VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZÁSUVNÝ MODUL DO BLENDERU PRO PŘEVOD MODELŮ NA VEKTOROVOU GRAFIKU

BLENDER PLUGIN FOR CONVERSION OF MODELS TO VECTOR GRAPHIC

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JIŘÍ KOPÁČEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

144955

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Kopáček Jiří, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Softwarové inženýrství
Název:	**Zásuvný modul do Blenderu pro převod modelů na vektorovou grafiku**
Kategorie:	Počítačová grafika
Akademický rok:	2022/23

Zadání:

1. Naučte se základy práce s 3D modelovacím programem Blender (verze 3.0 a výše).
2. Seznamte se se skriptováním v Blender Python API.
3. Navrhněte vylepšení stávajícího pluginu pro převod modelů na vektorovou grafiku (svg).
4. Implementujte navržené rozšíření pluginu a demonstrujte jejich použití na různých typech dat.
5. Zdokumentujte a zveřejněte výsledný plugin pro použití dalšími uživateli.
6. Plugin zveřejněte a vytvořte video prezentující výsledky vaší práce.

Literatura:

- Blender API documentation
- Juntao, Li, Baoquan, Liu. Primitive z-sorting. EP3794561A1 European Patent Office.
- Sutherland, Ivan, Hodgman, Gary W. Reentrant Polygon Clipping. Communications of the ACM, vol. 17, pp. 32–42, 1974.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, experimenty vedoucí k plné implementaci.

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

Vedoucí práce:	**Milet Tomáš, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

Tato práce řeší návrh a implementaci několika rozšíření už dříve existujícího modulu (pluginu) pro software Blender. Tento plugin dříve sloužil k převodu 3D modelů ve scéně Blenderu na soubor 2D vektorové grafiky ve formátu SVG. Mezi nová rozšíření patří podpora převodu také pro křivky, texty, nástroj Grease Pencil či anotace a jejich vzájemné hloubkové řazení a také podpora třídění do kolekcí, využívání kamerových objektů a tvorba animací. Mezi vylepšení existujících částí patří kompletní přepracování uživatelského rozhraní, rozšíření možností novým materiálovým systémem a oprava některých starších chyb kódu. Výsledkem implementace je nová verze pluginu, která umožňuje uživateli efektivně tvořit složitější vektorové obrázky v Blenderu s více možnostmi stylizace výstupu.

Abstract

This thesis deals with the design and implementation of several extensions for an existing module (plugin) for Blender software. This plugin could be used for conversion of 3D models in a Blender scene to a 2D vector graphics file in the SVG format. New extensions include the support of curve, text, Grease Pencil or annotation conversions and their depth sorting, as well as support of collection sorting, using camera objects and creating animations. Improvements of existing parts include a complete user interface rework, improving capabilities with a new material system and several fixes to existing code. The result of the implementation is a new version of the plugin which allows the user to efficiently create more complex vector graphics in Blender with enhanced output styling options.

Klíčová slova

převod modelu, převod křivky, 3D na 2D, 3D na SVG, model na SVG, křivka na SVG, vektorová grafika, animace, hloubkové řazení, Blender, plugin, addon, Python, SVG, CSS

Keywords

model conversion, curve conversion, 3D to 2D, 3D to SVG, model to SVG, curve to SVG, vector graphics, animation, depth sorting, Blender, plugin, addon, Python, SVG, CSS

Citace

KOPÁČEK, Jiří. *ZÁSUVNÝ MODUL DO BLENDERU PRO PŘEVOD MODELŮ NA VEKTOROVOU GRAFIKU*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

ZÁSUVNÝ MODUL DO BLENDERU PRO PŘEVOD MODELŮ NA VEKTOROVOU GRAFIKU

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Tomáše Mileta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Kopáček
23. dubna 2023

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Miletovi, Ph.D. za pravidelné konzultace, a především za původní myšlenku, na které bylo toto zadání a celkově práce postavena. Také bych chtěl poděkovat své rodině za veškerou podporu během studia. Dále bych chtěl jmenovitě poděkovat pár následujícím legendám v abecedním pořadí:

Lukáš Chochrun. Ten, kde to vše začalo.

Jan Kopáček a Martin Novák. Za vybudování závislosti na virtuálním světě u osmiletého dítěte. Nevím, jestli to byl dobrý nebo etický nápad, ale dospělo to až sem.

Jakub Konopa. Za všechny jeho intelektuálně stimulující podněty, kterými nás všechny neustále proti naší vůli zahrnuje. A za adoptování Meziho.

Lukáš Macek. Nevím za co.

David Mačuda. Dealer brambor. Homie. Můj oblíbený Pražan. Přední fanoušek Slavie nebo tak něco, nevím, nesleduju hokej.

Ondřej Marcián. The man, the myth, the legend. Roudží OG. Kulturně založen.

Václav Matuška. Neznám žádného Václava.

Jan Mezírek. Em tři z jedný. Šampion čtvrté ligy slovního fotbalu. Nejsilnější Vysočan.

Jan Mlýnek. Anticlutch. Ichtyloptyl. Zaprodán Epicu. Známými také nazýván Žrout. Nemůže za to.

Jakub Pojsl. Kolega bývalý, přítomný a snad i budoucí.

Martin Polášek. Škoda slov.

Robert Ross. Tribunus Plebis. Jogging sensei. Velký fanda zeměpisu.

Jonáš Sasín. Kolega bývalý.

Štěpán Šindelka. Lol. [TODO: Vymyslet, za co tomuto člověku děkovat]

Aleš Undercutter. Tomu neděkuju, ale píšu si ho sem, abych nezapomněl, co mi udělal.

Náhodní lidi, které jsem kdy potkal v MMO hrách. Většinu z nich jsem neznal dlouho a nevím odkud pocházeli, ale z nějakého důvodu si je pamatuju.

Obsah

Kapitola 1

Úvod

V oblasti počítačové grafiky existují dva základní přístupy k reprezentaci 2D obrazu, kterými jsou rastrová a vektorová grafika. Rastrová grafika definuje hodnoty barev bodů v daném rozlišení a při snaze tento obraz zvětšit dochází ke ztrátě kvality obrazu. Obraz vektorové grafiky se naopak skládá z přesně definovaných tvarů, které jsou vykresleny až při jejím zobrazení v libovolném rozlišení. Díky tomu lze dosáhnout stejné kvality obrazu i při zmenšování a zvětšování, což je vhodné například v prostředí webových stránek.

Protože je však potřeba přesně definovat každý základní útvar, tvorba vektorových obrázků může být poměrně pracná a náročná, obzvlášť v případě, kdy je cílem vyobrazit složitou scénu či objekt v prostoru. V takové situaci je nezbytné brát v potaz úhel pohledu či perspektivu, čímž se tvorba stává ještě více složitější. Tvorba takových obrázků by byla snazší a praktičtější, pokud by bylo možno situaci namodelovat ve 3D a poté „vyfotit“. Takovým způsobem lze ale získat pouze rastrový obrázek v omezeném rozlišení.

Řešením tohoto problému se zabývala bakalářská práce [10], na kterou tato diplomová práce navazuje. V původní práci byl využit nástroj Blender jakožto 3D modelovací software, který lze rozšiřovat tvorbou tzv. pluginů (Add-onů, zásuvných modulů) s pomocí rozhraní pro programování aplikace – Blender Python API (dále pouze API). Výsledkem práce byl plugin, který rozšiřoval funkcionalitu Blenderu o možnost převodu modelů ve 3D scéně na obrázky vektorové grafiky, a to v podobě souborů formátu SVG (Scalable Vector Graphics).

Cílem této práce bylo využít předchozích i nově získaných znalostí Blenderu, jeho API a tvorby pluginů k rozšíření možností původního pluginu. Bakalářská práce byla zaměřena spíše na hlubší řešení některých konkrétních problémů vykreslování jako například hloubkové řazení, nežli na poskytování většího množství grafických nastavení pro úpravu výsledného obrazu. Tato diplomová práce naopak není zaměřena „do hloubky“ na řešení specifického problému, ale „do šířky“, zabývá se tedy návrhem a implementací většího počtu různě komplexních rozšíření, které by obecně měly přispívat ke zkvalitnění možností vizuální úpravy výstupu či usnadnění tvorby složitějších obrázků. Mimo jiné také byla část rozšíření zaměřena na podporu převodu nových typů objektů v Blenderu, nejen modelů.

Na úvod práce jsou blíže popsána prostředí relevantní pro tuto práci a je zrekapitulován současný stav a možnosti původních pluginů Blenderu pro převod modelů do vektorové grafiky v kapitole 2. Dále se práce zabývá jednotlivými rozšířeními v kapitole 3. Seznam všech rozšíření i s krátkým popisem a odkazy lze nalézt v podkapitole 3.1. Dále tato kapitola obsahuje 13 podkapitol, z nichž každá představuje jiné rozšíření. Popis většiny rozšíření je v rámci těchto podkapitol rozdělen do 5 sekcí: současný stav a cíl, formáty dat a teorie, návrh, implementace, výsledky. Před závěrem je krátce zhodnocen výsledný stav pluginu v samostatné kapitole 4. Celkově je na závěr shrnuta celá práce v kapitole 5.

Kapitola 2

Shrnutí současného stavu

Tato kapitola slouží k obecnému seznámení se s prostředími relevantními pro tuto práci jako jsou Blender či formát SVG. Dále se zabývá také dvěma už existujícími pluginy, jejich možnostmi a určením směru dalšího vývoje. Prvním původním pluginem je skript „Viewport to SVG“ od uživatele Liero[1] z roku 2013 (aktualizován v roce 2019 pro verzi Blenderu 2.80), dále označován jako plugin A. Druhým je plugin „Blender Model To SVG[2]“ tvořený v rámci bakalářské práce [10] z roku 2021 pro verzi Blenderu 2.90, dále označován jako plugin B.

2.1 Úvod do souvisejících prostředí

Práce se na jedné straně převodu pohybuje v prostředí Blenderu, jeho API a podpory tvorby pluginů. Na straně druhé poskytuje výstupní vektorovou grafiku ve formátu SVG, se kterým částečně souvisí také například jazyky XML, HTML či CSS. Tato podkapitola tedy slouží k jejich stručnému a obecnému úvodu.

Blender, Blender API a pluginy Blenderu

Blender[3] je svobodná a otevřená sada nástrojů pro tvorbu 3D počítačové grafiky a podporuje téměř všechny její aspekty od modelování a animací přes simulace až po výsledné vykreslování, mimo jiné zahrnuje také úpravu videí či tvorbu her. Je víceplatformní a použitelný na počítačích se systémy Windows, Linux či Macintosh. Jeho rozhraní využívá knihovnu OpenGL.

Funkcionalitu Blenderu je možno dále rozšiřovat využitím Blender API[4], které umožňuje vytvářet skripty v jazyce Python pro úpravu celé aplikace. Blender má v sobě obsažen interpret jazyka Python, který je načten při jeho spuštění a je aktivní po celou dobu běhu Blenderu. Uživatel tak může například upravovat jakákoliv data scény, modifikovat nastavení, vytvářet nové prvky uživatelského rozhraní nebo nové nástroje, protože kromě základních knihoven nabízí Python Blenderu také speciální knihovny poskytující přístup k datům a funkcím Blenderu a jeho API.

Skripty jazyka Python lze v Blenderu buďto přímo spustit, nebo importovat jako moduly. Přímé spuštění zahrnuje načtení skriptu do textového editoru a jeho spuštění nebo psaní příkazů do interaktivní konzole. Naopak importování skriptu jako nového rozšířují-

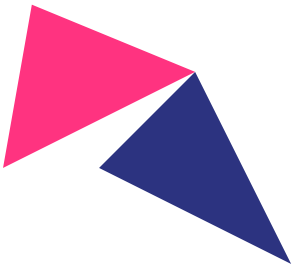
[1] Vlákno fóra Blenderu publikující plugin: https://blenderartists.org/t/svg-output-script/566412
[2] Veřejný repozitář pluginu: https://github.com/Craszh/BlenderModelToSVG
[3] O Blenderu: https://www.blender.org/about/
[4] O Blender API: https://docs.blender.org/api/current/info_quickstart.html

cího modulu lze provést například definicí skriptu jakožto pluginu, který je možno poté instalovat přímo do Blenderu a libovolně jej aktivovat a deaktivovat v seznamu pluginů v nastavení celé aplikace. Po aktivaci plugin okamžitě rozšíří Blender o definovanou funkcionalitu, kterou je v rámci této práce myšleno umožnění převodu objektů v Blenderu na vektorovou grafiku ve formátu SVG.

Formát SVG a související jazyky

Výstup je reprezentován souborem ve formátu SVG[5] (Scalable Vector Graphics), který je založen na jazyce XML a slouží k popisu 2D vektorové grafiky. Podporován je například téměř všemi současnými webovými prohlížeči. Jednotlivé vektorové útvary v něm jsou zapisovány jako prvky jazyka XML. Soubor ve formátu SVG tedy může mít například následující obsah, jehož vykreslení lze vidět na obrázku 2.1:

```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
     <polygon points="100,200 300,300 200,100" fill="rgb(44,51,128)"/>
     <polygon points="0,200 30,30 200,100" fill="rgb(255,51,128)"/>
</svg>
```



Obrázek 2.1: Příklad vykreslení souboru ve formátu SVG. Zdrojový soubor pro tento obrázek má stejný obsah, jako je uveden výše.

Vzhled jednotlivých útvarů lze upravovat pomocí prvků jazyka CSS, se kterými je formát SVG kompatibilní. Je možno tak například využít definice tříd CSS pro specifikaci vizuálních vlastností a přiřadit jednotlivé prvky SVG těmto třídám, čímž prvky zdědí jejich vlastnosti. Mimo jiné podporuje formát SVG ve webových prohlížečích také vkládání dalších jazyků jako například některé prvky jazyka HTML, či vkládání skriptů jazyka JavaScript.

Shrnutí

Tato podkapitola shrnula použitá prostředí, nástroje a formáty pouze obecně. Pro snadnější porozumění jsou specifičtější informace o Blender API a výstupním formátu vždy shrnuty až když jsou důležité, tedy v sekcích teorie podkapitol jednotlivých rozšíření, pro která jsou právě relevantní.

Následující podkapitoly úvodu do prostředí už blíže představují existující pluginy a jejich současné možnosti.

[5] Specifikace verze 1.1 formátu SVG: https://www.w3.org/TR/SVG11/

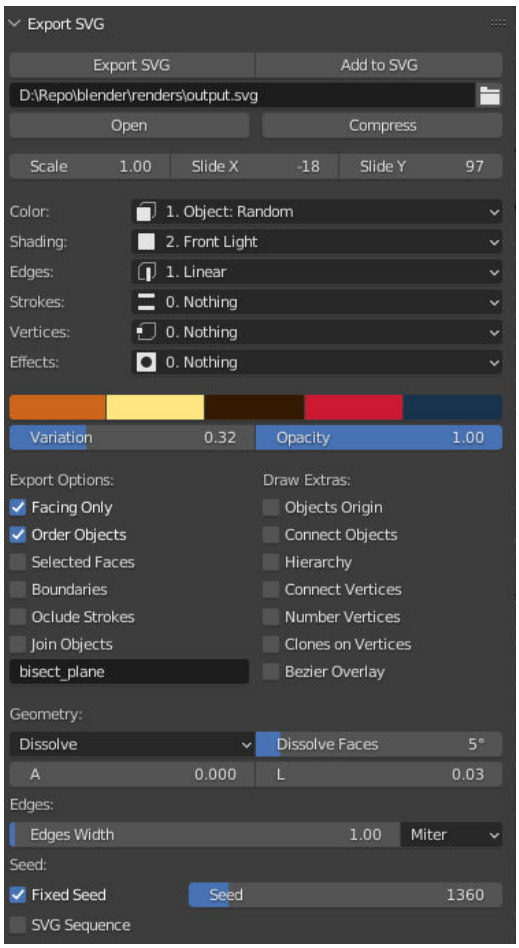
2.2 Plugin A – Viewport to SVG (2013)

Přestože tento plugin příliš neřeší některé problémy jako například přesnější řazení objektů, nabízí větší množství grafických přizpůsobení převodu oproti pluginu B, navíc dokáže také převádět křivky. Jeho uživatelské rozhraní lze vidět na obrázku 2.2.

Plugin umožňuje vybrat selekcí jednotlivé objekty některých podporovaných typů ve scéně Blenderu. Po kliknutí na tlačítko všechny tyto objekty převede a uloží do specifikovaného souboru ve formátu SVG. Pozice a celkově tvary objektů potom budou odpovídat tomu, jak objekty vypadaly při pohledu na 3D scénu uvnitř Blenderu při kliknutí na tlačítko.

V první hlavní sekci nabízí nastavení vzhledu různých prvků, která platí globálně, tedy jsou stejná pro všechny převáděné objekty. Tato nastavení zahrnují:

- Mód obarvování objektů (žádné, náhodné, dle zvolené palety, materiálu, ...),
- typ stínování (žádné, přední či zadní světlo, stínování podle hloubky stěny, ...),
- styl hran (žádné, souvislé, čárkované, ...),
- efekty tahů (žádné, zakřivení, protáhnutí tahu, šířka podle úhlu pohledu, ...),
- styl vrcholů (žádné, viditelné, různě velké podle úhlu pohledu, ...),
- speciální efekty (žádné, exploze objektu, záměna stěn za čtverce, kruhy, ...).



Obrázek 2.2: Rozhraní pluginu A

Po sekci s výběrem palety barev, variace barev a průhlednosti následují další nastavení pro export jako například vynechání odvrácených stěn, řazení objektů podle hloubky, převod pouze vybraných stěn modelů a další.

Mimo to lze také nastavit dokreslování dodatečných prvků jako například vyznačení středů objektů, spojení objektů úsečkami a propojení či očíslování vrcholů.

Poslední sekce zahrnuje simplifikaci modelů před převodem a dodatečné parametry dříve zvolených efektů. Na závěr poskytuje plugin možnost převodu animace Blenderu. Nejedná se však o nijak animovaný soubor SVG, ale o převod každého definovaného snímku na časové ose scény Blenderu do samostatného souboru SVG.

2.3 Plugin B – Blender Model To SVG (2021)

Druhý plugin vycházející z bakalářské práce nenabízí příliš širokou škálu možností úpravy vzhledu výstupu, neboť práce řešila hlavně problematiku hloubkového řazení zaklíněných objektů souvisejících s jejich korektním vykreslením. Hlavní výhody oproti pluginu prvnímu

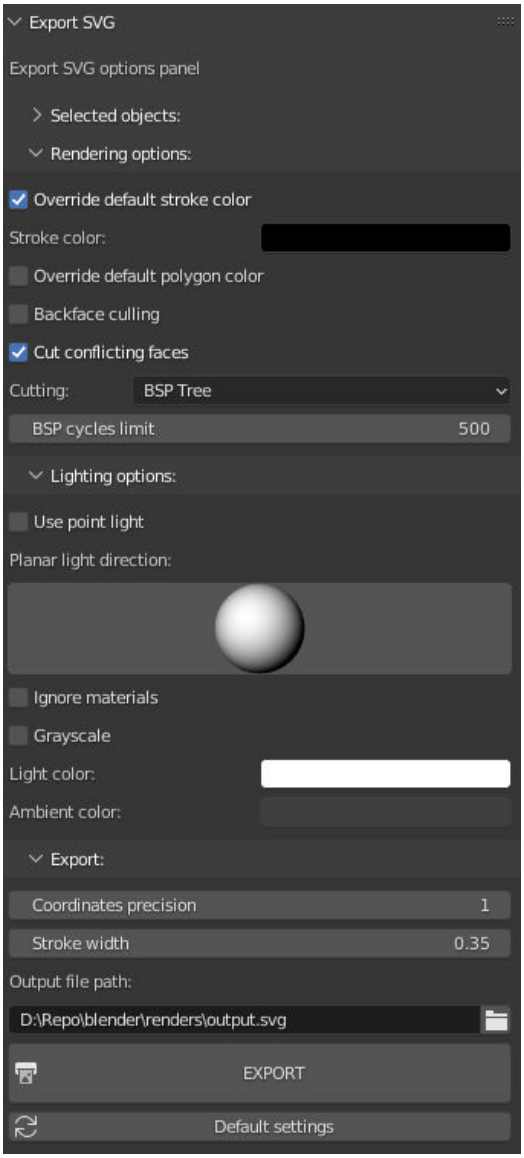
zahrnují rozsáhlejší možnosti nastavování zdroje světla a především přesnější (ačkoliv ne zcela přesné) řezání a řazení zaklíněných modelů a jejich stěn. Uživatelské rozhraní lze vidět na obrázku 2.3.

Základní funkcionalita je stejná jako u prvního pluginu A. Selekcí lze vybrat jednotlivé objekty ve scéně Blenderu a kliknutím na tlačítko je všechny převést a uložit do specifikovaného souboru, opět ze stejného pohledu, jako byl pohled na scénu v prostředí Blenderu při kliknutí na tlačítko. Příklad vstupu a výstupu převodu lze vidět na obrázku 2.4.

V první hlavní sekci nastavení „Rendering options“ lze přizpůsobovat převod modelů. Lze nastavit globální barvu okrajů všech výsledných polygonů (mnohoúhelníků, dále bude používán pro stručnost termín polygon) v SVG souboru, která je při výchozím nastavení stejná jako jejich výplň. Stejně tak lze nastavit globální barvu výplní, které jsou při výchozím nastavení určovány barvou materiálu (v materiálové sekci Viewport Display: Color) a osvětlením.

Další nastavení se týkají vynechání odvrácených stěn modelů při převodu a volby, zdali vzájemně zaklíněné stěny modelů řezat na více stěn tak, aby nedocházelo ke konfliktům. K tomu samotný plugin nabízí 3 různé algoritmy pro detekci a řešení konfliktů. Žádný z nich neřeší problém perfektně, ovšem většinou mohou vést ke zlepšení přesnosti výsledků. Na závěr této sekce lze také zvolit, jakým způsobem polygony řadit do souboru.

Další sekce nastavení „Lighting options“ umožňuje volbu mezi bodovým a směrovým světlem. Také lze nastavit vypnutí vlivu materiálů na výslednou barvu či převod na černobílý obraz. Nakonec lze také především nastavit barvu zdroje světla a barvu okolního světla.

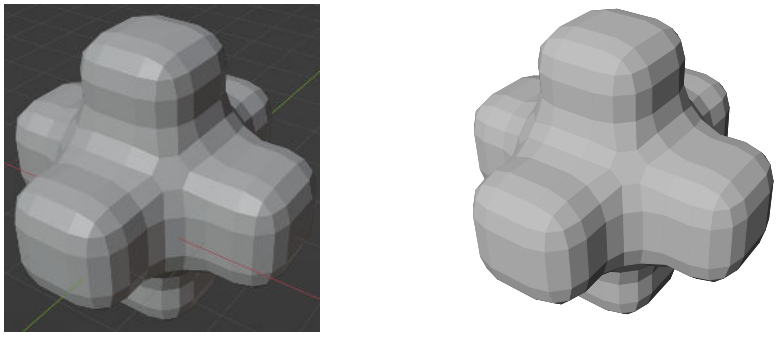


Obrázek 2.3: Rozhraní pluginu B

V závěrečné sekci „Export“ jsou už pouze zbylá globální nastavení jako přesnost souřadnic zapsaných do souboru SVG, šířka okrajů polygonů a cesta k souboru, do kterého má být výsledek převodu zapsán. Posledními prvky uživatelského rozhraní jsou tlačítko pro zahájení převodu a také tlačítko pro resetování nastavení do výchozího stavu.

2.4 Určení směru vývoje

Na základě popisů funkcionalit obou pluginů lze usoudit, že každý z nich je trochu jinak zaměřen. První plugin A zcela nepochybně nabízí výrazně bohatší možnosti grafické stylizace svých výstupů společně s možností převodu křivek. Druhý plugin B se více zabývá



Obrázek 2.4: Příklad funkcionality pluginu. Nalevo lze vidět pohled na scénu v Blenderu, při kterém byl převod proveden. Napravo lze vidět vykreslený výstupní soubor SVG, který plugin vygeneroval.

přesností vykreslení při složitých situacích nastávajících u zaklíněných modelů a zároveň nabízí volnější nastavení osvětlení.

Ideální by tedy byla existence takového pluginu, který kombinuje výhody obou předchozích. Jako první možné řešení se jistě jeví spojení těchto pluginů dohromady. Zde však nastává několik problémů. Prvním ale ještě snad překonatelným by byl souhlas původního autora prvního z pluginů, který tento plugin vytvořil v roce 2013 a pouze jednou aktualizoval v roce 2019 pro novou verzi Blenderu.

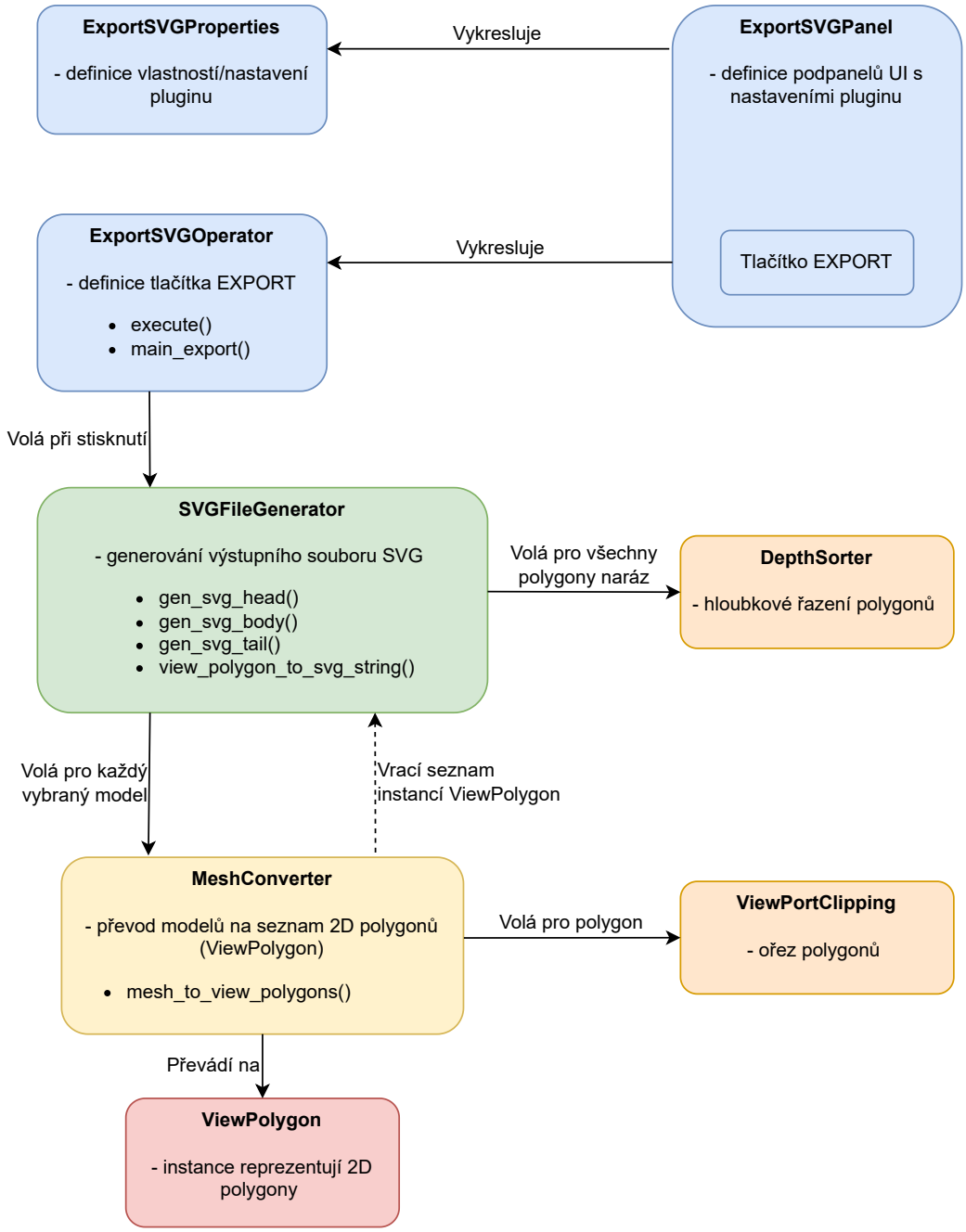
Tím zásadnějším je však problém se samotnými zdrojovými kódy a architekturou pluginů. Zdrojový kód prvního pluginu zdánlivě nebere příliš ohled na udržovatelnost a rozšiřovatelnost. Samotný kód není příliš kvalitně dokumentován, je v něm mícháno více jazyků u komentářů a identifikátorů a mnoho proměnných má identifikátory dlouhé pouze pár znaků. Především jsou však pluginy odlišné v architektuře a celkovému přístupu k převodu. Vyjmenování těchto překážek už by vyžadovalo příliš detailní rozbor implementace, pro shrnutí lze však říct, že spojení obou pluginů dohromady by ve výsledku spíše znamenalo rozšíření jednoho pluginu o funkcionalitu druhého bez ušetření významného množství úsilí.

Tato skutečnost tedy rozhodla o dalším směru vývoje, který se stal obsahem této práce a kterým je rozšíření druhého pluginu B o vetší množství dodatečných funkcionalit, jež značně vylepšují především možnosti stylizace výstupu. Rozšířený plugin byl navíc implementován na aktuálnější verzi Blenderu 3.2, základní chod byl testován až pro verzi 3.5.

2.5 Současná architektura pluginu B

Protože se práce zabývá rozšiřováním pluginu B, je při implementaci jednotlivých rozšíření a jejich popisu navazováno na už existující architekturu. Aby byl popis implementace v rámci textu kompletní, je potřeba v této podkapitole architekturu původního pluginu alespoň obecně shrnout a uvést hlavní části jeho struktury.

Nejlépe lze architekturu shrnout následujícím diagramem na obrázku 2.5, který popisuje její hlavní třídy se vzájemnými vazbami a případně zmiňuje nejvýznamnější metody jejich rozhraní.



Obrázek 2.5: Zjednodušený diagram původní architektury pluginu. Uživatelské rozhraní tvoří třídy ExportSVGPanel (panel), ExportSVGProperties (jednotlivá nastavení) a ExportSVGOperator (tlačítko pro převod a jeho obsluha). Jádro převodu tvoří třída SVGFileGenerator starající se o generování jednotlivých částí souboru SVG. K tomuto účelu tato třída využívá třídu MeshConverter pro převod modelů na 2D polygony (reprezentované instancemi třídy ViewPolygon). Dalšími podpůrnými třídami jsou ViewPortClipping pro ořez a DepthSorter pro hloubkové řazení polygonů.

Kapitola 3

Rozšíření

V následující kapitole jsou na úvod představena všechna hlavní rozšíření v pořadí jejich průběžné implementace. Toto pořadí bylo zvoleno z důvodu návaznosti textu, neboť některá z pozdějších rozšíření občas závisí na těch dřívějších.

Ve všech dalších podkapitolách je každé rozšíření ve stejném pořadí nejprve krátce popsáno z hlediska aktuálního stavu pluginu a specifikací cíle, kterého má dosáhnout. Poté vždy následuje souhrn vstupních a výstupních dat potřebný k návrhu rozšíření zahrnující především poznatky o formátu SVG a o Blenderu a jeho API či architektuře, tato sekce také může případně zmiňovat obecnou teorii z počítačové grafiky, pokud je taková teorie nutnou znalostí pro návrh a implementaci. Po tomto souhrnu následuje návrh vysvětlující funkcionalitu a způsob práce uživatele s navrhovaným rozšířením, včetně jeho zakomponování do už existující struktury pluginu. Dále je popsána implementace daného rozšíření, všechny popisy implementací v rámci této práce jsou prezentovány formou diagramu architektury s komentářem. Na závěr každé podkapitoly jsou formou obrázků prezentovány výsledky dosažitelné díky danému rozšíření.

3.1 Přehled rozšíření

Tato úvodní podkapitola slouží jako stručný přehled a rozcestník implementovaných rozšíření obsahující jejich seznam s krátkým popisem a ilustračním obrázkem 3.1. V rámci práce byla implementována v daném pořadí následující rozšíření.

1. **Převod křivek** (podkapitola 3.2). Rozšíření umožňuje převod objektů v Blenderu typu „CURVE“ na prvky formátu SVG <path> s různými možnostmi přizpůsobení vizualizace.

2. **Převod textu** (podkapitola 3.3). Rozšíření umožňuje převod objektů v Blenderu typu „FONT“ s různými možnostmi přizpůsobení vizualizace. Je podporováno také více typů převodu na formát SVG. Lze převádět text jako model na prvky <polygon>, jako soubor křivek na prvky <path> nebo jako řetězec na prvek <text>

3. **Univerzální hloubkové řazení různých typů** (podkapitola 3.4). Rozšíření řeší univerzální způsob hloubkového seřazení objektů po implementaci předchozích dvou rozšíření, kvůli kterým jsou nově v procesu převodu míchány různé typy objektů.

4. **Převod kolekcí** (podkapitola 3.5). Rozšíření přidává podporu převodu kolekcí objektů v Blenderu s využitím seskupovacích prvků formátu SVG <g>. Zároveň umož-

ňuje nastavení způsobu hloubkového řazení kolekcí. Jedno z těchto nastavení také replikuje funkcionalitu vrstev v klasických editorech 2D grafiky.

5. **Podpora kamerových objektů** (podkapitola 3.6). Rozšíření přidává podporu objektů v Blenderu typu „CAMERA“, čímž umožňuje pořizovat výsledné obrázky nejen z aktuálního pohledu na scénu, ale také z pohledu kamerových objektů. Zároveň je umožněn převod z více kamer zároveň, díky čemuž lze jedním tlačítkem vytvořit více vektorových obrázků z různých pohledů současně.

6. **Podpora výplní vzorkem** (podkapitola 3.7). Rozšíření přidává možnost nastavit jako výplň objektů libovolný nakopírovaný vzorek ve formátu SVG <pattern>.

7. **Přepracování uživatelského rozhraní** (podkapitola 3.8). Rozšířením je kompletní přepracování uživatelského rozhraní do nové podoby lépe podporující všechna nastavení týkající se nově implementovaných rozšíření. Původní rozhraní totiž nebylo pro všechna nová rozšíření vhodně strukturováno a po přidání nových prvků bylo nepřehledné a neintuitivní.

8. **Materiálový systém** (podkapitola 3.9). Rozšíření replikuje funkcionalitu materiálů v Blenderu pro úpravu vzhledu jednotlivých převáděných objektů. Umožňuje definovat vizuální atributy SVG pro každý materiál Blenderu. Objekty potom mají na vektorovém obrázku takové vizuální vlastnosti, jaké měl nastaven jejich materiál v Blenderu. To umožňuje uživateli mnohem granulárnější přístup k úpravám vzhledu než jen globální nastavení společné pro všechny objekty. K tomuto účelu je použit prvek jazyka CSS <style>, který lze využít také při vykreslování SVG.

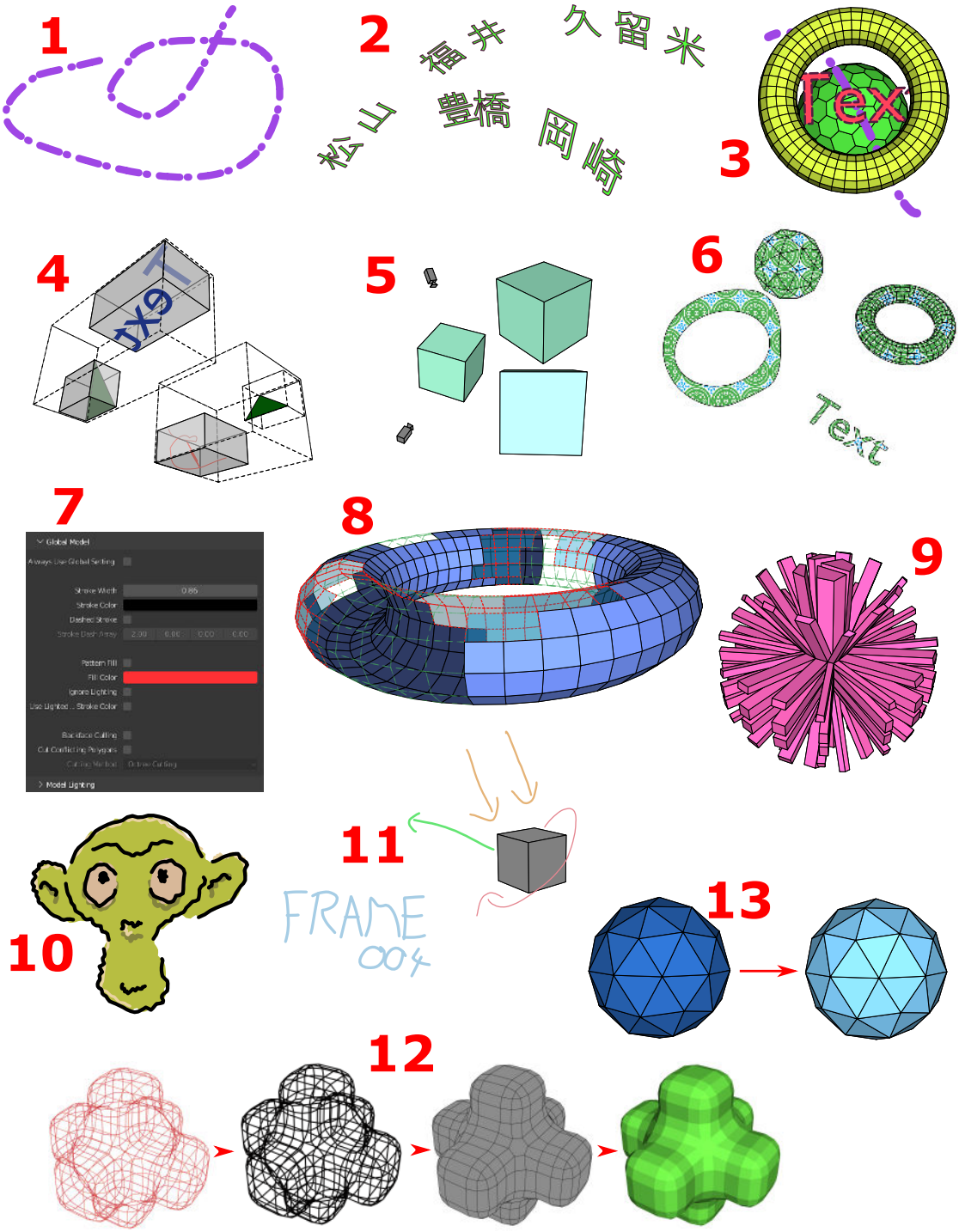
9. **Podpora evaluace** (podkapitola 3.10). Jedná se o menší rozšíření umožňující převádět objekty po evaluaci, tedy s ohledem na aktivní animace či modifikátory.

10. **Převod Grease Pencil** (podkapitola 3.11). Rozšíření umožňuje převod objektů v Blenderu typu „GPENCIL“ na prvky formátu SVG <path> s různými možnostmi nastavení vizualizace. Brána je v potaz i vrstvová struktura těchto typů objektů, kterou se snaží rozšíření při převodu zachovat.

11. **Převod anotací** (podkapitola 3.12). Rozšíření umožňuje převod anotací v Blenderu na prvky formátu SVG <path>. Protože anotace mají velmi málo nastavení vizualizace, rozšíření se snaží o přesný převod se zachováním všech vizuálních vlastností včetně vrstev a priorit.

12. **Podpora animací** (podkapitola 3.13). Rozšíření umožňuje tvorbu animovaných vektorových obrázků s použitím prvků pro animaci jazyka CSS. Pro každý materiál umožňuje definovat jeho vizuální vlastnosti jako sérii klíčových snímků, mezi kterými poté probíhá na vektorovém obrázku interpolace dle nastavených vlastností animace.

13. **Korekce barev a další úpravy** (podkapitola 3.14). Rozšíření je souhrnem menších úprav, ladění, vylepšování architektury a řešení některých předchozích nedostatků. Jedná se spíše o vylepšování méně kvalitních částí kódu a proto není popisováno příliš detailně. Jednou z významnějších a viditelnějších úprav je však korekce převodu barev, které původní plugin nezpracovával správně.



Obrázek 3.1: Ilustrační obrázek všech rozšíření, číslování odpovídá výše uvedenému seznamu: 1: převod křivek, 2: převod textu, 3: řazení různých typů dle hloubky, 4: převod kolekcí, 5: podpora kamer, 6: podpora vzorků, 7: nové uživatelské rozhraní, 8: materiálový systém, 9: podpora evaluace, 10: převod Grease Pencil, 11: převod anotací, 12: podpora animací, 13: korekce barev.

3.2 Převod křivek

V této podkapitole je přiblíženo rozšíření umožňující převod objektů v Blenderu typu „CURVE“ na prvky formátu SVG <path>.

Původní stav a cíl

Původní plugin nedokáže převádět křivky a v Blenderu objekty typu „CURVE“ při převodu ignoruje. Neobsahuje žádné metody pro práci s takovým typem ani vhodný způsob jejich reprezentace či převodu na formát SVG.

Cílem rozšíření je implementovat převod objektů tohoto typu, umožnit uživateli převod křivek modelovaných v prostoru scény na 2D křivky ve formátu SVG a do určité míry umožnit nastavování vizuálních vlastností pomocí atributů SVG.

Vstup, výstup a teorie

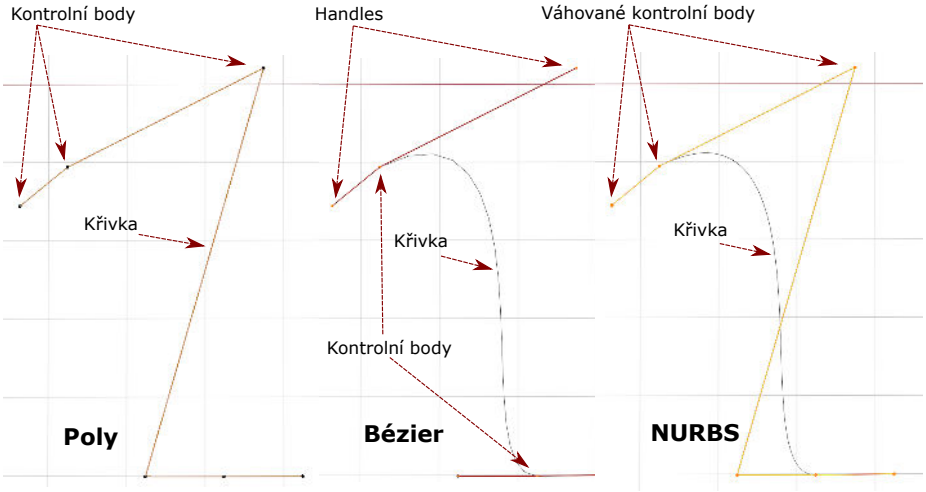
Obecně se ve 2D i 3D počítačové grafice křivky využívají v mnoha oblastech, od 2D a 3D modelování přes definici fontů až po určování drah objektů v rámci animačních sekvencí. Nebývají většinou definovány přesně bod po bodu, standardně jsou reprezentovány soustavou parametrů určité funkce, pomocí které jsou později při vykreslování samotné křivky počítány jednotlivé body její trajektorie [2, s. 177–178].

V rámci Blenderu existují různé typy objektů, které mohou být ve scéně umístěny. Jedním z těchto typů je typ „CURVE“, jehož objekty slouží pro definici výše zmiňovaných křivek a jsou taktéž vyjádřeny matematickými funkcemi. Křivky v Blenderu mají širokou škálu využití jako například alternativní reprezentace polygonového meshe (polygonová síť, dále jen mesh), reprezentace tras animovaných objektů či způsobů interpolace a další.

Důležitější je samotná struktura těchto objektů. Křivky jsou v Blenderu složeny ze substruktur zvaných spline, kterých může jedna křivka obsahovat více. Spline definuje tvar křivky a sám je složen z více kontrolních bodů (v Blenderu zvaných Control Points). Tyto kontrolní body na sebe navazují a tvoří spline, jehož tvar může být upravován selekcí a transformací těchto bodů, neboť právě mezi nimi probíhá interpolace výsledného tvaru celé křivky. Blender implementuje 3 typy splinů [5, sekce „Modeling > Curves“], jejich rozdíly lze vidět na obrázku 3.2.

V rámci formátu SVG je nejvhodnějším prvkem pro reprezentaci křivek prvek <path>. Tento prvek je velmi versatilní a využívá se obecně pro kreslení a definici tvarů, zároveň je však na poměry SVG formátu relativně složitý. Jednotlivé tvary vykresluje jako sérii rovných či zakřivených tahů [12, sekce „SVG > Tutorials > Paths“].

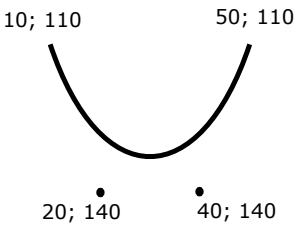
Základním atributem tohoto prvku je parametr „d“. Nastavováním jeho hodnoty lze specifikovat celkový tvar křivky jakožto sérii parametrizovaných příkazů. Příkaz je vždy reprezentován jedním písmenem určujícím typ příkazu a následně několika čísly určujícími parametry příkazu [6].



Obrázek 3.2: Typy křivek v editoru Blenderu. Nalevo lze vidět nejjednodušší typ „Poly“, ve kterém neprobíhá interpolace mezi kontrolními body a využívá se spíše jen při převodu meshů na křivky. Uprostřed lze vidět typ „Bézier“, který obsahuje 2 typy bodů, kontrolní body a jejich páry tzv. handles (úchyty, dále pouze handles). Segmenty křivky prochází kontrolními body a páry handles definují zakřivení segmentu. Napravo je poté typ „NURBS“ (z anglického Non-Uniform Rational B-Splines), který je oproti předchozímu typu složitější, neobsahuje handles a jeho kontrolní body mají navíc nastavitelné váhy.

Existuje 6 kategorií s celkem 10 typy příkazů, pro stručnost jsou však uvedeny pouze ty relevantní pro tuto práci. U formátu druhého níže uvedeného příkazu si lze všimnout jisté podobnosti mezi jeho třemi parametry (koncový bod a jeho dva kontrolní body) a mezi skupinami tří bodů, ze kterých se skládají spliny typu „Bézier“ v Blenderu (kontrolní bod a jeho dva handles). Ukázku využití zmíněných příkazů lze také vidět na obrázku 3.3.

- Typ „M“ je základní příkaz s dvěma parametry pro pozici, který přesunuje „kurzor“ na danou pozici. Nekreslí čáru, využívá se především ke specifikaci, na jakém místě začít kreslit.
- Typ „C“ je jedním ze 3 příkazů pro tvorbu zakřivených tahů. Vyžaduje celkem 6 parametrů popisujících 3 pozice: 2 pozice pro kontrolní body na začátku a na konci tahu a jednu pozici pro určení koncového bodu tahu. Příkaz vykreslí křivku začínající na aktuální pozici kurzoru a končící v koncovém bodě tahu, jejíž zakřivení je ovlivněno dvěma kontrolními body. Zároveň přesune kurzor na konec tahu.



Obrázek 3.3: Vykreslení křivky `<path d="M 10 110 C 20 140, 40 140, 50 110"/>` s vyznačením pozic kontrolních bodů

Kromě specifikace tvaru lze dalšími atributy specifikovat také vzhled výsledné křivky jako například šířka a barva tahu nebo styl výplně.

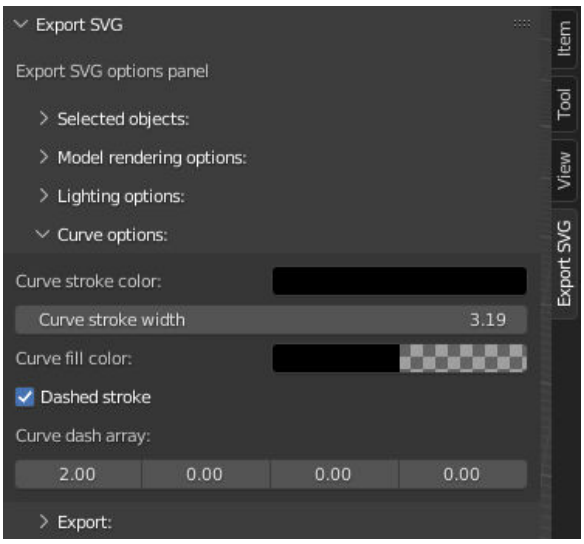
Návrh rozšíření

Práce se samotnými křivkami bude z pohledu uživatele probíhat stejně jako dosavadní práce s modely. Jednoduše budou vybrány objekty ve scéně a po kliknutí na tlačítko pro zahájení převodu budou převedeny na vektorovou grafiku, pouze s tím rozdílem, že tentokrát budou brány v potaz i objekty typu „CURVE“. Podporovány však nebudou všechny typy křivek (resp. jejich splinů), ale pouze druhý typ „Bézier“, který se jevil jako nejkompatibilnější pro převod na prvky <path> využívající pouze zmíněné příkazy M a C. Kromě samotného tvaru křivky bude také zohledněno u každého splinu nastavení „Cyclic“, kterým lze v Blenderu specifikovat, zdali má daná křivka být uzavřena tahem z koncového bodu zpět k počátečnímu.

Oproti převodu modelů na polygony by byl trochu složitější způsob ořezávání částí křivky ležících mimo pohled kamery. Kontrolní body křivky ležící v poloprostoru za rovinou kamery budou jednoduše při převodu ignorovány. Části ležící v poloprostoru před kamerou ale mimo pohled kamery, opět pro jednoduchost, nebudou nijak zvlášť ořezávány. Nástroje pro vykreslování formátu SVG automaticky ořezávají části ležící mimo specifikovaný rozsah okna, není tedy potřeba vynakládat úsilí navíc k řešení problému, který nijak neovlivní vizualizaci. Navíc může být pro uživatele žádoucí, aby soubor obsahoval celou křivku, pokud by jej dále nějak upravoval či měnil rozsah okna.

Z hlediska vizualizace je podstatnější přizpůsobení vzhledu křivek. V rámci rozšíření bude vytvořena další sekce uživatelského rozhraní panelu pluginu určená nastavení vzhledu křivek, ve které bude možno nastavit hodnoty několika podporovaných atributů, které se poté projeví na výsledném vektorovém obrázku. Podporovanými atributy formátu SVG budou barva a průhlednost tahu a výplně a přerušovanost a šířka tahu.

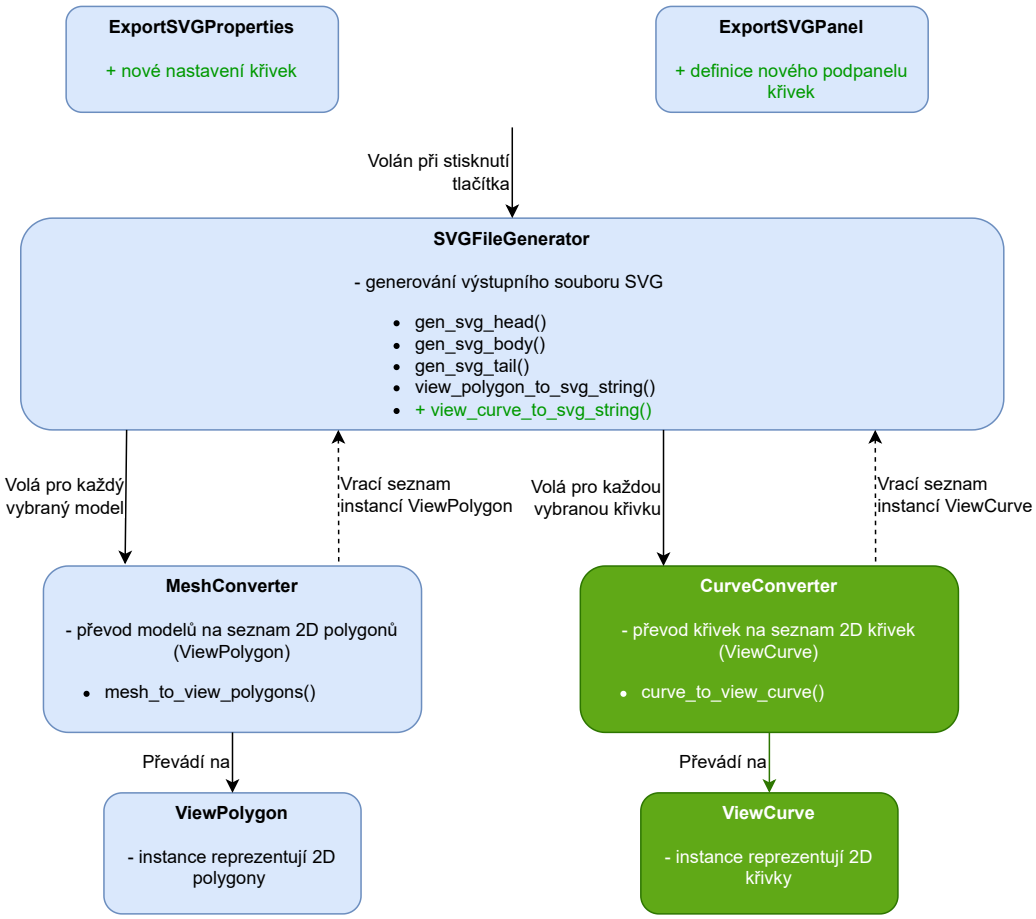
Pro lepší představu lze výsledné uživatelské rozhraní vidět na následujícím snímku obrazovky 3.4.



Obrázek 3.4: Nová sekce „Curve options“ s nastavením křivek po prvním rozšíření

Implementace rozšíření

V původním pluginu se skládal převod modelů na polygony ze dvou hlavních tříd: Převaděč modelů (třída `MeshConverter`) a Reprezentace výsledných polygonů SVG (třída `ViewPolygon`). Pro implementaci převodu křivek byl zvolen analogický postup, jehož princip lze popsat následujícím diagramem úprav 3.5, který popisuje změny oproti původní architektuře popsané v předchozí kapitole na obrázku 2.5.

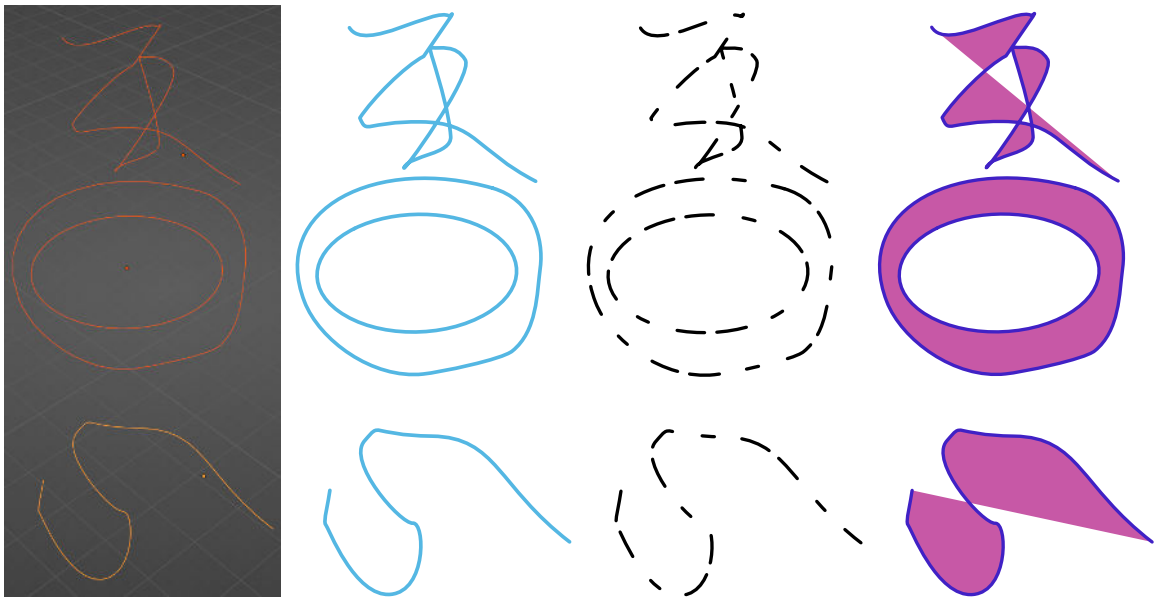


Obrázek 3.5: Diagram hlavních změn po implementaci převodu křivek. Zeleně jsou vyznačeny nově přidané části. Analogicky k převodu modelů byla vytvořena třída CurveConverter pro převod 3D křivek scény Blenderu na 2D křivky reprezentované instancemi nové třídy ViewCurve. Tyto třídy využívá už existující třída SVGFileGenerator pro převod všech křivek ve scéně při generování výstupního souboru SVG.

Samotný převod křivky se skládá z převodu pozic všech jejích kontrolních bodů a handles ve 3D prostoru z lokálních souřadnic do souřadnic scény a následně do 2D souřadnic pohledu kamery. Tyto pozice už jsou poté zapisovány do příkazů prvků <path> výsledného souboru v takovém pořadí, aby bylo dosaženo stejného tvaru. Bližší popis převodu a implementačních detailů lze nalézt ve zdrojovém kódu a jeho komentářích.

Výsledky

V rámci rozšíření byl úspěšně implementován převod křivek ve scéně Blenderu na vektorovou grafiku, který zároveň umožňuje do jisté míry přizpůsobovat jejich výsledný vzhled. Výsledky dosažitelné při různých kombinacích nastavení jsou představeny na následujícím obrázku 3.6.



Obrázek 3.6: Příklad výsledků rozšíření. Vlevo lze vidět snímek obrazovky převáděné scény v Blenderu se 3 křivkami různých tvarů, na pravých 3 obrázcích lze vidět vykreslené výstupní soubory vektorové grafiky s různými nastaveními převodu.

3.3 Převod textu

V této podkapitole je přiblíženo rozšíření umožňující převod objektů v Blenderu typu „FONT“ na prvky formátu SVG, konkrétně na prvky <polygon>, <path> nebo <text>. Jeden ze tří způsobů převodu je závislý na předchozím rozšíření, které implementovalo převod křivek.

Původní stav a cíl

Původní plugin nedokáže převádět textové objekty a v Blenderu objekty typu „FONT“ při převodu ignoruje. Podobně jako u křivek sice neobsahuje žádné metody pro práci s tímto typem, podporuje však už některé možné způsoby reprezentace 2D tvarů použitelných pro reprezentaci textu, jmenovitě polygony a křivky.

Cílem rozšíření je implementovat převod objektů tohoto typu na různé typy prvků formátu SVG vhodné pro reprezentaci textu a zároveň podobně jako u křivek umožnit do jisté míry upravovat vzhled výstupu pomocí atributů SVG. Díky tomu by uživatel mohl snadno vytvářet 2D vektorové obrázky zobrazující libovolně prostorově otočený text, což je v běžných nástrojích pro tvorbu 2D vektorových obrázků velmi obtížné.

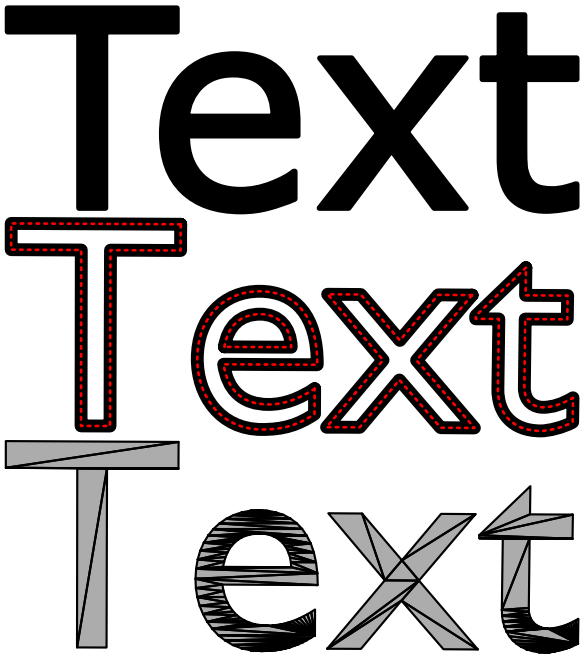
Vstup, výstup a teorie

Jedním z dalších často používaných typů objektů v Blenderu je typ „FONT“. Tento typ slouží k reprezentaci textových objektů, které umožňují vytvářet a vykreslovat 2D i 3D text. Textovému objektu lze v Blenderu přiřadit jakožto obsah řetězec textu. Blender potom namapuje jednotlivé znaky na odpovídající geometrii ve scéně. Kromě zabudovaného základního fontu dokáže Blender také importovat externí fonty [5, sekce „Modeling > Text“].

Dále Blender umožňuje širokou škálou nastavení upravovat zobrazení textu ve scéně. Jedná se nejen o klasická nastavení v podobě tučného textu, řádkování či mezer mezi znaky, ale také o specifické vizuální efekty, jako například deformování toku textu pomocí křivky [14]. Ve výchozím nastavení jsou jednotlivé znaky ve scéně pouze 2D plochy podobné vyplněné 2D křivce. Lze na ně také aplikovat modifikátory jako například Extrude (protažení do třetího rozměru), čímž z textu může vznikat i 3D model.

Důležitou vlastností Blenderu relevantní pro další postup je také konverze objektů, která umožňuje převod různých typů objektů na jiné, kompatibilní typy. V případě textových objektů je podporována konverze na objekty typu „MESH“, „CURVE“ a „GPENCIL“, díky které lze převést text na sérii polygonů, křivek a objekty Grease Pencil (tyto objekty jsou relevantní až v pozdějším rozšíření popsaném v podkapitole 3.11).

V rámci formátu SVG lze uvažovat o více způsobech reprezentace 2D textu znázorněných na obrázku 3.7.



Obrázek 3.7: Znázornění způsobů reprezentace textu ve vektorové grafice. Nahoře je text reprezentován textovým prvkem, uprostřed křivkami (vyznačenými červeným čárkovaným tahem), dole polygony (s šedou výplní a černými okraji pro názornost).

- Prvek <text>. Nejintuitivnější prvek SVG pro reprezentaci textu. Výhodou je zachování a snadná úprava původního řetězce textu. Nevýhodou jsou omezené vlastnosti vizualizace znemožňující napodobení 3D textu v prostoru nebo omezený výběr fontů.

- Prvek <path>. Pokud je o textu uvažováno jako o vyplněných křivkách, lze použít také tento prvek pro jeho reprezentaci. Výhodou je možnost reprezentovat libovolný font libovolně otočeného textu v prostoru, a to za použití relativně malého množství tvarů oproti následujícímu prvku <polygon>. Nevýhodou je ztráta informace o původním obsahu textu, neboť v souboru SVG už není uložen čistý textový řetězec.
- Prvek <polygon>. Pokud je o textu uvažováno jako o vyplněné 2D ploše, lze jej zcela jistě reprezentovat také skupinou polygonů stejného tvaru jako původní text. Výhodou je opět možnost reprezentace libovolného tvaru a fontu. Další výhodou je možnost reprezentace 3D textu vzniklého například protažením 2D textu do třetího rozměru. Nevýhodou je velké množství polygonů potřebných pro zachování původních zakřivených tvarů a ztráta informace o původním obsahu textu.

Návrh rozšíření

Základní práce s textovými objekty bude z pohledu uživatele opět probíhat stejně jako u modelů a křivek. Ořezávání bude také řešeno podobně jako u křivek. Textové objekty s pozicí za kamerou budou při převodu kompletně ignorovány a ořezávání částí ležících mimo pohled kamery bude opět ponecháno na formátu SVG a jeho vlastnosti ořezávání tvarů ležících mimo rozsah okna.

Protože každý dříve zmíněný způsob reprezentace textu může být vhodný pro jiné potřeby, rozšíření bude umožňovat všechny 3 způsoby převodu. Uživatel bude moci zvolit, zdali chce objekt konvertovat na model a převést na prvky <polygon>, konvertovat na křivky a převést na prvky <path>, nebo generovat obyčejný prvek <text> na 2D pozici odpovídající původnímu umístění textového objektu v prostoru scény.

K těmto 3 možnostem budou implementovány pro praktické použití ještě 2 další. Pokud má text sloužit jakožto popisek, není vždy žádoucí, aby byl pootočen v prostoru. Přesného vzájemného zarovnání kamery a otočení textu by však bylo pracné při pohledu na 3D scénu dosáhnout. Proto budou nabídnuty alternativní možnosti převodů na model a křivky, které geometrii objektů při převodu pootočí tak, aby text byl otočen směrem na kameru a na výsledném obrázku se tak tvářil jako „rovný“ 2D text (názorný příklad lze později vidět v sekci výsledků na obrázku 3.10).

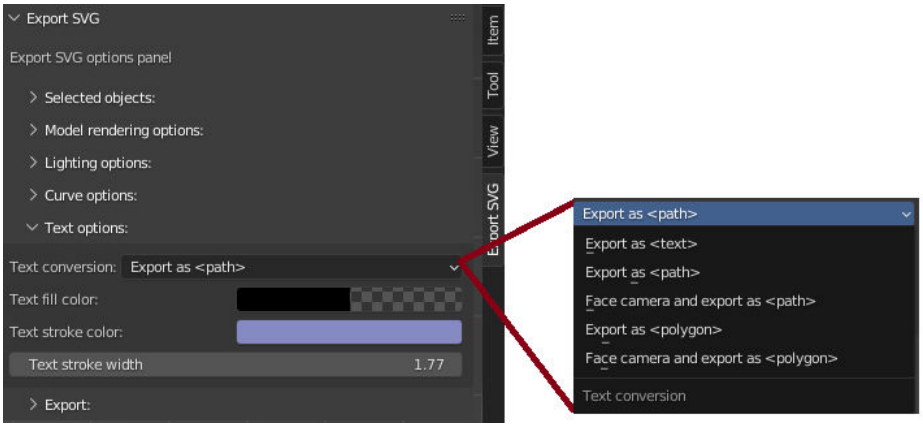
Z hlediska vizualizace budou podporovány stejné vizuální atributy jako dříve, tedy barva a průhlednost okrajů a výplně a šířka tahu. Podobně jako u křivek bude rozšířeno také uživatelské rozhraní, jehož příklad lze vidět na snímku obrazovky 3.8.

Implementace rozšíření

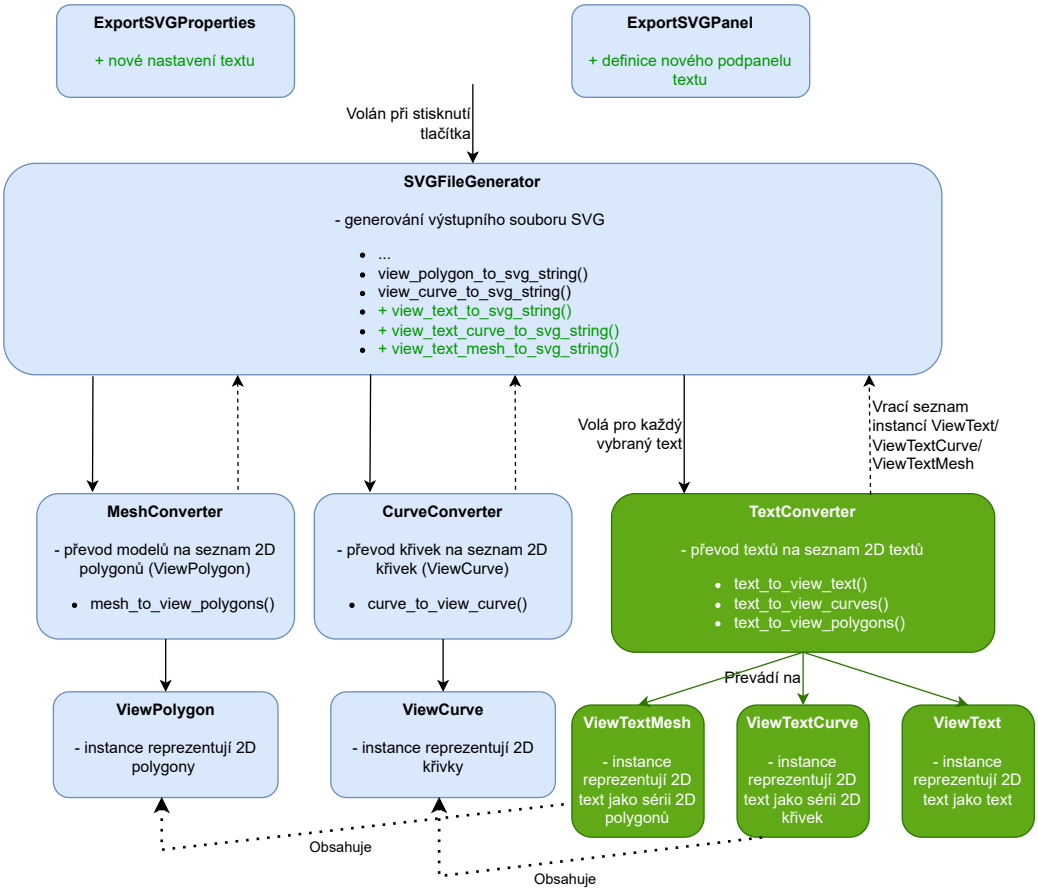
Protože se opět jedná o rozšíření převodu dalšího typu objektů, velká část implementace je v principu totožná s křivkami, lze ji opět popsat diagramem úprav 3.9.

Výsledky

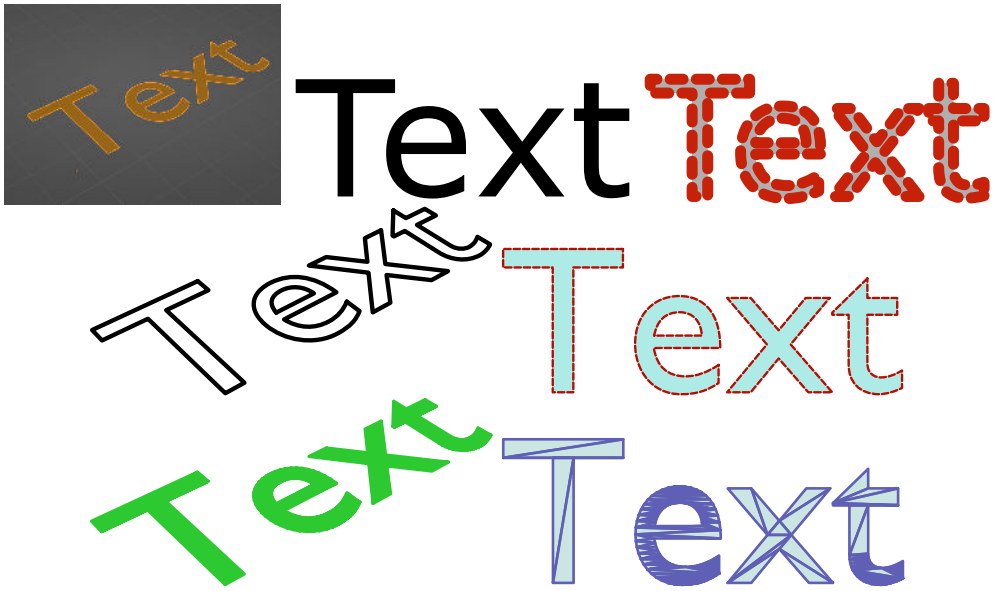
V rámci rozšíření byl úspěšně implementován převod textů ve scéně Blenderu na různé typy reprezentace ve vektorové grafice, který zároveň umožňuje do jisté míry přizpůsobovat jejich výsledný vzhled. Výsledky dosažitelné při různých kombinacích nastavení jsou představeny na obrázcích 3.10 a 3.11.



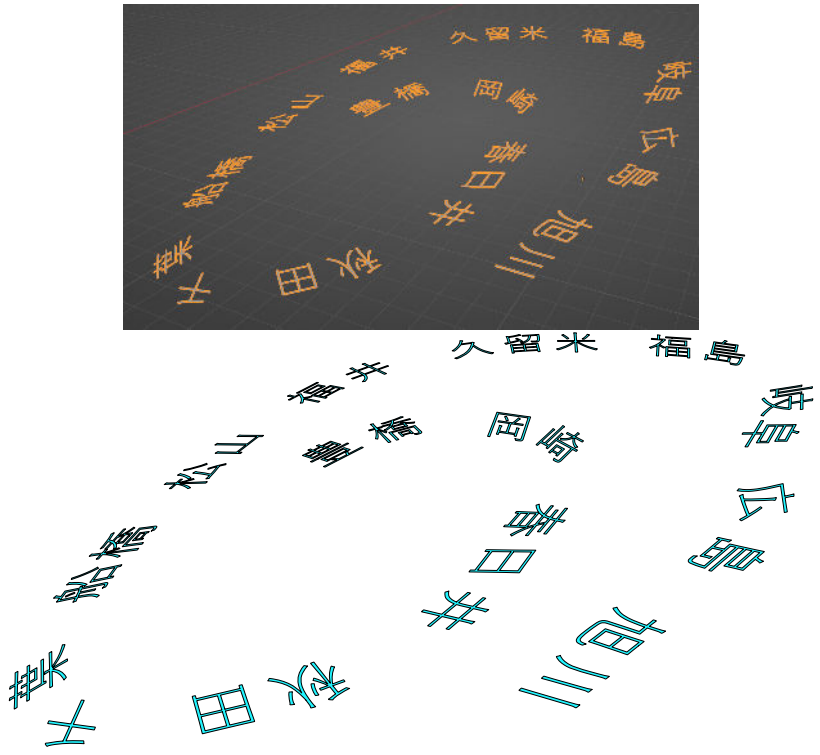
Obrázek 3.8: Nová sekce nastavení textu (Text options) po druhém rozšíření



Obrázek 3.9: Diagram hlavních změn po implementaci převodu textu. Zeleně jsou vyznačeny nově přidané části. Analogicky k modelům a křivkám byla implementována nová třída TextConverter pro převod textů ve scéně na 2D texty reprezentované instancemi tentokrát 3 různých tříd ViewTextMesh (pro polygonový text), ViewTextCurve (pro křivkový text) a ViewText (pro běžný text). Tyto třídy opět využívá třída SVGFileGenerator pro převod textů ve scéně při generování výstupního souboru SVG.



Obrázek 3.10: Příklad výsledků rozšíření. Vlevo nahoře lze vidět snímek obrazovky převáděné scény v Blenderu, na zbylých obrázcích lze vidět vykreslené soubory vektorové grafiky s různými nastaveními převodu. Na prvním řádku převod na text, na druhém řádku převod na křivky bez otočení a s otočením, na třetím řádku převod na polygony bez otočení a s otočením.



Obrázek 3.11: Příklad výsledků rozšíření při převodu japonského textu s importovaným fontem deformovaným v prostoru podle křivky. Nahoře lze vidět snímek obrazovky převáděné scény v Blenderu, na spodním obrázku vykreslený soubor vektorové grafiky při převodu textu na křivky.

3.4 Univerzální hloubkové řazení různých typů

V této podkapitole je přiblíženo rozšíření řešící problematiku vzájemného řazení různých typů prvků. Toto rozšíření navazuje na dříve implementovaná rozšíření pro převod křivek a textu.

Původní stav a cíl

Původní plugin a bakalářská práce se detailně zabývaly vzájemným hloubkovým řazením polygonů. Po implementaci dvou předchozích rozšíření ale nyní v převodu figurují nové typy prvků, křivky a texty. Při jejich implementaci nebyl řešen způsob vzájemného řazení, nejdříve jsou vykreslovány polygony, poté křivky a na závěr text, a to bez ohledu na jejich vzájemnou pozici či vzdálenost od kamery.

Cílem tohoto rozšíření je vylepšení architektury zvýšením míry abstrakce tříd reprezentujících jednotlivé typy prvků takovým způsobem, aby do budoucna bylo možno všechny prvky vzájemně řadit jedním způsobem bez ohledu na jejich typ. Prvky byl měly být řazeny jak v rámci jednoho typu (text blíže ke kameře by měl překrýt text daleko od kamery), tak také mezi typy (křivka nejblíže ke kameře by měla překrýt text dále od kamery a ten by měl zase překrýt křivku nejdále od kamery). Cílem však není detailní řešení složitých situací a vzájemně zaklíněných objektů jako v původní práci, spíše jen přibližné řazení na úrovni objektů. Původní práce však věnovala velké množství úsilí vylepšení řazení zaklíněných polygonů, toto rozšíření se tedy bude snažit tuto funkcionalitu u typu polygon zachovat a zaintegrovat do nového řešení.

Vstup, výstup a teorie

Rozšíření se týká především architektury, není potřeba tedy uvádět nové poznatky o Blenderu, jeho API či formátu SVG.

V principu řazení znamená vytvoření určitého pořadí v seznamu prvků na základě nějakého jejich společného atributu. Prvky jsou v této situaci polygony, křivky či texty (a v budoucnu i jiné typy), každé reprezentovány instancemi jiných tříd. Protože se jedná o hloubkové řazení, za společný atribut lze považovat vzdálenost od kamery, resp. od bodu, ze kterého je scéna pozorována.

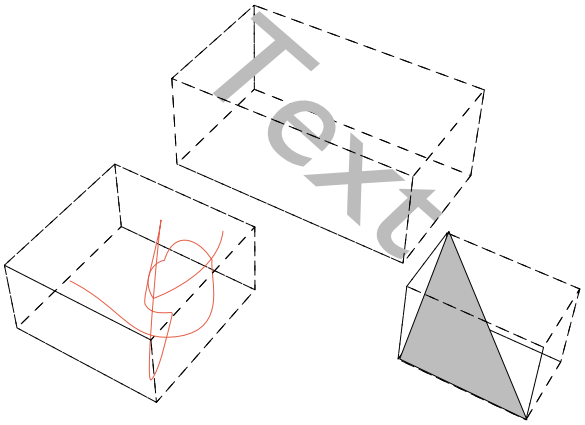
Problém nastává s určováním této vzdálenosti jakožto jediné hodnoty, neboť je otázkou, zdali uvažovat například nejbližší či nejvzdálenější vrchol polygonu nebo zdali u křivek uvažovat například průměr vzdáleností všech kontrolních bodů od kamery.

Jako možné řešení se jeví výpočet bounding boxů (ohraničujících kvádrů) pro každý objekt, což jsou nejmenší možné útvary s co nejmenšími objemy v prostoru ve tvaru kvádru, ve kterých leží, resp. do kterých se vlezou celé jejich objekty. Jejich výpočet už může být specifikován zvlášť pro každý typ objektu. Příklad ohraničení bounding boxy pro různé typy objektů lze vidět na obrázku 3.12.

Poté je možno hloubkové řazení zjednodušit na seřazení objektů například podle minimální či maximální hloubky bounding boxu nebo podle hloubky jeho středu.

Návrh rozšíření

U samotného řazení bude uživatel mít možnost nastavit, zdali chce prvky obecně řadit podle minimální, střední či maximální hloubky bounding boxů.



Obrázek 3.12: Vyznačení bounding boxů pro různé typy objektů

Řazení bude probíhat na dvou úrovních: nejdříve v rámci jednoto typu a poté mezi všemi typy současně. Při převodu všech objektů jednoho typu budou nově vzniklé prvky tohoto typu seřazeny jednoduše podle hloubky (až na polygony, které jsou řazeny speciálními algoritmy implementovanými v původní práci a neřídí se jednoduše hloubkou). Na vyšších úrovních převodu poté bude druhé řazení probíhat tak, že bude opakovaně hledán mezi všemi seřazenými typovými skupinami objekt s největší hloubkou, který bude vykreslen (resp. zapsán do souboru) a odstraněn z jeho příslušné skupiny.

Ačkoliv se tento návrh může zdát zbytečně komplikovaný oproti jednomu prostému seřazení všech typů dohromady podle hloubky, jeho hlavní vlastností je zachování pořadí z nižší úrovně v rámci jednoho typu objektů, díky kterému je možno zachovat speciální algoritmy pro řazení polygonů z původní bakalářské práce.

Možnosti řazení budou pro uživatele ještě vylepšeny dodatečnou funkcionalitou umožňující prioritní řazení typů. U každého typu objektů bude uživatel moci zvolit prioritní řazení a prioritní hodnotu. V takovém případě bude daný typ vykreslován až nakonec a objekty tohoto typu budou tedy překrývat všechny ostatní typy objektů bez ohledu na jejich skutečnou hloubku. Tato funkcionalita může být vhodná například pro tvorbu popisků, kdy uživatel chce zajistit, aby popisky tvořené křivkami byly vždy vpředu a nebyly nikdy překryty žádnými modely.

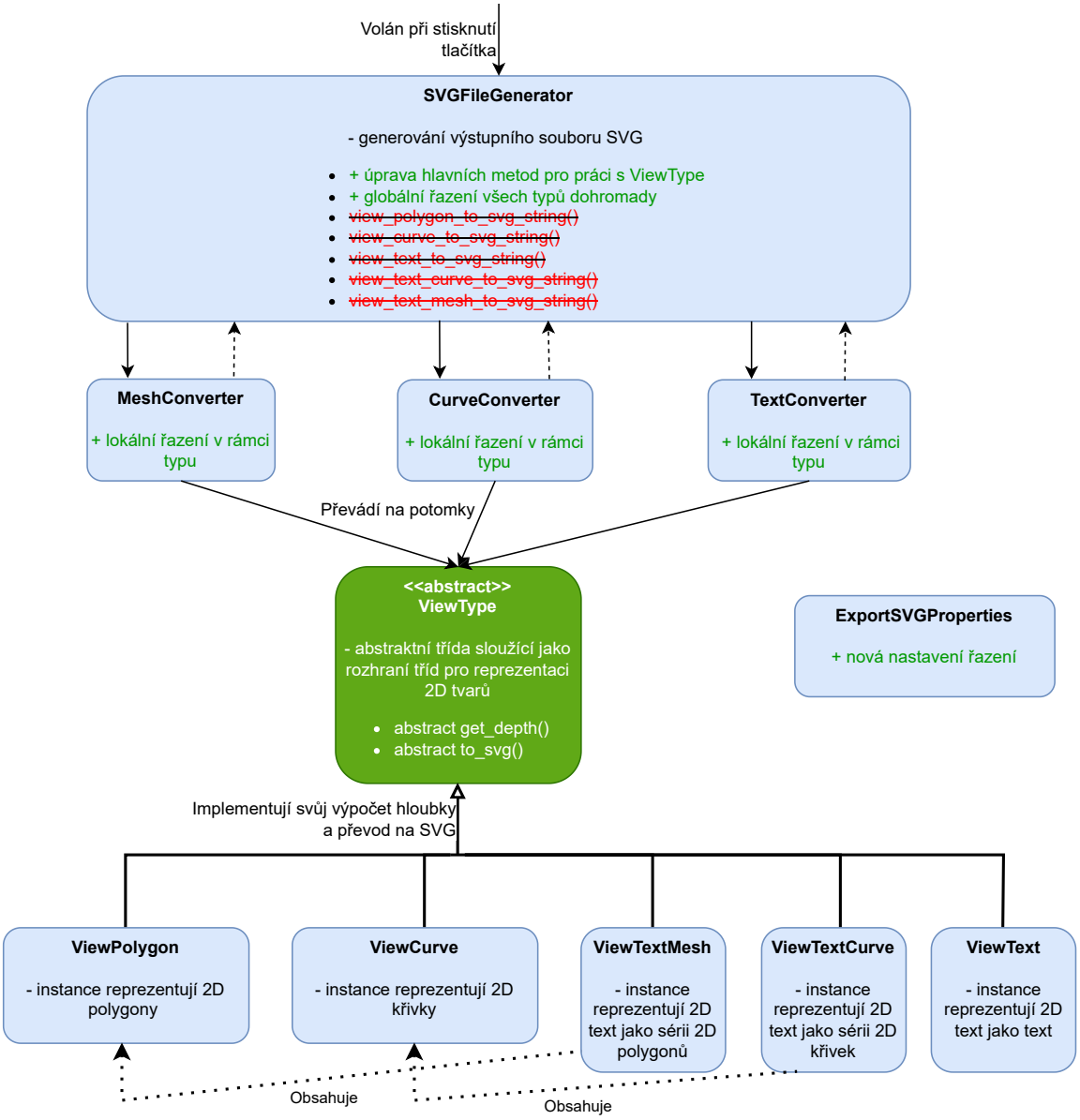
Implementace rozšíření

Úpravy v architektuře lze opět nejlépe vystihnout diagramem 3.13 zobrazujícím hlavní změny ve struktuře tříd a jejich rozhraní.

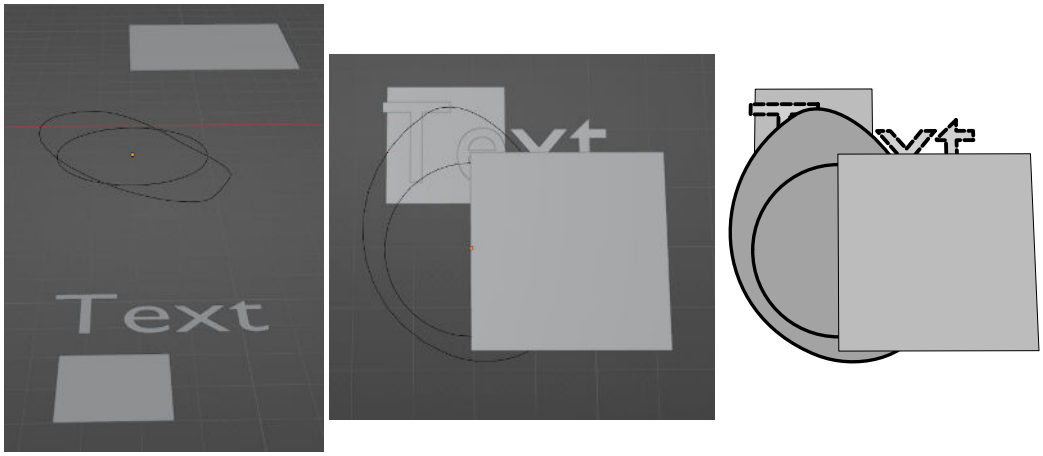
Výsledky

V rámci rozšíření bylo úspěšně implementováno globální řazení objektů schopné řadit současně mezi sebou různé typy objektů na základě jejich bounding boxu. Zároveň byla zachována možnost využití přesnějších řadicích algoritmů polygonů z původního pluginu. Příklad výsledku implementace lze vidět na obrázku 3.14.

Na závěr je však nutno dodat, že přestože rozšíření implementovalo možnost prioritně řadit některé typy objektů, tato funkcionalita byla později z pluginu odstraněna. Důvodem byla implementace následujícího rozšíření, které umožňuje snadno a intuitivně stejnou funkcionalitu replikovat a zároveň poskytuje mnohem více možností úpravy řazení objektů.



Obrázek 3.13: Diagram hlavních změn po implementaci řazení typů. Zeleně jsou vyznačeny nově přidané části, červeně odebrané části. Byla vytvořena nová abstraktní třída ViewType zastřešující všechny třídy pro reprezentaci 2D tvarů, která zajišťuje jednotnost jejich rozhraní nabízejících metody pro výpočet hloubky a pro převod 2D tvaru na řetězec formátu SVG. Díky jednotné práci s 2D tvary byla značně zjednodušena třída SVGFileGenerator, čímž bylo umožněno také implementovat jednotné hloubkové řazení fungující nezávislé na konkrétním typu 2D tvaru.



Obrázek 3.14: Řazení více typů mezi sebou. Na levém snímku obrazovky lze vidět hloubky objektů různých typů pohledem na scénu z boku, uprostřed poté pohledem shora. Na pravém obrázku lze vidět převedený vykreslený soubor vektorové grafiky s korektním překrytím objektů podle hloubky.

3.5 Převod kolekcí

V této podkapitole je přiblíženo rozšíření přidávající podporu převodu objektů seskupených do kolekcí Blenderu na skupiny prvků ve formátu SVG využitím seskupovacího prvku <g>.

Původní stav a cíl

Původní plugin nebere žádný ohled na hierarchii objektů a jejich rozdělení do kolekcí Blenderu.

Cílem tohoto rozšíření je umožnit uživateli lépe organizovat výsledný soubor SVG, a to rozdělením prvků SVG do skupin na základě rozdělení jejich původních objektů do kolekcí. Zároveň je cílem umožnit tyto skupiny v souboru řadit podle pořadí kolekcí v Blenderu, čímž by bylo možno do jisté míry replikovat funkcionalitu vrstev běžných editorů 2D grafiky.

Vstup, výstup a teorie

Objekty v Blenderu nejsou ve skutečnosti přímo součástí scény, ale nachází se v hlavní databázi reprezentované souborem. Na objekty se poté z různých scén pouze odkazuje. Pokud je objekt ve scéně odkazován, stává se součástí tzv. scene collection (kolekce scény), do které vždy patří všechny objekty scény [5, sekce „Assets, Files & Data System“].

Objekty však může uživatel dále lépe organizovat definicí vlastních kolekcí, do kterých lze objekty přiřazovat. Tyto kolekce nemusí být nutně disjunktní, neboť jeden objekt může být součástí více kolekcí [5, sekce „Scenes & Objects > Collections“].

Každá kolekce může být libovolně pojmenována a řazena v hierarchii objektů scény. Kolekce zároveň mohou obsahovat jiné kolekce a být dále zanořeny. Celé kolekce lze i skrývat, čehož bude využito v pozdějším rozšíření.

Obdobnou možnost seskupování nabízí i formát SVG v podobě prvku <g>, který seskupuje všechny ostatní prvky zapsané v jeho těle a kteří se tak stávají jeho potomky. Všechny transformace aplikované na tento prvek jsou poté aplikovány i na jeho potomcích, kteří dědí i jeho atributy. Zároveň lze se seskupenými prvky snadněji manipulovat v editorech vekto-

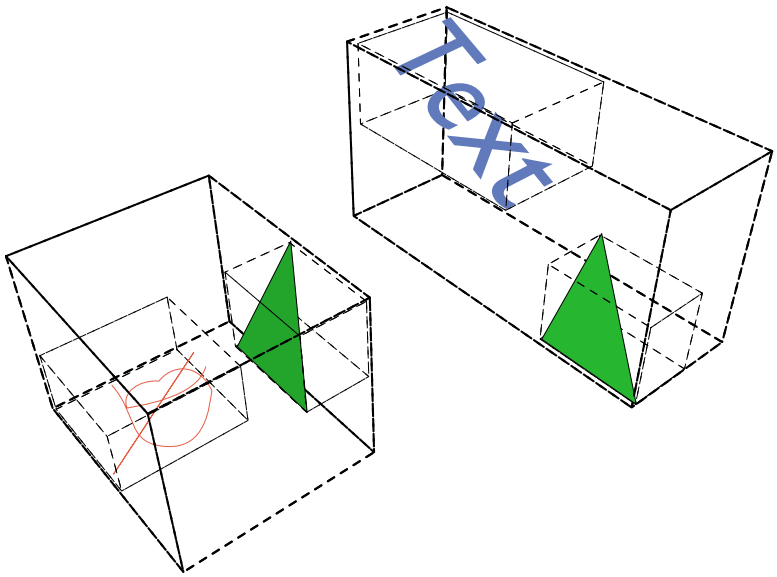
rové grafiky. Stejně jako kolekce lze skupiny zanořovat či pojmenovávat, resp. jim přidělovat identifikátory.

Na rozdíl od kolekcí je však každý prvek SVG potomkem vždy pouze jednoho přímého předka <g>.

Návrh rozšíření

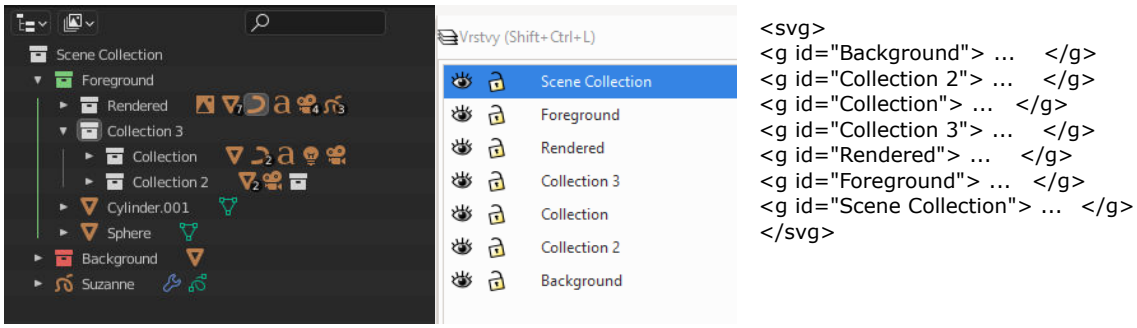
Uživateli bude poskytnuta v nastavení pluginu volba, zdali chce při převodu rozdělovat objekty podle kolekcí. Ve výchozím nastavení bude plugin pracovat stejně jako doposud, kdy jsou všechny vybrané objekty scény převedeny do jednoho společného prvku <g>. Pokud uživatel však nastaví rozdělení, bude převedena každá kolekce, která obsahuje alespoň 1 převáděný objekt, na samostatný prvek <g>. Tomuto prvku <g> bude přidělen identifikátor odpovídající jménu kolekce v Blenderu. To však přináší několik návrhových problémů, které je nutno vyřešit.

- Hlavní problém vychází z odlišnosti kolekcí Blenderu a prvku <g> týkající se zmíněné disjunkce. Nelze podporovat vlastnost přiřazení objektu do více kolekcí zároveň. Při převodu bude tedy objekt považován za součást pouze té kolekce, do které byl přiřazen jako první (resp. kolekce, která je na prvním indexu v jeho seznamu kolekcí).
- Druhým menším problémem je kompatibilita jmen. Blender podporuje ve svých identifikátorech velké množství speciálních znaků, naopak identifikátory SVG jsou v tomto ohledu velmi omezené. Nekompatibilní kolekce budou automaticky přejmenovány.
- Poslední otázkou je pořadí zápisu výsledných skupin <g> do souboru a tedy i jejich hloubkové řazení. Využity budou opět bounding boxy, tentokrát však počítané pro celé kolekce na základě bounding boxů jejich objektů, jak je znázorněno na obrázku 3.15. Uživateli bude nabídnuto řadit kolekce podle nejmenší, střední či největší hloubky jejich bounding boxu.



Obrázek 3.15: Hierarchické bounding boxy. Na obrázku lze vidět 2 kolekce, jejichž bounding boxy jsou počítány tak, aby zahrnovaly bounding boxy všech jejich objektů.

Jako speciální čtvrtý způsob řazení také bude existovat možnost řadit kolekce či skupiny <g> podle pořadí v seznamu objektů scény. Tento typ řazení by měl odpovídat 2D vrstvám v běžných editorech 2D grafiky. Kolekce na vrcholu seznamu tedy budou v popředí a budou vykreslovány až nakonec. Kolekce níže naopak budou v souboru zapsány dříve a budou vyššími kolekcemi překryty. Zanoření kolekcí nebude bráno v potaz. Pro pochopení lze srovnání s editorem 2D grafiky vidět na následujícím obrázku 3.16.



Obrázek 3.16: Ukázka využití pořadí kolekcí k reprezentaci pořadí vrstev. Vlevo lze vidět snímek obrazovky s kolekcemi v hierarchii objektů v Blenderu. Uprostřed lze vidět, jak by vypadalo ekvivalentní uspořádání 2D vrstev v nástroji Inkscape. Napravo lze vidět ekvivalentní zápis v souboru SVG, který odpovídá předchozím uspořádáním (musí být v opačném pořadí, neboť soubor SVG je vykreslován v pořadí shora dolů).

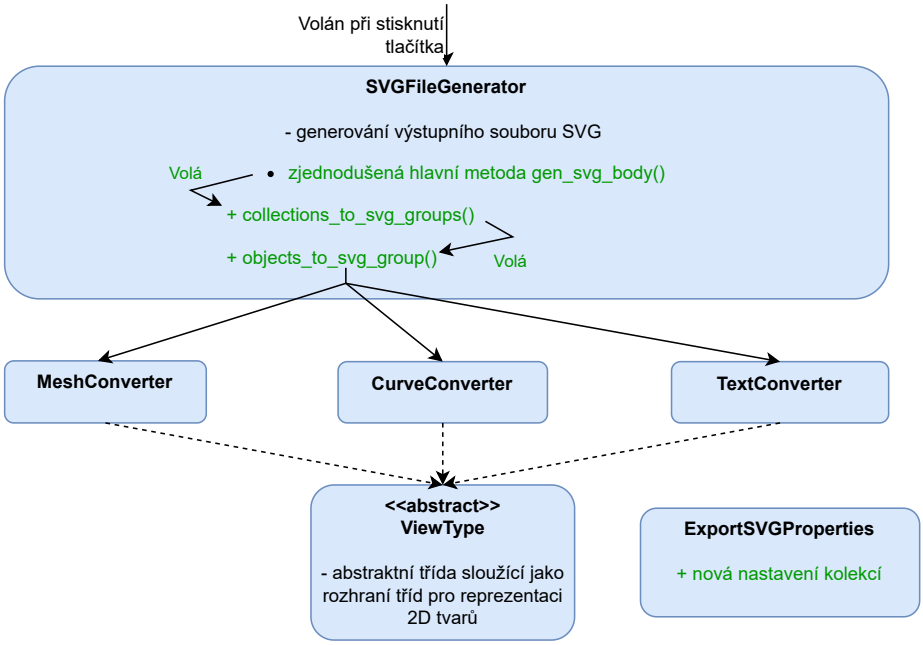
Tento způsob řazení svými možnostmi zcela překrývá možnosti prioritního řazení typů z předchozího rozšíření, prioritní řazení typů bude tedy po tomto rozšíření odstraněno.

Implementace rozšíření

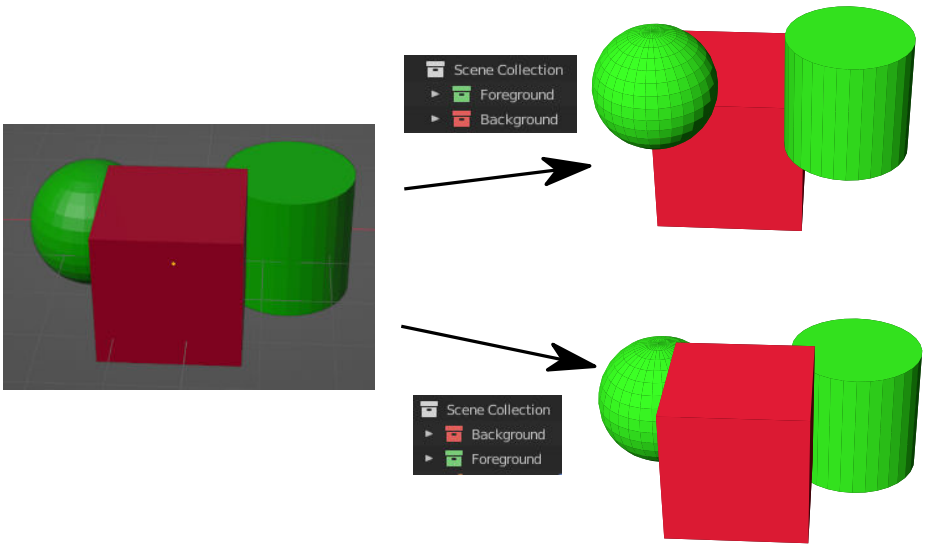
Změny v architektuře lze vidět na zjednodušeném diagramu změn 3.17.

Výsledky

V rámci rozšíření byla úspěšně implementována možnost převodu a rozdělení prvků SVG do skupin odpovídajících kolekcím scény Blenderu. Byla také implementována možnost řazení kolekcí na základě jejich pořadí v seznamu objektů, čímž byla do určité míry replikována funkcionalita vrstev 2D prvků. Příklad řazení lze vidět na obrázku 3.18.



Obrázek 3.17: Diagram hlavních změn po implementaci převodu kolekcí. Zeleně jsou vyznačeny nově přidané části. Změny proběhly pouze v implementaci převodu v rámci třídy SVGFileGenerator, jejíž hlavní metoda gen_svg_body(), která doposud sama generovala celé tělo souboru, byla dále rozdělena na samostatné metody pro převod kolekcí a pro převod objektů.



Obrázek 3.18: Využití řazení kolekcí jakožto vrstev. Nalevo lze vidět snímek obrazovky s původní scénou v Blenderu, objekty ve scéně jsou rozřazeny do kolekcí dle barevných označení. Napravo lze vidět 2 různé převody lišící se v pořadí kolekcí a jejich výsledné obrázky. Pořadí vykreslení objektů je možno ovlivňovat bez ohledu na jejich skutečnou hloubku ve scéně.

3.6 Podpora kamerových objektů

V této podkapitole je přiblíženo rozšíření přidávající podporu kamer. Rozšíření umožňuje využít tyto objekty pro tvorbu obrázků namísto aktuálního pohledu na scénu.

Původní stav a cíl

Jediný způsob pořizování vektorových obrázků v původním pluginu je z aktuálního pohledu uživatelovi kamery na scénu. Tento způsob práce s kamerou je sice intuitivní a snadný, ale může být problematický například v situaci, kdy uživatel chce vytvořit nový či upravit existující obrázek, který pořizoval ze specifického úhlu. Hledat opět ten stejný úhel pohledu je velmi pracné, obzvlášť když bylo vytvořeno více obrázků dané scény z různých úhlů a je nutno ji po úpravě znovu „fotit“, tedy znovu nastavovat pohled na stejná místa se stejným otočením.

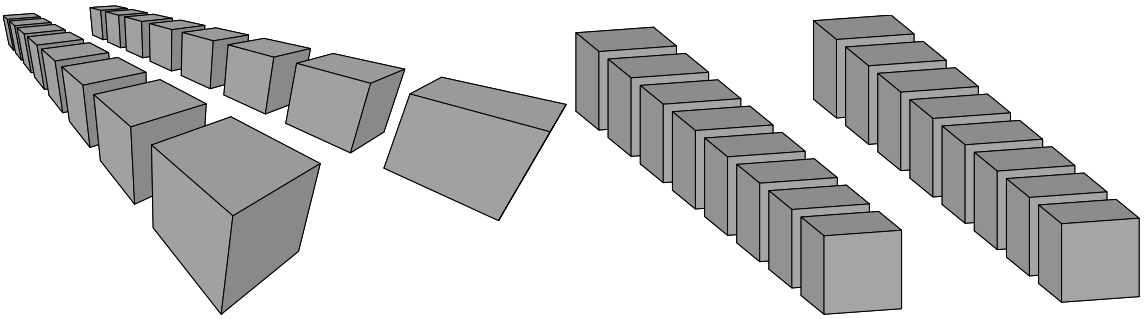
Cílem tohoto rozšíření je umožnit uživateli k řešení těchto problémů využít v Blenderu už existující kamerové objekty. Uživatel by měl moci do scény umisťovat kamery a po upravení nastavení využívat pro převod pohled kamery namísto vlastního pohledu. Zároveň bude uživateli umožněno těchto kamer využívat více současně pro tvorbu mnoha vektorových obrázků jedním kliknutím.

Vstup, výstup a teorie

Typ „CAMERA“ je v Blenderu typ objektů, které umožňují vykreslovat obrázky a definují, jaká část scény je na nich viditelná. Jakožto objekt však neobsahují žádnou geometrii a nejsou tedy viditelné [5, sekce „Rendering > Cameras“].

Tyto objekty mají několik vlastností, které mohou výsledný obraz ovlivňovat. Jednou z hlavních je výběr ze tří typů čočky, srovnání dvou z nich lze vidět na obrázku 3.19.

- Perspective – odpovídá perspektivní projekci, objekty v dálce se zmenšují a rovnoběžky konvergují [8, s. 301–303].
- Orthographic – odpovídá ortografické projekci, objekty mají stejnou velikost bez ohledu na vzdálenost a rovnoběžky zůstávají rovnoběžné [8, s. s. 315–316].
- Panoramic – speciální mód projekce, který lze použít pouze v jiném než výchozím render engine (vykreslovacím jádru) Blenderu, není dále brán v potaz.



Obrázek 3.19: Srovnání perspektivní (vlevo) a ortografické (vpravo) projekce řad kostek v Blenderu

Dalšími vlastnostmi jsou například posunutí čočky kamery na obou osách, nastavení vzdáleností pro ořezávání objektů a hloubku ostrosti nebo nastavení objektu jakožto ohniska pro zaostření. Vlastností existuje ještě mnohem více, avšak nejsou relevantní pro základní fungování tohoto rozšíření a dále už nebudou zmiňovány.

Z pohledu formátu SVG nelze nijak jednoduše integrovat více pohledů do jednoho souboru SVG kromě skládání obrázků vedle sebe. Každý pohled kamery lze ale reprezentovat samostatným obrázkem a tedy i samostatným souborem vektorové grafiky.

Návrh rozšíření

Práce s pluginem ve výchozím nastavení zůstane pro jednoduchost nepozměněna, uživatel bude převádět všechny vybrané a viditelné objekty z jeho aktuálního pohledu na jeden soubor vektorové grafiky.

Uživateli bude však také umožněna volba mezi převodem aktuálním pohledem, nebo kamerovými objekty. Po zvolení druhé možnosti bude práce s kamerovými objekty probíhat z pohledu uživatele podobně jako doposud s běžnými objekty. V potaz budou brány pouze kamery, které jsou součástí aktuální uživatelovi selekce objektů. Pro intuitivnější ovládání bude uživatelské rozhraní zobrazovat výpis aktuálně detekovaných kamer v selekci, které budou použity pro převod.

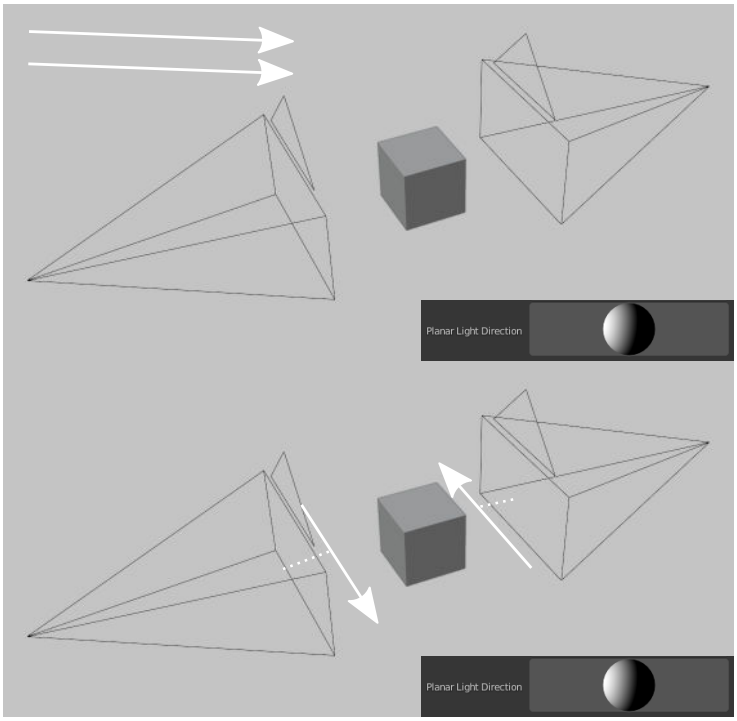
Pokud bude zvolen pouze 1 kamerový objekt, převod proběhne stejně jako doposud do zvoleného souboru. Pokud bude kamer zvoleno více, bude zvlášť pro pohled každé kamery vygenerován samostatný soubor formátu SVG pojmenovaný spojením zvoleného názvu výstupního souboru a názvu kamery.

Jedním problémem při použití kamerových objektů je určování směrového zdroje světla. Při použití směrového světla je jeho směr nastavován v uživatelském rozhraní pluginu a doposud byl relativní k pohledu na scénu. Otázkou je, jak by se nastavení mělo chovat u kamerových objektů. Řešením bude možnost uživatele nastavit relativnost při použití směrového světelného zdroje. Pokud nastavení zvoleno nebude, směr světla bude relativní pouze k aktuálnímu pohledu uživatele na scénu a bude tedy konstantní pro všechny kamery. Pokud bude nastaveno relativní světlo, bude nastavený směr ke každé kameře relativní a otočen podle směru pohledu dané kamery. Lépe situaci popisuje obrázek 3.20.

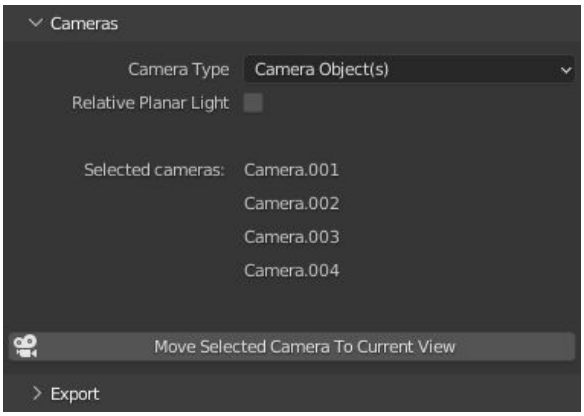
Další nevýhodou je horší ovladatelnost při umisťování a rotaci kamerového objektu oproti prostému pohledu na scénu. Plugin tedy bude nově obsahovat tlačítko, které přesune a otočí vybranou kameru takovým způsobem, aby její pohled byl totožný s aktuálním pohledem na scénu. Blender sám o sobě podobnou funkci nabízí, toto tlačítko by ji ale mělo udělat více přístupnější.

Nakonec bude také podporováno i několik výchozích vlastností kamer při převodu, konkrétně perspektivní a ortografická projekce, rozlišení (resp. zachování poměrů stran rozlišení), přiblížení, změny velikosti a posun čočky na obou osách.

Pro nastavení kamer bude vytvořena nová sekce uživatelského rozhraní, jejíž podobu lze vidět na snímku obrazovky 3.21.



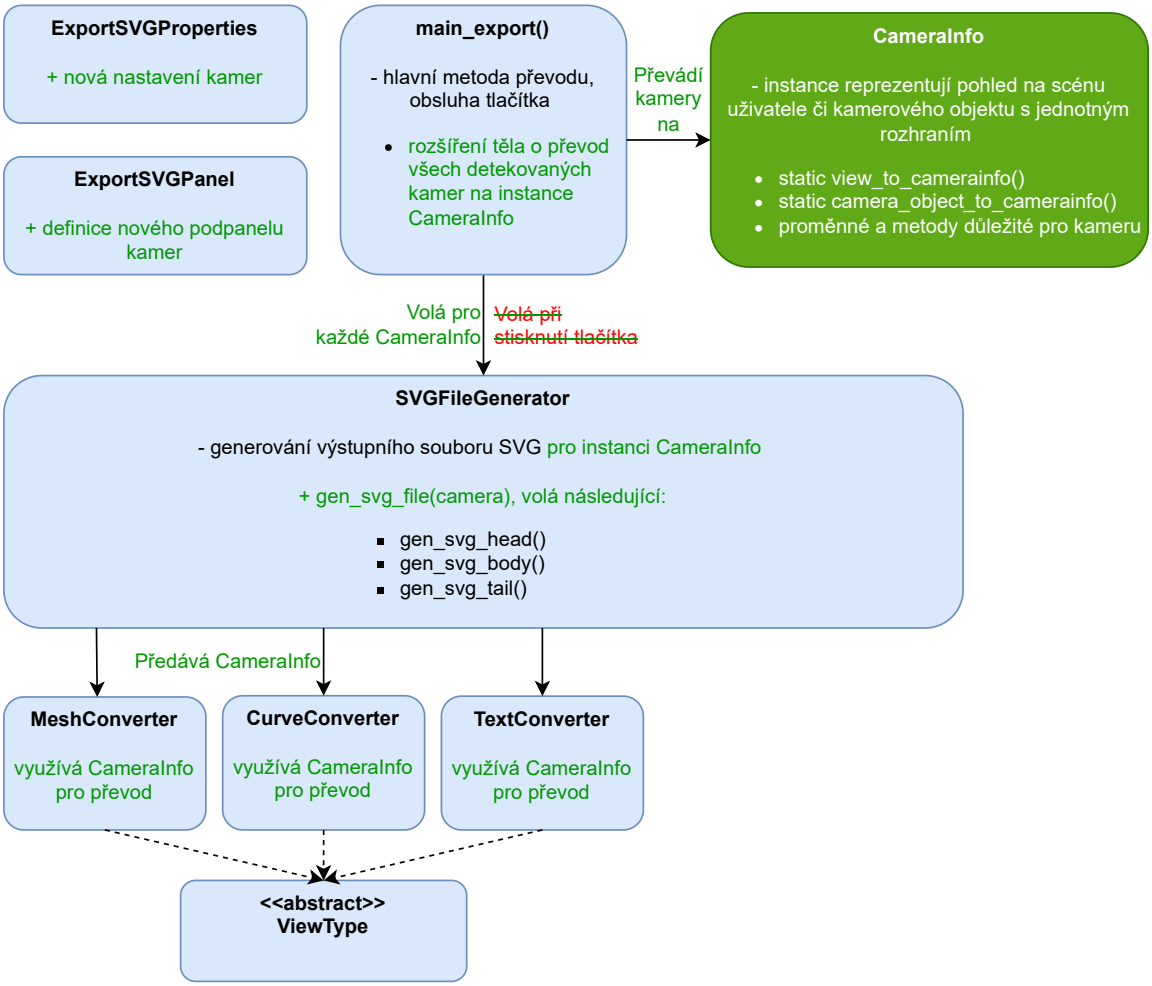
Obrázek 3.20: Srovnání absolutního a relativního směrového světla. Nahoře lze vidět absolutní směrové světlo, kde bílou šipkou vyznačený směr světla je stejný pro obě opačně otočené kamery a nastavení se vztahuje k uživatelově pohledu na scénu. Na spodním obrázku lze vidět relativní směrové světlo, kdy se nastavení směru vztahuje na každý pohled kamer individuálně.



Obrázek 3.21: Sekce uživatelského rozhraní s ovládáním kamer při výběru kamerových objektů

Implementace rozšíření

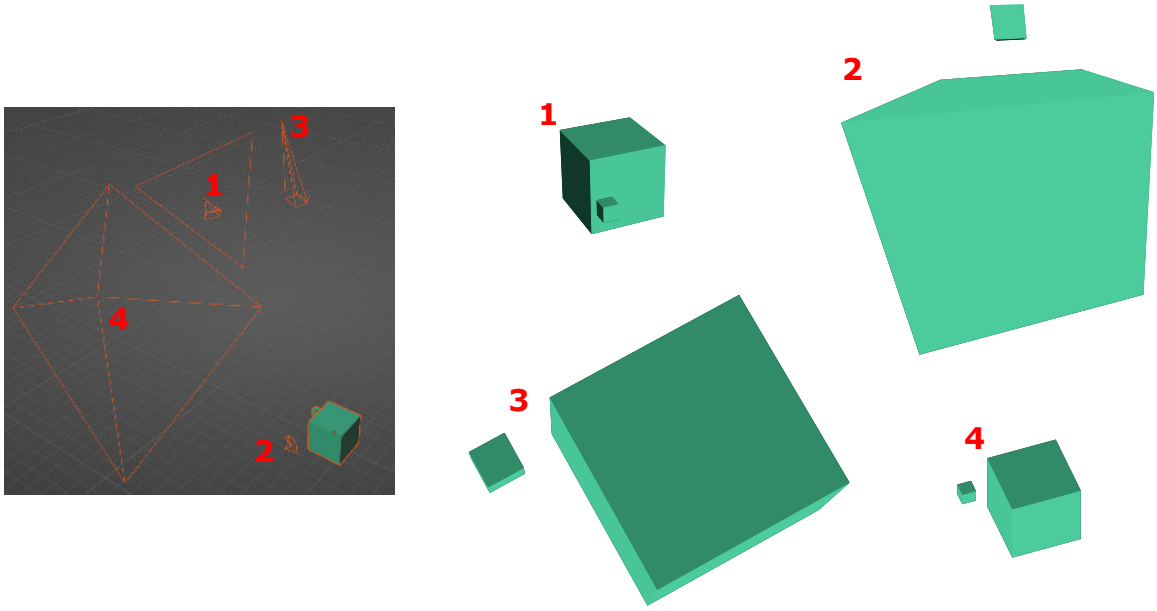
Implementace vyžadovala zásah do architektury napříč celým procesem převodu, neboť původní plugin byl od počátku plánován s využitím pouze uživatelova pohledu na scénu. Změny v architektuře lze vidět na diagramu změn 3.22.



Obrázek 3.22: Diagram hlavních změn po implementaci podpory kamerových objektů. Zeleně jsou vyznačeny nově přidané části, červeně části odebrané. Nově jsou při startu exportu převedeny všechny použité kamery na instance nové třídy CameraInfo, ve kterých jsou zaobaleny všechny důležité proměnné a metody pro převod. Generování souboru třídou SVGFileGenerator je nově voláno zvlášť pro každou instanci CameraInfo, zároveň byl přepracován celý převod ve všech třídách tak, aby využívaly pouze dostupné proměnné z instance CameraInfo a nesahaly tak do celého globálního kontextu Blenderu jako doposud. Další rolí třídy CameraInfo je také vytvoření společného rozhraní pro pohled uživatele na scénu a pohled kamery, které jsou samy o sobě strukturou a nabízenými metodami zcela odlišné.

Výsledky

V rámci rozšíření byla úspěšně implementována podpora kamerových objektů, které může uživatel nově využívat pro definování pohledů na převáděnou scénu nebo pro tvorbu více obrázků současně. Výsledky implementace lze vidět na následujícím obrázku 3.23.



Obrázek 3.23: Ukázka využití více kamer pro generování více obrázků současně. Nalevo lze vidět pohled na scénu Blenderu s 2 různě velkými kostkami a 4 kamerami. Napravo lze vidět korespondující výstupy pro každou z kamer. Jednotlivé kamery mají různé kombinace nastavení a parametrů, kamera číslo 4 je navíc ortografická. Všechny 4 obrázky lze vygenerovat jedním kliknutím na tlačítko Export.

3.7 Podpora výplní vzorkem

V této podkapitole je přiblíženo rozšíření umožňující uživateli nastavovat vzorky jakožto výplně 2D objektů s využitím prvku formátu SVG <pattern>. Samotná tvorba vzorků je však ponechána na uživateli, který k tomuto účelu může využít například kterýkoliv z dostupných online generátorů vzorků uvedených v části s výsledky.

Původní stav a cíl

V původním pluginu lze nastavovat jako výplň výsledných polygonů pouze danou barvu se specifikovanou průhledností, která případně může být po nastavení ovlivněna osvětlením, stále se však jedná o jednolitou výplň.

Cílem tohoto rozšíření je umožnit uživateli namísto jedné barvy použít jako výplň libovolný vzorek. Nastavení takové výplně by mělo být umožněno jak pro původní modely, tak i pro nově podporované křivky a texty z předchozích rozšíření.

Vstup, výstup a teorie

Z hlediska Blenderu nejsou potřeba žádné nové poznatky, neboť k reprezentaci vzorků nebudou využívány žádné vlastnosti scény či jejích objektů.

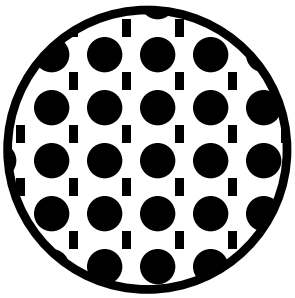
Rozšíření je komplexnější především z hlediska SVG, protože prvek pro reprezentaci vzorků <pattern> je jeden ze složitějších prvků tohoto formátu. Zároveň lze však jeho využitím dosáhnout mnoha vizuálních efektů.

K definici prvku <pattern> lze využít prvek <defs>. Tento prvek slouží k definici grafických objektů v souboru SVG, které lze později v těle souboru využívat. Prvky definované uvnitř <defs> přitom nejsou přímo vykreslovány, pouze pokud je na ně odkázáno.

Samotný prvek <pattern> umožňuje ve svém těle definovat libovolné doposud používané základní tvary. Dále lze k vzorku definovat také výšku a šířku, které specifikují, jak velká je jedna „dlaždice“ (anglicky tile) vzorku. Po dosažení hranic této dlaždice je dále vzorek do nekonečna opakován vykreslováním totožných dlaždic. Lze definovat také odsazení počátku generování vzorku od levého horního rohu [12, sekce „SVG > Tutorials > Patterns“].

Definice vzorků je komplikovaná hlavně z důvodu složitější práce s jednotkami jak samotného vzorku, tak všech jeho prvků. Později navržený způsob, jakým plugin bude se vzorky pracovat, ale umožňuje tyto komplikace obejít a nebudou zde pro stručnost podrobněji popisovány.

Prvky <pattern> lze také označovat identifikátorem. Vzorek poté může být použit u libovolného prvku jako výplň, a to odkazem na identifikátor odpovídajícího vzorku. Příklad definice a použití vzorku lze vidět na následujícím obrázku 3.24.



```
<defs>
  <pattern id="pat1" x="5" y="5" width="30" height="30">
    <circle cx="10" cy="10" r="10"/>
    <rect x="20" y="20" width="5" height="10"/>
  </pattern>
</defs>

<circle cx="100" cy="100" r="80"
stroke-width="5" stroke="#fffff" fill="url(#pat1)"/>
```

Obrázek 3.24: Příklad vzorků formátu SVG. Napravo lze vidět řetězec formátu SVG definující vzorek a kruh se vzorkovou výplní, nalevo vykreslení tohoto řetězce.

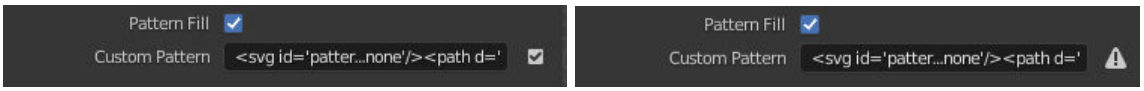
Návrh rozšíření

Protože se vzorky formátu SVG skládají z dalších definovaných prvků, jsou jejich vizuální možnosti neomezené, neboť se prakticky jedná o tvorbu nového vektorového obrázku uvnitř stávajícího. Jakákoliv implementace tvorby vzorků přímo uvnitř pluginu v Blenderu by byla omezená a nepraktická oproti kterémukoliv existujícímu generátoru vzorků. Plugin se tedy nebude pokoušet funkcionalitu generátorů kopírovat, ale bude pouze zajišťovat import prvku <pattern>, jeho překopírování do výsledného souboru a nastavení výplní pro určené prvky.

Nastavování vzorků by tedy pro uživatele mělo probíhat následovně. Uživatel si po nastavení vzorkové výplně u modelů, křivek či textů zvolí nástroj pro generování vzorků SVG. Většina z těch nejpopulárnějších nástrojů umožňuje tvořit a upravovat vzorek, jehož výsledný kód SVG lze následně zkopírovat. Tento kód poté uživatel vloží do příslušného pole pluginu. Plugin poté při převodu přečte kód a vyextrahuje z něj prvek <pattern>, který dále uloží do definiční sekce generovaného souboru a pro určené prvky nastaví jejich výplň jako odkaz na tento prvek.

Práce s pluginem bude však pro uživatele poněkud méně intuitivní kvůli omezenému výběru prvků uživatelského rozhraní podporovaných Blender API. Ideální by pro reprezentaci vzorku bylo například okno s rozsáhlým textovým polem, které by jasně zobrazovalo

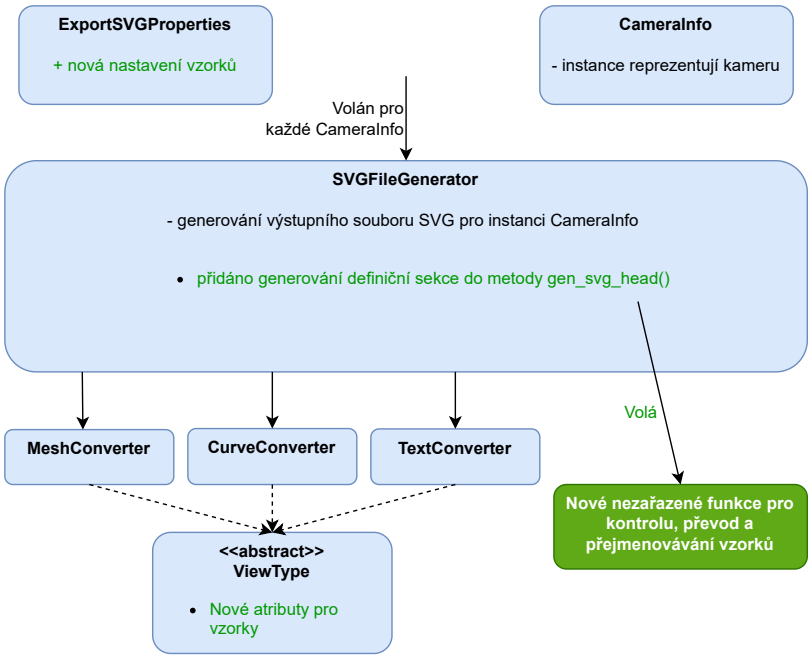
celý nakopírovaný kód. Blender API ovšem v současné verzi podporuje pouze jednořádkové prvky uživatelského rozhraní pro vlastnosti typu řetězec, uživatel tak tedy uvidí jen malou část nakopírovaného kódu. Pro zlepšení intuitivnosti rozhraní bude vedle řádku pro vložení kódu zobrazována ikona, která bude značit, zdali je kód ve správném formátu a prvek <pattern> byl úspěšně načten, jak lze vidět na snímcích obrazovky 3.25.



Obrázek 3.25: Ukázka nových uživatelských prvků pro vkládání vzorků. Prvek rozhraní Blenderu pro vkládání řetězců má pouze jednořádkové pole, většinu vkládaného řetězce tak nelze vidět. Na pravé části obou obrázků lze vidět pomocnou ikonu, která uživateli dává informaci o tom, zdali je v poli platný (vlevo) či neplatný (vpravo) vzorek.

Implementace rozšíření

Využití prvků <pattern> vyžadovalo první zásah do generování hlavičky souboru SVG. Hlavní změny architektury lze vidět na následujícím diagramu 3.26



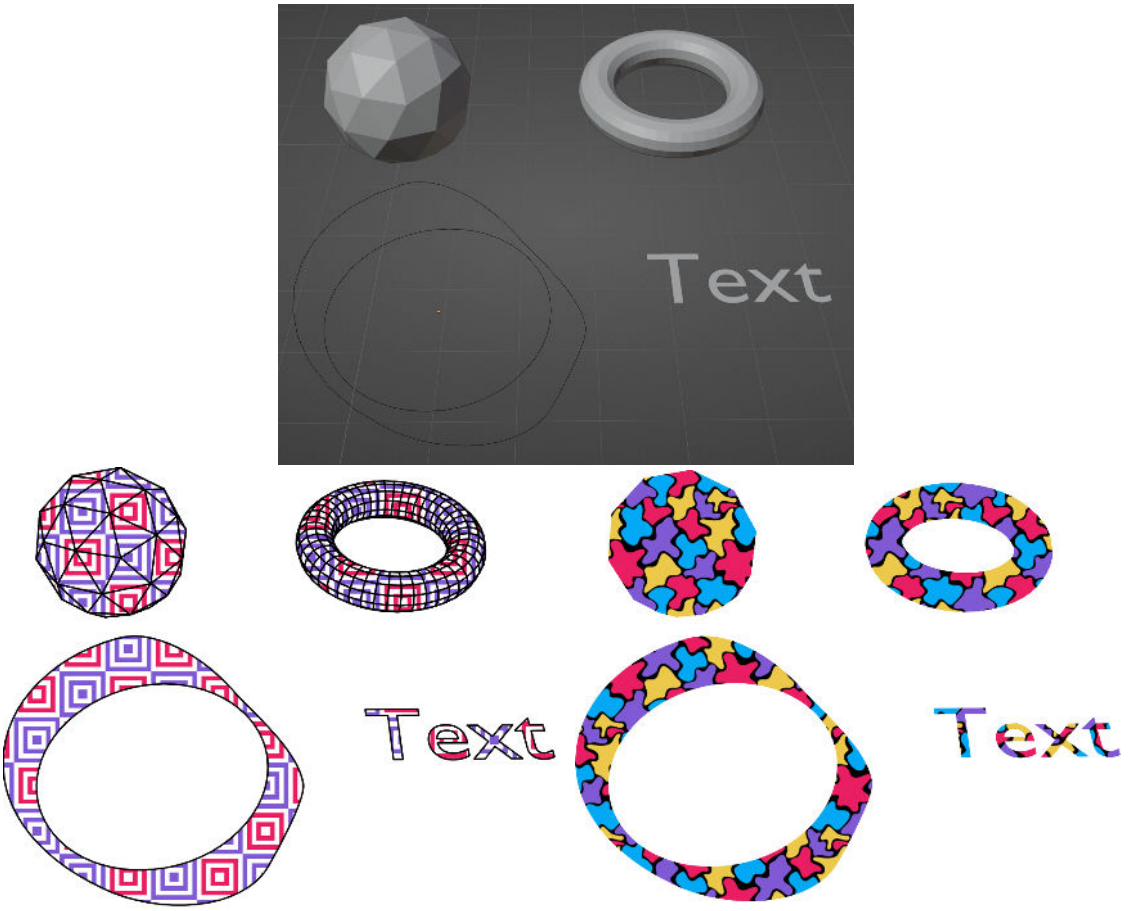
Obrázek 3.26: Diagram hlavních změn po implementaci podpory vzorků. Zeleně jsou vyznačeny nově přidané části. Nebyly vytvářeny nové třídy, pouze byly rozšířeny vlastnosti existujících tříd pro reprezentaci 2D tvarů. Dále byla rozšířena metoda pro generování hlavičky souboru, ve které nově musí probíhat generování definičních řetězců použitých vzorků. Pro tyto účely byly implementovány nové pomocné funkce, které nejsou součástí žádné třídy.

Ke kontrole formátu vkládaných řetězců a práci se stromovou strukturou formátu XML byla použita knihovna jazyka Python `ElementTree`[1].

[1] Dokumentace knihovny: https://docs.python.org/3/library/xml.etree.elementtree.html

Výsledky

V rámci rozšíření byla úspěšně implementována možnost používat vzorky jakožto výplně na výsledném obrázku pro modely, křivky i text. Uživatel si může s pomocí online generátorů vzorků[2] přizpůsobit libovolný vzorek, jehož kód může nakopírovat do rozhraní pluginu a ten ověří, zdali nalezl platný prvek <pattern> formátu SVG. Plugin poté při převodu z nakopírovaného textu vyextrahuje první nalezený vzorek a nastaví jej jako výplň pro dané objekty. Výsledky dosažitelné tímto postupem lze vidět na následujících obrázcích 3.27.



Obrázek 3.27: Generování obrázků se vzorky. Nahoře lze vidět snímek obrazovky původní scény v Blenderu, níže potom vykreslené výstupní soubory se vzorkovými výplněmi. Při zneviditelnění tahů na pravém obrázku lze docílit efektu „vykrajování“ ze vzorku pomocí objektů scény.

Rozšíření však nefunguje zcela vždy, neboť je limitováno podporou vzorkového prvku formátu SVG. Vzorky jsou obecně sice viditelné při zobrazení ve webovém prohlížeči, v běžných editorech 2D grafiky jako například Inkscape se ovšem při otevření souboru s obrázkem zobrazí pouze černá výplň namísto vzorku. Obrázek lze sice dále upravovat a samotné vzorky zůstanou i přesto v souboru zachovány, není však podporován například export do formátu PDF, který opět vzorek nahradí černou výplní.

[2]Například ke 13. 2. 2023 patří mezi první výsledky ve vyhledávačích: https://pattern.monster/, https://10015.io/tools/svg-pattern-generator, https://fffuel.co/ooorganize/, https://www.visiwig.com/patterns/, všechny tyto nástroje umožňují zkopírovat kód SVG.

3.8 Přepracování uživatelského rozhraní

Následující podkapitola se věnuje přepracování uživatelského rozhraní a přibližuje větší množství různě významných modifikací práce s pluginem. Podkapitola nenásleduje stejnou strukturu jako všechny předchozí rozšíření, ale obsahuje pouze seznam změn v návrhu rozhraní a ovladatelnosti pluginu, neboť samotná implementace a práce s API je z pohledu rozhraní triviální. Toto rozšíření zároveň navazuje na všechna předchozí rozšíření, jejichž nastavení musela být do pluginu a jeho uživatelského rozhraní zaintegrována.

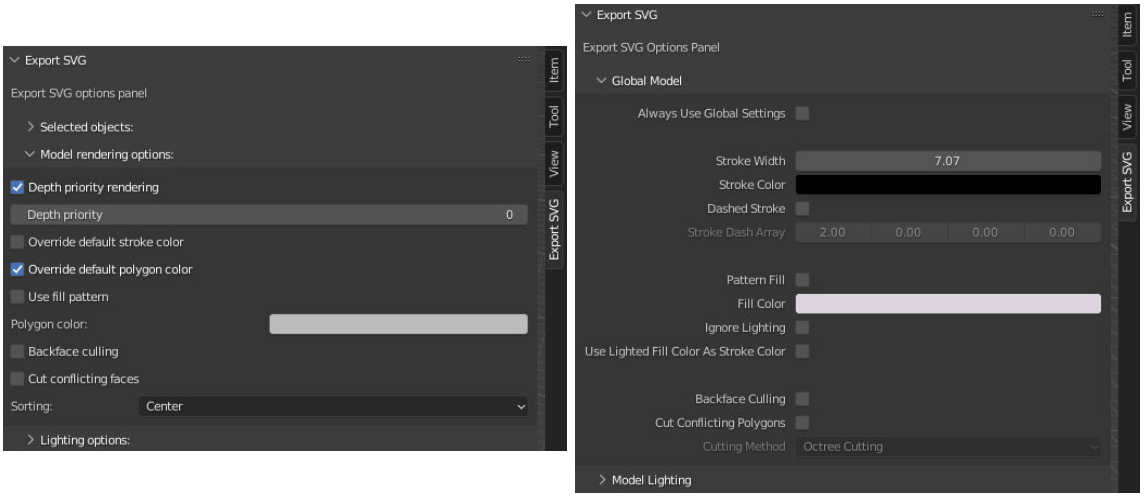
Původní stav a cíl

Ačkoliv dřívější uživatelské rozhraní (dále pouze UI) bylo pro původní plugin dostačující, po implementaci mnoha předchozích rozšíření významně narostl počet uživatelem nastavitelných parametrů převodu a jejich postupné přidávání do panelu pluginu značně znepřehlednilo celé rozhraní. Původní plugin také využívá nevhodné prvky pro reprezentaci některých parametrů, například checkbox (zaškrtávací pole) namísto dropdown menu (rozbalovacího seznamu). Dále není vždy jasné, jakým způsobem se některé kombinace nastavení navzájem ovlivňují a zdali nastavení jednoho parametru převodu na určitou hodnotu nepřekryje jiný související parametr.

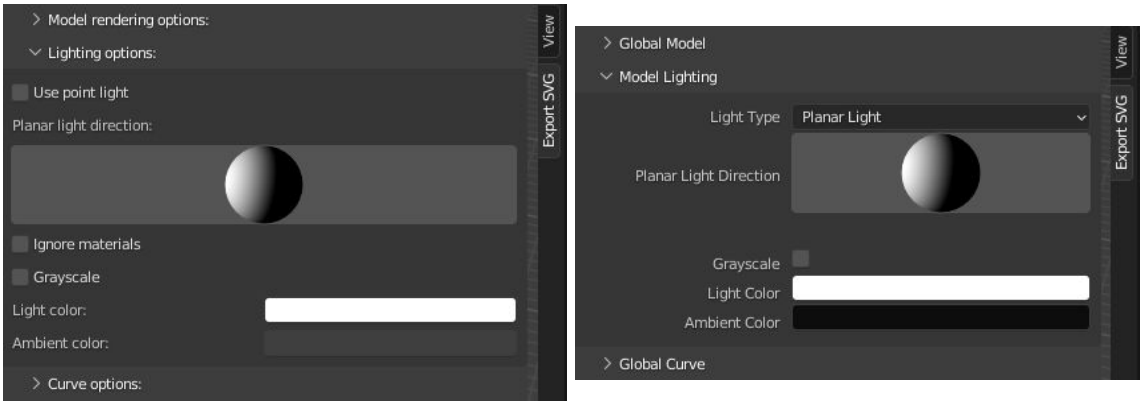
Celkové přepracování UI má za cíl řešení těchto problémů, tedy restrukturalizaci a modifikaci jednotlivých sekcí a jejich prvků za účelem zpřehlednění panelu pluginu, vylepšení logiky UI pro intuitivnější práci s nastavením, rozšíření možností interakce s pluginem (například volba způsobu selekce převáděných objektů) a zlepšení zpětné vazby uživateli.

Seznam změn v rozhraní

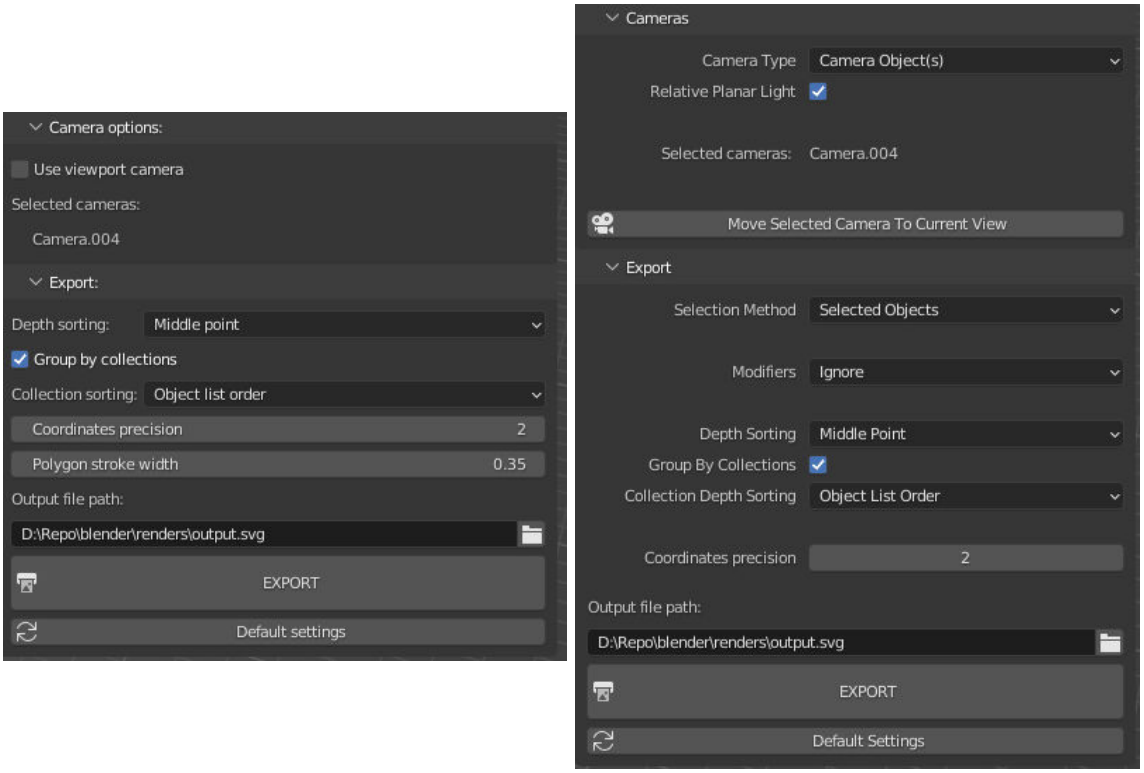
Změny UI a jeho fungování lze předvést na snímcích obrazovky 3.28, 3.29 a 3.30 zobrazujících rozhraní před a po úpravách, včetně popisků významnějších změn.



Obrázek 3.28: Srovnání sekce nastavení vlastností modelů hlavního panelu (obdobně jako modely byly upraveny i sekce pro křivky a text). Vlevo lze vidět původní stav, vpravo upravenou verzi. Lze si všimnout nového rozvržení prvků do sloupců dle poměru 4:6, kde levou část zabírá popisek prvku a pravou samotný interaktivní prvek, po vzoru online návodu [1]. Tento způsob rozvržení byl použit i pro zbytek pluginu, zároveň jej používá i celé prostředí Blenderu, tudíž je plugin více konzistentní s okolním prostředím.



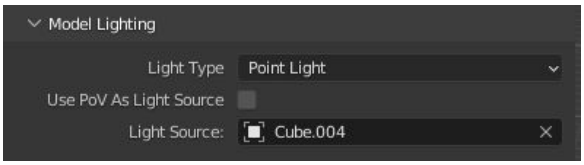
Obrázek 3.29: Srovnání sekce nastavení osvětlení hlavního panelu před (vlevo) a po úpravách (vpravo). Lze si všimnout například přepracování zaškrtávacího pole pro volbu typu světla na rozbalovací seznam, který je mnohem intuitivnější. Tato úprava byla rovněž provedena u všech nevhodných zaškrtávacích polí ve zbytku pluginu.



Obrázek 3.30: Sekce nastavení kamer a výstupu před (vlevo) a po úpravách (vpravo)

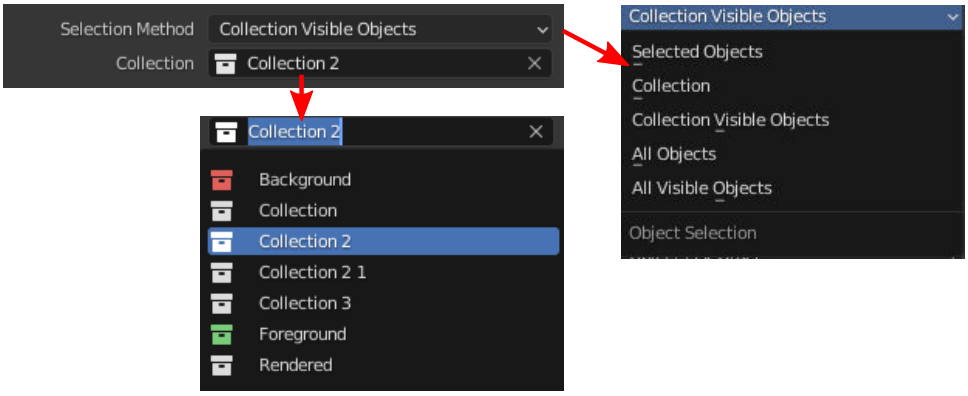
Kromě úprav viditelných na obrázcích lze zmínit několik dalších. Jednou je například přidání logiky při zobrazování prvků rozhraní. Uživatel už nově nemůže měnit hodnoty parametrů, které kvůli jiným nastavením nejsou brány při převodu v potaz (například nelze měnit parametry čárkování tahu, pokud je samotné čárkování vypnuto, nebo nelze měnit vliv osvětlení na tahy polygonů, pokud je osvětlování samotných polygonů vypnuto) Tohoto účelu je dosaženo dle situace buďto zešednutím prvku pomocí speciální funkce API, pokud momentálně nemá význam, nebo jeho úplným skrytím.

Také došlo k několika úpravám ovladatelnosti pluginu týkajících se výběru objektů. První z nich je způsob výběru bodového světla. Doposud plugin hledal v seznamu objektů objekt typu světlo, první na který narazil poté považoval za zdroj světla. Nově je uživateli umožněno přímo v hlavním panelu nastavit libovolný objekt scény jako zdroj světla, může jím tedy být i některý z převáděných modelů, jak lze vidět na obrázku 3.31.



Obrázek 3.31: Jako zdroj bodového světla je možno nově nastavit libovolný objekt, například model kostky.

Poměrně významnou změnou v ovládání pluginu je také nová možnost nastavení způsobu výběru objektů pro převod. Jedná se o změnu v základním fungování pluginu, který doposud převáděl všechny objekty, které v okamžik převodu byly součástí uživatelovi selekce. Nové možnosti lze vidět na obrázku 3.32.



Obrázek 3.32: Nastavení alternativních způsobů výběru objektů pro převod. Nově lze zvolit, zdali převádět všechny objekty v selekci, všechny objekty ve scéně, všechny viditelné objekty ve scéně (tedy všechny objekty, které nebyly schovány nastavením „Hide in Viewport“), všechny objekty v dané kolekci nebo všechny viditelné objekty v dané kolekci. V případě výběru využitím kolekce je zobrazen další rozbalovací seznam pro výběr převáděné kolekce. V rámci této kolekce potom funguje převod jako doposud, lze tedy tuto kolekci dále rozdělovat na podkolekce a převádět je na samostatné skupiny.

Na závěr lze zmínit už pouze drobnější úpravy týkající se například úprav stylizace názvů a popisků tak, aby byly konzistentnější jak mezi sebou, tak i s výchozím prostředím Blenderu.

3.9 Materiálový systém

V této podkapitole je přiblíženo jedno z nejrozsáhlejších rozšíření umožňující uživateli využívat materiálový systém Blenderu ke specifikování vizuálních vlastností téměř všech typů převáděných objektů s využitím jazyka CSS. Konkrétně jeho tříd a prvků <style>, které formát SVG podporuje, alespoň tedy v běžných prohlížečích či například nástroji Inkscape.

Původní stav a cíl

V původním pluginu a v rámci předchozích rozšíření lze nastavovat vizuální vlastnosti (různé atributy formátu SVG jako šířka tahu, čárkovaný tah, průhlednost či vzorek výplně, dále bude používán pro stručnost pouze výraz vizuální vlastnosti) pouze globálně pro všechny objekty daného typu současně, nelze tedy například nastavit dvěma křivkám dvě různé barvy. Jedinou specifickou výjimkou jsou barvy výplní modelů, kterým lze přiřazovat materiály a nastavovat individuálně alespoň barvu výplně, ale už nelze nastavovat žádnou jinou vlastnost.

Cílem rozšíření je zobecnění této specifické výjimky a umožnění nastavování všech vizuálních vlastností pro všechny typy objektů pomocí materiálů. Rozšíření by mělo uživateli umožnit nastavovat vizuální vlastnosti nejen globálně, ale také jednotlivě pro dané objekty nebo pro libovolné skupiny objektů současně.

Díky napojení na materiálový systém Blenderu popsaný v následujícím teoretickém souhrnu také bude mít uživatel možnost definovat vizuální vlastnosti předem jakožto materiál, který už poté pouze aplikuje na jednotlivé objekty k úpravě jejich vzhledu. Tyto materiály také díky Blenderu bude moci libovolně ukládat a přenášet mezi jednotlivými typy objektů, scénami, či dokonce projekty Blenderu a tím libovolně kopírovat už existující styl SVG.

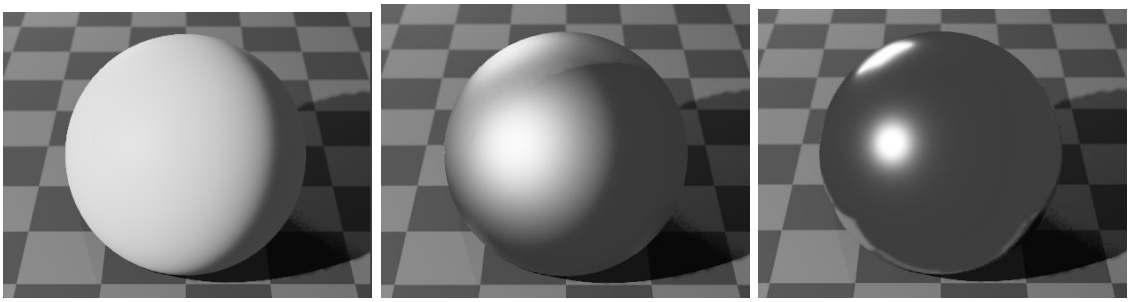
Vstup, výstup a teorie

Přestože se nejedná o převod nového typu objektů, z teoretického hlediska je potřeba řešit opět stejné tři otázky: jak reprezentovat granulárnější nastavení vizuálních vlastností v Blenderu, jak ho reprezentovat ve formátu SVG a jak mezi těmito reprezentacemi provést převod.

Materiály v Blenderu

Z hlediska Blenderu se jako nejideálnější řešení jeví už dříve několikrát zmíněné materiály. Materiály obecně v počítačové grafice lze chápat jako soubor hodnot, které definují vlastnosti nějakého objektu. Nejčastěji slouží ke zjednodušenému popisu fyzikálních vlastností povrchu za účelem výpočtu barvy a intenzity světla odraženého směrem k pozorovateli při dopadu na objekt, a to takovým způsobem, aby výsledný obraz pozorovateli dával dojem, jako že je objekt vyroben z nějakého specifického materiálu [11].

V Blenderu lze přiřazovat různým typům objektů jako modely či křivky jejich vlastní materiály, čehož je využíváno například k tvorbě efektů plastu, skla, kovu nebo kůže. Materiál však nemusí být přiřazen nutně k objektu, jedná se o individuální datový blok, který lze vytvořit samostatně, specifikovat jeho vlastnosti a poté jej přiřazovat libovolnému počtu objektů odkazem. Objekt zároveň může odkazovat na více materiálů, například model může mít pro každou svou stěnu přiřazen jiný materiál [5, sekce „Rendering > Materials“]. Příklad aplikace různých materiálů na stejný objekt lze vidět na obrázku 3.33.



Obrázek 3.33: Příklad vizualizace totožného modelu v Blenderu s různými nastaveními materiálu k dosažení kovového a skleněného efektu

Myšlenka využití materiálů v rámci tohoto rozšíření je prostá. Ačkoliv jsou materiály primárně určeny k nastavování vlastností pro vykreslování realističtějších rastrových obrázků, není důvod je nevyužít také pro nastavování vlastností pro vykreslování vektorových obrázků. Namísto nastavování textury či koeficientů pro odraz světla lze uvažovat o nastavování například vzorku výplně nebo barvy a šířky tahu a okrajů.

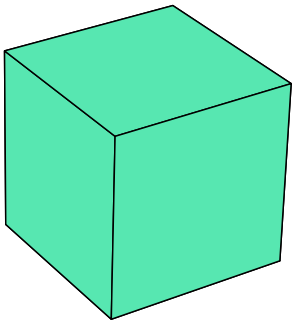
Samotné materiály Blenderu ve výchozím nastavení samozřejmě nezahrnují možnost specifikovat vlastnosti formátu SVG. Lze ovšem materiály o tyto vlastnosti rozšířit s využitím Blender API. Systém definice a rozšiřování vlastností v Blenderu je velmi důležitý pro tvorbu pluginů, jeho přesnější vysvětlení ovšem příliš zabíhá do implementačních detailů, nebude pro stručnost tedy dále v této sekci popisován. Jeho detailní popis lze nalézt v dokumentaci API [3, sekce „Property Definitions“]. Pro pochopení tohoto rozšíření stačí pouze fakt, že v rámci pluginu lze k materiálům dodefinovat libovolné nastavitelné skupiny vlastností různých datových typů.

Styly SVG s využitím jazyka CSS

Ke vhodné reprezentaci materiálů ve formátu SVG je nutno využít i jiných jazyků, které formát SVG podporuje a umožňuje vkládat do svého těla za účelem změny vizuálních vlastností jednotlivých prvků.

Z hlediska formátu SVG se jeví jako možnost reprezentace skupin vizuálních vlastností prvek <style>, původně převzatý z jazyka HTML. Uvnitř tohoto prvku lze definovat pravidla pro vizuální vlastnosti v jazyce CSS jakožto třídy vlastností. Jazyk CSS zároveň umožňuje tyto třídy přiřazovat prvkům v souboru (tedy jednotlivým tvarům formátu SVG). Definované vizuální vlastnosti dané třídy jsou poté aplikovány na všechny prvky, které do dané třídy patří [16, sekce 6.10 a 6.11].

Namísto současné specifikace vizuálních vlastností uvnitř každého prvku by tedy teoreticky mělo být možno nejdříve v hlavičce dokumentu definovat pomocí jazyka CSS vizuální vlastnosti pro jednotlivé třídy a v každém prvku poté pouze odkazovat na jeho příslušnou třídu. Tím by zároveň došlo i ke značnému zmenšení velikosti dat u rozsáhlejších obrázků. Příklad využití této kombinace jazyků lze vidět na následujícím obrázku 3.34.



```
<defs>
 <style>
    .cube {
         stroke-width : 1.0;
         stroke : rgb(0,0,0);
         stroke-opacity : 1.0;
         fill : rgb(87,231,177);
         fill-opacity : 1.0;
    }
 </style>
</defs>

<polygon points="107,67 107,55 100,49 100,61" class="cube"/>
<polygon points="107,55 119,51 111,46 100,49" class="cube"/>
<polygon points="118,63 119,51 107,55 107,67" class="cube"/>
```

Obrázek 3.34: Příklad využití tříd uvnitř formátu SVG. Napravo lze vidět řetězec formátu SVG/CSS definující třídu s vizuálními vlastnostmi a 3 polygony, které reprezentují tvar kostky a patří do této třídy. Nalevo je vykreslení tohoto řetězce.

Kompatibilita materiálů v Blenderu a stylových tříd vektorové grafiky by tedy po následujícím přirovnání měla být zřejmá:

- Systém materiálů Blenderu = prvek SVG <style>
- Materiál Blenderu = třída CSS
- Vlastnosti materiálu Blenderu = atributy třídy CSS
- Objekt Blenderu = prvek/skupina prvků formátu SVG
- Přiřazení materiálu k objektu Blenderu = přiřazení prvku SVG ke třídě CSS

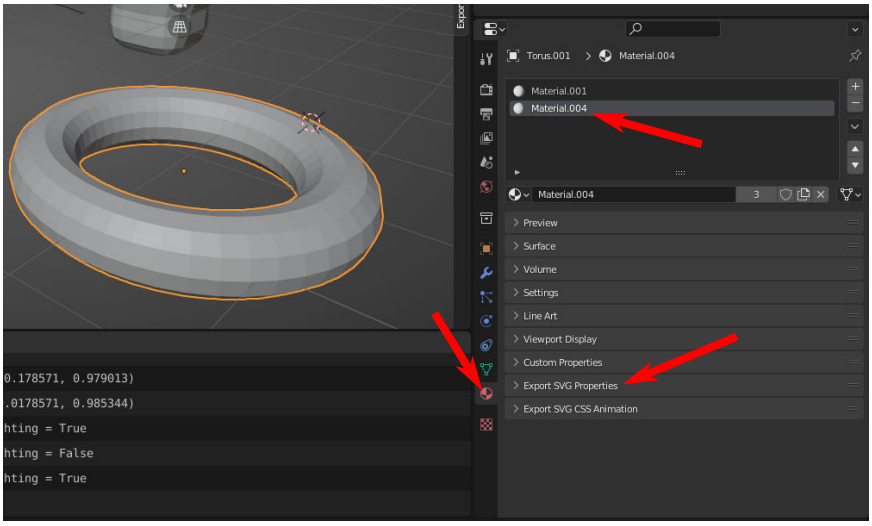
Návrh rozšíření

Možnosti nastavování vizuálních vlastností objektů budou z pohledu uživatele rozděleny do 2 částí. Současný způsob nastavování vlastností globálně pro všechny objekty daného typu sice neumožňuje individuální úpravy, jeho výhodou je však intuitivní, rychlá a snadná použitelnost, která nevyžaduje interakci s materiály. Z tohoto důvodu zůstane funkce zachována a v hlavním panelu pluginu bude stále možnost tímto způsobem vzhled upravovat.

Pro individuální (dále lokální) nastavení vzhledu bude muset uživatel využít materiály, které budou nabízet stejné možnosti jako hlavní panel, pouze omezené pro daný materiál.

Zároveň bude potřeba vytvořit pohled na tato nová nastavení materiálu, kterým bude nový panel s totožnou strukturou jako existující panely pro nastavování globálního vzhledu v hlavním panelu. Nový panel bude umístěn přímo mezi výchozími panely sloužícími pro definici vlastností upravovaného materiálu, neboť je relevantní pouze v kontextu právě vybraného materiálu. Příklad umístění panelu lze vidět na obrázku 3.35.

Objekt ovšem nemusí mít vždy přiřazen materiál, v tom případě bude panel materiálu pouze zobrazovat současná globální nastavení z hlavního panelu. Globální nastavení vzhledu bude aplikováno na objekt tehdy, pokud daný objekt nemá přiřazen žádný materiál, nebo pokud bude zaškrtnuta pro daný typ objektů nová vlastnost „Always Use Global Settings“, což bude nastavení, které umožní uživateli vypnout vliv materiálů a vždy použít globální nastavení pro všechny objekty daného typu.



Obrázek 3.35: Umístění panelu umožňujícího nastavení nově definovaných vlastností materiálů. Po selekci objektu lze vybrat záložku s materiály, kde se po výběru materiálu mezi panely zobrazí nový panel s názvem Export SVG Properties.

Při převodu bude každý použitý materiál převáděn na odpovídající třídu jazyka CSS. Globální nastavení každého typu budou také převáděna na samostatné třídy CSS, přestože ve skutečnosti materiálem v Blenderu nejsou.

Posledním problémem návrhu je řešení polygonů. Kvůli osvětlení mohou polygony stejného materiálu nabývat různých barev výplně a okrajů, není tedy možno je popsat třídou specifikující jednu pevnou barvu. U materiálů, které nevypínají vliv světla na barvu, budou tedy generovány 2 třídy. Jedna pro nepolygonové typy a druhá speciálně pro polygony, která nebude určovat barvu výplně a okrajů. Polygony budou tyto informace obsahovat ve vlastním těle jako doposud.

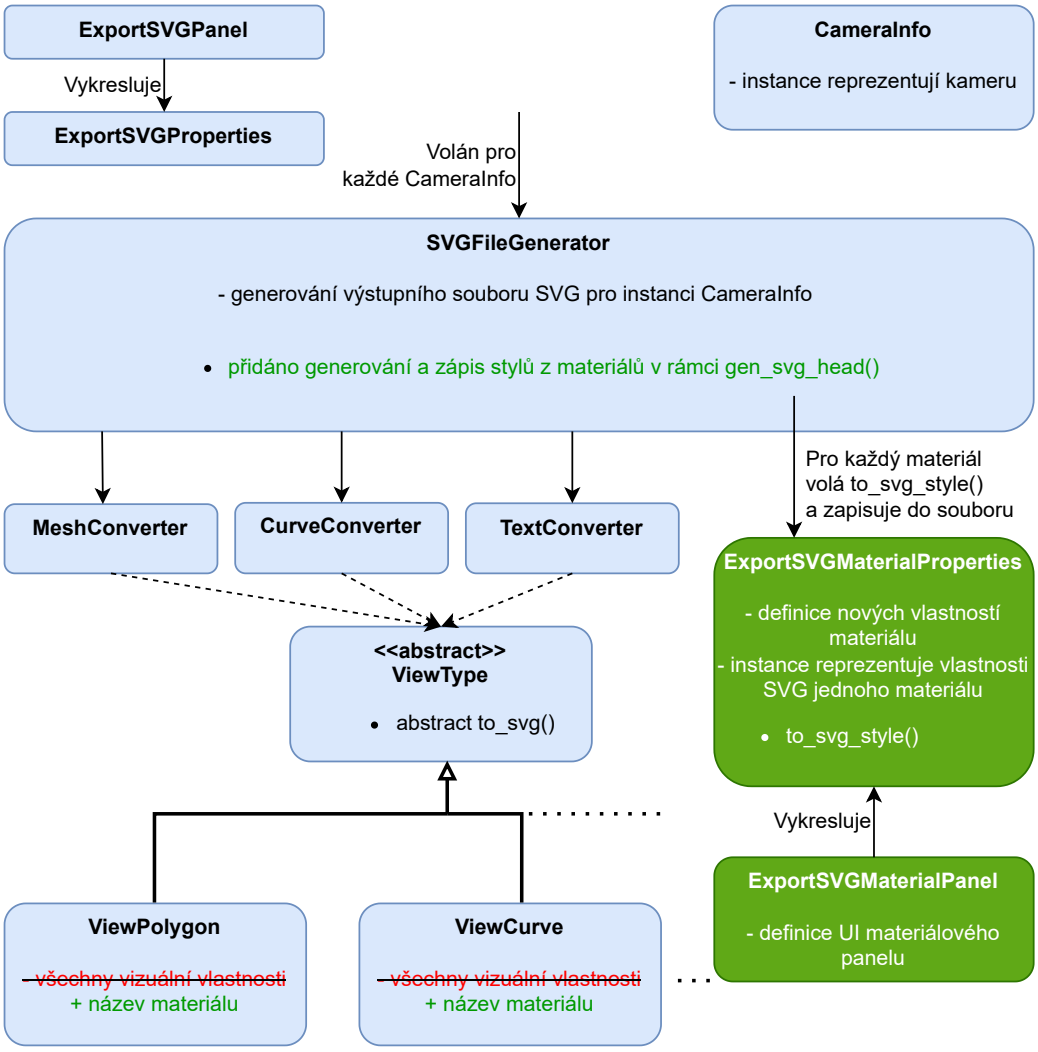
Implementace rozšíření

Implementace byla v rámci tohoto rozšíření složitější, neboť zasahovala do základů pluginu a vyžadovala definici velkého množství dodatečných vlastností a úprav rozhraní metod převodu. Diagram hlavních změn architektury lze vidět na obrázku 3.36.

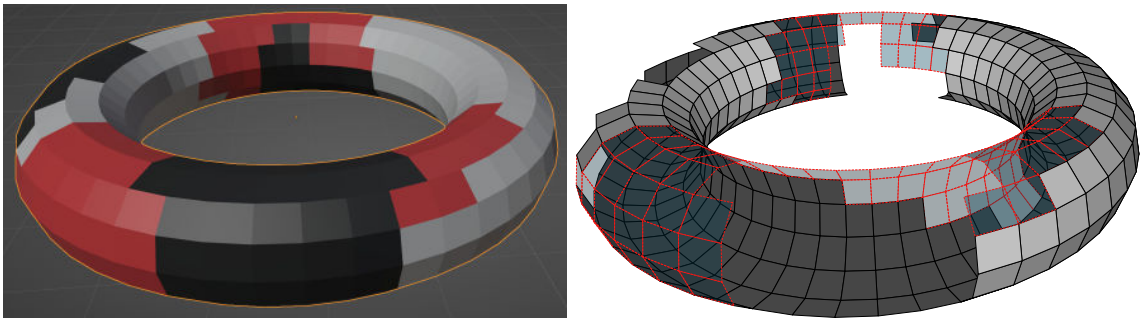
Výsledky

V rámci rozšíření byla úspěšně implementována vazba na systém materiálů v Blenderu. Uživatel může nově kromě globálního nastavení vizuálních vlastností objektů specifikovat tyto vlastnosti pouze pro daný materiál, který lze poté libovolně aplikovat na jednotlivé objekty či přenášet mezi scénami a projekty. Zároveň je tak značně snížena redundance ve výsledném souboru SVG. Výsledky, kterých lze díky tomuto rozšíření nově dosáhnout, lze vidět na obrázcích 3.37 a 3.38.

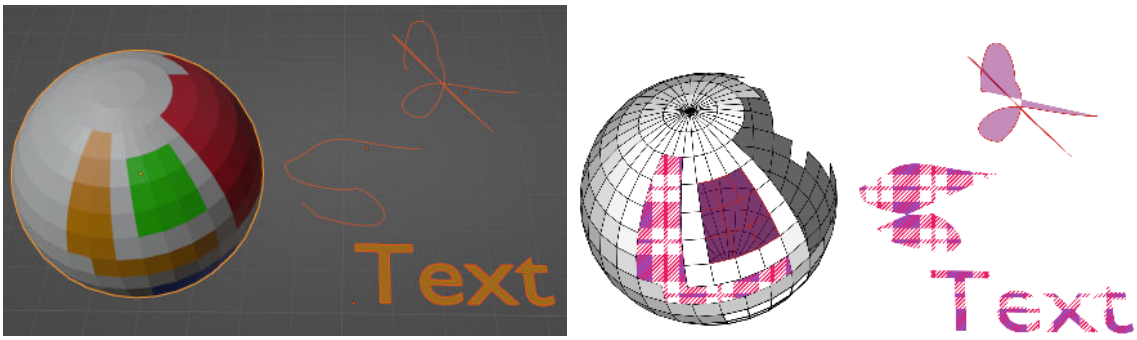
Systém napojení na materiály je navíc velmi versatilní a snadno rozšiřitelný do budoucna. Lze jej využít k úpravě vzhledu či jiných vlastností kteréhokoliv v budoucnu implementovaného typu objektů, pokud tedy tento typ může mít nastaven materiál. Navíc lze materiály velmi snadno rozšiřovat o vlastnosti reprezentující nové atributy formátu SVG.



Obrázek 3.36: Diagram hlavních změn po implementaci materiálového systému. Zeleně jsou vyznačeny nově přidané části, červeně části odebrané. Byla definována nová skupina vlastností v podobě třídy ExportSVGMaterialProperties, která rozšiřuje výchozí vlastnosti materiálů v Blenderu, a to tak, že každý materiál nově obsahuje její instanci. Každá instance této třídy navíc umí převést sebe sama na definici třídy a jejích vizuálních vlastností ve formátu SVG/CSS. Při generování hlavičky souboru SVG ve třídě SVGFileGenerator je pro každý použitý materiál ve scéně převedena jeho instance ExportSVGMaterialProperties na definiční řetězec vizuální třídy jazyka CSS a následně je zapsána do hlavičky. Díky tomu už jednotlivé 2D tvary v těle souboru nemusí obsahovat všechny vizuální vlastnosti, ale pouze odkazují na tyto třídy definované v hlavičce.



Obrázek 3.37: Příklad využití více materiálů v jednom modelu. Nalevo lze vidět původní scénu v Blenderu s modelem, na kterém jsou barvami odlišeny skupiny stěn modelu se stejným materiálem. Napravo lze vidět vykreslený výstupní soubor vektorové grafiky. Bílému materiálu byla nastavena bílá výplň s černými okraji, červenému průhledná modrá výplň s červenými okraji a černý materiál byl nastaven na neviditelný, resp. kompletně průhledný.



Obrázek 3.38: Příklad sdílení materiálů mezi více objekty. Nalevo lze vidět původní scénu v Blenderu s modelem s více materiály, křivkami a textem. Napravo lze vidět vykreslený výstupní soubor. Text a jedna z křivek sdílí stejný materiál jako oranžově vyznačená plocha modelu. Druhá křivka sdílí s modelem materiál jeho zeleně vyznačené plochy.

3.10 Podpora evaluace

Po předchozím značně rozsáhlém rozšíření je v této podkapitole přiblíženo krátké rozšíření umožňující uživateli převádět objekty v jejich stavu po evaluaci, tedy například po aplikaci modifikátorů, kvůli kterým toto rozšíření bylo původně implementováno a na které je převážně zaměřeno.

Původní stav a cíl

V původním pluginu není na evaluaci objektů brán ohled. Přestože v uživatelově pohledu na scénu je objekt vykreslen například včetně modifikátorů, na vektorový obrázek je převedena pouze základní neupravená geometrie objektu. Uživatel tak tedy musí pro převod modifikovaného objektu tyto úpravy nejdříve destruktivně aplikovat. Tím je však objekt permanentně upraven a nelze vlastnosti jeho modifikátoru dále upravovat aniž by uživatel provedl kroky zpět v historii celé scény, což je značně nepraktické, obzvlášť pro více objektů současně.

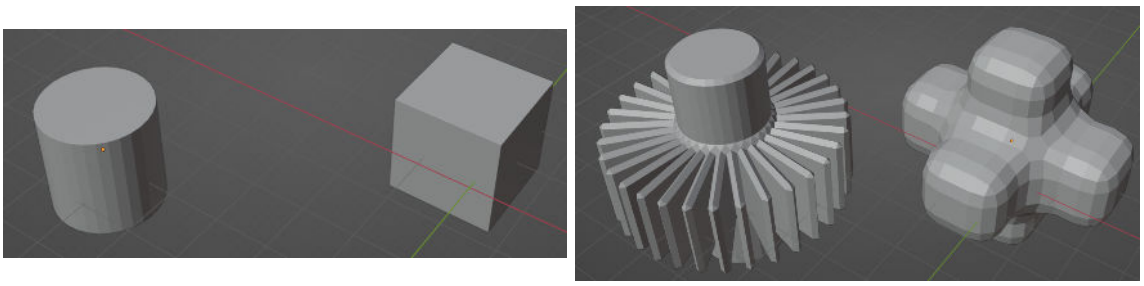
Cílem tohoto rozšíření je dát uživateli možnost výběru, zdali chce vybrané objekty převést takovým způsobem, jakoby byly všechny jejich úpravy aplikovány, aniž by tuto akci musel provádět sám ručně.

Vstup, výstup a teorie

Relevantní částí Blenderu pro toto rozšíření je graf závislostí (Dependency Graph, depsgraph). Jedná se o graf scény, kde uzly jsou jednotlivé entity scény (jako například objekty) a hrany jsou závislosti mezi nimi. Tento graf existuje za účelem efektivní aktualizace scény a zaručuje, že pokud dojde k její změně, tak jsou přepočítávány pouze entity závislé na měněné entitě, nikoliv celá scéna. Zároveň graf zohledňuje i různé dynamické aktualizace scény jako animaci či modifikátory objektů. Lze v něm tedy nalézt i přepočítané kopie objektů scény ve stavu po evaluaci těchto modifikací [4].

Blender API poskytuje snadný přístup k tomuto grafu. Zároveň API implementuje u každého objektu metodu, které lze graf závislostí předat a která vrací výslednou podobu daného objektu po aplikaci všech jeho modifikací.

Nejčastějším příkladem takových modifikací jsou právě výše zmíněné modifikátory, což je skupina operací ovlivňujících geometrii objektu nedestruktivním způsobem. Tyto operace pouze ovlivňují způsob jakým je objekt zobrazován a vykreslován, nikoliv přímo jeho základní geometrii jako například pozice vrcholů [5, sekce „Modeling > Modifiers“]. Příklad velmi jednoduchých modifikátorů lze vidět na následujícím obrázku 3.39.



Obrázek 3.39: Příklad základních modifikátorů objektů. Nalevo lze vidět původní objekty ve scéně Blenderu. Napravo lze vidět ty stejné objekty po aplikaci stejné dvojice modifikátorů, prvním z nich je modifikátor Geometry Nodes s jediným uzlem pro extruzi, druhým je modifikátor Bevel pro vyhlazení hran a vrcholů.

Blender nabízí kolem 50 různých typů modifikátorů aplikovatelných na různé typy objektů s mnoha nastavitelnými parametry této aplikace. Mezi nimi je i extrémně versatilní modifikátor zvaný „Geometry Nodes“, který umožňuje aplikovat další podoperace reprezentované uzly zapojenými do rozsáhlé sítě. V Blenderu je možno použít kolem 200 různých typů uzlů, opět s různými parametry. Bližší popis funkcionality všech modifikátorů včetně indexu typů lze nalézt v manuálu Blenderu [5, sekce „Modeling > Modifiers > Modify“].

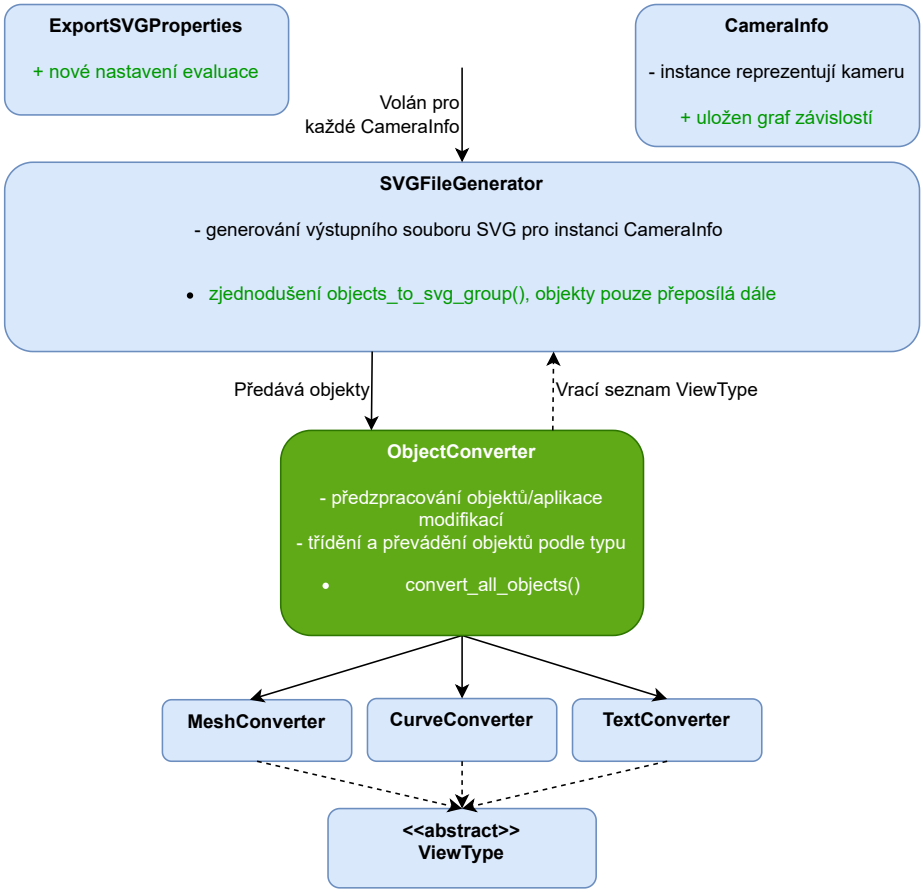
Z hlediska formátu SVG není zapotřebí žádných nových poznatků.

Návrh rozšíření

Z pohledu uživatele bude práce s rozšířením v rámci pluginu triviální. Uživatel bude moci pracovat s modifikátory a jinými úpravami stejně jako doposud skrze existující menu Blenderu. V hlavním panelu pluginu v sekci Export bude pouze přidána možnost volby, zdali objekty před převodem na vektorovou grafiku evaluovat, či ponechat v základním stavu.

Implementace rozšíření

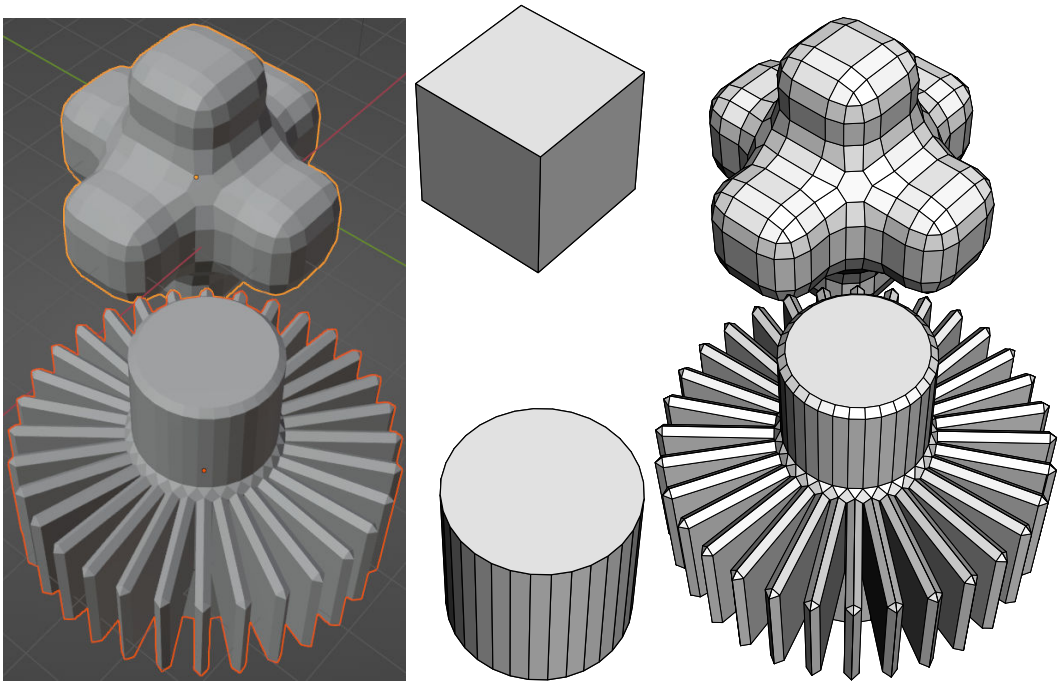
Samotná implementace aplikování evaluací byla díky možnostem Blender API poměrně snadná. Jádro implementace, kterým je převod objektu z původního na modifikovanou kopii, se skládá pouze z volání 2 speciálních metod API. Z důvodu zlepšení architektury a rozšiřitelnosti kódu však došlo v rámci rozšíření k dalším úpravám, které lze vidět na následujícím diagramu změn 3.40.



Obrázek 3.40: Diagram hlavních změn po implementaci podpory evaluace. Zeleně jsou vyznačeny nově přidané části. Byla vytvořena nová třída ObjectConverter, která zastřešuje všechny třídy pro převod jednotlivých typů. Nově tedy třída SVGFileGenerator nemusí třídit objekty a pouze je předává této třídě pro převod, zároveň třída ObjectConverter představuje společný vstupní bod převodu všech typů objektů, ve kterém je možno provádět jejich společné předzpracování, jako je v případě tohoto rozšíření evaluace objektů.

Výsledky

V rámci rozšíření byla úspěšně implementována podpora evaluace objektů, díky které může uživatel převádět objekty s modifikacemi bez nutnosti jejich destruktivní aplikace. Kvůli některým specifickým implementačním problémům ovšem konkrétně samotné modifikátory nefungují při aplikaci na křivky a text, ale pouze na modely a později implementované typy Grease Pencil. Výsledky implementace lze vidět na následujícím obrázku 3.41



Obrázek 3.41: Ukázka převodu objektů s evaluací. Nalevo lze vidět snímek obrazovky původní scény v Blenderu s 2 modifikovanými objekty. Uprostřed lze vidět vykreslený výstupní soubor bez aktivní evaluace, napravo s aktivní evaluací.

3.11 Převod Grease Pencil

V této podkapitole je přiblíženo rozšíření umožňující převod objektů v Blenderu typu „GPENCIL“ (Grease Pencil) na prvky <path> formátu SVG. Protože jsou tyto objekty v SVG reprezentovány křivkami, rozšíření je závislé na dřívějším rozšíření týkajícím se převodu křivek.

Původní stav a cíl

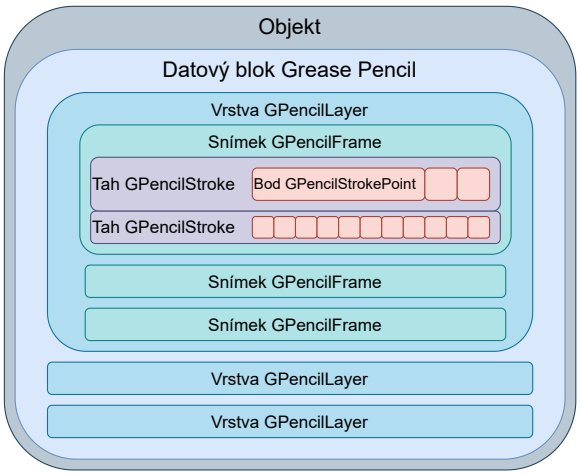
Původní plugin objekty typu Grease Pencil při převodu ignoruje. Po předchozích rozšířeních už však dokáže převádět křivky, které sdílí s tímto typem mnoho společných rysů a díky kterým už plugin obsahuje potřebné třídy pro reprezentaci 2D křivek formátu SVG.

Cílem rozšíření je implementovat převod objektů tohoto typu, který je na rozdíl od objektů typu „CURVE“ vhodnější pro rychlé a snadné kreslení do scény.

Vstup, výstup a teorie

Grease Pencil je speciální typ objektů v Blenderu, který umožňuje na základě vstupu z polohovacích zařízení jako myš nebo pero grafického tabletu umisťovat skupiny bodů do 3D scény jakožto tahy. Příkladem jeho využití jsou 2D animace, neboť umožňuje definovat jiné tahy pro každý časový okamžik. Veškeré aspekty využití a práce s nástrojem Grease Pencil týkající se například různých módu kreslení, masek či principu animací v Blenderu ovšem nebudou v rámci této sekce popisovány, neboť tento typ objektů je poměrně komplexní a i pouze stručný popis jeho základních rysů by byl příliš rozsáhlý. Další informace lze případně nalézt v přehlednějším formátu v manuálu Blenderu [5, sekce „Grease Pencil“].

Záměrem této sekce je objasnění struktury těchto objektů z hlediska Blender API, která je relevantní pro jejich převod na formát SVG. Kvůli podpoře animací a 2D kreslení je jejich struktura poněkud složitější oproti dříve rozebíraným křivkám a má svou specifickou hierarchii [3, sekce „Types > GreasePencil“], kterou lze shrnout obrázkem 3.42.



Obrázek 3.42: Vizualizace hierarchické struktury objektů typu Grease Pencil. Objekt scény v sobě obsahuje datový blok Grease Pencil. Ten je složen z více vrstev (GPencilLayer) v daném pořadí, kdy podobně jako v 2D grafice prvky vyšší vrstvy budou vždy překrývat prvky nižší vrstvy bez ohledu na skutečnou hloubku ve scéně. Každá vrstva je složena z více snímků (GPencilFrame) pro různé časové okamžiky animace. V rámci každého snímku jsou uloženy všechny jeho tahy (GPencilStroke). V samotných tazích je pak uložen seznam jednotlivých bodů (GPencilStrokePoint), které tah tvoří, s jejich prostorovými souřadnicemi.

Podobně jako k většině ostatních typů objektů v Blenderu je možno i k objektům Grease Pencil přiřazovat materiály, které lze poté aplikovat odděleně na jednotlivé podčásti objektu, v tomto případě na jednotlivé tahy.

V rámci formátu SVG není potřeba uvádět žádné nové poznatky, neboť samotná geometrie těchto objektů není ničím jiným než kolekcemi mnoha bodů spojených do jednotlivých tahů. Takový typ geometrie je snadno reprezentovatelný dříve popsaným prvkem <path> pro reprezentaci křivek z rozšíření 3.2.

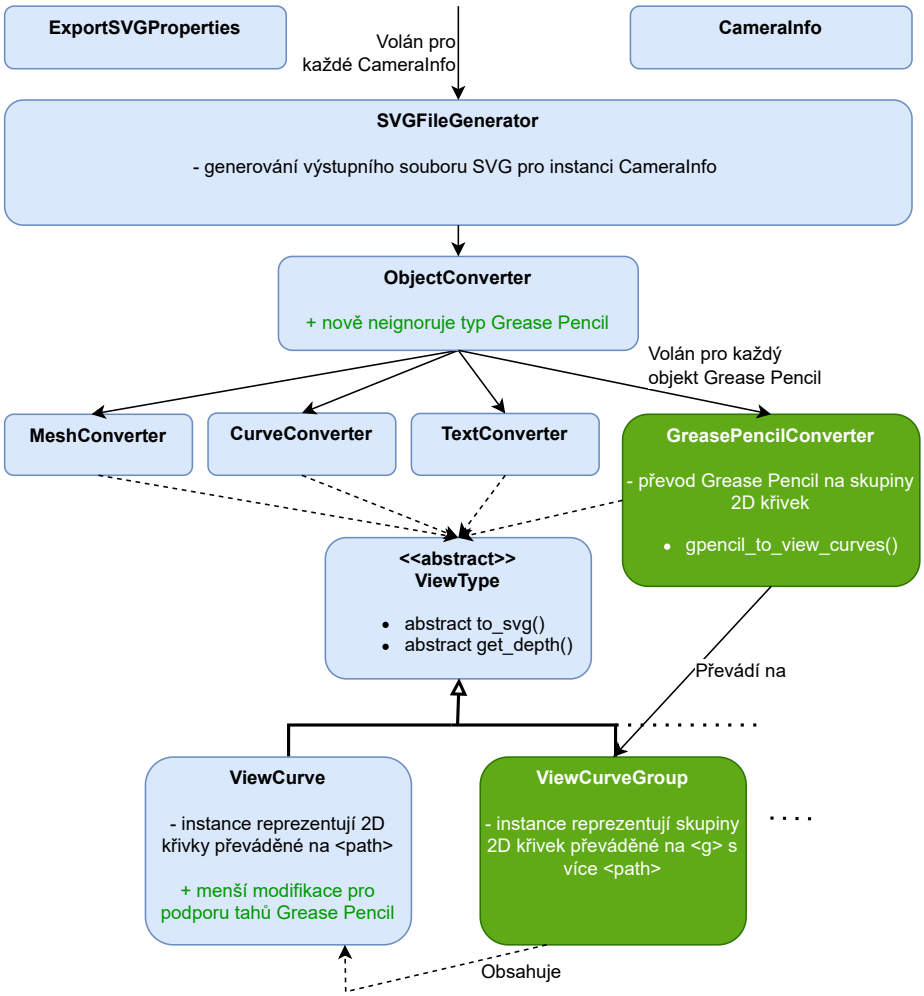
Návrh rozšíření

Základní práce s Grease Pencil objekty bude z pohledu uživatele stejná jako u předchozích typů. Po kliknutí na tlačítko pro převod budou nově převáděny i objekty typu „GPENCIL“. Nastavení vizuálních vlastností v uživatelském rozhraní budou tyto typy sdílet s křivkami, neboť se prakticky jedná pouze o alternativní reprezentaci křivek.

Celý objekt bude reprezentován jedním seskupovacím prvkem <g> s více prvky <path> reprezentujícími jednotlivé tahy. Tyto prvky <path> budou převáděny v takovém pořadí, v jakém jsou jejich odpovídající vrstvy, aby byl zachován původní vzhled objektu co se týče pořadí vrstev. Pro každou vrstvu bude pro jednoduchost převáděn pouze aktuální snímek animace (tedy ten snímek animace, který v okamžik převodu scéna zobrazuje).

Implementace rozšíření

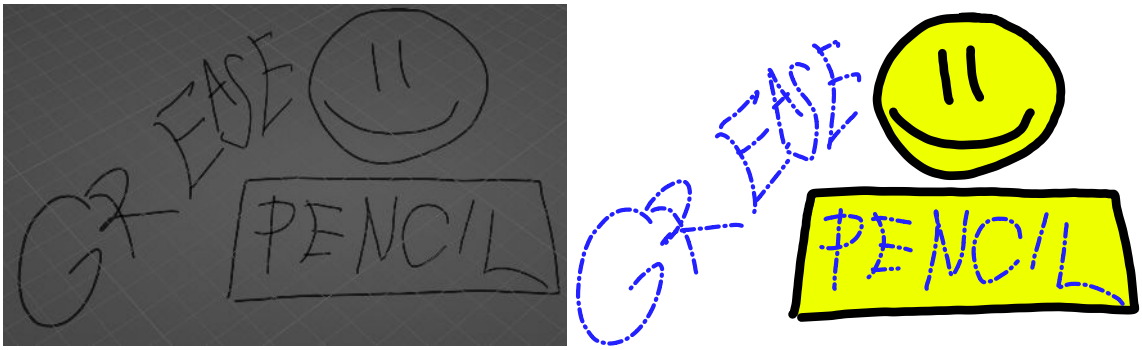
Implementace rozšíření byla díky už stávající architektuře pluginu a podpoře křivek relativně prostá. Byly zachovány stejné principy jako u předchozích typů, jak lze vidět na diagramu změn 3.43.



Obrázek 3.43: Diagram hlavních změn po implementaci převodu objektů Grease Pencil. Zeleně jsou vyznačeny nově přidané části. Analogicky k předchozím typům objektů byla vytvořena třída GreasePencilConverter pro převod objektů Grease Pencil na 2D křivky. Tyto skupiny 2D křivek jsou reprezentovány instancemi nové třídy ViewCurveGroup, které v podstatě pouze seskupují už existující instance třídy ViewCurve reprezentující jednotlivé 2D křivky.

Výsledky

V rámci rozšíření byl úspěšně implementován převod objektů Grease Pencil na křivky vektorové grafiky, který je kompatibilní s předchozími rozšířeními jako podpora evaluace či materiálů. Výsledky dosažitelné při různých kombinacích nastavení lze vidět na následujících obrázcích 3.44 a 3.45.



Obrázek 3.44: Ukázka převodu objektu Grease Pencil. Nalevo lze vidět snímek obrazovky původní scény v Blenderu s objektem, kterému jsou nastaveny dva různé materiály s různými vizuálními vlastnostmi SVG pro jeho tahy, napravo vykreslený výstupní soubor.

Obrázek 3.45: Ukázka převodu složitějšího objektu Grease Pencil. Nalevo lze vidět původní scénu v Blenderu se složitějším objektem Grease Pencil s více vrstvami a materiály. Napravo lze vidět vykreslení převedeného souboru. Lze si všimnout korektního zachování pořadí vrstev jednotlivých tahů. Proměnlivost intenzity (šířky) čar v jednotlivých částech tahů ovšem není zachována, neboť formát SVG umožňuje pouze konstantní šířku tahu.

3.12 Převod anotací

V této podkapitole je stručně přiblíženo rozšíření umožňující převod anotací scény v Blenderu na prvky <path> formátu SVG. Toto rozšíření navazuje na předchozí rozšíření týkající se převodu objektů Grease Pencil.

Původní stav a cíl

Původní plugin sice s anotacemi nepracuje, po předchozích rozšířeních už však dokáže převádět křivky a objekty Grease Pencil, které s anotacemi úzce souvisí. Potřebné komponenty pro převod už tedy z větší části obsahuje.

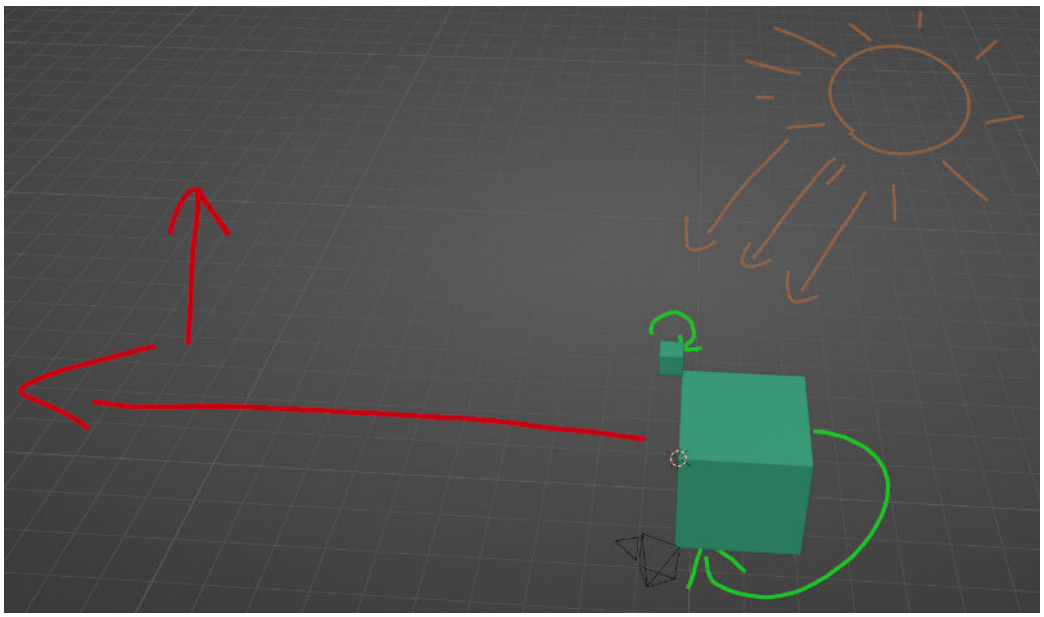
Cílem rozšíření je umožnit uživateli kreslit a převádět popisky využitím systému anotací.

Vstup, výstup a teorie

Blender poskytuje možnost využití speciálního nástroje zvaný anotace. Tento nástroj je dostupný ve více typech editorů a je použitelný pro kreslení či psaní popisků. Jedním z podporovaných editorů je i uživatelův pohled na 3D scénu, v rámci kterého celý plugin doposud pracuje [5, sekce „User Interface > Annotations“].

Systém anotací poskytuje několik způsobů jejich kreslení, anotace lze kreslit ručně táhnutím myši nebo kreslit jakožto rovné čáry či polygony specifikací jednotlivých vrcholů, poté je možné části anotací také umazávat.

Tahy anotací jsou vždy kresleny na nějaké z vrstev, které určují jejich vzájemné překrytí. U každé vrstvy lze nastavit několik málo parametrů. Pro určení vzhledu je možno nastavit barvu vrstvy, její průhlednost a šířku jejích tahů. Dále lze nastavit, zdali jednotlivé tahy vrstvy mají být vždy vykresleny před všemi ostatními objekty bez ohledu na hloubku, nebo zdali mají být překryty objekty, které se z pohledu kamery nachází před nimi. Příklad nakreslených anotací lze vidět na snímku obrazovky 3.46.



Obrázek 3.46: Příklad anotací nakreslených ve scéně Blenderu v podobě zelených, červených a oranžových tahů

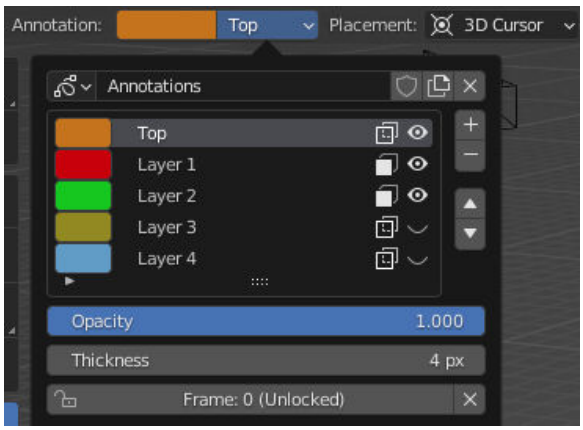
Z pohledu Blender API jsou anotace reprezentované stejným datovým typem jako objekty Grease Pencil z předchozího rozšíření, popis jejich struktury lze tedy nalézt v podkapitole 3.11. Hlavním rozdílem je však fakt, že u anotací není datový blok Grease Pencil „zaobalen“ do žádného nositelského objektu, namísto toho je navázán přímo na samotný kontext scény a anotace tak samy o sobě ani objektem nejsou. To znamená, že s anotacemi nelze provádět stejné operace jako s běžnými objekty, tedy jim například nelze přiřadit materiál nebo je zařadit do kolekce.

V rámci formátu SVG lze anotace snadno reprezentovat stejně jako objekty Grease Pencil pomocí prvků <path>.

Návrh rozšíření

Protože se anotace vyhýbají svou podstatou doposud převáděným objektům, je cílem udělat práci s nimi co nejsnadnější z pohledu uživatele. Uživatel bude moct v nastavení zvolit, zdali chce při převodu konvertovat i anotace. V takovém případě budou všechny viditelné anotační vrstvy převedeny na odpovídající křivky na výsledném obrázku.

Anotace neumožňují použití materiálů, jejich vzhled tedy bude určen jejich skutečnou podobou ve scéně a bude jí odpovídat jedna ku jedné. Respektovány budou při převodu všechny nastavitelné parametry anotací, kterými jsou jejich barva, šířka tahu, průhlednost, viditelnost a také možnost překrytí všech objektů bez ohledu na hloubku. Zachováno také bude pořadí anotačních vrstev, jejich vzájemné překrytí i možnost kreslit anotace na 2D pohled namísto do 3D scény. Nastavování vzhledu anotací bude probíhat ve výchozím prostředí Blenderu, které lze vidět na snímku obrazovky 3.47.



Obrázek 3.47: Uživatelské rozhraní nastavení anotací ve výchozím prostředí Blenderu

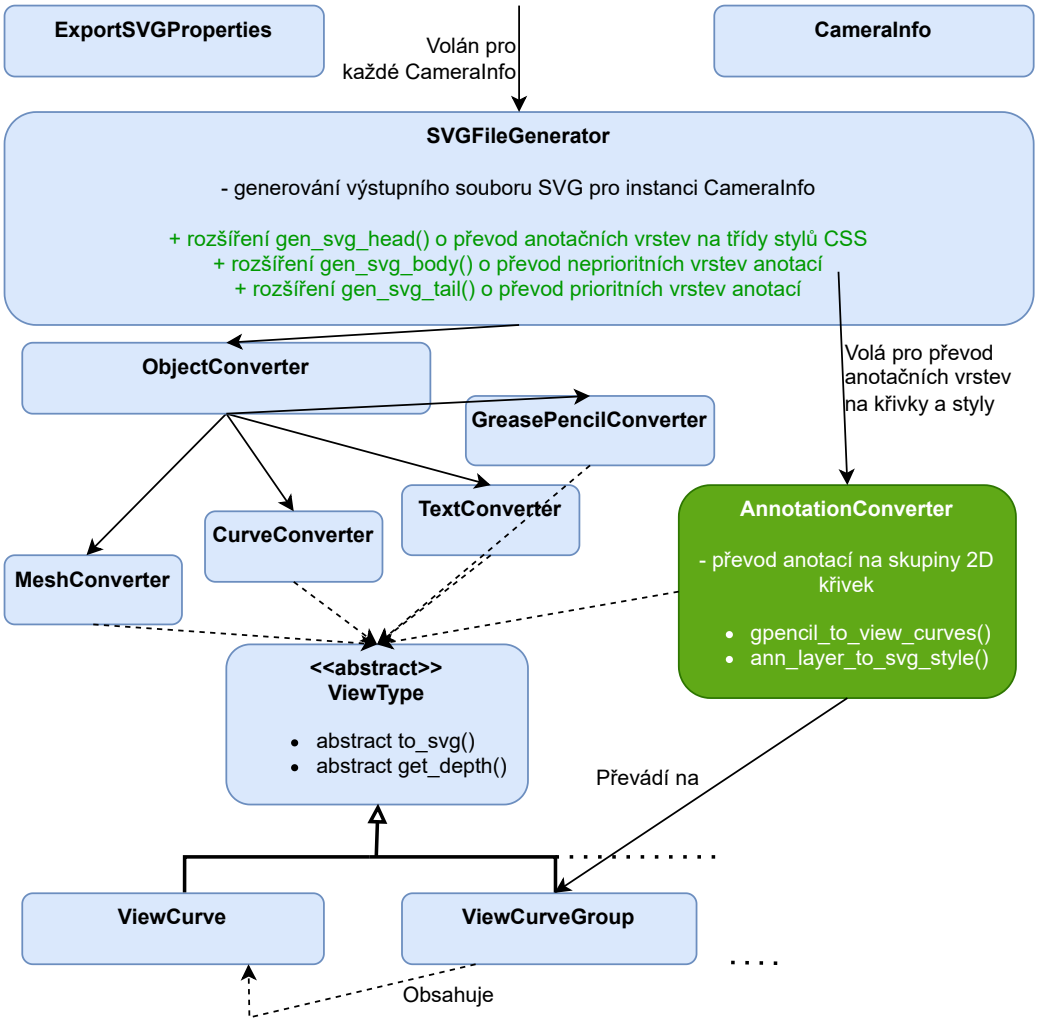
Implementace rozšíření

Implementace samotného převodu anotací byla triviální díky předchozímu rozšíření, které už do pluginu zavedlo převaděč typu Grease Pencil a třídu pro reprezentaci skupiny křivek. Komplikovanější byla integrace anotací do procesu převodu, neboť plugin doposud převáděl vždy pouze objekty. Diagram změn lze vidět na obrázku 3.48.

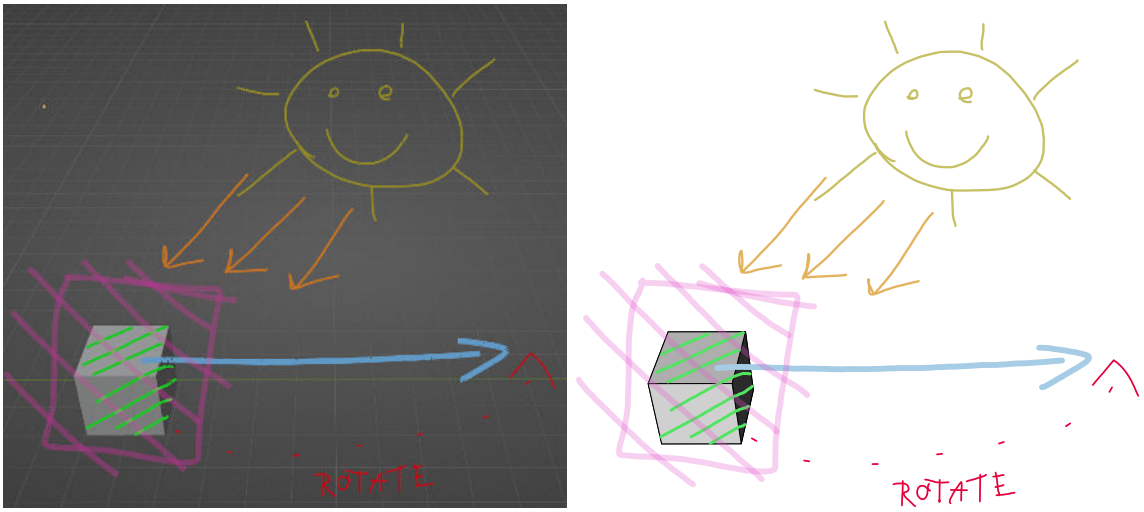
Integrace anotací do převodu vyžadovala tvorbu mnoha výjimek na vyšších úrovních procesu, čímž poměrně zdegradovala architekturu kódu v těchto místech. Prozatím tento problém nebyl dále řešen, v budoucnu by však při dalších úpravách byla vyžadována rozsáhlejší refaktorizace.

Výsledky

V rámci rozšíření byl úspěšně implementován převod anotací, díky kterému může uživatel do scény snadno a rychle vkreslovat poznámky. Výsledky dosažitelné tímto rozšířením lze vidět na obrázku 3.49.



Obrázek 3.48: Diagram hlavních změn po implementaci převodu anotací. Zeleně jsou vyznačeny nově přidané části. Analogicky k předchozím typům byla vytvořena třída AnnotationConverter pro převod anotací na skupiny 2D křivek, která je z velké části podobná třídě GreasePencilConverter. Tato třída však existuje nezávisle od ostatních a nevyužívá ji třída ObjectConverter, neboť anotace nejsou objekty. Musí být tedy zvlášť speciálně volána ve třídě SVGFileGenerator při generování hlavičky (převod stylů anotační vrstvy na třídu stylů CSS), při generování těla (převod anotací na 2D křivky) a při generování konce souboru (převod prioritních anotací překrývajících ostatní objekty na 2D křivky).



Obrázek 3.49: Příklad převodu anotací. Nalevo lze vidět snímek obrazovky původní scény v Blenderu s objektem a několika popisky různých vrstev s různými barvami, průhlednostmi a šířkami tahů. Napravo lze vidět vykreslený výstupní soubor převodu této scény.

3.13 Podpora animací

V této podkapitole je přiblíženo poslední rozsáhlé rozšíření umožňující uživateli vytvářet animované vektorové obrázky s využitím animačních prvků a atributů jazyka CSS, které formát SVG podporuje. Toto rozšíření staví na materiálovém systému představeném v rámci podkapitoly 3.9.

Původní stav a cíl

Původní plugin byl určen pro export pohledu na scénu a lze s ním tedy vytvářet pouze statické obrázky. Avšak například při tvorbě diagramů může být pro uživatele vhodné, aby tvary na obrázku měnily své vizuální vlastnosti v závislosti na čase. Může jít o zvýraznění důležitých částí změnou barvy či pohybem, nebo o postupné zobrazování prvků obrázku.

Cílem tohoto rozšíření je tedy umožnit uživateli vytvářet vektorové obrázky, které jsou schopny plynutím času měnit vizuální atributy svých jednotlivých prvků.

Vstup, výstup a teorie

Rozšíření podobně jako v případě materiálů řeší tři problémy: reprezentaci animací v Blenderu, které uživatel nastavuje, reprezentaci animací v souboru SVG a převod mezi těmito reprezentacemi.

Blender sám o sobě animace podporuje zabudovanou časovou osou, na které je možné vytvářet animační sekvence [5, sekce „Editors > Timeline“]. Samotná práce s touto osou ale může být poměrně složitá pro uživatele nevyužívající animace v Blenderu, obzvlášť, pokud by toto rozšíření muselo implementovat značné modifikace pro potřeby kompatibility s pluginem. Komplikací s jejím využitím pro reprezentaci animací souborů SVG je více, pro stručnost dále nebude tento prvek Blenderu blíže popisován, neboť nebude ve výsledném pluginu nijak využit.

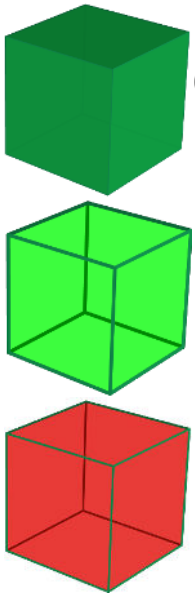
Pro samotnou reprezentaci animací bude jednodušší implementovat vlastní menu určené specificky pro animace SVG. Z hlediska Blender API bude tedy pro toto rozšíření relevantní

pouze systém materiálů v Blenderu a možnost rozšiřovaní jejich vlastností definicí nových skupin vlastností. Tyto části byly podrobněji popsány v teoretické sekci dřívější podkapitoly materiálů 3.9.

Mnohem důležitější je způsob reprezentace animací v souborech SVG, které lze animovat více způsoby[3]. Později byl v rámci návrhu k animaci zvolen pro svou jednoduchost jazyk CSS, jako jediný je tedy dále podrobněji popsán.

Animace CSS se zjednodušeně skládají ze 2 hlavních částí. Jazyk umožňuje ke každé třídě v hlavičce souboru (které v rámci tohoto pluginu reprezentují materiály z dřívějšího rozšíření) přidávat atribut `animation`, kterým lze definovat vlastnosti jako délka cyklu animace, zpoždění, směr či rychlost opakování a další. Hlavní vlastností je ale název, resp. odkaz na klíčové snímky animace [17, sekce „animation“].

Druhou částí jsou poté tyto samotné klíčové snímky, které lze opět definovat v hlavičce souboru, tentokrát jako `@keyframes název`. Uvnitř tohoto prvku lze specifikovat jednotlivé snímky uvedením jejich procentuální hodnoty, která označuje, jakou část jednoho animačního cyklu snímek reprezentuje [12, sekce „CSS > Guides > Animations“]. Lépe lze definici pochopit na následujícím příkladu 3.50.




```
.mat_ColorCube {
          stroke-width : 1.0;
          animation: 10.0s anim_mat_ColorCube;
}

@keyframes anim_mat_ColorCube {
       30% {
          stroke-width : 5.0;
          fill : rgb(0,255,0);
          fill-opacity : 0.5;
       }

       60% {
          stroke-width : 3.0;
          fill : rgb(255,0,0);
          fill-opacity : 0.5;
       }
}
```

Obrázek 3.50: Příklad využití animací jazyka CSS uvnitř formátu SVG. Napravo lze vidět řetězec formátu SVG definující vizuální třídu pro polygony tvořící kostku včetně atributu 10 sekund dlouhé animace a jejích klíčových snímků. Nalevo je potom vykreslení polygonů této třídy v čase 0 sekund (výchozí stav, odpovídá také času 10 sekund), 3 sekundy (šířka tahu 5 a zelená průhledná výplň) a 6 sekund (šířka tahu 3 a červená průhledná výplň) od počátku animace. Časům 0 a 10 sekund od počátku odpovídá výchozí stav pouze tehdy, pokud pro ně nebyly definovány klíčové snímky (0% a 100%). Mezi těmito obrázky, resp. mezi atributy prvků na obrázcích, je v časech mezi klíčovými snímky interpolováno.

Poslední relevantní částí je nový, dosud nepoužitý atribut CSS `transform`. Tento atribut umožňuje aplikovat na prvky v souboru transformace, tedy posunutí, změnu měřítka, rotaci a zkosení [17, sekce „transform“]. Zároveň podporuje interpolaci, lze ho tedy využít v rámci animací pro animování pohybu.

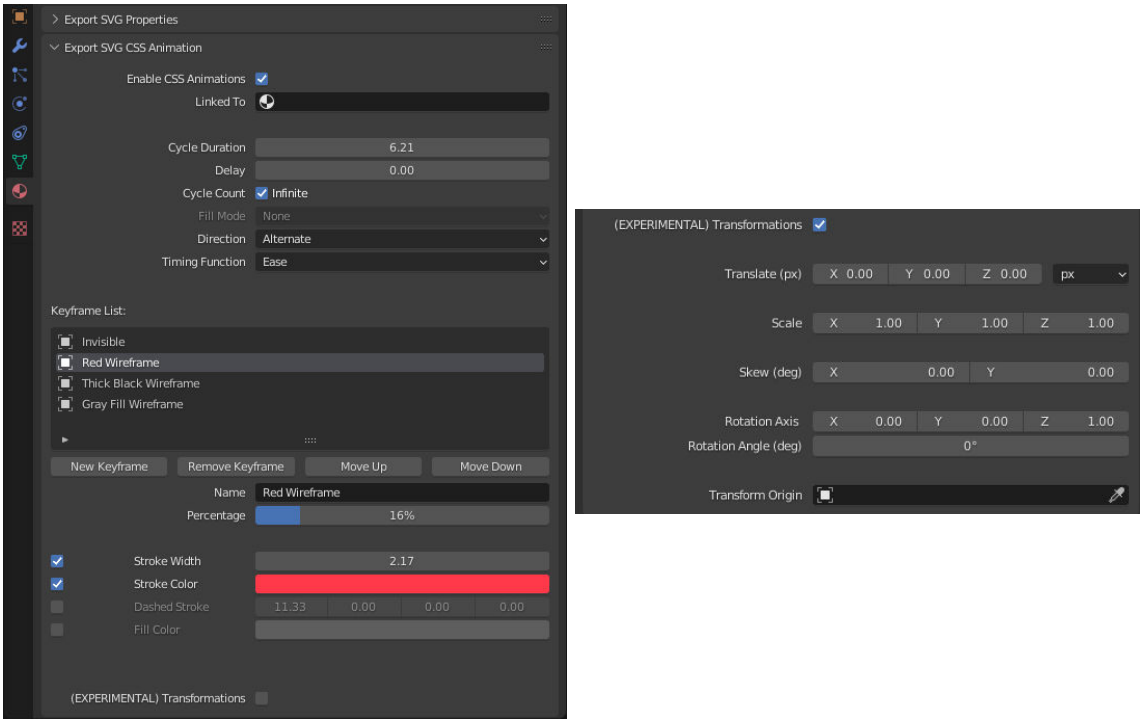
[3]Stručný přehled možností lze nalézt například zde: https://www.w3.org/TR/SVG2/animate.html

Návrh rozšíření

Animace budou z pohledu uživatele navázány na materiály kvůli kompatibilitě tříd a animací CSS. Pro každý materiál bude uživatel moct kromě definice vizuálních vlastností aktivovat animace. To mu umožní definovat globální vlastnosti animace pro daný materiál, konkrétně délku, zpoždění, směr, počet iterací a způsob interpolace. Dále mu to umožní pro daný materiál definovat seznam libovolného počtu klíčových snímků, pro které bude moci opět určit jejich vizuální vlastnosti včetně případných transformací.

Uživateli bude umožněno pro každý snímek animovat barvu i průhlednost výplně a tahu, šířku tahu a čárkování tahu a všechny typy transformací. Pro specifikaci pořadí snímků a jejich umístění v čase bude možno definovat jejich dříve zmíněnou procentuální hodnotu v rámci cyklu. Pro případ použití více materiálů ve scéně či objektu, které by uživatel chtěl animovat stejným způsobem, bude existovat také možnost propojení animací materiálů. Uživatel pro daný materiál zvolí v sekci animací vzorový materiál, vlastnosti jeho animací poté budou zkopírovány z toho vzorového a nebude potřeba je znovu nastavovat.

Pro vizualizaci vlastností týkajících se animací materiálu bude vytvořen nový panel UI, který bude umístěn na stejném místě jako panel pro vlastnosti z materiálového rozšíření. Pro snazší pochopení nastavování animací lze výsledný stav vidět na obrázku 3.51.

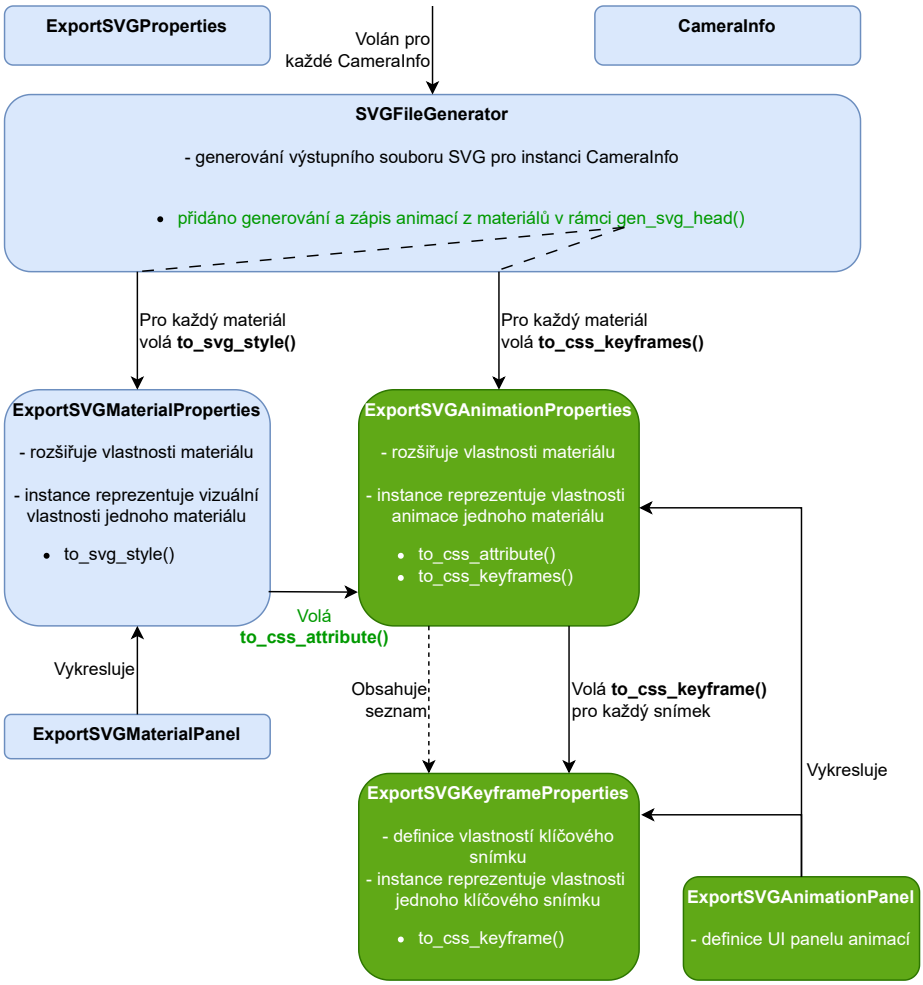


Obrázek 3.51: Výsledný stav nového panelu pro animace. Nalevo lze vidět nový panel animací umístěný na kartě materiálů hned pod materiálovým panelem z dřívějšího rozšíření. V první polovině panelu se nacházejí globální nastavení animace, v druhé polovině poté seznam klíčových snímků s jejich individuálními nastaveními. Napravo lze vidět rozbalenou část panelu pro specifikaci transformací zvoleného klíčového snímku.

K vizualizaci seznamu klíčových snímků bude sloužit speciální prvek `UIList`, který je oproti většině částí Blender API týkajících se UI poměrně složitý, k jeho implementaci tedy bude kromě dokumentace využit i online návod komunity Blenderu [7].

Implementace rozšíření

Implementace vyžadovala definici velkého množství vlastností napojených na materiály a integraci převodu do generování hlavičky souboru. Diagram změn lze vidět na obrázku 3.52.



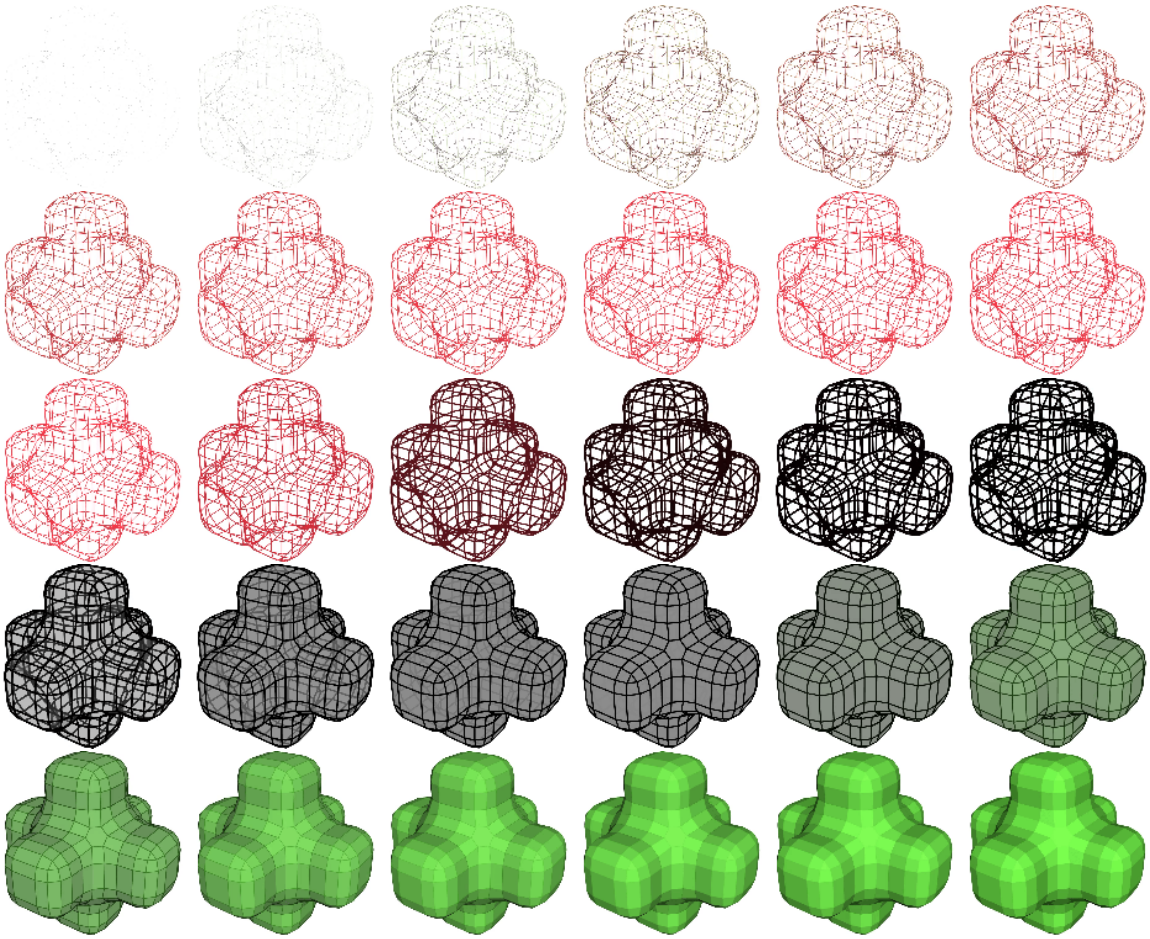
Obrázek 3.52: Diagram hlavních změn po implementaci podpory animací. Zeleně jsou vyznačeny nově přidané části. Původně byly při generování hlavičky nalezeny všechny použité materiály v převáděných objektech a instance vlastností SVG každého materiálu Blenderu byly poté převedeny na třídy stylů jazyka CSS a zapisovány do souboru. Animace fungují podobně, instance vlastností animace ExportSVGAnimationProperties každého materiálu jsou převáděny na definice klíčových snímků CSS. Oproti dřívějšku se však tento převod zanořuje, neboť vlastnosti animace obsahují seznam klíčových snímků (instance ExportSVGKeyframeProperties) a pro každý materiál je tedy převáděn postupně každý jeho klíčový snímek.

Výsledky

V rámci rozšíření byla úspěšně implementována funkce animování výstupních obrázků navázaná na dříve implementovaný systém materiálů. Uživatel může nově vytvářet animované obrázky vektorové grafiky, na kterých mohou jednotlivé prvky měnit své pozice, tvary a vizuální vlastnosti v závislosti na čase. Animace ovšem využívají prvky jazyka CSS, lze je tedy

zobrazit pouze v takových prostředích, které tyto typy animací podporují, tedy například v běžných webových prohlížečích, nikoliv však ve statických 2D editorech.

Kvůli samotné povaze animací nelze jejich výsledky snadno prezentovat v rámci statického textu. Na následujících obrázcích 3.53 lze vidět zachycené alespoň jednotlivé momenty při zobrazení souboru v prohlížeči v různých časových okamžicích.



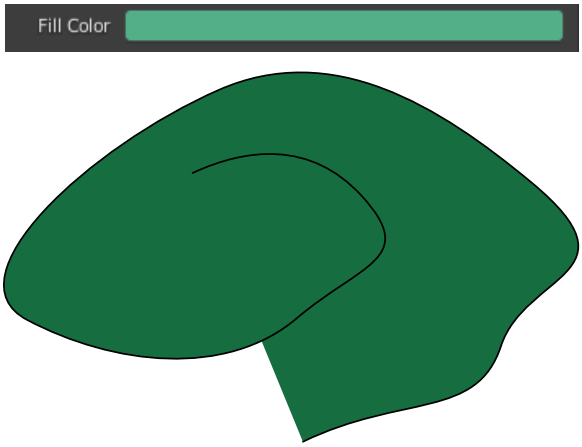
Obrázek 3.53: Ukázka výsledných animací. Lze vidět sekvenci 30 obrázků, které s rozestupem zhruba 0,3 sekundy vyobrazují 10 sekundovou animační sekvenci při zobrazení výsledného souboru ve webovém prohlížeči. Pro objekt v Blenderu byla definována animace CSS pomocí několika klíčových snímků: průhledné tahy i výplň, úzké červené tahy a průhledná výplň, široké černé tahy a průhledná výplň, široké černé tahy a šedá výplň. Sekvence neměla definován konečný klíčový snímek, tudíž se na konci cyklu vzhled objektu vrací do původního stavu neovlivněného animacemi, jímž byly neviditelné tahy a zelená výplň ovlivněná osvětlením. Tvorba takového objektu a definice této sekvence v Blenderu zabere se základní orientací v menu pluginu méně než 3 minuty. Po její definici lze díky funkcionalitě propojování animací materiálů aplikovat takovou sekvenci na libovolný materiál pár kliknutími.

3.14 Korekce barev a další úpravy

Toto rozšíření je souhrnem menších úprav a optimalizací týkajících se především kódu, které nelze zařadit k jiným rozšířením. Protože se z větší části jedná o čistě implementační detaily, nebude pro stručnost většina změn v této podkapitole podrobněji popisována, neboť nemá viditelný dopad na výstup či práci s pluginem. Výjimkou je však nejvýznamnější oprava týkající se korekce barev, která má zásadní dopad na výslednou kvalitu obrazu.

Korekce barev

Při implementaci nových rozšíření byla nalezena chyba v původním pluginu týkající se převodu barev specifikovaných v Blenderu na barvy zapisované do souboru SVG. Důsledkem chyby byly odlišné barvy ve výsledném obrazu oproti původnímu nastavení v Blenderu, jak lze vidět na následujícím obrázku 3.54.

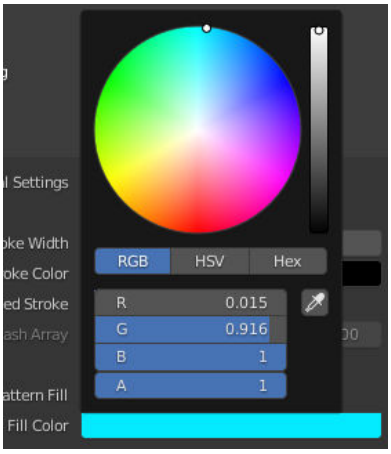


Obrázek 3.54: Příklad špatného převodu barev v původním pluginu. Nahoře lze vidět zvolenou barvu výplně (hodnoty RGB: 82, 175, 136), dole poté vykreslený výsledný soubor s křivkou, která by při správném fungování měla mít totožnou barvu výplně (má však hodnoty RGB: 22, 109, 63).

Problém vycházel z odlišných barevných prostorů využívaných v Blenderu a ve formátu SVG. Barevný prostor definuje množinu barev, se kterými lze pracovat. Tento prostor může být lineární, což znamená, že intenzita barev roste lineárně s jejich numerickými hodnotami. Barvy mohou být sčítány či násobeny a výsledná barva bude korektní. Pokud je prostor nelineární, znamená to, že grafem závislosti mezi numerickými hodnotami definujícími barvu a její skutečnou intenzitou je křivka, nikoliv přímka [15].

Formát SVG využívá standard sRGB, který je nelineární [16, sekce 12.1], Blender naopak interně používá lineární barevný prostor [5, sekce „Rendering > Color Management“],

Chyba v původním pluginu byla zapříčiněna neznalostí způsobu definice barev v Blenderu a zanedbáním barevných prostorů. V Blenderu jsou barvy ukládány jako vlastnost typu `FloatVectorProperty`, která může nabývat většího množství různých podtypů. V původním pluginu byl naivně zvolen podtyp `COLOR`, který umožňuje uživateli na vykreslený prvek tohoto podtypu v UI kliknout a definovat barvu standardním výběrem jako v běžných grafických nástrojích. Příklad lze vidět na obrázku 3.55. Při převodu byly tyto barvy pouze čteny a jejich hodnoty převedeny z intervalu 0-1 na interval 0-255 jednoduchým násobením.



Obrázek 3.55: Vykreslení vlastnosti podtypu `COLOR` v prostředí Blenderu po kliknutí na odpovídající prvek UI

Podtyp `COLOR` ovšem slouží k reprezentaci barvy lineárního prostoru. Při jejím převodu a použití ve formátu SVG, který využívá nelineární sRGB, dojde k chybě a výsledná barva neodpovídá původní. Jak bylo zjištěno v rámci této opravy, v Blenderu existuje ještě jiný podtyp pro definici barev `COLOR_GAMMA`, který reprezentuje barvy nelineárního prostoru a je vhodnější pro snadný převod na sRGB.

Chybu tedy bylo možné napravit dvěma způsoby. Prvním bylo přenastavení podtypů všech používaných vlastností pro ukládání barev v pluginu na podtyp `COLOR_GAMMA`. Druhým byla změna výpočtu při převodu a zápisu barev do souboru SVG. Před vynásobením hodnot barev x (z intervalu 0-1) hodnotou 255 je v takovém případě nutno ještě provést korekci na novou hodnotu x_c dle následujících vzorců 3.1 a 3.2 [9].

$$x_c = 12.92 \cdot x \qquad pro\ x < 0.0031308 \tag{3.1}$$

$$x_c = 1.055 \cdot |x|^{\frac{1}{2.4}} - 0.055 \qquad pro\ x \geq 0.0031308 \tag{3.2}$$

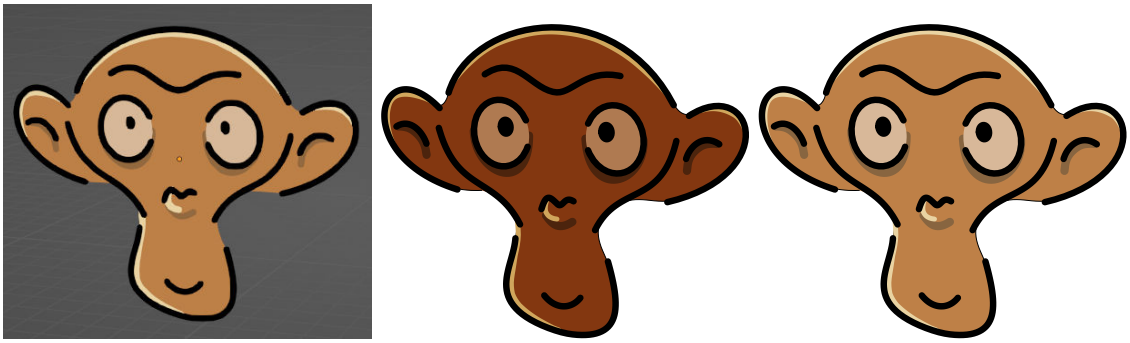
Pro snížení náročnosti výpočtu byl zvolen první způsob, neboť přepočet barev je v rámci převodu prováděn velmi často a počet volání roste s počtem převáděných prvků. Je proto vhodnější, aby funkce byla co nejjednodušší.

Výsledek před korekcí a po korekci lze vidět na obrázku 3.56.

Další úpravy

Jak už bylo výše zmíněno, zbylé úpravy v rámci tohoto rozšíření jsou příliš implementačně zaměřené, nebudou tedy pro stručnost všechny detailně rozebírány. Jako příklady úprav lze však uvést třeba:

- optimalizace polí při řazení využitím speciálního typu jazyka Python `Deque` (Double Ended Queue, oboustranná fronta), který snižuje časovou složitost odebírání prvků globálního řazení z obou konců seznamu na O(1) [13, sekce „deque“],
- optimalizace zápisů do souboru (na disk) tvorbou jednoho dlouhého řetězce, který je poté celý zapsán do souboru najednou, namísto soustavného zapisování každého prvku,



Obrázek 3.56: Srovnání před korekcí a po korekci barev. Nalevo lze vidět původní scénu v Blenderu s objektem tvořeným křivkami, jehož barvy odpovídají nastaveným barvám pro převod. Uprostřed lze vidět převod bez korekce, napravo převod po korekci.

- vylepšování kvality kódu zjednodušením předávání parametrů při volání jednotlivých metod převodu, redukce počtu parametrů téměř všech metod na 2-5 (předtím některé metody po postupném rozšiřování pluginu měly dokonce až 10 parametrů),
- vylepšování kvality kódu zaváděním globálních proměnných pro konstanty a jiné refaktorizace.

Kapitola 4

Zhodnocení výsledné implementace

Následující kapitola je souhrnem výsledné implementace. Jedná se však o spíše kratší zhodnocení, neboť výsledky rozšíření byly průběžně demonstrovány v každé předchozí podkapitole.

Jako první příklady praktických výsledků lze napřed zmínit některé obrázky použité v této práci, které byly částečně nebo zcela tvořeny s využitím implementovaného pluginu a jeho rozšíření, například obrázky 3.7, 3.12, 3.15 nebo 3.19.

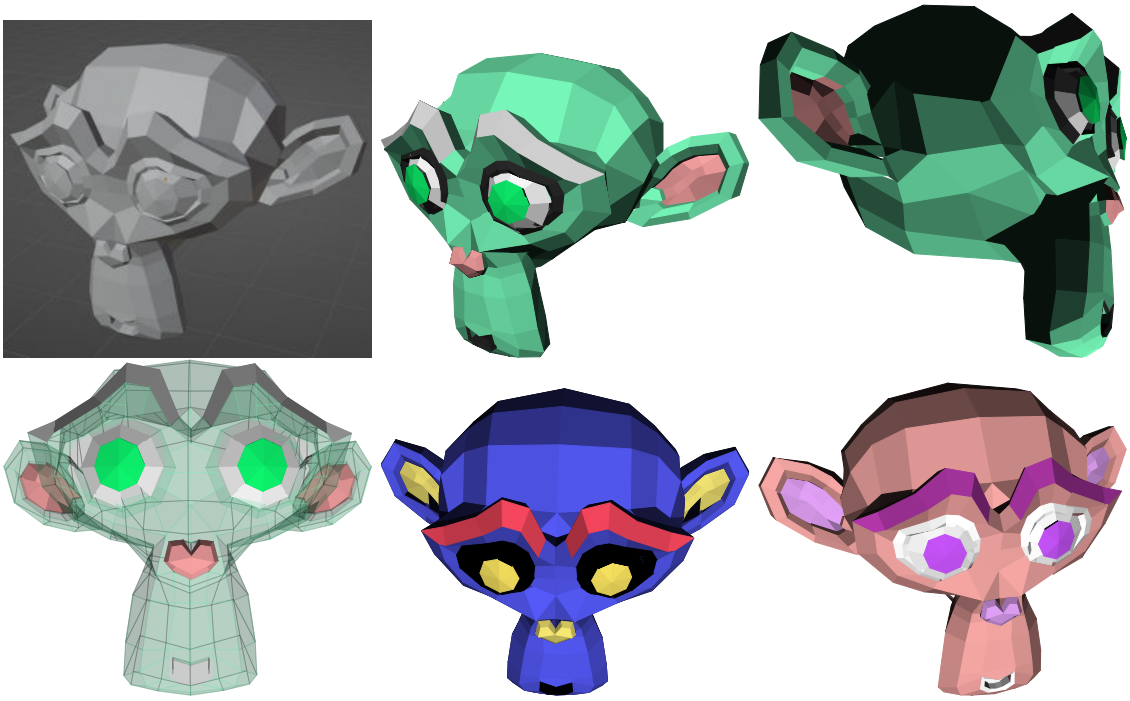
Výsledky dosažitelné jednotlivými rozšířeními také samozřejmě byly generovány pomocí samotného pluginu. Kvůli velkému množství možností a nastavení, které už plugin nabízí, není jednoduše možné prezentovat v rozumném rozsahu všechny typy výsledků, kterých s ním lze dosáhnout. Obzvláště pokud by se začaly jednotlivé části a rozšíření mezi sebou kombinovat, nemluvě o prakticky nekonečném počtu možných modelů či typů objektů v Blenderu, na kterých by funkce pluginu mohly být demonstrovány.

Zbytek této kapitoly je tedy věnován pouze pár dalším ukázkám použití pluginu na složitějších modelech 4.1, 4.2, 4.3 a 4.4, zaměřených převážně na využívání materiálového systému. Případná časová měření v popiscích probíhala na sestavě s CPU AMD Ryzen 5 3400G a GPU NVIDIA GeForce GTX 1660 SUPER 6GB. Zápis polygonů probíhal na pevný disk s rychlostí 7200 otáček za minutu. Při testování byla použita verze Blenderu 3.2.0 na operačním systému Windows 11 64-bit.

Na úplný závěr kapitoly lze zmínit některé nevizuální výsledky, konkrétně metriky délky kódu. Ty se oproti původnímu pluginu změnily následovně:

- Počet řádků kódu (včetně prázdných řádků): nárůst z 2552 na zhruba 7426
- Počet metod: nárůst z 63 na zhruba 150
- Počet tříd: nárůst ze 17 na 43

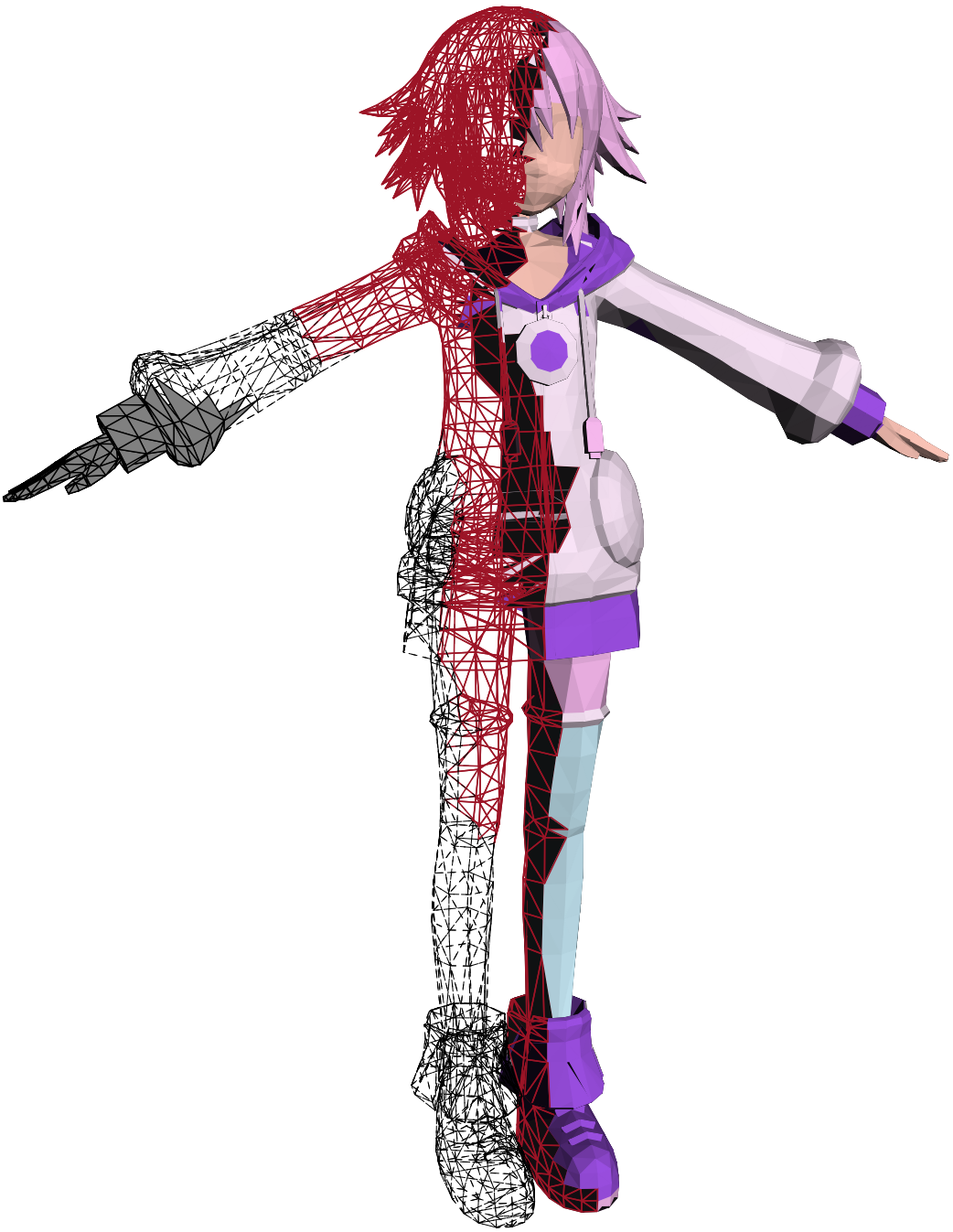
Dále byla přepsána i dokumentace pro ostatní uživatele z původního 7stránkového návodu k instalaci a základnímu použití na 14stránkovou uživatelskou příručku rozdělenou do dvou sekcí: stručný úvod a podrobný popis všech nastavení. Příručka byla taktéž doplněna o dodatečné vysvětlivky starších částí a o ilustrační obrázky.



Obrázek 4.1: Ukázka využití materiálů pro stylizaci převáděného modelu. Vlevo nahoře lze vidět snímek obrazovky s původním modelem v Blenderu, na zbylých obrázcích poté vykreslené výsledné soubory s modelem převedeným podle různých stylů materiálů, úhlů pohledu a směrů osvětlení. Každý obrázek obsahuje zhruba 400 polygonů a jeho převod trval méně než 0,05 sekund. Výsledné soubory mají vždy velikost přibližně 56 kB.

Obrázek 4.2: Ukázka převodu složitějšího modelu domu s aplikací materiálů pro obarvení stěn. Vlevo lze vidět snímek obrazovky s původním modelem v Blenderu, napravo vykreslené výsledné soubory využívající materiály pro obarvení jednotlivých polygonů. Oba výstupní soubory obsahují zhruba 1000 polygonů, mají velikost přibližně 200 kB a jejich převody trvaly přibližně 0,13 sekund.

Obrázek 4.3: Převáděný model postavy z počítačové hry, který obsahuje celkem 7982 polygonů. Tento stejný model byl použit pro demonstraci výsledků v původní bakalářské práci [10], ve které byl pouze převeden na šedé vystínované polygony. Výsledek převodu v této práci lze vidět na následujícím obrázku 4.4.



Obrázek 4.4: Vykreslení výstupního souboru převedeného modelu z předchozího obrázku. Využitím selekce lasem uvnitř Blenderu byly snadno tahem myši z modelu vykrojeny skupiny stěn, na které byly aplikovány materiály pro vybarvení a pro různé efekty. Lze tak rychle a efektivně vytvořit vektorový obrázek reprezentující nahlédnutí dovnitř modelu. Zároveň nelze opomenout ani fakt, že všechny tyto efekty je možno snadno animovat. Lze tak například vytvořit normální obrázek této postavy, který po několika sekundách plynule přejde do výše viditelného stavu a odkryje tak strukturu modelu v rámci krátké animace. Celý obrázek má 7982 polygonů a velikost 1015 kB, jeho převod trval 0,93 sekund.

Kapitola 5

Závěr

Cílem práce bylo vylepšit stávající plugin pro nástroj Blender, který doposud umožňoval pouze jednoduché převádění modelů na obrázky vektorové grafiky, o další dodatečné funkce jako například převody dalších datových typů či rozšíření možností úpravy vzhledu výsledných tvarů.

V rámci práce byly splněny postupně všechny body zadání. Nejdříve byly nastudovány základy práce s některými částmi Blenderu, poté bylo detailněji prostudováno téma skriptování v Blender Python API. Následně byla navržena jednotlivá vylepšení pluginu, která byla dále implementována. Jejich použití bylo v každé podkapitole též demonstrováno na příslušných scénách a typech dat s různými kombinacemi nastavení. Nová verze pluginu byla zdokumentována jak po stránce kódu pro další možný vývoj, tak také ve formě rozsáhlejší uživatelské příručky pro praktické použití. Společně s touto příručkou byl plugin zveřejněn pro ostatní uživatele[1]. V rámci posledního bodu zadání bylo také vytvořeno a zveřejněno krátké demonstrační video[2].

Nejdříve bylo navrženo a implementováno 8 prvotně zamýšlených vylepšení. Těmito rozšířeními jsou: převod křivek, převod textu, hloubkové řazení různých typů, převod kolekcí, podpora kamerových objektů, podpora vzorkových výplní, přepracování uživatelského rozhraní a materiálový systém. Po implementaci původních 8 vylepšení byla navržena a implementována 4 další: podpora evaluace, převod objektů typu Grease Pencil, převod anotací a podpora animací. Na závěr ještě jako menší vylepšení pluginu byly opraveny některé původní chyby a zlepšena kvalita kódu.

Výsledný plugin byl dále zveřejněn a je možno jej volně stáhnout a nainstalovat do nástroje Blender. Původně byl plugin vylepšován a testován pro verzi Blenderu 3.2, jeho instalace, spuštění a základní funkce byly však také krátce otestovány na v současnosti nejnovější verzi 3.5.

Plugin obsahuje stejnou základní funkcionalitu jako jeho původní verze, tedy převod modelů scény v Blenderu na soubory vektorové grafiky ve formátu SVG s několika možnostmi modifikace tohoto převodu jako volba typu a pozice zdroje světla či jeho barvy. Tuto základní funkcionalitu dále rozšiřuje o všechna výše uvedená vylepšení. Díky těmto možnostem usnadňuje tvorbu vektorových obrázků především při vyobrazování prostorových scén a dále například umožňuje k těmto obrázkům snadno kreslit popisky nebo přidávat 2D animace.

[1] Publikovaný plugin online: https://github.com/Craszh/BlenderModelToSVG

[2] Video demonstrující použití pluginu dostupné online: https://youtu.be/khuWp8s0lBE

Rozšíření pluginu se však týkala spíše přidávání nových funkcí namísto vylepšování těch stávajících. Nejsou proto řešeny některé původní problémy předchozí práce jako například přesnější hloubkové řazení či vylepšování stínování. Tato problematika tedy stále zůstává otevřená a lze ji zařadit na seznam dalších možných vylepšení pluginu. Protože je navíc plugin implementován v prostředí Blenderu, které samo o sobě poskytuje mnoho funkcí a je velmi versatilní, lze stále uvažovat o celé řadě dalších budoucích vylepšení jeho stávajících funkcí či rozšíření o funkce nové. Kvůli velkému množství dalších návrhů byl jejich seznam zařazen do samostatné přílohy A, která do budoucna může sloužit jako inspirace pro další vývoj některých funkcí.

Literatura

[1] Akimov, N. *User Interface Elements Alignment By Columns* [online]. 2020 [cit. 2023-03-13]. Dostupné z: https://b3d.interplanety.org/en/user-interface-elements-alignment-by-columns/.

[2] Beneš, B., Sochor, J., Felkel, P. a Žára, J. *Moderní počítačová grafika*. 2. vyd. Computer Press, 2004. ISBN 80-251-0454-0.

[3] Blender Foundation. *Blender Python API Documentation Version 3.2* [online]. Blender Foundation, 2022 [cit. 2023-03-06]. Dostupné z: https://docs.blender.org/api/3.2/.

[4] Blender Online Community. *Depsgraph – Blender Developer Wiki* [online]. 2021 [cit. 2023-03-14]. Dostupné z: https://wiki.blender.org/wiki/Source/Depsgraph.

[5] Blender Online Community. *Blender Reference Manual Version 3.2* [online]. Blender Foundation, 2022 [cit. 2023-03-06]. Dostupné z: https://docs.blender.org/manual/en/3.2/.

[6] Coyier, C. *The SVG 'path' Syntax: An Illustrated Guide* [online]. 2021 [cit. 2023-03-11]. Dostupné z: https://css-tricks.com/svg-path-syntax-illustrated-guide/.

[7] Gangl, D. *Using UiLists in Blender* [online]. 2021 [cit. 2023-03-10]. Dostupné z: https://sinestesia.co/blog/tutorials/using-uilists-in-blender/.

[8] Hughes, J. F., Dam, A. van, McGuire, M., Sklar, D. F., Foley, J. D. et al. *Computer Graphics: Principles and Practice*. 3. vyd. Addison-Wesley, 2013. ISBN 978-0-321-39952-6.

[9] International Color Consortium. *SRGB Color Space – Standard RGB Color Space* [online]. 1999 [cit. 2023-03-12]. Dostupné z: https://color.org/chardata/rgb/srgb.xalter.

[10] Kopáček, J. *Blender plugin pro převod modelů na vektorovou grafiku*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23537/.

[11] Lengyel, E. *Mathematics for 3D Game Programming and Computer Graphics*. 3. vyd. Course Technology Press, 2011. ISBN 978-1-4354-5886-4.

[12] MDN Web Docs Community. *Mozilla Developers Network Web Docs* [online]. 2023 [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web.

[13] Python Online Community. *Python Documentation – Collections – Container Datatypes* [online]. Python Software Foundation, 2023 [cit. 2023-03-14]. Dostupné z: https://docs.python.org/3/library/collections.html.

[14] Selin, E. *Blender Text Object: A Complete Guide* [online]. 2020 [cit. 2023-03-11]. Dostupné z: https://artisticrender.com/blender-text-object-a-complete-guide/.

[15] Sewell, S. *Gamma And Linear Space – What They Are And How They Differ* [online]. 2016 [cit. 2023-03-12]. Dostupné z: https://www.kinematicsoup.com/news/2016/6/15/gamma-and-linear-space-what-they-are-how-they-differ.

[16] W3C SVG Working Group. *Scalable Vector Graphics (SVG) 1.1 Specification (Second Edition)* [online]. W3C, 2011 [cit. 2023-03-09]. Dostupné z: https://www.w3.org/TR/SVG11/Overview.html.

[17] W3Schools.com. *W3Schools CSS Reference* [online]. 2023 [cit. 2023-03-10]. Dostupné z: https://www.w3schools.com/cssref/index.php.

Příloha A

Další možná rozšíření

V následující příloze je uveden seznam dalších možných oprav, vylepšení a rozšíření, na která v rámci této práce už nezbyl čas nebo prostor. Návrhy nejsou seřazeny dle významnosti, ale spíše podle logických celků.

- Vylepšení převodu křivek
  - Podpora více materiálů v rámci jedné křivky (například pro každý spline)
  - Podpora NURBS křivek
  - Implementace dalších vizuálních atributů SVG
- Vylepšení převodu textu
  - Podpora převodu textu na typ Grease Pencil
  - Podpora dalších vizuálních atributů SVG
  - Rozšíření převodu na <text> o více možností než pouze nastavení velikosti fontu
- Vylepšení řazení
  - Přesnější řazení jednotlivých typů, případně vhodnější aproximace hloubky než pouze výpočtem bounding boxu
- Vylepšení podpory kamer
  - Intuitivnější způsob selekce kamer
  - Implementace (optimalizace) ořezu objektů ležících mimo záběr kamery, případně implementace podpory nastavení v Blenderu umožňujícího ořezávání objektů blízko či daleko od kamery (nastavení Clip Start a End)
  - Podpora efektu nastavení hloubky ostrosti u kamerových objektů, například pomocí speciálních filtrů formátu SVG pro rozmazání nezaostřených prvků
  - Podpora integrace pohledu více kamer do jednoho souboru, například skládáním menších obrázků vedle sebe do jednoho velkého, nebo generováním speciálních tlačítek či záložek do souboru SVG, které po kliknutí skryjí obraz z pohledu současné kamery a vykreslí obraz z pohledu jiné odpovídající kamery
- Vylepšení podpory vzorkových výplní
  - Vhodnější a intuitivnější možnosti pro nakopírování řetězce se vzorkem

- Vylepšení uživatelského rozhraní a zpětné vazby
  - Implementace logu kamery/převodu, do kterého budou při konverzi zapisovány podstatné informace (chyby, výjimky, statistiky), které budou následně po ukončení převodu uživateli zobrazeny či vypsány
  - Implementace ukazatele průběhu převodu indikujícího, jaká část převodu už je hotová a jak dlouho zbývá do konce
  - Implementace dialogových oken (například pro potvrzení spuštění převodu)
  - Implementace sumarizace převodu zobrazujícího důležité informace týkající se konverze ještě před potvrzením jejího spuštění (například seznam převáděných objektů, kamer, typ osvětlení, odhad počtu výsledných prvků)
- Vylepšení animací
  - Vhodnější podpora transformací, například možnost vytvořit libovolně dlouhý seznam různých typů transformací s různými jednotkami a možností měnit jejich pořadí
  - Podpora animace dalších atributů SVG/CSS
  - Implementace speciálních módů převodu animace, například možnost generování animace spouštějící se pouze po najetí kurzoru či po kliknutí
  - Celkové přepracování systému animací, zrušení návaznosti animací na materiály (například možností definovat animace jakožto samostatné entity přiřaditelné k materiálům, umožnění tvorby určitých šablon animací snadno aplikovatelných na jednotlivé objekty/materiály)
  - S předchozím bodem související možnost aplikace více animací současně na jeden objekt/materiál (podporováno formátem SVG/CSS)
  - Podpora skutečných Blender animací namísto pouhých CSS animací, například převodem všech klíčových snímků scény Blenderu na samostatné obrázky uložené v jednom souboru, zviditelněním pouze jednoho z obrázků současně a možností přepínat mezi viditelnými obrázky například stiskem klávesy (možný problém s náročností rychlého vykreslování v prohlížeči při větším množství prvků na obrázku)
- Rozšíření o podporu převodu rastrových obrázků
  - Práce původně obsahovala další rozšíření týkající se podpory převodu rastrových obrázků umístěných do scény Blenderu. Nepodařilo se však toto rozšíření implementovat v takové podobě, která by byla uživatelský přívětivá a která by poskytovala výhodu oproti běžným editorům 2D grafiky. Rozšíření tak bylo z práce v počátečním stádiu vývoje pro stručnost smazáno. Hlavním problémem bylo určování přesných rozměrů obrázků na výsledném pohledu a způsob jeho pootočení v prostoru. Nestihlo se ovšem prozkoumat všechny možnosti, rozšíření tedy stále nemusí být nutně zavrženo.
- Podpora/využití dalších zajímavých prvků formátu SVG, například prvky filter či gradient

- Implementace různých speciálních efektů a módů převodu (například zvýraznění nebo číslování vrcholů a kontrolních bodů či efekt exploze polygonů ze staršího pluginu), případně další možnosti úprav a deformací výsledného obrazu

Na závěr lze zmínit některé abstraktnější návrhy na rozšíření z původní bakalářské práce, kam patří například zpřesnění řazení a řezání vzájemně zaklíněných prvků ve scéně, vylepšení osvětlení či přidání více speciálních módů stínování objektů. Teoreticky lze také uvažovat o vytvoření primitivního rozhraní pro programování pluginu, například umožnění uživateli vkládat vlastní skripty jazyka Python pro výpočet stínování či řazení objektů.