj strukturi. Implementacija uvlačenja mogla bi iterativno pomicati svaku od točaka za neku malu udaljenost u smjeru suprotnom od normale točke (što bi onda trebalo ići prema unutrašnjosti). Kada točka naiđe na neku drugu ili čak nakupinu više njih, trebala bi stati s kretanjem te se stopiti u jednu točku. Bridovi koji povezuju takve točke trebali bi nastati po jednostavnoj formuli: ako su točke bile povezane prije spajanja, i nisu se stopile u isto središte, trebale bi biti povezane i nakon. Slično kao i algoritam implementiran u radu, ovaj pristup bio bi učinkovit na dijelovima modela koji su cilindričnog oblika. Poboljšanje bi se moglo uočiti na mjestima gdje implementiran algoritam griješi ili za modele s lošijom geometrijom. Problemi s ovakvim pristupom uočili bi se na modelima s gustom mrežom točaka i mnoštvom detalja, te bi za njih bilo potrebno napraviti nekakav oblik pojednostavljenja geometrije.

Zaključak

Izrada armature za 3D modele može biti zamoran dio posla animiranja objekata. Generiranje takve armature, ili njenih dijelova bi barem u neku ruku pomoglo animatorima i dizajnerima sa štednjom dragocjenog vremena. Dubljom analizom ovoga naizgled jednostavnog problema možemo uočiti da rješavanju možemo pristupiti na mnogo različitih načina. Svaki od pristupa može se optimizirati za određene vrste modela, no algoritam koji bi radio generalizirano nije lako ostvariti. U ovome radu opisan je jedan od načina generiranja armature koji se temelji na petljama rubova. Implementirani algoritam radi dobro za skup jednostavnijih modela, no s kompliciranijim modelima s više detalja dolazi do problema i grešaka. Iako ideja za generalizirani algoritam generiranja nije ostvarena uspješno, razvijeni proizvod se može koristiti kao pomoć pri manjim projektima ili u ranim stadijima izrade modela. Razdvajanje algoritma u dva neovisna dijela pokazalo se efikasnim jer omogućuje korisniku da ručno ispravi manje nedostatke nastale u dijelu koji je skloniji greškama. Korištenje Blendera za implementaciju rješenja bio je odličan izbor, a pokretanje algoritma je jednostavno i odlično se uklapa u ostatak procesa razvoja animacija.

Literatura

[1] *Mesh Topology*, 24.6.2018.

http://wiki.polycount.com/wiki/Topology

[2] I. Baran i J. Popović, 2007. *Automatic Rigging and Animation of 3D Characters*,

http://people.csail.mit.edu/ibaran/papers/2007-SIGGRAPH-Pinocchio.pdf

[3] *Auto Character Setup*, 25.6.2018.

https://www.autocharactersetup.com/

[4] *Mixamo*, 25.6.2018.

https://www.mixamo.com/

[5] Blender Foundation, *BVHTree Utilities,* 26.6.2018.

https://docs.blender.org/api/blender_python_api_2_75_3/mathutils.bvhtree.html

Sažetak

Rad opisuje jedan od načina izrade animacija u 3D dizajnu te objašnjava moguće pojednostavljenje i ubrzanje procesa izrade. U ovom slučaju ubrzanje je ostvareno proceduralnim generiranjem armature za gotove modele. Razrađen je jedan od mogućih algoritama, a temelji se na petljama rubova mreže objekta. Opisani pristup podijeljen je u dva dijela, sadrži nekoliko parametara te omogućuje korisniku intervenciju u slučaju manjih pogrešaka. Rad pokriva detaljno objašnjenje implementacije u programskom alatu Blender i sadrži ocjenu izrađenog algoritma na nekoliko modela različitih svojstava. Na kraju su navedeni mogući pristupi za poboljšanje implementacije te alternativni načini generiranja kostura.

Ključne riječi: armatura, proceduralno generiranje, 3D animiranje, Blender

Abstract

This thesis describes one of the possible ways for creating animations in 3D design and possible approach for simplifying and accelerating creation process. In this case simplification is achieved by procedural generation of armature for prepared models. Within the scope of the project an algorithm is implemented, and it's based on edge loops of object mesh. Described algorithm is divided in two parts, parametrized and enables user intervention in case of minor errors. Thesis includes detailed explanation of implementation in 3D modeling tool Blender and evaluation of algorithm on several test models with various properties. Possible alternative approaches for generating armatures and methods for optimizing created product are listed in the end.

Keywords: armature, procedural generation, 3D animation, Blender