FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Věnná města českých královen - Validace kvality meshe 3D modelů
Student:	Viktor Káčer
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Věnná města Českých Královen je projekt zabývající se zobrazením historického prostředí ve virtuální realitě.

1. Proveďte rešerši aktuálních 3D modelovacích technik a obvyklých chyb meshe 3D modelu vzniklých při modelování a přenosu mezi formáty.
2. Analyzujte algoritmy validace a případné opravy meshe 3D modelů.
3. Pomocí metod softwarových inženýrství navrhěte zásuvný modul kontrolujíci kvalitu meshe 3D modelů.
4. Implementujte prototyp zásuvného modulu v programu Blender.
5. Zásuvný modul podrobte uživatelským testům a testům na modelech z aplikace Věnná města českých královen.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 2. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Věnná města českých královen - validace kvality meshe 3D modelů

Viktor Káčer

Katedra softwarového inženýrství
Vedúci práce: Ing. Jiří Chludil

13. mája 2021

Poďakovanie

Rád by som poďakoval vedúcemu tejto bakalárskej práce, pánovi Ing. Jiřímu Chludilovi, za ochotu, odborné rady a čas strávený pri tvorbe tejto práce. Taktiež by som chcel poďakovať svojej rodine, priateľke a kamarátom za podporu počas písania práce, ale aj celého štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 13. mája 2021 . . . . . .

Odkaz na túto prácu

Káčer, Viktor. *Věnná města českých královen - validace kvality meshe 3D modelů*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Táto práca sa zaoberá základmi 3D modelov a ich elementmi, ako sú vrcholy, hrany a plochy, ktorých vzájomné usporiadanie sa nazýva geometria. Vysvetlené sú spôsoby, ako je možné geometriu modelov reprezentovať a v akých formátoch je možné ju ukladať. Obsahom práce sú taktiež spôsoby, ako modely správne vytvárať a definícia celej modelovacej pipeline. Pre správnu tvorbu modelov sú vysvetlené základné modelovacie techniky ako sculpting, box modeling a photogrammetry. Kontrastne sa práca venuje aj najčastejším chybám, ktoré v geometrii vznikajú. Ide o chyby ako otočené normály, diery a dvojitá alebo non-manifold geometria. Pozornosť je venovaná aj dôvodom, prečo dané chyby vznikajú a ako je možné ich manuálne opraviť. Práca sa venuje automatickým opravám chýb za pomoci algoritmov, ktoré sú rozdelené na lokálne a globálne podľa ich pôsobenia. Dané algoritmy sú stručne vysvetlené a vzájomne porovnané, sú určené ich výhody, nevýhody a možné požiadavky na vstup. Konkrétne ide o algoritmy, ktoré slúžia na vypĺňanie dier, odstránienie šumu a iných degenerácií v geometrii. Získané znalosti najčastejších problémov budú použité pri návrhu prototypu zásuvného modulu do programu Blender, ktorého implementácia je napísaná v programovacom jazyku Python. Modul je schopný identifikovať a opravovať otočené normály, dvojitú geometriu alebo diery. Hlavným výstupom práce je časť modulu opravujúca budovy z projektu VMCK, ktorý bol hlavnou inšpiráciou tvorby tejto práce.

Klíčová slova oprava meshu, validácia meshu, časté problémy s meshom, elementy meshu, základy 3D modelovania, zásuvný modul do programu Blender

Abstract

This work deals with the basics of 3D models and their elements, such as vertices, edges and surfaces, the mutual arrangement of which is called geometry. The ways in which the geometry of the models can be represented and in what formats it can be saved are explained. The work also includes ways to create models correctly and the whole modeling pipeline is defined. Basic modeling techniques such as sculpting, box modeling and photogrammetry are explained for the correct creation of models. In contrast, the work also deals with the most common errors that occur in geometry. These will be errors such as flipped normals, holes, double or non-manifold geometry. Attention is also paid to the reason why the errors occur and how they can be corrected manually. The work deals with automatic error correction using algorithms, which are divided into local and global according to their operation. The given algorithms are briefly explained and compared with each other, their advantages, disadvantages and possible input requirements are determined. Specifically, these are algorithms that are used to fill holes, remove noise and other degenerations in geometry. The acquired knowledge of the most common problems will be used in the design of a prototype plug-in for Blender, the implementation of which is written in the Python programming language. The module is able to identify and correct flipped normals, double geometry or holes. The main output of the work is a part of the module repairing buildings from the VMCK project, which was the main inspiration for the creation of this work.

Keywords mesh repair, mesh validation, common mesh problems, elements of mesh, 3D modeling basics, Blender addon

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V súčasnosti sa 3D modely používajú takmer v každom priemysle, počnúc architektúrou, lekárstvom, simuláciami, virtuálnou realitou až po zábavný priemysel. Na ich výrobu sa používajú rôzne techniky, spôsoby, štýly a programy. Táto rôznorodosť, ale spôsobuje problémy.

Chyby spôsobené 3D modelovaním môžu byť malé a nenápadné ako napr. zlé vykresľovanie plôch či chýbajúca geometria v počítačových hrách, ale aj závažné, ako napríklad diera v modeli určenom pre simuláciu aerodynamiky lietadiel. Dané problémy potrebujeme detekovať a následne opraviť, či už manuálne alebo za pomoci algoritmu na opravu chýb.

Tento text vznikol ako úvod do 3D sveta, kde si v prvej kapitole predstavíme históriu 3D počítačovej grafiky a jej možnej budúcnosti. Povieme si, akými spôsobmi definujeme 3D modely a predvedieme si základné elementy, z ktorých je bežný model zostavený. Ďalej sa budeme venovať rôznym dátovým štruktúram a najčastejším 3D formátom, v ktorých sa ukladá geometria a už spomínané elementy. Ako poslednú vec kapitoly si uvedieme základy modelovania, dôvody, prečo modelovať pre marketing, VR, hry a iné druhy. Voľne si zadefinujeme štýly *realizmus* a *štylizácia*, predstavíme si modelovacie techniky ako box modeling, sculpting a photogrammetry a zostavíme si modelovaciu pipeline používanú v zábavnom priemysle.

Druhá kapitola je venovaná chybám v geometrii. Spomenieme si často používané programy na tvorbu modelov a popíšeme najčastejšie chyby, ktoré vznikajú pri používaní programu Blender. Taktiež si vysvetlíme iné časté problémy ako zlý manifold, n-uholníky a póly z vizuálneho hľadiska.

V predposlednej, tretej kapitole sa pozrieme na korektnú analýzu algoritmov na opravu a validáciu meshov 3D modelov. Formálne rozdelíme problémy na lokálne, globálne a geometrické. Definujeme rozdiely medzi *upstream* a *downstream* aplikáciami a pozrieme sa na algoritmy pomocou lokálneho alebo globálneho prístupu. Pôjde konkrétne o algoritmy opravujúce manifold, medzery, diery, odstraňovanie degenerácií, prienikov, šumu a iné.

V poslednej praktickej kapitole zadefinujeme funkčné a nefunkčné požiadavky, z ktorých vytvoríme návrh prototypu zásuvného modulu do programu Blender, ktorý implementujeme v jazyku Python. Z implementačného hľadiska sa pozrieme na súborovú štruktúru nášho modulu, aký je životný cyklus modulov v Blenderi a aké dátové štruktúry a triedy sme použili. V ďalších častiach vytvoríme inštalačné, používateľské a programátorské príručky a náš výsledný zásuvný modul otestujeme.

Hlavnou motiváciou tejto práce je napraviť zlý stav modelov budov z projektu *Věnná Města Českých Královen* (ďalej už len VMCK). Ich zlý stav je zapríčinený prevodmi medzi rôznymi formátmi a aplikáciami, ktoré pozmenili ich vnútornú štruktúru.

Kapitola 1

Cieľ práce

Cieľom našej práce je zoznámiť čitateľa so základmi 3D modelov, ich elementmi, modelovacími technikami a problémami, ktoré vznikajú počas tvorby geometrie 3D modelu. Preberieme dátové štruktúry a najčastejšie 3D formáty, ktoré majú za úlohu uložiť rôzne elementy modelu. Druhým cieľom tejto práce je vytvoriť porovnanie algoritmov a metód, ktoré opravujú problémy v geometrii, ako napríklad diery či otočené normály.

Posledným cieľom práce je implementácia zásuvného modulu do programu Blender, ktorý bude chyby hľadať a opravovať. Výsledný modul bude schopný opraviť výzor a geometriu budov, ktoré sú použité v projekte *Věnná města českých královen*, tie sú hlavnou motiváciou vzniku tejto práce.

Kapitola 2

3D Model a Základy Modelovania

V tejto kapitole si predstavíme, čo je to 3D model, prejdeme si rýchlu históriu 3D sveta a pozrieme sa, z čoho sa 3D model môže skladať. Taktiež si povieme, k čomu je užitočné modelovanie, aké štýly a techniky modelovania poznáme a pozrieme sa, ako vyzerá základná modelovacia pipeline v zábavnom priemysle.

2.1 3D Model

2.1.1 Definícia a motivácia

3D model, je troj-dimenzionálna reprezentácia objektu, vytvorená za pomoci programu, špecializovaného na tvorbu modelov.

Je ale 3D modelovanie podstatné? Bude mať význam aj v budúcnosti? Odpoveď je áno. Od vzniku 3D modelov sa ich významnosť dostáva vždy do viac a viac priemyslov. O aké priemysly sa tu jedna?

- Architektúra
- Dizajn
- Filmy
- Hry
- Simulácia
- Skeny
- Zdravotníctvo

Samozrejme, každý z týchto priemyslov požaduje iné techniky tvorby 3D modelu, o ktorých si povieme v nasledujúcich sekciách. Môžeme však s istotou povedať, že základné znalosti 3D modelovania budú v budúcnosti užitočné. Techniky sa možno zmenia, ale základy ostanú.

2.1.2 História

V tejto časti práce si povieme základnú históriu 3D modelovania a celkového 3D sveta, ako sa vyvíjali a ktorým smerom sa môžu uberať. Informácie sú čerpané z [1, 2, 3].

2.1.2.1 Pre-história

Prvé základy 3D modelovania položili ľudia dávno pred tým, než boli vynájdené počítače. Začalo to približne v roku 3 pred Kristom Grékom Euclidom, ktorý položil základy geometrickej matematiky.

Neskôr, v 16. storočí, René Descartes vyvinul analytickú geometriu, vďaka ktorej sme mohli presne počítať polohy a ich vzdialenosti.

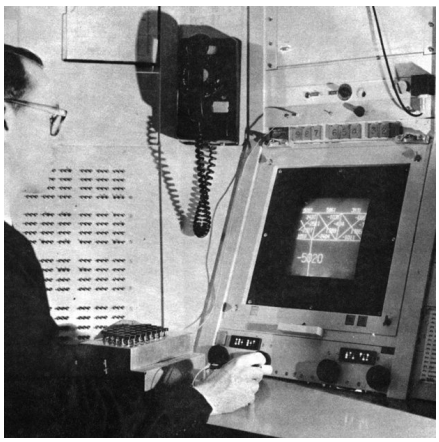
V 18. storočí anglický matematik James Joseph Sylvester vynašiel maticové počítanie, ktoré je teraz základom každého 3D programu.

Na prelome rokov 1950 sa začali používať počítače (SAGE computer system) pre výpočty, tie boli určené primárne pre vojenské účely.

2.1.2.2 60. roky 20. storočia

Počítače sa začali objavovať a s nimi aj prvé metódy vykresľovania priamok a geometrických útvarov. Dnes to berieme ako samozrejmosť, no v 60. rokoch to patrilo medzi hlavné témy počítačovej grafiky. V tomto období sa preslávila práca od Pierre Bézier, ktorý spracovával parametrické krivky a povrchy. IBM v tomto období vyvíjala prvé algoritmy, ktoré určovali, čo sa bude vyzobrazovať. Je to základ terajšieho ray tracingu.

Zároveň sa v roku 1963 objavil prvý CAD program, Sketchpad od autora Ivan Sutherland. Sketchpad, alebo taktiež známy pod menom „Robot Draftsman“, mal ukázať, že počítače je možné používať aj pre dizajn a umelcov, nie len pre matematikov.



Obr. 2.1: Ivan Sutherlands používajúci jeho aplikáciu Sketchpad [4]

2.1.2.3 70. roky 20. storočia

Vznik programu ADAM, rok 1971, šlo o CAD systém navrhnutý tak, aby fungoval, na čo najviac počítačoch. Dôvodom bolo to, že vznikali nové firmy na tvorbu dizajnu. Stále sa ale jednalo o CAD „solid“ modelovanie, ktoré bolo prijaté mnohými firmami, ale mnohé univerzity sa pokúšali prísť na lepší systém zobrazovania a tvorby 3D modelov.

Jednou z týchto univerzít bola University of Utah, kde Gouraud a Phong vynašli nový spôsob zobrazovania 3D objektov. Na danej univerzite vznikol prvý slávny grafický model, išlo o nádobu na čaj, takzvanú „Utah Teapot“, ktorú prvý krát vyzobrazil Martin Newell.

Medzitým vzniklo prvý krát „keyframe“ založené animovanie. Xeroc PARC vynašiel prvý program na kreslenie. Ed Catmull vynašiel Z-Buffer algoritmus a Turned Whitted vynašiel rekurzívny ray tracing, ktorý sa stal neskôr štandardom fotorealizmu. Objavili sa prvé hry ako Pong a Pac-Man.

Obr. 2.2: ADAM terminály nachádzajúce sa v *Lockheed Engineering Department* [5]

2.1.2.4 80. roky 20. storočia

IBM v roku 1981 vydalo svoj prvý počítač, ktorý rozšíril dostupnosť CAD programov pre viac ľudí ako len pre inžinierov a automobilové spoločnosti. Neskôr v roku 1983 vznikol AutoCAD, ktorý ponúkal takmer všetku funkcionalitu predošlých CAD programov, no len za 20% sumy.

V strede desaťročia prišiel Jim Blinn s „blobby“ modelmi a konceptom mapovania textúr, ktorý dovolil architektom vytvárať, pred tým nevídané vizualizácie.

Vznikli BSP stromy, dizajn a modelovanie v CAD programoch sa začalo učiť na mnohých univerzitách. Cieľom sa stala skôr animácia, než len vyzobrazovanie. Pokrokov si všimli herné štúdia a Hollywood. V tomto období vznikol taktiež Adobe Photoshop.

Obr. 2.3: Prvá verzia programu AutoCAD [5]

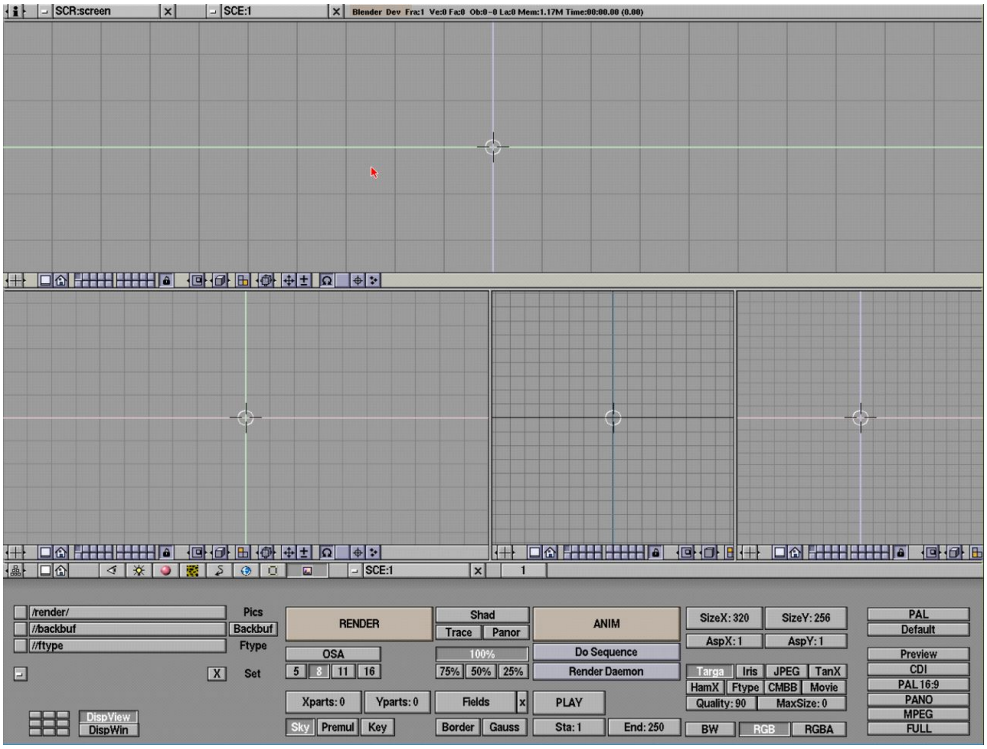
2.1.2.5 90. roky 20. storočia

V roku 1992 sa OpenGL stalo štandardným grafickým API. Vznikali dynamické systémy, ktoré povoľovali animácie s kolíziami, gravitáciu, trenie a prvé systémy na zachytenie pohybu. Vznikali prvé grafické karty od spoločností 3dfx a Nvidia. Neskôr vznikol Unreal Engine a Blender. V zábavnom priemysle vznikla hra Unreal Tournament a z filmov sem patrí Toy Story, Terminátor alebo Jurský Park.

2.1.2.6 Súčasnosť a budúcnosť

V dnešnej dobe je svet bez 3D grafiky nepredstaviteľný. Modely a efekty sa nachádzajú v každej reklame či filme. Počítačové hry sú populárne, ako ešte neboli, a zjednodušené zachytávanie skenov pomocou metód fotogrametrie sa stáva bežnou súčasťou mobilných telefónov.

V budúcnosti sú naplánované herné engines, ktoré dokážu spracovať nekonečné množstvo polygónov, reálne osvetlenie v reálnom čase a VR je na vzostupe.



Obr. 2.4: Prvá verzia programu Blender [6]

2.1.3 Reprezentácia 3D modelu

3D model vieme reprezentovať dvoma hlavnými kategóriami, ide o: [7, 8]

1. **Solid** Modely sú definované objemom objektu, ktorý reprezentujú. Tieto modely sú používané primárne pre vedecké a lekárske simulácie. Zväčša sú zostavené pomocou voxelov alebo technikou nazývanou „Constructive Solid Geometry“.

   a) **Voxelové modely** Voxel môžeme vnímať ako kubickú jednotku, reprezentujúcu 3D modely. Najčastejším prirovnaním býva, že voxel je 3D pixel. Ide, teda, o solídne zloženie modelu. Kvalita sa udáva rozlíšením, ktoré mení veľkosť voxelov.

   b) **Constructive Solid Geometry** Technika modelovania, ktorá využíva booleovské operácie. Za pomoci jednoduchých objektov vytvorí zložitejšie solid modely.

2. **Shell** alebo inak nazývané, ako **Boundary** či **B-Reps** modely, reprezentujú povrch alebo okraje objektu, nie jeho objem. Môžeme si ich predstaviť ako nekonečne tenkú škrupinu od vajíčka. Takmer všetky modely vo filmoch a hrách sú tohto štýlu.

Aby tieto objekty dávali zmysel ako reálne objekty, musia byť manifoldné. Najčastejšou reprezentáciou sú 3D meshe.

3D modely popísané svojím povrchom sú zložené z dvoch častí.

Topológia Organizácia prvkov modelu (vrcholy, hrany a plochy) a spojenia medzi nimi.

Geometria Definícia daných komponentov (body, krivky a povrchy). Plocha je časť povrchu, hrana je kúsok krivky a vertex leží v bode.

2.1.4 Elementy 3D modelu

2.1.4.1 3D Mesh

Známy aj pod názvami polygónový mesh alebo geometria. Je to dátová štruktúra geometrických dát, ktorá povoľuje reprezentáciu povrchu za pomoci polygónov [9]. O aké geometrické dáta ide si uvedieme nasledovne:

Vrchol/Vertex Bod v 3D priestore. Nesie dáta ako lokácia, farba, normály či koordináty textúr.

Hrana/Edge Spojenie dvoch vrcholov. Nesie dáta ako označenie rezov, alebo udáva váhy ovplyvnenia.

Plocha/Face Uzavretá množina hrán, kde trojuholníková plocha má 3 hrany, štvoruholníková plocha má 4 hrany a podobne. Polygón je plocha ležiaca v rovine. Nesie dáta ako normály alebo materiál.

2.1.4.2 Groups

Dávajú spôsoby, ako niesť informácie o skupinách vrcholov, hrán alebo plôch. Najčastejšie ide o takzvané „Vertex Groups“. Slúžia na určovanie podobjektov, označovanie istých atribútov, alebo umožňujú animácie za pomoci kostí, tzv. „Skeletal Animations“.

2.1.4.3 Farba vrcholov

Je súčasťou dát vrcholov, ku ktorým sa pridá RGB hodnota, určujúca jeho farbu. Ide o najzákladnejší spôsob zafarbenia 3D modelov, kde sa farba medzi vrcholmi interpoluje. Nevýhodou je vysoká potreba vrcholov pre detailné zafarbenie.

2.1.4.4 UV mapa

Slúži na reprezentáciu 3D geometrie v 2D priestore. Mesh je potrebné „nastrihať a rozložiť“ v hranách. UV mapa rieši problém farby vrcholov, pretože môžeme namapovať akúkoľvek textúru na UV koordináty. Dané koordináty textúru prenesú na 3D geometriu.

2.1.4.5 Materiál

Udáva, aký shader sa má použiť počas renderovania 3D objektu. Informácie o materiály vedia držať len plochy. Každá plocha môže mať nastavený iný materiál, no čím viac použijeme materiálov, tým je proces renderovania dlhší.

2.1.4.6 Animácie

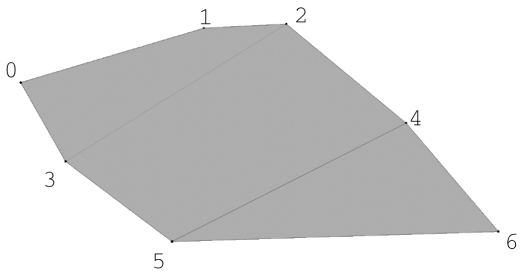
Spôsobujú zmenu polohy vrcholov, čím vzniká pohyb. Niektoré formáty sú schopné ukladať animácie za pomoci „skeletal animations“ spôsobu. Ten používa váhy pôsobenia vrcholov, kde kosti ovplyvňujú viaceré vrcholy. Iné, naopak, pre každý snímok animácie zostavia manuálne množiny všetkých vrcholov s ich polohami. Zmena indexu množiny v čase vytvára pohyb.

2.1.5 Dátové štruktúry

Slúžia na uchovávanie geometrie 3D modelu. Konkrétne ide o usporiadanie vrcholov, hrán, plôch a ich normál. Existujú viaceré dátové štruktúry, podľa toho, čo uprednostňujú. Môže ísť o menšiu veľkosť dát, lepšie vyhľadávanie v okolí alebo rýchlosť vkladania nových vrcholov [10, 11, 12, 13, 14].

Face-Vertex Meshes Jedná sa o jednoduchý list vrcholov a množín plôch. Každá plocha drží ukazovateľ na vrcholy, z ktorých je stvorená, nazývané „indices“. (Množné číslo od slova index.)

Ide o najprimitívnejší spôsob ukladania dát. Typicky ide o dáta akceptovateľné moderným hardware-om. Pracuje sa s plochami, ktoré majú práve 3 vrcholy, čiže všetko je tvorené z trojuholníkov. Veľkou nevýhodou je, že susednosť nie je nijak uložená.



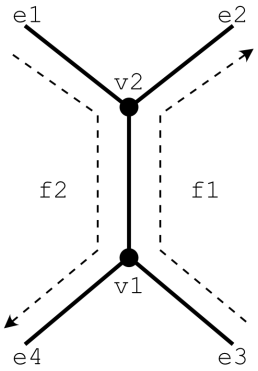

```
Vertices
0: (-0.8;  0.9; -0.1)
1: ( 0.8;  0.9; -0.1)
2: ( 1.0;  0.1;  0.1)
3: (-1.0;  0.0; -0.1)
4: ( 0.9; -1.0; -0.2)
5: (-1.0; -1.0;  0.0)
6: ( 0.0; -1.9;  0.0)

Faces
0: 0, 1, 2, 3
1: 3, 2, 4, 5
2: 5, 4, 6
```

Obr. 2.5: Face-Vertex schéma

Winged-Edge Nazývaná aj ako okrídlená hrana. Je to štruktúra, kde každá hrana ukazuje na dva vrcholy, dve plochy a štyri (v CW alebo CCW poradí) hrany, ktoré sa jej dotýkajú. Každá hrana obsahuje referenciu na jednu okrajovú hranu a pre každý vrchol existuje referencia na jednu hranu, ktorá je s ním spojená.

Táto dátová štruktúra explicitne popisuje geometriu a topológiu plôch, hrán a vrcholov. To umožňuje rýchly prechod po štruktúre a uľahčuje to prácu algoritmu „Subdivision Surface“. Vieme takto pristupovať k dátam v konštantnom čase s miernou pamäťovou záťažou.

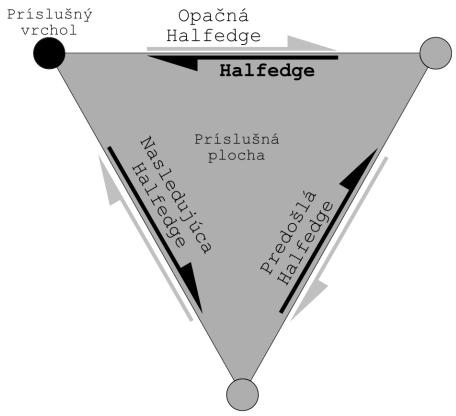
Nevýhodou je jej komplikovaná definícia, a to, že hrana je obojstranná, no v štruktúre je hrana orientovaná, takže pri výbere ukazovateľa na nasledujúcu hranu pri prechode okolo plochy, je potrebné používať „if“ podmienky. Tie spôsobujú spomalenie.



Obr. 2.6: Winged-Edge schéma

Half-Edge Meshes Taktiež aj „Doubly Connected Edge List“ (DCEL) či polovičné hrany, je podobná predošlej Winged-Edge dátovej štruktúre s tým rozdielom, že jednu hranu rozdelí na dve podľa ich smeru, čím sa snaží vyhnúť predošlému problému a snaží sa presne odpovedať na otázky ohľadom susednosti. Podporuje však len manifold objekty.

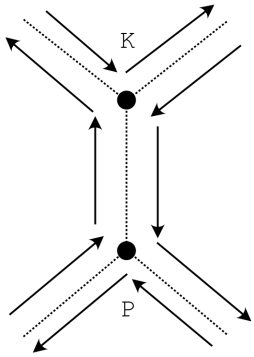
Tieto polovičné hrany tvoria „linked list“ okolo plochy (v CW alebo CCW poradí), ktorú ohraničujú. Každá táto hrana drží ukazovateľ na plochu, ktorú obchádza, na vrchol na jej konci a na opačnú polovičnú hranu. Vrcholy držia ukazovateľ na jednu polovičnú hranu, ktorá ho používa ako počiatočný bod a plocha drží ukazovateľ na jednu polovičnú hranu, ktorá ju definuje.



Obr. 2.7: Half-Edge schéma

Quad-Edge Meshes Ukladá hrany, polovičné-hrany a vrcholy bez referencii na plochy. Tie sú implicitne definované. Využíva pozorovania, podobne ako predošlé štruktúry, že hrana v topológii sa nachádza presne medzi dvoma plochami a vrcholmi. Pamäťové nároky sú vyššie.

Všetky hrany držia referencie na okolité 4 hrany. Každá z týchto referencií odkazuje na ďalšiu hranu (v CCW poradí) okolo vrchol alebo plochy a každá z tých referencií reprezentuje buď počiatočný vrchol hrany, pravú plochu, koncový vrchol alebo ľavú plochu.



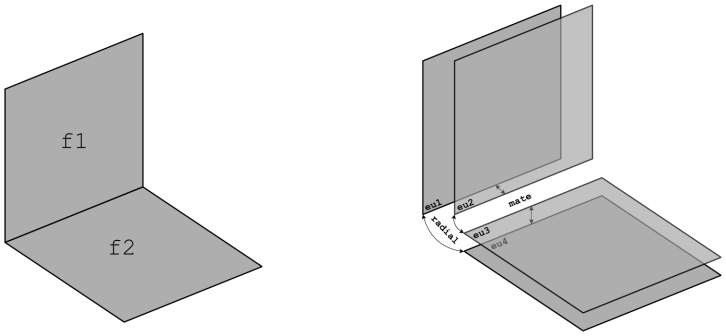
Obr. 2.8: Quad-Edge schéma

Radial Edge (RE) Podporuje non-manifoldné geometrické modelovanie v „boundary“ reprezentácii, ktorá ukladá informácie o susednosti v topológii a povoľuje reprezentáciu „wireframe“, povrchov a „solid“ meshov.

Entity podporované sú: r regióny, s „shells“ (povrchy), f „face“ (plochy), l „loops“ (cykly), e „edge“ (hrany) a v „vertex“ (vrcholy). A navyše *fu* face-use, *lu* loop-use, *eu* edge-use a *vu* vertex-use.

„Uses“ reprezentujú, ako sú topologické entity navzájom použité v non-manifold modeli a umožňujú ich použitie v solid modeloch.

Entita plochy napríklad neobsahuje žiaden smer. Obsahuje však „face-use“ k popisu všetkých možných orientácií danej plochy. Takýto popis obsahujú obe strany plochy. „Mate“ ukazovatele spájajú edge-uses na opačných stranách týchto plôch. Každá hrana nesie počet párov užitia, ktorý je rovný počtu párov „face-uses“, s ktorými sa dotýka. Podobne má vrchol počet „vertex-uses“ podľa počtu hrán, s ktorými je v kontakte cez „edge-uses“.

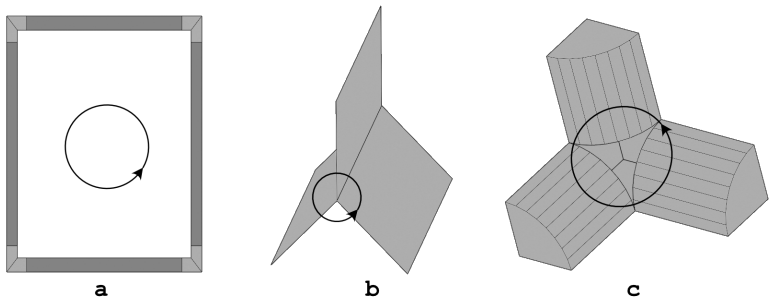


Obr. 2.9: Radial-Edge schéma

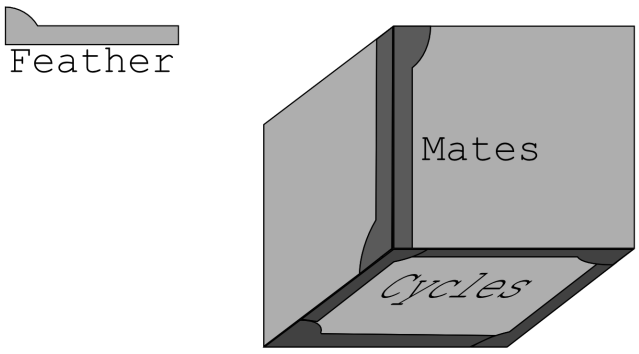
Feather Je dátová štruktúra non-manifoldnej topológie sústrediacej sa na susedné vzťahy a okrajové informácie. Reprezentuje tieto vzťahy v Euklidovskom priestore. Podporuje entity ako vrcholy, hrany, plochy a regióny.

Regióny sú ohraničené „shells“ (povrchmi), plochy sú ohraničené cyklami a okraje sú obmedzené diskmi. Tento model však nepotrebuje reprezentáciu pomocou „fans“, „blades“ alebo „wedges“. Namiesto toho vznikla jediná entita označená ako „feather“, ktorá ich nahrádza.

„Pierko vyzerá ako „edge-use“ v RE, no pierko nie je súčasťou hrany a ani vrcholu. Je to len reprezentujúca entita.“ (Yamaguchi a Kimura, 1995)



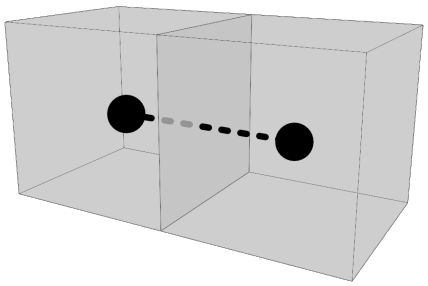
Obr. 2.10: a) Loop cycle b) Radial cycle c) Disk cycle



Obr. 2.11: Ukážka feather mates a cycles

Dual Half Edge (DHE) Je dátová štruktúra súvisiaca s RE a Half-Edge štruktúrami, no je založená na rozšírenej verzii quad-edge štruktúry.

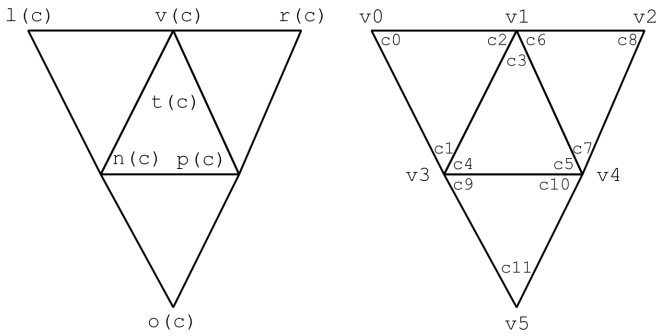
DHE používa 2 hlavné štruktúry, ide o „dual“ a „primal“, ktoré sú spojené, aby tvorili graf. Dual časť sa spája s topológiou a ukazovateľmi pre navigáciu, zatiaľ čo primal reprezentuje geometriu 3D modelu.



Obr. 2.12: Susedné bunky, zdeľajúce plochu sú spojené za pomoci „dual edge“

Corner-Tables Ukladá všetky informácie o topológii a spojoch v dvoch jednoduchých poliach V a O.

Oblasť pri vrchole sa nazýva „corner“ (roh), daný vrchol značíme ako „v(c)“. Roh oproti označujeme ako „o(c)“. Ďalej značíme ľavé, pravé, nasledujúce a predošlé vrcholy ako, „l(c)“, „r(c)“, „n(c)“ a „p(c)“. Trojuholník, v ktorom sa roh nachádza sa označuje ako „t(c)“. Vďaka tomuto vieme pristupovať každý polygon v meshi.

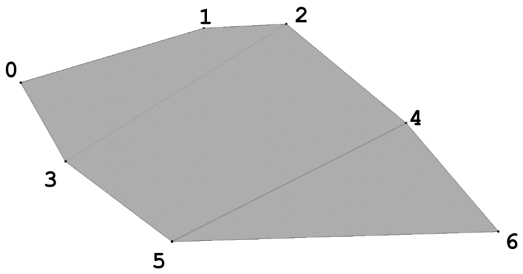


	V[c]	O[c]
0	0	5
1	3	-
2	1	-
3	1	11
4	3	8
5	4	0
6	1	-
7	4	-
8	2	4
9	3	-
10	4	-
11	5	3

Obr. 2.13: Corner-Table schéma. Chýbajúce záznamy značia non-manifold mesh

Vertex-Vertex Meshes Je štruktúra, ukladajúca len vrcholy a referencie na ich susedov. Informácie o hranách aj plochách vieme zistiť implicitne z reprezentácie.

Štruktúra nie je populárna, operácie nad ňou sú pomalšie a na vytvorenie hrán a plôch je potrebné prejsť celú štruktúru. Jej výhodou je ale malá pamäťová nenáročnosť.




```
Vertices
0: (-0.8;  0.9; -0.1)
     v1, v3
1: ( 0.8;  0.9; -0.1)
     v0, v2
2: ( 1.0;  0.1;  0.1)
     v1, v3, v4
3: (-1.0;  0.0; -0.1)
     v0, v2, v5
4: ( 0.9; -1.0; -0.2)
     v2, v5, v6
5: (-1.0; -1.0;  0.0)
     v3, v4, v6
6: ( 0.0; -1.9;  0.0)
     v4, v5
```

Obr. 2.14: Vertex-vertex schéma

Polygon Soup Je nijak neusporiadaná štruktúra polygónov, zvyčajne trojuholníkov, ktoré vznikajú v procese modelovania v programoch, ako je Maya alebo Blender.

2.1.6 Formáty

Elementy 3D modelu, ktoré sa rozhodneme zanechať, udáva formát, v ktorom budeme 3D modely ukladať. Formát taktiež udáva v akej dátovej štruktúre bude niesť informácie o geometrii. Dáta sa ukladajú v binárnej alebo „plain text“ forme. Konkrétne sa pokúšame uložiť geometriu (vrcholy, hrany a plochy), výzor (materiál, farba), scénu (kamera, osvetlenie) a animácie. Nie však každý model vie ponechať všetky tieto dáta [15].

Formáty delíme na:

Proprietárne Formáty vytvorené istou firmou, ktoré nie sú verejne dostupné. Slúžia prevažne k ukladaniu dát zo špecifického firemného software. Blender ukladá svoje dáta ako .blend súbory, AutoCAD ako .dwg. Dané formáty tvoria problém, pretože nie sú kompatibilné medzi viacerými software.

Neutrálne Formáty riešia problém kompatibility medzi software. Sú vytvorené prevažne pod Open Source licenciou, ktorú môže každý software implementovať na Import alebo Export.

Pozrime sa teraz presnejšie na stavebné bloky formátov 3D súboru.

1. **Kódovanie Geometrie 3D Modelu** Patrí k najzákladnejším vlastnostiam každého formátu. Ak by formát nepodporoval kódovanie geometrie, nemôže to byť 3D formát. Tieto formáty delíme na 3 časti:

   Approximate Mesh Ide o nepresný odhad 3D modelu, za použitia neprelínajúcich sa geometrických útvarov, taktiež sa to nazýva teselácia. Najčastejšie sa používajú trojuholníky, kde poradie vrcholov udáva zároveň aj normálu plochy. Nepresné preto, lebo sa oblé povrchy odhadujú len za pomoci rovných plôch.

   Precise Mesh Presný model vytvorený za pomoci parametrických povrchov. Konkrétne ide o rôzne krivky ako Non-Uniform Rational B-Spline alebo NURBS.

   CSG Ako už bolo raz spomenuté, ide o skladanie primitívnych meshov pomocou booleovských operácií do komplexnejšieho modelu.

2. **Výzor 3D Modelu** Môžeme rozdeliť na dve časti:

   Texture Mapping Používa 2D textúry na 3D modeloch, ktoré sú priradené pomocou materiálu a UV mapy.

   Atribúty Slúžia k priradeniu vlastností častiam meshu. Mení sa farba vrcholov, materiál plôch alebo rezy na hranách.

3. **Informácie o Scéne** Dôležité informácie, ktoré väčšina formátov neukladá. Ide o informácie ako kamery, svetlá a ich atribúty ako transformácie, ohnisková vzdialenosť, intenzita či farba.

   Taktiež sa môže držať informácia o iných 3D modeloch, ich viditeľnosti a vzájomné vzťahy.

4. **Animácie** Sa ukladajú hlavne vo formátoch používaných vo filmovom alebo hernom priemysle, preto sú podporované len niektorými formátmi.

   Skeletal Animation Ide o najpopulárnejší spôsob ukladania animácií. Daný model má priradenú takzvanú kostru, „skeleton“, ktorá je zložená z virtuálnych kostí. Pohyb kostí mení polohu meshu, za pomoci atribútov vrcholov.

   Kosťami hýbeme za pomoci FK - Forward Kinematics, čiže pohyb kosťou ovplyvní všetky kosti na ňu neskôr pripojené. Môžeme si to predstaviť ako pohyb ramennou kosťou. S ňou sa pohne predlaktie a dlaň.

Druhým spôsobom je IK - Inverse Kinematics, ide o presne opačný pohyb, kde dlaň zmení polohu nami vybraných kostí pred ňou.

Iné spôsoby Medzi, iné, menej populárne spôsoby patrí animovanie úplnou zmenou sady vrcholov, zmenou textúry, ktorá mení výzor objektu, napríklad textúra televízie alebo tečúcej vody. Textúrou môžeme animovať aj posunutie vrcholov.

Ako si zvoliť správny formát? Treba si položiť otázku, k čomu 3D model potrebujeme, čo potrebujeme, aby podporoval, v akom formáte pracujú ľudia okolo mňa a v akom formáte potrebuje model zákazník.

Teraz sa pozrieme bližšie na konkrétne štyri často používané formáty. Ide len o obmedzený výber, pretože formátov je viac ako 100.

2.1.6.1 Wavefront

Je zložený z dvoch formátov, ide o .obj a .mtl. ASCII verzia formátu je verejne dostupná, zatiaľ čo binárna verzia je proprietárna. Podporuje ukladanie aproximovaných súborov ako aj precíznych modelov. Aproximované modely nemusia byť tvorené len z trojuholníkových plôch [16].

Geometria 3D Modelu Formát dokáže uchovať polygónovú mesh a „free-form“ geometriu za pomoci matíc, Bezier, B-Spline, Cardinal kriviek. Najčastejšie ale uchováva informácie o geometrických vrcholoch, koordinátoch textúr, normály vrcholov a plôch.

Výzor 3D Modelu Uchováva informácie o farbe a textúre vo formáte .mtl. MTL súbor je „knižnicou“, ktorá môže obsahovať definície materiálov. Každý z nich špecifikuje farbu, textúru a odrazové schopnosti daného materiálu. Špecifikácia je postavená na Phongovom odrazovom modeli. V roku 2015 bol rozšírený, aby podporoval PBR model [17].

Scéna Dokáže definovať viacero objektov, no bez kamery a svetiel.

Animácie Nepodporuje dopredu determinované animácie či štruktúry kostí.

Listing 2.1: Ukážka .obj súboru na kocke.

```
mtllib OBJ.mtl
o Cube
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
vn -1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 1.0000
vn 0.0000 -1.0000 0.0000
vn 0.0000 1.0000 0.0000
usemtl Material
s off
f 1//1 2//1 4//1 3//1
f 3//2 4//2 8//2 7//2
f 7//3 8//3 6//3 5//3
f 5//4 6//4 2//4 1//4
f 3//5 7//5 5//5 1//5
f 8//6 4//6 2//6 6//6
```

- mtllib: Definuje, aký .mtl súbor použiť.
- o: Meno objektu, nepovinné.
- v: Poloha vertexov, povinné.
- vn: Vertex normály, nepovinné.
- usemtl: Špecifikuje konkrétny materiál, z .mtl súboru, ktorý použiť.
- s: Smooth shading, nepovinné.
- f: Elementy plôch, viacero štandardov.

Listing 2.2: Ukážka .mtl súboru.

```
newmtl Material
Ns 323.999994
Ka 1.000000 1.000000 1.000000
Kd 0.800000 0.800000 0.800000
Ks 0.500000 0.500000 0.500000
Ke 0.000000 0.000000 0.000000
Ni 1.450000
d 1.000000
illum 2
```

- newmtl: Názov materiálu.
- Ns: Specular highlight.
- Ka: Ambient color.
- Kd: Diffuse color.
- Ks: Specular color.
- Ke: Emmisive color.
- Ni: Index of refraction (IoR).
- d: Definuje, ako priehľadný v pozadí materiál je.
- illum: Špecifikuje, ktorý osvetľovací model použiť.

2.1.6.2 STereoLithograpy

STL je jeden z najstarších a najpoužívanejších formátov, používa sa hlavne v 3D tlači. Podporuje binárny a ASCII zápis [18].

Geometria 3D Modelu Podporuje len polygonálne meshe, ktoré sú tvorené len z trojuholníkov. Iné formáty a geometrické tvary nie sú podporované. Každý trojuholník je definovaný pozíciou každého z jeho troch vrcholov, s možnosťou definovať vlastnú normálu. Normálu nie je nutné zapisovať.

Výzor 3D Modelu ASCII štandard nepodporuje ukladanie výzoru meshu. Binárny formát sa pokúsil vytvoriť niekoľko spôsobov, ako zakódovať informáciu o farbe vrcholov. Žiaden sa ale neuchytil.

Scéna Nepodporuje viacero objektov, svetlá ani kamery.

Animácie Nepodporuje dopredu determinované animácie či štruktúry kostí.

Listing 2.3: Základná štruktúra .stl formátu.

```
solid name
    facet normal nx ny nz
        outer loop
            vertex v1x v1y v1z
            vertex v2x v2y v2z
            vertex v3x v3y v3z
        endloop
    endfacet
endsolid name
```

- Solid name, určuje názov objektu.
- Facet normal určuje, kam smeruje normála daného polygónu.
- Loops určujú polohy troch vertexov v 3D priestore.

2.1.6.3 Filmbox

FBX je proprietárny formát, binárny aj ASCII, vytvorený firmou Autodesk. Ide o populárny formát používaný v hernom a filmovom priemysle. Autodesk sprístupnil SDK, vďaka ktorému sa dá súbor vo formáte čítať, zapísať a konvertovať. Firma Blender Foundation sa pokúsila o rozlúštenie jeho binárneho formátu. Nasledujúce dáta sú pomerne presným odhadom, čo .fbx dokáže [19].

Geometria 3D Modelu Podporuje polygónové meshe, geometrické objekty a krivky. Meshe nemusia byť uložené len vo forme trojuholníkov. Dokáže uložiť aj vlastné normály.

Výzor 3D Modelu Podporuje rôzne materiály, farby vrcholov a mapovanie textúr.

Scéna Dokáže zakódovať viacero objektov, kamery a osvetlenie s mnohými atribútmi.

Animácie Podporuje skeletal animations.

Ako vidíme .fbx dokáže spracovať každú dôležitú časť 3D formátu.

2.1.6.4 Graphics Language Transmission Format

V skratke .gltf, pre jeho ASCII verziu, ktorá môže odkazovať na iné súbory alebo .glb, pre jeho binárnu verziu, ktorá má všetky dáta zakódované. Je

otvorený štandard vytvorený a udržiavaný spoločnosťou „Khronos Group“. Podporuje všetky základné stavebné bloky [20].

Geometria 3D Modelu Predvolene podporuje geometriu zloženú z trojuholníkov, podporuje taktiež módy založené na bodoch a čiarach. Meshe môžu byť zložené z uzlov.

Výzor 3D Modelu Podporuje zmenu farby vrcholov a rôzne atribúty pomocou uzlov. V najnovšej verzii podporuje PBR materiály.

Scéna Podporuje uchovanie celej scény, viacero objektov, kamier, svetiel, materiálov a uzlov.

Animácie Podporuje key frame animácie kostí, transformácie pomocou uzlov a animácie pomocou Morph Targets.

Pre viac informácii, odkaz na dokumentáciu glTF.

2.2 Základy Modelovania

Modelovanie je proces tvorby ľubovoľného 3D objektu, za použitia programu, na tvorbu 3D modelov. Ide o virtuálne remeslo, ktoré používa geometrické útvary a matematické vyjadrenia, pre tvorbu 3D geometrie.

Preberieme si dôvody, prečo niečo modelujeme, štýl, v ktorom model tvoríme a technickú časť tvorby.

2.2.1 Dôvody modelovania

V tejto sekcii si prejdeme dôvody vytvárania 3D modelu. Nebudeme sa zaoberať technickou časťou tvorby, ale pozrieme sa, k čomu sa daný 3D model bude používať, pretože dôvod za vznikom je niekedy dôležitejší ako kvalitné 3D spracovanie [21].

2.2.1.1 Produkty pre marketing

3D modely, určené pre produktové fotografie, sú najjednoduchšie. Nemusia byť vytvorené kvalitne alebo správnymi technikami. Ide len o výzor, modely musia vyzerať dobre aj napriek správnosti či reálnosti.

Bežne sa porušujú pravidlá ako prekrývajúca sa geometria, diery v modeloch, použitie n-uholníkov, nefyzikálne správajúce sa materiály či osvetlenie a mnoho ďalších trikov, ktoré z pohľadu kamery nie sú viditeľné.

Pre tvorbu 3D modelových produktov sa môže použiť väčšina modelovacích techník. Treba však vybrať správnu v pomere rýchlosti a kvality.

2.2.1.2 VR a AR

Tvorba modelov, určených pre virtuálnu a rozšírenú realitu, sa v dnešnej dobe považuje za jednu z najzložitejších odvetví. Je tomu tak, pretože dané aplikácie potrebujú vyobraziť modely čo najdôveryhodnejšie, aby čo najlepšie zapadli do reálneho sveta, a to z každého uhlu pohľadu.

Pri tvorbe týchto modelov nemôžu vznikať diery, nedostatky v osvetlení alebo rôzne naškálované UV mapy v istých častiach modelu. Model bude vždy viditeľný z každej strany, a je potrebné na to brať silný ohľad.

2.2.1.3 Hry

Hry patria obtiažnosťou medzi produktové a VR modely. Je dôležité, aby boli vytvorené kvalitne s použitím správnej topológie a nemali problémy s osvetlením. No používajú sa tu aj triky, ktoré sú vo VR zakázané, ako napríklad diery v častiach modelu, ktoré z pohľadu hráča nikdy viditeľné nebudú. Taktiež sa používa škálovanie UV máp pre rôzne časti modelu, podľa ich významnosti.

2.2.1.4 Iné druhy

Medzi zvyšné, nespomenuté, no dôležité dôvody tvorby 3D modelu patria odvetvia, ako napríklad architektúra, 3D tlač, animácie a filmy.

Pri architektúre si treba dávať pozor na správne rozmery a pomery objektov. Je vhodné použiť CAD programy.

Model určený pre 3D tlač je nutné vhodne navrhnúť na časti, ktoré do seba bude možné spojiť a zvoliť správnu veľkosť detailu tak, aby ho daná tlačiareň dokázala vytlačiť.

Animácie a filmy si potrpia na kvalitnej topológii a často si nechávajú možnosť geometriu neskôr zmeniť. Je tomu tak preto, lebo sa niektoré časti modelu nezmestia do záberu, alebo vytvárajú zbytočný detail, ktorý je potrebný odstrániť.

2.2.2 Štýly modelovania

Pojmom štýly modelovania myslíme štýl, v ktorom sa rozhodujeme modelovať, ide o: [22, 23].

2.2.2.1 Realizmus

Realistická grafika sa snaží vytvoriť čo najviac realistické digitálne modely. Nemusí to ale priamo znamenať, že dané modely a textúry sú v obrovských veľkostiach, skôr len chcú vzbudiť dojem, že je niečo možné, reálne. Realizmus si môžeme voľne rozdeliť na podštýly.

Foto-Realizmus Jeho cieľom je dosiahnuť čo najviac realistické zobrazenie scény, zväčša offline rendering a obrovské veľkosti dát. Využitie pre produktové vizualizácie alebo architektúru.

Realizmus Ako sme už spomínali, má za účel iba vytvoriť pocit, niečoho reálneho. Dáta sú veľkostne prijateľné, je možné spracovávať v reálnom čase. Využitie napríklad v simuláciách alebo tréningoch.

Nereálny-Realizmus Cieľom je vytvoriť niečo najviac dôveryhodne, čo by mohlo existovať, no neexistuje. Najčastejšie využitie je v hrách alebo filmoch. Všetci vieme, že draci neexistujú, no môžu vytvoriť pocit reality.

Možností na rozdelenie je samozrejme viac, mnohé použitia sa prekrývajú. Ide len o obrazné rozdelenie.

2.2.2.2 Štylizáca

Štylizovaná grafika sa pokúša vyvolať emócie, zdôrazniť tvary či farby, alebo zjednodušiť niečo komplexné. Príklady toho, čo môžeme považovať za štylizované modely.

Low-Poly Hlavným účelom low-poly štýlu je dostať počet polygónov, na čo najmenšiu hodnotu. Samozrejme, aj v tomto sú výnimky. Preferujeme niečo, čo vyzerá dobre oproti nižšiemu počtom polygónov.

Cartoon Ide o klasické štylizovanie, videné, napríklad v rozprávkach. Zväčšené oči, inak zdôraznené proporcie alebo výrazné použitie farieb.

Podobne ako v rozdelení realizmu, ide len o obrazné okruhy vytvorené na ukážku.

Pojmom štýly modelovania môžeme ale aj chápať, akého štýlu/formy je to, čo ideme modelovať. Túto kategóriu delíme na dve časti, ide o:

Organic Sú to modely tvorené živou hmotou, ľudia, zvieratá, niekedy rastliny alebo stromy.

Hard-Surface Modely tvorené neživými objektmi, ako napríklad kamene, autá, roboti alebo taktiež stromy.

Hranica medzi organickým a hard-surface modelovaním sa môže prekrývať, ako sme videli, napríklad pri stromoch. Môžeme ale vytvárať dizajny, ktoré budú potrebovať oba štýly.

2.2.3 Techniky modelovania

Alebo aj inak nazvané metódy modelovania, sú postupy, ktoré môžeme použiť pri tvorbe 3D modelu. Tieto techniky sú väčšinou zviazané s programom, ktorý používame. Rôzne techniky je vhodné ovládať z dôvodu zvolenia tej najlepšej pre model, ktorý vytvárame.

Techniky sa ale vždy dajú naučiť a s príchodom nových technológií sa menia. Preto je dôležitejšie poznať základy, či už umelecké alebo dôvody a štýly modelu, ktorý potrebujeme vytvoriť.

Teraz si povieme niečo o existujúcich technikách. Zoznam je skôr ilustračný a voľne vytvorený, než presný. Techník môže byť toľko, koľko je nástrojov. Preto sa v nasledujúcej časti budú vyskytovať len tie najviac známe a kategorizovateľné [24, 25].

2.2.3.1 Polygon/Box modeling

Najviac známa a populárna technika 3D modelovania. Používame všetky stavebné bloky 3D modelu ako vrcholy, hrany a plochy. Tie presúvame, rotujeme a zväčšujeme a používame nad nimi operácie ako „Extrude, Edge Loop, Insert“. Je to technická manuálna práca, často používajúca ortografické zobrazenie pre väčšiu precíznosť.

Mesh najčastejšie vytvárame zo štvor-stranných plôch, ktoré nazývame quads. Je tomu tak z dôvodu, aby nám fungovali nástroje, ktoré potrebujú pre ich funkčnosť „flow“ v topológii, ide napríklad o edge loop alebo predvídateľnosť funkčnosti Sub-D metódy.

Subdivision Surface slúži na zvyšovanie hustoty polygónov ich rekurzívnym delením na (najčastejšie) štyri podčasti. Týmto pomáha k dosiahnutiu lepšej zaoblenosti povrchu, ale aj k zvýrazneniu niektorých jej hrán za použitia „creases“.

Existuje však rozdiel, medzi Box a Polygon modelingom. Box vytvára modely už z existujúcich primitív, ako kocka či cylinder a manipuluje s plochami. Naopak Polygon modeling začína od jediného vrcholu a pracuje viac s vrcholmi a hranami. Rozdiely sú ale zanedbateľné, preto ich ponechávame v jednej kategórii.

Spoločné majú to, že používajú rovnakú sadu nástrojov na vytvorenie rovnakého cieľa, ktorým je najčastejšie hard-surface model.

Programy určené pre túto techniku sú napríklad Blender, Maya alebo Max.

2.2.3.2 Sculpting

Je modelovacou technikou, nazývanou aj digitálne sochárstvo, ktorá sa pokúša, čo najviac oddialiť od technického sveta a priblížiť tomu umeleckému.

Naďalej využívame vrcholy, hrany a plochy, no teraz ich netransformujeme postupne (po jednom) pomocou klasických modelovacích nástrojov, ale pracujeme s väčšími zhlukmi vrcholov, ktoré ovplyvňujeme štetcami.

Začať môžeme s modelom, ktorý má vysokú hustotou polygónov, ktorú nejak upravujeme, nie je to však populárna metóda, pretože nemôžeme veľmi meniť tvar objektu a je limitujúca.

Ďalším spôsobom je začať s nízkou hustotou, transformovať ju a počas tvorby použiť nástroje na „remeshing“. Tie vytvoria nové zloženie polygónov, zachovávajúc tvar modelu. Hustotu postupne zvyšujeme, až nám vznikne model, ktorý potrebujeme. Nástroje na remeshing sú, napríklad Voxel Remesh, Quad Remesh alebo Dynamesh.

Najviac všestrannou je multiresolution metóda. Podobne, ako Sub-D technika, delí mesh rovnomerne, s tým rozdielom, že mesh delí na levely. Každý level drží svoju informáciu o stave geometrie, čiže polohu vrcholov. Výsledkom je možnosť vykonávať zmeny v leveloch bez toho, aby vyššie levely ovplyvňovali nižšie a zmeny na nižších leveloch menili stav na vyšších. Príkladom môže byť zmena siluety modelu, ktorý má vytvorené mikrodetaily na povrchu.

Sculpting ortografické zobrazenia používa vo veľmi obmedzenom množstve. Najčastejšie pri odhadovaní proporcii alebo v špeciálon móde tvorby modelov, známom ako ShadowBox.

Ide o spôsob 2D kreslenia na 3 rôzne plochy, ktoré sú na seba kolmé a priesečník našich kresieb vytvorí model v 3D. V iných prípadoch sa používa klasické perspektívne zobrazenie.

Touto technikou sa najčastejšie vytvárajú organické modely ako ľudia alebo iné tvory. V hard-surface modelingu sa používa hlavne počas prvého štádia vytvárania dizajnu alebo na konci, pri pridávaní mikrodetailu.

Programy určené na sculpting sú, napríklad ZBrush alebo Blender.

2.2.3.3 Krivky a NURBS

Použitím tejto techniky väčšinou prestávame využívať polygónových meshov, ale vytvárame parametrické povrchy, ktoré dokážeme kontrolovať pomocou kontrolných bodov.

Pomocou NURBS vytvárame objekty buď priamym zostavením z kriviek, alebo vytvoríme len akúsi kostru či piliere. Medzi nimi budeme interpolovať, čím vytvoríme hladkú plochu spájajúcu dané krivky. Touto technikou sú postavené lietadlá, lode alebo iné vozidlá.

Obyčajné krivky vieme použiť napríklad k rotačnej symetrii okolo nami špecifikovaného bodu. Tým vytvoríme povrch (Screw modifier v Blenderi). Taktiež môžeme pridať falošnú hrúbku, alebo použiť iný objekt na pridanie hrúbky, ktorý je deformovaný pozdĺž krivky.

Pre prácu s pravými krivkami používame programy ako FreeCAD alebo Fusion360, kde je výhodou nekonečná presnosť, „bevel“ na akejkoľvek hrane a dokonalé boolean operácie. Vymeníme to, ale za rýchlosť práce a jednoduchosť tvorby. Pre prácu s krivkami, ktoré budú používať polygóny, môžeme použiť ktorýkoľvek program z kategórie polygon/box modeling.

2.2.3.4 Simulácie

Sú techniky, kedy model vytvára počítač, väčšinou bez vplyvu užívateľa, na základe algoritmov predom nastavených parametrov. Medzi simulácie zaraďujeme napríklad:

Cloth Slúži na výpočet polohy a pokrčenia oblečenia postáv. Najpoužívanejší software je Marvelous Designer.

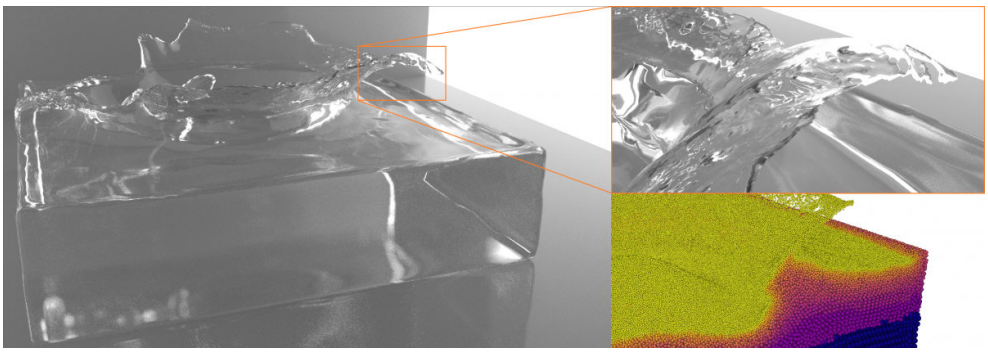
Fluid Môžeme rozdeliť na viacero častí, napríklad tekutiny, plyny alebo oheň. Každá z týchto častí ma svoje osobitné nastavenia, ich funkčnosť je jasná z názvu. Najpopulárnejší program k simulácii týchto efektov je Houdini, pre real time to je EmberGen.

Particles Ide o systém, simulujúci vysoký počet objektov, väčšinou jedného druhu s interakciami medzi nimi. Môže ísť napríklad o výbuch alebo iskrenie, väčšina moderných programov podporuje nejaký systém na tvorenie particle simulácii.

Collision Simuluje kolízie medzi objektami alebo vzájomné kolízie objektu.

Soft-Body Vytvára simulácie objektu, ktorý nie je pevný, ale dokáže sa ohýbať. Môžeme použiť k vytvoreniu želatíny alebo svalovej hmoty.

Ako pri mnohých kategóriách, znova ide o obrazné rozdelenie, ktoré sa nemusí vzájomne nevylučovať. Najpopulárnejší software pre simulácie všetkého druhu je Houdini.



Obr. 2.15: Ukážka fluid simulácie [26]

2.2.3.5 Photogrammetry

Je jedna z najnovších techník, funguje na princípe fotografovania reálneho objektu. Ten sa následne v programoch spracuje na point cloud. Zo vzniknutého point cloudu sa zostaví 3D mesh s UV mapou. Farba bodov z point cloudu sa následne „zapečie“ do textúry objektu a vznikne náš odfotený model.

Výhodou tejto techniky je výsledný foto-realizmus modelu, keďže je vytvorený z fotiek reálneho objektu. Nevýhodou je však kvalita geometrie. Často krát nemá dostatočné rozlíšenie, má odpojené časti, diery. Takmer vždy je potrebné vytvoriť novú topológiu a preniesť na ňu detaily skenu. Ďalšou nevýhodou je základ v reálnom svete, nie vždy môžeme naskenovať všetky strany objektu a niektoré vôbec neexistujú.

Táto technika sa používa na tvorbu organických aj hard-surface modelov. Používajú sa programy ako Reality Capture alebo MeshRoom.

Obr. 2.16: Ukážka vytvárania fotografií [27]

2.2.4 Modelovacia pipeline

Modelovacia pipeline je postup tvorby 3D modelu od začiatku až po koniec. Každý jedinec a štúdio má svoje nástroje a postupy, ako tvoriť 3D model. Skúsime si ale ukázať, ako môže vyzerať jeden konkrétny postup, ktorý je používaný primárne v zábavnom priemysle. Samozrejme, kroky v postupe sa môžu prelínať, k niektorým sa môže vracať, alebo sa môže začať úplne odznova.

2.2.4.1 Block-Out

Blockout slúži k obraznému zostaveniu modelu, ktorý chceme vytvoriť. Používajú sa hlavne primitívne objekty, ktoré spájame pomocou booleovských operácii. Následne ich upravujeme za pomoci techník polygónového modelovania. Môžeme ale použiť aj iné techniky. Najčastejšie sa používa kombinácia viacerých, pretože ide len o blockout.

Druhou často používanou technikou je Sculpting alebo Kit-Bashing. Slúži k rýchlemu konceptovaniu z už predom vytvorených objektov. Pri organických modeloch to môžu byť dopredu vytvorené ruky, hlavy alebo oči a v hard-surface modeloch to môžu byť rôzne detaily, ventilátory alebo kolesá.

Výhodou blockoutu sú rýchle iterácie. Model zostavíme, necháme ho zhodnotiť, a ak nie je dobrý, začneme odznova. Avšak ak model vyhovuje, prejde sa na presnejší blockout, kedy sa dôraznejšie zvážia všetky proporcie, pridajú menšie detaily a model sa predá do ďalšej časti.

Je možné použiť akýkoľvek program, s ktorým užívateľ vie pracovať.

2.2.4.2 High-Poly

High-Poly slúži k pridaniu detailu nášmu blockoutu. Nad blockoutom sa najčastejšie použije algoritmus automatickej retopológie a na ten sa použije multiresolution metóda. High poly vieme rozdeliť do 3 častí.

Low-Frequency Súčasti modelu, ktoré vidno z veľkej vzdialenosti a tvoria hlavnú objemovú časť modelu alebo siluetu. Pre nás to je väčšinou vstup z blockoutu. Ide o stovky až tisícky polygónov.

Napríklad ide o siluetu, proporcie a najväčšie svaly ľudského tela.

Mid-Frequency Časti modelu, ktoré pridávajú menšie objemy na predošlú frekvenciu. Ide o vrstvy, ktoré môžeme vidieť, ak obrázok rozmažeme alebo prižmúrime oči a zbavíme sa tak zbytočných detailov. Ide o tisícky až desať tisícky polygónov.

Pre príklad ide o tukové vrstvy alebo malé svaly na ľudskom tele.

High-Frequency Ide o časti modelu, ktoré len pridávajú detail na predošlé vrstvy a sme schopní bez nich vnímať model ako hotový, pridávajú len

najvyšší možný realizmus. Je to mikroskopický detail, ako napríklad póry v koži. Pôjde rádovo o stotisíc až milión polygónov.

Frekvencie by mali vznikať v danom poradí a ich dôležitosť je ním taktiež určená. Najdôležitejšia je najnižšia frekvencia, pretože zachytáva objekt samotný a stredná frekvencia akoby objekt spojí do jedného celku.

Najmenej dôležitá je posledná, najvyššia frekvencia, pridáva ultra realizmus, no vie pridať aj zbytočný šum. Príkladom sú sochy, ktoré takmer nikdy nemajú vysokú frekvenciu, pretože ju bolo nemožné vytesať.

Najčastejšie sa používa ZBrush alebo Blender.

2.2.4.3 Retopo/Low-Poly

Retopológia je časť procesu, kedy sa na high-poly model položí nový low poly mesh, ktorý neobsahuje milióny polygónov, ale len niekoľko desať tisíc. Presný počet sa neurčuje, pre AAA postavy v hernom priemysle to je približné maximum 100-tisíc.

Na tvorbu sa používa väčšina programov ovládajúcich polygónové modelovanie, osobitný program je napríklad TopoGun.

2.2.4.4 UVs

Aby sme mohli náš nový low poly mesh otextúrovať, je potrebné mu vytvoriť UV mapu rozrezaním jeho povrchu do plochy. Objekt sa rozrezáva v častiach, ktoré sú najviac zakryté, alebo budú najmenej viditeľné z pohľadu kamery, pretože často krát je tieto rezy vidno. Rozrezané „ostrovy“ je následne potrebné poukladať na 2D plochu tak, aby texel-density bola všade rovnaká.

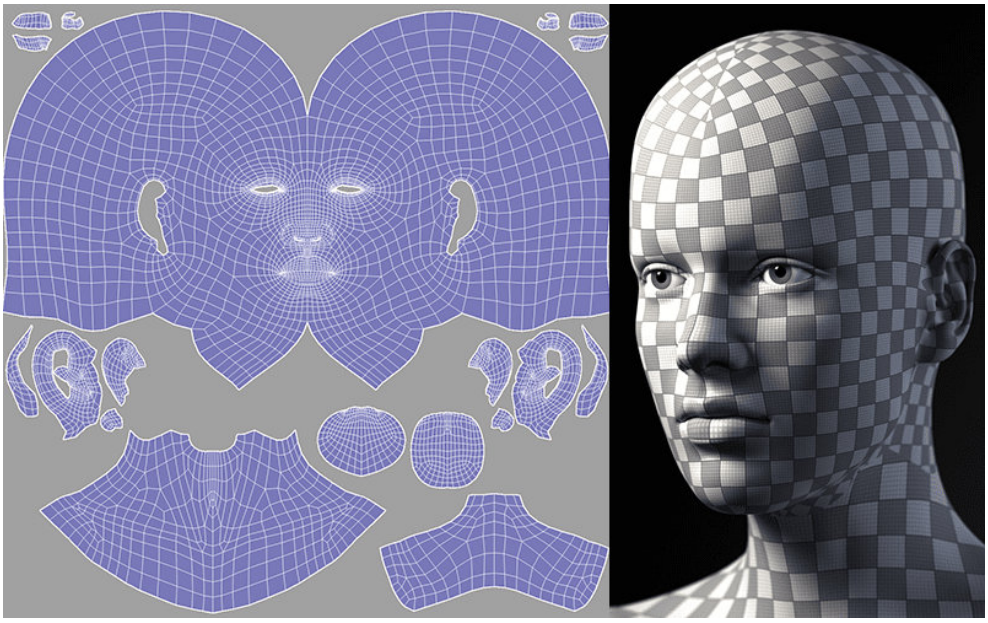
Texel-density v skratke udáva veľkosť detailu pixela v pomere veľkosti rozrezaného ostrovu. Chceme, aby bola všade rovnaká z dôvodu zanechania detailu a dobrej náväznosti textúr.

Zároveň musíme brať na vedomie, aké textúry budeme chcieť použiť a podľa toho správne neorientovať UV ostrovy. Ak chceme, aby rovné čiary ostali rovné, je potrebné si dať pozor, aby ostrov, ktorý bude mapovať textúru, nebol natočený. Ak by bol, vzniká riziko, kedy naše priamky budeme ukladať do diagonálnej pixelovej mriežky.

Ako je už zvyčajné, väčšina programov upravujúca polygóny dokáže tvoriť UV mapy. Špecializovaným programom je RizomUV.

2.2.4.5 Texturing

Je proces pridávania textúr alebo farbenia nášho 3D modelu. V súčasnosti sa v hernom priemysle používa Physically Based Rendering (PBR). Ten využíva rôzne textúry, na dosiahnutie efektu. Konkrétne sú to mapy, ako napríklad: Albedo (farba modelu, bez tieňov), Roughness (ostrosť odrazeného svetla), Normal (mení normály meshu), ...



Obr. 2.17: Ukážka UV mapy [28]

PBR je len podmnožina všeobecnej odrazovej funkcie (general reflectance function). GRT je 16-dimenzionálna funkcia pracujúca s uhlami, intenzitami, bodmi vstupu/výstupu a časom prijatého svetla. Ďalšou podmnožinou môže byť napríklad funkcia BSDF, s ktorou pracuje Blender. BSDF dokáže spracovať PBR materiály. Vo filmovom priemysle, sa naopak, začína používať BTF. BTF využíva informácie z textúr, ktoré zaberajú niekoľko desiatok gigabytov.

Z programov používaných na textúrovanie je v hernom priemysle najviac používaný Substance Painter a vo filmovom priemysle Mari. Rozdiel je vo veľkosti textúr, ktoré dokážu spracovať a v počte UDIM miest, na koľko je UV mapa rozdelená.

2.2.4.6 Rigging a animácie

Ak chceme objekt animovať, je potrebné mu vytvoriť kostru, tento proces sa nazýva rigging. Kostra sa nezhoduje s kostrou reálneho objektu, ide skôr o miesta, s ktorými potrebujeme pohybovať. Každá kosť takto ovplyvňuje pomocou vertex weight groups isté vrcholy, s ktorými vie pohybovať, čím deformuje mesh objektu.

Ak pohyby týchto kostí zanimujeme do key-frame animácií, získame tým animácie celého objektu.

Samostatné kategórie nie sú preberané do hĺbky, každá z nich je svojim odvetvím v 3D svete. Spracuvávajú sa približne v tomto poradí až po retopológiu, odvtedy je model možné poslať osobitne na UV mapy a rigging.

Kapitola 3

Chyby Meshu a 3D Modelu

V tejto kapitole si povieme najčastejšie problémy, ktoré vznikajú pri tvorbe polygónových meshov. Taktiež sa pozrieme na spôsoby, ako sa dajú tieto problémy riešiť, či už manuálne alebo automatizovaným procesom.

Všetky ukážky problémov a ich riešení budú znázornené v programe Blender. V ňom budeme používať workflow určený pre tvorbu modelov do VR hier, presnejšie do herného enginu Unreal Engine, na ktorom je postavená VR aplikácia projektu VMCK. To presnejšie znamená, že sa budeme pokúšať tvoriť našu geometriu len za pomoci štvorcových polygónov, mesh bude musieť byť uzavretý a prezentovateľný z každej strany.

3.1 Programy a Aplikácie

VMCK Je projekt prezentujúci českú históriu. V našom prípade ide o mesto Hradec Králové, pre ktoré vzniká VR aplikácia. Tá bude slúžiť k jeho vizualizácií, ceste časom a rekonštrukcií zaniknutých častí mesta.

Unreal Engine Ide o herný engine prvý krát predstavený v roku 1998, odvtedy bol známy jeho kvalitou v oblasti grafiky. Vďaka tomu sa v posledných rokoch pretvára na nástroj vyzobrazovania a vytvárania realtime grafiky, ktorý slúži na vizualizácie aj v odvetviach, ako je architektúra, filmy a správy.

My ho budeme využívať ako vizualizátor pre našu VR aplikáciu VMCK.

Blender Je open-source 3D grafický program, prvý krát predstavený v roku 1994, používaný k tvorbe animovaných filmov, vizuálnych efektov a 3D modelov slúžiacich v mnohých odvetviach.

Budeme používať ako nástroj na tvorbu modelov a riešenie problémov vzniknutých v meshoch, ktoré sú dodané treťou stranou.

3.2 Najčastejšie problémy

V tejto podsekcii si povieme, aké sú najčastejšie problémy, s ktorými sa stretávame počas modelovania v programe Blender.

3.2.1 Transformácie

Medzi jednu z najčastejších chýb určite patrí zlé nastavenie „scale“ objektu v Blenderi. Scale je 3D vektor, ktorý udáva v osiach x, y, z zväčšenie objektu. To znamená, že scale hodnota sa drží na úrovni objektu a nie meshu. Môžeme, teda, kľudne vymeniť mesh dáta a daný objekt rozhodne, ako bude mesh vyzerať.

Scale by mala byť ideálne nastavená vo všetkých dimenziách na hodnotu 1. Je tomu tak preto, aby sedeli rozmery medzi objektom, meshom a vznikala kompatibilita s inými aplikáciami. Tu môže vznikať dvojaký problém.

1. Scale má nastavené vo všetkých osách rovnaké nejednotkové hodnoty.

2. Scale má nastavené v osách vzájomne iné hodnoty.

Oba z týchto problémov sa dajú v Blenderi riešiť použitím príkazu „Apply Scale“, ten nastaví scale vo všetkých osách na jedničku. Mesh dáta sa týmto nezmenia, ale modifiers, ktoré mesh ovplyvňujú, sa zmeniť môžu. Hlavným príkladom je Bevel Modifier, ktorý môže zmeniť šírku hrán, ktoré na meshi vytvára. Riešením, ak sa jedná o prvý prípad, je hodnotu, ktorá sa nachádza v modifieri, podeliť hodnotou scalu pred operáciou „Apply Scale“. Ak sa jedná o prípad číslo dva, jednoduché automatizované riešenie neexistuje a je potreba vytvárať zmeny na úrovni meshu.

Ak v prípade číslo dva pracujeme na úrovni meshu, môžu vznikať nečakané problémy na jednoduchých operáciach, ako napríklad „Extrude“, kde nová plocha, ktorú vytvoríme môže byť zvláštne deformovaná, alebo nemusí byť vytvorená v smere normály, ako je to nastavené defaultne. Tu sa dá problém riešiť dvojakým spôsobom, môžeme znova použiť operáciu „Apply Scale“, čím problémy v editácií meshu zmiznú a za predpokladu, že sme nepoužívali žiadne modifiers, môžeme modelovať ďalej. Druhým riešením je nezväčšovať na úrovni objektu, ale vždy na úrovni meshu, týmto postupom nikdy nenastane zmena hodnôt scale na úrovni objektu, vďaka čomu nevzniknú problémy pri úprave meshu.

Scale môže spôsobovať problémy aj pri tvorbe UV máp, ak scale nie je jendotkový, je vysoká šanca, že nami vytvorená UV mapa bude natiahnutá a textúry na ňu namapované taktiež.

To isté môže platiť, ak exportujeme objekt do inej aplikácie, v ktorej môže vyzerať úplne inak a nemusia na ňom fungovať pred definované shaders, ktoré sa spoliehajú na veľkosť meshu.

Posledným problémom, ktorý si spomenieme, je scale nastavený na príliš veľké hodnoty. To môže spôsobiť dva problémy, prvým z nich je, že model sa nezobrazí celý a druhý je extrémnejší, a tým sú vznikajúce chyby pri zaokrúhľovaní floating point čísel. Jednoduchým riešením je objekt zmenšiť na reálne veľkosti, čím sa vyrieši problém zaokrúhľovania aj zlého zobrazenia. Ak však máme model v jeho skutočnej veľkosti a vzniká len problém pri zobrazovaní, je možné nastaviť hodnoty „Near/Far Planes“ tak, aby zachytili celý objekt.

Problémy môže vytvárať aj lokácia objektu, tu vzniknuté problémy však nie sú až tak drastické ako pri zlej veľkosti. Lokáciou myslíme miesto, kde sa objekt nachádza, ale aj miesto, kde sa nachádza mesh v závislosti k jeho „Origin Point“ (alebo aj stred objektu).

Ak je mesh a objekt so stredom na približne rovnakom mieste a v okolí stredu súradnicového systému, problémy nevznikajú. Avšak ak celý objekt posunieme príliš ďaleko od stredu, začne nám vznikať zaokrúhľovací problém, podobne ako so zväčšeným objektom. Tento problém však nevzniká často.

Vzájomná poloha meshu a stredu objektu je problém významnejší. Blender štandardne pracuje so stredom objektu ako s jeho „ťažným bodom“, to znamená, že operácie rotácie alebo posunu sú závislé na ňom. Taktiež niektoré modifiers, ako napríklad „Mirror Modifier“ pracujú podľa stredu objektu, nie podľa stredu objemu objektu. Chyby, ktoré týmto spôsobom vzniknú patria medzi menšie problémy, nie veľmi ovplyvňujúce prácu s meshom. Väčšina z nich sa dá opraviť operátorom „Apply Location“.

Problémy však vznikajú pri exportovaní objektu. Nie každý formát musí nutne podporovať „Origin Point“, takže objekt, ktorý vytvoríme a správne mu priradíme stred, no posunieme ho vedľa počiatku súradnicového systému, môže byť v iných aplikáciách o tú vzdialenosť posunutý. Riešením je objekt vždy umiestniť do stredu súradnicového systému pred tým, ako ho budeme exportovať a uistiť sa, že origin point je taktiež nastavený v strede. Týmto sa zamedzia problémy vznikajúce s formátmi, ktoré nepodporujú stredy objektov, ale pracujú len so stredmi súradnicového systému.

3.2.2 Otočené normály

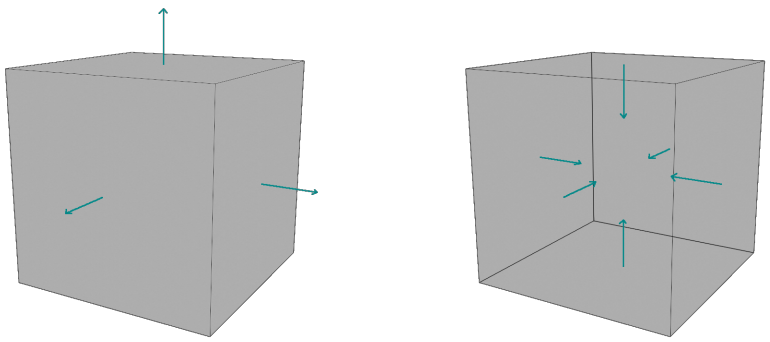
Normála je vektor kolmý k danej ploche. V 3D grafike slúži k výpočtu osvetľovania a tieňovania, taktiež je ale určená k orientovaniu niektorých modelovacích nástrojov.

V predošlej kapitole sme si povedali, že pracujeme s takzvanými „Surface“ objektmi, to znamená, že naše plochy sú nekonečne tenké a normálu môžu mať len jednu. Najčastejším spôsobom reprezentovania normál je využitie zápisu vrcholov, ktoré tvoria plochu, pre ktorú chceme normálu vytvoriť. Ich poradie, či už v smere, alebo proti smere hodinových ručičiek udáva, ktorým smerom bude normála natočená.

V tomto momente nastáva náš problém otočených normál. Ak usporiadanie vrcholov nášho meshu nie je jednoznačné, môžu niektoré normály ukazovať smerom do objektu, nie od neho. Výsledkom toho sú zle výpočty udávajúce osvetlenie meshu. Blender to v defaultnom „Flat Shadingu“ veľmi nerieši a plochy zobrazuje správne, naopak UE plochy otočené smerom od kamery ani nezobrazuje.

Riešenie tohto problému je pomerne triviálne, stačí nájsť plochy, ktoré majú zlú orientáciu, a tie pomocou príkazu „Flip“ otočíme. Príkaz funguje na plochách, ktoré sú označené. Ak chceme použiť automatickú opravu, máme na výber operátory „Recalculate Inside/Outside“, ktoré nasmerujú normály, snáď, správnym smerom.

Existuje viac problémov, ktoré existujú s normálami, povieme si o nich v neskoršej sekcií, ktorá sa nebude venovať najčastejším problémom.



Obr. 3.1: Kocka s otočenými normálami

3.2.3 Dvojitá geometria

Ako samotný názor implikuje, ide o problém, kedy sa v priestore nachádza na rovnakom mieste viacero vrcholov, hrán alebo plôch. Tento problém sa prejavuje zlým zobrazovaním, pretože sa v danom mieste môžu vyskytovať normály, ktoré ukazujú iným smerom, alebo sa prejavuje nefunkčnou geometriou pri používaní nástrojov spoliehajúcich sa na topológiu ako „Loop Select“ či „Subdivision Surface Modifier“.

Reálne ide však len o dvojité vrcholy, v rámci jedného meshu. Riešenie je jednoduché. Blender obsahuje nástroj s menom „Mege by Distance“, ktorý zrúti všetky vrcholy, ktoré sú od seba vzdialené na istú vzdialenosť do jedného vrcholu, čím sa zbavíme duplicitnej geometrie.

3.3 Chyby Meshu a Shadingu

3.3.1 Non-Manifold

Non-manifold je mesh, ktorý nemôže exitovať v reálnom svete. Manifold je naopak geometria, ktorá lokálne pripomína n-rozmerný euklidovský priestor, aj keď celkovo môže vyzerať inak. Pre nás to znamená, že mesh gule je zložený z malých plochých trojuholníkov. Manifoldný mesh je taktiež spojitý a nemá žiaden začiatok ani koniec, ako napríklad torus alebo kocka. Konkrétne pracujeme v dvoj-rozmernom manifoldnom priestore, z ktorého vytvárame 3D objekty. Vďaka tomuto faktu je možné z 3D geometrie vytvoriť UV mapu. Ak je nemožné vytvoriť UV mapu takú, aby všetky normály jej plôch smerovali rovnakým smerom, objekt je non-manifold.

Ak je objekt non-manifold, tak to v digitálnom svete väčšinou nie je problém, pretože je bežné vytvárať veci, ktoré nemôžu existovať. Ak však chceme, aby naša geometria bola vytvorená správne, je potrebné, aby bola manifold. Bez toho by nebolo možné vytvárať mesh technikou booleovských operácií, správne renderovať efekty ako refraction, vytvárať simulácie plynov alebo vody a 3D tlačiarne by model nevedeli vytlačiť.

Medzi konkrétne geometrické problémy patria diery v geometrii, vnútorné plochy, hrany spájajúce iný počet ako 2 plochy, vertex spájajúci oddelené povrchy, odpojené vrcholy a hrany, plochy bez hrúbky a susedné plochy s opačnými normálami.

Automatická oprava non-manifold meshu neexistuje, je však možné každý z daných problémov identifikovať automaticky a manuálne sa rozhodnúť, ako problém opraviť [29, 30, 31, 32].

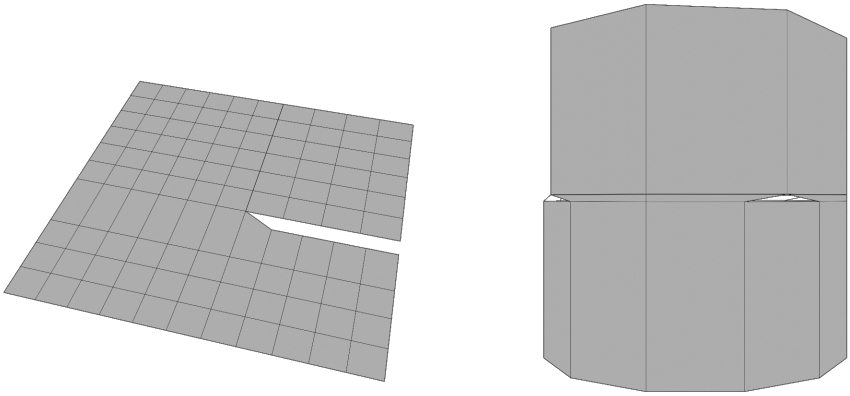
Diery Sú miesta, kde v geometrii chýbajú polygóny, čím vzniká nereálny objekt s nekonečne tenkou hrúbkou. Môže ísť konkrétne o diery alebo o takzvané „cracks“ alebo „T-vrcholy“ Nie sú tým myslené „topologické“ diery, ako napríklad v objekte torus [33].

Takéto diery sú problém, pretože vytvárajú problémy pri počítaní simulačných algoritmov pri zobrazovaní, kedy vnútro objektu viditeľné cez dieru sa stane priehľadné a 3D tlačiarne nie sú schopné daný objekt vytvoriť.

Diery môžu vzniknúť napríklad pri zlom procese skenovania, kde sa časť modelu nepodarí a vznikne diera, taktiež môže vzniknúť booleovskými operáciami. Na opravu dier, ktoré nie sú veľké, je možné použiť vstavané operácie v Blenderi ako „Grid Fill“, „Fill“, alebo manuálne vytvoríme chýbajúcu geometriu. V niektorých prípadoch vzniknú N-gony, o ktorých si povieme viac v ďalšej časti.

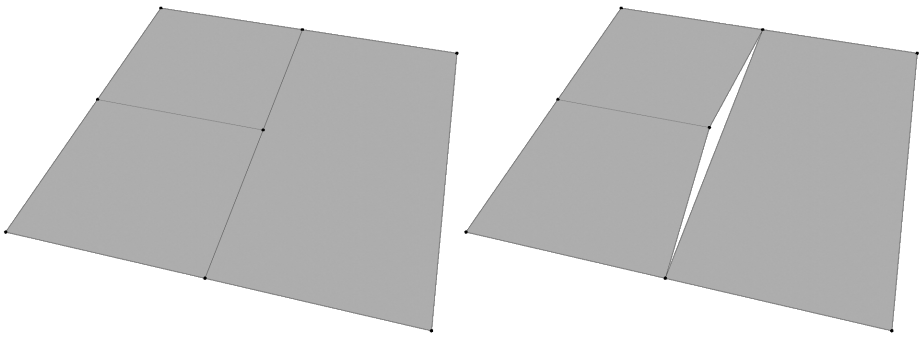
„Cracks“ sú „úzke“ diery, zárezy na kraji objektu alebo miesta, kde sa spájajú meshe s rozdielnym počtom polygónov. Takáto geometria sa nespojila a vznikli malé dierky. Na opravu sa používa takzvaný „stitching“,

kde sa počet polygónov na oboch stranách nejakým spôsobom dorovná, či už pridaním alebo odobraním a hrany sa následne k sebe pripoja a zlúčia.



Obr. 3.2: Crack na okraji objektu a pri znížení počtu hrán

„T-vrcholy“ sú susedné polygóny, ktoré vyzerajú, že sú spojené, pretože hrany ležia v rovnakom mieste, no na jednej časti polygónu je v strede vrchol navyše. Opravu prevedieme pridaním nového vrcholu a vytvoríme trojuholníkové plochy, kde vertex chýbal. Následne dané vrcholy zlúčime.



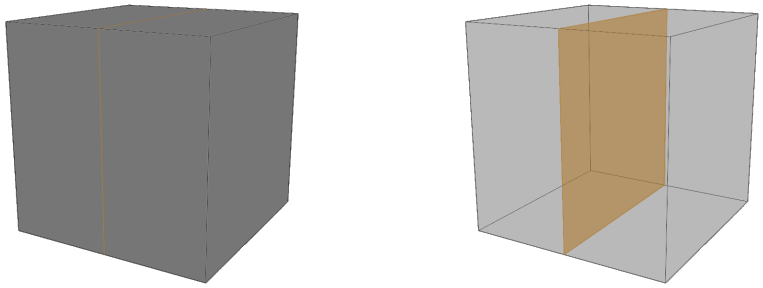
Obr. 3.3: T-vrcholy

Vnútorné plochy Ide o plochy, ktoré sa nachádzajú vo vnútri geometrie, no sú spojené s povrchom.

Spôsobujú problém, pretože povrchové plochy nevedia, kam topologicky pokračovať. Spôsobuje to problém pri algoritmoch spoliehajúcich sa na topológiu a vytvárajú problém pri shadingu. Hrana, na ktorú sa dané plochy napájajú, nemá jasne určenú normálu a vytvára artefakty na povrchu.

Vznik týchto plôch počas modelovania vzniká väčšinou len nedbalosťou. Napríklad označením nežiadúcich hrán a pokusom ich vyplniť. Užívateľ si nič nevšimne, no vznikne nová interná plocha. Podobne môže vzniknúť vnútorná plocha pri zle nastavených booleovských operáciách, alebo pri použití „Mirror Modifier“ so zapnutým zlučovaním v strede, kde sa spoja nežiadúce plochy.

Tieto vnútorné plochy môžeme jednoducho vymazať bez následku. Blender natívne podporuje príkaz „Interior Faces“, ktorý dané plochy označí.



Obr. 3.4: Vnútorné plochy

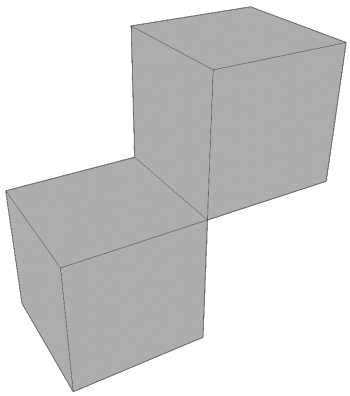
Hrany spájajúce nie 2 plochy Môže ísť buď o prípad, kedy je hrana spojená len s jednou plochou. To nastáva v momentoch, kedy je v meshi diera, alebo ak je mesh nekonečne tenký (Plane). Druhý prípad nastane, ak je hrana spojená s viac ako 2 plochami. Vtedy znova nastáva prípad, kde topológia nevie kam pokračovať.

Vznik môže byť zapríčinený nepozornosťou a omylným spojením dvoch osobitných povrchov, napríklad pri použití príkazu „Merge By Distance“, kedy sa nám spoja vrcholy, aj ktoré sme nechceli.

Dané hrany vieme nájsť pomocou príkazu „Non Manifold“, v ktorom môžeme označiť možnosť „Boundaries“, čo nám označí hrany, ku ktorým je pripojená len jedna plocha. Môžeme zaškrtnúť ale aj možnosť „Multiple Faces“, ktorá nám označí hrany, ku ktorým je pripojených viac ako

2 plochy. Taktiež môžeme nájsť konkrétny prípad takejto hrany a použiť „Select Similiar/Amount of Faces Around an Edge“.

Takto nájdené hrany môžeme buď odstrániť, rozdeliť mesh na 2 rôzne povrchy, priestor vyplniť, alebo pridať hrúbku objektu.



Obr. 3.5: Dva povrchy zdieľajúce jednu hranu

Vertex spájajúci oddelené povrchy Ide o vertex spájajúci dva rozdielne povrchy, hovorí sa mu aj „Bow Tie“ kvôli výzoru.

Vytvára to problém, pretože ide o spoj bez hrúbky a o len jeden vrchol, na ktorý je položená priveľká váha. Takéto vrcholy pokazia shading a algoritmy spoliehajúce na topológiu.

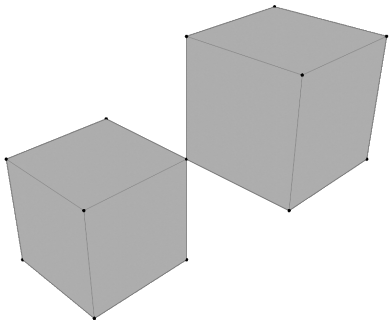
Chyba tohto druhu môže nastať taktiež napríklad použitím operácie „Merge By Distance“, ktorá spojí nežiadané vrcholy, alebo nepozorným modelovaním, kde sa použije vrchol z iného povrchu.

Oprava je podobná ako v predošlom prípade. Môžeme povrchy rozdeliť a vytvoriť z nich dva objekty, alebo manuálne odstrániť zlý vrchol a následne meshe opraviť.

Geometria bez objemu Je geometria ako vrchol v predchádzajúcej časti, alebo len vytvorená plocha, ktorá má nulovú hĺbku.

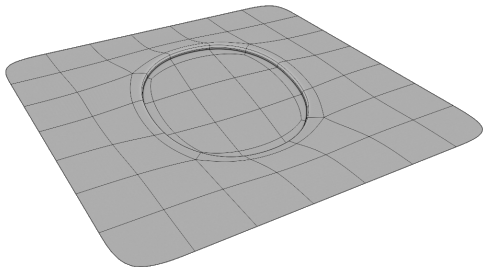
V realite neexistuje nič, čo by malo nekonečne malú hrúbku, ako to je u polygónov. Aj papier je hrubý niekoľko desatín milimetrov. Preto sú tieto plochy problém, pretože ich nie je možné vytvoriť v reálnom svete, narozdiel od virtuálneho, kde ide o základ „Shell modelovania“.

Tento problém môže vzniknúť mnohými spôsobmi. Môže ísť len o plochu bez akejkoľvek hrúbky. Taktiež to môže byť objekt s dierou, alebo ako v predošlom prípade jediný vrchol.



Obr. 3.6: Vrchol spájajúci oddelené povrchy

Mesh tvorený len v rovine môžeme opraviť napríklad použitím „Solidify Modifier“, ktorý z našej plošnej geometrie vytvorí „solidný“ mesh. Ak ide o diery alebo spomínaný vrchol, môžeme použiť riešenia použité vyššie.



Obr. 3.7: Geometria bez objemu (plochá geometria)

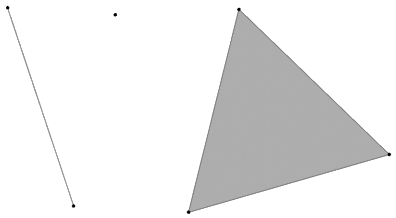
Odpojená geometria Ide o geometriu, ktorá nie je nijak spojená so zvyškom meshu. Môže ísť o odpojené vrcholy, hrany, plochy alebo aj celé povrchy.

Takéto oddelené časti spôsobujú problém taktiež preto, lebo nie je možná ich existencia v reálnom svete a môžu narúšať stavbu meshu. Napríklad nie je možné, aby existovala odpojená geometria od zvyšku inej geomet-

rie. Simulácie by nedávali správne výsledky a 3D tlačiarne by sa pokúšali tlačiť bez spojov so zvyškom geometrie.

Odpojená geometria môže vzniknúť, napríklad, ak vložíme nový objekt v takzvanom „Edit Mode“. Druhý spôsob je manuálne oddelenie geometrie. Tretím spôsobom je, ak geometriu skopírujeme a necháme na tom istom mieste. V tomto prípade vznikne geometria odpojená a aj zdvojená.

Tieto problémy vieme vyriešiť jednoducho, je však potrebné manuálne zhodnotiť, o ktorú z chýb ide. Ak ide o rozdielne povrchy, stačí ich rozdeliť do dvoch objektov alebo spojiť. Ak ide o manuálne rozdelenie, je potrebné sa zamyslieť, čo malo byť vytvorené a následne geometriu buď znova spojiť, alebo rozdeliť do dvoch objektov. Ak ide o omylné vytvorenie, stačí geometriu vymazať.



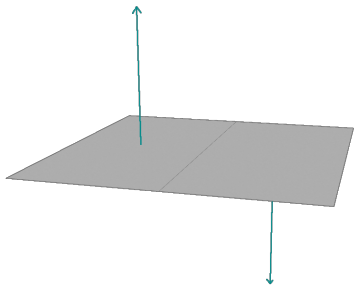
Obr. 3.8: Odpojená hrana a vrchol od plochy

Susedné plochy s otočenými normálami Sú plochy, ktoré ležia pri sebe a sú aj spojené, no ich normály ukazujú opačným smerom.

Problém to je, ako už bolo spomenuté skôr v texte, kvôli shadingu a topológii. Ide, viac menej, o hladký povrch, ktorý z plochy na plochu zmení svoju orientáciu, čo je v reálnom svete nemožné. Preto ani naše algoritmy nevedia, ako plochu zobraziť, či jej vypočítať vyhladzovania povrchu.

Daný problém môže vzniknúť, ak použijeme operáciu „Extrude“ do opačného smeru z hrany, ktorej by sme nemali. Menej často sa stáva, že pre vybrané polygóny, omylom, klávesovou skratkou normály otočíme, alebo sa pomieša poradie počas exportu.

Riešenie je jednoduché, plochy s otočenými normálami opravíme už spomínaným príkazom „Flip“ alebo príkazmi „Recalculate Inside/Outside“.



Obr. 3.9: Susedné plochy s otočenými normálami

3.3.2 N-Gons

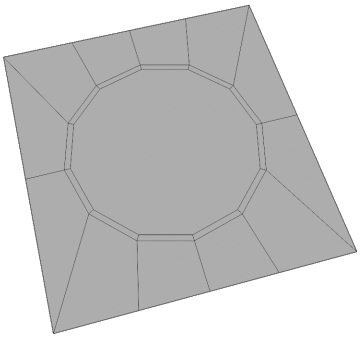
Sú polygóny obsahujúce väčší počet, než 4 okolité hrany. Inak nazývané n-uholníky. Vďaka tomu delíme polygóny do troch kategórií, „tris“, „quads“ a „N-gons“. Niektoré texty zaraďujú trojuholníky k mnohouholníkom, čomu sa my vyhneme. Aj napriek tomu, že používame „quad workflow“, kde všetko, čo nie je štvoruholník, by sa malo považovať za n-uholník. Rozdeľujeme to tak z dôvodu, že výskyt trojuholníkov nie je až tak zlý, ako výskyt n-uholníkov [34].

Prečo je ale výskyt n-uholníkov zlý? Ako sme už spomínali v predošlých sekciách, počas procesu renderovania sa každý polygón rozdelí na trojuholníky. Trojuholník takto deliť nemusíme, štvoruholník je možné rozdeliť len dvoma spôsobmi, no n-uholník sa dá rozdeliť mnohými spôsobmi.

Každé takéto automatické delenie môže vytvoriť trojuholníky inak, samotný Blender obsahuje 6 spôsobov. Štyri z nich sú určené pre štvoruholníky (Beauty, Fixed, Fixed Alternate, Shortest Diagonal) a 2 pre n-uholníky (Beauty, Clip).

Existujú formáty aj programy také, ktoré exportovanie a importovanie n-uholníkov ani nezvládnu. Medzi takéto formáty patrí .obj a software ZBrush. Ten pri importovaní nahlási chybu s možnosťou automatickej triangulácie.

Prečo je teda používanie n-uholníkov a automatické delenie na trojuholníky zlé/nedostatočné? Medzi dôvody, okrem už spomínanej kompatibility, patria:



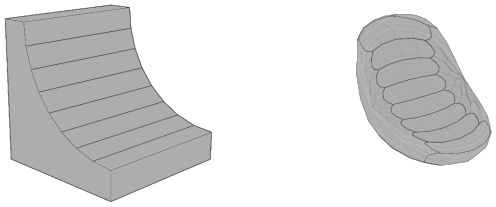
Obr. 3.10: N-uholník v strede štvorcovej mriežky

Rendering a Shading Problémy sú primárne vytvorené na zakrivených povrchoch. Ak sú povrchy rovné a okrajové hrany mnohouholníka označené ako „Sharp“, tak všetky normály vzniknutých trojuholníkov z n-uholníka majú rovnaký smer. Takéto n-uholníky problém pri vykresľovaní nevytvárajú. Ak sú ale okrajové hrany neoznačené, normály sa pokúsia interpolovať. To spôsobí rozdielne normály na okrajových hranách, kde spoločne s náhodnou triangulóciou vzniknú shading problémy.

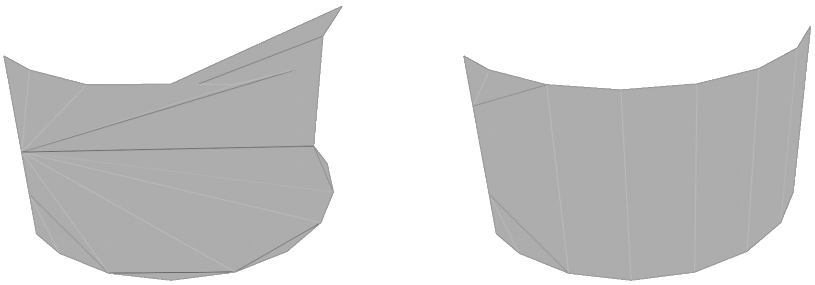
Pri zakrivených povrchoch tieto problémy vznikajú takmer okamžite, pretože rôzne trojuholníky s odlišnými normálami nevedia, ako správne „plochý“ mnohouholník zakriviť.

Topológia Mnohouholníky úplne ničia fungovanie „Edge Loops“, pretože hrany nevedia, ako cez ne pokračovať. Taktiež narúšajú funkčnosť algoritmu „subdivision surface“, a to aj napriek tomu, že z nich vždy urobí geometriu zloženú len zo štvoruholníkov. Samotná funkčnosť, teda, nie je narušená, no výsledok je. Namiesto zaoblených povrchov vznikajú rôzne artefakty ako natiahnuté hrany, prekrývajúce sa polygóny alebo úplná zmena siluety geometrie.

Deformácie Ako už vieme, mnohouholník je rovinný útvar, takže jeho ohnutie je nemožné. Avšak, každý mnohouholník sa mení na trojuholníkovú sieť, ktorá sa už ohnúť dá. A to je zdrojom problémov pri deformáciách, pretože sa pokúšame deformovať niečo, čo toho nie je schopné. Môže sa ale v našom aktuálnom software tváriť, že je. Pri exporte môže vzniknúť úplne nová sieť trojuholníkov, ktoré nami vytvorené zakrivenie nebude podporovať. Výsledkom toho sú artefakty, shading problémy a prelínajúca sa geometria.



Obr. 3.11: Zdeformovaný n-uholník po Sub-D



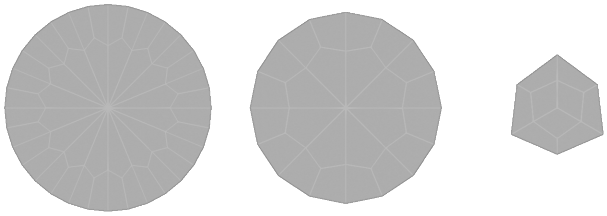
Obr. 3.12: Clip a beauty triangulácia

3.3.3 Poles

Inak nazývané póly alebo hviezdy, sú vrcholy, ku ktorým je napojený iný počet hrán, než štyri. Je ale potrebné si uvedomiť, že póly, s 3 alebo 5 hranami sú v modeloch vytvorených technikou „quads only“ nevyhnuteľné.

Póly o veľkosti 1 a 2, sú buď len čisté hrany alebo okraj plošného objektu, tieto patria do non-manifold sekcie.

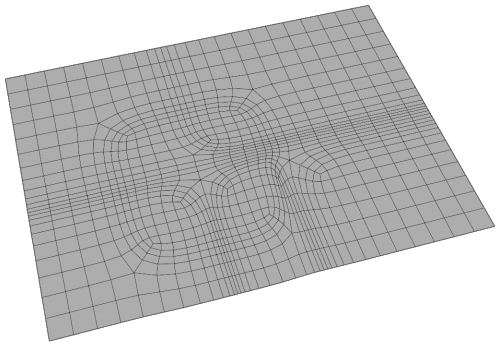
Naopak, všetky póly, ktoré majú viac ako 5 hrán, sa dajú redukovať na rozmedzie 3 až 5. Prečo nám ale póly vadia? Ide znova o „smoothing“, shading a „Sub-D“ workflow. Dané póly na jednom mieste zhromažďujú priveľa geometrie a ukončujú „edge loops“. Ak sa na túto geometriu použije akýkoľvek vy-



Obr. 3.13: Ukážka pólov

hladzovací/deliaci algoritmus, hustota dvojnásobne narastie a vzniknú miesta, kde budú vznikať artefakty známe ako „pitching“. Je to ťahanie geometrie k jednému bodu. To spôsobuje nepravidelný shading a môže vytvárať nečakanú geometriu pri použití „Sub-D“ algoritmu.

Takéto póly priamo v Blenderi detekovať nevieme. Môžeme ale označiť vrchol, ku ktorému sú napojené 4 hrany a použiť operáciu „Select Similiar/Not Equal“. Tá nám označí všetky póly, nevýhodou je, že ostáva označený aj pôvodný vrchol, ktorý nie je pól.



Obr. 3.14: Zhustenie geometrie kvôli pólom

Kapitola 4

Analýza Algoritmov na Opravu a Validáciu Geometrie

V tejto kapitole si zopakujeme niektoré chyby vznikajúce v meshoch 3D modelu z predošlej kapitoly (rozdelíme ich na lokálne, globálne a geometrické) a pozrieme sa na konkrétne algoritmy, ktoré riešia opravu a validáciu meshov. Taktiež sa pozrieme na potreby upstream a downstream aplikácií. Celá kapitola bude parafrázovať a prekladať obsah článku [35].

Algoritmy slúžiace k validácií problémov v meshi majú za úlohu problémy nájsť, bez nutnosti ich riešenia. Na druhej strane, algoritmy opravy majú za úlohu problémy nájsť a aj opraviť. Z toho dôvodu budeme v tomto texte považovať algoritmy na opravu aj za algoritmy na validáciu a venovať sa práve im.

4.1 Rozdelenie problémov

4.1.1 Lokálna spojitosť

Ide o skupinu problémov, ktorá sa zaoberá len spojitosťou medzi prvkami modelu. Patria sem zväčša problémy, ktoré sú v predošlej kapitole pod názvom non-manifold, konkrétne ide o:

Izolované vrcholy V predošlej kapitole boli izolované vrcholy označené ako odpojená geometria. Je to vrchol alebo iný element, ktorý nie je spojený so zvyškom.

Visiace hrany Nazývané aj holé hrany, sú hrany, ku ktorým nie je pripojená žiadna plocha, no sú spojené so zvyškom geometrie. Môžeme ich odstrániť, no niekedy môžu niesť zaujímavú alebo dôležitú informáciu o geometrii.

Komplexné hrany Taktiež označované ako singulárne alebo non-manifold hrany, sú hrany spájajúce viac ako 2 plochy. Ide o častý problém, bez jasného riešenia.

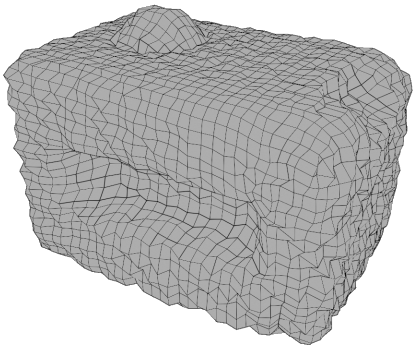
Singulárny vrchol Ide o vrchol spájajúci oddelené povrchy z minulej kapitoly.

4.1.2 Globálna topológia

Do tejto kategórie spadá topologická charakteristika celého povrchu meshu 3D modelu.

Orientácia normál Ide o problém z minulej kapitoly označený ako otočené normály, kde plochy meshu môžu mať odlišné orientácie a tie spôsobujú problémy pri zobrazovaní.

Topologický šum Je problém nepravidelnej topológie s vysokým šumom v geometrických dátach. Tento problém najčastejšie vzniká pri prevode obrázkov alebo point-cloud dát na polygónovú sieť.



Obr. 4.1: Šum

4.1.3 Geometrické problémy

K tejto rodine problémov patria tie, ktoré vieme charakterizovať polohou ich vrcholov a je potrebné ju riešiť v spojitom karteziánskom priestore.

Diery a medzery V predošlej kapitole pod menom diery, sú problémy, kedy v geometrii chýbajú polygóny, čím vznikne diera. Diery a medzery sa od seba odlišujú podľa ich veľkostí. Medzeru definujeme ako prázdnu plochu medzi dvoma stranami, ktoré majú na seba naväzovať. Diera

je všeobecne väčšia, kde chýba celá časť geometrie a naväzovať nemusí. Čiže medzera spája dve oddelené postupnosti hrán, zatiaľ čo diera vytvára okraj jedného cyklu. Diery samotné delíme na malé, stredné a veľké. Algoritmy určené na ich opravu označujeme ako „vypĺňanie dier“ a „uzavretie medzier“.

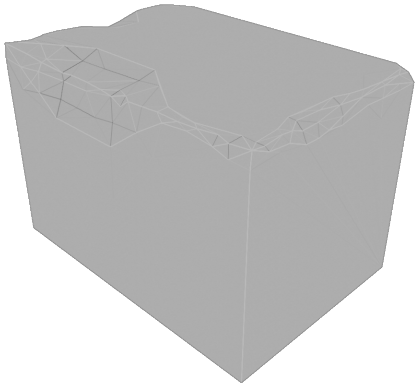
Ide o najviac riešenú tému v rámci opravy modelov, preto obsahuje mnoho algoritmov od najviac primitívnych, kde sa len snažia vyplniť dieru pomocou okrajov, až po algoritmy skúmajúce správanie vrcholov v iných podobných častiach a pomocou spojitosti sa ich snažia vypĺňať. Najčastejšie tieto algoritmy tvoria trojuholníkové siete.

Vypĺňanie dier komplikujú takzvané ostrovčeky polygónov vo vnútri dier, tie sa niekedy obtiažne napájajú, a preto je ich lepšie niekedy odstrániť a vyplniť. Toto rozhodnutie sa ale necháva na užívateľoch, taktiež ako niektoré možnosti, ktoré diery vyplniť, čo ešte považovať za medzery a ako ich vyplniť.

Degenerované elementy Najčastejšie ide o trojuholníky, ale aj hrany, či celé povrchy. Sú to elementy, ktoré majú nulovú plochu, dĺžku alebo objem. Ak hovoríme o takmer zdegenerovaných elementoch, majú skoro nulové hodnoty, no nie rovné nule. To predstavuje problém, pretože pre tieto elementy nevieme vypočítať normály alebo barycentrické koordináty. Dané elementy môžeme buď odstrániť, zjednotiť do samotného vrcholu alebo použiť „remeshing“ algoritmus, ak náš mesh potrebuje rovnomernú geometriu.

Samo-Prieniky Časti meshu, ktoré cez seba navzájom prechádzajú. Problém vzniká, pretože pracujeme s povrchmi a tie prieniky neobsahujú, taktiež môžu vznikať problémy pri renderovaní istých materiálov, alebo pri viditeľnosti prekrývajúcich sa častiach objektu. Nájsť takéto časti meshu je pomerne jednoduché, no riešenie je na používateľovi. Môže ísť totiž o chybu, kde stačí presunúť vrcholy, môže ísť o zlé animácie, kde je potrebné prerobiť váhy vrcholov a môže ísť o celkovú chybu v modelovaní, kde je potrebné prerobiť celý objekt.

Chamfers Inak nazývané bevels alebo „zaoblené okraje“, sú miesta, kde na ostrých hranách sú pridané pomocné okrajové hrany v presnej vzdialenosti, aby ich zaoblili, čím sa dosiahne väčšej reality. Niekedy musia byť tieto chamfers v dokonalých vzdialenostiach, čo sa pri skenovaní často krát nepodarí a vznikne nepravdivý výzor. Naopak, niekedy sa pri skenovaní chamfers môžu pridať na ostré miesta, čo taktiež pokazí stavbu meshu.



Obr. 4.2: Nesprávne vytvorené okraje

4.2 Rozdelenie Aplikácií

Ako sme už spomínali, rôzne aplikácie požadujú rôzne stavy geometrie, pre ich správnu funkčnosť. Z toho dôvodu sa delia na dve kategórie.

4.2.1 Upstream aplikácie

Sú aplikácie takzvané zdrojové, čiže tie, v ktorých model vzniká, alebo je generovaný. Chyby v modeloch vznikajú z dvojakých dôvodov, môže ísť o:

Charakter Mesh môže vzniknúť jeho digitálnym vymodelovaním alebo naskenovaním reálneho objektu. Modelovanie najčastejšie vytvorí problémy ako non-manifold alebo prieniky geometrie. Naskenované objekty majú najčastejšie problém s dierami, topologickým šumom alebo so zaoblenými hranami.

Konverzia Ak náš model nie je tvorený z polygónov, je potrebné ho na ne previesť. Vznik môže byť z kriviek, point cloudu alebo iných techník popísaných v tabuľke. Najčastejšie problémy, ktoré vzniknú sú prieniky alebo medzery.

4.2.2 Downstream aplikácie

Sú aplikácie, ktoré prijímajú mesh, či už na ďalšiu úpravu, alebo len vyzobrazenie. Vytvoriť rozdelenie pre každú aplikáciu je nemožné, preto použijeme niektoré dôvody modelovania z kapitoly 2, ako napríklad vizualizácia či simulácie. V tabuľke budú následné dôvody a problémy zaškrtnuté podľa toho, ktoré sa nehodia.

Tabuľka 4.1: Charakter upstream dát. X značí typické a x menej typické problémy

Charakter	diery	medzery	prieniky	degenerácie	singuláry	šum	chamfers
Digitálne	X					X	X
Reálne		X	X	x	X		

Tabuľka 4.2: Konverzia upstream dát. X značí typické a x menej typické problémy

Konverzia	diery	medzery	prieniky	degenerácie	singuláry	šum	chamfers
Teselácia		X	X	x			
Zjednotenie hĺbkových textúr			X	x	x		
Kontúra rastrových dát					X		
Kontúrovanie implicitných funkcií				X		x	X
Rekonštrukcia z bodov	x	x				x	x
Triangulácia výškových máp							
Získanie povrchu zo solid modelu					X		

Tabuľka 4.3: Dôvody modelovania pri downstream dátach. X značí typické potrebné opravy a x menej typické

Dôvody	diery	medzery	prieniky	degenerácie	singuláry	šum	chamfers
Vizualizácia	X	x					x
Modelovanie	X	X		X	x	x	
Rýchle prototypovanie / Blockout	X	X	X		X		
Processing	X	X	x	X	X	x	x
Simulácie	X	X	X	X	X	X	x

4.3 Algoritmy

V tejto sekcii sa pozrieme na zhotovené zhrnutie v článku [35] a uvedieme si navrhnuté metódy opráv meshu 3d modelu. V daných metódach si prejdeme „lokálny“ a „globálny“ prístup opravy. Lokálne metódy opravujú len zasiahnutú časť geometrie a zvyšok ignorujú. Ich výhodou je, že nemusia spracovať viac dát a neovplyvnia to, čo nemajú. Dôsledkom toho však je nie vždy presná oprava v závislosti na celý mesh. Globálne opravy zväčšia používajú rôzne „remeshovacie“ algoritmy, ktoré zmenia celý model. Výhodou je komplexnejšia oprava, nie len za pomoci polygónov počas procesu analýzy alebo opravy, a možný lepší koherentnejší výsledok so zvyškom meshu. Nevýhodou je, že sa často krát upravujú aj časti meshu, kde chyba nebola a môže vzniknúť nová.

V nasledujúcich častiach si prejdeme zhrnutie daných algoritmov, ktoré budú uvedené v tabuľkách ako možnosti. Nepôjde o výber ideálneho algoritmu, ani o ich detailné vysvetlenie či zhrnutie všetkých algoritmov.

4.3.1 Lokálny prístup

Lokálny prístup je vhodné použiť, ak mesh obsahuje menej roztrúsených problémov na jej povrchu. Dané algoritmy problémy nájdu a pokúsia sa ho opraviť bez toho, aby zmenili zvyšok geometrie.

4.3.1.1 Manifoldná spojitosť

Meshe, ktoré sú non-manifold, obsahujú komplexné hrany či singulárne vrcholy a je možné ich kategorizovať do dvoch skupín. Tie, ktoré sú viazané tzv. regulárnymi množinami a tie, čo nie sú. Skupina viazaná regulárnymi množinami aj naďalej predstavuje správne definovaný objem objektu až na miesta, v ktorých je objekt non-manifold.

Takéto meshe algoritmus rozdelí na viacero regulárnych elementov, ktoré budú manifoldné, podľa algoritmov uvedených nižšie. Vo všeobecnosti, kde nie je možné použiť takéto jednoduché algoritmy, sa používajú globálne remeshovacie algoritmy.

Algoritmy, uvedené v Guéziec et al. 2001 a Rossignac and Cardoze 1999, konvertujú non-manifoldné množiny na viacero manifoldných podmnožín meshov. Dané algoritmy môžu, ale vyprodukovať mesh so samo-prienikmi. Tie nespĺňajú požiadavky manifoldnosti. Komplexné hrany s $2k$ plochami rozdelí na k hrán, kde každá bude mať presne dve susedné plochy. Singulárne vrcholy sa zduplikujú a rozdelia.

Stratégiou, uvedenou v Rossignac and Cardoze 1999, je použiť čo najmenej duplikovaní, Guéziec et al. 2001 pridal operácie zjednotenia okrajových hrán meshu odrezaných pozdĺž komplexných hrán, buď uzatvorením cyklov pomocou pinchingu alebo zlepením okrajov pomocou snappingu.

Tabuľka 4.4: Tabuľka zhrňujúca algoritmy určené na opravu manifoldu

Algoritmus	opravené problémy	požiadavky na vstup	optimálne
Rossignac and Cardoze 1999	kombinatorické singularity	bez okrajov	X
Guéziec et al. 2001	kombinatorické singularity	bez okrajov	—
Attene et al. 2009 (kombinatorické)	kombinatorické singularity	bez okrajov	—
Attene et al. 2009 (geometrické)	kombinatorické a geometrické singularity	bez okrajov, degenerácií, prienikov	—

Nadstavbou týchto algoritmov bol algoritmus snažiaci sa vykonávať tieto opravy vo viacerých dimenziách, s tým, že niektoré problémy neopraví a výsledky budú „kvázi-manifoldné“. Uvedené v De Floriani et al. 2003.

Pre špeciálny prípad troch dimenzií, Attene et al. 2009 upravil predošlý algoritmus a navrhol dva algoritmy, kde jeden zaručí geometrický manifold a druhý kombinačný manifold.

4.3.1.2 Uzatváranie medzier

Medzery sa väčšinou nachádzajú na spojoch dvoch okrajov povrchov, kde nesedí geometria. Z toho dôvodu sú často malé a úzke, pretože ide len o teselačné alebo zaokrúhľovacie chyby. Z tohto dôvodu, väčšina metód uzatvára medzery podľa zarovnania/spojenia ich okrajov a podľa priestorovej vzdialenosti.

Najjednoduchší spôsob uzatvárania medzier je zlučovanie vrcholov, ktoré sú od seba mierne vzdialené do jedného vrcholu. V Blenderi poznáme príkaz „merge by distance“. Túto techniku použil Rock a Wozny 1992.

Druhým prístupom je postupné uzatváranie hrán, ktoré použili Shend a Meier 1995 a aj Barequet a Kumar 1997, nazvali to uzatváranie pomocou „zipsu“. Aby sa vyhli nezrovnalostiam, uzatvárajú hrany s najmenšou vzdialenosťou od seba.

Levoy 1994 vyriešil prípad „negatívnych medzier“, to sú tie, ktoré sa prekrývajú tak, že ich odrezal a spojil.

Vďaka uzatváraniu párov hrán, nevznikajú komplexné hrany. Avšak negatívnym dôsledkom toho je, že niekedy sa hrany nemusia uzavrieť a môže vzniknúť non-manifold geometria. Borodin et al. 2002 preto rozhodol, že povolí spájanie hrán tak, aby mohli spájať viac ako dve susedné plochy.

Patel et al. 2005 navyše pridal rozlišovanie medzi druhmi medzier. Ak išlo o veľmi tenké, použil na nich metódu zipsu. Naopak ak boli medzery väčšie, pridával novú trojuholníkovú geometriu, aby príliš nenarušil pôvodnú stavbu objektu.

Pomedzi to sa Barequet a Sharir 1995 pokúšali vytvoriť menej „agresívnu“ metódu. Ako prvý krok sa globálne vyhľadajú okrajové krivky, na ktorých sa

Tabuľka 4.5: Tabuľka zhrňujúca algoritmy určené na uzatváranie medzier a ich požiadavky na vstup, parametre a možné problémy, ktoré vytvoria

Algoritmus	požiadavky na vstup	parametre	možné nové problémy
Rock and Wozny 1992	veľmi malé medzery	šírka medzery	prieniky, degenerácie, singularity
Sheng and Meier 1995	—	šírka medzery	prieniky, degenerácie
Barequet and Kumar 1997	—	šírka medzery	prieniky, degenerácie
Turk and Levoy 1994	prekrývajúce sa časti	prah medzery	prieniky, degenerácie
Borodin et al. 2002	—	—	prieniky, degenerácie, singularity
Patel et al. 2005	—	—	prieniky, degenerácie, singularity
Barequet and Sharir 1995	—	prah medzery	prieniky, degenerácie
Bischoff and Kobbelt 2005	—	šírka medzery, rozlíšenie	degenerácie

neskôr uskutoční spájanie. K nájdeniu vhodných zodpovedajúcich medzier sa používa heuristický algorimus kriviek v 3D použitý na vzorkovanie okrajových hrán.

Vzhľadom na to, že všetky tieto metódy sa spoliehajú na spájanie okrajov viacerých „patchov“, tak medzery medzi vnútrom a okrajom, či medzi dvomi komponentami meshu, nie sú spojené. Existuje viacero medzier, ktoré zatiaľ nedokážeme efektívne spojiť za použitia lokálnych metód, preto sa v týchto prípadoch odporúčajú metódy globálne.

4.3.1.3 Vypĺňanie dier

Na rozdiel od medzier, diery sú väčšinou chýbajúce časti povrchu, preto je potrebné vytvárať novú geometriu, nie ju spájať.

Prvé metódy nachádzali uzavreté cykly okrajových hrán meshu. Následne boli vyplnené trojuholníkmi. Takéto taktiky nazývame „chamtivé“ stratégie založené na jednoduchej heuristike, ktoré využili napríklad Bohn a Wozny 1992, Mäkäla a Dolenc 1993, Varnuska et al. 2005. Taktiež ale môžeme použiť komplikovanejšie metódy náhodnej triangulácie, ktoré avšak potrebujú, aby bola diera prevedená do roviny, Roth a Wibowoo 1997.

Brunton et al. 2010 navrhol simulované žíhanie (annealing), založené na „rozprestieracom“ princípe, ktorý zväčšuje pravdepodobnosť, že sa diera bude dať previesť do roviny. Pfeifle a Seider 1996 pridávajú nové vrcholy do dier, aby sa priblížili Delaunay triangulácií.

Predošlé metódy nebrali ohľad na kvalitu geometrie, ktorú vytvorili. Mohlo sa teda stať, že komplexné diery, ťažko preveditelné do roviny, mohli vytvoriť

nežiadúce polygóny. Z toho dôvodu Barequet a Sharir 1995 naimplementovali techniky dynamického programovania tak, aby našli minimálnu plochu dier. Leipa 2003 tento prístup vylepšil pridávaním penált za nespojitosť okolo okrajov dier. Bac et al. 2008 využil týchto zrýchlení a vytvoril vypĺňanie, ktoré sa aj vyhladzuje. Wei et al. 2010 zobral do úvahu aj rôzne vnútorné uhly. Nevýhodou metód dynamického programovania bola jeho zložitosť pri dierach so stovkami hrán. V tých istých rokoch vyšlo mnoho iných algoritmov, ako Radial Basis Function interpolation, NURBS fitting, curvature energy minimalization, Moving Least Squares (MLS) projection.

Neskoršia metóda od Tekumalla a Cohen 2004 sa pokúša vyriešiť problém, že všetky predošlé metódy môžu vytvoriť samo-prieniky. Nové trojuholníky sa testujú už len s existujúcim meshom, kde sa niekedy môžeme dostať do deadlocku a dieru neuzavrieť. Z toho dôvodu Wagner et al. 2003 pridal náhodnú optimalizačnú techniku (simulované žíhanie), ktorá navyše dokázala trojuholníky odstrániť, aby sme mohli uniknúť deadlockom.

Popísané metódy vypĺňajú diery pomocou tzv. „disc-topology“. Neberú v úvahu vzťahy okrajových hrán, ani vnútorných „ostrovčekov“, aby vytvorili vhodnejšiu topológiu.

Podolak a Rusinkiewicz 2005 preto pristupujú k problému globálne, aby ho opravili lokálne. Vstupný mesh je spracovaný v priestore štvorstenmi, ktoré sú zarovnané s pôvodnými plochami. Vďaka tomuto objemovému prístupu je možno pomocou „graph-cut“ technník dosiahnuť zaplnenie komplexných dier a vyhnutiu sa samo-prienikov. Avšak vstup nesmie mať žiadne samo-prieniky, singularity ani degenerácie, aby bol zaručený manifold výstup.

Zvyšné metódy budú popísané v nasledujúcej sekcii venujúcej sa globálnym problémom.

4.3.1.4 Dokončenie meshu

Predošlé metódy na uzatváranie dier vytvárali vo výsledku hladké povrchy, buď za použitia interpolácie alebo vyhladzovacieho algoritmu. Vytvoriť ale hladký povrch pre mesh, ktorý nie je hladký, vyzerá nereálne. Z tohto dôvodu vzniklo viacero algoritmov na dokončovanie meshov. Tie vyplnia diery dôveryhodnejšie, pretože sa snažia imitovať okolie toho, čo sa pokúšajú zaplniť.

Jedna špeciálna trieda metód problémy vôbec neopravuje, ale nahradí celý vstupný mesh za nový. Ide hlavne o výmenu hlavy (Blanz a Vetter 1999, Kähler et al. 2002, Blanz et al. 2004), tela (Allen et al. 2003, Angueolv et al. 2005) alebo zubov (Kähler et al. 2002, Savchenko a Kojekine 2002). Vo všeobecnosti to je však veľmi malá trieda objektov, ktoré sú vždy rovnaké, iba sa mierne líšia tvarom. Väčšina potrebuje aj nejakú formu informácie, ako objekt správne „prelepiť“. V Blenderi si to môžeme predstaviť ako použitie shrinkwrap modifier.

Tabuľka 4.6: Tabuľka zhrňujúca algoritmy a metódy určené na vyplnenie dier s ich požiadavkami na vstup, parametrami a garanciou výstupu, ktorý nebude generovať prieniky

Algoritmus	požiadavky na vstup	parametre	bez prienikov
Bohn and Wozny 1992	—	—	—
Mäakelä and Dolenc 1993	—	—	—
Varnuska et al. 2005	—	—	—
Roth and Wibowoo 1997	pomerne rovinné okraje dier	—	—
Brunton et al. 2010	jednoduché okrajové cykly	—	—
Pfeifle and Seidel 1996	—	—	—
Barequet and Sharir 1995	—	—	—
Liepa 2003	—	—	—
Zhao et al. 2007	—	—	—
Branch et al. 2006	—	—	—
Lévy 2003	—	—	—
Pernot et al. 2006	—	—	—
Wang and Oliveira 2007	pomerne rovinné okraje dier	MLS polomer	—
Tekumalla and Cohen 2004	—	MLS polomer	X
Wagner et al. 2003	—	simulované žíhanie	X
Podolak and Rusinkiewicz 2005	bez degenerácii, prienikov, singularít	—	X

Iný prístup je ponechanie pôvodného meshu a použitie len záplaty v mieste diery. Je to však náročný proces pracujúci s bodovými-vzorkami a nie s topológiou (Shard et al. 2004, Bendels et al. 2005, Breckon a Fisher 2005, Part et al. 2006, Xiao et al. 2007). Takže nie je priamo použiteľný ako oprava meshu 3D modelu. Je potrebné použiť remeshing algoritmus, preto môžeme považovať túto techniku ako mierne „globálnu“. Avšak teoreticky môžeme nahradiť len chýbajúcu časť a zmeniť teseláciu len v jej blízkosti, preto ju kategorizujeme do „lokálnych“ problémov.

Neskôr bola navrhnutá metóda Sharf et al. 2004, ktorá je inšpirovaná kontextovým dopĺňaním 2D obrázkov. Čiže v geometrií sa nájdu podobné časti a nimi sa pokúsi zaplniť diera. Na podobnom princípe s rôznymi obmenami funguje viacero algoritmov.

Kraevoy a Sheffer 2005 predstavili metódu, ktorá používa template mesh, ktorý je dosadený buď globálne alebo lokálne, na neúplnú vstupnú geometriu, aby z neho použili útržky. Je však potrebná korekcia od užívateľa, aby sme

Tabuľka 4.7: Tabuľka obsahuje algoritmy na dokončovanie geometrie s ich požiadavkami na vstup, parametrami a uvádza problémy, ktoré môžu danou metódou vzniknúť

Algoritmus	požiadavky na vstup	parametre	možné nové problémy
Sharf et al. 2004	— (bodový-základ)	rozlíšenie	šum
Bendels et al. 2005	— (bodový-základ)	počet levelov	šum
Breckon and Fisher 2005	— (bodový-základ)	veľkosť okna	šum
Park et al. 2006	— (bodový-základ)	rozlíšenie	šum
Xiao et al. 2007	— (bodový-základ)	niekoľko	šum
Nguyen et al. 2005	pomerne rovinné okraje dier	približný prah, počet levelov	degenerácie, prieniky
Xu et al. 2006	pomerne rovinné okraje dier	obrázky zarovnané s modelom	degenerácie, prieniky
Jia and Tang 2004	pomerne rovinné okraje dier	tensor voľby	degenerácie, prieniky
Kraevoy and Sheffer 2005	manifold s okrajom	predlohy a korešpondencie	prieniky
Pauly et al. 2005	—	databáza modelov, kľúčové slová	degenerácie, prieniky

získali správnu základnú geometriu. Pauly et al. 2005 využil viacerých modelov a vytvoril databázu, kde podľa kľúčových slov zadaných používateľom je vybraná najlepšia možná kombinácia na zaplnenie chybnej geometrie.

4.3.1.5 Odstránenie degenerácií

Botsch a Kobbelt 2001 použili odrezávaciu techniku na detekciu a odstránenie degenerovaných trojuholníkov z manifold meshu. Daný algoritmus dokáže odstrániť degenerované plochy vytvorené v CAD programoch. Rizikom je, že degenerácie spôsobujú numerické problémy v algoritmoch, preto používa robustné predikáty na rozdelenie plôch s kontrolovaným prístupom.

Attene 2010 používa na odstraňovanie degenerácií kombináciu zmenšenia hrán a prehodení.

Oba tieto algoritmy určujú degenerácie podľa uhlov v trojuholníku. Ak je uhol príliš ostrý ($\leq \epsilon$) alebo príliš tupý ($\geq \epsilon$), je trojuholník označený za degenerovaný, kde ϵ je parameter. Ak je parameter rovný 0, ide o priamo degenerované plochy, ak je malý, ide o tzv. takmer-degenerované plochy. Oba algoritmy taktiež zmenšia kratšiu hranu, čím sa odstránia tenké trojuholníky.

Ani jeden z algoritmov však negarantuje kompletne odstránenie degenerácií a oba môžu vytvoriť non-manifold geometriu.

Tabuľka 4.8: Tabuľka zhrňujúca metódy riešenia problémov s degeneráciami a samo-prienikmi. Udáva, čo dokáže aká metoda opraviť, požiadavky na vstup, parametre, garanciu úspechu a presnosť

Algoritmus	opravy	požiadavky na vstup	parametre	správnosť	presnosť
Botsch and Kobbelt 2001	degenerácie	manifold	uhol	—	odhad
Attene 2010	degenerácie, samo-prieniky, diery	—	uhol	—	odhad
Bischoff and Kobbelt 2005	samo-prieniky, medzery	manifold	tolerancia, šírka medzery	X	odhad
Campen and Kobbelt 2010	samo-prieniky	bez okrajov, degenerácii	—	X	presný
Granados et al. 2003	samo-prieniky	—	—	X	presný

4.3.1.6 Odstránenie samo-prienikov

Bischoff a Kobelt 2001 navrhli metódu, ktorá odstránia prieniky medzi dvoma časťami teselovaného CAD modelu pomocou voxel-mriežky (alebo spatial subdivision). To zaručuje rýchlosť a presnosť, pretože sa opravuje len lokálne v miestach, kde nastala chyba. ϵ_0 udáva vzdialenosť, v ktorej sa vykoná oprava od zasiahnutého miesta. Algoritmus taktiež spojí medzery, ktoré mohli vzniknúť podľa parametru γ_0, určujúceho šírku medzery nastavenú užívateľom.

Attene 2010 navrhol postup, ktorý používa viacero opravovacích algoritmov. Postup bol navrhnutý na opravu neupravených naskenovaných dát, bez požiadavok na vstup. Metóda prevedie vstup na orientovaný manifold, uzavrie diery a odstráni degenerované plochy. Na opravu prienikov taktiež používa „spatial subdivision“ ako predošlý algoritmus s tým rozdielom, že z princípu naskenovaných dát sú trojuholníky malé, takže prieniky neopravuje, ale odstráni zasiahnuté trojuholníky a následne ich vyplní ako diery. Tento prístup však negarantuje správny výsledok vždy, ale len v istých prípadoch.

Riešenie prienikov vie byť náročné z dôvodu obmedzenej presnosti geometrických výpočtov. Campen a Kobbelt 2010 tento problém vyriešili konvertovaním meshu do rovinnej BSP reprezentácie v medzikroku. Takže prieniky nie je potrebné riešiť až kým nie je potrebný výsledok. Ak je pri vstupe poskytnutá presná číselná precíznosť, algoritmus garantuje úspešnosť s rýchlosťou a nízkou náročnosťou na pamäť. Na vstupe je však potrebné mať uzavretú mesh bez degenerácií.

Opačný prístup poskytol Granados et al. 2003, kde nie je potrebná zadaná precíznosť. Vďaka tomuto sa vyhneme problému, kde je na vstupe potrebná mesh bez otvorov alebo visiacich elementov. Avšak nevýhodou je rýchlosť a požiadavky na pamäť.

Tabuľka 4.9: Tabuľka algoritmov, ktoré sa pokúšajú obnoviť skorumpované ostré hrany meshu. Tabuľka obsahuje požiadavky na vstup, parametre a možné nové problémy

Algoritmus	požiadavky na vstup	parametre	možné nové problémy
Kobbelt and Botsch 2003	manifold	interaktívne	samo-prieniky
Attene et al. 2005	manifold, bez degenerácií	—	samo-prieniky, degenerácie
Chen and Cheng 2008	manifold, bez degenerácií	—	samo-prieniky
Wang 2006	bez šumu, degenerácií	dva prahy	samo-prieniky

4.3.1.7 Obnova ostrých hrán

Kobbelt a Botsch 2003 prišli s interaktívnym prístupom obnovovania skorumpovaných ostrých hrán. Používateľ má za úlohu vytvoriť niekoľko „fish-bone structures“, ktoré sú automaticky zteselované a nahradia pôvodné chamfers. Tento postup síce nie je automatický, no dovoľuje interaktívnu prácu s modelom a následnú zmenu veľkosti chamfers, z toho dôvodu je ale odporúčané metódu použiť len pre menšie modely.

Attene et al. 2005 prišiel s EdgeSharpener metódou, ktorá automaticky detekuje a zaostrí správne hrany. Funguje na základe kontrolovania „dihedral“ uhla medzi hladkými regiónmi v mieste hrán, trojuholníky nachádzajúce sa medzi týmito plochami považujeme za aliasing artefakty. EdgeSharpener odhadne originálne ostré hrany pomocou dvoch rovín, a potom rozdelí pôvodné hrany a trojuholníky, kde z novo vzniknutých vrcholov zostaví ostré hrany.

Chen a Cheng 2008 sa zaoberajú problémom obnovy ostrých hrán vo vnútri hladkých častí. Metóda funguje iteratívne, kde sa v každom kroku upravujú normály plôch a skorumpovaná ostrá hrana nemusí byť reprezentovaná len jedným pruhom trojuholníkov ako v predošlom algoritme.

Wang 2006 prišiel s návrhom, kde povolí vkladať vnútorné vrcholy do chamfers. Nanešťastie, táto metóda nedokáže automaticky zistiť, či ide o ostrú hranu, alebo len mixovanie dvoch plôch, preto je potrebný vstup od používateľa.

Nevýhodou všetkých spomínaných metód je, že pridávajú alebo upravujú trojuholníky a existujúcu geometriu tak, že je pravdepodobný vznik samoprienikov.

Tabuľka 4.10: Tabuľka zhrňujúca algoritmy na odstránenie šumu z geometrie. Obsahuje algoritmy, ich možné opravy, požiadavky na vstup a parametre

Algoritmus	opravy	požiadavky na vstup	parametre
Taubin 1995	šum	uzavretý manifold	λ, μ, n
Fleishman et al. 2003	šum, ponechanie prvkov	manifold	σ_c, σ_s
Jones et al. 2003	šum, ponechanie prvkov	—	$\sigma_{šum}$
Hildebrandt and Polthier 2004	šum, zaostrenie prvkov	manifold	λ, r
Fan et al. 2010	šum, zaostrenie prvkov	manifold	n

4.3.1.8 Odstránenie šumu z geometrie

Odstraňovanie šumu je široká kategória rôznych metód, ktoré nie sme schopní v tejto podsekcii prebrať. Z toho dôvodu sa pozrieme len na niektoré z metód.

Najjednoduchšia metóda odstránenia šumu je použitie tzv. „Laplacian“ vyhladzovania. To funguje iteratívne, kde v každom kroku posunie vrchol bližšie k stredu ťažiska okolitých vrcholov. Daný postup vyhladí celú mesh, no zároveň ju zmenší. Taubin 1995 používa $\lambda|\mu$ algoritmus, ktorý mení smery medzi zmenšovaním a zväčšovaním. Vollmer et al. 1999 naopak po vyhladení posunie vrcholy zas smerom, z ktorého prišli.

Daný algoritmus ale vyhladí všetko, bez ohľadu na to, či ide o reálny šum, alebo len formu geometrie. Taktiež vyhladí aj ostré hrany alebo chamfers. Fleishman et al. 2003 vyriešili tento problém adaptovaním bilaterálnym filtrom, používaným v 2D grafike, aby fungoval na meshe, čím vznikol prvý vyhladzovací algoritmus, ktorý ponechával niektoré príznaky. Na vstupe je počet iterácií a dva parametre, aby bol výsledok podľa predstáv používateľa.

Jones et al. 2003 prišiel s iným prístupom, ako dosiahnuť algoritmu, ktorý vyhladí len isté časti s tým, že nie je iteratívny a nepotrebuje žiadne požiadavky na vstupnú geometriu.

Hildebrandt a Polthier 2004 prišli na metódu založenú na „anisotropic mean curvature flow filtering“, ktorá dokáže odstrániť šum a zaostriť ostré hrany. Sun et al. 2007 tento algoritmus mierne zlepšil.

Fan et al. 2010 prišiel s metódou, kde predpokladá, že povrch je vymodelovaný z viacerých šumivých častí, ktoré by mali byť hladké. Vďaka tomu na ich prienikoch dokáže ponechať ostré hrany a v iných miestach, kde sa nachádza šum, ho vyhladiť.

Treba však brať v úvahu, že všetky tieto algoritmy ľubovoľne posúvajú vrcholy a nezaručujú, že nevytvoria samo-prieniky alebo degenerované elementy.

4.3.1.9 Odstránenie topologického šumu

Oprava topológie je mnoho krát potrebná, aby nevznikali rôzne defekty ako tunely či „handles“ (od teraz budeme hovoriť o rukoväti). Pri opravách je možné využiť znalosti predošlej topológie, no v istých prípadoch to vie výrazne zvýšiť komplexnosť algoritmov. Taktiež je ideálne, ak by nová topológia bola riadená používateľom, čo však nie je vždy možné.

Na opravu topológie používame dva rôzne prístupy. Jeden z nich pracuje priamo s geometriou, druhý, častejšie používaný, pracuje s voxelovými reprezentáciami. Niektoré algoritmy z prvej kategórie potrebujú ako medzikrok vytvorenie voxelovej mriežky.

Oprava topológie založená na geometrii Cieľom Fishl et al. 2001 bolo odstrániť všetky rukoväte z modelu mozgu za pomoci nafukovania. V striedavých krokoch vyhladzovania a projekcie geometriu na guľu sa rukoväte hľadajú podľa veľkých záhybov na namapovanej geometrii. Geometria okolo každej rukoväte je odstránená a diery sú uzavreté pomocou diskových záplat.

Za použitia „local wave front traversal“ algoritmu navrhnutého Guskov a Wood 2001 sa nájdu malé tunely a rukoväte. V nich sa nájdu možné rezy, ktoré nerozdelia mesh na viacero častí. V daných nových rezoch sa geometria rozdelí a uzavrie v inom mieste. Veľkosť tunelov a rokovätí, ktoré sa uzavrú, sa nastavuje pomocou parametrov.

Efektívnejší spôsob, prispôsobený na opravu naskenovaných objektov, je predstavený v Attene a Falcidieno 2006. Zrýchlenie je vďaka pozorovaniu, že dané objekty majú dĺžky hrán približne rovnaké.

El-Sana a Varshney 1997 navrhli metódu, kde sa po povrchu vstupného meshu bude váľať guľa s polomerom α, ak sa guľa istých častí nedotkne, zaplnia sa. Požiadavka na vstup je, aby bola geometria bez krajov. Nevýhodou tohto algoritmu je, že zjednoduší aj iné časti geometrie, než sme potrebovali a vo väčšine prípadov je takmer nemožný na použitie v praktickom svete.

Všetky spomínané algoritmy vytvárajú novú geometriu, ktorú nekontrolujú, vzniká teda šanca samo-prienikov, tá však nie je vysoká.

Oprava topológie založená na voxeloch Ak je vstup v diskrétnej objemovej reprezentácií, tak algoritmus od Wood et al. 2004 vykoná osovozarovnané preskenovanie objemu, to nájde všetky rukoväte, vypočíta ich veľkosti a výberovo ich odstráni. Odstránenie rukovätí sa udeje ešte vo voxelovom modeli. Metóda funguje za pomoci tzv. „Reeb grafu“.

Metóda navrhnutá Szymczak a Vanderhyde 2003 má rovnaký cieľ, no nepoužíva Reeb graf ale „topology sensitive carving“. Vďaka tomu je

Tabuľka 4.11: Tabuľka zhrňujúca algoritmy určené na opravu topológie

Algoritmus	požiadavky na vstup	parametre	možné nové problémy
El-Sana and Varshney 1997	bez okrajov	polomer	samo-prieniky, chamfers
Guskov and Wood 2001	orientovaný manifold	prah	samo-prieniky
Fischl et al. 2001	orientovaný manifold	—	samo-prieniky
Attene and Falcidieno 2006	—	prah	samo-prieniky
Shattuck and Lehay 2001	bez veľkých dier	—	(chamfers)
Han et al. 2002	bez veľkých dier	—	(chamfers)
Szymczak and Vanderhyde 2003	bez veľkých dier	prah	(chamfers)
Wood et al. 2004	bez veľkých dier	prah	(chamfers)
Zhou et al. 2007	bez veľkých dier	dva prahy	(chamfers)
Ju et al. 2007	bez veľkých dier	cieľový „tvar“	(chamfers)

jednoduchšie spracovávať modely s viacerými rukoväťami s nevýhodou menšej precíznosti.

Metóda od Zhou et al. 2007 používa adaptívnu mriežku, ktorá dokáže spracovávať obrovské modely rýchlo a efektívne.

Algoritmus uvedený od Ju et al. 2007 dovoľuje zjednodušenie nadbytočnej topológie, ale aj dovoľuje úpravu topológie objektu, aby sa rovnal cieľovému tvaru. Takto tento algoritmus dokáže ľahko odstrániť tunely, rukoväte s minimálnym dopadom na vstupnú geometriu, ale ako aj ostatné algoritmy, mierne odchýlenie kvôli voxelovej štruktúre je nevyhnuteľné.

4.3.2 Globálny prístup

Metódy popísané v predošlej podsekcii odstránili vždy len jeden nedostatok ako diery či samo-prieniky. Chyby samé o sebe nevytvárajú problémy, až kým meshe nechceme použiť v iných aplikáciách. Nájsť dané problémy a zistiť, či spĺňajú požiadavky manifoldnosti, nie je až taký problém, ako sa ich pokúsiť opraviť v lokálnej mierke bez ohľadu na celý mesh. Zároveň, takéto lokálne „opravy“ môžu spôsobiť prieniky s inými časťami meshu.

Z toho dôvodu boli vytvorené globálne algoritmy, ktoré budú brať ohľad na mesh celý a jeho vzájomné vzťahy medzi elementmi a nedostatkami. Keďže ani manifoldnosť nie je potrebná sama o sebe, ale slúži len na popis toho, či niečo môžeme vytvoriť, sa často používa medzi krok, kde namiesto „shell“ reprezentácie polygónov použijeme „solid volume“. Tieto volumetrické reprezentácie nám umožňujú vytvoriť širšie/voľnejšie pravidlá a udávajú správnosť

Tabuľka 4.12: Globálne metódy opravy meshu 3D modelu. Tabuľka obsahuje autorov algoritmu, požiadavky na vstup a spôsob, akým algoritmus určuje znamienko toho, čo je vnútro a čo je vonkajšok

Algoritmus	požiadavky na vstup	metóda určenia znamienka
Oomes et al. 1997	bez veľkých dier/medzier	postupné zaplavovanie
Andújar et al. 2002	bez veľkých dier/medzier	postupné zaplavovanie
Curless and Levoy 1996	orientované range meshe	line-of-sight
Furukawa et al. 2007	orientované range meshe	line-of-sight
Davis et al. 2002	orientované	normály a difúzia
Guo et al. 2006	orientované	normály a difúzia
Masuda 2004	orientované	normály a difúzia
Sagawa and Ikeuchi 2008	orientované	normály a minimalizovanie plochy
Shen et al. 2004	orientované	normály a MLS interpolácia
Verdera et al. 2003	orientované	normály a inpainting
Ju 2004	bez veľkých medzier	počty parít
Nooruddin and Turk 2003	—	počty parít a ray-stabbing
Bischoff et al. 2005	—	morfológia a zplavovanie
Hétroy et al. 2011	—	zmenšovanie membrán
Hornung and Kobbelt 2006	—	morfológia a graph-cut
Spillmann et al. 2006	—	počty parít
Murali and Funkhouser 1997	bez veľkých dier	globálna optimalizácia

kontúrovacích metód, ktoré zaručujú správny prechod medzi objemovými reprezentáciami do polygónových. Problémom je teda len určenie, ktorá časť patrí dnu a ktorá von (určenie znamienka). Nevýhodou je však napríklad pri voxelovej reprezentácii možná strata ostrých hrán alebo detailov menších ako voxel. Dané detaily len vytvoria šum na povrchu.

Keďže väčšina z metód opravuje viacero ako len jeden problém, nedelíme ich podľa problému, ktorý opravia, ale podľa požiadavkov na vstup do algoritmu. Mnoho krát požiadavky na vstup nie sú striktné, väčšina algoritmov vždy vytvorí nový manifoldný mesh, no výsledok nemusí byť podľa očakávaní.

4.3.2.1 Vstup bez dier alebo medzier

Prvé metódy len konvertovali vstup na jednu z objemových reprezentácií a následne ju zmenili na polygónovú sieť. Dôležitým krokom je priradenie znamienka, či sa nachádza voxel dnu alebo von. Ak pracujeme s objektmi bez dier (alebo s dierami menšími ako voxel), je možné použiť jednoduché postupné zaplavovanie od elementov, ktorých znamienko poznáme, Oomes et al. 1997. Ak nemáme túto informáciu, môžeme začať od elementu, ktorý sa nachádza

von z ohraničenia vstupu. To nám dá vonkajšiu obálku ako výsledok, všetky vnútorné štruktúry sú zničené, Andújar et al. 2002. Vedľajší efekt diskretizačnej procedúry je potlačenie šumu a detailu vysokej frekvencie. Algoritmus od He et al.1996 explicitne aplikuje aj „low-pass“ filter.

Ak však vstupný mesh obsahuje veľké diery, nie je možné určiť orientáciu znamienok, preto sa používajú niektoré z nasledujúcich metód.

4.3.2.2 Vstup so známou orientáciou

Najčastejšou príčinou, prečo objekt obsahuje diery, je zakrytie skeneru. Väčšina algoritmov je tvorená teda tak, aby vedeli opravovať diery z 3D skenov. Vďaka princípu získavania skenov vieme určiť orientáciu polygónov, čo je využité pri mnohých metódach získavajúcich informácie o znamienku orientácie.

Curless and Levoy 1996 využili „line-of-sight“ informácií skenovacieho procesu a roztriedili voxely na tie, ktoré sú „určite vonku“ alebo „možno dnu“. Toto rozdelenie zaručilo uzavretie všetkých dier, no v regiónoch, kde bolo málo informácií, vznikala nežiadúca geometria. Kvôli zlepšeniu Furukawa et al. 2007 pridal ďalšie rozdelenia a využil aj „line-of-light“ informácií.

Pre dosiahnutie viac vierohodného výsledku, Davis et al. 2002 navrhol proces „diffusion“, alebo aj rozširovania. Vstupný mesh je konvertovaný do znamienkového „distance field“ v blízkosti pôvodného povrchu a pole je následne iteratívne rozširované, čím sa uzatvoria diery do istej vzdialenosti. Guo a Masuda 2006, 2004 vytvorili varianty tejto metódy určené pre nie hladké povrchy.

Tieto metódy iteratívne zmenšujú diery a medzery. Ich okraje rastú a zmenšujú sa nezávisle na sebe, čím sa ľubovoľne zlučujú. Po použití viacerých iterácií sa aj vyhladzujú. Iné metódy reagujú na diery viac globálne, čo pomáha pri ich zaplnení.

Sagawa a Ikenuchi 2003, 2008 rozdelili vonkajšie a vnútorné voxely podľa normálov vstupného meshu. Následne voxely po okrajoch sa iteratívne menili na základe minimalizácie plochy diery.

Shen et al. 2004 popisuje konštrukciu implicitného povrchu na základe pohybu najmenšieho počtu štvorcov vstupného meshu. Diery a medzery do istej šírky sa tak globálne zrastú.

Verdera et al. 2003 používajú parciálne diferenciálne rovnice za účelom hladkej výplne dier. Nevýhodou je, že zadávanie okrajových podmienok pre isté diery je náročné.

4.3.2.3 Ľubovoľný vstup

Ju 2004 popisuje metódu, ako vyplniť diery vo voxelovej verzii vstupného meshu. Výstup je garantovane uzavretý mesh, no kvôli explicitnému vypĺňaniu dier, ktoré majú okrajový cyklus, je pravdepodobné, že medzery, ktorých okraj nie je cyklický, sa nespoja.

Navyše, všetky doterajšie algoritmy sa len pokúšali vyplniť diery a medzery, zatiaľ čo mesh môže obsahovať prebytočnú geometriu, ktorej sa treba zbaviť, aby bol výsledok manifold. Napríklad použitím kitbash metódy môžu vzniknúť dvojité steny či prieniky, s ktorými sa populárne metódy ako „dual contouring“ Mäntylä 1988 nevedia vysporiadať a tvoria non-manifold meshe.

Spôsob, ako sa vyhnúť problémom s vnútornými stenami, je použiť len vonkajšiu obálku modelu a vnútro odignorovať. K tomuto je možné použiť metódu „ray-stabbing“, kde sa posielajú lúče cez model a všetko medzi prvým a posledným dotykom lúču sa berie ako vnútro. Toto samozrejme nebude fungovať pre diery a medzery, z toho dôvodu sa posiela viacero lúčov v rôznych uhloch, Nooruddin and Turk 2003.

Bischoff et al. 2005 taktiež používajú vonkajšiu obálku, no na uzavretie dier a medzier využívajú morfologické operácie, ktoré aplikujú na zvoxelovaný vstup. Vonkajšia obálka sa tak nájde pomocou postupného zaplavovania, ktorému pomáhajú „octree“ stromy pre zväčšenie rýchlosť.

Hétroy et al. 2011 používajú diskrétne membránové zmenšovanie, aby dosiahli vonkajšej obálky. Hornung a Kobbelt 2006 používajú rovnaké postupy, s tým rozdielom, že výsledný povrch nájdu pomocou „graph-cut“ prístupu, ktorý dostáva zaplnenie dier s najmenšou možnou plochou.

Všetky predošlé metódy uzavrú diery, medzery a zbavia sa nadbytočnej geometrie, no odstránia aj vnútorné diery, ktoré môžu byť úmyselné, preto boli vytvorené metódy na základe počtu parít, ktoré sú schopné tieto vnútorné časti ponechať, Nooruddin and Turk 2003, Spillmann et al. 2006

Úplne rozdielny prístup používajú Murali a Funkhouser 1997, originálnu geometriu nemenia v problémových častiach, no môžu mierne zmeniť jej teseláciu. Používajú objemové rozdelenie pomocou ľubovoľných mnohostenov, ktoré sú zarovnané k pôvodným polygónom za pomoci BSP-stromu, namiesto klasických voxelov. Vďaka tomu dokážu vyriešiť všetky problémy spomínané vyššie, no táto metóda je silne závislá na BSP strome a mesh obsahuje priveľké diery, výsledok nemusí byť správny.

Kapitola 5

Návrh a Implementácia

V tejto kapitole sa pozrieme na funkčné a nefunkčné požiadavky, z ktorých vytvoríme návrh na prototyp zásuvného modulu do programu Blender. Implementácia a testovanie sa nachádzajú v nasledujúcich kapitolách.

5.1 Funkčné Požiadavky

Funkčné požiadavky vyjadrujú to, čo by mal náš prototyp zásuvného modulu dokázať, ale aj to, čo by nemal. Na tvorbu funkčných požiadaviek použijeme znalosti z predchádzajúcich kapitol, v ktorých sme si povedali o častých problémoch, a ako ich riešiť.

Začneme tým, čo by náš prototyp mal byť schopný dokázať opraviť alebo detekovať:

- Opraviť transformácie
- Detekovať a opraviť orientáciu normál
- Opraviť nesprávnu geometriu
- Detekovať n-uholníky a trojuholníky
- Non-manifold
  - Detekovať a opraviť diery
  - Detekovať vnútorné plochy
  - Detekovať odpojenú geometriu
  - Detekovať komplexné/singulárne elementy

Naopak, náš prototyp nebude implementovať žiadne nové algoritmy alebo metódy z predchádzajúcej kapitoly. Bude využívať len funkcie, ktoré Blender (alebo jeho vstavané zásuvné moduly) už obsahuje.

5.2 Nefunkčné Požiadavky

Nefunkčné požiadavky vyjadrujú obmedzenia kladené na náš prototyp zásuvného modulu. Konkrétne pôjde o:

- Zásuvný modul bude vytvorený do programu Blender
- Zásuvný modul bude implementovaný v jazyku Python

5.3 Návrh Prototypu

V tejto sekcii si vytvoríme návrh nášho prototypu, povieme si, ktoré zo spomínaných problémov bude riešiť a akým spôsobom sa ich pokúsi vyriešiť.

Opravu transfromácií budeme aplikovať vstavanými funkciami.

- *Apply Location* nastaví lokáciu (polohu) origin pointu daného objektu do stredu súradnícového systému (na súradnice [0;0;0]).
- *Apply Rotation* nastaví rotáciu objektu na nulové hodnoty.
- *Apply Scale* nastaví scale objektu na jednotkévo hodnoty.

Dané opravy bude možné vykonať len v objekt móde.

Vizualizáciu orientácie normál budeme zobrazovať pomocou dvoch vstavaných nástrojov.

- *Face-Orientation* zafarbí mesh v object aj edit móde. Farba bude modrá pre správne a červená pre nesprávne otočené normály.
- *Display normals* v edit móde zobrazí malé čiary buď v mieste vrcholu, „split-vrcholu“ alebo plochy, kde smer čiar udáva orientáciu normály daného elementu.

Opravu orientácie normál rozdelíme na dve časti, podľa toho, či je možné ich vykonať v objekt alebo edit móde.

1. *Objekt mód*
   - *Shading* umožníme nastaviť na *Flat* a *Smooth*, ku ktorému sprístupníme možnosť zapnúť a upraviť *Auto Smooth* uhol. Zmena bude vykonaná na celom objekte.
   - Pridáme ľahko viditeľné tlačidlo na odstránenie *Custom Split Normals*.
   - Vytvoríme operátor pridávajúci *Weighted Normal Modifier*.

2. *Edit mód*
   - *Shading* dovolíme upraviť na *Flat* a *Smooth* konkrétnej ploche, nie celému objektu a ponecháme nastavenie *Auto Smooth* uhla
   - *Flip Normals* nám umožní otočiť normály plôch, ktoré máme vybrané.
   - *Recalculate Outside/Inside* automaticky určí orientáciu normál von alebo dnu na všetkých označených plochách.

Oprava nesprávnej geometrie bude opäť vykonaná v *objekt* a *edit* móde osobitne.

1. *Objekt mód* bude vykonávať opravu geometrie za pomoci *modifiers*.
   - *Decimate Modifier* odstraňuje trojuholníky meshu troma odlišnými spôsobmi, ktoré si zvolí užívateľ.
   - *Remesh Modifier* vytvára úplne novú geometriu na základe starej, kde si užívateľ znova môže vybrať jednu zo štyroch metód.
   - *Triangulate Modifier* zmení všetky plochy na trojúholníkové.
   - *Weld Modifier* spája vrcholy, ktoré sú pri sebe blízko na vzdialenosť prahu.
2. *Edit mód*
   - *Merge by Distance* spája len označené vrcholy, ktoré sú pri sebe blízko na vzdialenosť prahu.
   - *Vyplnenie dier* vykonáme za pomoci troch operátorov:
     a) *Grid Fill* vyplní mriežkou ak má dostupné dva cykly.
     b) *Ngon Fill* vyplní dieru n-uholníkom.
     c) *Triangle Fill* vyplní dieru sadou trojuholníkov.
   - *Separate* na rozdelenie *Selected* častí alebo *by Loose Parts*.

Označenie zlej geometrie bude vyhľadávať a označovať elementy meshu, ktoré sme si popísali v predchádzajúcich kapitolách ako nevyhovujúce.

- *Select Linked* využijeme na označenie všetkých elementov, ktoré sú priamo spojené s už dopredu vyznačeným elementom.
- *Select Triangles/Ngons* nájde a označí všetky trojuholníky alebo n-uholníky v meshi.
- *Select Non Manifold* označí všetky non-manifold elementy meshu.
- *Select Interior Faces* nájde všetky vnútorné plochy.
- *Select Holes* nájde všetky hrany, ktoré sú spojené len s jednou plochov.
- *Select Complex Elements* nájde a označí všetky komplexné alebo singulárne elementy.

Opravu budov vykonáme podľa viacerých už spomínaných operátorov. Tie deštruktívne opravia výzor budov z projektu VMCK. Operátor nazveme *VMCK Fix*. Ten pridá modifiers určené na opravu, odstráni vlastné normály, bude zobrazovať model auto smooth shadingom s uhlom 10 stupňov a všetky modifiers aplikuje.

Modifier utilities ako bonus dodáme operátor pridávajúci viaceré modifiers určené k oprave budov a operátor aplikujúci všetky existujúce modifiers.

5.4 Implementácia

V tejto sekcii si prejdeme, ako sme prototyp naimplementovali, aké sme použili triedy a dátové štruktúry, vzťahy medzi nimi s ostatnými časťami API a ich životný cyklus zásuvného modulu. Prezrieme si štruktúru súborov, a aj ako do nej dopĺňať novú funkčnosť, či použiť nami vytvorené operátory. Na záver si prezrieme inštalačnú a užívateľskú dokumentáciu.

5.4.1 Štruktúra súborov

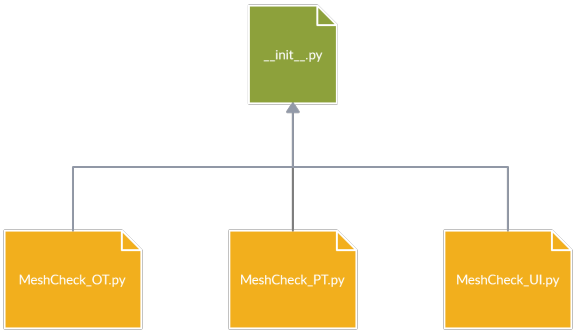
Celý náš zásuvný modul je obsiahnutý v štyroch súboroch, *__init__*, *MeshCheck_PT*, *MeshCheck_UI*, *MeshCheck_OT*.

__init__ Ide o hlavný súbor, v ktorom sa nachádzajú základné informácie o zásuvnom module v slovníku *bl_info*, zároveň je to súbor, ktorý hľadá Blender, aby mohol zásuvný modul nainštalovať. Taktiež sa do tohto súboru importujú všetky ostatné súbory a sú v ňom následne zaregistrované.

MeshCheck_PT Je súbor obsahujúci naše *properties*, čiže dátové štruktúry, ktoré nám umožňujú vytvárať „checkboxes“ a „sliders“ v našich paneloch. Tento súbor musí byť zaregistrovaný ako prvý, inak nebude fungovať.

MeshCheck_UI Súbor je zodpovedný za zobrazovanie nášho zásuvného modulu v „Side Panel“, obsahuje všetok kód zodpovedný za UI. Pre každé rozdelenie panelu existuje samostatná trieda, ktorá v sebe drží poradie a operátory, ktoré má zobraziť.

MeshCheck_OT Obsahuje všetky operátory, čiže funkčnosť zásuvného modulu. V tomto súbore sa nachádza hlavná logická časť spracuvávajúca požiadavky na vykonanie akcií alebo funkcií.



Obr. 5.1: Štruktúra súborov

5.4.2 Životný cyklus

Životný cyklus zásuvného modulu vyjadruje proces inštalácie, prípravy, fungovania a ukončenia práce a funkčnosti zásuvného modulu.

1. Inštalácia slúži k tomu, aby sme informovali Blender o existencií nášho zásuvného modulu. Ak máme jediný python súbor, nainštaluje sa ten, ak ich máme viacero, inštaluje sa súbor *__init__*. Blender si tieto súbory prejde a pokúsi sa v nich nájsť slovník *bl_info*, z toho získa informácie o mene modulu, autorovi, verziách a iných veciach. Ak je všetko v poriadku, Blender zásuvný modul nainštaluje. V opačnom prípade (zlé názvy alebo štruktúra súborov) sa modul nenainštaluje.

2. Registrácia prebehne pri každom spustení programu Blender. Skontrolujú sa základné dáta poskytnuté pri inštalácií, následne sa prejdú python súbory v priečinku *AppData* (Windows) a ak je všetko v poriadku, zásuvný modul sa zobrazí a bude fungovať. Ak nastane chyba, zobrazí sa v termináli.

   Registrácia prebieha vo vstavanej funkcii *register_class*, ktorej musíme poskytnúť všetky mená tried, ktoré chceme používať.

3. Beh modulu označuje štádium životného cyklu, kedy všetky predošlé kroky prebehli v poriadku a je možné využívať všetkých funkcií a operátorov pridaného zásuvného modulu.

4. Deregistrácia prebieha pri ukončovaní programu Blender. V ten moment sa skontrolujú zaregistrované moduly a následne sa deregistrujú. Ak by sa zásuvné moduly nederegistrovali, vznikal by problém množenia modulov.

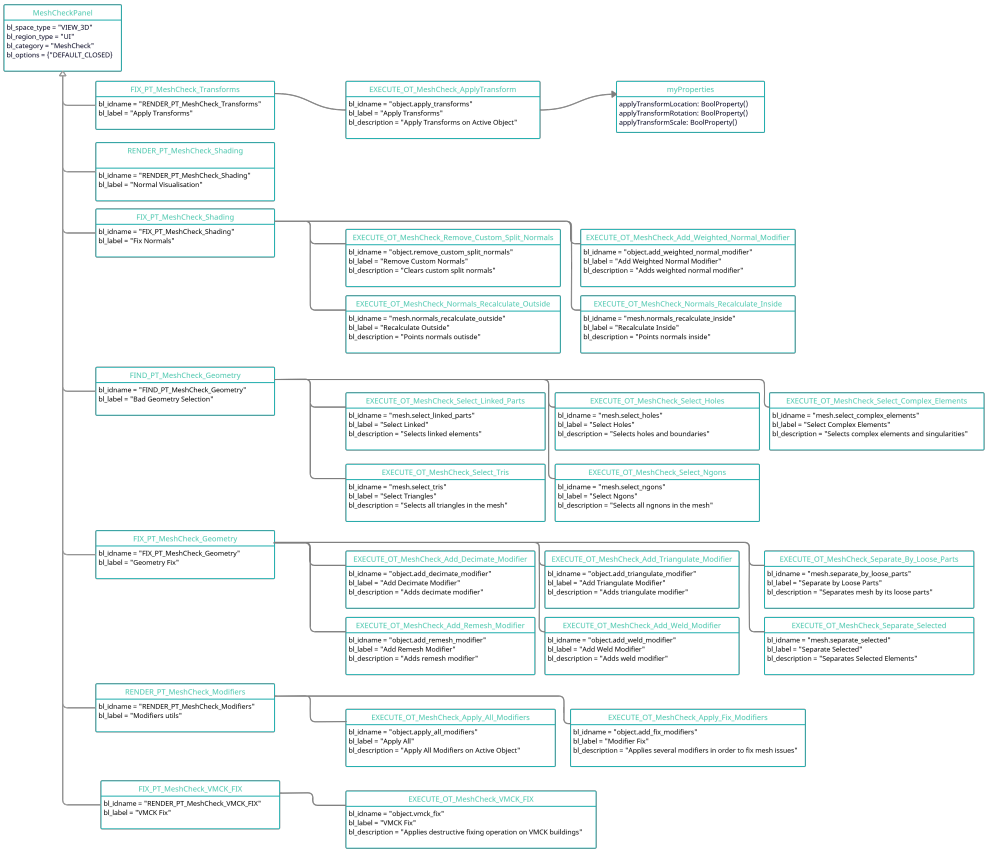
   Deregistrácia funguje podobne ako registrácia, existuje vstavaná funkcia *unregister_class*, ktorej musíme poskytnúť všetky mená tried, ktoré sme zaregistrovali.

5.4.3 Dátové štruktúry a triedy

Nami vytvorené čisté Python dátové štruktúry sú len troje „touples“ s menom *classes*. Tie obsahujú mená tried, ktoré sú nami vytvorené. Je tomu tak z dôvodu, aby sme cez ne mohli iterovať a postupne ich zaregistrovať a odregistrovať.

Dátové štruktúry z Blender API, ktoré sme použili, sa nazývajú properties. V našom kóde sme použili konkrétne BoolProperty, s názvami *applyTransformLocation*, *applyTransformRotation* a *applyTransformScale*. Tie sme použili na vytvorenie „Check Boxu“, ktorý určoval nášmu operátoru, ktoré z transformácií aplikovať.

Tried máme približne 30, z toho dôvodu ich nebudeme všetky vypisovať v texte, ale v diagrame tried, kde si ukážeme aj ich vzájomné vzťahy.



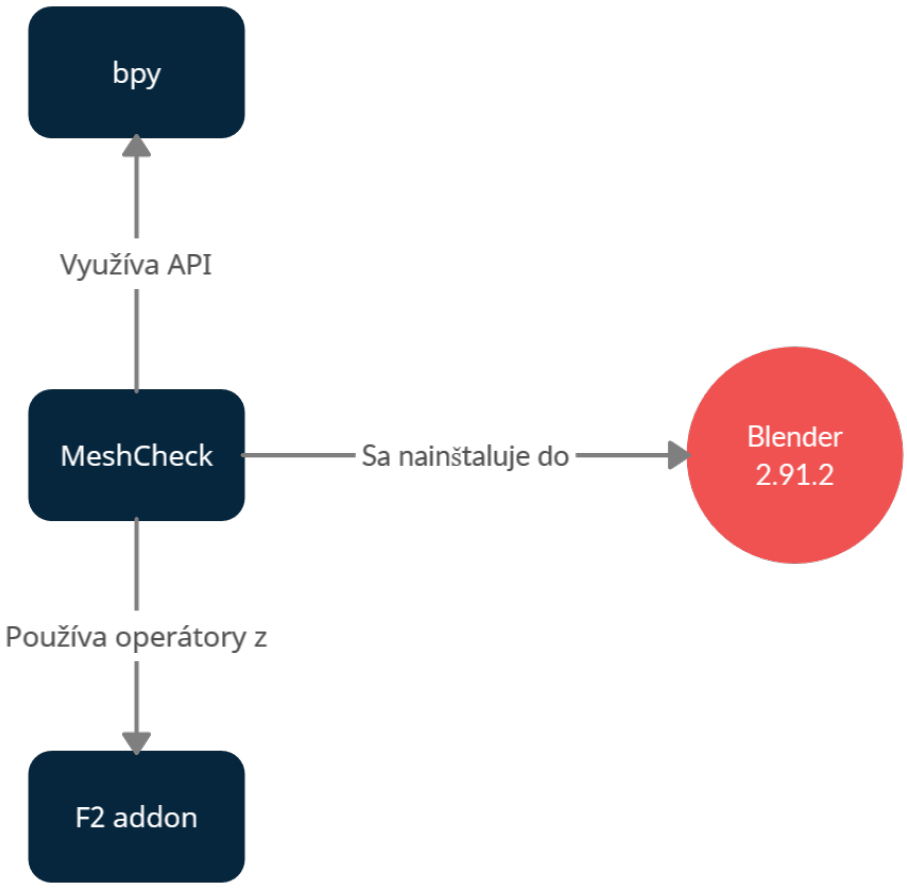
Obr. 5.2: Diagram tried (Odporúčame zobraziť diagram z priložených súborov, kde sa nachádza vo vysokom rozlíšení.)

5.4.4 Schéma nasadenia

Naša schéma nasadenia zobrazuje, čo všetko potrebuje náš zásuvný modul k fungovaniu a do akej verzie programu Blender ho plánujeme nainštalovať.

Blender je napísaný v jazyku C/C++, no je nad ním vytvorená obálka zabaľujúca všetku funkcionalitu do jazyka Python. Vďaka tomu je jednoduché pomocou modulu *bpy* pristupovať ku všetkým operátorom a vstavaným funkciám.

Modul *bpy* nám navyše dovoľuje rozšíriť funkčnosti programu Blender pridávaním nových panelov alebo operátorov. Nami vytvorený zásuvný modul toho využíva a implementuje nové panely a operátory, používajúce iné už existujúce a vstavané operátory. Náš modul používa operátor aj iného predom nainštalovaného zásuvného modulu s názvom F2. Z tohto modulu je využitá funkcionalita vypĺňania dier n-uholníkom.



Obr. 5.3: Diagram nasadenia

5.4.5 Inštalačná príručka

Ak chcete náš zásuvný modul nainštalovať, je potrebné vykonať nasledujúce kroky:

1. Otvorte Blender
2. Edit ⇒ Preferences ⇒ Add-ons ⇒ Install...
3. Nájdite náš zásuvný modul *MeshCheck.zip* ⇒ Install Add-on
4. Uistite sa, že je „Check-Box“ zaškrtnutý pri pridanom module
5. Vľavo dole kliknite na paličky ⇒ Save Preferences
6. Zásuvný model je už úspešne nainštalovaný do *Side Panel*

Tento návod funguje vo verzii 2.91.2 (Win10), rozdiely s inými verziami môžu byť v lokácií *Preferences* menu alebo im môže chýbať menu pre *Save Preferences*, v najhoršom prípade nebude možné modul nainštalovať, kvôli rozdielnym verziám.

5.4.6 Používateľská príručka

Nainštalovaný zásuvný modul sa nachádza v „Side Panel“, ten je možné otvoriť stlačením tlačidla *N*, napravo sa bude nachádzať záložka s názvom *MeshCheck*.

Záložka *MeshCheck* otvorí nový panel, kde bude viacero kategórií podľa problému, ktorý zobrazujú alebo opravujú. Aby sa zobrazili všetky možné kategórie a fungovali ich operátory, je potrebné, aby ste mali označený nejaký mesh a nachádzali sa v *Object* alebo *Edit* móde. Konkrétne ide o kategórie:

- *Apply Tranfsorms* kategória je povolená len v objekt móde, umožňuje aplikovať transformácie objektu.
- *Normal Visualisation* kategória je povolená v objekt a edit móde, povoľuje zobrazovať orientáciu normál.
- *Fix Normals* kategória je povolená v objekt a edit móde, umožňuje opravu normál zmenou shadingu, odstránením vlastných normál alebo pridaním modifiers.
- *Bad Geometry Selection* kategória je povolená len v edit móde, umožňuje nám označovať zlú geometriu ako dvojité elementy a non-manifold.
- *Geometry Fix* kategória je povolená v objekt aj edit móde, umožňuje nám opravovať chyby manifoldnosti, dvojitej geometrie, dier a pridáva modifiers.

- *Modifiers utils* kategória dostupná len v objekt móde obsahuje nástroje k práci s modifiers ako pridanie opráv budov a aplikovanie všetkých modifiers.

- *VMCK Fix* kategória dostupná len v objekt móde obsahujúca jediný operátor opravujúci budovy z aplikácie VMCK.

Každá zo spomínaných kategórií obsahuje viacero operátorov, ktorých vysvetlenie je jasné z názvu alebo z ich popisu dostupného, ak nad nimi necháte položenú myšku. Taktiež je prevažná väčšina týchto nástrojov vstavaná v Blenderi, tým pádom k ním existuje oficiálna dokumentácia a v tej našej vysvetlenie jednotlivých nástrojov vynecháme.

Oficiálna dokumentácia je dostupná tu `https://docs.blender.org/`.

5.4.7 Programátorská príručka

Pre doplnenie funkčnosti modulu alebo využitia už nami implementovaných operátorov odporúčame zo začiatku prečítať sekcie *Štruktúra súborov* a *Životný cyklus*, v nich sa prejde základná štruktúra zásuvného modulu.

Ak ide o doplnenie funkčnosti modulu o nový operátor, ktorý je možné použiť, je potrebné vytvoriť novú triedu s korektným názvom a nekonfliktným identifikátorom *bl_idname* v súbore *MeshCheck_OT*. V danej triede použite funkciu funkciu *execute*, do ktorej vložíte funkčnosť operátora.

Vami vytvorenú triedu je potrebné zaregistrovať, stačí ak jej názov pridáte do dátovej štruktúry *classes* na konci súboru. Hlavný súbor zavolá registráciu v danom súbore a ten zaregistruje všetky triedy z dátovej štruktúry. Rovnakým spôsobom sa odinštaluje.

Pokiaľ chcete pridať váš operátor do už existujúcej kategórie v paneli, stačí pridať dva riadky kódu na miesto, kde chcete, aby sa operátor vykreslil. Ide konkrétne o príkazy *row = layout.row()*, ktorý vytvorí nový riadok. Následne stačí použiť príkaz *row.operator("bl_idname")*, ktorý pridá tlačidlo spúšťajúce váš operátor. *bl_idname* je identifikátor vášho operátora.

Pre pridanie vlastnej kategórie do panelu je potrebné vytvoriť novú triedu v súbore *MeshCheck_UI*, ktorá bude dediť triedu *MeshCheckPanel*. Taktiež je potrebné triedu zaregistrovať a odregistrovať rovnakým spôsobom, aký bol popísaný pri operátoroch. Ak chcete vytvoriť úplne novú záložku, nie je potrebné dediť, no je potrebné vytvoriť novú triedu s viacerými informáciami.

Pre tvorbu tried a panelov je odporúčané použiť už existujúci kód ako predlohu.

Ak chcete použiť nami vytvorený operátor, je potrebné mať nainštalovaný náš zásuvný modul. Následne stačí použiť *bl_idname* nášho operátora vo vašom paneli, alebo skopírovať jeho funkčnosť do vami vytvoreného kódu. Nami vytvorený kód je pod licenciou `http://www.gnu.org/licenses`, to znamená, že je možné ho preberať a upravovať.

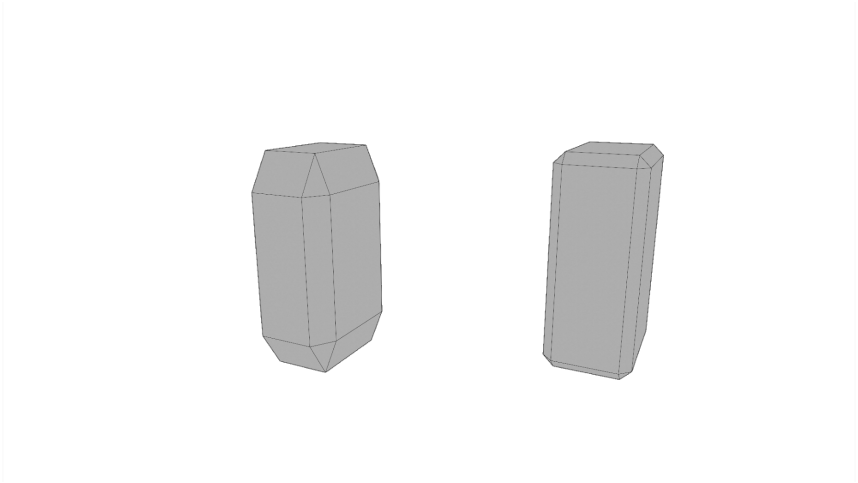
5.5 Testovanie

Testovanie nášho zásuvného modulu prebehne v troch fázach, v prvej vytvoríme problémy s geometriou a použijeme náš modul k nájdeniu daných problémov a ich následnému vyriešeniu, v druhej fáze otestujeme automatickú opravu modelov budov z projektu VMCK a v poslednej fáze prebehne užívateľské testovanie zásuvného modulu.

5.5.1 Prvá fáza

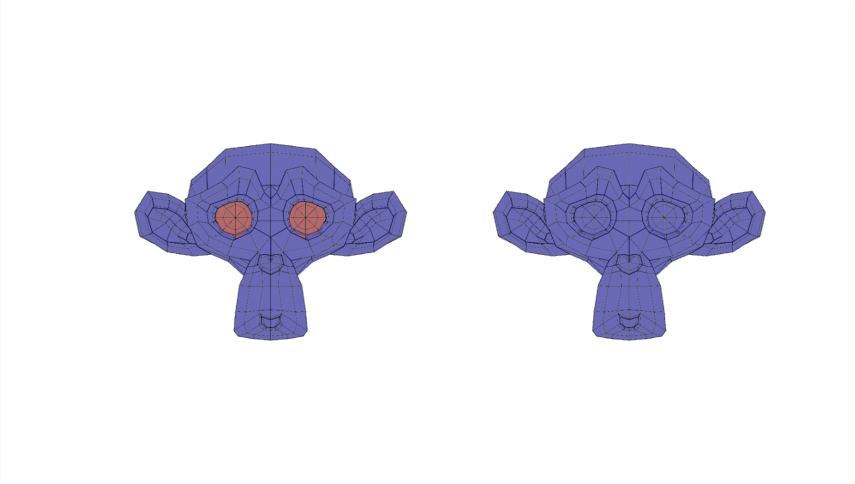
V tejto fáze otestujeme funkčnosť zásuvného modulu na umelo vytvorených problémoch, čím ukážeme jeho užitočnosť a funkčnosť počas modelovania, ale aj pri kontrole kvality. Testovať sa budú najčastejšie problémy ako neaplikované transformácie, dvojitá geometria, otočené normály, diery v geometrii a vnútorné hrany.

Už spomínaným najčastejším problémom počas modelovania v programe Blender je zle nastavený scale objektu. Prejavuje sa zvláštnym fungovaním viacerých nástrojov, z tohto dôvodu sme túto kategóriu zvolili ako prvú. Nástroj na kontrolu scalu sme nepridali, pretože je ľahké ho skontrolovať vo vedľajšej záložke.



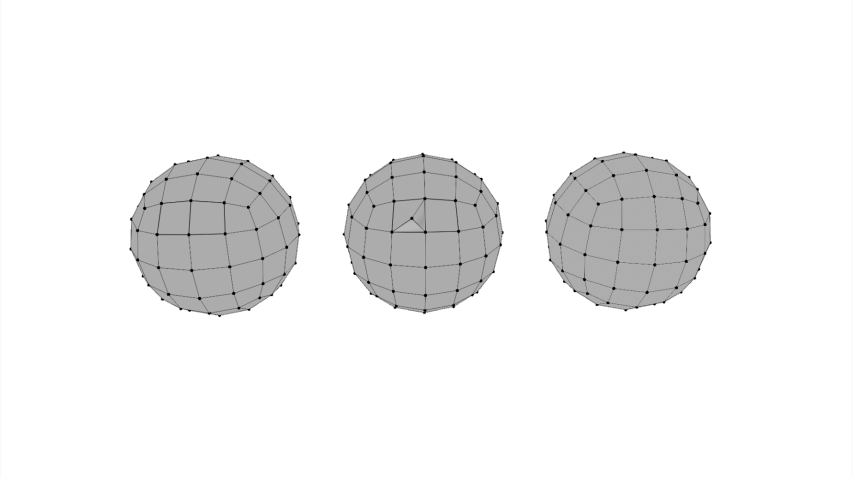
Obr. 5.4: Ukážka opravy scalu objektu s *bevel modifier*

Ďalším veľmi častým problémom je zlé nastavenie normál. Náš zásuvný modul ich dokáže zobraziť aj opraviť. Na vizualizáciu problému sme v tomto prípade použili *Normal Orientation*, vďaka tomu sme videli, že normály očí sú otočené zlým smerom. Prepli sme sa teda do *edit* módu, kde sme ich opravili za pomoci *Recalculate Outside*, mohli sme ich ručne otočiť, no k tomu by ich bolo potrebné označiť.



Obr. 5.5: Ukážka opravy normál

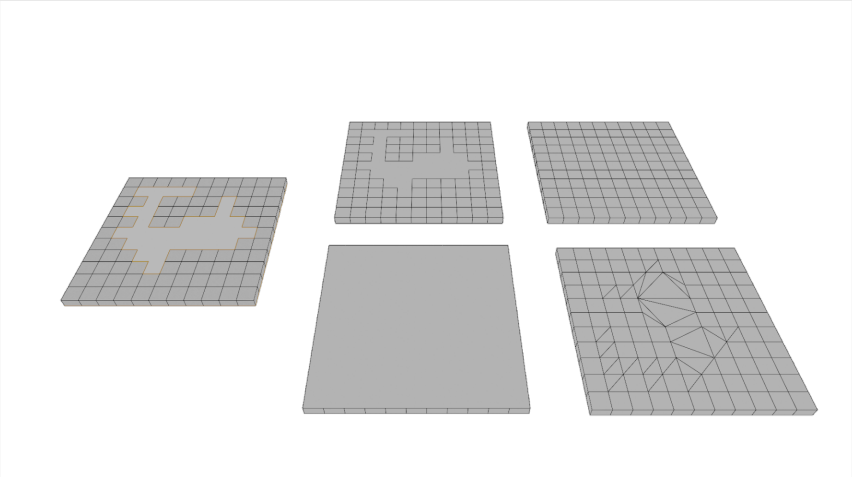
Problém dvojitej geometrie náš modul vizualizovať nevie, vieme ho ale vyriešiť dvoma spôsobmi, ide o *Weld Modifier* a *Merge by Distance*. Sám Blender ale zobrazuje dvojité hrany mierne hrubšou čiarou a problém sa ukáže, ak budete chcieť pohnúť vrcholom, a pod ním sa objaví nový. My sme použili metódu *Merge by Distance*, pretože nám ukáže koľko vrcholov zjednotila.



Obr. 5.6: Ukážka opravy dvojitej geometrie. Vľavo je objekt s dvojitou geometriou, v strede objekt, kde sme posunuli jeden z dvojitých vrcholov, napravo opravený mesh

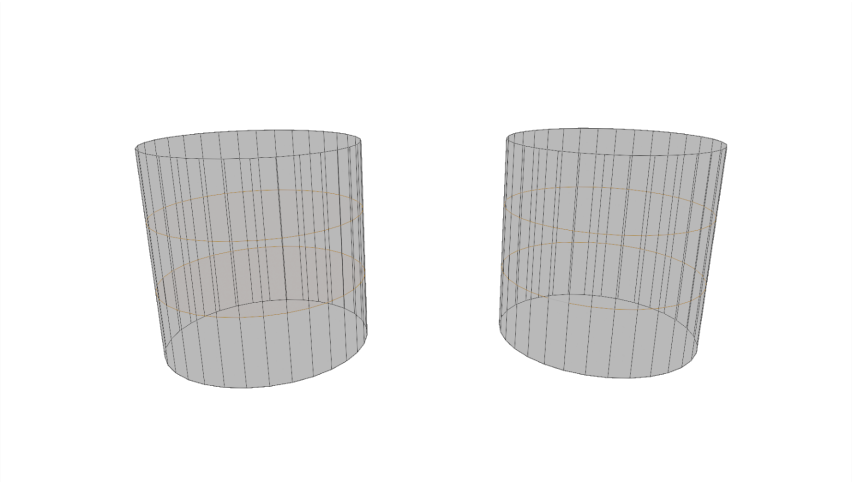
Diery v geometrii sme otestovali označením všetkých dier v meshi príkazom *Select Holes*, tento príkaz našiel dvoje diery, jedna komplexná na hornej časti a druhú na spodnej časti, kde chýba celá strana. Použitím príkazu *Ngon Fill*

sa obe diery uzavrú, avšak n-uholníkom. Existuje druhý postup, kde sme ponechali označenú len hornú dieru a na ňu použili príkaz *Triangle Fill* a spodnú dieru sme rovnakým postupom vyplnili príkazom *Grid Fill*.



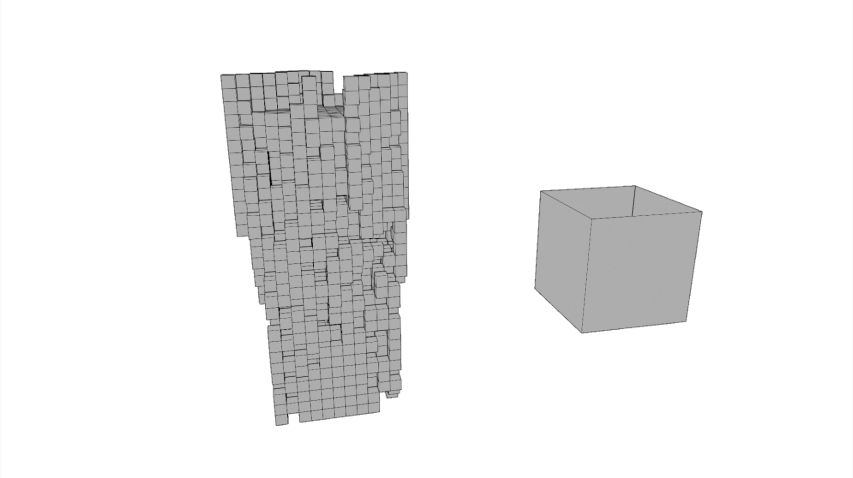
Obr. 5.7: Ukážka opravy dier. Vľavo plocha s dvoma dierami, v strede opravená plocha n-uholníkom a napravo plocha opravená trojuholníkmi a štvorcami. Zobrazuje sa horná a spodná strana

Ďalšou testovanou funkciou je označenie vnútorných plôch. K označeniu vnútoných plôch sme použili príkaz *Select Interior Faces*. Označené plochy sme v našom prípade len odstránili, pretože boli nežiadané. Avšak je možné použiť príkaz *Separate Selection*, ktorý označené plochy prevedie na vlastný objekt.



Obr. 5.8: Ukážka opravy vnútorných plôch

Poslednou nami testovanou funkciou je aplikácia všetkých *modifiers*, tie nám tvoria objekt, aký je možno vidieť na obrázku vľavo, po ich aplikovaní objekt vyzerá rovnako. Ak ich však neaplikujeme a použijeme zlé nastavenia pri exporte a importujeme objekt naspäť, získame objekt vpravo.



Obr. 5.9: Ukážka aplikácie *modifiers*

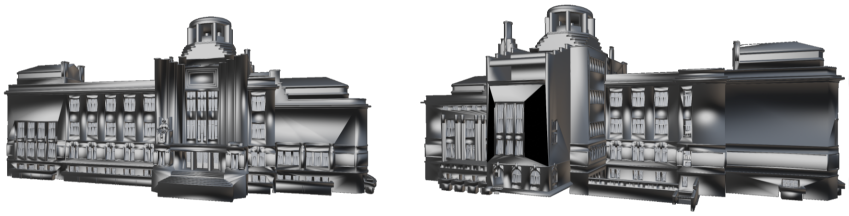
Podobným spôsobom sme otestovali všetky funkcie a operátory, potrebným operátorom sme pridali *classmethod poll*, ktorá zakazuje fungovaniu alebo zobrazovaniu sa operátorom, na základe jednoduchých podmienok.

5.5.2 Druhá fáza

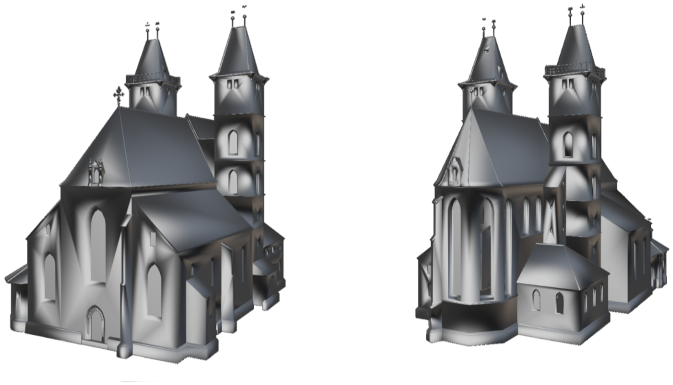
V tejto fáze otestujeme funkcionalitu zásuvného modulu na troch najkomplexnejších modeloch budov, ak bude nástroj fungovať pre ne, môžeme prehlásiť, že bude fungovať aj pre jednoduchšie budovy. Ukážeme si výsledky pred a po, vytvoríme porovnanie a na konci zhodnotíme výsledky opráv, ich správne, ale aj nesprávne časti.

Najkomplexnejší model s najviac problémami je označený ako *múzeum*, druhým modelom bude kostol označený ako *svduch*, ktorý obsahuje takisto mnoho rôznych problémov a naším posledným modelom bude *vodarna*.

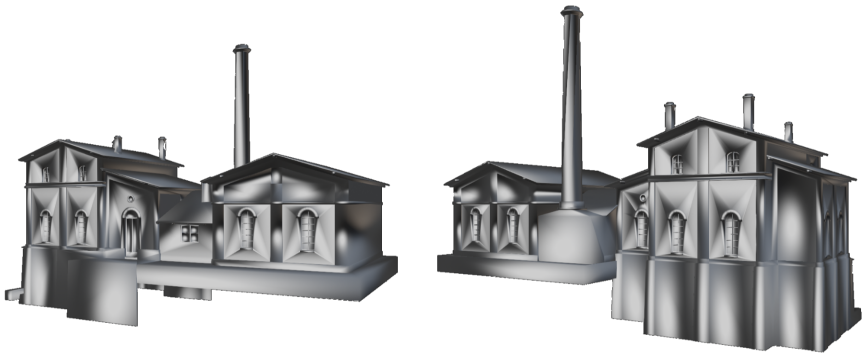
Ako môžeme vidieť, všetky z týchto modelov obsahujú niekoľko jasných problémov s tieňovaním, normálami a ich výraznou trianguláciou. Ďalším problémom je dvojitá geometria, tú však vidieť nemôžeme, no náš modul ju opravuje. Medzi ďalšie problémy patria aj vnútorné plochy alebo prekrývanie elementov, to však náš zásuvný modul neopravuje z dôvodu, že pri týchto opravách sa používa „remeshing“ algoritmus, kvôli ktorému by sme stratili cenné detaily, UV mapy, textúry a v mnohých prípadoch by ani nefungoval, pretože dané meshe obsahujú diery, ktoré taktiež neopravujeme, kvôli vysokej náchylnosti k chybám a chýbajúcim textúram.



Obr. 5.10: Múzeum pred opravou

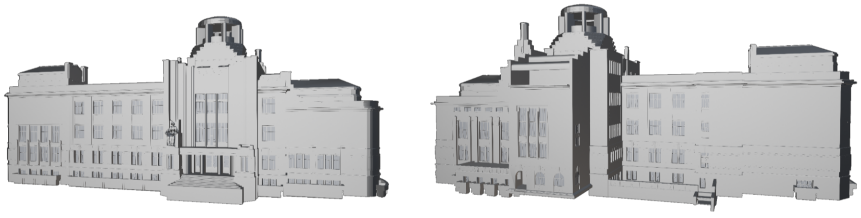


Obr. 5.11: Svduch pred opravou



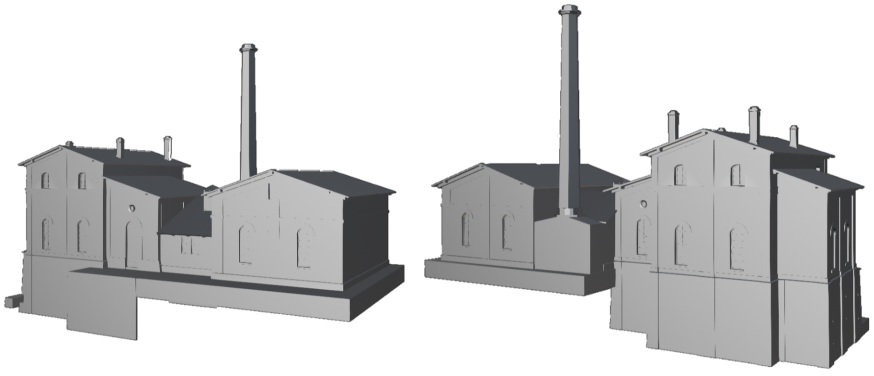
Obr. 5.12: Vodarna pred opravou

Nasledujúce obrázky znázornia tie isté modely po aplikovaní našej opravy s názvom *VMCK Fix*, tá opravuje vlastné normály, upravuje uhol, kedy sa normály majú zobrazovať hladko, odstraňuje dvojitú geometriu, odstráni nadbytočnú geometriu z rovinných plôch a vytvorí konštantnú trianguláciu, aby nevznikali problémy pri prenose do iných aplikácií.



Obr. 5.13: Múzeum po oprave

Obr. 5.14: Svduch po oprave



Obr. 5.15: Vodarna po oprave

Môžeme vidieť, že opravené modely vyzerajú objektívne lepšie a nemajú výrazné problémy s normálami a trianguláciou. Nejde však o dokonalú opravu, v istých miestach si môžeme všimnúť nedostatky. Ide hlavne o zaoblené miesta ako komín na *vodarni* a kupola či zaoblené okraje *múzea*. Tieto problémy sú spôsobené spomínaným uhlom, kedy sa majú veci zobrazovať ako hladké, ak daný uhol zvýšime, začnú sa znova objavovať problémy aj na rovných plochách. Druhým, menej nápadným problémom sú pretrvávajúce problémy

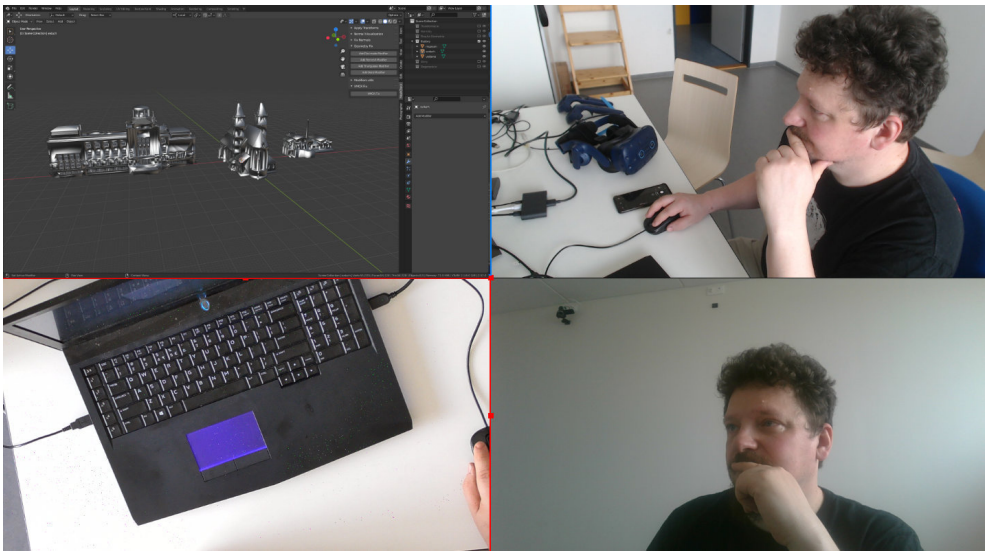
s trianguláciou pri oknách. Najlepšie je tento pretrvávajúci problém vidieť v modele *vodarna*, pravý obrázok, horné okná. Problém je spôsobený rovnako ako predošlé, v tomto prípade však treba uhol znížiť.

5.5.3 Tretia fáza

V tejto poslednej fáze podrobíme zásuvný modul užívateľským testom a uvedieme si výsledky, ktoré sme počas testovania získali. Inštrukcie pre testerov, testovací scenár a .blend súbor s rôzne poškodenými modelmi, sú v priložených súboroch.

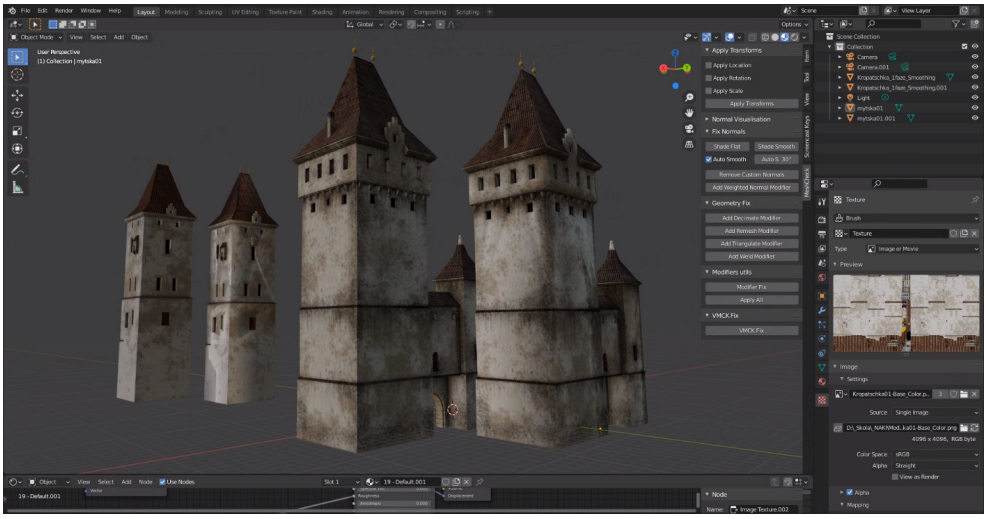
Z dôvodu pandémie COVID-19 bolo užívateľské testovanie vykonané na diaľku alebo v obmedzenom počte ľudí.

Testovanie, ktoré prebehlo v školskom laboratóriu, sa zúčastnili traja testeri, jedným z nich bol aj vedúci práce. Testovanie prebehlo podľa očakávaného scenára a s predvídateľnými výsledkami. Testeri, ktorí sú skúsenejší v programe Blender prešli testom rýchlo a modul nemali problém ovládať. Na druhú stranu, menej skúsení užívatelia mali problém nájsť správne nástroje v kategóriách, ktoré sme vytvorili. Z toho môžeme povedať, že je potrebné zapracovať na menách kategórií a operátorov, taktiež by bolo vhodné pridať lepšie popisy priamo v programe.



Obr. 5.16: Vedúci práce Ing. Jiří Chludil na záberoch počas užívateľského testovania v školskom laboratóriu

Diaľkové testovanie prebehlo cez *Microsoft Teams* s vedúcim práce, kde sme otestovali funkčnosť zásuvného modulu na modeloch budov s textúrami, taktiež sme prebrali výsledky celého testovania. Ukázalo sa, že žiadna z našich opráv nie je voči textúram a UV mapám deštruktívna, vďaka čomu je možné opravu reálne použiť.



Obr. 5.17: Výsledok diaľkového používateľského testovania. Dva páry budov, kde ľavá z nich je opravená

5.6 Budúci Rozvoj

Nami vytvorený zásuvný modul však nepovažujeme za dokončený, preto si v tejto sekcii povieme, ako by sme ho naďalej mohli rozširovať alebo upravovať. Návod, ako konkrétne upraviť/rozšíriť modul, sa nachádza v predošlej sekcii s menom *Programátorská príručka*.

5.6.1 Rozšírenie existujúcej funkcionality

Aktuálne implementované nástroje na opravu a validáciu geometrie síce fungujú, no zaslúžili by si úpravy k uľahčeniu práce s nimi.

Najväčším kandidátom na takúto úpravu je *Select Holes*, ktorý označí všetky diery naraz. Táto funkcionalita je síce užitočná, no bolo by viac užitočné ak by sa diery označovali postupne a nie naraz. Takúto funkcionalitu je možné vyriešiť za pomoci *vertex groups*, kde by sme každú dieru pridelili do vlastnej skupiny. Postupnou iteráciou cez tieto skupiny by sme mohli priamo diery aj opravovať a vytvoriť vďaka tomu nový operátor, ktorý diery nájde a postupne opraví.

Nástroje na prepočítanie normál *Recalculate Inside/Outside* by si tiež zaslúžili miernu úpravu. Tie fungujú len na označenej geometrii, no takmer vždy ich chceme aplikovať na celom meshi. Bolo by teda ideálne automaticky označiť celú mesh, prepočítať jej normály a vrátiť sa k pôvodne označeným elementom.

Posledný z existujúcich nástrojov, ktorý by si zaslúžil úpravu je rôzne označovanie non-manifold alebo degenerovaných elementov. Niektoré z týchto označení sú podporované iba v *Edge* móde a bolo by vhodné sa doň automaticky prepnúť.

5.6.2 Pridanie novej funkcionality z programu Blender

Blender je software pridávajúci novú funkcionalitu každý deň a komplexný software, kde jeho všetka funkcionalita nemusí byť známa ani pre jeho dlhoročných užívateľov. Z tohto dôvodu je vhodné modul naďalej rozširovať a podporovať novými funkciami, ktoré budú implementované v nasledujúcich verziách.

Taktiež je vhodné pridať už existujúcu funkcionalitu, o ktorej sme počas písania práce nevedeli. Ide o funkcie ako *Mesh Analysis*, ktorá sa nachádza vo *Viewport Overlays*, rôzne opravy geometrie nachádzajúce sa v *Mesh - Clean Up* alebo iné opravy, o ktorých ešte stále nevieme.

5.6.3 Vytvorenie novej funkcionality

Vytvorenie novej funkcionality je najlepší, ale aj najnáročnejší spôsob, ako rozšíriť náš zásuvný modul. K rozšíreniu môžeme použiť niektoré z algoritmov, ktoré sú spomínané v tretej kapitole. Môže ísť o algoritmy slúžiace k automatickej retopológii, vypĺňaniu dier alebo automatickej oprave non-manifold geometrie.

Taktiež je možné rozšíriť modul aj novými dodatkami do UI, ako napríklad automatické zafarbovanie degenerovaných elementov, pretínajúcich sa plôch alebo okrajov dier.

Záver

Hlavným cieľom tejto práce bolo vytvoriť zásuvný modul do programu Blender, ktorý bude opravovať modely budov z projektu *Věnná Města Českých Královen*. Ten sme navrhli, vytvorili mu funkčné a nefunkčné požiadavky, vysvetlili si jeho implementáciu, vytvorili rôzne príručky a jeho funkčnosť otestovali. Modul však nemusíme považovať za úplný, je naďalej možné ho rozšíriť novými operátormi, rozdeliť označené problémy do podproblémov, vylepšiť jeho usporiadanie, alebo pridať úplne nové vlastné algoritmy.

Inšpiráciou pre tvorbu nových algoritmov môže byť výsledok ďalšieho cieľa práce, a tým je súhrn algoritmov. V tom sa nám podarilo vytvoriť zhrnutie a porovnanie rôznych algoritmov a metód riešiace problémy vznikajúce v geometrii 3D modelu. Jedná sa napríklad o uzatváranie dier, oprava degenerovaných elementov alebo odstraňovanie šumu. Taktiež sme si vysvetlili rozdiel medzi *upstream* a *downstream* aplikáciami a povedali si o rozdieloch medzi lokálnym a globálnym prístupom pri oprave problémov. Samotné problémy sme rozdelili do troch kategórií.

V neposlednom rade sme vytvorili veľké zhrnutie základov 3D modelov a modelovania. Podarilo sa nám vytvoriť rýchle historické zhrnutie a náhľad do budúcnosti. Povedali sme si o možných reprezentáciách 3D objektov a dátových štruktúrach, ktoré ich vyjadrujú. Uviedli sme si z akých elementov sa skladá 3D model a predviedli si najpoužívanejšie dátové formáty, ktoré dokážu ukladať niektoré z daných elementov. V kategórii o základoch modelovania sme prebrali prečo modelovať, ako modelovať a v akom štýle, uviedli sme aj klasickú modelovaciu pipeline. Ako poslednú vec sme si ukázali najčastejšie problémy, ktoré vznikajú s geometriou, ak pracujeme s programom Blender.

Bibliografia

1. *HISTORY OF 3D MODELING: FROM EUCLID TO 3D PRINTING* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://ufo3d.com/history-of-3d-modeling.
2. *History of 3D Rendering* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://pixelperfect-studios.com/history-of-3d-rendering/.
3. *A History of Computer Graphic Modeling* [online]. 2014 [cit. 2021-04-06]. Dostupné z: https://www.digitalschool.ca/a-history-of-computer-graphic-modeling/.
4. EIKENES, Jon Olav. *Navimation : a sociocultural exploration of kinetic interface design*. 2010. Dizertačná práca.
5. KULKARNI, Rahul. *Evolution of CAD: From light pens to Synchronous Technology!* [online]. 2017 [cit. 2021-05-05]. Dostupné z: https://medium.com/technical-illustration/evolution-of-cad-from-light-pens-to-synchronous-technology-549cc8eef5d0.
6. OROKRO. *I GOT BLENDER 1.0 WORKING!* [online]. 2020 [cit. 2021-05-05]. Dostupné z: https://www.reddit.com/r/blender/comments/hn6avd/i_got_blender_10_working_video_coming_soon/.
7. *How to represent 3D Data?* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://towardsdatascience.com/how-to-represent-3d-data-66a0f6376afb.
8. *3D Model and CAD Model* [online]. 2021 [cit. 2021-04-06]. Dostupné z: https://www.sculpteo.com/en/glossary/3d-model-definition/.
9. *Polygon mesh* [online]. 2021 [cit. 2021-04-06]. Dostupné z: https://en.wikipedia.org/wiki/Polygon_mesh.
10. BOGUSLAWSKI, Pawel. Modelling and analysing 3D building interiors with the dual half-edge data structure. 2011. ISSN 2220-9964. Dostupné z DOI: 10.3390/ijgi5020019.

11. ASGHARI, Maryam; SADAT, Mojtaba T; BOGUSLAWSKI, Pawel. „An Overview on Boundary Representation Data Structures for 3D Models Representation “. In: *International Symposium & Exhibition on Geoinformation (ISG)*. 2013.

12. RUDRARAJU, Anirudh; DAS, Suman. Digital data processing strategies for large area maskless photopolymerization. In: *International Solid Freeform Fabrication Symposium, The University of Texas at Austin*. 2009.

13. ROSSIGNAC, Jarek; SAFANOVA, Alla; SZYMCZAK, Andrzej. Edgebreaker on a Corner Table: A Simple Technique for Representing and Compressing Triangulated Surfaces. 2003. ISBN 978-3-642-62801-6. Dostupné z DOI: 10.1007/978-3-642-55787-3_3.

14. SHIN, Hayong; PARK, Joonchul; CHOI, Byoung; CHUNG, Yunchan; RHEE, S. Efficient topology construction from triangle soup. In: 2004, s. 359–364. ISBN 0-7695-2078-2. Dostupné z DOI: 10.1109/GMAP.2004.1290060.

15. *The Most Common 3D File Formats* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/.

16. *Wavefront OBJ File Format* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml.

17. *Wavefront Material Template Library (MTL) File Format* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000508.shtml.

18. *STL (STereoLithography) File Format Family* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml.

19. *Reading and writing Filmbox (FBX) files* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://www.sidefx.com/docs/houdini/io/fbx.html.

20. *GlTF-specification-2.0* [online]. 2016 [cit. 2021-04-06]. Dostupné z: https://github.com/KhronosGroup/glTF/blob/master/specification/2.0/README.md.

21. *TYPES OF 3D MODELING: WHICH ONE TO CHOOSE?* [online]. 2018 [cit. 2021-04-06]. Dostupné z: https://ufo3d.com/types-of-3d-modeling-different-industries.

22. *What You Need to Know About 3D Art: [And the Harsh Truth About Taxonomy of 3D Game Art Styles]* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://kevurugames.com/blog/what-you-need-to-know-about-3d-art-and-the-harsh-truth-about-taxonomy-of-3d-game-art-styles/.

23. *Top 6 Styles for 3D Art Development* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://3d-ace.com/press-room/articles/top-6-styles-3d-art-development.

24. *20+ Styles of 3D Modeling in 20 Minutes* [online]. 2018 [cit. 2021-04-06]. Dostupné z: https://www.creativeshrimp.com/3d-modeling-styles.html.

25. *10 Different types of 3D modeling techniques* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://artisticrender.com/10-different-types-of-3d-modeling-techniques/.

26. WINCHENBACH, Rene; HOCHSTETTER, Hendrik; KOLB, Andreas. Infinite Continuous Adaptivity for Incompressible SPH. *ACM Trans. Graph.* 2017, roč. 36, č. 4. ISSN 0730-0301. Dostupné z DOI: 10.1145/3072959.3073713.

27. *Photogrammetry: Advanced Capture* [online]. 2020 [cit. 2021-05-05]. Dostupné z: https://www.cinecommunities.org/photogrammetry-advanced/.

28. *UV Mapping* [online] [cit. 2021-05-05]. Dostupné z: https://www.vectary.com/docs/uv-mapping/.

29. *Non-Manifold Edges: Facts & Fixes* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://all3dp.com/2/non-manifold-edges/.

30. *How to fix non-manifold geometry* [online]. 2021 [cit. 2021-04-06]. Dostupné z: https://www.sculpteo.com/en/3d-learning-hub/create-3d-file/fix-non-manifold-geometry/.

31. *What is a non-manifold mesh and how to fix it* [online]. 2020 [cit. 2021-04-06]. Dostupné z: https://sinestesia.co/blog/tutorials/non-manifold-meshes-and-how-to-fix-them/.

32. *What is non-manifold geometry?* [online]. 2014 [cit. 2021-04-06]. Dostupné z: https://blender.stackexchange.com/questions/7910/what-is-non-manifold-geometry.

33. AKENINE-MÖLLER, Tomas; HAINES, Eric; PESCE, Angelo; HOFFMAN, Naty; IWANICKI, Michał; HILLAIRE, Sébastien. *Real-Time Rendering 4th Edition*. Boca Raton, FL, USA: A K Peters/CRC Press, 2018. ISBN 978-1-13862-700-0.

34. *Are Ngons Really That Evil in 3D Modeling?* [online]. 2019 [cit. 2021-04-06]. Dostupné z: https://www.creativeshrimp.com/ngons-tutorial.html.

35. ATTENE, Marco; CAMPEN, Marcel; KOBBELT, Leif. Polygon Mesh Repairing: An Application Perspective. *ACM Computing Surveys*. 2013, roč. 45. Dostupné z DOI: 10.1145/2431211.2431214.

Dodatok **A**

Zoznam použitých skratiek

napr. Napríklad

VMCK Věnná Města Českých Královen

VR Virtuálna Realita

AR Augmentovaná Realita

CAD Computer Aided Design

API Application programming interface

tzv. Takzvané

CW Clockwise

CCW Counterclockwise

DCEL Doubly connected edge list

UE Unreal Engine

PBR Physically Based Rendering

GRF General Reflectance Function

BSDF Bidirectional Scattering Distribution Function

BTF Bidirectional Texture Function

MLS Moving Least Squares

API Application Programming Interface

UI User Interface

Dodatok B

Obsah priloženého súboru

- `readme.txt` ..... stručný popis obsahu súboru
- `MeshCheck.zip` ..... zásuvný modul
- `VMCK - Validace kvality meshe 3D modelů.pdf` ..... text práce
- `src`
  - `MeshCheck` ..... zdrojové kódy implementácie
    - `__innit__.py`
    - `MeshCheck_OT.py`
    - `MeshCheck_PT.py`
    - `MeshCheck_UI.py`
  - `Text` ..... zdrojová forma práce vo formáte LaTeX
    - `Images` ..... obrázky použité v práci
    - `cvutlogobwen.pdf`
    - `cvutlogobw.pdf`
    - `FITthesis.cls`
    - `mybibliographyfile.bib`
    - `prohlaseni1.tex.txt`
    - `Sablona_sk_BP_UTF-8.tex`
    - `zadanie.pdf`
  - `Užívateľské testy`
    - `UžívateľskéTesty.rar` ..... Podklady k užívateľským testom.