Jak sestavit a připravit 3D model pro web: od mračna bodů po zjednodušený mesh

Matouš Melecký

2. 8. 2020

Obsah

1 Úvod **2**

2 Výroba modelu **2**
2.1 Použitý software . . . 2
2.1.1 Cloud Compare . . . 2
2.1.2 Meshlab . . . 2
2.1.3 Blender . . . 2
2.2 Od mračna bodů k 3D modelu . . . 2
2.2.1 Čištění a unifikace . . . 2
2.2.2 Převod na polygonovou síť (mesh) . . . 3
2.3 Úprava modelu . . . 4
2.4 Tvorba textury . . . 5
2.4.1 Parametrizace . . . 5
2.4.2 Nabarvení texturové mapy . . . 6
2.5 Optimalizace modelu pro webové použití . . . 7
2.5.1 Přeměna pomocí skriptu . . . 7
2.5.2 Přeměna pomocí více nástrojů . . . 7

3 Webový framework **9**
3.1 Funkcionalita . . . 9
3.1.1 Anotace . . . 9
3.1.2 Ovládací prvky . . . 10
3.1.3 Načítací obrazovka . . . 10
3.2 Technické detaily aplikace . . . 12
3.2.1 Použité technologie . . . 12
3.2.2 Adresářová struktura . . . 12
3.2.3 Galerie . . . 12

1 Úvod

Tento dokument popisuje postup tvorby 3D modelů historických objektů a jejich zveřejnění na webu společně s anotacemi, pomocí jednoduchého frameworku. Prvotní data na zpracování byla ve formě obarveného 3D mračna bodů, získaného pomocí 3D skeneru, konkrétně FARO. Vstupní formát byl tedy .fls, který je využíván právě FARO skenery.

2 Výroba modelu

2.1 Použitý software

Při výrobě modelu byl využit pouze volně dostupný open source software. Zde jsou uvedeny hlavní nástroje potřebné ke zpracování modelu, menší skripty budou uvedeny v průběhu textu.

2.1.1 Cloud Compare

Použitá verze: 2.10.2

Cloud Compare je software pro práci s mračny bodů a našlo tedy své využití v úvodních fázích procesu.

2.1.2 Meshlab

Použitá verze: 2016.12

Meshlab dokáže aplikovat množství filtrů a algoritmů na samotný 3D model ve formátu mesh, který se vytvoří z mračna bodů. Jeho hlavní využití byla decimace a simplifikace modelů, což umožnilo zmenšit jejich velikost natolik, že je bylo možno vyexportovat do webového formátu.

2.1.3 Blender

Použitá verze: 2.80.75

Blender byl využit k získání kvalitnějších texturových map. Základní export textury umožňuje i Meshlab, nedosahuje však tak dobrých výsledků.

2.2 Od mračna bodů k 3D modelu

2.2.1 Čištění a unifikace

Vstup: obarvené .fls skeny z FARO skeneru
Výstup: soubor .ply, obsahující všechny skeny pohromadě

Mračno bodů z FARO skenu se nachází v několika .fls souborech a jeho velikost se pohybuje v řádu gigabajtů. Prvním krokem je převedení těchto souborů do formátu, s kterým umí CloudCompare lépe pracovat a navíc je

méně paměťově náročný. Ideálním pracovním formátem se ukázal být formát .ply (Polygon File Format).

Zároveň s tímto krokem můžeme provést více operací. Důležitý je výpočet normál, které bude CloudCompare potřebovat v dalším kroku k převodu na polygonovou síť.

Jelikož je celý proces výroby modelu poměrně náročný na výpočetní výkon, můžeme v závislosti na výkonnosti počítače provést tzv. subsampling, který spočívá v tom, že je definována minimální vzdálenost dvou bodů v prostoru a algoritmus odstraní body, které leží moc blízko. Tato operace dokáže velmi zmenšit objem dat, se kterými je nutné pracovat, zároveň ale ztratíme detail. Pokud je tedy počítač dost výkonný, není nutno subsampling provádět, výsledný model bude více detailní. Další zjednodušování modelu je nutno provádět ve fázi polygonové sítě, ale zde se již dají použít více sofistikované algoritmy, které dosáhnou dobrých výsledků při zachování detailů.

Tato fáze je dobře automatizovatelná. Níže je přiložen příklad jednoduchého skriptu v jazyce BASH, který pro každý .fls soubor provede výše popsané operace.

```
#!/bin/bash
# $1 - slozka s .fls soubory
cd "$1"
for i in *.fls; do
        [ -f "$i" ] || break
        '/cesta/k/CloudCompare/CloudCompare' -SILENT
        -COMPUTE_NORMALS -O "$1/$i" -C_EXPORT_FMT PLY
        -AUTO_SAVE OFF -SS SPATIAL 0.0085 -SAVE_CLOUDS
done
```

Použití: nazevskriptu cesta/k/slozce/se/soubory

Přepínač *SS* přijímá parametry subsamplingu, jeho typ a minimální vzdálenost. *COMPUTE NORMALS* vypočítá normály. Podrobnější popis použití CloudCompare z příkazové řádky lze nalézt zde. Samozřejmě existuje možnost použít již v tomto kroku uživatelské prostředí CloudCompare, což však může být nepohodlné, pokud je .fls souborů mnoho.

Po převodu do .ply otevřeme všechny soubory v uživatelském prostředí CloudCompare. Na obrazovce by se mělo objevit mračno bodů ze skenu. Můžeme aplikovat několik dalších filtrů - například SOR filtr (Statistical Outlier Filter), který detekuje body, které jsou hodně vzdálené od svých sousedů. Také je vhodné odstranit části skenu, které nechceme ve výsledném modelu.

Nakonec je vhodné výsledek uložit jako jeden .ply soubor.

2.2.2 Převod na polygonovou síť (mesh)

Vstup: mračno bodů ve formátu .ply
Výstup: polygonová síť (model) ve formátu .ply

Převod zvládne jak CloudCompare, tak Meshlab. V tomto postupu byl zvolen převod v CloudCompare.

Filtr pro převod nalezneme pod *Plugins* → *PoissonRecon*. Tento algoritmus je zejména vysoce odolný proti šumu a jeho výsledkem jsou hladké povrchy s minimálním množstvím artefaktů. Důležitým parametrem algoritmu je Octree depth, tedy hloubka stromu, který slouží v běhu algoritmu. V zásadě platí, že čím větší je hloubka, tím detailnější model dostaneme (např. hloubky 10 - 12 by měly vytvořit detailní model – za cenu dlouhého běhu algoritmu).

Jedna nevýhoda tohoto algoritmu pro 3D modely interiérů spočívá v tom, že vytváří vodotěsné povrchy a jako meziprodukt obalí výsledný model do oválného tvaru. Ten snadno odstraníme v dalších krocích.

2.3 Úprava modelu

Vstup: model ve formátu .ply ve vysokém rozlišení
Výstup: několik redukovaných modelů v nižších rozlišeních

Po otevření souboru v MeshLabu můžeme použít několik jednoduchých filtrů, které se nalézají pod záložkou *Filters* → *Cleaning and Repairing*.

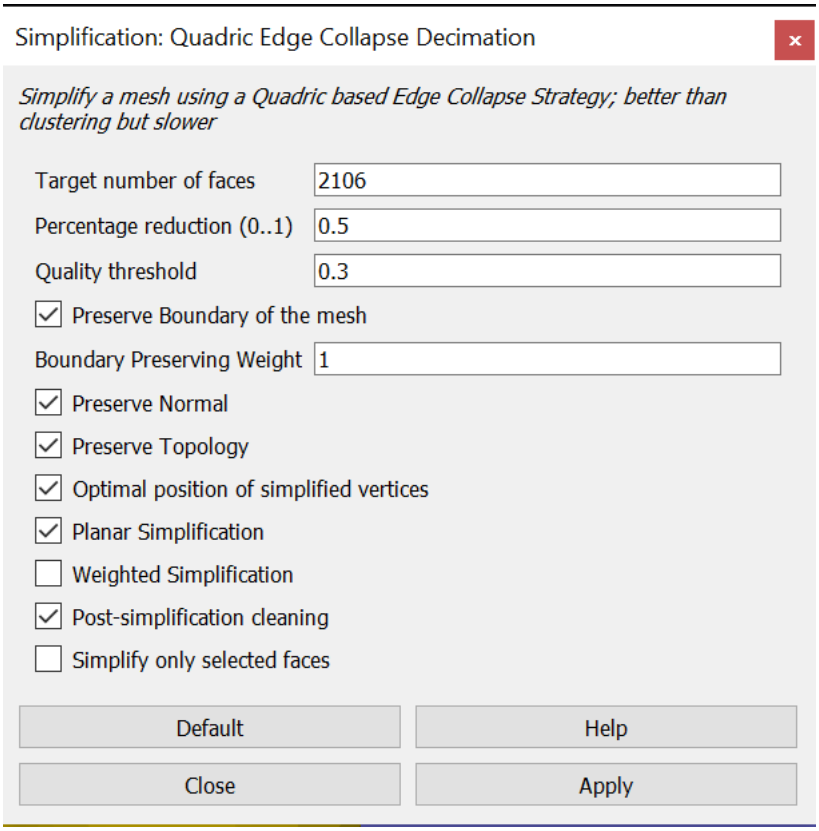
K odstranění skořápky vytvořené Poisson algoritmem lze využít funkcionalitu *Selection* → *Select Faces Longer than...* Jak její název napovídá, vybere ty polygony ze sítě, které mají hranu delší než specifikovaná hodnota. Vždy je nutno ověřit, jestli jsme nenastavili moc malou délku a filtr nevybere i oblasti modelu, které bychom chtěli zachovat.

Pokud se však nastaví správná hodnota, měla by být vybrána pouze skořápka. Pak stačí stisknout tlačítko DELETE a vybraná oblast se smaže. Může se stát, že některé zbytky skořápky zůstanou – pokud budou velké, dají se jednoduše odstranit ručně. Pro odstranění menších slouží *Filter* → *Remove Isolated Pieces*.

Dalším krokem po aplikování těchto drobných filtrů a očištění modelu je zjednodušení sítě. Model se nyní skládá z mnoha polygonů, což znamená, že v sobě obsahuje mnoho detailů, ale zároveň je náročný na výkon a většina uživatelů na internetu by jej ani nestáhla v rozumném čase. Proto Meshlab nabízí možnost decimace modelu – což je přesně zjednodušení sítě, které se snaží redukovat počet polygonů a přitom co nejvíc zachovat tvar modelu.

Nejkvalitnější výsledky produkuje *Remeshing* → *Quadric Edge Collapse Decimation*. Základní parametry filtru umožňují zejména nastavení rozlišení výsledného modelu. Je možno buď zadat požadovaný počet polygonů zjednodušeného modelu, nebo jeho kvalitu určit jako procento původního modelu. Dále je možno nastavit další omezení, kterých by se měl algoritmus držet. Ty zlepší kvalitu výsledku, na druhou stranu poběží algoritmus déle a nemusí se mu povést model dostatečně zjednodušit. Nejlepší je experimentovat, zkusit několik různých rozlišení s různými parametry a porovnat je. I u poměrně velkých modelů – např. celý kostel, se lze dostat z deseti milionů hran do řádů stovek tisíc.

Všechny výsledné modely je dobré ukládat ve formátu .ply.



Obr. 1: Quadric Edge Collapse Decimation

2.4 Tvorba textury

Zatím naše modely byly obarveny na bázi vrcholových barev. To znamená, že každý vrchol polygonu měl specifikovánu RGB barvu. V praxi se však nepoužívá obarvení podle vrcholů, ale tvoří se textura. Princip textur je jednoduchý – polygony se rozloží do mnoha částí a ke každé se přiřadí část obrázku, který specifikuje její barvu. Textura tedy sestává z jednoduchého 2D obrázku ve standartním formátu a informací o tom, kam se má která část obrázku zobrazit na model. Mezi jednu z výhod textur patří to, že je počítače umí lépe zobrazovat – grafické karty jsou jim uzpůsobeny. Navíc výsledek zpravidla vypadá lépe. Dobrá textura dokáže zakrýt zjednodušení geometrie modelu.

2.4.1 Parametrizace

Vstup: model s nižším rozlišením ve formátu .ply – ten, který chceme použít a vytvořit pro něj texturu

Výstup: model ve formátu .ply, který v sobě obsahuje údaje o parametrizaci textury

Nejprve potřebujeme získat parametrizaci textury, tedy provést ono rozdělení polygonů na části, aby pak bylo možno texturu namapovat na 3D objekt. Tuto operaci je možno provádět ručně, či spíš poloautomaticky. V případě velkého modelu je toto však nesmírně pracné a nepraktické a vyplatí se spíše využít nějakého automatického postupu.

Blender umožňuje jak ruční nastavení, tak plně automatické. Nejprve je nutno přepnout na *Edit Mode*. Po zmáčknutí klávesy U se objeví nabídka týkající se tvorby texturové mapy. Pro automatické provedení operace můžeme využít funkce *Smart UV Project*. Tento algoritmus je náročný a zabere mnoho času.

Vytvoření texturové mapy umožňuje i Meshlab – jeho výsledky však nebyly dost kvalitní.

2.4.2 Nabarvení texturové mapy

Vstup: původní .ply model ve vysokém rozlišení, model s nižším počtem polygonů a údaji o parametrizaci textury
Výstup: model .ply nebo .obj spolu s texturou

Po uložení modelu s texturovou mapou z Blenderu jej znova otevřeme pro další práci v Meshlabu. Ten však používá trochu jiný způsob kódování informace mapy. To však není problém, pomocí *Per Vertex Map Into Per Wedge* lze mapu převést do požadovaného formátu.

Poté načteme do Meshlabu i původní soubor ve vyšším rozlišení a aplikujeme filtr *Texture* → *Transfer: Vertex Attributes To Texture*. Jako zdrojový model zadáme původní model a jako cílový model s parametrizací. Musíme také určit velikost textury. Je zvykem udávat velikosti textur jako mocniny dvou, protože grafické karty jsou optimalizovány na tyto rozměry. Tedy například 512, 1024, 2048, 4096... Je možné udělat i více detailní textury, ale ty jsou již velmi náročné na výkon a na některých zařízeních nemusejí vůbec fungovat.

Také je dobré zaškrtnout *Assign Texture*, aby byla nová textura přiřazena k modelu.

Kvůli následujících kroků je dobré model exportovat ve formátu .obj, protože s ním pak budeme pracovat.

2.5 Optimalizace modelu pro webové použití

V předchozí kapitole jsme vytvořili decimovaný, otexturovaný model, který však je zatím ve formátu, který není určen pro webovou reprezentaci. Provedeme tedy v této kapitole několik dalších optimalizací, které pomůžou tomu, aby webový zážitek při prohlížení modelu byl co možná nejlepší.

Formát 3D modelu, který je užíván na webu a byl pro toto použití přímo vyvinut, se nazývá .gltf (GL Transmission Format) a jeho ještě více úsporná verze .glb. Je to podobné, jako s obrázky na webu, které bývají ve formátu .jpg. .glb reprezentuje data úsporně a navíc umožňuje kompresi dat, tzv. DRACO kompresi, která zajistí to, že se přes síť pošle komprimovaný model, což šetří data uživatele i serveru, který se pak v prohlížeči uživatele rozbalí, aby mohl být zobrazen.

2.5.1 Přeměna pomocí skriptu

Vstup: decimovaný model ve formátu .obj nebo .fbx
Výstup: model .glb, připraven na použití na internetu

Pro systém Linux lze využít skript v jazyce BASH, který převede .obj soubor na optimalizovaný .glb soubor připravený k použití v internetovém frameworku. Skript se nachází v příloze.

Použití:

```
obj2optimizedGlb.sh input.obj output_name.glb
```

2.5.2 Přeměna pomocí více nástrojů

Zde je vlastně popsáno to, co zvládá skript výše.

Potřebné programy:

- **NodeJS a NPM** – jsou potřeba k instalaci balíčků zmíněných dále. Ke stažení zde.
- **obj2gltf** – nástroj do příkazové řádky pro převod .obj do .gltf. Je možné místo něj použít některý konvertér na internetu – ty jsou však méně výkonné. Instalace pomocí NPM.
- **gltf-pipeline** – balíček umožňující různé operace nad .gltf a .glb soubory. Instalace pomocí NPM.
- **Nástroj pro kompresi .jpg** – například jpegoptim, nebo kterýkoliv jiný program.

Nejprve převedeme .obj soubor na .glb. Toho docílíme pomocí příkazu:

```
obj2gltf --binary -i model.obj -o temp.glb
```

Přepínač *binary* zajistí převod na .glb. Můžeme jej vynechat a převést na gltf. Následuje příkaz:

```
gltf-pipeline -i temp.glb -o tmp/tmp.gltf --separate --json
```

který rozloží .glb soubor na jeho jednotlivé části do složky tmp, zejména nás zajímají soubory textur. Ty posléze převedeme do .jpeg, pokud byly v jiném formátu a aplikujeme na ně kompresi v libovolném nástroji. Převedené textury necháme ve složce tmp a staré vymažeme.

Nakonec vše sestavíme do finálního .glb souboru pomocí příkazu:

```
gltf-pipeline -i tmp/tmp.gltf -o model.glb --binary --draco
--draco.compressionLevel=10
```

Zároveň tak aplikujeme DRACO kompresi. Přepínač *draco.compressionLevel* nastavuje úroveň komprese – vyšší číslo značí vyšší kompresi.

Složku *tmp* i *tmp.glb* můžeme vymazat a *model.glb* je náš výsledný model, který můžeme umístit na web například pomocí frameworku níže.

3 Webový framework

Tato část popisuje základní funkcionalitu, schopnosti a technologie, na kterých je postaven framework pro prohlížení anotovaných 3D modelů na internetu.

3.1 Funkcionalita

Webový framework umožňuje prohlížení 3D modelů, anotací a galerií na internetu a to pro všechny typy zařízení, které výkonově zvládnou zobrazení daného modelu, včetně chytrých telefonů a tabletů. Může být nastavený do dvou módů zobrazení – interierového a vnějšího.

V interierovém zobrazení se uživatel pohybuje prostorem pomocí šipek či kláves WASD a myši (popřípadě joystickem na zařízeních bez fyzické klávesnice). Slouží hlavně k prohlížení 3D modelů zevnitř – například interiéru kostelu. Uživatel se může pohybovat ve všech směrech a není v pohybu nijak limitován.

Vnější zobrazení slouží k prohlížení objektů zvenčí – například sochy, exteriéru památek atp. Uživatel se nepohybuje prostorem, ale otáčí pomocí myši samotný objekt.

Framework podporuje zobrazení ve více jazycích. Je tedy možné mít anotace přeloženy do různých jazyků.

3.1.1 Anotace

Anotace jsou na modelu znázorněny arabskými číslicemi. Po kliknutí na číslici se zobrazí příslušná anotace. Součástí anotace může také být galerie obrázků, kdy každý z nich může mít svůj popisek a po rozkliknutí se otevře v novém okně v plné velikosti.

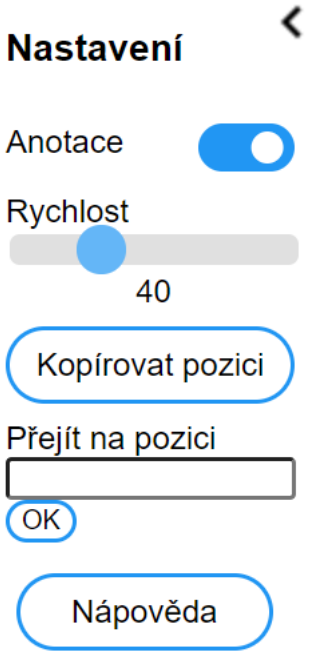
Anotace mohou obsahovat v podstatě jakýkoliv HTML obsah a jsou tedy značně flexibilní.

Obr. 2: Anotace

3.1.2 Ovládací prvky

Většinu nastavení skýtá postranní ovládací panel, který se zobrazí po kliknutí na šipku v levém horním rohu obrazovky. Umožňuje přepnout jazyk (pokud je k dispozici), vypnout a zapnout zobrazování anotací, nápovědu, změnu rychlosti pohybu v prostředí a kopírovat pozici kamery a pak na ni přejít. Přičemž poslední dvě možnosti jsou k dispozici pouze v modelu s interiérovým zobrazením. Při prohlížení na zařízení se senzorem polohy se v panelu navíc nachází tlačítko pro zapnutí a vypnutí tohoto senzoru.

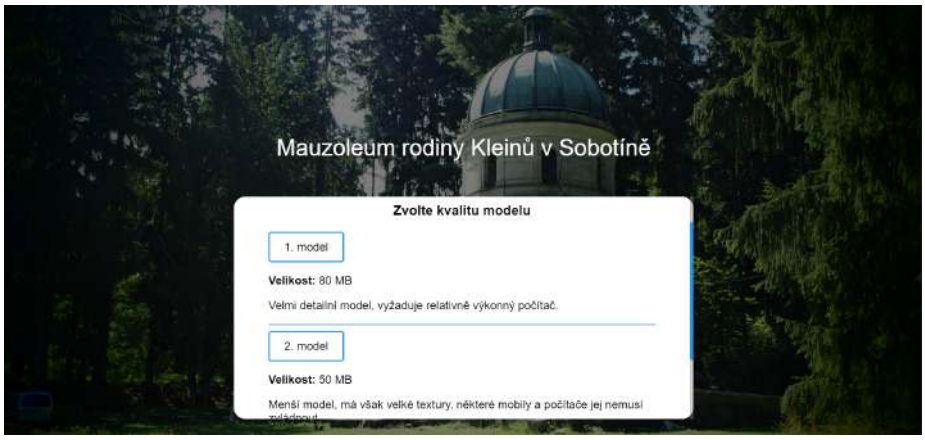
Další prvek, který se nachází pouze v režimu interiéru, je tlačítko se znakem domu, které se nachází v levém dolním rohu obrazovky. To slouží k návratu na původní pozici, pokud se uživatel například ztratí.



Obr. 3: Ovládací panel

3.1.3 Načítací obrazovka

Načítací obrazovka zobrazuje název modelu, vybranou fotku a informace o tom, kolik modelu se již stáhlo. Dále některé modely mohou být dostupné ve více rozlišeních a u nich je navíc na začátku možnost volby kvality.



Obr. 4: Volba kvality modelu

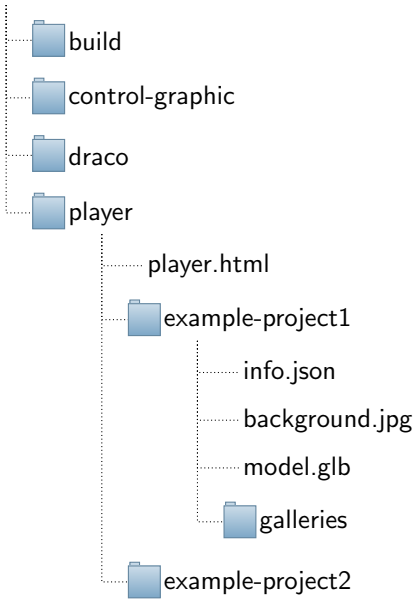
3.2 Technické detaily aplikace

3.2.1 Použité technologie

Celý framework je implementován pomocí technologií na straně uživatele a klade tedy minimální technické požadavky na server. K zobrazení modelu a tlačítek anotací je použit JavaScriptový framework A-Frame, který je založen na WebGL. Dále je využit šablonový systém Handlebars k implementaci modularity a parametrizace aplikace. Ta sestává z hlavního HTML souboru a JavaScriptových skriptů, které implementují veškerou logiku. Aplikaci je ve formě URL parametru předán název složky, ve které jsou detaily konkrétní 3D scény ve formátu JSON a samotným modelem a obrázky (např. *... player.html?projekt1*).

3.2.2 Adresářová struktura

Celá aplikace na serveru vypadá zhruba takto:



Složky *build, control-graphic* a *draco* obsahují potřebný kód aplikace, s těmi není potřeba manipulovat. Ve složce *player* se nachází hlavní .html soubor aplikace a složky s 3D projekty.

Každý projekt musí mít soubor *info.json* s detaily o scéně a *background.jpg*, což je soubor, který se zobrazí na načítací obrazovku. Specifikace *info.json* je určena ukázkovým souborem s komentáři. Ve složce *galleries* jsou případné galerie.

3.2.3 Galerie

K anotacím je možno připevnit mnoho fotek pomocí galerií. Navíc každá fotka může obsahovat svůj vlastní popisek. Galerie jsou implementovány s ohledem na

efektivitu stahování. Využívají proto tzv. líné načítání. To znamená, že fotky se načítají jen tehdy, když jsou potřeba (např. po otevření anotace, přejetí na další fotku atd.). Toto výrazně zrychluje načítání celého projektu, pokud je v něm obsaženo velké množství fotek.