VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BLENDER ADD-ON PRO RIGGING 3D MODELŮ

BLENDER ADD-ON FOR 3D RIGGING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ROSTISLAV HORÁK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Horák Rostislav**
Program: Informační technologie
Název: **Blender add-on pro rigging 3D modelů**
Blender add-on for 3D Rigging
Kategorie: Počítačová grafika

Zadání:

1. Naučte se základy práce s 3D modelovacím programem Blender (verze 2.8 a výše)
2. Seznamte se se skriptovacím Blender Python API
3. Nastudujte možná a již existující řešení dané problematiky
4. Navrhněte add-on pro rigging (vytváření animační kostry) pro libovolné 3D modely, včetně algoritmů a UI
5. Add-on implementujte a demonstrujte jeho použití na různých typech dat
6. Zdokumentujte a zveřejněte výsledný add-on pro použití dalšími uživateli
7. Vytvořte video reprezentující výsledky vaší práce

Literatura:

- Baran, Ilya, and Jovan Popović. "Automatic rigging and animation of 3d characters." *ACM Transactions on graphics (TOG)* 26.3 (2007): 72-es.
- Allen, Eric, and Kelly L. Murdock. *Body language: advanced 3D character rigging*. John Wiley & Sons, 2011.
- Pan, JunJun, et al. "Automatic rigging for animation characters with 3D silhouette." *Computer Animation and Virtual Worlds* 20.23 (2009): 121-131.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4, experimenty vedoucí k bodu 5

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem projektu je vytvoření Blender rozšíření, které automaticky vygeneruje kostru 3D modelu a navíc bude otevřené a snadno rozšiřitené o další metody vyjmutí kostry. Základem projektu je kostra rozšíření, která bude obsahovat dynamicky načítaný seznam implementovaných metod, ze kterých si následně může uživatel vybrat. Projekt je implementován v jazyce Python pomocí Blender API. Částí práce bude implementace metody vyjmutí kostry metodou použití 3D siluety, důraz je kladen zejména na rychlost, při zachování co největší přesnosti.

Abstract

The project objective is to create a Blender addon that can automatically generate a skeleton for 3D model and on top of that will be open and easily extendable by more skeleton extraction methods. The core of this project is a framework of this addon which will contain a list of implemented methods from which the user can choose. The project is implemented in Python with Blender API. Part of the project is an implementation of skeleton extraction method using 3D silhouette, emphasis is placed on speed while retaining certain accuracy.

Klíčová slova

3D model, rigging, Blender rozšíření, skinning, kostra, vyjmutí kostry, animace

Keywords

3D model, rigging, Blender add-on, skinning, skeleton, skeleton extraction, animation

Citace

HORÁK, Rostislav. *Blender add-on pro rigging 3D modelů*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

Blender add-on pro rigging 3D modelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Inženýra Tomáše Chlubny. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Rostislav Horák
12. května 2021

Poděkování

Nejprve musím poděkovat vedoucímu práce Ing. Tomáši Chlubnovi za ochotu a velmi rychlé odpovědi na moje otázky. Dále musím určitě poděkovat členům a přispivatelům Blender Foundation za úžasný kus technologie, jakou Blender je.

Obsah

Úvod

Blender je světově rozšířený, svobodný a open-source 3D editor, pro který existuje mnoho tzv. addonů (rozšíření). Díky širokým možnostem pro rozšíření a volně dostupným zdrojovým kódům se neustále zlepšuje. Blender má rozsáhlé vestavěné skriptovací prostředí v programovacím jazyce Python. Skripty jsou implementovány pomocí jazyka Python a Blender API, nebo-li programového rozhraní Blenderu. Cílem této práce bylo položit základ rozšíření Blenderu, které je otevřené, a tím cílí k budoucí inovaci řešení problému automatického, takzvaného rigging a následně skinning, což jsou vytváření kostry modelu (kde kostra je abstrakce modelu) a navázání vygenerované kostry na samotný 3D model. Kostry se používají k abstrakci topologie modelu, a tím zjednodušení další práce s modelem jako například animace, analýzu tvaru, virtuální navigace, segmentace modelu.

Vytváření koster a jejich napojování na model je povětšinou manuální, časově náročná a dosti repetitivní práce, proto je snaha tuto činnost automatizovat. S čím dál větším zastoupením 3D grafiky nejen v herním, ale i filmovém průmyslu, je důležité najít části práce, které jde zjednodušit, automatizovat, a tím poskytnout časovou úsporu a redukování množství repetitivní práce.

Rozšíření obsahuje dynamicky načítaný seznam metod vyjmutí kostry z modelu, ze kterého si uživatel může vybrat, kterou chce použít. Použitá metoda vygeneruje kostru, kterou může následně uživatel upravit nebo rozšířit podle potřeby a naváže ji na model. Projekt bude cílit na metody zaměřené na rychlost, při zachování určité přesnosti, a které jsou použitelné na 3D polygonální mesh, bez potřeby konverze, jako například voxelizace, zabírajících mnoho času.

Součástí práce bude implementace metody vyjmutí kostry pomocí 3D siluety. Metoda vyjmutí kostry pomocí 3D siluety se snaží získat 3D siluetu modelu, ze které určí středovou křivku pomocí výpočtu Delauného triangulace. Stejně získá druhou 3D siluetu kolmou k první, pomocí které upraví křivku získanou z první siluety podle 3. osy. Výsledné rozšíření Rigger je dostupné na Gitlab[1].

V následující kapitole 1 je uvedena teorie o Blenderu, vysvětlení pojmů rigging, skinning, o metodách získání kostry z 3D modelu. V kapitole 2 jsou shrnuta a předvedena současná řešení. V kapitole 3 je návrh rozšíření, rozvrhnutí uživatelského rozhraní, navrhnutí kostry, pro zajištění jednoduché rozšiřitelnosti a návrh rigovací metody, která bude implementovaná. V kapitole 4 je popis implementace – problémy a jejich řešení. Další kapitolou 5 je Testování, které se zaměřuje na testování metody, její časové náročnosti a přesnosti výsledků, testování rozšíření jako celku. Poslední kapitolou 6 je závěr a shrnutí práce.

[1] Blender rozšíření Rigger dostupné na: https://gitlab.com/TechMarine/rigger-blender-addon

Kapitola 1

Teorie

Kostra je velmi užitečná struktura, která abstrahuje model do 1D křivky, která zachovává topologii modelu, ale ztrácí jeho objem. 1D je myšleno, že nemá žádný objem ani obsah. Jednodimenzionální objekt ve 3D je stále jednodimenzionální. S kostrou se pak lépe provádí další operace, jako je animace 3D modelu, analýza tvaru nebo analýza pohybu člověka z videa, popřípadě propojení těchto dvou a tím animace 3D modelu z videa. S postupným vývojem Blenderu se objevily rozšíření, které se snaží nějakým způsobem zjednodušit a automatizovat rigging a práci s tím spojenou, ale nejsou perfektní. Také ze stany metod vyjmutí nebo vestavění kostry je snaha najít nějakou robustní a přitom výpočetně jednoduchou metodu. Problém nastává v různém popisu modelů a tudíž existuje mnoho metod, které ale nelze v určitých situacích použít a nebo jsou daleko od optimálního řešení. [5]

1.1 Blender

Blender[1] je svobodý a otevřený program pod licencí GPL, dostupný zdarma. Blender je multiplatformní, funguje stejně dobře na operačních systémech Linux, Windows a Mac, toho dosahuje použitím OpenGL. Blender obsahuje sadu nástrojů pro práci s 3D grafikou. Nástroje umožňují modelování, rigging, animace, simulace, renderování, tvorbu a úpravu videa i audia a tvorbu her. Mimo 3D podporuje i 2D krslení a animace. Pro pokročilé uživatele poskytuje Blender API (programové rozhraní Blenderu), které umožňuje tvorbu skriptů a celých rozšíření, jejich cílem je úprava aplikace a specializovaných nástrojů.

Blender prošel velkým rozvojem, v roce 2019 proběhla velká změna ve verzi 2.8 jak uživatelského rozhraní, tak Blender API. V současnosti se řadí mezi vrchol 3D sad nástojů.

Díky otevřenosti a přístupnému Blender API [4] pro skripty a rozšíření existuje velké množství všemožných nástrojů a rozšíření na samotném BlenderMarket[2], Gitlab, Github nebo jinde na internetu.

[1] https://www.blender.org/
[2] https://blendermarket.com/

1.2 Rigging

Pojem Rigging 3D modelu je proces analogický k vazačství. Vazač (anglicky rigger) je člověk zabývací se přípravou nákladu pro přesun jeřábem. Z pohledu 3D grafiky je rigging příprava modelu, vytvořením kostry (abstrakce topologie modelu) a navázání kostry na model (skinning), pro následné použití kostry jako zjednodušení deformace modelu, například k animaci 3D modelu.

Nejčastějším způsobem vytváření rigů je manuální tvorba kostry, následný skinning použitím skinning algoritmu a následná úprava vah. U profesionálních rigů je tato práce iterativní, nejdříve vytvoří kostru, provede skinning, upraví váhy, zkusí animace, upraví kostru, a tak dokola, dokud nedosáhne požadovaného narigovaného modelu. Různé druhy rigovacích metod jsou popsány dále.

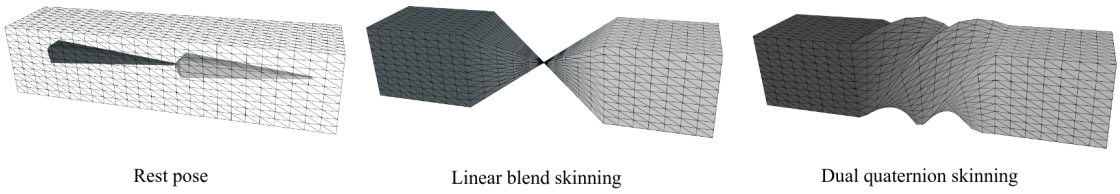
1.3 Skinning

Pojem skinning je proces navázání obalu (kůži, anglicky skin) 3D modelu na kostru. Jednotlivé vertexy se navazují na jedlotlivé kosti, některé vertexy jsou ovlivňovány několika kostmi zároveň, kde síly vlivu jsou váhovány 1.2. Výsledkem je propojení kostry a modelu tak, že pokud se pohne kost, pohnou se vertexy, které jsou navázány na kost a velikost pohybu záleží na váhách.

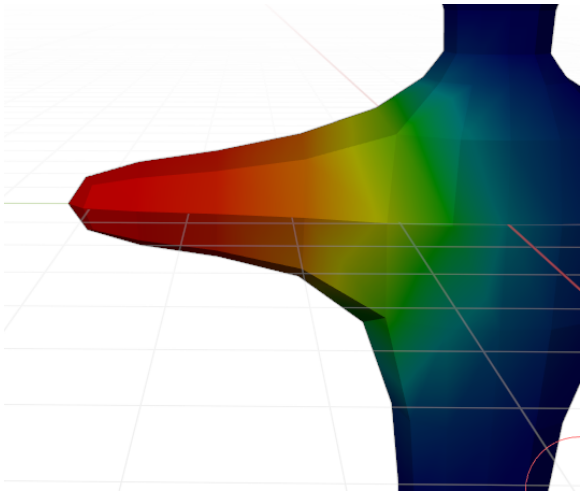
V minulosti, v některých hrách byl tuhý skinning, který měl limit počtu kostí na vertex jedna. Takže vertex mohl být navázán pouze na jednu kost, to zajišťovalo větší rychlost a díky malému počtu polygonů to kvalitě moc neubíralo. Při zvyšování detailu 3D modelů byly tyto limity zvětšeny, aby poskytly plynulé přechody kostí a kvalitnější animace.

Proces skinningu lze dělat manuálné pomocí malování vah, ale v praxi se většinou použije nějaký skinning algoritmus a výsledek algoritmu uživatel doladí manuálním malováním vah.

Nejjednoduším příkladem skinning algoritmu je Heat algoritmus. Kdy Heat algoritmus prohledává okolí vertexů a hledá nejbližší kosti, ke kterým přidává váhy. Při následné animaci můžou vznikat artefakty, znázorněno na obrázku 1.1, výsledky animace záleží na použité skinning transformační metodě. Často používanou skupinou deformačních metod jsou Přímé metody. Příme metody vypočítávají uzavřené výrazy, bez numerických optimalizací, takže jsou většínou rychlé a umožňují velkou míru paralelizace. Příme metody se dále dělí na lineární, multi-lineární, nelineární. Celkových metod a jejich kategorií je spousta a pořád přibývají nové.[8]



Obrázek 1.1: **Skinning metody deformace**. Vlevo je originální narigovaný model v klidovém stavu. Uprostřed je anomálie nazývaná candy-wrapper, způsobená rotací pomocí skinning metody lineární směsi. Napravo je jedno z řešení anomálie pomocí metody duálních kvaternionů, která se řadí mezi nelineární skinning metody. (převzato z [8]).



Obrázek 1.2: **Váhy vertexů vůči kosti ruky**. Od červené (největší váha) po modrou (nulová váha).

1.4 Metody používané pro rigging

Existuje snaha generování kostry z obrazu (fotka, video) nebo 3D modelu, pro následné animování, analýzu tvaru, segmentaci na podčásti modelu nebo virtuální navigaci. Vytvoření kostry danému 3D modelu není jednoduchá záležitost a to ani manuálně. Spoustu začínajících animátorů rigging a spojené práce odradí a pro pokročilé je to většinou repetitivní a časově náročná nutnost. Proto je snaha tuto činnost automatizovat a ušetřit tak čas. Metody jsou často založené na výpočetní geometrii.

Druhy metod pro rigging

Metody lze rozdělit několika způsoby. [5]

Podle způsobu získání kostry:

- Vyjmutí kostry z 3D modelu – kostru vytváří analýzou modelu, nebo jeho transformací, a to různými přístupy metod.
- Vestavění kostry do modelu – má vytvořenou již existující kostru, a tu se snaží napojit na 3D model, za použití transormací jako je zmenšení, zvětšení, rotace aplikovaných na celou kostru a nebo její podčást.

Podle způsobu reprezentace 3D modelu:

- Objemové – před samotnou analýzou se musí model převést na diskrétní objemovou reprezentaci, a to buď voxelizací nebo diskretizovanou funkcí pole [11].
- Geometrické – model reprezentovaný polygonálním meshem nebo seskupením bodů [5].

Výhodou metod vestavění již existující kostry do 3D modelu je možnost následně použít již vytvořené animace pro danou kostru. Problém nasvává při atypických modelech pro které nemáme předem připravenou kostru nebo modely, které jsou dále upravené, pak vestavěná kostra nemusí být optimální a bude potřebovat další úpravy.

Metody vyjmutí kostry mají výhodu, že je lze použít na jakýkoliv 3D model, ze kterého vytváří novou kostru. Nová kostra nemá předvytvořené animace, které se musí dodělat nasledovně po vytvoření. Kvalita metod je různorodá a záleží i na případech užití. Metody se liší i podle výpočetní náročnosti, která se odvíjí mimo jiné z reprezentace 3D modelu. Pokud metoda vyžaduje objemovou reprezentaci 3D modelu, ale dostane geometrickou, tak musí 3D model převést do objemové reprezentace, se kterou může daná metoda pracovat, problém však může nastat při převodu, a to snížení přesnosti reprezentace původního modelu, která závisí na velikosti vzorkování. Pokud je potřeba převod reprezentace 3D modelu naopak z objemové na geometrickou reprezentaci, použije se metoda vyjmutí povrchu. Problém převodu nese s sebou velkou náročnost na výpočetní výkon. Další kritérium metod, které ovlivňuje kvalitu výsledné kostry je náchylnost k šumu modelu, šumem jsou myšleny malé změny v povrchu modelu.

Metody většinou generují křivkovou kostru, to je 1D abstrakce původního 3D modelu tvořená křivkami, ta je pro účely animací 3D modelů stále moc složitá a převádí se na jednodužší IK (inverse-kinematics) kostru, která se skládá z propojení malého počtu rovných čar, reprezentujících podčásti modelu (například předloktí, jedlotlivé články prstů).

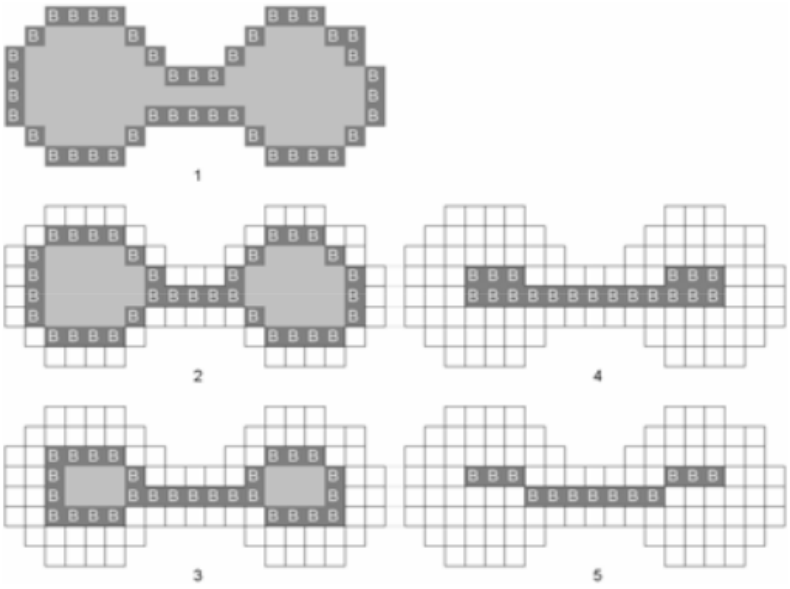
Využití

Využití koster je velké. Jedním je navigace ve virtuálním prostoru, kde se generuje cesta bez kolizí. Používá se například ve virtuální endoskopii, pro virtuální, bezkolizní průchod lidskými orgány. Další použití je v počítačové grafice k animaci 3D modelů, používá se IK kostra a velmi usnadňuje tvorbu animací. Ve zdravotnictví je kostra využívána k analýze a kvalifikaci anatomických struktur ze skenovaných dat, jako jsou žíly a nervové struktury, pro které byly vytvořeny specifické algoritmy. Pro rozpoznávání 3D objektů se používá kostra pro porovnání s kostrami známých objektů. Využívá toho, že kostra je zjednodušená verze 3D modelu. Další využití v počítačové grafice je transformace více modelů do jednoho nebo rozdělení 3D modelu na jeho podčásti. Existují také různé metody opravy nebo rekonstrukce neúplného 3D meshe. Kostry mají využití i v analýze dat, díky vlastnosti zjednodušení (abstrakce), které kostry přináší. Používá se i opačný případ, kdy se vytváří kostra a z ní se generuje model. Například podle váh na kostře udavájící tloušťku v daném místě.[5]

Rozdělení

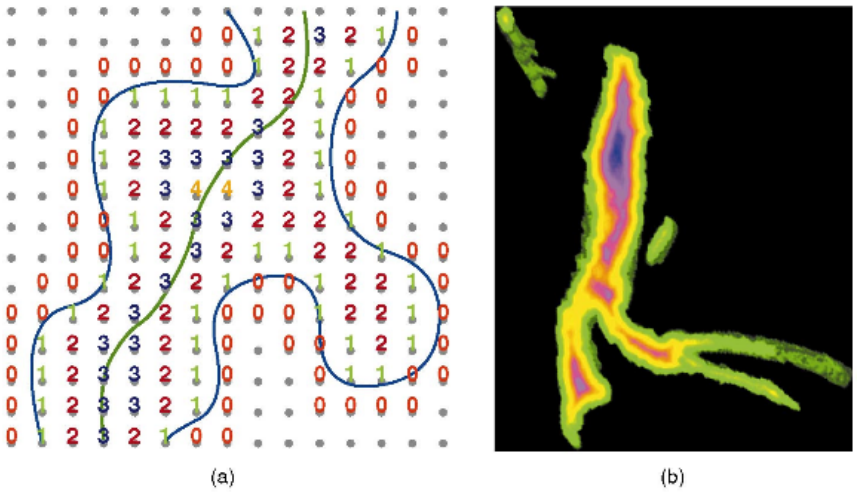
Metody vyjmutí kostry z modelu lze rozdělit na základní čtyři kategorie. Ačkoliv jsou metody rozděleny do kategorií, většinou metody využívají pricipů dvou nebo více kategorií dohromady. Metody se také mohou dělit na 2D a 3D, s tím že některé se principiálně mohou používat na oboje, příkladem je získání kostry pomocí 3D siluety, která je rozšířením 2D metody [11, 5].

Zeštíhlovací metody iterativně odebírají voxely z povrchu 3D modelu, dokud nedosáhnou požadované tloušťky, přitom se snaží zachovat topologii modelu. V každé iteraci jsou testovány všechny vnější voxely pokud je podle podmínek lze odebrat nebo zachovat. Postup je na ukázán na obrázku 1.3. Tyto podmínky jsou většinou vyhodnocované prohledáváním lokálního prostoru kolem daného voxelu za pomocí masek (například 3x3x3, případně větší). Největším problémem je nastavení zeštíhlování tak, aby nebylo odebráno více voxelů než je potřeba, problém činní krajové voxely, pokud by nebyly zavedeny podmínky řešící zachování krajových voxelů kostry, docházelo by k degeneraci kostry do jediného voxelu. Zeštíhlovací metody jdou rozdělit dále na směrové zeštíhlovací metody, metody se sekvenčním zeštíhlováním podčástí a plně paralelní. Směrové zeštíhlovací metody odebírají voxely v každé iteraci pouze z jednoho směru. Metody se sekvenčním zeštíhlováním podčástí nejdříve rozdělí diskrétní prostor na podčásti a poté provádí iterace nad podčástech zvlášť. Plně paralelní metody v jedné iteraci posoudí odstanění nebo zachování všech povrchových voxelů a až poté je odstraní.



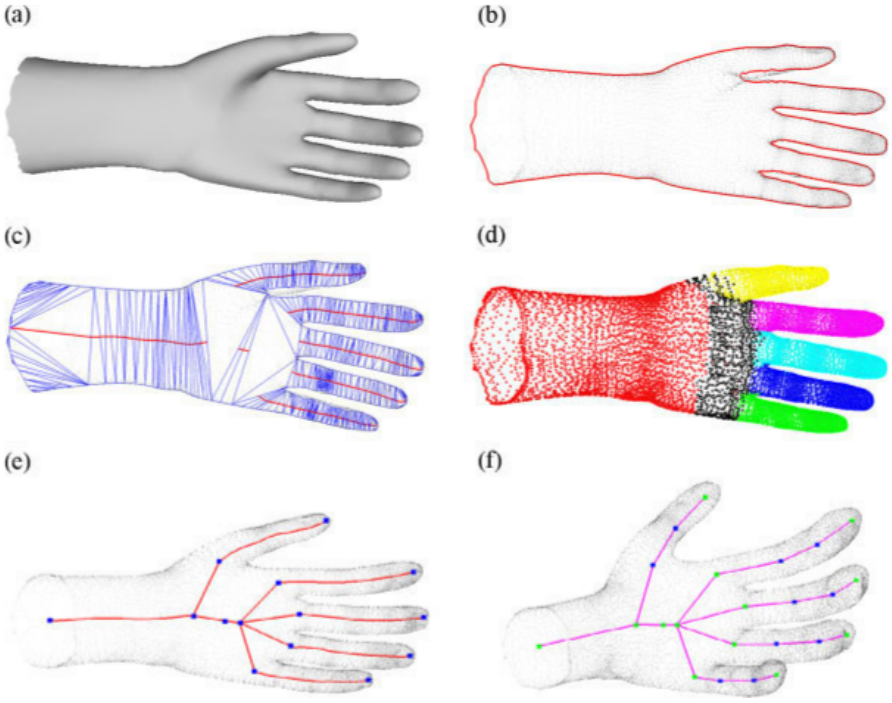
Obrázek 1.3: **Zeštíhlování**, ukázka na 2D modelu. Povrch je tmavý s písmenem "B", ukázány jsou postupné iterace (převzato z [5]).

Metody používající pole vzdáleností Pole vzdáleností definuje pro každý vnitřní bod modelu nejkratší vzdálenost k povrchu modelu. Může používat různé vzdálenostní funkce jako je Eukleidovská vzdálenost nebo různé aproximace. Z pole je následovně nutno najít kandidátní voxely, ze kterých bude tvořena kostra modelu. Existují různé metody pro nalezení kandidátních voxelů. Jedním způsobem je použití již výše zmíněného zeštíhlování, které bude využívat informace vzdálenosti k postupnému odstraňování krajních voxelů. Dalším způsobem je procházení gradientu a postupné značení kadnidátních voxelů [3]. Ukázka je na obrázku 1.4. Některé metody vyprodukují velké množství kandidátních voxelů, které je následně nutné redukovat a vytvořit ze zbytku finální kostru. Pro redukci se využije například shlukování. Používané metody pro finální propojení kandidátních voxelů jsou kupříkladu metody minimální kostry, nalezení nejkratší cesty a další grafové algoritmy. Možnost je i nejdříve propojit, například najít nejkratší cestu a pak redukovat nepotřebný zbytek.



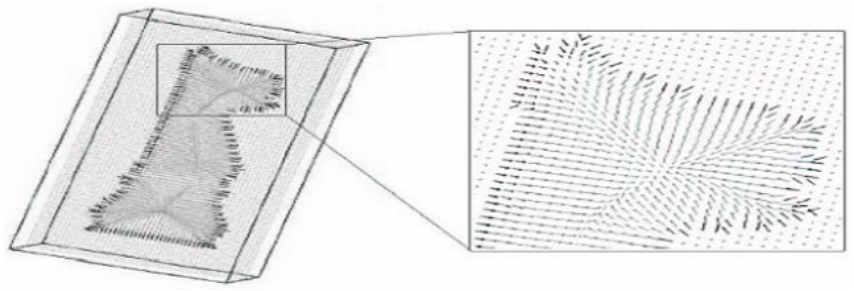
Obrázek 1.4: **Pole vzdáleností**, řez 3D modelem. (a) Diskrétní vzdálenost od povrchových hodnot pole (0 značí povrch v poli) zaokrouhlená na integer. Zelená křivka je proložením výsledné kostry. (b) Vzdálenost vyjádřena spojitě barevnou mapou, zelená–nejkratší vzdálenost a modrá–nejdelší vzdálenost (převzato z [3]).

Geometrické metody jsou používané na modely reprezentované polygonálními meshy nebo seskupením bodů. Častý způsob je použití Voroného diagramu generovaného buď vertexy v případě polygonálního meshe nebo samotnými body seskupením bodů. Poté se pomocí Voroného diagramu může aproximovat středová plocha modelu a po jejím zjednodušení na 1D získáme kostru. Další metodou je vměstnání maximálních kruhů (2D) nebo koulí (3D), jejichž středy jsou body kostry. Metody používající Reebův graf, který je popisem původního tvaru, kde hrany reprezentují podčásti modelu a uzly jejich propojení. Samotný Reebův graf ale není kostrou a musí být vestavěn do modelu. Existují různé variace a rozšíření Reebova grafu. Možností je i shlukování vertexů nebo bodů a pak jejich propojení na hranici sousedících shluků. Existují i metody, které rovnou transformují (smršťí) původní model do objemu blízkému 0 a následně transformují výsledek transformace na křivkovou kostru, příkladem může být vyjmutí kostry pomocí Laplacova vyhlazování [1], tuto metodu popisuje obrázek 1.5.



Obrázek 1.5: **Geometrická metoda.** Použití Delauného triangulace na získanou 3D siluetu modelu. (a) původní model. (b) získání 3D siluety z modelu. (c) aplikovaná Delauného triangulace na 3D siluetu. (d) Analýza Delauného triangulace, rozdělení na podčásti. (e) Základní křivková kostra. (f) Křivková kostra převedana na animační IK kostru. (převzato z [11]).

Metody používající funkce obecných polí Obecnými poli jsou myšlena ostatní pole než jsou pole vzdáleností, a to jsou obecná pole potenciálů, kde pro vnitřní bod je vypočítán potenciál jako suma potenciálů generovaná povrchem modelu. Příkladem je funkce elektrostatického pole použitá pro generování potenciálu v modelu, funkce viditelné odpuzovací síly, založená na Newtonovské odpuzující síle. Obecně je kostra vytvořena z lokálních extrémů, lokální extrémy jsou nalezeny prohledáním vzniklého vektorového pole. Na obrázku 1.6 je ukázka pole odpudivích sil. Oproti polím vzdáleností jsou méně senzitivní na šum povrchu, jelikož využívají plochu povrchu modelu a ne jen jeden nejbližší bod, na druhou stranu jsou složitější na výpočetní výkon.

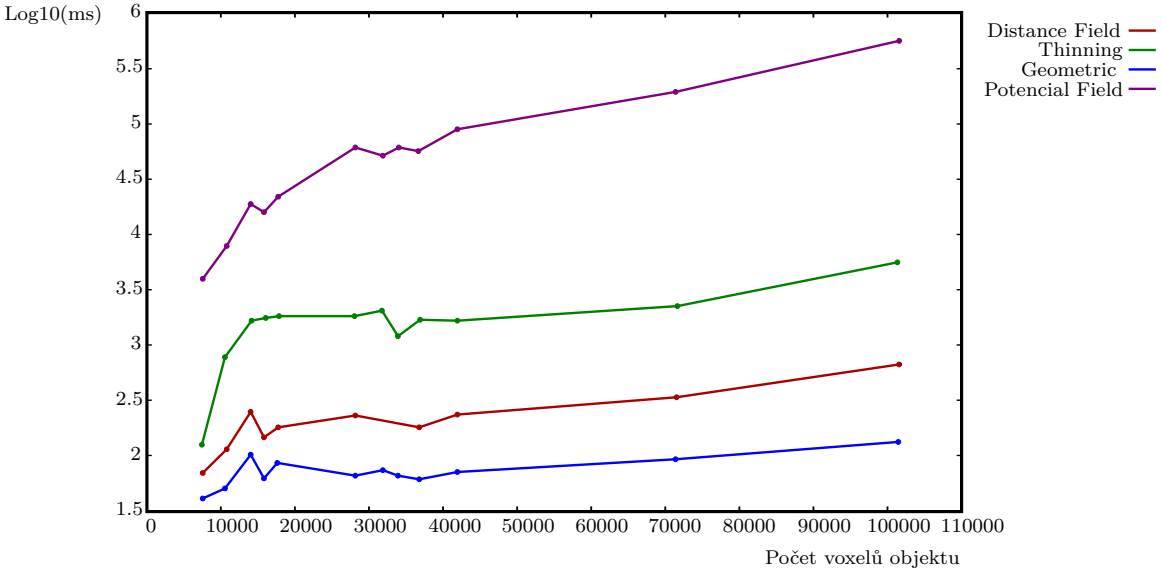


Obrázek 1.6: **Pole odpudivích sil.** 3D šachová figurka. Odpudivé síly jsou znázorněny vektory, na obrázku jde vidět lokální extrémy (tmavší křivka uprostřed modelu), kterými bude proložena kostra (převzato z [6]).

Další přístupy Existují další přístupy například získání kostry z 3D dynamického meshe, to je mesh, který má definovanou animaci neboli změnu modelu v čase. Pomocí analýzy této změny modelu v čase je schopen rozpoznat části 3D modelu a za pomocí reebova grafu vytvoří kostru. [13]

Srovnání přístupů metod

Ačkoliv jde porovnat základní myšlenka každé kategorie, uvedeno na obrázku 1.7, tak existuje obrovské spektrum metod. Většina metod se skládá z několika kategorií dohromady (více kroků metody) a taky každá metoda může pro určité případy generovat špatné výsledky. Některé metody potřebují pomoc uživatele, jako je nastavení vah nebo jiných parametrů. Metody založené na principu získání středové plochy potřebují provést další operace aby získaly ze středového prostoru výslednou kostru. Metodu vybíráme hlavně podle požadavků použití, tedy záleží jaké modely (vstupní data) budeme potřebovat zpracovat to může být nejen tvar, ale i třeba orientace, která může některým metodám vadit a některým ne. Požadavky na rychlost můžou být také důležité, s tím se pojí volba metody pro získání vedlejšího produktu navíc, třeba pole potenciálů, které pak lze využít pro další zpracování. Dalším důležitým prvkem je výstup metody. Výstupy se mohou velmi lišit, a to jak kvalitativně, tak reprezentací výstupu jako takového. S výstupními daty může být potřeba provést nějaké úpravy, to ovšem záleží na potřebě dané implementace.

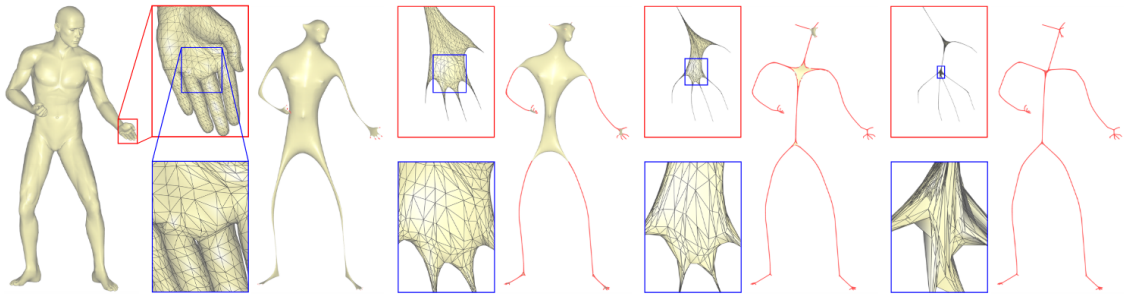


Obrázek 1.7: **Srovnání výpočetních nároků.** podle pořadí od nejrychlejších (od spodu): Geometrické, Pole vzdáleností, Zeštíhlovací, Pole potenciálů (převzato z [6]).

Příkladem může být Laplacovo vyhlazování [1]. Metoda postupně v iteracích aplikuje Laplacovo vyhlazování a tím smršťuje model do podoby blízké nulovému objemu. Výhodou je v tomto případě použití polygonálního meshe, metoda nepotřebuje převést model do objemové reprezentace, tím ušetří výpočetní náročnost. Obrázek 1.8 ukazuje postupné iterace. Musí se opatrně zadat váhy, aby metoda fungovala dobře. Špatné váhy smrští model moc a výsledná kostra pak ztratí na kvalitě, části kostry můžou zmizet, protože v extrémním případě vyhlazování skončí v bodě. Naopak správné váhy vytvoří kostru se správnou topologií modelu.

Skinning je proveden jednoduše, jelikož metoda při smršťování zachovává vertexy, ty jdou přiřadit při shlukování ke kostem. Avšak kostra je pořád potřeba převést na IK kostru, vhodnou pro animaci 3D modulů.

Nejvýraznější kategorie této metody je Geometrická (metoda pracuje přímo s polygonálním meshem), ale způsobem spíše Zeštíhlovací (ačkoliv neostraňuje voxely, tak prakticky zeštíhluje).



Obrázek 1.8: **Laplacovo vyhlazování.** Postupné iterace, přiblížení na změny po smrštění. (převzato z [1])

Kapitola 2

Blender rozšíření řešící rigging

Mimo obecných rozšíření řešících rigging existují rozšíření pro specifické použití, jako je například rigging automobilů, rigging obličejů. Zaměření této práce je obecné řešení riggingu modelů. V současnosti dominují dvě rozšíření s úplně rozdílnými přístupy metod.

- Rigify – založeno na vestavění již existujících částí kostry z 3D modelu a tudíž lze využít již existujicí animace přímo vytvořené pro danou kostru.
- Auto-Rig Pro – rozsáhlé rozšíření založené na vyjmutí kostry z 3D modelu. Zaměřuje se zejména na rigging bipedálních postav.

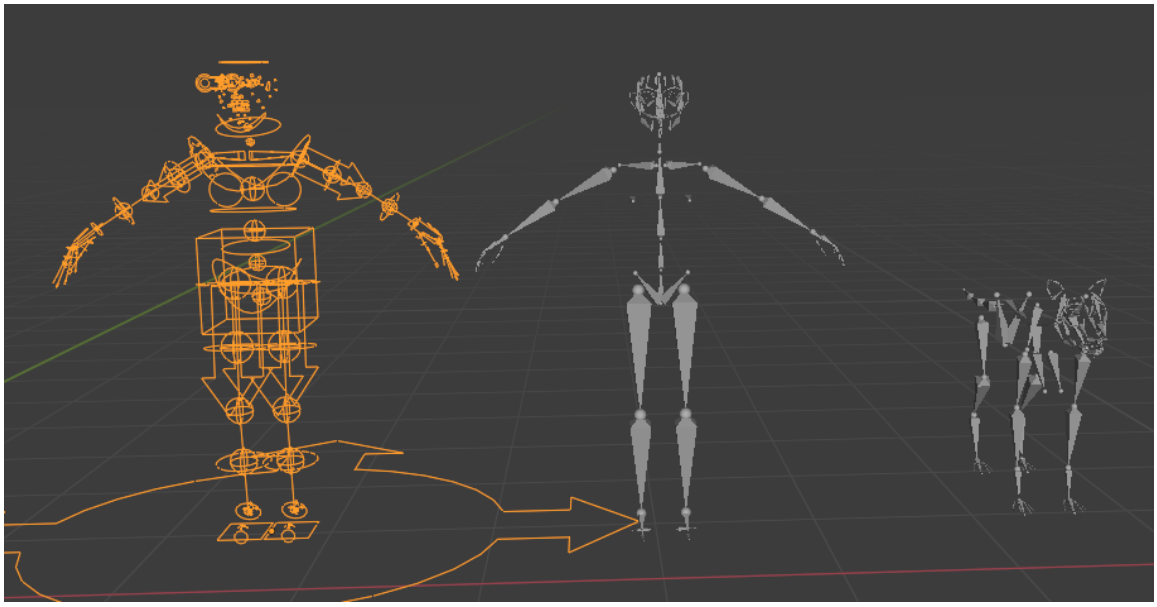
2.1 Rigify

Rigify je vestavěné rozšíření Blenderu, které urychluje práci manuálního vytváření rigu. Založené je na vestavování meta-rigu do modelu, avšak nejde o automatizované vestavění, to musí udělat uživatel sám a vygenerovaná kostra se nenapojí na model (skinning), to také musí uživatel udělat sám. Tím nastává předpoklad na zkušenosti uživatele s rigováním modelů, a tudíž není použitelný pro nezkušené uživatele. Meta-rig je složenina předem vytvořených částí jako jsou ruce, nohy, páteř, hlava, obličej, které se jen napojují na sebe. Výhoda meta-rigu je sdílení animací pro modely používající stejný meta-rig, použití předvytvořených animací pro daný meta-rig a více modelů. Samotné rozšíření poskytuje základní meta-rigy, které jdou upravovat a rozšiřovat. Po vytvoření meta-rigu a jeho nastavení tak, aby odpovídal originálnímu modelu, jde vygenerovat kontrolky rigu podle modelu. Pomocí kontrolek je jednodužší animování modelu, než přes samotné kosti. Na obrázku 2.1 je ukázka meta-rigů a kontrolek. Velkou výhodou je, že je zdarma a rovnou přibalený k Blenderu, takže stačí pouze zapnout.

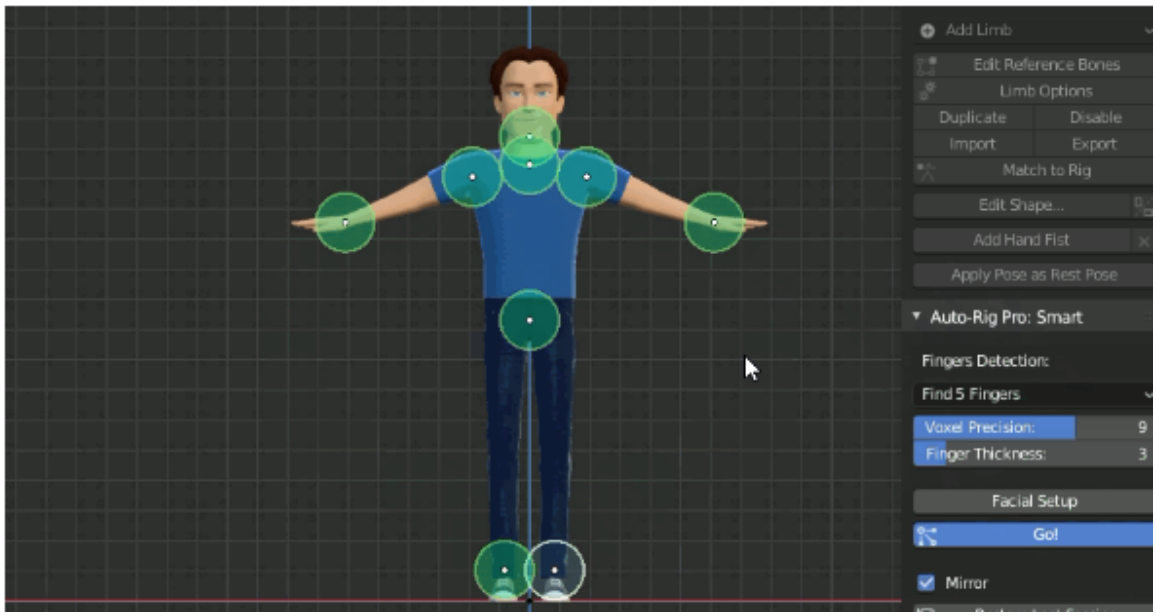
2.2 Auto-Rig Pro

Vytváří kostru z modelu za pomocí specifikace kritických bodů uživatelem, uživatel postupně určí kde se nachází části těla jako jsou krk, ramena, zápěstí, přídavně je možné specifikovat obličej. Ukázka určení částí těla je znázorněna na obrázku 2.2. Za pomoci této specifikace generuje kostru a z ní dokáže vygenerovat kontrolky, ty jsou dále potřeba navázat zpět na model (skinning), které taky zajišťuje rozšíření. V uživatelském rozhraní je nastavení, jako je citlivost metody. Rozšíření nezvládne rigovat příliš jednoduché modely, modely, které mají složité předměty, jako náramky, nástroje, mají prsty moc blízko

sobě. Také je omezeno pouze na bipedální modely. Pořád ale negeneruje bezchybně a tak je potřeba finální úprava, ale z velké části proces zjednodušuje široká sada nástrojů pro úpravu rigu. Nevýhodou je, že je to placené rozšíření.



Obrázek 2.1: **Rigify.** Oranžové nalevo jsou vygenerované kontrolky z prostředního meta-rigu, pro ukázku jsou uvedeny meta-rig člověka a vlka.



Obrázek 2.2: **Auto-Rig Pro.** Body označují části těla pro usnadnění generace kostry, vpravo jsou vidět prvky uživatelského rozhraní. (převzato z [2])

Kapitola 3

Návrh rozšíření

Zatím neexistuje metoda, která by perfektně řešila všechny případy automatického vytváření kostry z modelu, ale naopak přibývají stále nové metody řešení, které jsou v něčem lepší a v něčem horší než ostatní. Klíčové je vybrat správnou metodu pro cílové použití.

Co se týče oblasti kolem Blenderu a jeho rozšíření, tak existují spíše rozšíření pro urychlení práce rigování modelů. Stále je ale potřeba znalosti a zkušenosti člověka zabývajícího se riggingem. Je pravda, že v této oblasti se stanovenému cíli této práce dostává nejblíže Auto-Rig Pro, který se snaží práci rigování automatizovat, ale není otevřený, rozšiřitelný ani zdarma. Pořád u něho zůstává finální úprava kostry po vygenerování, tato úprava by v ideálním případě nebyla nutná. Ale to je skoro nemožná vlastnost, vždy se najde u pokročilých uživatelů nebo profesionálů potřeba věci upravit podle jejich potřeb k dokonalosti. Tento projekt cílí na běžného uživatele, který nemusí umět rigging, ale aby mu rozšíření co nejvíce automatizovalo tuto činnost bez jeho zásahu a zároveň umožnilo náročnějším uživatelům jednoduchou úpravu výsledné kostry. Hlavně cílí na otevřenost a inovaci, aby lidé mohli jednoduše přispívat například novými způsoby řešení (nové metody) a zároveň aby bylo rozšíření zdarma dostupné. Tím cílí na experimentování s novými metodami.

Základem této práce je navrnout kostru rozšížení pro Blender, tato kostra bude obsahovat mechanismus načítací vytvořené metody řešící rigging, ze kterých si bude moct uživatel vybrat, kterou chce použít. Současně musí být rigovací metody jednoduše rozšiřitelné, toho dosáhne dynamickým importováním rigovacích metod z jedné určité složky metod. Vytváření nových metod bude zjednodušeno předpřipravenou šablonou rigovací metody, kterou bude potřeba pouze vyplnit, implementovat potřebné metody. Využije prvků uživatelského rozhraní Blenderu, pro vytvoření uživatelského rozhraní, které bude jednoduše ovladatelné a kompatibilní s Blenderem.

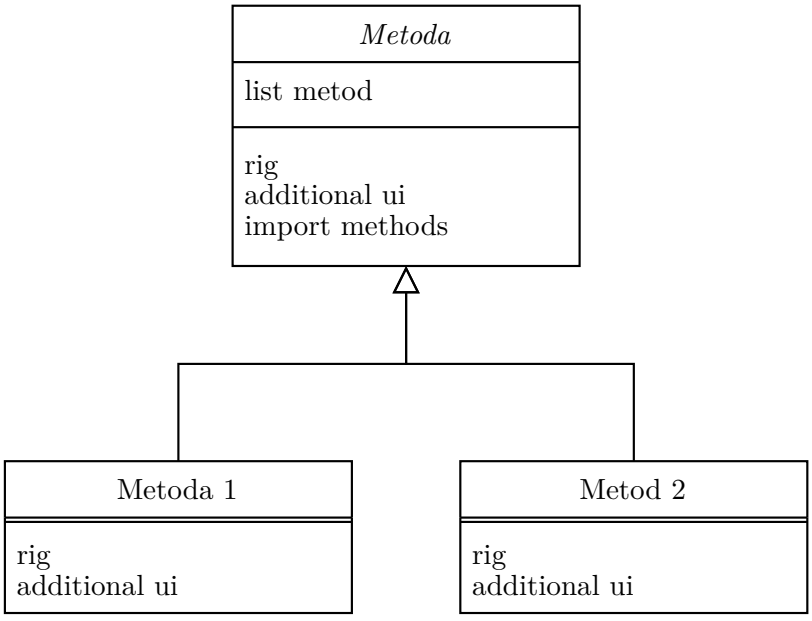
Zároveň rozšíření implementuje jednu metodu, a to metodu vyjmutí kostry z modelu pomocí 3D siluety [11], na které budou ukázány funkce rozšíření. Metoda byla vybrána díky jejím vlastnostem práce s polygonálními meshy, tím eliminuje nutnou transformaci reprezentace 3D modelu, jelikož ve většině případů se v Blenderu pracuje s polygonálními meshy. Ve spojení s tím, že to je geometrická metoda, která má malou výpočetní náročnost a tím rychlejší vytvoření kostry než ostatní typy metod. V průběhu vytváření kostry metoda umožňuje návázání kostry na model (skinning) pomocí Heat algoritmu. Ačkoliv je tato metoda relativně rychlá, tak si zachovává relativně přijatelnou kvalitu výsledku.

3.1 Kostra rozšíření

Kostra funguje s nadtřídou rigovacích metod `rigger_method`, která vytváří rozhraní pro konkrétní rigovací metody. Konkrétní metody jsou potomky `rigger_method` a implementují metodu `rig()`, ve které se vykonává vlastní rigging a metodu `additional_ui()`, která rozšiřuje panel uživatelského rozhraní o další prvky, které se zobrazí po vybrání konkrétní rigovací metody v dropdown menu. Tato struktura je znázorněna na schématu 3.1

Rozšíření o další metody je zjednodušeno pomocí šablony, ve které stačí dát jméno zděděné třídě `rigger_method` a implementovat rigovací metodu v metodě `rig()`, popřípadě implemontovat rozšíření panelu, ve kterém budou dodatečné parametry. Pro rozšíření panelu je nutné implementovat metodu `additional_ui()`, je vhodné vytvořit Blender `PropertyGroup`, ve kterém budou jednotlivé hodnoty, které je možné použít v metodě `additional_ui()`. Před použitím je však PropertyGroup nutno registrovat. Další možností rozšíření panelu jsou položky typu `Operator`. Operátory jsou v uživatelském rozhraní Blenderu většinou tlačítka, operátory provádějí všemožné funkce. Operátor je nutné buď vytvořit a registrovat, pak lze použít v uživatelském rozhraní nebo volat z kódu, druhou možností je použít jakýkoliv již vytvořený a registrovaný operátor. Rodičovská metoda obsahuje seznam potomků, kteří jsou načítáni pomocí metody `import_methods()`. Zděděné třídy jsou rozděleny do vlastních souborů ve složce `methods`, ze které jsou dynamicky importovány a vloženy do seznamu implementovaných metod, které jsou pak nabídnuty uživateli jako položky dropdown menu v panelu.

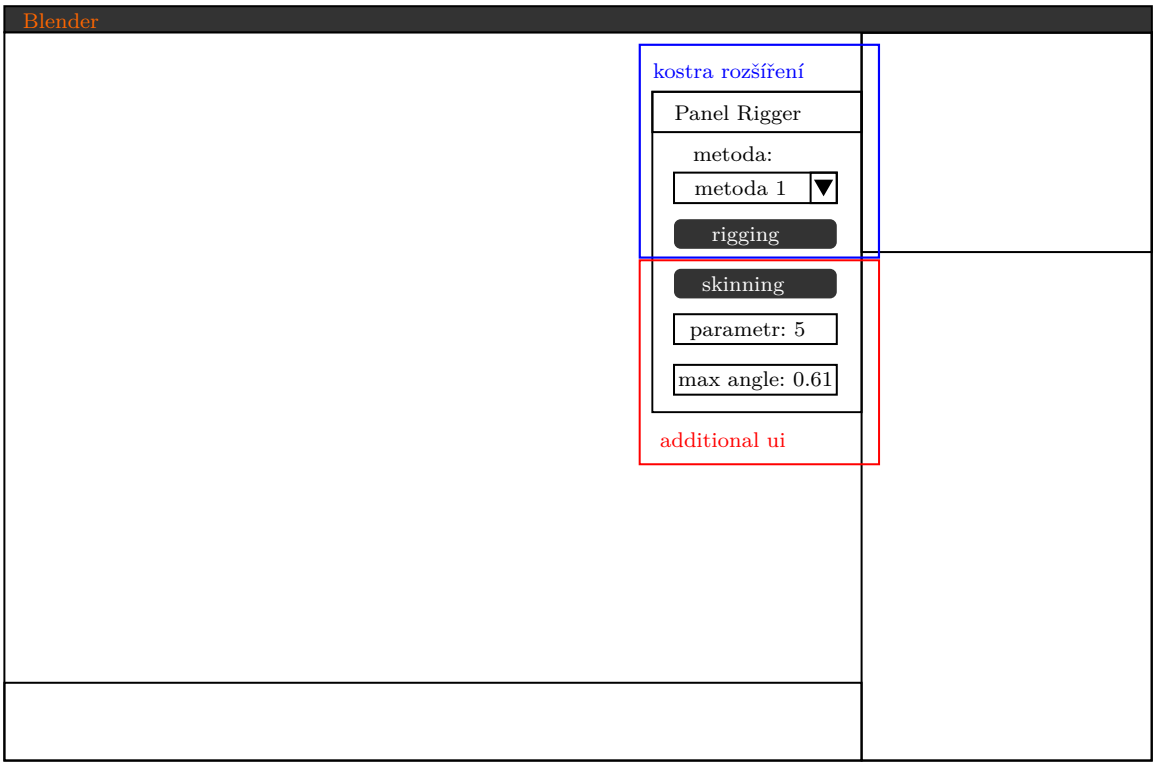
Zbytek kostry zpřístupňuje prostředky vybrané metodě, jako například přístup k označenému modelu k rigování, předání reference panelu pro přidání rozšiřujících parametrů nebo tlačítek metody.



Obrázek 3.1: **Metody – struktura tříd.** Rodičovská třída Metoda obsahuje statický list dědících Metod, které importuje pomocí importovací metody. Zděděné metody implementují metodu rig(), která provádí rigging modelu a metodu additional ui, která rozšiřuje uživatelské rozhraní o další elementy.

3.2 Uživatelské rozhraní

Základem uživatelského rozhraní je panel rozšíření, znázorněn na obrázku 3.2. Panel obsahuje dropdown menu, jehož obsahem je seznam implementovaných metod, provádějících rigging, ze kterých si uživatel vybere, kterou chce použít. Seznam rigovacích metod se dynamicky načítá ze složky rigovacích metod ve složce rozšíření. Dále obsahuje tlačítko pro start rigovací metody zvolené v dropdown menu. Některé metody budou potřebovat externí parametry, které se budou moci nastavit dále v panelu. Po vybrání metody v dropdown menu se zobrazí rozšiřující sekce panelu. V rozšířené sekci bude možné parametry rigovací metody měnit. Uživatelské rozhraní používá prvky Blender API, pomocí kterého je implementováno. Tím zajišťuje kompatibilitu a plynulý přechod rozšíření do Blenderu jako celku.



Obrázek 3.2: **Panel rozšíření** je rozdělen na dvě části. Část kostry rozšíření je statická, obsahuje dropdowwn menu se seznamem implementovaných rigovacích metod a tlačítko rigging spouštějící danou metodu. Druhá část additional ui je volitelná část, její obsah se mění podle vybrané metody, u některých metod může chybět.

3.3 3D silueta

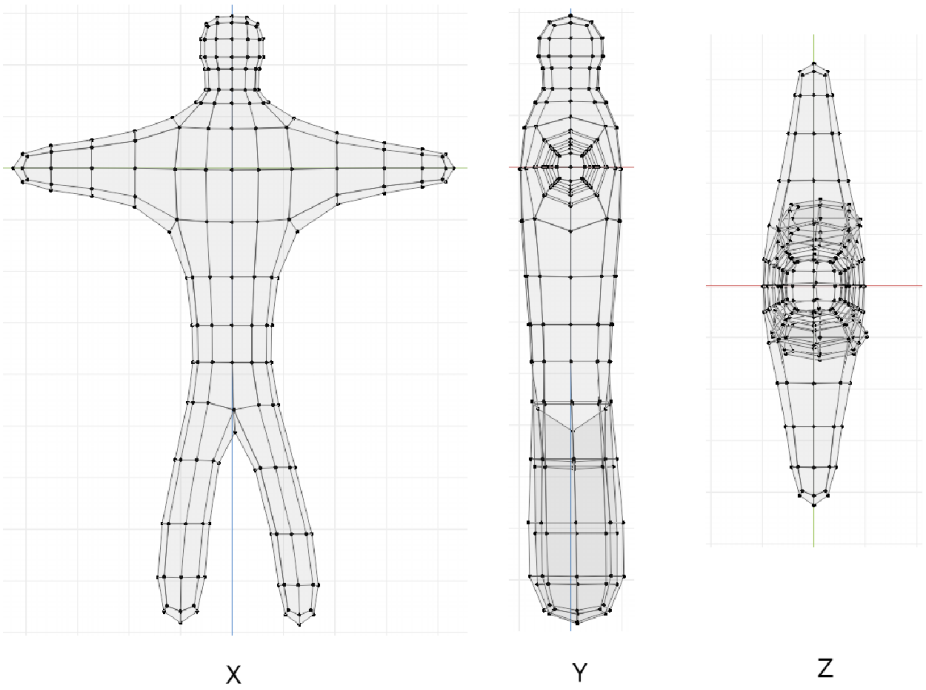
Metoda rozšiřuje metodu používanou na 2D modely. Ve 2D máme již siluetu jako samotný model a tak stačí pouze najít středovou osu a proložit jí křivku. Ve 3D se to provede pomocí dvou na sebe kolmých pohledů, pro jednoduchost se používá náhledu podle nějaké osy, pak se třetí osa zanedbá a vznikne dvojrozměrná skupina vertexů. Ze skupiny 2D vertexů musí získat obrys–siluetu. To provede nalezením nejvyššího vertexu, ten určí jako začáteční vertex a snaží se o maximalizaci úhlů při hledání následujícího vertexu. Maximalizace úhlů se provádí jako získání vertexu z okolí, se kterým předchozí dva vertexy svírají největší úhel. Vertexy v okolí jsou vertexy, které mají vzdálenost od současného vertexu menší nebo rovnu délce nejdelší hrany vycházející ze současného vertexu. Po nalezení počátečního vertexu je výsledkem 2D silueta s hloubkou (3. osa), tedy 3D silueta.[11]

Nejdříve musí metoda zjistit, podle kterých os se bude získávat kostra. To zajistí podle analýzy variance vertexů z pohledu dané osy, znázorněno na obrázku 3.3. Hledá osu s největší variancí, výpočet 3.1, tu pak určí jako primární.

Variance značí průměrnou odchylku hodnot od střední hodnoty. Metoda se snaží nalést promítnutí s největší variancí, aby co nejvíce eliminovala možnost zastínění částí modelu, a tím získání nejlepší 3D siluety modelu.
Výpočet variance x:

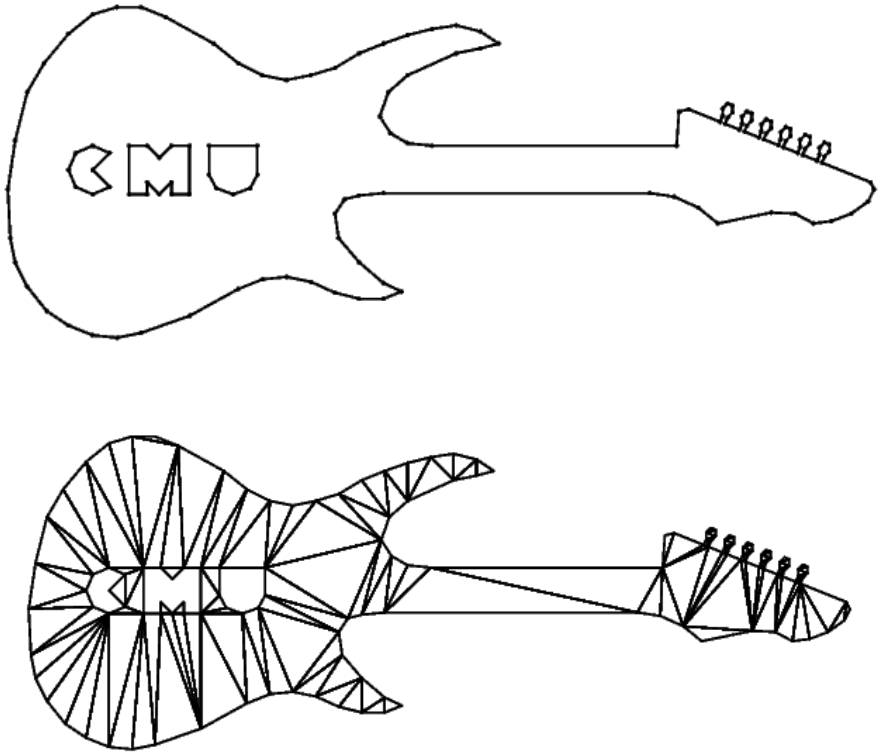
$$var(x) = \frac{\sum_{i=1}^{n}(y_i - m_y)^2 + \sum_{i=1}^{n}(z_i - m_z)^2}{n-1} \tag{3.1}$$

Kde, var(x) je hodnota variance promítnutí x, n je počet vertexů, m je střední hodnota souřadnic v dané ose a y_i, z_i jsou y,z souřadnice i–tého vertexu.



Obrázek 3.3: **Variance promítnutí**. Největší varianci má promítnutí podle X, druhou největší má Y a nejmenší Z.

Po získání osy vytvoří 3D siluetu, průchodem vertexů modelu a značením obrysových vertexů, na ten je následně použita omezená Delauného triangulace.

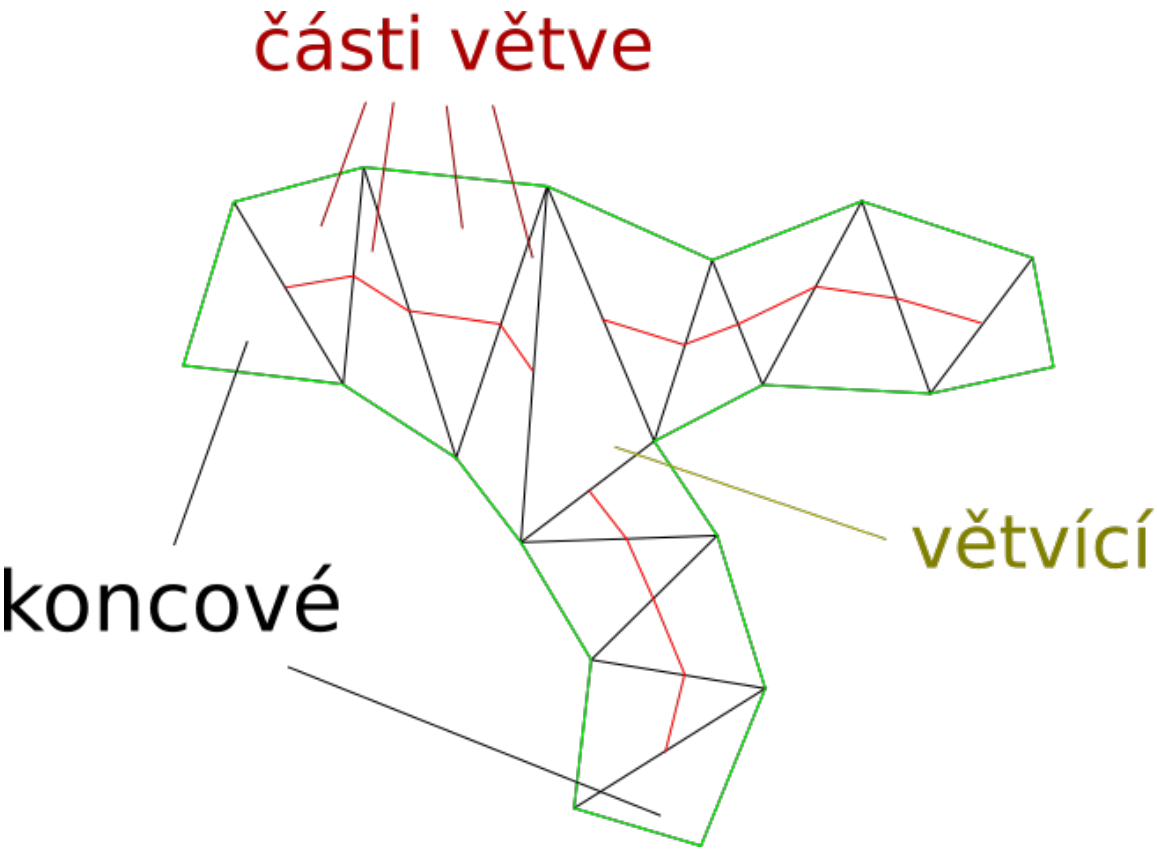


Obrázek 3.4: **2D model kytary**, stejný model triangulován omezenou Delauného triangulací. (převzato z [12])

Delauného triangulace provádí triangulaci skupiny vertexů, přičemž splňuje vlastnost, že obsah kruhu definovaného vrcholy trojúhelníku neobsahuje žádný jiný vrchol. Omezená Delauného triangulace pracuje navíc s hranami, které omezují triangulovaný prostor. Omezená Delauného triangulace je vidět na obrázku 3.4. Omezená Delauného triangulace se blíží Delauného triangulaci s dodržením omezení hran. Proto ne všechny trojúhelníky Omezené Delauného triangulace splňují Delauného vlastnost o vrcholech v kruhu definovaném vrcholy trojúhelníku. [12, 10]

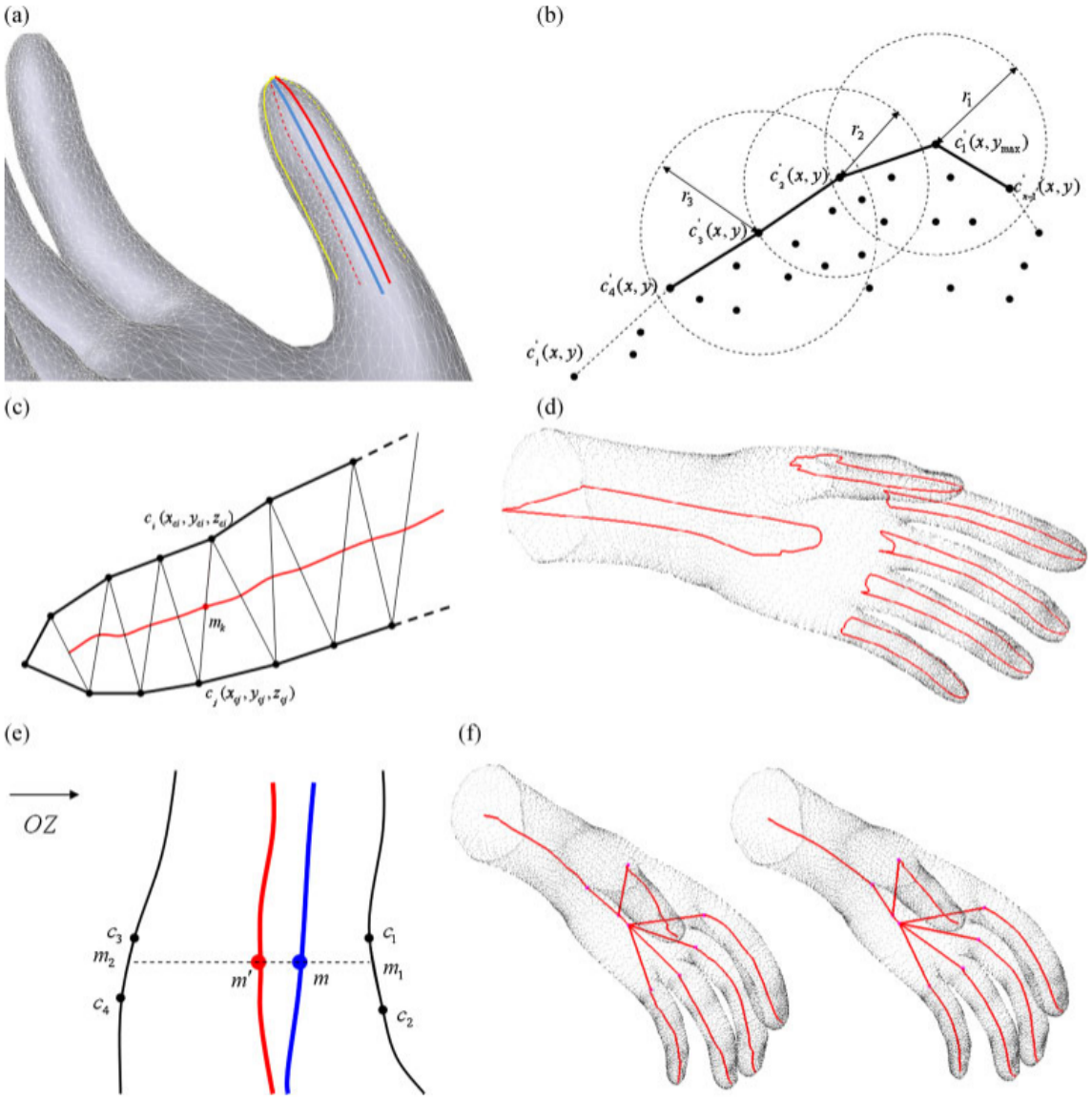
Z výsledku triangulace metoda pozná o jakou část se jedná tak, že trojúhelníky s dvěmi stranami obrysovými je vrchol kosti, pokud je pouze jeden obrysový tak je to průběh kosti a pokud není žádný obrysový, jedná se o větvení kostí. Rozdělení je znázorněno na obrázku 3.5. Spojením středních bodů vnitřních stran dostane části větví kostry, které dále potřebují spojit v místech větvení. Spojení větví proběhne prohledáním větvících trojúhelníků, kterým se vypočítá středový bod trojúhelníku a středové body hran větvícího trojúhelníku. Pro každý větvící trojúhelník se najdou větve, které mají stejné souřadnice počátečního nebo koncového bodu větve, jako souřadnice středového bodu jedné z hran větvícího trojúhelníku, nalezené větvi se změní nalezený začátek nebo konec na středový bod větvícího trojúhelníku.

Dále pro zarovnání křivky podle třetí osy, vycentruje křivku v modelu pomocí sekundární 3D siluety, která je kolmá na první siluetu a přitom má větší varianci ze zbylých dvou os. Postup je znázorněn na obrázku 3.6.



Obrázek 3.5: **Rozdělení trojúhelníků**. Rozdělení trojúhelníků vycházejících z omezené Delauného triangulace 3D siluety.

Skinning se provádí v mezikroku, kdy jsou vytvořeny pouze kosti a ještě není vytvořeno jejich propojení, to znamená, že jsou propojeny pouze vnitřní strany trojuheníků s jednou obrysovou stranou a právě na tyto kosti je provedeno shlukování vertexů, tím se přidělí vertexy ke kostem. Nakonec je potřeba vyřešit převedení křivkové kostry na IK kostru vhodnou pro animace. Počátek kostry se umístí na souřadnice středového bodu prvního větvícího trojúhelníku v seznamu větvících trojúhelníků. Od počátečku kostry se bude procházet seznamem větví a budou se hledat větve, které mají začáteční nebo koncové souřadnice jako středový bod větvícího trojúhelníku. Těmito větvemi se bude postupně procházet, pokud bude úhel mezi dvěma segmenty větve větší než daný limit nebo nastane konec větve, vytvoří novou kost a přidá ji do kostry. Pokud konec větve zkončí na souřadnicích dalšího větvícího trojúhelníku, bude se procházet stejně jako počáteční trojúhelník, jinak narazí na konec větve a pokud zbývají projít další větve, začne je procházet.



Obrázek 3.6: **3D silueta.** (a) Dvě na sebe kolmé 3D siluety a výsledná střední osa. (b) Postup získání obrysu 3D siluety. (c) Delauného triangulace a následné získání střední osy (křivkové kosti). (d) Nalezení sekundární 3D siluety, pro zpřesnění středové osy ve 3D, pro každý segment. (e) Zpřesnění středové osy podle sekundární 3D siluety. (f) Rozdíl mezi kostrou před a po zpřesnění. (převzato z [11])

Kapitola 4

Implementace

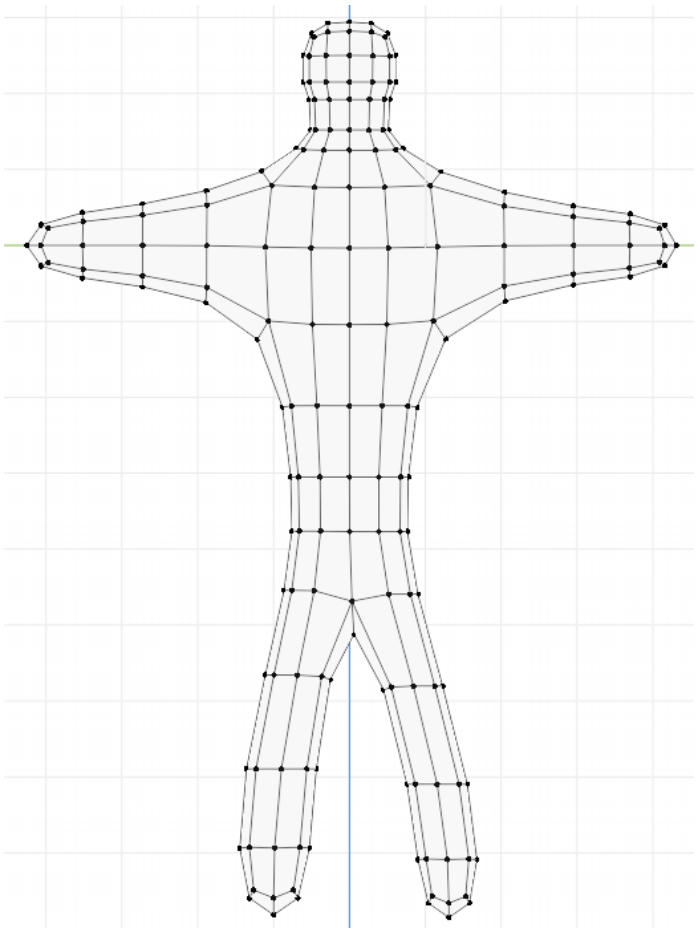
Cílem práce bylo implementováno rozšíření pro Blender nazvané Rigger, které provádí automatické vytváření kostry 3D modelu a umožňuje další jednoduché rozšiřování o další riggovací metody. Částí rozšíření je kostra, která se stará o provázání ostatních částí a zajišťuje jednoduchost dalšího rozšíření. Pomocí kostry jsou dynamicky načítané metody provádějící rigging (vytváření kostry). Jedna metoda byla implementována, konkrétně metoda využívající 3D siluetu a následnou omezenou Delauného triangulaci. Další důležitá část je uživatelské rozhraní, které je součástí kostry, a které je dále rozšiřováno podle zvolené metody. Uživatelské rozšíření je vytvořeno pomocí Blender API, tudíž používá prvky uživatelského rozhraní Blenderu a celý projekt je implementován v programovacím jazyce Python. Pro zjednodušení tvoření nových rigovacích metod, je předpřipravena šablona metody, kterou stačí doplnit.

4.1 Kostra rozšíření

Vlastní základ kostry je úplně normální Blender rozšíření, které pomocí `__init__.py` inicializuje rozšíření, registruje Blenderu použité třídy. Navíc provede dynamické importování rigovacích metod ze složky `methods`, která se nachází ve složce rozšíření. Zároveň vytváří objekt typu PropertyGroup, který obsahuje data jako jsou názvy importovaných rigovacích metod a slouží pro jejich navázání na uživatelské rozhraní Blenderu.

Jedna z registrovaných tříd je `rigger_method`, která slouží jako nadtřída konkrétních rigovacích metod. Právě `rigger_method` zajišťuje rozhraní pro již konkrétní rigovací metody, a to metodami `rig(objekt)`, která dostane právě aktivní objekt a má za úkol samotné vytvoření kostry. Model, který slouží k představení funkce je vidět na obrázku 4.1. Dále zprostředkuje metodu `additional_ui(layout)`, která tvoří rozhraní pro přidání dodatečných položek do panelu uživatelského rozhraní. Nadtřída se také stará o dynamické importování riggovacích metod a to ze složky `methods`, společně s importem vytvoří potřebné datové struktury pro uživatelské rozhraní. Třída `rigger_method`, obsahuje již dříve zmíněné datové struktury pro uživatelské rozhraní a to list `panel_list`. Python slovník `implemented_methods`, který obsahuje reference na instance rigovacích metod podle jména ve tvaru `{"jméno metody" : instance_metody}`, který pomáhá používat konkrétní rigovací metodu vybráním instance metody podle zadaného jména, tím umožňuje používání dynamicky importovaných, předem neznámých metod. Obsahuje i reference na vybraný objekt `rigged_object` a výslednou kostru `armature`, které jsou dále využívány pro případné další zpracování.

Další důležitou částí kostry rozšíření je `Rigger_operator`, je to třída typu `Operator`, třídy typu Operátor vykonávají nějakou činnost, jsou nutné registrovat Blenderu, ale pak jsou možné kdekoliv v Blenderu použít, a to v uživatelském rozhraní jako tlačítka nebo klávesové zkratky nebo prostým voláním. Právě `Rigger_operator`, zařizuje použití konkrétní rigovací metody na právě aktivní objekt, který metodě předá. Jako doplňující operátor kostra obsahuje `Skinning_operator`, která zařizuje skinning s automatickými váhami (heat algoritmus), poskytovaný Blenderem, na výsledek rigovací metody (dříve zmíněné `rigged_object` a `armature`).



Obrázek 4.1: **Model určený k testování** Model je promítnut v ose, ve kterém bude vykonávána rigovací metoda. Jdou vidět jednotlivé vertexy a hrany modelu, druhá půlka modelu je v zákrytu, jelikož má stejné souřadnice v promítnutí do 2D, ale jinou hloubku.

4.2 Uživatelské rozhraní

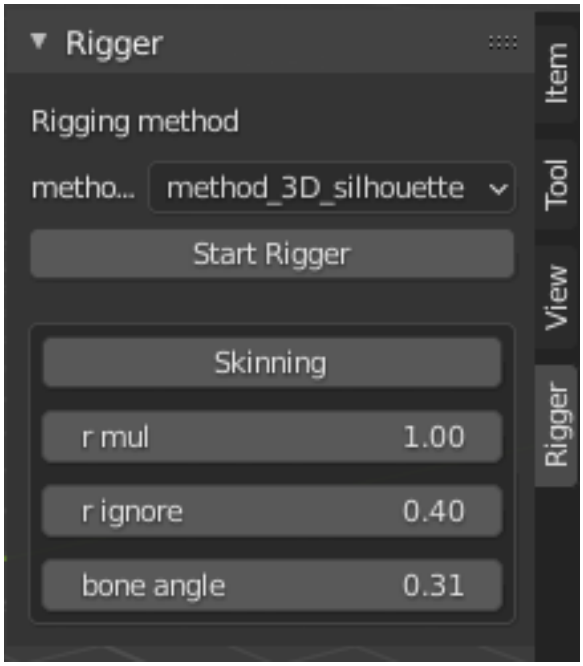
Rozšíření používá komponentu Blender uživatelského rozhraní `Panel`, kde vytváří nový panel pojmenovaný podle jména rozšíření Rigger. Komponenty v panelu se dají rozdělit na část kostry a část metody, která je volitelná.

Část kostry se skládá z dropdown menu, jehož obsah je dynamicky načítán a obsahuje dostupné rigovací metody, také obsahuje operátor `Rigger_operator`, který má podobu tlačítka, které vykonává metodu `rig()` rigovací metody zvolené v dropdown menu. Samotné dropdown menu je implementováno pomocí Blender typu `EnumProperty` v již zmíněném `PropertyGroup`, konkrétně `Rigger_Props`, který uchovává data o načtených metodách ve formátu (jméno metody pro referenci metody, jméno metody pro zobrazení v dropdown menu, popisek). Část kostry uživatelského rozhraní je k vidění na obrázku 4.2.

Obrázek 4.2: **Panel uživatelského rozhraní**. Ukazuje část kostry uživatelského rozhraní rozšíření. V panelu je dropdown menu, jehož obsahem jsou dynamicky načtené rigovací metody metody (na obrázku vpravo) a dále obsahuje tlačítko spouštěcí vybranou rigovací metodu.

Podle zvolené metody se rozšíří panel o další komponenty, a to pomocí implementace metody `additional_ui(layout)`, které se předá layout, což je reference panelu, a právě tomuto layoutu lze přidat další komponenty. V případě rigovací metody používající 3D siluetu, rigovací metoda rozšiřuje panel o operátor skinning a parametry metody jako je násobek r (radius–poloměr oblasti sousedních vertexů), modifikátor r značící vzdálenost ignorovaných vertexů a bone angle, úhel pro určení výsledných kostí. Rozšíření panelu v případě vybrání 3D siluety je vidět na obrázku 4.3.

Parametry jsou implementovány jako Properties v GroupProperty metody, která uchovává data a navazuje na panel uživatelského rozhraní, před použitím je nutné GroupProperty registovat Blenderu a udělat referenci pro přístup k položkám.



Obrázek 4.3: **Rozšířený panel** pro účely vybrané rigovací metody. Rozšiřující část je označená v rámečku, který obsahuje tlačítko provádějící skinning a tři parametry metody, které jsou udány číselnou hodnotou a jdou buď přímo nastavit na požadovanou hodnotu nebo lze klikem a táhnutím kurzoru měnit jejich hodnotu. Příklad ukazuje rozšíření panelu v případě metody používající 3D siluetu.

4.3 Metoda využívající 3D siluetu

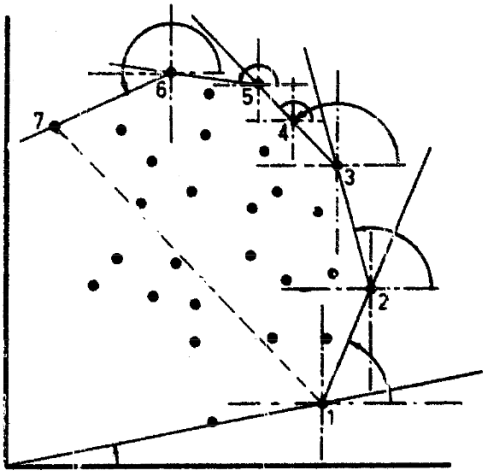
Pro vytvoření rigovací metody byla použita šablona vytvořená pro tento účel. Jelikož součást rigovací metody je použití omezené Delauného triangulace, nejdříve se vykoná import knihovny triangle přes pip. Pip je již součástí Pythonu přibaleného k Blenderu, tudíž ho není nutné instalovat. Knihovna triangle zprostředkovává knihovnu originálně v programovacím jazyce C/C++ řešící triangulace, tím lze využít již optimalizovaných algoritmů.[12]

1. Prvním a stěžejním krokem této rigovací metody je vytvořit 3D siluetu modelu, tento krok je velmi důležitý, jelikož se od něj odvíjí jak bude výsledek kvalitní a zda vůbec půjde kostru vytvořit. Jelikož metoda promítá model do 2D, může vznikat zastínění některých částí 3D modelu. Pro redukci zastínění je nutné správně vybrat osu ve kterém se model promítne, pro jednoduchost a rychlost se počítá pouze se zanedbáním jednoho rozměru (x,y,z). Pro automatické zjištění osy promítnutí se porovnává variance ve zbylých dvou osách (třetí je zanedbaná), výsledně se vybere promítnutí s největší variancí. Pro výpočet variance používá Python knihovnu statistics.

2. Po úspěšně zvolené ose může metoda začít hledat siluetu modelu, a to je právě část, která je obtížná na úspěšné dokončení. Základem problému je nalezení obrysu konkávního modelu (concave hull), metoda se snaží procházet model podobně jako Jarvis March, ten je ovšem pro konvexní modely, které jsou mnohem jednodužší definovat. Pro rozšíření hledání konkávního modelu rozšiřuje Jarvis march o prohledávání

pouze lokálního okolí definovaného kolem současného vertexu. Ačkoliv tato metoda pracuje rychle, tak v některých případech zkončí zacyklením. Rigovací metoda nejdříve najde počáteční vertex, a to ten, který má největší souřadnice v jedné ose. Pro příklad pokud je promítnutí podle `x`, metoda hledá vertex s největší souřadnicí `z`.

Jarvis march Algoritmus pro získání obrysu konvexního modelu, znázorněn na obrázku 4.4. Nejdříve musí najít krajní bod, který hledá tak, že najde počáteční bod mimo sadu bodů a otáčí přímku v nějakém směru, až najde první bod. Od Počátečního bodu hledá další bod s nejmenším úhlem, úhel ale musí být na stejnou stranu, jako rotovala počáteční přímka. Nalezený bod s nejmenším úhlem se stává novým bodem a hledá další bod s nejmenším úhlem, po směru jako v prvním kroku. Pokračuje hledáním obrysových bodů, dokud nenarazí na počáteční bod a uzavře obrys.

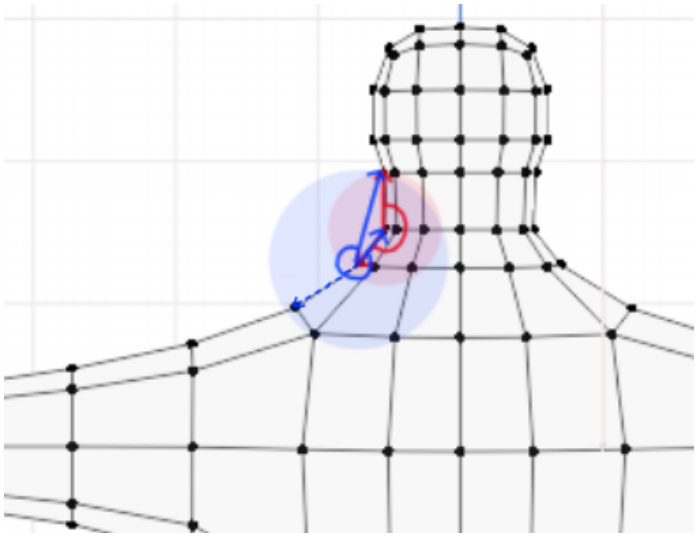


Obrázek 4.4: **Jarvis march**. Ukazuje geometrickou interpretaci algoritmu. (převzato z [9])

3. Poté metoda prochází okolní vertexy, okolím se myslí vertexy vzdálené maximálně jako nejdelší hrana vycházející z počátečního vertexu. Metoda se snaží vybrat z okolních vertexů vertex, se kterým maximalizuje úhel mezi předchozím, současným a novým vertexem. Vybírá do té doby, dokud nedorazí zpět na počáteční vertex a právě to může být v některých situacích problém. Na začátku vytvoří vektor (0,1), tedy směřující nahoru, a v kombinaci s nejvyšším bodem, tvoří neutrální začátek. Vytvoří i prázdnou siluetu, kterou tvoří list vertexů a list segmentů (hran), zároveň rovnou vloží počáteční vertex do siluety.

4. Následně iterativně prochází vertexy dokud nedorazí k počátečnímu vertexu nebo nebude mít žádné, ještě neprojité, sousední vertexy. V jednotlivých iteracích nejdříve získá hrany, které obsahují současný vertex. Z těchto hran najde nejdelší a tím určí rozsah okolí `r`.

   Současný vertex se přidá do seznamu použitých vertexů a prohledá všechny vertexy modelu, pokud mají vzdálenost od současného vertexu kratší nebo stejnou jako okolí `r`, tak se buď použijí jako kandidátní vertexy siluety na další iteraci nebo pokud jsou moc blízko současnému vertexu, do určité meze dané parametrem z uživatelského rozhraní, tak se vloží do seznamu použitých vertexů. Použití seznamu již projitých

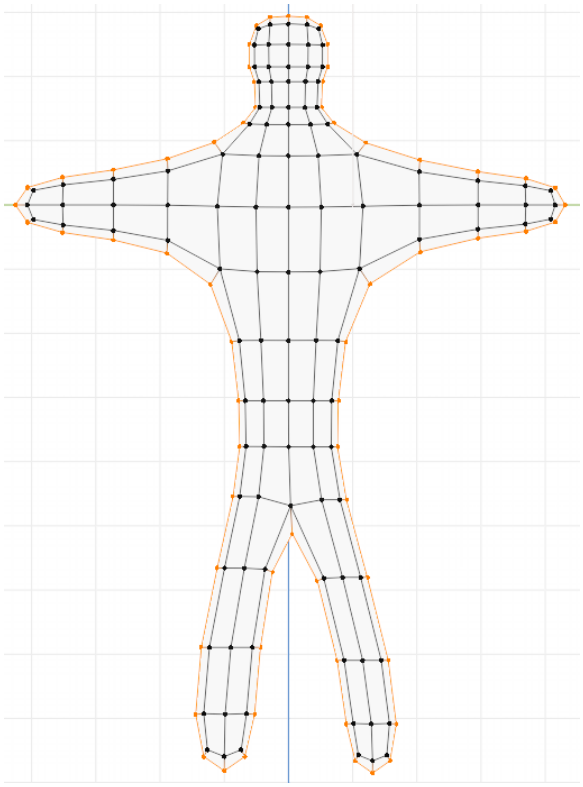


Obrázek 4.5: **Problém zacyklení** vzniká při hledání vertexu s největším úhlem. Na obrázku jsou vidět dva kroky, první je červený, který proběhne správně a získá následující vertex. Druhý krok je modrý, a ten, než aby našel požadovaný vertex, který je označen přerušovaně, nalezne již zpracovaný vertex, a tudíž začíná cyklit. Problém nastává hlavně v konkávních částech modelu. Problémem jsou vertexy, které spadají do oblasti sousedů, ale které jsou za prvním vektorem, v našem případě nalevo, tudíž dosahují velmi vysokých úhlů. Zároveň už byly zpracovány a nebo byly blízko zpracovaného vertexu. V blízkém okolí vertexů na okraji promítnutí modelu je mnohem více vertexů než uprostřed modelu. To je dáno tím, že okraje modelu jsou plochy, které jsou blíže rovnoběžnosti s osou promítnutí, než kolmosti a u středových ploch to je naopak. Tento problém se metoda snaží omezit zavedením použitých vertexů, které obsahují nejen vertexy, které byly zpracovány, ale i vertexy z jejich blízké oblasti. Velikost blízké oblasti je určena parametrem od uživatele jako část r (sousedkého okolí).

vertexů a hlavně mez ignorovaných vertexů bylo nutné zavést z důvodu častého cyklení v některých částech modelu, jelikož při promítnutí ze 3D do 2D se mnoho vertexů promítne velmi blízko sobě, a to hlavně na krajích, kde se hledá silueta, v dalších cyklech se při maximalizaci úhlu vrací zpět, vedle předchozího bodu, kde mnohdy úhel bezproblému překračuje 300 stupňů, a tudíž je maximálním úhlem. Obrázek 4.5 znázorňuje problémy při maximalizaci úhlu.

Vybrání nového vertexu funguje prohledáním seznamu kandidátních vertexů a maximalizováním úhlu mezi kandidátním, současným a předchozím vertexem. Tento výpočet nejdříve vypočítá vektor mezi současným a kandidátním vertexem, vektor předchozího a současného vertexu je dostupný z minulé iterace. Na počátku je předem dán jako vektor (0,1), tudíž směřující nahoru, (0,1) byl vybrán, protože ostatní vertexy mohou mít maximání výšku stejně velkou jako počáteční vertex. Výsledný úhel je dán odečtem úhlů získaných funkcí `atan2()`, která získá úhel od kladné osy x. Výsledný úhel je tedy v rozmezí $-\pi$ až $+\pi$, pro maximalizaci je potřeba přesunout zápornou část do kladné a to příčtem 2π pokud je výsledek záporný, tím získáme hodnoty nad π. Po vybrání nového vertexu se ze současného stává předchozí a z nového současný, zároveň se vektor mezi předchozím a současným nahradí za invertovaný vektor, kterému se

podařilo maximalizovat úhel, ten je potřeba invertovat z důvodu správného výpočtu úhlů. Před další iterací se přidá nový vertex do siluety pro budoucí zpracování.



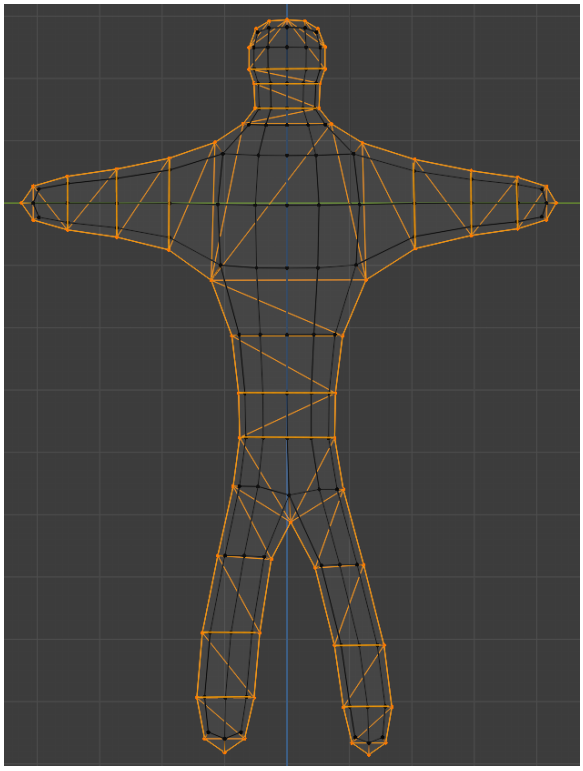
Obrázek 4.6: **Získaná silueta modelu**. Oranžově jsou označeny vertexy a hrany získané siluety modelu. Metoda nalezla správné vertexy a hrany, vytvořila uzavřenou siluetu, která je potřeba pro správné analyzování modelu.

5. Po skončení prvního (globálního) prohledání můžou vzniknout mezery, které je potřeba spojit. K tomu slouží prohledávání na lokální úrovni, kde je cílem propojit siluetu. Zároveň projde vertexy siluety a získá hrany siluety, které jsou potřebné k omezení Delauného triangulace. Prochází již nalezené vertexy siluety a prohledává jejich sousedy, to jsou vertexy sdílející hranu. Pokud v sousedících vertexech najde vertex následující v siluetě, tak je část již propojena a jen uloží hranu do segmentů siluety. Pokud ovšem vertex následující současný vertex v siluetě není v sousedních vertexech, musí najít cestu k následovníkovi. Samotné hledání cesty lze shrnout jako nalezení nejkratší cesty.

   Používá podobnou techniku jako globální prohledávání, tou je prohledávání sousedních vertexů a nalezení maximálních úhlů. Kde vychází ze současného vertexu siluety a cílovým vertexem se stává následovník v siluetě. Navíc neprohledává vertexy v okolí `r`, ale pouze ty, se kterými sdílí hranu. Z důvodu náchylnosti k zacyklení, používá seznam již použitých vertexů k eliminaci zacyklení. Odstraněním již použitých vertexů ze seznamu sousedů, problém je pak nedokončená silueta, ze které pak nejde vytvořit kostra. Proto je vytvořena druhá verze metody, která řeší lokální hledání cesty v siluetě pomocí algoritmu A* [7], právě A* hledá nejkratší cestu podle heuristiky vzdálenosti k cíli. Práce implemetuje modifikaci této metody, aby našla cestu,

nevýhodou je kvalita výsledku, která nemusí být nejlepší, protože nemusí jít po okraji modelu, tedy chtěné siluetě. Výsledek nalezené siluety v modelu je vidět na obrázku 4.6.

6. Na získanou siluetu, se vykoná omezená Delauného triangulace, tu zprostředkovává knihovna triangle. Pro správnost triangulace je potřeba předat seznam vertexů siluety a seznam hran siluety, který bude uzavřený. Před samotným triangulováním musí být přeloženy indexy vertexů v seznamu hran siluety, překlad je implementován pomocí Python slovníků, a to i pro zpětný překlad. Indexy jsou přeloženy z indexu v modelu jako celku do indexu v seznamu vertexů siluety. Výsledek triangulace je na obrázku 4.7

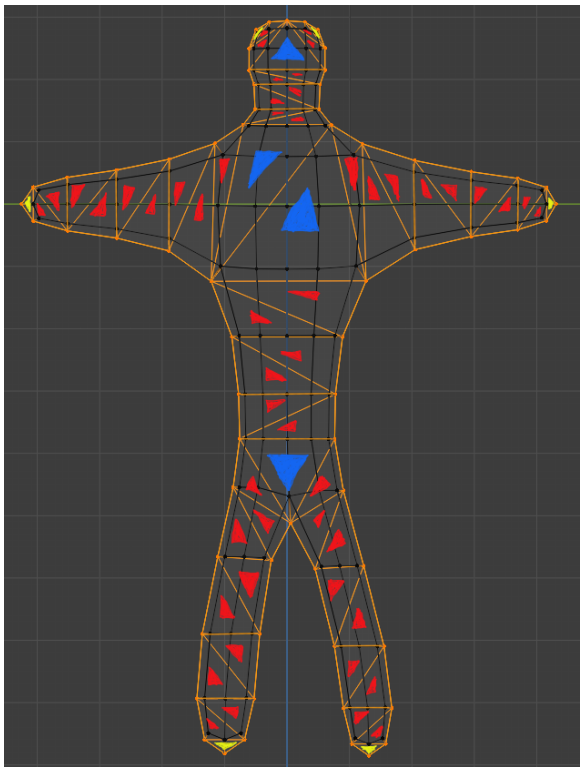


Obrázek 4.7: **Výsledek omezené Delauného triangulace**. Oranžově jsou označeny vertexy a hrany výsledné triangulace.

7. Výsledek omezené Delauného triangulace, přesněji její trojuhelníky musí roztřídit do kategorií podle hran, aby se rozpoznaly důležité části, jako je větvení kostry nebo koncové části kostí. Jelikož jsou trojuhelníky definovány seznamem indexů vertexů, je potřeba trojuhelník nejprve transformovat jako seznam hran, pro jednodužší porovnání. Před porovnáním je nutné přeložit přes slovníky indexy vertexů v hranách trojuheníku zpět na originální indexy vertexů v modelu jako celku, aby vůbec mohlo dojít k porovnání, jelikož silueta obsahuje hrany s originálními indexy.

   Rozdělení trojuhelníků se dělí na kategorie podle počtu vnějších hran, to jsou ty hrany, které jsou v seznamu siluety a porovnává se zda hrana (dvojice indexů) trojúhelníku obsahuje oba indexy v hraně siluety a to v jakémkoliv pořadí. Podle počtu vnějších hran se dělí trojúhelníky na větvení, to jsou trojúhelníky bez vnější hrany, spojují více

větví kostry. Trojúhelníky s jednou vnější hranou jsou časti větve, je z nich složena vetev a pokud má trojúhelník dvě vnější hrany, jedná se o koncový prvek větve, větev kostry v něm končí. Pokud by měl trojúhelník všechny tři strany vnější, znamenalo by to, že trojuhelník je celý model a v tom případě se ignoruje a nic se nevytvoří. Rozdělení trojúhelníků je k vidění na obrázku 4.8.



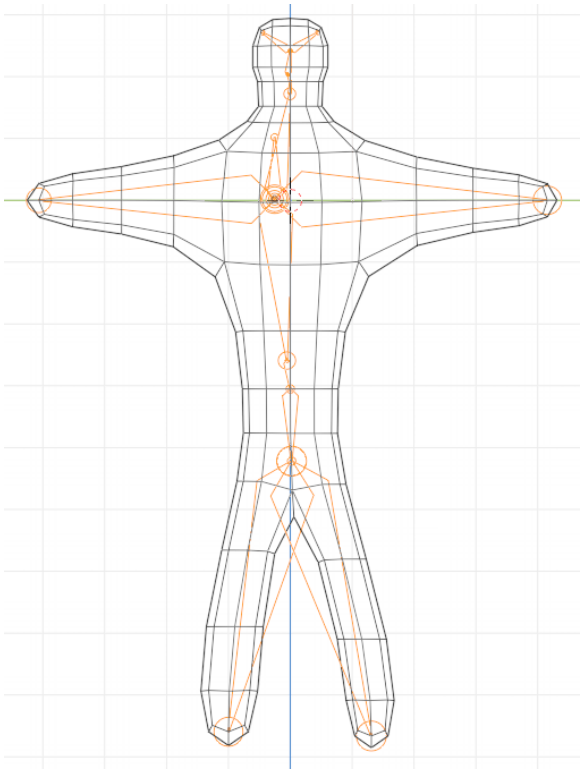
Obrázek 4.8: **Rozdělení trojúhelníků do kategorií**. Trojúhelníky triangulace se dělí na tři kategorie podle počtu vnějších hran. Trojúhelníky se dvěmi vnějšími hranami jsou označeny žlutě a představují koncové trojúhelníky. Trojúhelníky s jednou vnější hranou jsou označeny červeně, představují části větve kostry, kdy větev jimi prochází. Trojúhelníky bez vnější větve jsou označeny modře, představují větvení, spojují více větví dohromady.

8. Všechny trojuhelníky vypočítají středy vnitřních hran, tedy těch hran, které nejsou v siluetě a uloží je do seznamu podle kategorie ve formátu listu středových bodů hran. Pro příklad, trojúhelníku větvení se vypočítají středy všech tří hran `a,b,c` a uloží se jako `bones[0].append([a,b,c])`, kde `bones` je seznam kategorií částí kostí a index 0 je právě seznam trojúhelníků s nulou vnějších hran.

9. Po rozdělení je určené, které trojuhelníky jsou konce, větvení a části větve kosti. Tyto části se nyní spojí do jednotlivých větví, vytvoří list větví. Projde všechny trojuhelníky větvení a jejich středové body hran určují záčatek nebo konec větve. Pokud bude brát středový bod hrany větvícího trojúhelníku jako počáteční bod větve a následně bude hledat v seznamu částí kostí navazující segmenty kosti na počáteční bod, vytvoří větev kostry s počátečním bodem, koncovým bodem a seznamem jednotlivých segmentů kostry jdoucích postupně zasebou.

10. Když jsou předpřipravené větve kostry a určené větvící trojůhelníky, může započít proces vytváření výsledné kostry. Všechny větvící trojúhelníky musí získat středový bod trojúhelníku, pomocí kterého bude moci propojit větve. Před samotným vytvořením kostry je potřeba najít větvící trojúhelníky, které spolu sousedí a spojit je dohromady. Spojení probíhá prohledáním všech větvících trojúhelníků, pokud mají některé dva větvící trojúhelníky stejnou souřadnici středového bodu hrany, tak jsou sousedé a oba změní svůj středový bod trojúhelníku na středový bod sdílené hrany. Následně se upraví začátky a konce větví, které vychází nebo končí ve větvícím trojúhelníku, na středový bod větvícího trojúhelníku.

11. V této fázi je už vše předpřipraveno pro samotné vytvoření kostry. Rovnou se vytvoří počáteční (root) kost ve středovém bodě prvního větvícího trojúhelníku a udělá se kopie seznamu větví, který se bude postupně procházet a budou se z něj mazat zpracované větve, aby nedošlo k několika vytvořením kostí z jedné větve a to i v opačném směru. Algoritmus prochází od počátečního bodu (počáteční kosti) postupně větve kvůli správnému řešení závislostí kostí, tedy nastavování referencí na rodičovské kosti.

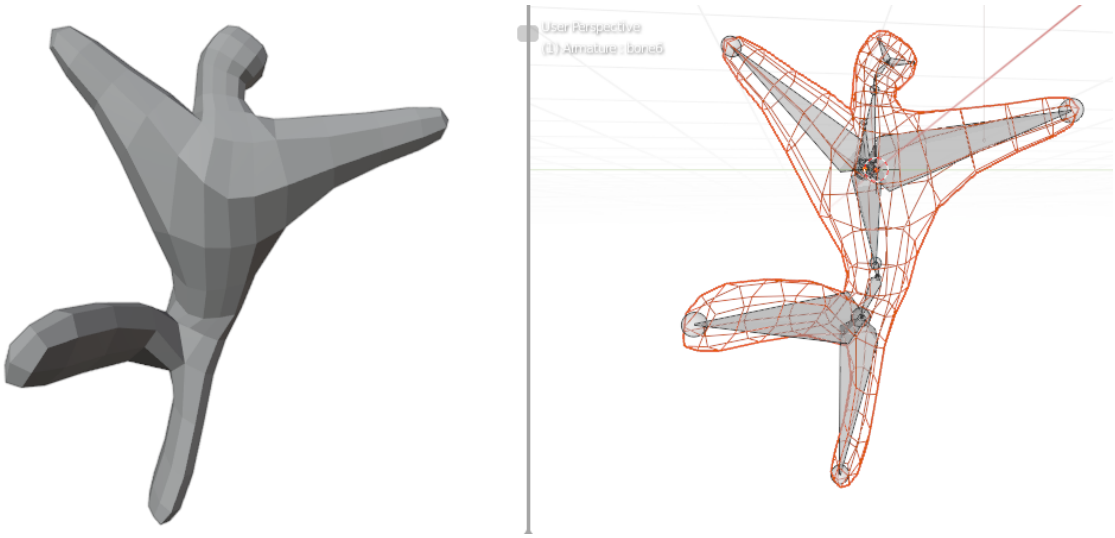


Obrázek 4.9: **Výsledná kostra**. Oranžově je znázorněna výsledná kostra. Části větví jsou redukovány do větších celků porovnáním úhlů.

Nejdříve získá větve, které začínají od počátečního bodu (počáteční kosti) a postupně prochází segmenty větve, mezi kterýma vypočítává úhel, pokud se úhel liší o více než je zadaná hodnota, tak vytvoří novou kost a pokračuje prohledávat segment. Pokud prohledávaní segmentů větve skončí v koncovém trojúhelníku, prohledávání končí. Pokud ovšem narazí na větvící trojúhelník, předá trojúhelníku referenci kosti, aby kosti začínající v nalezeném větvícím trojúhelníku, měli referenci na rodičovskou kost. Dále přidá do seznamu prohledávaných větví větve, které jsou ještě neprohledané

a vychází z nově nalezeného větvícího trojúhelníku. Finální kostra je vidět na obrázku 4.9. Finální kostra se nenaváže na model (skinning) hned, nejdříve se nechá uživateli provést finální úpravy, až bude uživatel spokojen s výsledkem kostry, zmáčknutím tlačítka v panelu rozšíření se provede skinning.

12. Skinning implementuje pomocí rozšíření panelu uživatelského rozhraní o operátor `Skinning`, který spouští skinning operátor vestavěný v Blenderu, který využívá automatické nastavování vah podle Heat algoritmu, podobně jako v návrhu metody. Na obrázku 4.10 je ukázán funkční výsledek.



Obrázek 4.10: **Výsledný narigovaný model**. Model je propojen s kostrou a s její pomocí jde s modelem jednoduše pracovat.

4.4 Publikace

Výsledné rozšíření Rigger je publikované na Gitlab[1], kde jsou popsány úkony pro instalaci a způsoby pro další příspěvky do rozšíření. Gitlab obsahuje zdrojové kódy, které jsou pod licencí GPL.

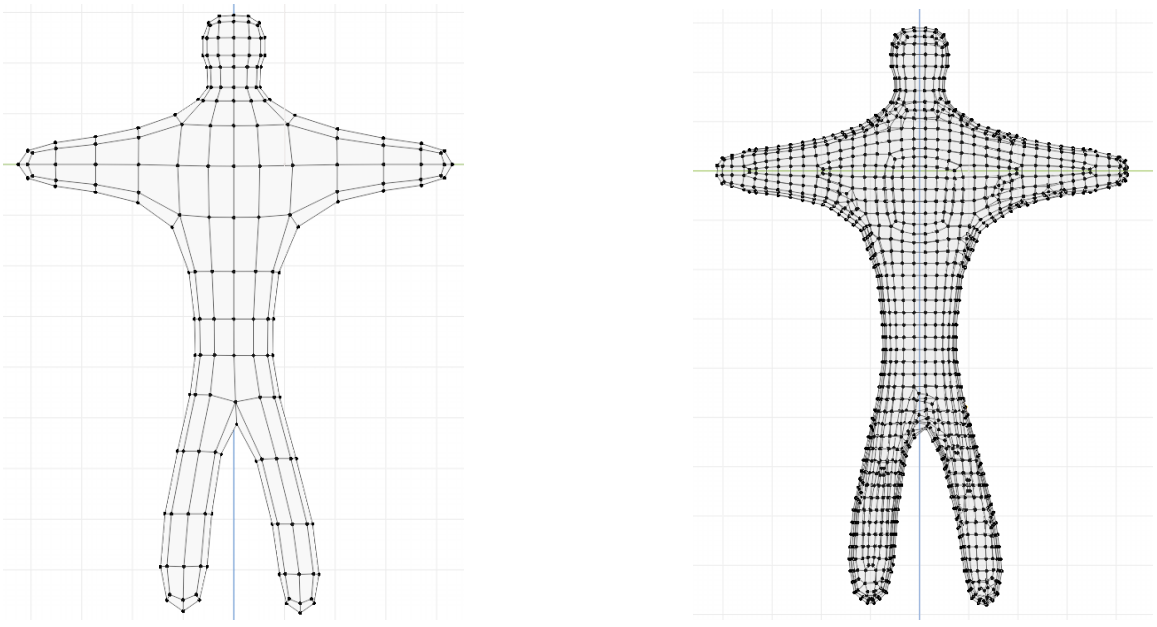
[1]Blender rozšíření Rigger dostupné na: https://gitlab.com/TechMarine/rigger-blender-addon

Kapitola 5

Testování

Testování klade důraz hlavně na časovou náročnost a funkcionalitu. Největší problémovou částí, která způsobuje nefunkčnost, je průchod hledání siluety. Problém nastává buď nenalezením počátečního vertexu, tudíž neukončení hledání siluety a pokračování v další iteraci, a tím získání chybné siluety. Druhá možnost je zacyklení, kdy se při hledání siluety vrací na předchozí již zpracovaný vertex nebo na ještě nezpracovaný vertex blízko předchozímu zpracovanému vertexu. Zacyklení je k vidění na obrázku 4.5.

Pokud by metoda získala pokaždé správnou siluetu, zbytek metody funguje dobře a relativně rychle s dobrými výsledky. Snahou omezit problémy při hledání siluety došlo ke zpomalení metody. V některých případech může pomoci použití nástroje remesh pro uniformní rozmístění vertexů modelu, které může pomoci metodě najít 3D siluetu 5.1.



Obrázek 5.1: **Remesh testovacího modelu**. Obrázek ukazuje výsledek po úpravě testovacího modelu nástrojem remesh s velikostí voxelu 0,25 metru. Model je negativně pozměněn v místech, kde byly ostré úhly, se oblast zaoblila, přitom vznikly anomálie, jako jsou mezi nohama testovacího modelu. Originální model je vlevo, upravený model pomocí nástroje remesh je vpravo. V určitých případech může remesh pomoci ke správnému nálezu siluety.

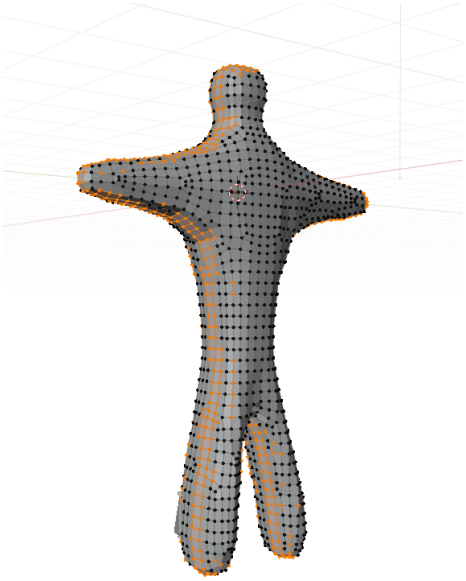
5.1 Funkcionalita

Instalace rozšíření Rigger byla testována na operačním systému Linux a Windows 10, s verzí Blenderu 2.92 a 2.91. Pro instalaci byla použita standardní Blender utilita pro zprávu rozšíření, přes tlačítko install a vybrání souboru Rigger.zip. Po instalaci je nutné rozšíření Rigger aktivovat, tím se importují soubory a registrují se potřebné třídy. Problém může nastat v některých případech s importem knihovny triangle, kterou se rozšíření snaží nainstalovat pomocí pip. Instalace knihovny pomocí pip může požadovat potřebné uživatelské oprávnění.

V případě přidání nové rigovací metody, se musí správně pojmenovaný a implementovaný soubor s rigovací metodou úmístit do složky methods, která se nachází ve složce rozšíření, nazvaná Rigger. Na testovaném operačním systému linux se složky s rozšířeními nachází v `~/.config/blender/scripts/`. Pro importování rigovací metody je potřeba Blender restartovat, pokud není otevřený stačí znovu spustit.

Testovací model, který byl použit k demonstraci implementace je možné narigovat pomocí výchozího nastvení parametrů bezproblému, výsledek na obrázku 4.10.

Drobné problémy, hlavně při vytváření siluety lze vyřešit pomocí změny parametrů, jako je násobič velikosti sousedského okolí a násobek sousedské oblasti, který má za účel vyřadit vertexy v blízkém okolí. U spousty případů se stále nepodaří ani změnou těchto hodnot získat použitelný výsledek.



Obrázek 5.2: **Špatně získaná silueta modelu**. Oranžově jsou zvýrazněny vertexy siluety. Z obrázku jde vidět problém nalezení počátečního vertexu, a tím vykonání několika cyklů kolem celého modelu. Protože metoda nemůže znova procházet již použité vertexy, prochází po těch, které jsou vedle již použitých vertexů. Taková silueta je k analýze modelu nepoužitelná, nevznikne použitelná kostra.

Pokud testovací model změníme na větší počet uniformně rozmístěných vertexů, pomocí Blender nástroje remesh, nastaveného na velikost voxelů 0,25 metru, nastane problém při vytváření kostry v kroku získání siluety. Chybné získání siluety je k vidění na obrázku 5.2.

Remesh může v některých případech pomoci rozprostřít uniformně vertexy, a tím zajistit lepší průběh hledání siluety. Jelikož ale nástroj remesh mění model, může vytvořit horší model, a tím zvětšit neúspěšnost získání správné siluety.

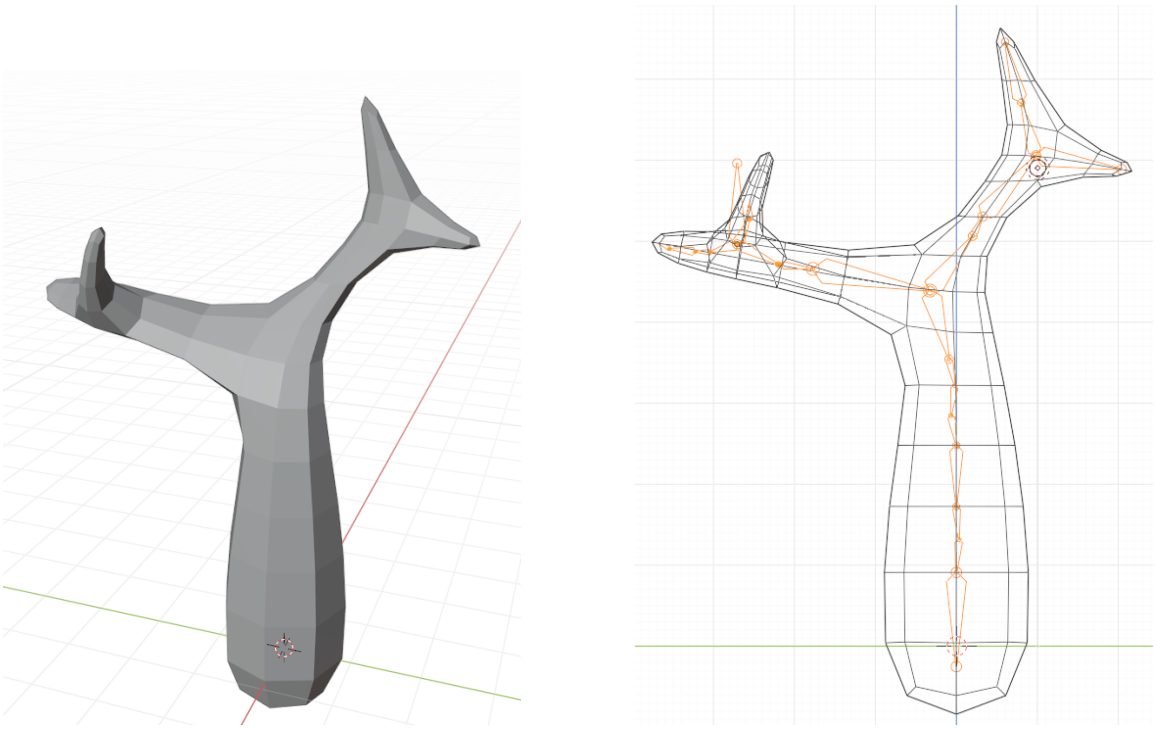
5.2 Časová náročnost

Výpočetní doba byla testována na procesoru AMD Ryzen 5600x.

Model, který byl použit k demonstaci implementace, obsahuje 290 vertexů, 576 hran. Rigovací metoda se v tomto případě provede za 0,08468 vteřiny.

Model stromu, k vidění na obrázku 5.3, obsahuje 218 vertexů a 432 hran, doba vytvoření kostry je 0,049519 vteřiny.

Testovací model panáčka, po změně nástrojem remesh na velikost voxelů 0,25 metru, obsahuje 2147 vertexů a 4290 hran, rigovací algoritmus se provede za 11,505365 vteřiny. Problémem je zacyklení, několikrát nachází siluetu. Končí s neprávným výsledkem, kvůli špatně získané siluetě 5.2.



Obrázek 5.3: **Testovací model strom**. Obrázek ukazuje model stromu a jeho výslednou kostru, která je správně vygenerovaná.

Časová náročnost roste exponeniálně s počtem vertexů a hran, důvodem je zvětšování seznamů a pak jejich porovnávání. Příkladem těchto seznamů je seznam použitých vertexů, který se průběhem algoritmu plní. Algoritmus zpomaluje s každou následující iterací.

Metoda je použitelná na jednodužší modely, trpí zastiňováním, některé části nemusí analyzovat, protože jsou zastíněné promítnutím do 2D. U velkých, složitých modelů bude výpočet trvat dlouhou dobu a jeho výsledek nemusí být použitelný. Pro jednoduché modely slouží rychle s poměrně dobrým výsledkem rigu.

Kapitola 6

Závěr

Cílem práce bylo vytvoření rozšíření pro Blender, které vykonává rigging, tedy vytvoření a navazání kostry na 3D model[1]. Zároveň je dále jednoduše rozšiřitelné pomocí dynamického načítaní rigovacích metod. Pro tvorbu nových rigovacích metod je předpřipravená šablona, která stačí vyplnit a vložit do příslušné složky s rigovacími metodami.

Nejprve byly nastudovány, různé druhy metod řešící automatické generování kostí. Z metod byla vybrána rigovací metoda používající 3D siluetu, jelikož měla malou výpočetní náročost a relativně přijatelnou kvalitu výsledku. Zároveň bylo důležité nastudovat prácí s Blenderem a Blender Python API, které je stěžejní pro tvorbu rozšíření. Důležitou částí rozšíření je uživatelské rozhraní, které je vytvořeno pro jednoduchou práci s rozšířením.

Implementovaná metoda používající 3D siluetu je použitelná na jednodužší 3D modely. Pro jednoduché modely metoda funguje poměrně úspěšně a rychle. U složitějších modelů má problémy detekovat siluetu modelu. Při řešení vylepšení robustnosti této metody přišla o část rychlosti, hlavně u větších modelů. Značné vylepšení by byla výměna algoritmu hledání siluety podle maximalizace úhlů za robustnější metodu. Robustnější metoda by mohla využít skutečnosti, že v modelu, který má rovnoměrně rozmístěné vertexy, po promítnutí do 2D jsou vertexy umístěny hustěji ke kraji modelu.

Naučil jsem se pokročilejší fungování programu Blender a dozvěděl nové přístupy řešení problémů, hlavně z oblasti výpočetní grafiky, o kterých jsem dříve mnoho nevěděl.

Do budoucna bych rozšířil Rigger o další rigovací metodu, která by byla lepší a hlavně robustnější, než implementovaná metoda používající 3D siluetu. Také bych se snažil najít náhradu za současnou maximalizaci úhlů u metody získání 3D siluety, jelikož zbytek metody funguje dobře a rychle. S poměrně kvalitním základem rozšíření bude vytváření dalších rigovacích metod jednodužší. Zkusil bych experimentovat s různými typy metod. Metoda používající 3D siluetu vhodná k rigování jednoduchých 3D modelů nebo jejich prototypování.

[1] Blender rozšíření Rigger dostupné na: https://gitlab.com/TechMarine/rigger-blender-addon

Literatura

[1] Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen Or, D. a Lee, T.-Y. Skeleton Extraction by Mesh Contraction. *ACM Transactions on Graphics*. 2008, sv. 27, č. 3.

[2] Veber, L. *Auto-Rig Pro Documentation* [online]. 2021 [cit. 2021-01-17]. Dostupné z: http://www.lucky3d.fr/auto-rig-pro/doc/auto_rig.html.

[3] Bitter, I., Kaufman, A. E. a Sato, M. Penalized-Distance Volumetric Skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*. 2001, sv. 7, č. 3.

[4] Blender Foundation. *Docs.blender.org/api/* [online]. 2021 [cit. 2021-01-01]. Dostupné z: https://docs.blender.org/api/current/.

[5] Cornea, N. D., Silver, D. a Min, P. Curve-Skeleton Properties, Applications and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*. 2007, sv. 13, č. 3, s. 530–548.

[6] Cornea, N. D., Silver, D., Yuan, X. a Balasubramanian, R. Computing Hierarchical Curve-Skeletons of 3D Objects. *The Visual Computer*. 2005, sv. 21, č. 11, s. 945–955.

[7] Cui, X. a Shi, H. A*-based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security*. 2011, sv. 11, č. 1.

[8] Jacobson, A., Deng, Z., Kavan, L. a Lewis, J. Skinning: Real-time Shape Deformation. In: *ACM SIGGRAPH 2014 Courses*. 2014.

[9] Jarvis, R. A. On the identification of the Convex Hull of afinite set of points in the plane. *Information Processing Letters*. 1973, sv. 2, s. 18–21.

[10] Kallmann, M., Bieri, H. a Thalmann, D. Fullydynamic constrained delaunay triangulations. *InGeomet-ric Modelling for Scientific Visualization. Springer-Verlag*. 2003, s. 241–257.

[11] Pan, J., Yang, X., Xie, X., Willis, P. a Zhang, J. J. Curve-Skeleton Properties, Applications and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*. 2007, sv. 13, č. 3, s. 530–548.

[12] Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: Lin, M. C. a Manocha, D., ed. *Applied Computational Geometry: Towards Geometric Engineering*. Springer-Verlag, Květen 1996, sv. 1148, s. 203–222. Lecture Notes in Computer Science. From the First ACM Workshop on Applied Computational Geometry.

[13] Tierny, J., Vandeborre, J.-P. a Daoudi, M. Fast and precise kinematic skeleton extraction of 3D dynamic meshes. *19th International Conference on Pattern Recognition (ICPR 2008)*. 2008.