České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Kamil Hendrich**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Zásuvné moduly pro Google Sketchup**

Pokyny pro vypracování:

Prostudujte rozhraní pro vytváření zásuvných modulů v aplikaci Sketchup. Soustřeďte se na problematiku přidávání objektů do scény a vytváření vlastního náhledu na scénu. Navrhněte a implementujte dva vlastní moduly. První modul umožní interaktivní vložení předdefinovaných geometrických objektů do scény a bude podporovat specifikaci vybraných parametrů objektů (např. měřítka, barvy, textury). Druhý modul bude implementovat renderer založený na OpenGL, který zprostředkuje vlastní náhled na scénu současně se standardním zobrazením programu Sketchup. Tento modul rovněž umožní nahradit objekty vložené do scény složitější geometrickou reprezentací načtenou z externích souborů. Otestujte efektivitu implementace z hlediska časových a paměťových nároků na nejméně pěti vybraných modelech s různou složitostí.

Seznam odborné literatury:

[1] Scarpino, M. Automatic SketchUp: Creating 3-D Models in Ruby. (1st ed.). Eclipse Engineering LLC, , USA, 2010.
[2] Schreyer, A. C. Architectural Design with SketchUp - Component-Based Modeling, Plugins, Rendering, and Scripting (1st ed). Hoboken, NJ: J. Wiley & Sons, 2012.
[3] Ellis, P. G., Torcellini, P. A., Crawley, D. B. Energy design plugin: an EnergyPlus Plugin for SketchUp. Technical Report NREL/CP-550-43569. National Renewable Energy Laboratory, 2008.
[4] Sktechup SDK. http://www.sketchup.com/intl/en/developer/sdk_start.html

Vedoucí: Ing. Jiří Bittner, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

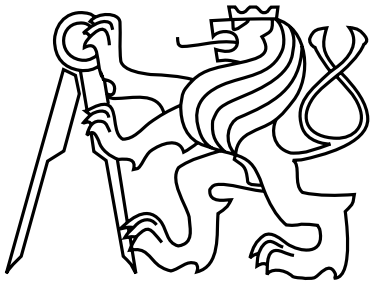
prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 2. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Bakalářská práce

Zásuvné moduly pro Google Sketchup

Kamil Hendrich

Vedoucí práce: Ing. Jiří Bittner, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Web a multimedia

21. května 2014

Poděkování

Rád bych poděkoval Ing. Jiřímu Bittnerovi, Ph.D., vedoucímu bakalářské práce, za jeho názory, postřehy, rady a pravidelné konzultace, a své rodině a přítelkyni za podporu při studiu.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Lánech dne 18. 5. 2014 .....

Abstract

SketchUp is a program for creating 3D models. Its greatest advantages are ease of use, own database of models (3D Warehouse) and link with Google Earth. The program allows creating, viewing and modifying of 3D models, working with textures and then publishing the created models to the public database. Additional features include the possibility to expand the program by many new functions using plugin modules (plugins). Plugins can be downloaded from the database (Extension warehouse), where are avaible many paid and unpaid extension, or you can create your own module according to your requirements.

Subject of the bachelor's thesis is an analysis of ways to add plugins to the SketchUp, which programming languages can be used to create plugins and a description of the development of plugins created according work assignment.

Abstrakt

SketchUp je program pro tvorbu 3D modelů. Mezi jeho největší výhody patří jednoduchost jeho ovládání, vlastní databáze modelů (3D Warehouse) a propojení s Google Earth. Program umožňuje tvorbu, prohlížení a modifikaci 3D modelů, práci s texturami a následné publikování vlastních modelů do veřejné databáze. Mezi jeho další přednosti patří možnost rozšířit program o mnoho nových funkcí pomocí zásuvných modulů (pluginů). Zásuvné moduly lze stáhnout z databáze (Extension warehouse), kde je k dispozici mnoho placených i neplacených rozšíření anebo si můžete vytvořit vlastní modul podle vašich požadavků.

Předmětem bakalářské práce je analýza toho jak lze zásuvné moduly do programu SketchUp přidávat, jakými způsoby a v jakých programovacích jazycích lze zásuvné moduly vytvářet a také popis vývoje zásuvných modulů vytvořených podle zadání práce.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Tato práce analyzuje způsoby, jak lze zásuvné moduly do programu SketchUp přidávat, jaké programovací jazyky lze použít pro tvorbu pluginů. Také popisuje základní struktury dat, které program používá k uchování informací o modelech. Dále popisuje podrobný postup při návrhu, tvorbě a testování pluginů vytvořených podle zadání práce.

Modelování je proces tvarování a vytváření 3D modelu, který může být reprezentován několika způsoby. Modely mohou být vytvořeny na počítači člověkem pomocí modelovacího nástroje, podle dat získaných měřicím přístrojem z reálného světa anebo na základě počítačové simulace. Nejobvyklejší reprezentace tvaru tělesa je hraniční reprezentace. Těleso je popsáno jako mnohostěn určený svými hranicemi (stěnami, hranami a vrcholy).

Existuje mnoho programů, které slouží pro tvorbu 3D modelů. Některé z nich jsou jednoduché a některé jsou určeny spíše pokročilým uživatelům. Práce se zabývá programem SketchUp, protože je velmi jednoduchý, a i přesto s ním lze dosáhnout působivých výsledků.

1.1 Specifikace cíle

Cílem je prozkoumat možnosti tvorby pluginů a analyzovat rozhraní poskytované programem SketchUp. Zjistit jaké programovací jazyky lze pro tvorbu pluginů použít a popsat základní datové typy používané pro ukládání modelů. Dále se pokusit najít dostupné pluginy pro SketchUp, které mají stejnou nebo podobnou funkčnost jako pluginy v zadání bakalářské práce. Prozkoumat jejich funkce a zjistit jejich nedostatky. Poté navrhnout vlastní pluginy, pokusit se je vytvořit a podrobně popsat postup jejich tvorby.

První plugin by měl zvládnout vložení jednoduchých objektů ve formátu OBJ do scény pomocí skriptu v jazyce Ruby. Na výběr by mělo být z několika modelů stromů a rostlin, které půjdou vkládat v několika různých velikostech.

Druhý plugin by měl vytvořit náhled na scénu v novém okně. Nejdříve jako výpis textových informací o modelu ve formě informací o jednotlivých objektech a pozicích jejich vrcholů. Později se pokusit o základní vykreslení scény s použitím OpenGL. Pro přístup k souborům ve formátu SKP by měl plugin využívat C API pokud to bude možné.

Třetí plugin by měl vkládat objekty ve formě billboardu (typicky jedna plocha otáčející se podle pozice kamery na které je textura znázorňující 3D model). Důvodem je, že SketchUp při

testování nedosáhl moc dobrých výsledků při vkládání složitějších 3D objektů do scény. Proto bylo zadání upřesněno. Billboard by měl být při vytvoření náhledu na scénu (renderingu) nahrazen detailním 3D modelem.

1.2 O programu SketchUp

SketchUp [11] je software pro tvorbu 3D modelů, původně vyvíjený od roku 1999 společností Last Software, poté společností Google a nyní společností Trimble. Mezi jeho největší výhody patří jednoduchost jeho ovládání (viz Obr. 1.2), vlastní databáze modelů (3D Warehouse (viz Obr. 1.3) [13]) a propojení s Google Earth [2].

Program umožňuje tvorbu, prohlížení a modifikaci 3D modelů, práci s texturami a následné publikování vlastních modelů do veřejné databáze 3D Warehouse. Výsledný model je možné umístit kdekoli na Zemi díky propojení s Google Earth. SketchUp je navržen pro architekty, návrháře, designery, strojaře, vývojáře her a díky jeho jednoduchosti ho zvládnou používat i uživatelé bez předchozích zkušeností s 3D editory.

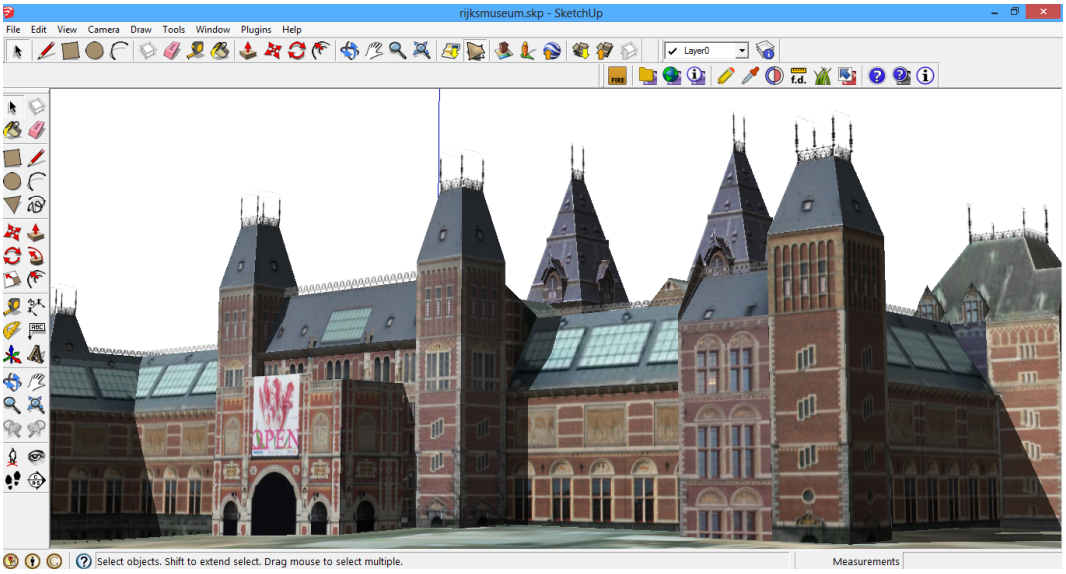
V současné době je k dispozici program SketchUp verze 14 (2014) ve variantách Make a Pro s podporou operačních systémů Windows a Mac OS X. Verze Make je zdarma, ale oproti verzi Pro má omezené možnosti modelovacích nástrojů. Verze Pro je placená, má více modelovacích nástrojů a umožňuje import a export více typů souborů než varianta Make.

Webová služba 3D Warehouse je rozsáhlá databáze modelů různých kategorií (budovy, rostliny, zvířata, lidé a další) kam mohou uživatelé přidávat své vlastní modely anebo stahovat hotové modely a použít je ke tvorbě vlastních projektů. Lze zde nalézt spoustu zajímavých modelů. Výhodou je možnost pokročilého vyhledávání modelů v databázi. Stažení modelů není zpoplatněno. Modely jsou zde uloženy většinou ve formátu SKP.

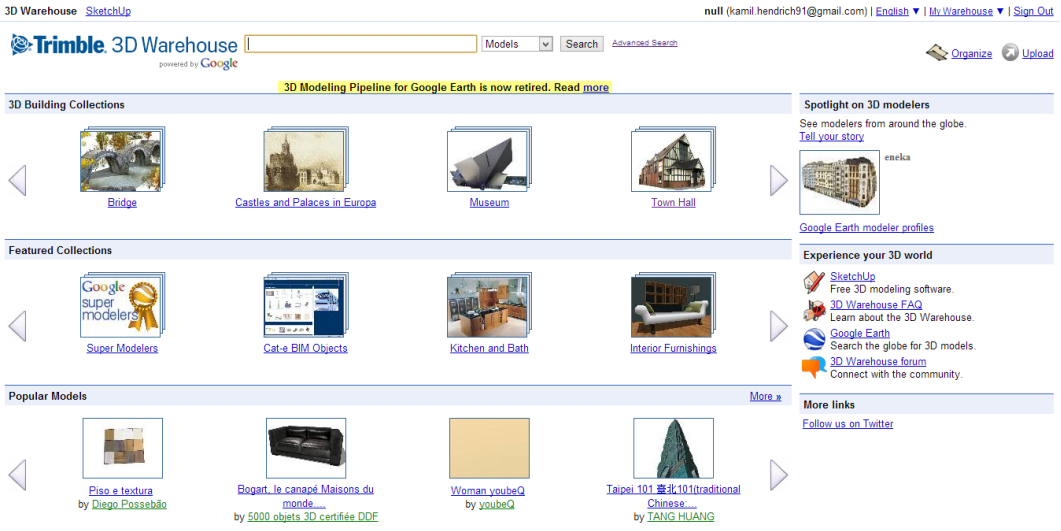
K dispozici je také služba Extension Warehouse [12], kde lze vybírat z několika stovek pluginů pro SketchUp z různých kategorií (interiérový design, renderování, import/export, animace, plánování, kreslení a další). Do databáze lze nahrávat vlastní pluginy. Některé pluginy ke stažení mohou být zpoplatněny, ale většinou je možné získat i časově omezenou verzi pluginu zdarma k vyzkoušení. Pluginy slouží k rozšiřování základních funkcí programu podle potřeb a požadavků jednotlivých uživatelů. Dokáží výrazně zvýšit rychlost modelování a ulehčit práci uživatelům. Na webu lze sehnat spoustu doplňků, které nejsou v databázi Extension Warehouse a přesto jsou velmi kvalitně zpracované.

Můj názor je, že SketchUp je velmi jednoduchý 3D editor vhodný pro začátečníky i pokročilé uživatele. Poskytuje všechny základní nástroje pro vytváření modelů a navíc má i spoustu dalších zajímavých funkcí. Největší podporu vývojářů a uživatelů měl v době, kdy byl vyvíjen společností Google. Současná dokumentace k C API na oficiálním webu je podle mě nedostatečná. Naštěstí na internetu existuje několik skupin a fór, které se zabývají problematikou tvorby pluginů pro SketchUp hlavně pomocí Ruby API a lze zde získat mnoho užitečných rad.

Obrázek 1.1: Logo programu SketchUp



Obrázek 1.2: Ukázka prostředí programu SketchUp



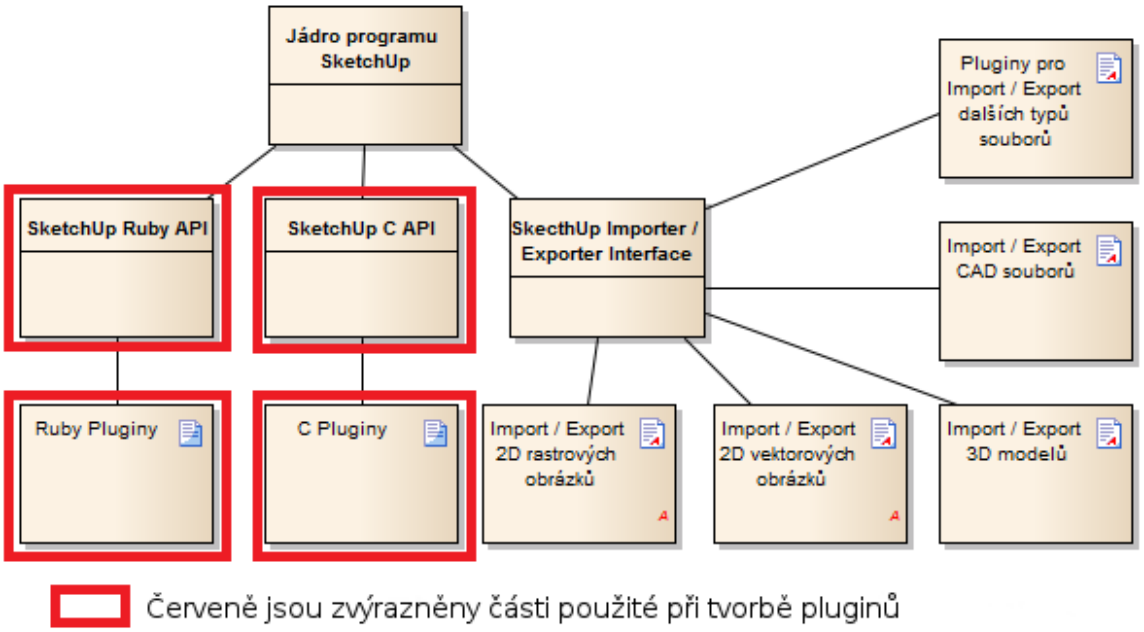
Obrázek 1.3: Ukázka webové stránky 3D Warehouse

Kapitola 2

Analýza

2.1 Popis rozhraní pro vytváření pluginů

SketchUp API (Application Programming Interface) je rozhraní pro komunikaci pluginů s programem SketchUp. Jsou k dispozici rozhraní pro programovací jazyk Ruby[5] (Ruby API) a jazyk C (C API) a dále rozhraní pro tvorbu Importérů a Exportérů (Importer/Exporter Interface) (viz Obr. 2.1). Protože zásahy do grafického rozhraní programu jsou velmi omezené, je k dispozici možnost tvorby grafického uživatelského rozhraní pomocí HTML stránky (Webdialog).



Obrázek 2.1: Rozhraní programu SketchUp

2.1.1 Základní datové typy

Důležité základní datové typy a struktury používané pro ukládání informací o modelech (viz Tabulka 2.1). Tyto struktury jsou společné pro Ruby API i C API.

Název struktury	**Popis**
Attribute dictionary	Struktura pro uložení dvojic dat (klíč, hodnota)
Camera	Objekt reprezentující kameru a uchovávající informace o ní
Component definition	Struktura obsahující informace o komponentě
Edge	Struktura uchovávající informace o hraně
Entities	Struktura obsahující kontejner se všemi objekty v modelu
Entity	Struktura reprezentující jeden objekt v modelu
Face	Struktura uchovávající informace o ploše (polygonu)
Material	Struktura uchovávající informace o materiálu(název,obrázek,velikost)
Model	Struktura pro uložení odkazu na model
Point	Struktura reprezentující bod v 3D prostoru (x,y,z)
Vertex	Struktura reprezentující bod a informaci o hraně které je součástí

Tabulka 2.1: Základní datové struktury

2.1.2 Ruby API

SketchUp Ruby API má velmi dobrou oficiální dokumentaci [10], která je rozdělena do kategorií a obsahuje popis všech struktur, funkcí a ukázky zdrojových kódů. Pomocí tohoto API můžete rozšířit a přizpůsobit program SketchUp, aby vyhovoval vašim potřebám. Slouží pro tvorbu pluginů a pokročilých nástrojů, které zvýší rychlost modelování a umožní vytvářet složité objekty mnohem jednodušeji. Pomocí skriptů lze automatizovat skoro cokoli, co lze udělat ve SketchUp ručně.

Základy SketchUp Ruby API

Příkaz `require 'sketchup.rb'` se používá k načtení souboru, který umožňuje použít funkce SketchUp Ruby API.

Při vývoji a ladění pluginů se bude hodit příkaz `Sketchup.send_action "showRubyPanel:"`, který po startu programu SketchUp spustí Ruby konzoly do které lze posílat kontrolní výpisy a ve které se zobrazují chyby, které nastaly při běhu programu (pluginu).

Pro přístup k modelu otevřenému v programu se používá příkaz `Sketchup.active_model`.

Všechny objekty v modelu jsou uloženy v kolekci objektů typu entity. Pro přístup k nim se používá příkaz `Sketchup.active_model.entities`. Třída entity reprezentuje všechny objekty, které můžou být v modelu. Všechny metody třídy entity jsou k dispozici i ve všech podtřídách (podtypech) této třídy.

Třída ComponentDefinition[6] se používá pro definici obsahu komponent aplikace SketchUp. Komponenty jsou kolekce entit, které mohou být opětovně použity v celém modelu. Úpravy původní definice se projevují na všech kopiích komponenty. Pro přístup k definicím komponent aktuálního modelu se používá příkaz `Sketchup.active_model.definitions`.

Třída ComponentInstance reprezentuje jednotlivé instance komponent vzniklé z definice komponent v rámci modelu. Proto, třída ComponentInstance obsahuje odkaz na odpovídající ComponentDefinition objekt.

Třída Material představuje definici materiálu (název, textura, barva, velikost a další paramtry), který může být použit k objektu typu Drawingelements (základní třída pro všechny objekty v modelu, které lze zobrazit). Materiál je nejčastěji aplikován na plochy (faces). Pro získání všech materiálů použitých v modelu slouží příkaz `Sketchup.active_model.materials`.

Třída UI obsahuje metody k tvorbě jednoduchých prvků uživatelského rozhraní. Například příkaz `UI.menu("PlugIns").add_item("Menu item")` přidá položku Menu item do záložky Plugins.

WebDialog je třída, která umožňuje vytvářet dialogová okna pomocí jazyka HTML a JavaScriptu. Dialogová okna komunikují s Ruby skriptem pomocí JavaScript funkcí. Slouží k vytváření složitých uživatelských rozhraní uvnitř programu SketchUp. Pro vytvoření webdialogového okna složí příkaz `UI::WebDialog.new()`.

Další zajímavou funkcí, kterou poskytuje Ruby API je možnost vytvářet pozorovatele (Observery), kteří reagují na akce, které dělá uživatel při práci s programem SketchUp (např. změna pozice kamery, vložení objektu, vybrání modelovacího nástroje). Na tyto akce pozorovatel reaguje vykonáním události. Pozorovatelé se vytváří založením třídy, která implementuje rozhraní typu Observer (např. ViewObserver, ModelObserver, ToolsObserver) a obsahuje požadované metody. Příslušný pozorovatel se musí přiřadit k určitému modelu při každém načtení modelu nebo vytvoření nového modelu.

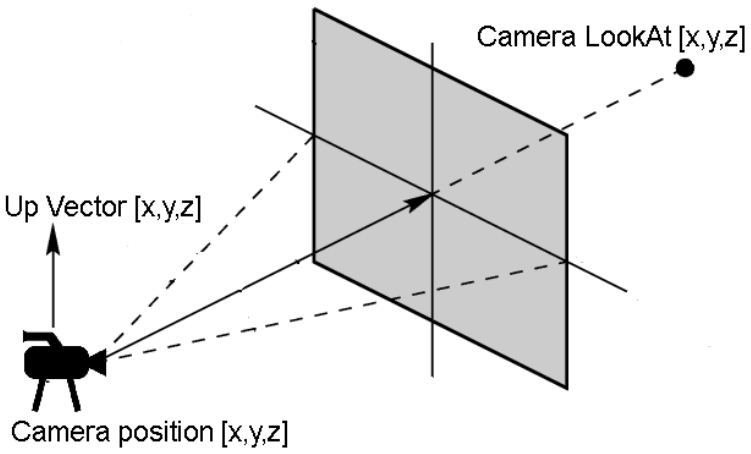
Základní operace s kamerou

Při práci s kamerou je důležité znát 3 základní hodnoty (viz Obr. 2.2). Aktuální pozici kamery (Camera position, Eye), směr pohledu kamery (LookAt, Target) a natočení kamery (Up vector). Tyto hodnoty stačí k vytvoření vlastní kamery anebo k načtení aktuální kamery pomocí Ruby skriptu. Dále se bude hodit znát hodnotu šířky záběru kamery (Field of view). Šířka záběru je úhel ve stupních, který určuje šířku zorného pole kamery v horizontální rovině.

V Ruby API lze přistupovat k základním nastavením kamery pomocí uložení aktuální kamery do proměnné view = `Sketchup.active_model.active_view`. A následně přistupovat k jednotlivým hodnotám kamery pomocí view.camera.eye, view.camera.target a view.camera.up.

Ukázka Ruby skriptu pro vytvoření kamery:

```
1  eye = [1000,1000,1000] #position
2  target = [0,0,0] #direction
3  up = [0,0,1] #upvector
4  my_camera = Sketchup::Camera.new eye, target, up
5  view = Sketchup.active_model.active_view
6  view.camera = my_camera
```



Obrázek 2.2: Pozice, směr a natočení kamery

2.1.3 C API

SketchUp C API je rozhraní pro čtení dat z modelů anebo zápis dat do modelů SketchUp. Toto rozhraní umožňuje čtení a úpravu souborů SketchUp (*.skp) bez nutnosti mít spuštěný program SketchUp. Je možné vytvářet nové, číst nebo upravovat stávající soubory. Nevýhodou je, že toto API neumožňuje modifikovat model, který je právě otevřený v programu SketchUp. Proto je nutné aktuální model pomocí Ruby skriptu nejdříve uložit na pevný disk a pak k němu přistupovat pomocí C API.

Dokumentace k C API [9] je podle mého názoru nedostatečná, protože se jedná pouze o vygenerovaný výpis z hlavičkových souborů (*.h). Obsahuje jen velmi stručné popisy metod a datových typů. Neobsahuje žádné příklady použití jednotlivých metod, což je problém, protože většinou existuje několik metod s podobným názvem i popisem, které se liší jen velmi málo, ale nelze je mezi sebou zaměnit. Některé části jsou vygenerovány špatně, například chybějící zahrnutí (include) hlavičkových souborů potřebných pro správnou funkci metody.

C API je vývojáři pluginů velmi málo používané a vývojáři programu SketchUp velmi málo podporované. Existuje pouze velmi málo příkladů použití a velmi málo diskuzních fór kde lze najít pomoc při řešení problémů.

C API je stále ve vývoji a v dnešní době stále není na úrovni Ruby API. Zatím ještě neobsahuje všechny funkce a proto v něm nelze realizovat to samé jako v Ruby API.

Základy SketchUp C API

Prvním krokem pro zprovoznění C API je stažení SketchUp SDK (Software Development Kit = sada nástrojů umožňující vývoj softwaru pro určitou aplikaci) z oficiálních webových stránek programu SketchUp a připojení (include) knihoven do projektu v MS Visual studiu nebo jiném programu. SDK obsahuje knihovny C API, Importer/Exporter interface a dokumentaci.

Při práci se SketchUp C API v programu se musí API nejprve inicializovat příkazem SUInitialize(). Pak teprve lze používat funkce a přistupovat k souborům ve formátu SKP. Na konci pogramu musíme API ukončit příkazem SUTerminate().

Všechny proměnné z API bychom měli při vytvoření inicializovat pomocí konstanty `SU_INVALID`.

Některé funkce vrací návratovou hodnotu ve fromátu `SU_ERROR_typ chyby` podle které lze poznat, jestli nastala chyba nebo vše proběhlo v pořádku. Tabulka 2.2 ukazuje několik základních typů chyb, s kterými jsem se setkal při vývoji pluginů. Kontrola návratových hodnot je velmi důležitá při ladění pluginu i pro zachytávání možných chyb za běhu pluginu.

Typ chyby	**Význam chyby**
`SU_ERROR_NONE`	Proběhlo bez chyby
`SU_ERROR_NULL_POINTER_INPUT`	Ukazatel na požadovaný vstupní parametr byl NULL
`SU_ERROR_INVALID_INPUT`	Objekt vstupující do funkce nebyl správně vytvořen
`SU_ERROR_NULL_POINTER_OUTPUT`	Ukazatel na požadovaný výstup je NULL
`SU_ERROR_INVALID_OUTPUT`	Objekt do kterého měl být zapsán výstup nebyl vytvořen
`SU_ERROR_GENERIC`	Nastala nespecifikovaná chyba
`SU_ERROR_MODEL_INVALID`	Ukazatel na model je neplatný a nelze jej načíst
`SU_ERROR_UNKNOWN_EXCEPTION`	Došlo k neznámé chybě

Tabulka 2.2: Chybové stavy

Pro získání informací o kameře z modelu existuje funkce SUModelGetCamera(), která má vstupní parametr odkaz na soubor ve formátu SKP. Tato funkce neumožňuje získat informaci o kameře z aktuálně otevřeného modelu, ale pouze pozici kamery při posledním uložení souboru. Informace o kameře lze získat pomocí funkce SUCameraGetOrientation(), která z odkazu na kameru získá pozici, směr a up vektor.

Pro získání všech instancí komponent z modelu slouží funkce SUModelGetComponentDefinitions(). Objekty uvnitř komponenty jsou uloženy stejně, jako ostatní objekty které nejsou součástí komponent a to v kontejneru typu Entities. Každá komponenta má svůj kontejner typu Entities a zároveň existuje ještě jeden kontejner pro všechny ostatní objekty v modelu, které nejsou součástí žádné skupiny ani komponenty.

C API nemá žádné funkce, které by umožňovaly přímou spolupráci s OpenGL.

2.1.4 Importer / Exporter interface

Rozhraní slouží pro tvorbu vlastních importérů a exportérů pro formáty souborů, které SketchUp neumí. Vytvořený importer/exporter je přidán do hlavního menu programu do sekce File->Import / Export. Importéry a exportéry lze vytvářet pomocí programovacích jazyků Ruby nebo C.

SketchUp Make (neplacená verze programu) dokáže importovat objekty ve formátech: .skp, .3ds, .dem, .ddf, .dae, .kmz a obrázky ve formátech .jpg, .bmp, .png, .tif, .tga, .psd. Exportovat dokáže pouze formáty .dae a .kmz nebo obrázky ve formátu .jpg, .bmp, .tif, .png.

Pomocí importérů je možné program SketchUp rozšířit a umožnit čtení z dalších formátů souborů. A pomocí nových exportérů může program SketchUp exportovat další formátů souborů.

Popis tvorby importéru

Importéry (exportéry) se implementují v programovacím jazyce Ruby (lze použít i programovací jazyk C) vytvořením vlastní třídy, která bude implementovat interface Sketchup::Importer a bude obsahovat všechny povinné metody.

Povinné metody jsou `description` (popis importéru v menu), `file_extension` (určuje jakou musí mít importovaný soubor příponu, jakého musí být typu), `id` (identifikační text), `supports_options?` (určuje zda je aktivní tlačítko Options), `do_options`(akce vykonaná při stisku tlačítka Options) a poslední metoda je `load_file` (která otevře a zpracuje importovaný soubor).

Ukázka zdrojového kódu

Zdrojový kód v programovacím jazyce Ruby ukazuje jak vytvořit Importér, jaké jsou jeho povinné metody a jak ho přidat mezi Importéry programu SketchUp.

```
1 class TextImporter < Sketchup::Importer
2   def description
3     return "Text Importer (*.txt)"
4
5   def file_extension
6     return "txt"
7
8   def load_file(file_path, status)
9             lines = IO.readlines(file_path)
10            UI.messagebox(lines)
11        return 0
```

2.1.5 Tvorba GUI

GUI (Graphical User Interface) je rozhraní, které umožňuje ovládat program SketchUp pomocí grafických ovládacích prvků (menu, tlačítka, formuláře, ikony a další).

Standardně lze GUI programu měnit pomocí Ruby API. Zásahy do GUI v programu SketchUp jsou omezené a lze přidávat jen základní prvky (tlačítka, položky v menu,atd.). Proto je tu další možnost vytvářet dialogová okna pomocí Webdialogů. Webdialogy umožňují načíst HTML stránku, ve které může být například formulář pro zadání hodnot, které pak lze použít v pluginu.

K vytvoření stránky lze použít technologie HTML (HyperText Markup Language pro tvorbu webových stránek), CSS (Kaskádové styly pro úpravu vzhledu weových stránek), JavaScript (skriptovací jazyk) a JQuery (JavaSkriptová knihovna s doplňkovými funkcemi) [7].

Pro ty kdo nemají zkušenosti s tvorbou HTML stránek, a přesto potřebují pro svůj plugin vytvořit nějaké pokročilé GUI je k dispozici neoficiální rozšíření SKUI (GUI Framework pro SketchUp). Je to knihovna, která umožňuje vytvářet WebDialogové GUI pouze pomocí Ruby kódu, bez nutnosti jakékoliv znalosti HTML, CSS, JS. Slouží pro vytváření uživatelské rozhraní pro pluginy do aplikace SketchUp.

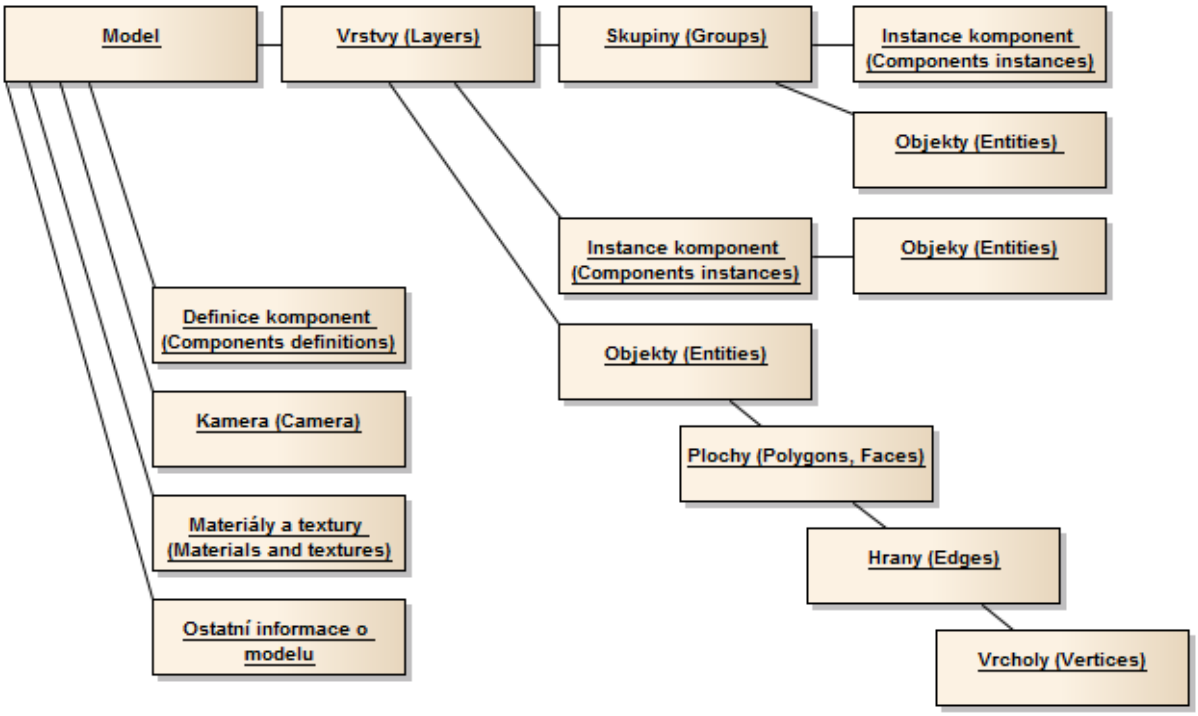
2.1.6 Logická struktura SKP souborů

Celý model (viz Obr. 2.3) je rozdělen na jednotlivé vrstvy (Layers). Vrstvy mohou obsahovat skupiny objektů (Groups), instance komponent (Components instances) anebo jednotlivé objekty (Entity). Skupiny objektů obvykle obsahují komponenty nebo jednotlivé objekty. Komponenty jsou složeny z komponent anebo jednotlivých objektů.

Objekty jsou uloženy jako kontejner (Entities), který reprezentuje geometrii objektů. Obsahuje informace o jednotlivých plochách (polygonech, Faces). Jednotlivé plochy jsou složeny z hran (Edges). Hrany jsou tvořeny jednotlivými vrcholy (Vertex, Point).

Model dále obsahuje informace o kameře (pozice, směr a up vektor), o použitých materiálech (texturách a jejich texturovacích souřadnicích), definice komponent (z definic se vytvářejí jednotlivé instance komponent) a další informace o modelu.

K jednotlivým částem modelu lze z programátorského hlediska přistupovat pouze přes Ruby API, C API anebo Import/Export Interface.



Obrázek 2.3: Logická struktura SKP souborů

2.1.7 Fyzická struktura SKP souborů

Formát SKP souborů je fyzické uložení dat v souborech typu SKP, neboli jakým způsobem jsou informace o modelu uloženy v těchto souborech. Vnitřní struktura souboru není viditelná, protože se jedná o binární soubor. Binární soubor je soubor skládající se z hlavičky (určuje jak data interpretovat) a vlastních binárních dat. V běžném textovém editoru lze soubor otevřít, ale je čitelná pouze hlavička SKP souboru za kterou následují binární data.

```
Ukázka hlavičky souboru ve formátu *.SKP
    SketchUp Model
       {8.0.4811}Version
       CAttributeNamed CBackgroundImage Camera
       CComponent CComponentBehavior CComponentDefinition
       CComponentInstance CConstructionGeometry
       CConstructionLine CConstructionPoint CCurve
       CDefinitionList CDib CDimension CDimensionLinear
       CDimensionRadial CDimensionStyle CDrawingElement
       CEdge CEdgeUse CEntity CFace CFaceTextureCoords
```

2.1.8 Struktura OBJ souborů

V pluginech používám volně dostupné 3D modely vegetace ve formátu OBJ a textury ve formátu JPG.

OBJ soubor má jednoduchý datový formát, který uchovává 3D geometrii pomocí pozice každého vrcholu, texturovacích souřadnic vrcholů, normál a ploch, které každý polygon definují jako seznam vrcholů. Vrcholy jsou obvykle uloženy v proti směru hodinových ručiček.

Základní typy dat používané uvnitř OBJ souborů jsou názvy objektů (o), vrcholy (v), texturovací souřadnice (vt), normály (vn), parametry (vp), materiály (mtlib) a plochy (f).

Komentáře v souboru jsou vytvořeny pomocí znaku#. Názvy objektů popisují jednotlivé části modelu. Vrchol jeautosketch reprezentován pomocí 3 (x, y, z) nebo 4 souřadnic (x, y, z, w). Texturovací souřadnice jsou uloženy pomocí dvou souřadnic (u, v). Normály jsou uloženy pomocí 3 souřadnic (x, y, z). Dále jsou v souboru ještě další parametry. Plochy jsou definovány pomocí seznamu obsahujícího indexy vrcholů, texturovacích souřadnic a normálových souřadnic. Materiály, které popisují vizuální parametry polygonů jsou uloženy v externích MTL souborech. V souboru OBJ může být odkazováno i na několik souborů s materiály. Soubory MTL můžou obsahovat jeden nebo více pojmenovaných definic materiálu. V OBJ souborech se na soubor s materiály odkazuje pomocí příkazu mtllib název_souboru.mtl a na jednotlivé materiály se odkazuje pomocí příkazu usemtl název_materiálu. Pokud jsou na model použity textury, tak odkaz na ně je uložen v souboru MTL.

```
Ukázka  souboru ve formátu *.OBJ
   # Blender v2.66 (sub 0) OBJ File: ''
   # www.blender.org
   mtllib flower1.mtl
   o Plant01
   v 27.287760 56.086643 74.935043
   v 25.220970 62.701042 72.925690
```

2.2 Analýza tvorby vlastních pluginů

2.2.1 Vložení objektu do scény

Pro vkládání objektů do scény budu vyvíjet dva různé pluginy, které si budou velmi podobné.

První plugin bude do modelu přidávat 3D modely ve formátu OBJ pomocí skriptu v programovacím jazyce Ruby. Bude možné vybírat z několika modelů stromů a rostlin v několika velikostech.

Druhý plugin bude do modelu vkládat billboardy [1]. Billboard (někdy nazýván Face-me component)je plocha s texturou modelu, která se otáčí podle pozice kamery, tak aby byla textura vždy viditelná. Důvodem vývoje tohoto pluginu jsou testy provedené na několika složitých 3D modelech ve formátu OBJ, při kterých se ukázalo, že vložení a vykreslení těchto modelů je velmi pomalé. Proto bude tento plugin vkládat pouze billboardy s obrázkem modelu (urychlení pluginu). Tyto billboardy by měl třetí plugin umět při vykreslení scény nahradit objekty ve formátu OBJ. Bude možné vybrat z několika modelů stromů a rostlin a z několika velikostí billboardu.

2.2.2 Náhled na scénu

Třetí plugin, který budu vyvíjet, by měl vytvořit náhled na scénu s využitím OpenGL. Plugin by měl využívat C API, pokud to bude možné.

Nejdříve by měl plugin umět vypsat všechny informace potřebné k vykreslení objektů na konzoli (seznam objektů, vrcholy a jejich pozice, použité materiály, pozice kamery a další informace o modelu). Poté by měl zvládnout vykreslit jednoduché objekty a billboardy vytvořené druhým pluginem by měl umět nahradit detailními 3D modely. Měl by zvládnout načítání materiálů a textur.

Důležité je se podrobně seznámit se strukturou souborů ve formátu SKP, s datovými typy používanými pro ukládání dat a s funkcemi C API, které umožňují přístup k datům v modelu.

2.3 Instalace pluginů

Existují 4 základní způsoby, jakými lze přidávat pluginy do programu SketchUp. Záleží na typu souboru se zdrojovým kódem, který je k dispozici (*.rb, *.rbz, *.exe) a na způsobu stažení pluginů (webový prohlížeč nebo integrovaný prohlížeč a instalátor v programu SketchUp).

2.3.1 SketchUp .rb soubory

Pro starší verze aplikace SketchUp (8.0.4811 a nižší) nebo soubory s příponou .rb , se pluginy instalují umístěním Ruby skriptu do příslušné složky:

- `C:\Program Files\Google\Google SketchUp 8\Plugins\` (pro OS Windows).

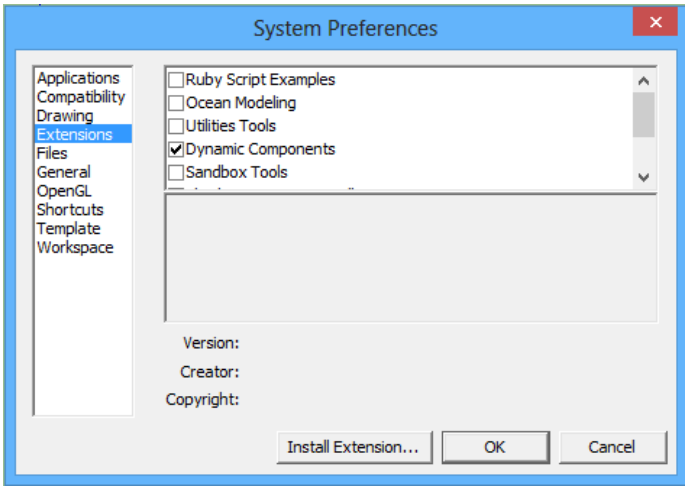
Pokud byla aplikace spuštěna při kopírování pluginů do složky "Plugins", musí se vypnout a znova zapnout, protože Pluginy se načítají při spuštění aplikace. Soubory umístěné do příslušné složky se automaticky načtou při každém zapnutí programu SketchUp.

2.3.2 SketchUp .rbz soubory

Google SketchUp 8 (8.0.16846 a vyšší) a SketchUp 2013 umožňují nainstalovat Ruby zip soubory (.rbz) přímo v aplikaci. Soubory .rb se přidávají stejně jako u starších verzí zkopírováním do složky Plugins.

Popis instalace Ruby skriptu ve formátu .rbz(pro OS Windows):

- Doporučuje se přihlásit se do systému jako správce.
- V menu programu SketchUp vyberte záložku Window > Preferences, zobrazí se dialogové okno System preferences (viz Obr. 2.4).
- Klikněte na záložku Extensions a v ní na tlačítko Instal extension.
- Otevře se dialogové okno, ve kterém vyhledejte Ruby soubor typu .rbz ve vašem počítači.
- Klikněte na tlačítko Otevřít. Plugin Ruby se objeví v seznamu Extensions.



Obrázek 2.4: Okno System preferences pro instalaci pluginů

2.3.3 Pluginy s vlastním instalátorem

Některé pluginy mají vlastní instalátor (.exe soubor), který doplněk nainstaluje do zvoleného umístění a pak přidá do programu SketchUp (pomocí Ruby skriptu). Většinou se jedná o placené pluginy. Některé z nich nejsou vytvořené v Ruby, ale používají Ruby skript pouze ke spuštění pluginu, který je napsán v jiném programovacím jazyce, nejčastěji v jazyce C.

2.3.4 Služba Extension Warehouse integrovaná v programu

V nových verzích programu (od SketchUp 2013) je k dispozici služba Extension Warehouse integrovaná přímo do prostředí SketchUp. Umožňuje prohlížení databáze, stažení a automatickou instalaci pluginů. Tuto funkci naleznete v záložce Window pod volbou Extension Warehouse. I nadále lze používat webovou verzi databáze, kterou naleznete na adrese extensions.sketchup.com, kde lze prohlížet a stahovat pluginy, ovšem bez možnosti automatické instalace.

2.4 Existující pluginy

Ve službě Extension Warehouse [12] jsem nalezl pluginy pro SketchUp, které se zabývají vkládáním objektů do scény a renderováním scény. Pluginy jsem hledal i mimo oficiální databázi na různých webových stránkách zabývajících se tvorbou pluginů pro SketchUp. Některé pluginy mohou být zpoplatněny a i přesto je většinou k dispozici 30 denní zkušební verze zdarma.

2.4.1 Plugin Treemaker

Plugin sloužící pro vkládání stromů do scény, který nabízí velmi mnoho možností nastavení (viz Obr. 2.5), spoustu modelů stromů a je velmi kvalitně zpracovaný. Poskytuje funkce velmi podobné pluginu, který je v zadání této práce. Jeho výsledný produkt je 2D objekt, který se otáčí podle pozice kamery ve scéně (billboard). Tento objekt je následně použit při renderingu scény. Ve svém pluginu bych se chtěl dopracovat k podobným možnostem nastavení.

2.4.2 Plugin CLF Scale and rotate multiple

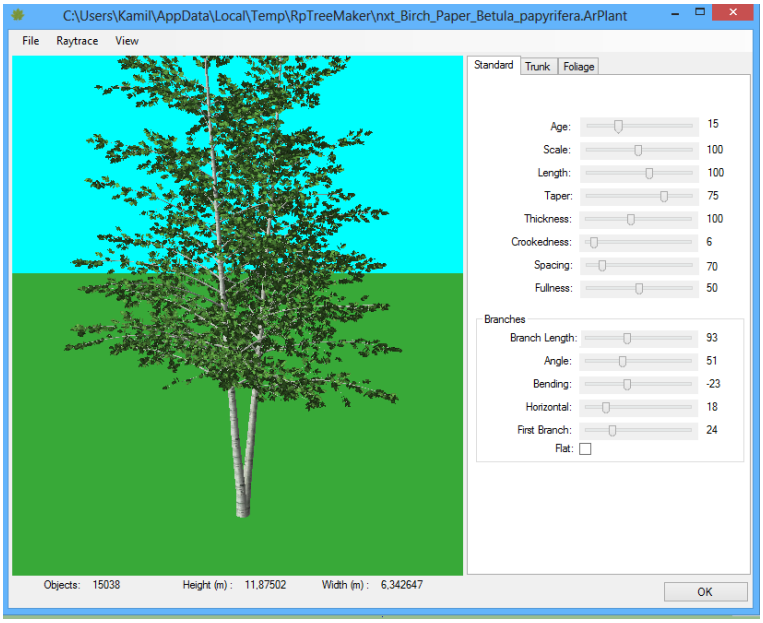
Plugin sloužící pro úpravu skupiny objektů. Umožňuje rotaci a škálování objektů a to náhodné s určením intervalů anebo uniformní s určením přesných hodnot. Ve svém pluginu bych chtěl použít stejný princip náhodného škálování a rotace s omezením na určitý interval pro vkládání skupiny stromů (např. vytvoření lesa).

2.4.3 Plugin CLF Extended views

Jednoduchý plugin pro změnu směru pohledu na scénu. Umožňuje změnu pozice kamery na 1 ze 7 předdefinovaných úhlů. Podobnou funkci bych chtěl použít v pluginu pro náhled na scénu, kde by šlo měnit pozici kamery na určité předdefinované úhly.

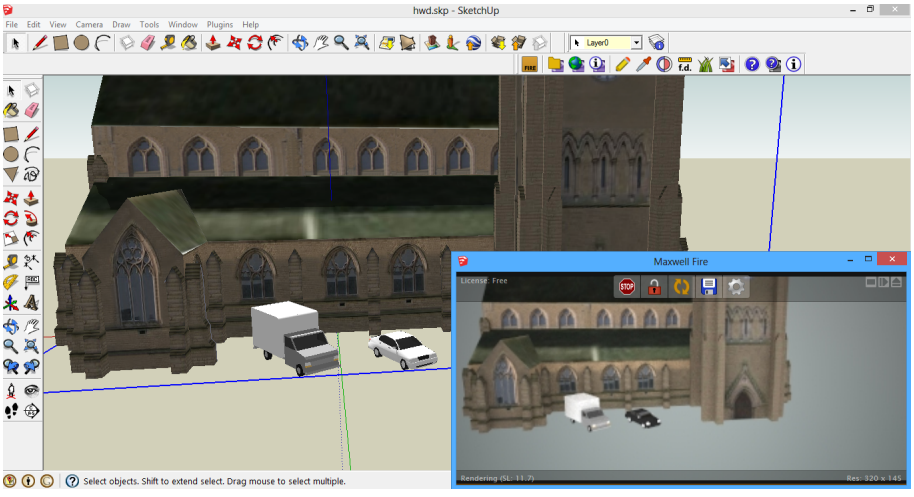
2.4.4 Plugin Maxwell for SketchUp

Velmi dobře fungující plugin, který dokáže vytvořit renderovaný náhled (viz Obr. 2.6) na scénu a následně uložit náhled jako obrázek. Při změně pozice kamery ve scéně se mění pozice kamery i v náhledu, ale při změně objektu ve scéně se objekt v náhledu nemění. Náhled se musí znova vykreslit pomocí tlačítka Maxwel Fire, které spustí export scény do externího



Obrázek 2.5: Treemaker plugin

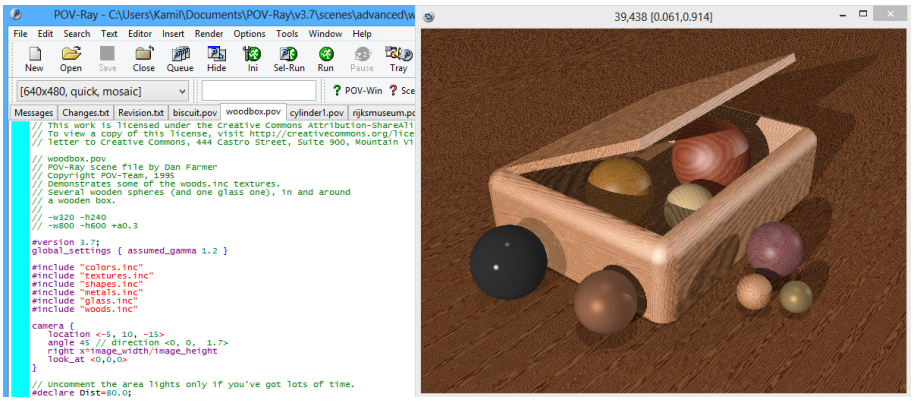
SKP souboru ze kterého pak plugin načítá data o modelu. Plugin je poměrně rychlý i pro složitější scény. Nemá problémy s vykreslováním textur a umí pokročilé stínování a výpočet globálního osvětlení.



Obrázek 2.6: Maxwell plugin

2.4.5 Ostatní programy

Vykreslením scény se moc pluginů pro program SketchUp nezabývá. Existují programy, které nejsou pluginy pro SketchUp, ale umožňují vykreslení souborů ve formátu SKP. Jedním z nich je například FluidRay RealTimeRenderer. Je to samostatná aplikace, která umí načítat modely ve formátu SKP a z nich vytvářet renderované obrázky. Poskytuje mnoho možností a nastavení. Jedná se o placenou aplikaci, kterou je možné zdarma vyzkoušet na 30 dní. Dalším programem, který se zabývá renderováním scény je například Pov-Ray (viz Obr. 2.7).



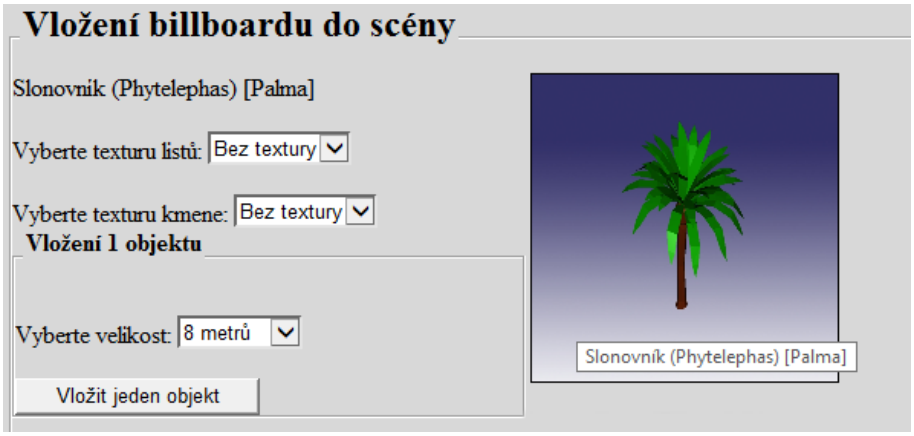
Obrázek 2.7: Program Pov-Ray

Kapitola 3

Realizace

3.1 Tvorba GUI

Ke tvorbě grafického uživatelského rozhraní jsem použil Ruby API pro přidání tlačítek do hlavního menu sloužících pro spouštění pluginů a částečně také Webdialog [4] pro ovládání pluginů. Webdialog je součást programu SketchUp, která umožňuje vytvářet grafické uživatelské rozhraní pomocí HTML stránky, která se načte do okna v programu (viz Obr. 3.1). Webová stránka může předávat jakékoli parametry do SketchUp (Ruby API) například pomocí JavaScript funkcí. Předané parametry jsou dále zpracovány pomocí Ruby skriptu.



Obrázek 3.1: Grafické uživatelské rozhraní vytvořené pomocí Webdialogu

3.1.1 Ukázka zdrojového kódu

Ukázka JavaScript funkce, která je volána při stisknutí tlačítka "Vložit"v pluginu Vložení objektu do scény. Tento kód zajišťuje předání parametrů zadaných uživatelem ve Webdialogovém okně do Ruby skriptu.

```
1 function call_ruby( callback_name, msg1, msg2 ) {
```

```
2           location = 'skp:' + callback_name + '@' + msg1 +
                'x' + msg2; }
```

Ukázka Ruby skriptu, která zajišťuje příjem zpráv od JavaScriptové funkce. Tuto zprávu rozdělí na jednotlivé části, které jsou oddělené znakem x.

```
1       wd.add_action_callback( "ruby_ac" ) do |dlg, msg|
2       msg1, msg2 = msg.split('x', 2)
```

Ukázka JavaScript funkce, která využívá pomocnou knihovnu JQuery. Tato funkce slouží pro změnu náhledu na objekt (obrázek, který ukazuje právě zvolený objekt) při změně hodnoty v HTML formuláři (změna vybraného modelu v SelectBoxu).

```
1 $(function(){
2           $("#obj").change(function(){
3           var selectedValue = $(this).find(":selected").val();
4           $("#modelImage").attr("src",path+imageArray[selectedValue]);
```

3.2 Vložení objektu do scény

Z několika webových stránek [14] [15] [16] jsem získal volně dostupné modely stromů a rostlin ve formátu OBJ a roztřídil je do několika kategorií (jehličnaté stromy, listnaté stromy, palmy, pokojové rostliny a ostatní rostliny). Tyto modely jsem upravil tak, aby měli všechny stejnou počáteční velikost.

Pomocí Ruby skriptu jsem přidal do menu programu do záložky "Plugins"volbu "Vložit objekt". Při stisku této položky v menu se zavolá kód, který zobrazí Webdialogové okno.

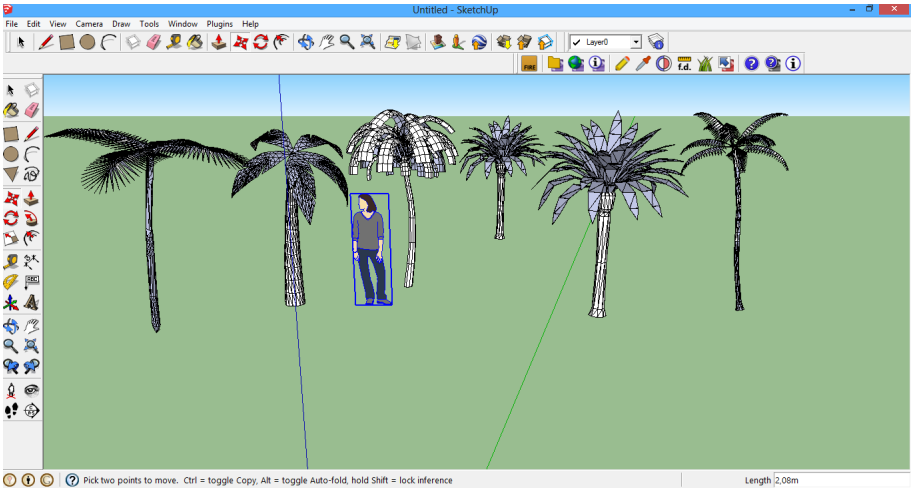
Pomocí Ruby skriptu a HTML stránky jsem vytvořil Webdialogové okno pro výběr modelu ze seznamu a volbu velikosti modelu. V tomto okně jsem pomocí JavaScriptu a JQuery funkcí vytvořil také náhled na aktuálně vybraný model, který se mění při změně hodnoty ve formuláři Webdialogu.

Informace zadané uživatelem do webdialogového okna se po stisku tlačítka Vložit odešlou pomocí JavaScriptové funkce do Ruby skriptu [4], kde se podle zvoleného modelu a velikosti zavolá funkce OBJ importéru. Tato funkce načte OBJ model ze souboru a podle zvolené velikosti ho zvětší, zmenší nebo ponechá v aktuální velikosti a vloží ho do scény (viz Obr. 3.2). Skript funguje, ale načítání modelů pomocí Ruby je poměrně pomalé a závisí na složitosti vkládaného objektu. V kapitole testování měřím rychlost vkládání v závislosti na počtu ploch v modelu.

3.2.1 Ukázka zdrojového kódu

Ukázka Ruby skriptu pro vložení položky do menu. Určuje do jaké záložky menu se vloží nový objekt. Existují určitá omezení, která určují, kam se dají vložit nové položky.

```
1 UI.menu("PlugIns").add_item("Vlozit_objekt") {
2           # Zde je kod, ktery se vykona pri stisku tlacitka v
                menu. }
```



Obrázek 3.2: Objekty vložené do scény

Ukázka Ruby skriptu pro vytvoření Webdialogového okna. Inicializace proměnných webdialogového okna. Nastavení HTML stránky, která se zobrazí v okně programu. Přidání zpětného volání, které přijímá parametry od JavaScript funkce. A vytvoření nového vlákna, které spustí webový prohlížeč a zobrazí HTML stránku. Tento kód je umístěn uvnitř bloku, který přidává tlačítko do menu.

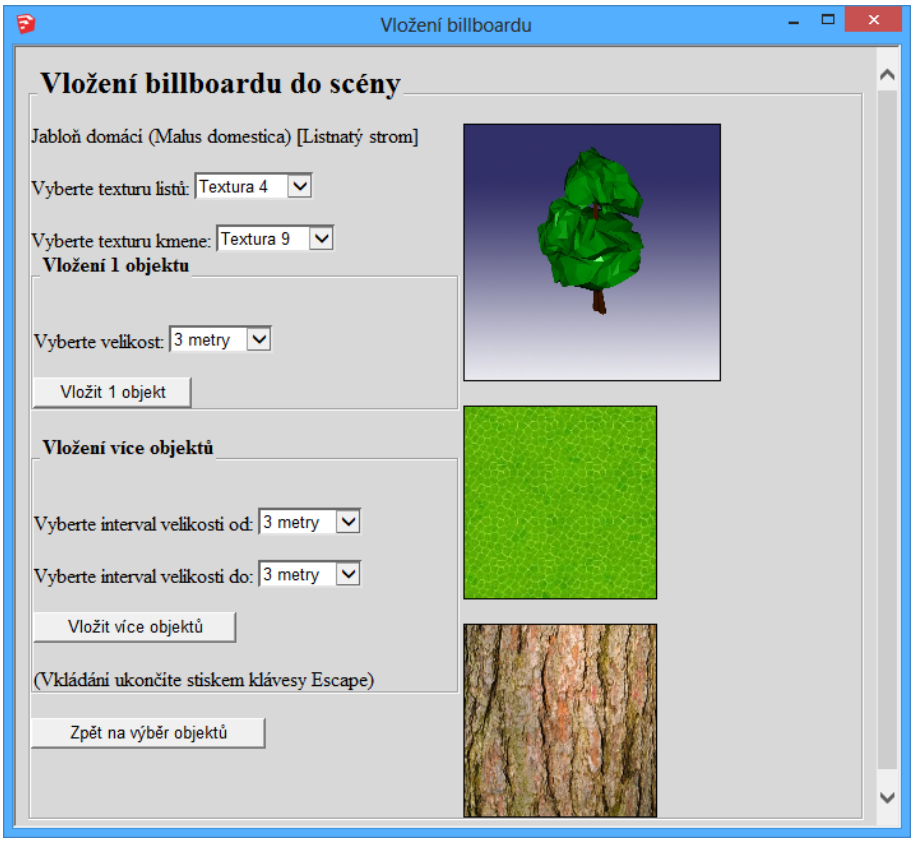
```
1 wd = UI::WebDialog.new( 'Box', true, 'chatterbox',450, 400,
    0, 0, true )
2         wd.add_action_callback( "ruby_ac" ) do |dlg, msg|
3         wd.close
4         wd.show_modal()
```

3.3 Vložení billboardu do scény

Billboard je plocha s texturou modelu, která se otáčí podle pozice kamery, tak aby byla textura vždy viditelná. Důvodem vývoje tohoto pluginu jsou testy provedené na několika složitých 3D modelech ve formátu OBJ, při kterých se ukázalo, že vložení těchto modelů pomocí Ruby skriptu je velmi pomalé. Vložené billboardy bude třetí plugin při vykreslení náhledu na scénu nahrazovat objekty ve formátu OBJ.

Výhodou vkládání billboardů je zrychlení oproti pluginu, který vkládal objekty ve formátu OBJ. Nevýhodou je, že SketchUp podporuje obrázky (textury) s průhledným pozadím, ale při zapnutém stínování je viditelný i stín neprůhledné části obrázku. Tento problém lze vyřešit přidáním masky, která kopíruje viditelnou část obrázku a podle ní se pak vytváří stín billboardu. Při tvorbě pluginu jsem masky pro textury nevytvářel, protože při vytváření náhledu na scénu budou billboardy nahrazeny OBJ modely. Další nevýhodou billboardů v programu SketchUp je že se otáčí pouze při změně pozice kamery do stran, ale vůbec nepočítá s náklonem billboardu při změně pozice kamery nahoru a dolů.

Pomocí webdialogu jsem vytvořil okno pro výběr objektu s náhledy jednotlivých objektů. Je možné vybrat z několika modelů stromů a rostlin. Po kliknutí na náhled se přejde na okno s nastavením parametrů (velikosti a textury)(viz Obr. 3.3). V tomto okně si uživatel také zvolí, jestli chce vložit jeden objekt podle nastavených parametrů nebo více objektů, které mění náhodně velikost v zadaném intervalu.

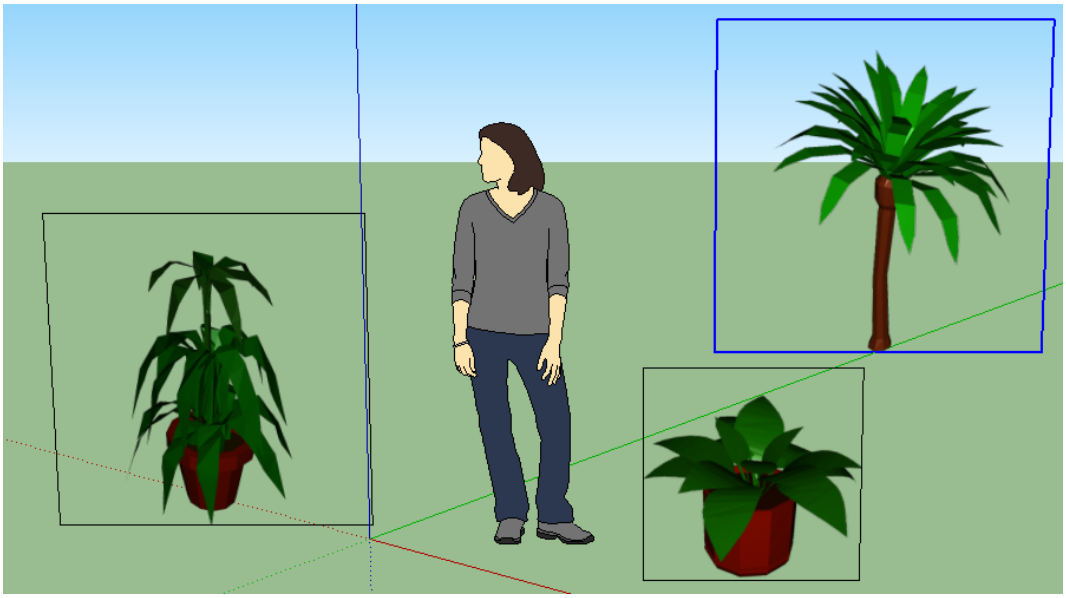


Obrázek 3.3: Ukázka pluginu pro tvorbu billboardů - nastavení parametrů

3.3.1 Tvorba billboardu

Billboardy v programu SketchUp je výhodné tvořit pomocí komponent, protože pouze komponenty mají vlastnost Always face camera, což znamená, že SketchUp sám vypočítá rotaci billboardu podle pozice kamery. Vlastnosti komponent jsou uloženy pomocí vlastnosti Behavior. Ta existuje pro každou komponentu.

Billboard je čtyřúhelník tvořen 4 vrcholy, na který je nanesena odpovídající textura. Ze čtverce je vytvořena komponenta, u které je nastavena vlastnost Always face camera na hodnotu true. Po vložení (viz Obr. 3.4) je billboard zvětšen nebo zmenšen na velikost zadanou v dialogovém okně. Pozici billboardu ve scéně určuje uživatel při jeho vložení. Billboard lze po vložení libovolně posouvat a škálovat pomocí standardních SketchUp nástrojů.



Obrázek 3.4: Ukázka pluginu - vložené billboardy

3.3.2 Component observer

Ruby API poskytuje možnost vytvářet pozorovatele (Observery), kteří reagují na akce, které dělá uživatel při práci s programem SketchUp. Pozorovatelé se vytváří založením třídy, která implementuje rozhraní typu Observer (např. ComponentInstanceObserver) a obsahuje požadované metody.

Funkce Ruby API pro vkládání jednotlivých instancí komponent nedovoluje editovat definici komponenty během vkládání, proto jsem musel implementovat ComponentInstanceObserver, který je volán po vložení každé instance do scény. Vždy po vložení komponenty dojde k jejímu škálování podle zadaného intervalu při vytvoření definice.

Tento observer se používá pro všechny billboardy (billboardy jsou v mém pluginu vytvořeny jako komponenty). Billboardy vložené pomocí volby "Vložit jeden objekt"se škálují podle zadané hodnoty a ty které jsou vloženy pomocí volby "Vložit více objektů"mají zadaný interval z kterého se vytvoří náhodná hodnota, podle které se mění jejich velikost.

Ukázka části zdrojového kódu ComponentInstanceObserver

```
1 class ComponentInstanceObserver < Sketchup::ModelObserver
2  def onPlaceComponent(instance)
3    for i in 0..definitionCount-1
4          cmp.instances.each{|ci|
5          if ci == instance
6                insertTransformation =
                   ci.transformation.origin;
7          end }
8        scaleTransformation = Geom::Transformation.scaling
           scale, 1.0, scale;
```

```
9             instance.transform!(scaleTransformation);
10            instance.transform!(insertTransformation);
```

3.3.3 Ukázka zdrojového kódu

Ukázka Ruby skriptu pro vytvoření Webdialogového okna.

```
1     UI.menu("PlugIns").add_item("Vytvorit_billboard") {
2         wd = UI::WebDialog.new( 'Vlozeni_billboardu', true,
              'chatterbox',650, 650, 0, 0, true );
3     # propojeni ruby skriptu s javascriptem
4     wd.add_action_callback( "ruby_billboard" ) do |dlg,
          msg|;
5     draw_billboard(msg); }
```

Ukázka JavaScriptu a jQuery zdrojového kódu.

```
1 // Plant object definition
2     function PlantObject (id, name, model, defaultScale) {
3         this.id = id;
4         this.name = name;
5         this.model = model;
6         this.defaultScale = defaultScale;
7         this.getObject = getObjectInfo; }
```

Ukázka Ruby skriptu pro vytvoření definice komponenty billboardu, která je použita pro vkládání jednotlivých instancí komponent (billboardů).

```
1 new_comp_def =
      Sketchup.active_model.definitions.add(componentName);
2 points = Array.new ; # define points for face
3     points[0] = ORIGIN ; # "ORIGIN" is a constant
4     points[1] = [100 * scale, 0, 0] ;
5     points[2] = [100 * scale, 0, 100 * scale] ;
6     points[3] = [0, 0, 100 * scale] ;
7 newface = new_comp_def.entities.add_face(points);
8 behavior = new_comp_def.behavior;
9 behavior.always_face_camera = true;
```

3.4 Náhled na scénu

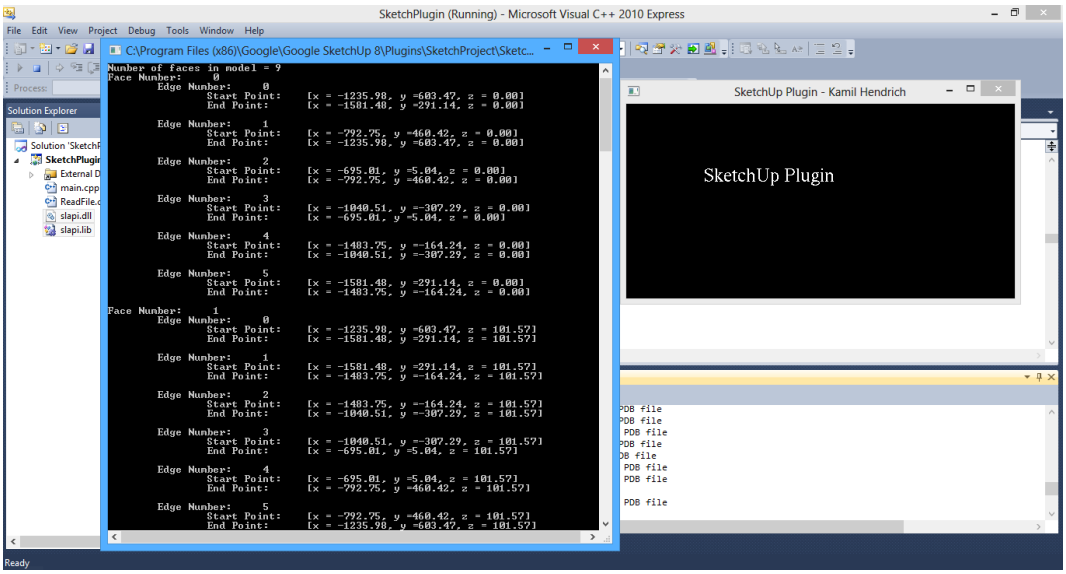
Vytvořil jsem Ruby skript, který přidá novou položku do menu programu do záložky Plugins. Tato položka spouští Ruby funkci, která uloží aktuální model do pomocného souboru do složky pluginu.

Model se musí uložit, protože C API nemůže přistupovat do modelu otevřeného v programu SketchUp. Dále funkce vytváří nové vlákno a spouští externí aplikaci naprogramovanou v jazyce C a předává jí jako parametr cestu k uloženému modelu.

Prostudoval jsem základní dokumentaci k C API a stáhnul si pomocné knihovny SketchUp SDK (C API, SketchUp Importer/Exporter Interface, základní dokumentace a ukázkové projekty). Vytvořil jsem projekt pomocí programu Microsoft Visual studio C++ 2010 Express. Vytvořil jsem aplikace v jazyce C, která využívá OpenGL. Tento program spouštím z programu SketchUp pomocí Ruby skriptu v novém vlákně.

Program využívá C API pro čtení informací o modelu ze souborů typu SKP. Při spuštění se na konzoli vypisují informace o modelu (počet ploch, jednotlivé hrany ploch a body které tyto hrany tvoří) (viz Obr. 3.5). Pomocí OpenGL vykresluji objekty a také nahrazuji billboardy pomocí OBJ modelů (viz Obr. 3.6). Každý billboard má ve svém popisu (vlastnost description, kterou má každá instance) uloženou cestu k OBJ modelu, který ho má nahradit. Modely jsou umístěny na odpovídající místo a v odpovídající velikosti podle transformační matice billboardů.

V aplikaci lze volit mezi volnou kamerou a kamerou, která se aktualizuje podle změny kamery v programu SketchUp pomocí View observeru. Objekty jsou vykreslovány pouze s výchozí hodnotou barvy, protože plugin nemá implementovanou část, která by načítala materiály nebo textury. Podporovány jsou všechny základní geometrické objekty (trojúhelník, obdélník, kruh a některé geometrické objekty z nich složené).

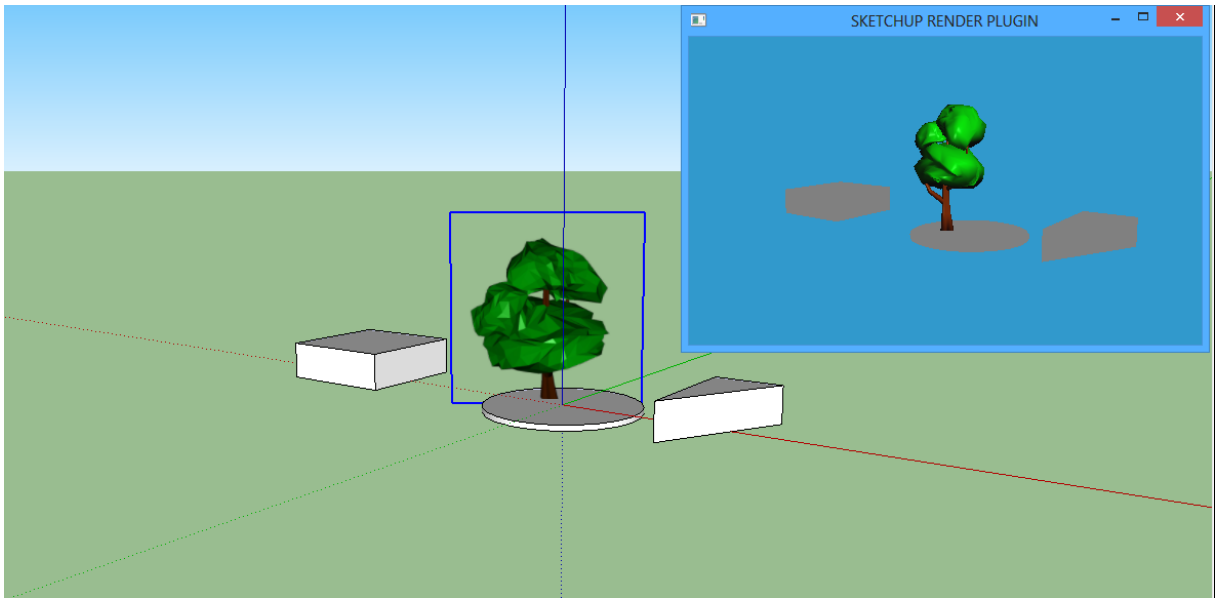


Obrázek 3.5: Program pro zobrazení informací o modelu

3.4.1 View observer

Ruby API poskytuje možnost vytvářet pozorovatele (Observery), kteří reagují na akce, které dělá uživatel při práci s programem SketchUp. Pozorovatelé se vytváří založením třídy, která implementuje rozhraní typu Observer (např. ViewObserver) a obsahuje požadované metody.

Protože C API neumožňuje přístup k právě otevřenému modelu, ale jen k souboru uloženému na pevném disku, musíme použít observer pro získávání změn kamery. Při změně



Obrázek 3.6: Ukázka náhledu na scénu

parametrů kamery se parametry uloží do souboru, do kterého může přistupovat plugin. Parametry jsou načteny ze souboru a plugin může reagovat na změnu kamery bez toho, aby se musel znovu spustit.

Ukázka zdrojového kódu pro vytvoření observeru a zápis do souboru

```
1 class CameraViewObserver < Sketchup::ViewObserver
2 def onViewChanged(view)
3     filePath = File.join( folderPath, "/Program/camera.txt"
        )
4     File.open(filePath, "w") do |f|
5     f.write(view.camera.eye.to_s);
6     f.write(view.camera.target.to_s);
7     f.write(view.camera.up.to_s);
```

3.4.2 Transformační matice

Každý objekt typu instance komponenty má v modelu uloženou svou transformační matici představující transformaci (posun = translation, rotaci = rotation a velikost = scale) ve trojrozměrném prostoru.

Matice (M) (viz Obr. 3.7) má 4 sloupce a 4 řádky. Obsahuje podmatici (R) velikosti 3x3 pro určení rotace kolem jednotlivých os, podmatici 3x1 určující posun objektu v prostoru a skalár W. Velikost objektu (změnu oproti původní velikosti) lze vypočítat pomocí velikostí tří vektorů vytvořených z jednotlivých sloupců rotační podmatice (viz Obr. 3.8).

$$M = \begin{bmatrix} R1x & R1y & R1z & Tx \\ R2x & R2y & R2z & Ty \\ R3x & R3y & R3z & Tz \\ 0 & 0 & 0 & W \end{bmatrix}$$

Obrázek 3.7: Transformační matice

$$Scale\ X = \sqrt[2]{R1x^2 + R2x^2 + R3x^2}$$

Obrázek 3.8: Příklad výpočtu změny velikosti v ose X

3.4.3 Ukázka zdrojového kódu

Ukázka Ruby skriptu pro vložení položky do menu. Určuje do jaké záložky menu, se má vložit nový objekt.

```
1 UI.menu("PlugIns").add_item("Spustit C plugin") {
2           programThread(path, "\""+modelPath+"\""); }
```

Ukázka Ruby skriptu, který slouží pro uložení aktuálního modelu do složky pluginu. Model se musí uložit, protože C API nemůže přistupovat do modelu otevřeného v programu SketchUp. Při změně modelu v programu se musí plugin znovu spustit, aby se změny uložili.

```
1 pathname = File.expand_path( File.dirname(__FILE__) )
2 modelPath = File.join(pathname,'SketchUpPlugin/test.skp')
3 Sketchup.active_model.save modelPath
```

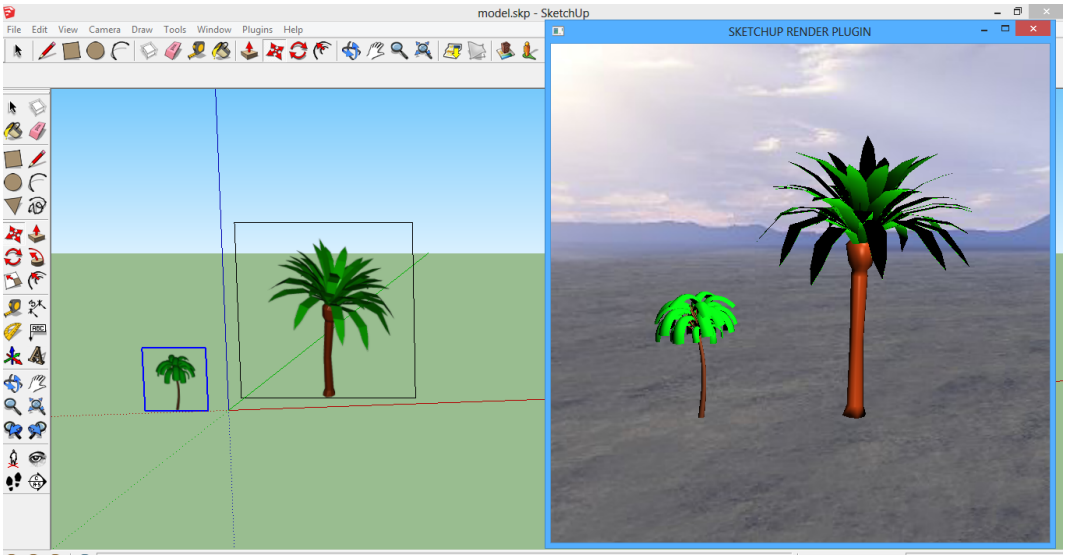
Ukázka Ruby skriptu, který slouží pro vytvoření nového vlákna pro spuštění externí aplikace. Vytvořit vlákno lze několika způsoby. Tento způsob vytvoří vlákno, které neblokuje program SketchUp. Aplikaci je při spuštění předán parametr obsahující cestu k uloženému modelu na pevném disku počítače.

```
1 timer = UI.start_timer(10,true){}
2           thread = Thread.new do
3                     system(path+" "+modelPath);
4           UI.stop_timer(timer);
```

Ukázka zdrojového kódu pro čtení informací z modelu pomocí C API. Z kontejneru entities jsou načteny všechny plochy (Face), hrany (Edge) které je tvoří a vrcholy (Vertex).

```
1 std::vector<SUFaceRef> faces(faceCount);
2 SUEntitiesGetFaces(entities, faceCount, &faces[0],
    &faceCount);
3 for (size_t i = 0; i < faceCount; i++) {
4   SUFaceGetNumEdges(faces[i], &edgeCount);
5   if (edgeCount > 0) {
```

```
6           std::vector<SUEdgeRef> edges(edgeCount);
7           SUFaceGetEdges(faces[i], edgeCount, &edges[0],
                &edgeCount);
8           for (size_t j = 0; j < edgeCount; j++) {
9             SUVertexRef startVertex = SU_INVALID, endVertex =
                  SU_INVALID;
10            SUEdgeGetStartVertex(edges[j], &startVertex);
11            SUEdgeGetEndVertex(edges[j], &endVertex);
```



Obrázek 3.9: Plugin pro zobrazení náhledu na scénu spuštený v prostředí SketchUp

Kapitola 4

Testování

Všechny pluginy jsem testoval v programech Google SketchUp verze 8 (8.0.16846) a Trimble SketchUp Make 2013 (13.0.4812). Doporučené verze pro správnou funkci všech pluginů je Trimble SketchUp 2013.

Testovací počítač je notebook značky Toshiba Satellite L850-1HN (Intel Core i5, 8GB RAM, Windows 8). Čas jsem měřil pomocí výpisů systémového času na Ruby konzoli programu SketchUp.

Pluginy testuji z hlediska paměťové a časové náročnosti (efektivita implementace pluginů) a také z hlediska použitelnosti pluginů reálnými uživateli (testování uživatelského rozhraní pluginů).

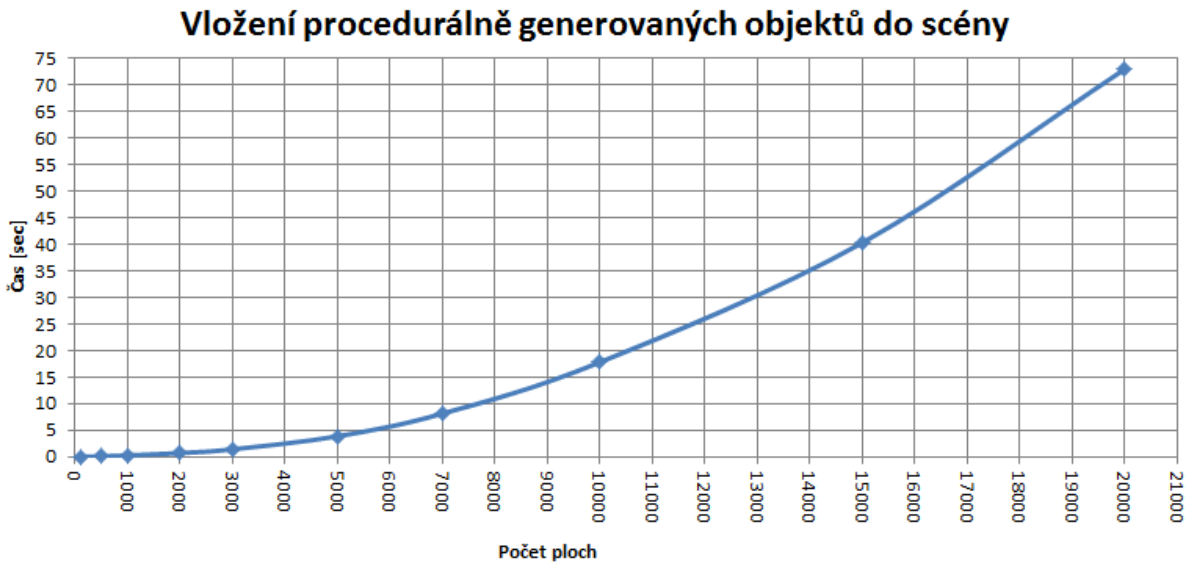
4.1 Procedurální vložení objektu do scény

4.1.1 Test rychlosti pluginu

Měření času potřebného pro vykreslení procedurálně generovaného objektu do scény pomocí Ruby skriptu bez načítání ze souboru v závislosti na počtu ploch (Faces) objektu (viz Tabulka 4.1). Objekty jsou kvádry skládané vedle sebe vytvářeny funkcí pomocí Ruby skriptu. Čas jsem měřil pomocí výpisů systémového času na Ruby konzoli programu SketchUp.

Počet ploch	100	500	1000	2000	3000	5000	7000	10000	15000	20000
Čas [s]	0,015	0.094	0,266	0,704	1,422	3,891	8,159	17,830	40,340	73,040

Tabulka 4.1: Časová náročnost vykreslování v závislosti na počtu generovaných ploch



Obrázek 4.1: Časová náročnost pluginu v závislosti na počtu generovaných ploch

Čas potřebný pro vykreslení objektů se zvyšuje se zvyšujícím se počtem ploch (viz Obr. 4.1). Pro velký počet ploch není čas zanedbatelný. Lze tedy říci, že vykreslování složitých objektů pomocí Ruby skriptu je pomalé.

4.1.2 Ukázka zdrojového kódu

Ruby skript, který se stará o procedurální vykreslování kvádrů do scény. Pomocí tohoto skriptu jsem testoval rychlost vykreslování objektů do scény pomocí Ruby v závislosti na počtu ploch.

```
1  time1 = Time.new
2    for step in 1..cubes
3          x1 = 0, x2 = width
4          y1 = run * step, y2 = run * (step + 1)
5          z = rise * step
6          pt1 = [x1, y1, z]
7          pt2 = [x2, y1, z]
8          pt3 = [x2, y2, z]
9          pt4 = [x1, y2, z]
10         new_face = entities.add_face pt1, pt2, pt3, pt4
11         new_face.pushpull thickness
12 time2 = Time.new
13 time3 = time2 - time1
14 puts "Time difference : " + time3.inspect
```

4.2 Vložení objektu do scény

4.2.1 Test funkčnosti pluginu

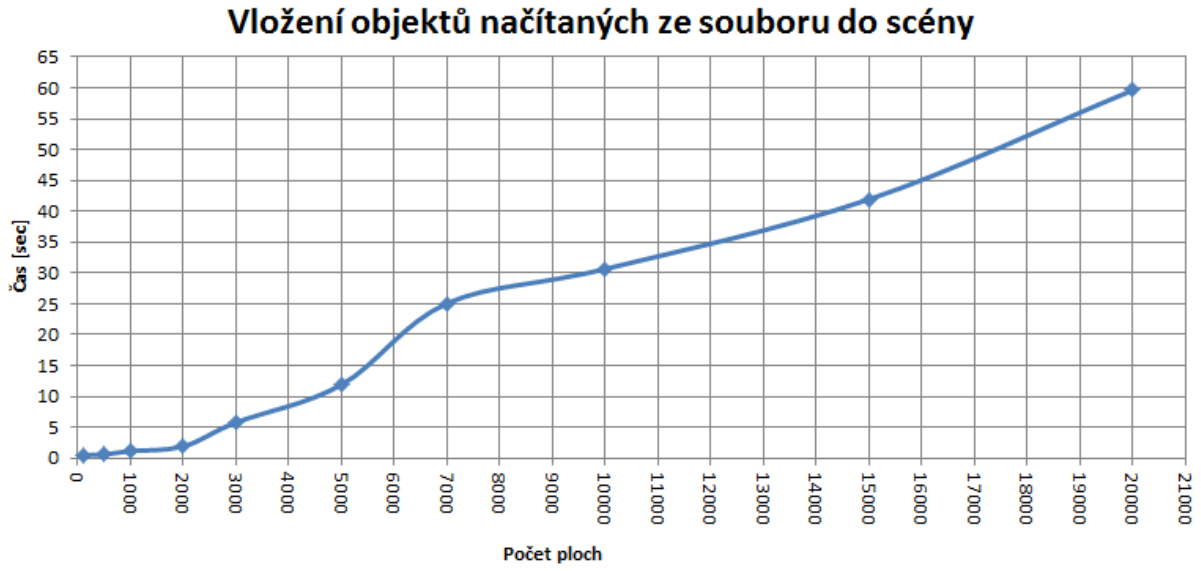
Testování pluginu jsem prováděl manuálně, tak že jsem zkusil vložit všechny objekty do scény v různých velikostech. Nenarazil jsem na žádnou chybu a všechny objekty byly v pořádku vloženy do scény. Největším problémem je vkládání složitých objektů, protože pomocí Ruby skriptu je velmi pomalé.

4.2.2 Test rychlosti pluginu

Měření času potřebného pro vložení objektu do scény v závislosti na počtu ploch (Faces) modelu (viz Tabulka 4.2). Zvolený model je načten ze souboru OBJ a podle zvolené velikosti přeškálován. Čas jsem měřil pomocí výpisů systémového času na Ruby konzoli programu SketchUp.

Počet ploch	100	500	1000	2000	3000	5000	7000	10000	15000	20000
Čas [s]	0,469	0,625	1,141	1,907	5,797	11,906	25,096	30,66	41,94	59,63

Tabulka 4.2: Časová náročnost pluginu v závislosti na počtu vkládaných ploch



Obrázek 4.2: Časová náročnost pluginu v závislosti na počtu vkládaných ploch

Čas potřebný pro vložení a vykreslení objektů se zvyšuje se zvyšujícím se počtem ploch (viz Obr. 4.2). Pro velký počet ploch není čas zanedbatelný. Lze tedy říct, že vkládání složitých objektů do scény pomocí Ruby skriptu a načítání dat z OBJ souboru je velmi pomalé. Plugin je pro velmi složité objekty skoro nepoužitelný.

4.2.3 Test paměťové náročnosti pluginu

Plugin pro vkládání objektů jsem otestoval pomocí programu Process Explorer. Změřil jsem náročnost na operační paměť potřebnou pro vložení objektu o určitém počtu ploch (viz Tabulka 4.3).

Počet ploch	1000	2000	5000	10000	20000
Paměť [MB]	10,7	9,8	10,2	15,6	32,3

Tabulka 4.3: Paměťová náročnost pluginu v závislosti na počtu vkládaných ploch

4.3 Vložení billboardu do scény

4.3.1 Test funkčnosti pluginu

Plugin jsem otestoval manuálně. Zkusil jsem vložit všechny objekty v různých velikostech a zvolil jsem různé textury. Testoval jsem funkci vložení 1 objektu i vložení více objektů. Vše fungovalo správně, všechny billboardy byly v pořádku vloženy do scény na správné místo a neobjevil jsem žádnou chybu.

Při testování jsem zjistil, že pozorovatelé událostí (observery) se po startu programu připojí k aktuálně otevřenému modelu a pokud uživatel vytvoří nový model anebo otevře nějaký existující model, observery nejsou připojeny. Proto jsem musel implementovat pozorovatele aplikačních událostí (AppObserver), který pokaždé znovu připojí pozorovatele událostí k aktuálnímu modelu.

4.3.2 Test rychlosti pluginu

Rychlost jsem měřil pomocí Ruby skriptu, který jsem vytvořil. Skript vkládal do scény zadaný počet billboardů a vypisoval systémový čas na Ruby konzoli programu SketchUp. Časovou náročnost v závisloti na počtu vkládaných billboardů ukazuje tabulka 4.4.

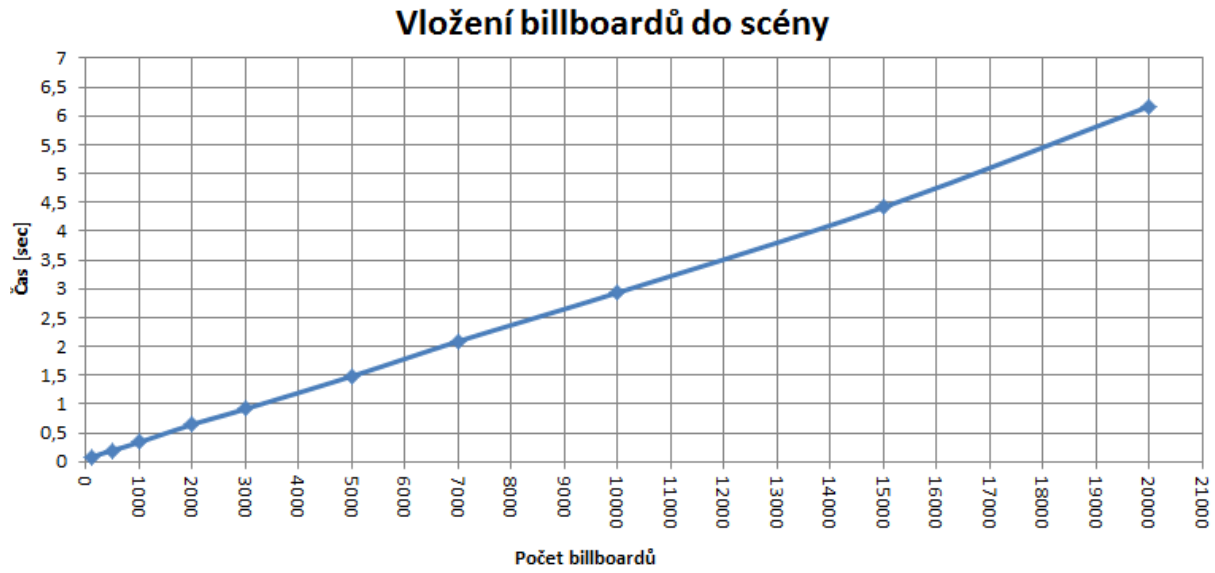
Počet billboardů	100	500	1000	2000	3000	5000	7000	10000	15000	20000
Čas [s]	0,083	0,197	0,338	0,646	0,917	1,484	2,089	2,935	4,415	6,169

Tabulka 4.4: Časová náročnost pluginu v závislosti na počtu vkládaných billboardů

Čas potřebný pro vložení billboardů se zvyšuje se zvyšujícím se počtem vkládaných billboardů (viz Obr. 4.3). I pro velký počet billboardů je čas skoro zanedbatelný. Lze tedy říct, že vkládání billboardů do scény pomocí Ruby skriptu je velmi rychlé.

4.3.3 Test paměťové náročnosti pluginu

Plugin pro vkládání billboardů jsem otestoval pomocí programu Process Explorer. Změřil jsem náročnost na operační paměť potřebnou pro vložení určitého počtu billboardů (viz Tabulka 4.5).



Obrázek 4.3: Časová náročnost pluginu v závislosti na počtu vkládaných billboardů

Počet ploch	1000	2000	5000	10000	20000
Paměť [MB]	11,2	13,4	19,8	32,5	54,7

Tabulka 4.5: Paměťová náročnost pluginu v závislosti na počtu vkládaných billboardů

4.3.4 Ukázka zdrojového kódu

Ruby skript, kterým jsem testoval rychlost pluginu v závislosti na počtu vkládaných billboardů.

```
1 def speedTest()
2         time1 = Time.new;
3         new_comp_def = draw_one_billboard
              (materialPath,modPath,0.5,1.0,nil,nil,1);
4         for i in 1..20000
5          Sketchup.active_model.active_entities.
               add_instance(new_comp_def, trans);
6         end
7         time2 = Time.new
8         time3 = time2 - time1
9         puts "Time difference : " + time3.inspect
```

4.4 Náhled na scénu

4.4.1 Test funkčnosti pluginu

Plugin dokáže vypisovat informace o modelech načítané ze souborů SKP pomocí C API. Tyto informace jsou vypisovány správně. Funkčnost jsem kontroloval na jednoduchých objektech, které jsem sám vytvořil tak, abych dokázal zkontrolovat správnost hodnot. Zvládá vykreslování jednoduchých geometrických objektů (trojúhelník, obdélník, kruh a některé geometrické objekty z nich složené). U složitých objektů může být vykreslení nepřesné. Plugin dokáže billboardy v modelu nahradit objekty ve formátu OBJ a umístit je na správné pozice podle pozice původních billboardů.

Při testování jsem zjistil, že pozorovatelé událostí (observery) se po startu programu připojí k aktuálně otevřenému modelu a pokud uživatel vytvoří nový model anebo otevře nějaký existující model, observery nejsou připojeny. Proto jsem musel implementovat pozorovatele aplikačních událostí (AppObserver), který pokaždé znovu připojí pozorovatele událostí k aktuálnímu modelu.

Testováním bylo odhaleno, že plugin nelze používat v programu SketchUp verze 8, protože tato verze má problémy se spouštěním aplikací v novém vlákně [3]. Plugin lze spustit a pracuje správně, ale aplikace SketchUp je nestabilní a může dojít k jejímu ukončení. Tato chyba byla ve SketchUp verze 2013 opravena, doporučuji tedy plugin používat pouze v této verzi anebo novější.

4.4.2 Test rychlosti pluginu

Měření času potřebného pro úplné spuštění pluginu a vykreslení všech objektů ve scéně pomocí OpenGL v závislosti na počtu ploch (Faces) (viz Tabulka 4.6). Čas jsem měřil pomocí výpisu systémového času na konzoli.

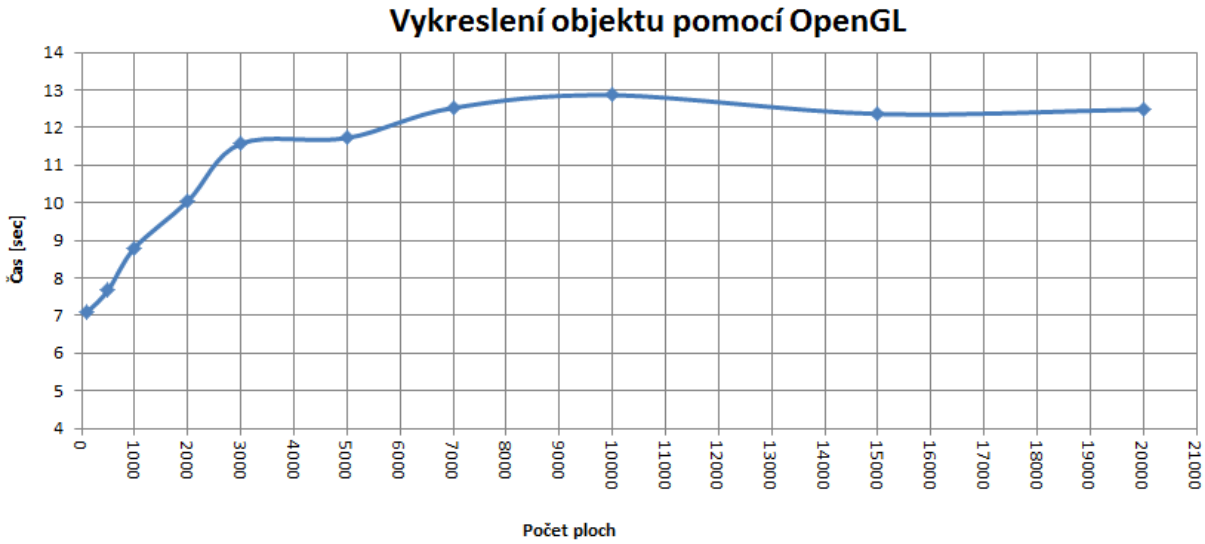
Počet ploch	100	500	1000	2000	3000	5000	7000	10000	15000	20000
Čas [s]	7,08	7,67	8,8	10,04	11,57	11,72	12,51	12,85	11,32	12,49

Tabulka 4.6: Časová náročnost pluginu v závislosti na počtu vykreslovaných ploch

Čas potřebný pro spuštění pluginu a vykreslení objektů je nízký i pro velký počet ploch (viz Obr. 4.4). Lze tedy říct, že vykreslováni objektů pomocí OpenGL je rychlé i pro složité modely.

4.4.3 Test paměťové náročnosti pluginu

Plugin pro vykreslování objektů jsem otestoval pomocí programu Process Explorer. Změřil jsem náročnost na operační paměť potřebnou pro spuštění pluginu a vykreslení objektu o určitém počtu ploch (viz Tabulka 4.7).



Obrázek 4.4: Časová náročnost pluginu v závislosti na počtu vykreslovaných ploch

Počet ploch	1000	2000	5000	10000	20000
Paměť [MB]	68,1	68,7	69,3	68,9	71,8

Tabulka 4.7: Paměťová náročnost pluginu v závislosti na počtu vykreslovaných ploch

4.5 Testování uživatelského rozhraní

Uživatelské rozhraní vytvořených pluginů jsem otestoval pomocí malé cílové skupiny osob vybrané ze studentů ČVUT. Cílovou skupinou jsou osoby ve věku 15 až 50 let, které mají alespoň základní zkušenosti s prací s počítačem a s nějakým 3D editorem.

4.5.1 Test uživatelského rozhraní

Do testovací skupiny se mi podařilo sehnat 5 osob. Každá osoba dostala seznam několika jednoduchých úkolů, které měla splnit. Já jako moderátor testu jsem pouze dohlížel na postup plnění úkolů a snažil se osoby neovlivňovat a neradit jim pokud nebylo opravdu potřeba.

Měřené veličiny jsou čas potřebný pro splnění každého úkolu v sekundách a míra správnosti splnění každého úkolů v procentech.

Od testu očekávám odhalení zásadních chyb v pluginech, kterých si jako autor nemůžu být vědom. Chci identifikovat problémy v použitelnosti pluginu, jestli nějaké existují a zjistit jak je plugin použitelný pro reálné uživatele. Jestli ho dokáží bez problému ovládat a orientovat se v grafickém prostředí pluginu.

4.5.2 Zadání testu

Úkol č.1 - Prvním úkolem je vytvořit pomocí pluginu "Vytvořit billboard"(naleznete ho v menu v záložce Plugins) objekt typu Datlovník pravý (Phoenix dactylifera) [Palma]. Zvolte velikost 8 metrů a nastavte libovolnou texturu listů a kmene. Vložte jeden objekt na libovolné místo ve scéně.

Úkol č.2 - Druhým úkolem je vytvořit 3 libovolné (různé) rostliny typu "Pokojová rostlina"a vložit je do scény. Pomocí pluginu "Vytvořit billboard"(naleznete ho v menu v záložce Plugins) vytvořte postupně 3 rostliny ve velikostech 0,4m nebo 1,0m, nastavte libovolnou texturu listů a kmene (stonku) a vložte je na libovolné místo do scény.

Úkol č.3 - Posledním úkolem je vytvořit pomocí pluginu "Vytvořit billboard"(naleznete ho v menu v záložce Plugins) 5 objektů typu Kokosovník ořechoplodý (Cocos nucifera) [Palma] a vložit je do scény. Zvolte správný objekt ze seznamu. Nastavte interval velikostí od 2m do 10m, nastavte libovolnou texturu listů a kmene a pomocí tlačítka "Vložit více objektů"vložte 5 objektů na libovolné místo do scény.

4.5.3 Vyhodnocení testu

Čas potřebný pro splnění jednotlivých úkolů je spíše orientační. Hlavním kritériem hodnocení je míra splnění zadaných úkolů vyjádřená v procentech. A celkové hodnocení je průměr výsledků v jednotlivých úkolech.

Osoba	čas 1 [sec]	míra [%]	čas 2 [sec]	míra [%]	čas 3 [sec]	míra [%]	celkové hodnocení [%]
1	27,34	100	36,52	100	34,60	100	100
2	31,74	100	45,68	100	49,81	100	100
3	47,35	100	55,37	100	63,63	100	100
4	25,68	100	46,20	100	52,42	100	100
5	58,60	100	128,34	85	86,50	100	95

Tabulka 4.8: Výsledek testu uživatelského rozhraní

Celkový výsledek testu je dobrý (viz Tabulka 4.8). Jen jedna osoba měla malý problém s orientací v grafickém uživatelském rozhraní a udělala malou chybu ve 2. úkolu, když se snažila použít funkci pro vložení více objektů místo toho aby vložila 3 rozdílné objekty. Ostatní osoby splnili úkoly bez problémů a ani po skončení testu si nestěžovali na žádné zásadní chyby v použitelnosti pluginu a orientaci v grafickém uživatelském rozhraní.

Kapitola 5

Návrh dalšího postupu

Vzhledem k rozsáhlosti projektu nebylo možné pluginy zcela dokončit. Hlavní důraz byl kladen na vytvoření funkčních pluginů se základními funkcemi, které půjdou jednoduše rozšířit. Tato kapitola slouží jako krátké shrnutí nedokončených funkcí a možných vylepšení.

5.1 Vložení objektu do scény

Vývoj pluginu byl zastaven, protože testy ukázaly, že vkládání modelů ve formátu OBJ pomocí Ruby je neefektivní a pro složité modely časově velmi náročné. Přesto bych jako další postup pro zlepšení pluginů navrhoval zkusit vkládat objekty do scény jinými způsoby anebo vkládat pouze velmi zjednodušené objekty s malým počtem polygonů. Dále prozkoumat možnosti jak vkládat objekty do scény i s požitím materiálů jejich jednotlivých částí, které jsou u OBJ modelů uloženy v souboru MTL.

5.2 Vložení billboardu do scény

Plugin funguje správně a splňuje všechny požadavky ze zadání. Jedním z možných vylepšení je přidání více modelů rostlin a stromů. Dále je možné ke každé textuře rostliny vytvořit masku, podle které by se vykresloval stín objektu při zapnutém stínování, protože SketchUp podporuje textury s průhledným pozadím, ale zobrazuje i stín průhledné oblasti. Při tvorbě pluginu jsem masky pro textury nevytvářel, protože při vytváření náhledu na scénu jsou billboardy nahrazeny OBJ modely.

5.3 Náhled na scénu

Prozkoumat další možnosti, jak získávat změny, které provede uživatel v prostředí programu SketchUp. Protože C API neumožňuje přístup k modelům, které jsou právě otevřeny v programu SketchUp. Proto je náhled na scénu po každé změně modelu uživatelem neaktuální dokud uživatel znovu nespustí plugin. Při každém spuštění si plugin aktuální model uloží na disk do svého adresáře a s ním pak může pracovat.

Plugin nemá implementováno načítání textur a materiálů z modelu, proto vykresluje objekty pouze s výchozí šedou barvou. Zvládá vykreslování jednoduchých geometrických objektů (trojúhelník, obdélník, kruh a některé geometrické objekty z nich složené). U složitých objektů může být vykreslení nepřesné. Proto navrhuji podrobněji prozkoumat možnosti, jak použít vrcholy získané z C API pro vykreslení všech druhů geometrických objektů.

Dalším možným vylepšením je použít textury zvolené při vytváření billboardů na otexturování modelů. Problémem je, že všechny modely by musely mít minimálně 2 materiály (1 pro kmen a 1 pro listy), které by se ve všech modelech jmenovali stejně, aby šlo identifikovat, na které části objektu se mají použít textury.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo najít několik existujících pluginů s podobnými funkcemi jako v zadání práce a otestovat je. Analyzovat možnosti tvorby pluginů pro program SketchUp pomocí různých programovacích jazyků. Výstupem by měli být vlastní pluginy vytvořené podle požadavků v zadání práce.

Nalezl jsem několik pluginů, které se zabývají vkládáním objektů (hlavně rostlin a stromů) do scény. Prozkoumal možnosti nastavení, které poskytují a inspiroval se pro tvorbu vlastního pluginu. Také jsem nalezl několik pluginů a programů, které se zabývají tvorbou náhledu na scénu. Prozkoumal jsem způsoby, jakými vytvářejí náhled a jak komunikují s programem SketchUp.

Analyzoval jsem rozhraní, pomocí kterých mohou pluginy komunikovat s programem SketchUp. Ruby API je vývojáři programu velmi podporované, má kvalitně vypracovanou dokumentaci a mezi tvůrci pluginů je hodně používané. C API neposkytuje stejné funkce jako Ruby API a vývojáři programu je podporováno mnohem méně. Dokumentace je oproti Ruby API hůře zpracovaná a mezi tvůrci pluginů je jen velmi málo osob, které ho používají pro vývoj svých pluginů.

Vytvořil jsem tři funkční pluginy. První umožňuje vložení vybraných modelů rostlin a stromů ve formátu OBJ do scény. Vývoj tohoto pluginu byl zastaven, protože vkládání složitých modelů bylo velmi neefektivní a časově náročné. Druhý umožňuje vložení zjednodušených modelů ve formě billboardů do scény. Ten se podařilo zcela dokončit podle všech požadavků. A třetí načítá informace o modelu ze souborů typu SKP, zobrazuje jejich textový výpis na konzoli a vytváří náhled na scénu pomocí OpenGL. Zvládá vykreslení základních geometrických objektů. Tento plugin poskytuje základní funkce podle zadání.

Vytvořené pluginy jsem otestoval z hlediska funkčnosti, časové a paměťové náročnosti a jeden z nich také z hlediska použitelnosti. Všechny pluginy fungují správně a poskytují základní požadované funkce. Vzhledem k rozsáhlosti projektu nebylo možné pluginy zcela dokončit, proto jsem navrhl další postup, jak pluginy dokončit a vylepšit.

Literatura

[1] Csaba Pozsárkó. Face-me Billboard Component, 2014.
`http://sketchucation.com/resources/tutorials/320-face-me-billboard-component`, stav z 15. 4. 2014.

[2] Google. Google Earth, 2014.
`http://www.google.cz/intl/cs/earth/`, stav z 1. 5. 2014.

[3] RATHBUN, D. System() crashes SketchUp, 2012.
`https://groups.google.com/forum/#!topic/sketchupruby/NTZGfYSY4Kc`, stav z 1. 5. 2014.

[4] RINEHART, M. Edges to Rubies — The Complete SketchUp Tutorial, 2011.
`http://www.martinrinehart.com/models/tutorial/`, stav z 1. 5. 2014.

[5] SCARPINO, M. *Automatic SketchUp: Creating 3-D Models in Ruby. (1st ed.)*. Eclipse Engineering LLC, 2010.

[6] SCHREYER, A. C. *Architectural Design with SketchUp - Component-Based Modeling, Plugins, Rendering, and Scripting (1st ed)*. 2012.

[7] The jQuery Team. jQuery, 2010.
`http://jquery.com/`, stav z 1. 5. 2014.

[8] The jQuery Team. jQuery Browser compatibility, 2010.
`http://jquery.com/browser-support/`, stav z 11. 5. 2014.

[9] Trimble Navigation Limited. Oficiální web programu SketchUp, 2014.
`http://www.sketchup.com/`, stav z 25. 4. 2014.

[10] Trimble Navigation Limited. SketchUp Ruby API vývojářská dokumentace, 2014.
`http://www.sketchup.com/intl/en/developer/docs/`, stav z 15. 4. 2014.

[11] Trimble Navigation Limited. SketchUp C API vývojářská dokumentace, 2014.
`http://www.sketchup.com/intl/en/developer/su-api/`, stav z 1. 5. 2014.

[12] Trimble Navigation Limited. SketchUp Extension warehouse, 2014.
`http://extensions.sketchup.com/`, stav z 1. 5. 2014.

[13] Trimble Navigation Limited. SketchUp 3D warehouse, 2014.
`https://3dwarehouse.sketchup.com/`, stav z 1. 5. 2014.

[14] web:models1. Archive 3D - databáze 3D modelů, 2014. http://archive3d.net/, stav z 15. 5. 2014.

[15] web:models2. Turbo Squid - databáze 3D modelů, 2014. http://www.turbosquid.com/, stav z 15. 5. 2014.

[16] web:models3. TF3DM - databáze 3D modelů, 2014. http://tf3dm.com/, stav z 15. 5. 2014.

Příloha A

Seznam použitých zkratek

GUI (Graphical User Interface) = grafické uživatelské rozhraní.

OpenGL (Open Graphics Library) = standard specifikující multiplatformní rozhraní (API) pro tvorbu aplikací počítačové grafiky.

HTML (HyperText Markup Language) = je značkovací jazyk pro hypertext pro vytváření webových stránek.

CSS (Cascading Style Sheets) = kaskádový styl je jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

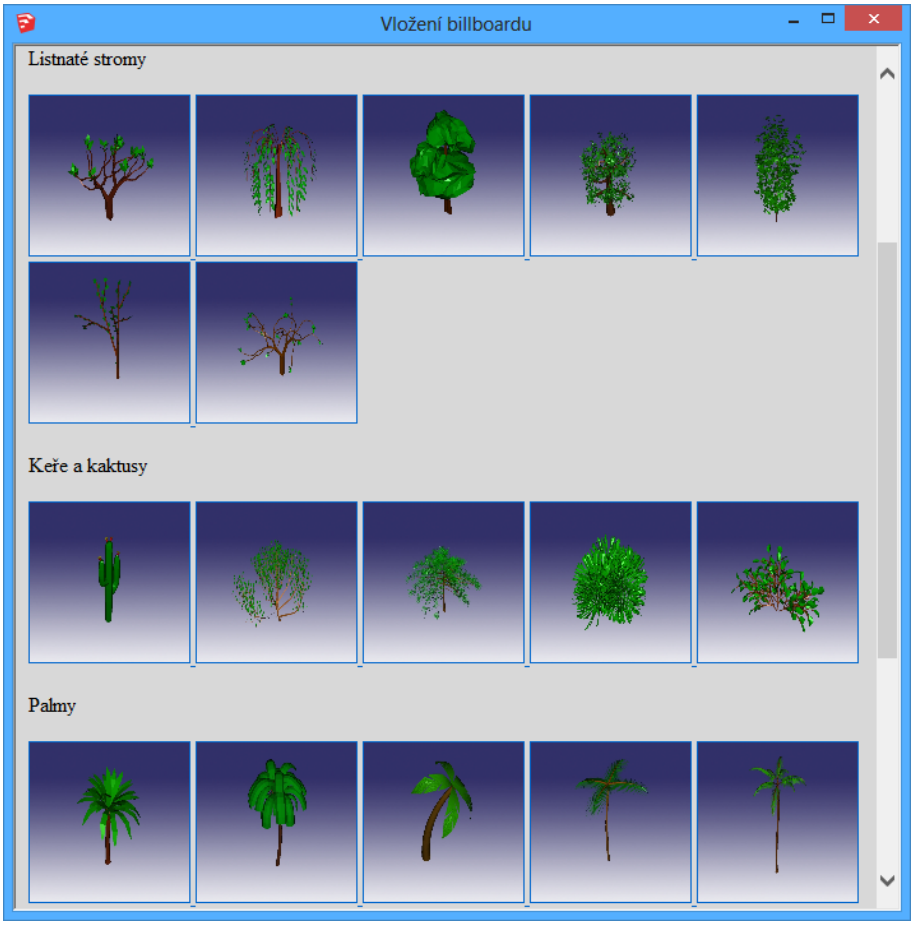
JS (JavaScript) = je multiplatformní, objektově orientovaný skriptovací jazyk. Zpravidla se používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky.

API (Application Programming Interface) = rozhraní pro komunikaci s aplikací. Jedná se o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny, které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu.

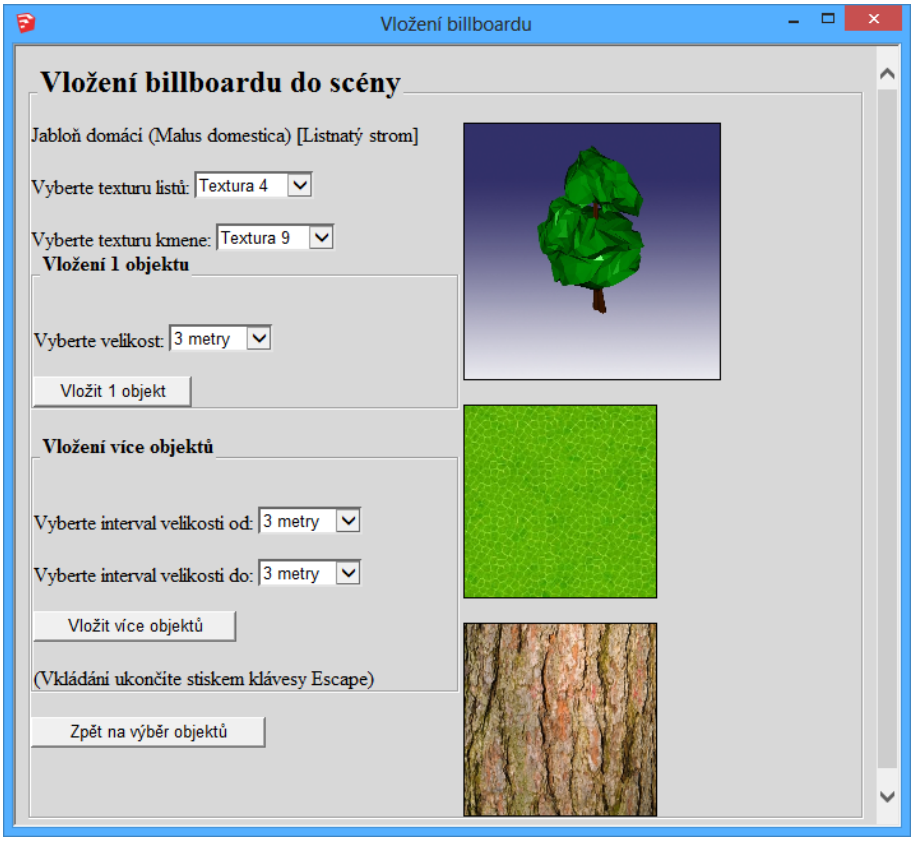
SDK (Software development kit) = sada nástrojů pro vývoj, která umožňuje vytváření aplikací pro určitý program, hardwarovou platformu a operační systém.

Příloha B

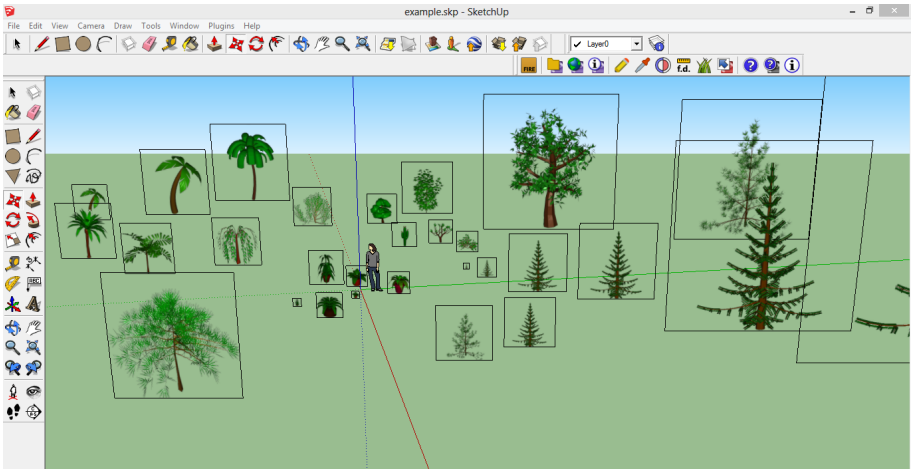
Obrázky aplikace



Obrázek B.1: Ukázka pluginu pro tvorbu billboardů - výběr objektu

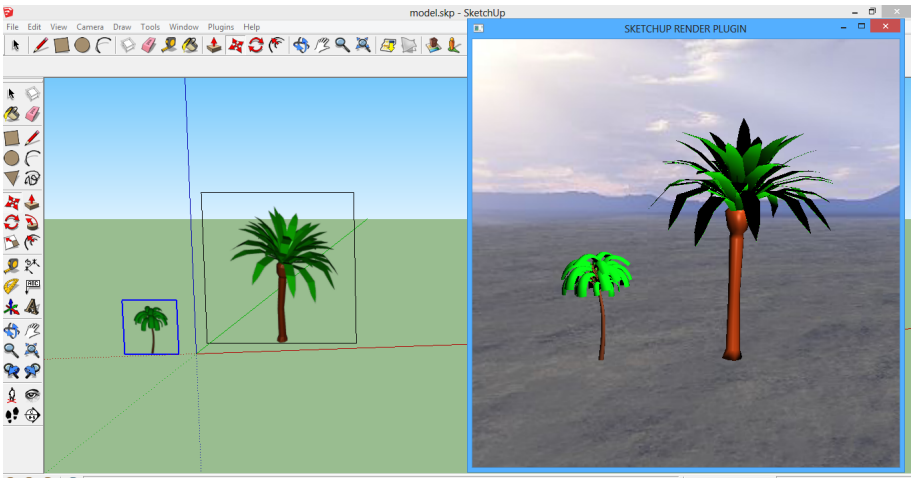


Obrázek B.2: Ukázka pluginu pro tvorbu billboardů - nastavení parametrů

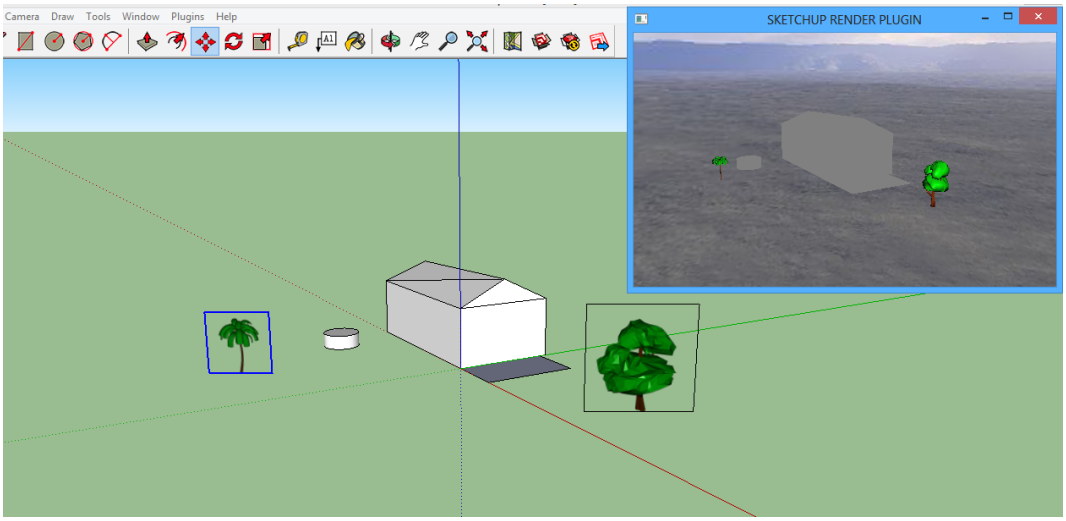


Obrázek B.3: Ukázka pluginu pro tvorbu billboardů - vložené billboardy

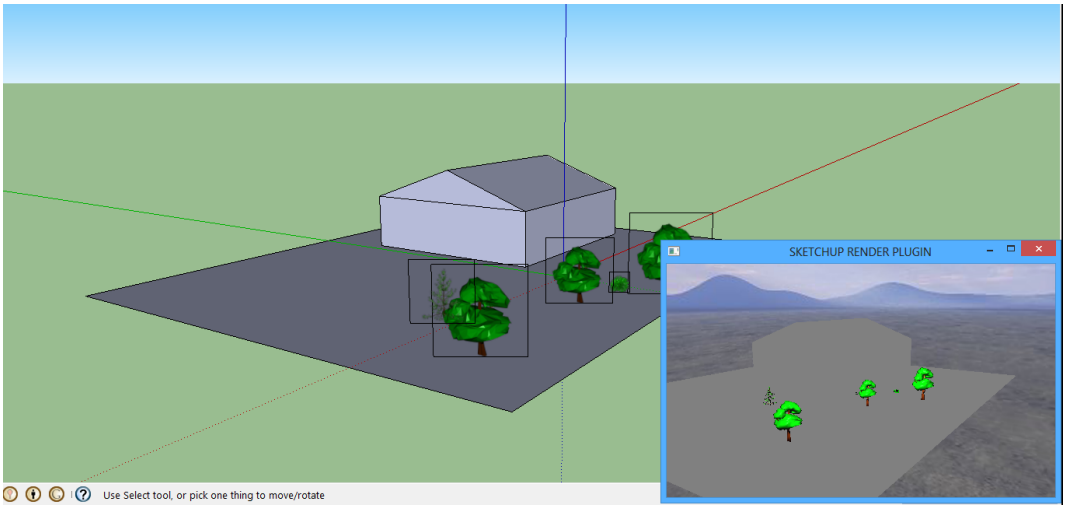
Obrázek B.4: Ukázka pluginu pro náhled na scénu - billboardy nahrazené 3D objekty



Obrázek B.5: Ukázka pluginu pro náhled na scénu - billboardy nahrazené 3D objekty



Obrázek B.6: Ukázka pluginu pro náhled na scénu



Obrázek B.7: Ukázka pluginu pro náhled na scénu

Příloha C

Instalační a uživatelská příručka

C.1 Příprava / Podmínky

Doporučené operační systémy pro správnou funkci všech pluginů jsou Microsoft Windows XP, Vista, 7, 8 nebo 8.1.

Podmínkou pro správnou funkci všech pluginů je mít nainstalovaný program SketchUp verze 2013 nebo vyšší. Dále mít nainstalovaný webový prohlížeč Microsoft Internet Explorer verze 8.0 nebo vyšší nebo jiný kompatibilní webový prohlížeč. A musí být povolen JavaScript.

Některé verze program SketchUp požadují .NET Framework verze 4.0.

Knihovna jQuery verze 1.x vyžaduje jeden z webových prohlížečů ve verzi Internet Explorer 6+, Safari 5.1+, Opera 12.1x, Chrome nebo Firefox [8].

C.2 Instalace

Jednotlivé archivy s pluginy se musí rozbalit do složky programu SketchUp obsahující Pluginy. Typicky je to: `C:\Program Files\Google\Google SketchUp 8\Plugins\` (pro OS Windows). Po rozbalení archivu do příslušné složky by se měli pluginy automaticky přidat do menu programu SketchUp. Jednotlivé pluginy jsou na sobě vzájemně nezávislé.

C.3 Hardwarové nároky

Minimální hardwarové nároky programu SketchUp jsou 1,6 GHz processor, 1 GB RAM, 500 MB volného místa na pevném disku, 3D grafická karta s 512 MB paměti a s podporou OpenGL verze 2.0 nebo vyšší.

Pro plugin vytvářející náhled na scénu je povinná grafická karta s podporou OpenGL verze 3.1 nebo vyšší.

Příloha D

Obsah přiloženého CD

- Elektronická verze bakalářské práce ve formátu PDF.
- Elektronická verze zadání bakalářské práce (oskenovaný originál) ve formátu JPG.
- Archiv s pluginem vložení objektu do scény ve formátu ZIP.
- Archiv s pluginem vložení billboardu do scény ve formátu ZIP.
- Archiv s pluginem náhled na scénu ve formátu ZIP.
- Archiv s projektem pro MS Visual studio ve formátu ZIP.
- Archiv ve formátu ZIP obsahující SketchUp SDK.
- Archiv ve formátu ZIP obsahující další knihovny a framework použitý při tvorbě pluginů.
- Soubor README.txt obsahující instalační a uživatelskou příručku.
- Soubor `SketchUp 2013 Make setup Windows.exe` je instalační soubor programu SketchUp.
- Soubor `examplemodel.skp` obsahující ukázkovou scénu vytvořenou v programu SketchUp.