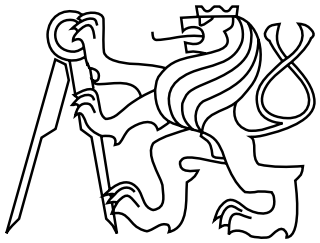
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ

# DIPLOMOVÁ PRÁCE

PRAHA 2014 Bc. Pavel TOBIÁŠ

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEODÉZIE A KARTOGRAFIE



# DIPLOMOVÁ PRÁCE

## VYUŽITÍ APLIKACE SKETCHUP PRO TVORBU JEDNODUCHÉHO INFORMAČNÍHO SYSTÉMU

Vedoucí práce: Ing. Petr Soukup, Ph.D.
Katedra mapování a kartografie

leden 2014

Bc. Pavel TOBIÁŠ

**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE**
**Fakulta stavební**
Thákurova 7, 166 29 Praha 6

# ZADÁNÍ DIPLOMOVÉ PRÁCE

studijní program: Geodézie a kartografie

studijní obor: Geodézie a kartografie

akademický rok: 2013/2014

Jméno a příjmení diplomanta: Bc. Pavel Tobiáš

Zadávající katedra: Katedra mapování a kartografie

Vedoucí diplomové práce: Ing. Petr Soukup, Ph.D.

Název diplomové práce: Využití aplikace SketchUp pro tvorbu jednoduchého informačního systému

Název diplomové práce v anglickém jazyce: Utilization of the SketchUp Application to Create a Simple Information System

Rámcový obsah diplomové práce: Analýza možností programu SketchUp jako základu informačního systému. Propojení prvků modelu s dalšími textovými a grafickými informacemi. Vývoj modulu pro správu těchto informací v aplikaci SketchUp. Export modelu a informací pro sdílení na webu.

Datum zadání diplomové práce: 23.9.2013 Termín odevzdání: 20.12.2013
(vyplňte poslední den výuky přísl. semestru)

Diplomovou práci lze zapsat, kromě oboru A, v letním i zimním semestru.

Pokud student neodevzdal diplomovou práci v určeném termínu, tuto skutečnost předem písemně zdůvodnil a omluva byla děkanem uznána, stanoví děkan studentovi náhradní termín odevzdání diplomové práce. Pokud se však student řádně neomluvil nebo omluva nebyla děkanem uznána, může si student zapsat diplomovou práci podruhé. Studentovi, který při opakovaném zápisu diplomovou práci neodevzdal v určeném termínu a tuto skutečnost řádně neomluvil nebo omluva nebyla děkanem uznána, se ukončuje studium podle § 56 zákona o VŠ č.111/1998 (SZŘ ČVUT čl 21, odst. 4).

*Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.*

.....
vedoucí diplomové práce

.....
vedoucí katedry

Zadání diplomové práce převzal dne: 23.9.2013

.....
diplomant

Formulář nutno vyhotovit ve 3 výtiscích – 1x katedra, 1x diplomant, 1x studijní odd. (zašle katedra)

Nejpozději do konce 2. týdne výuky v semestru odešle katedra 1 kopii zadání DP na studijní oddělení a provede zápis údajů týkajících se DP do databáze KOS.
DP zadává katedra nejpozději 1. týden semestru, v němž má student DP zapsanou.
(Směrnice děkana pro realizaci stud. programů a SZZ na FSv ČVUT čl. 5, odst. 7)

## ABSTRAKT

Cílem této práce je prozkoumat možnosti propojování modelu v aplikaci SketchUp s dalšími informacemi. Nejprve práce popisuje rozšíření funkcí této aplikace pomocí zásuvných modulů. Dále je zkoumána správa atributů objektů v modelu. Uvedeny jsou příklady zajímavých zásuvných modulů. Získané poznatky jsou dále využity k naprogramování vlastního zásuvného modulu pro správu atributů. Nakonec jsou zkoumány možnosti sdílení modelu na webu včetně dalších informací.

## KLÍČOVÁ SLOVA

SketchUp, 3D model, zásuvný modul, atributy

## ABSTRACT

The goal of this thesis is to explore possibilities of linking a model in the SketchUp application with additional information. First the work describes an extension of functions in this application using plugins. Next the administration of objects attributes in a SketchUp model is examined. Examples of interesting plugins are introduced. The acquired knowledge is also applied to program own plugin for the attribute management. Finally possibilities of sharing a model on the web including additional information are examined.

## KEYWORDS

SketchUp, 3D model, plugin, attributes

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Využití aplikace SketchUp pro tvorbu jednoduchého informačního systému“ jsem vypracoval samostatně, pouze s využitím odborných konzultací. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ..... .....

(podpis autora)

# PODĚKOVÁNÍ

Rád bych poděkoval Ing. Petru Soukupovi, Ph.D. za pomoc a ochotu při vedení mé diplomové práce. Dále děkuji své rodině a přátelům za podporu při psaní této práce. Zvláštní dík patří Bc. Martinu Doškářovi za pomoc při testování pluginu TIS a Ing. Miroslavu Tobiášovi za vymyšlení názvu tohoto programu.

# Obsah

# Úvod

Tvorbou 3D modelů v programu SketchUp se zabývala již řada semestrálních, bakalářských a diplomových prací. 3D modely, které byly vytvářeny v rámci projektů studentů programu Geodézie a kartografie, zobrazují zejména památkové objekty. Ať už se jedná o jednodušší modely, optimalizované pro zobrazování ve vrstvě fotorealistických budov v aplikaci Google Earth nebo o složité detailnější modely, jejichž složitost vyžaduje využití jiného způsobu prezentace, vždy tyto modely zajímavě přibližují skutečný vzhled modelovaných objektů.

3D modely mohou tedy sloužit jako vhodný doplněk prezentace objektů zájmu. Poskytují představu o vzhledu objektů, další informace potom uživatel získává většinou jinou cestou. Pokud zůstaneme u památkových objektů, mohou být těmito informacemi například údaje o jejich historii. Zajímavé by ovšem bylo, kdyby mohl uživatel získat informace o jednotlivých objektech v rámci modelu přímo v prostředí aplikace SketchUp. Tato aplikace by tak mohla vlastně sloužit jako jednoduchý informační systém. Taková možnost ale zatím nebyla příliš popsána.

Cílem této diplomové práce je tedy nejprve prozkoumat možnosti propojování modelu s dalšími informacemi. Takové informace mohou být buď umístěny na webu nebo může jít o informace, které se nacházejí v souborech uložených lokálně vedle vlastního souboru modelu. V práci bude nejprve popsáno rozhraní SketchUp Ruby API, které umožňuje přidávat další funkčnost programu SketchUp vytvářením zásuvných modulů, které jsou programovány v jazyce Ruby. Popsána bude také možnost správy atributů jednotlivých objektů modelu ve SketchUpu. Následovat budou postřehy z testování programů, které jsou použitelné pro propojení modelu v aplikaci SketchUp s dalšími informacemi. Většina z nich budou právě pluginy, tedy zásuvné moduly, pro program SketchUp.

Poznatky o SketchUp Ruby API a atributech v programu SketchUp budou dále v rámci diplomové práce využity pro naprogramování vlastního zásuvného modulu pro tuto aplikaci. Tento modul bude určen pro nastavování a prohlížení dalších informací například o částech památkových objektů. V textu této práce budou tedy nejdříve krátce popsány vlastnosti programovacího jazyka Ruby. Následovat bude podrobný popis navrženého modulu. Kromě popisu vlastností a nástrojů modulu bude popsáno i jeho testování na vzorovém modelu památkového objektu.

Zobrazování dalších informací o objektech v prostředí aplikace SketchUp je jistě zajímavé, v dnešní době je ale velké množství informací prezentováno prostřednictvím internetu. Poslední kapitoly práce se tedy budou zabývat možnostmi prezentace

3D modelů na webu včetně dalších informací o jejich částech. Základem pro takový informační systém bude opět model vytvořený v programu SketchUp. Popsány budou cesty pro export takového modelu do formátu vhodného pro zobrazování na webu a pro přiřazení dalších informací. Kromě krátkého obecného popisu bude zejména probráno testování programů, které toto umožňují.

# 1 Rozšíření programu SketchUp, atributy

SketchUp[1] je aplikace, která slouží pro tvorbu 3D modelů. Následující kapitola bude nejprve obsahovat její krátký popis. Dále budu popsány zásuvné moduly, pomocí kterých je možno přidávat do této aplikace další potřebné funkce. Nakonec bude rozebrána možnost správy atributů jednotlivých objektů modelu.

## 1.1 O programu SketchUp

Poprvé byl tento program představen v roce 2000 společností @Last Software. V roce 2006 potom zakoupila SketchUp společnost Google, která program vyvíjela od verze 6 do verze 8. Jedním z využití programu SketchUp bylo vytváření 3D modelů budov, které byly poté zobrazovány v aplikaci Google Earth[2]. V roce 2012 byl SketchUp poměrně překvapivě odkoupen společností Trimble. V souvislosti s přechodem na novou technologii tvorby 3D modelů pro Google Earth bylo od 1. října 2013 ukončeno přijímání modelů, které byly vytvořeny uživateli, do vrstvy fotorealistických budov [1]. Modely tak lze nyní sdílet pouze prostřednictvím Trimble galerie 3D objektů[3]. Program SketchUp je ale využíván i pro profesionální účely například architekty nebo stavebními a strojními inženýry. Těch se potom uvedené omezení příliš netýká, takže program SketchUp bude zřejmě i nadále hojně používán [2].

V současné době je nejnovější verzí Trimble SketchUp 2013. K dispozici je tento produkt buď bezplatně jako SketchUp Make, pro komerční účely je potom nutno zakoupit placenou verzi SketchUp Pro. V době psaní této práce stojí jedna licence Pro verze programu s příplatkem za roční podporu 474 euro. Uživatel Pro verze získá například možnost exportu do více 3D formátů, plnou funkčnost rozšíření *Dynamické komponenty* (viz 2.4) a další aplikaci LayOut, která slouží pro tvorbu výkresů a podkladů pro prezentaci modelu. Jak SketchUp Pro, tak SketchUp Make jsou k dispozici pro operační systémy Windows (od Windows XP) a Mac OS X (verze 10.7 nebo novější) [3].

Funkce programu SketchUp již byly popisovány mnohokrát v jiných pracích. Verze 6, v níž se modelování neliší od verzí novějších, je popisována např. v bakalářské práci, která se věnuje 3D modelu Hernychovy vily [4]. Pro získání představy o fungování programu lze použít také výuková videa na oficiálních stránkách programu SketchUp[4]. V souvislosti s popisem aplikace SketchUp je často zmiňován

[1] http://www.sketchup.com/
[2] http://www.google.cz/intl/cs/earth
[3] http://sketchup.google.com/3dwarehouse/?hl=cs
[4] http://www.sketchup.com/learn/videos?playlist=58&playlist=58

například netradiční nástroj *Push/Pull*, který umožňuje jednoduchým způsobem vytvářet trojrozměrné objekty z dvourozměrných ploch.

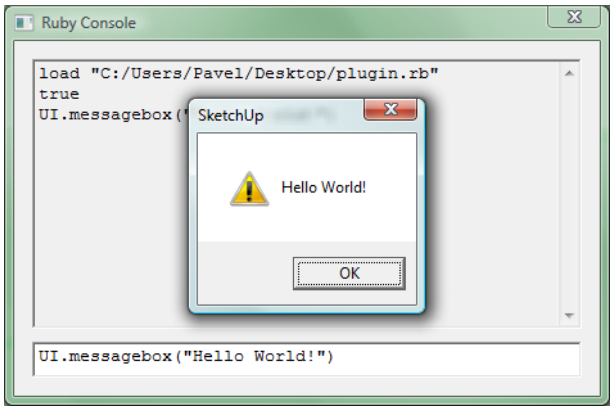
Dále v textu budou zmiňovány komponenty a skupiny uvnitř modelu v aplikaci SketchUp. Proto by bylo vhodné se o nich zmínit a hlavně popsat jejich odlišnosti. Skupina (Group) vznikne spojením několika entit (hran a ploch) modelu. Uživatel může v programu SketchUp vytvořit skupinu pomocí volby *Make Group* z kontextového menu, které je dostupné po kliknutí pravým tlačítkem myši na soubor vybraných entit. Vytvořená skupina se chová jako jeden objekt, což může být pro další modelování výhodné. Pro editaci skupiny (tedy editaci entit uvnitř skupiny) je nutno ji otevřít, nejjednodušeji dvojklikem levým tlačítkem myši. Pokud zkopírujeme vytvořenou skupinu, získáme nezávislé objekty, tedy nezávislé soubory entit, které jsou v modelu uloženy tolikrát, kolik je kopií skupiny.

Vytvoření komponenty (Component) je podobné vytvoření skupiny a je možné z kontextového menu volbou *Make Component*. Stejně jako u skupiny budou touto volbou sloučeny vybrané entity modelu do jednoho celku. Tento celek je uchováván jako definice komponenty (Component definition). V modelu jsou potom umístěny instance komponenty (Component instances). Definice komponenty obsahuje informace o poloze jednotlivých entit v rámci komponenty. Takové entity jsou tedy uchovávány pouze jednou, i když se v modelu nachází více instancí komponenty. U každé instance potom stačí uchovávat pouze její polohu v rámci modelu. Pokud otevřeme komponentu pro editaci (obdobně jako u skupiny), editujeme vlastně definici komponenty a provedené změny se projeví u všech instancí komponenty v modelu. Všechny definice v modelu lze prohlížet v okně *Components* (Window/Components/In Model).

## 1.2 SketchUp Ruby API, zásuvné moduly

Některé programy, které byly v rámci diplomové práce testovány, jsou programy samostatnými, využívají tedy model vytvořený v aplikaci SketchUp (v souboru `skp`), ale další funkce již zprostředkovávají na této aplikaci nezávisle. Většina dále popisovaných programů jsou ale „pouze“ pluginy pro aplikaci SketchUp. Pluginem se potom rozumí *zásuvný modul* (z anglického *to plug in* – zasunout, připojit), tedy program, který rozšiřuje možnosti dané aplikace a bez ní nefunguje [5].

Aplikace SketchUp je pro vytváření uživatelských doplňků vhodná, pro tento účel je s ní distribuováno rozhraní SketchUp Ruby API (Application Programmer's Interface). Jak je patrné z názvu, pro programování pluginů je v tomto případě používáno programovacího jazyka Ruby (viz 3). S využitím tohoto rozhraní je potom




Obr. 1.1: Ruby Console

možno vytvářet rozšíření, která mohou sloužit pro automatizaci kreslení v programu, nastavování atributů (viz 1.3) atd. Důležité je také zmínit, že pro plné využití Ruby API stačí i neplacená verze SketchUp Make.

Pro základní ovládání SketchUp Ruby API je v programu SketchUp vložena Ruby Console (Window/Ruby Console). Ta může sloužit pro zkoušení základních příkazů a metod a také pro odladění programu, kdy je v ní možno nechat vypisovat např. hodnoty zájmových proměnných a tak kontrolovat chod spuštěného programu.

Přímo pro vytváření pluginů samozřejmě nelze Ruby konzoli využít. Používá se proto vstupních souborů v textovém formátu, které se vytvářejí ve zvoleném textovém editoru. Takové vstupní soubory mají příponu *.rb. Aby s tímto souborem SketchUp pracoval, je nutno ho zkopírovat do adresáře, kde jsou uloženy všechny používané pluginy pro program SketchUp, tedy do adresáře *Plugins* uvnitř složky nainstalovaného programu. Plugin se automaticky spustí po restartu programu SketchUp. Každý takto vytvořený skript, tedy soubor `rb`, je již možno nazvat pluginem.

Pro pokročilé užití je možno vytvořit plugin jako rozšíření SketchUp Extension. K tomu je v Ruby API používána stejnojmenná třída `SketchupExtension`. Tato rozšíření potom lze ve SketchUpu vypínat a zapínat v okně *System Preferences* v záložce *Extensions* (Window/Preferences/Extensions). V tomto okně se nachází také volba *Install Extension*. Pomocí této volby je možné instalovat plugin, který je distribuován jako soubor `rbz`. To je případ složitějších rozšíření, která používají více souborů `rb` a další soubory pomocné. Soubor `rbz` je běžný ZIP archiv, kterému byla změněna přípona. Pomocí výše uvedené funkce se soubory obsažené v archivu zkopírují do adresáře *Plugins* [6].

Soubory `rb`, které se nenacházejí v adresáři *Plugins* programu SketchUp, lze spouštět také a to pomocí Ruby Console. K tomu slouží příkaz `load`, např.:

```
load 'C:/Users/Pavel/Desktop/plugin.rb'
```

Tímto způsobem lze také jednoduše znovu spouštět pluginy, ve kterých byly provedeny změny, bez nutnosti restartu aplikace SketchUp.

Velké množství pluginů, které jsou k dispozici, není dodáváno ve formě `rb` souborů, ale jako zašifrované soubory `rbs`. To zbavuje programátory nutnosti poskytovat svůj kód všem uživatelům k nahlédnutí. Proto lze bohužel fungování většiny dále popisovaných pluginů pouze odhadovat na základě jejich chování. Pro zašifrování je používán program Ruby Scrambler[5].

## 1.3 Atributy objektů v modelu

Uchovávání atributů objektů v modelu je možné i v neplacené verzi programu SketchUp. Každému objektu (jak jednoduchým entitám jako je hrana, tak složeným komponentám) tak lze přidávat další popisné informace, tedy atributy. Pro správu, tj. pro vkládání a prohlížení atributů, není v základní verzi SketchUpu žádný standardní nástroj, a je proto nutno použít Ruby API. Atributy jsou ve SketchUpu ukládány ve dvojicích *Name/Value* ve slovnících (AttributeDictionary) [8].

Pro nastavení atributů objektu je používána metoda `set_attribute`.

```
entity.set_attribute "AttributeDictionary", "Name", "Value"
```

Pro prohlížení atributů potom slouží metoda `[]`.

```
attrdicts = entity.attribute_dictionaries
attrdict = attrdicts["AttributeDictionary"]
value = attrdict["Name"]
```

V příkazech výše byly nejprve nalezeny všechny slovníky dané entity. Dále byl vybrán slovník požadovaný. Hodnota atributu v proměnné `value` je potom ze slovníku získána metodou `[]` pro klíč `Name` [9].

Na základě atributů funguje značná část pluginů a programů, které budou popsány dále včetně oficiálního pluginu Dynamic Components (Dynamické komponenty).

[5] http://www.sketchup.com/intl/en/developer/docs/utilities.php

# 2 Programy pro využití v aplikaci SketchUp

V současné době je k dispozici velké množství pluginů pro aplikaci SketchUp. Většinu z nich lze nalézt na webu *SketchUp Extension Warehouse*[1]. Pro účely diplomové práce byly proto nejen na tomto webu hledány zásuvné moduly, které umožňují propojení objektů modelu s dalšími informacemi. Dále budou popsány nalezené moduly a budou uvedeny i postřehy z jejich testování.

## 2.1 Sketchup Attribute Manager

SketchUp Attribute Manager[2] umožňuje základní práci s atributy – jejich nastavení a prohlížení. Atributy nastavené tímto programem je možné dále využít i v dalších popisovaných programech (např. 2.7).

Po nainstalování je Attribute Manager přístupný z kontextového menu SketchUpu po kliknutí na vybranou entitu pravým tlačítkem jako položka *Attributes*. Volbou *Show* je potom zpřístupněno grafické rozhraní pluginu, které umožňuje přidávání atributů. Nejprve je nutno vytvořit novou kategorii (*New Category*), poté je již možné vkládat nové dvojice Name/Value pomocí tlačítka *Add an attribute*. Možná je potom samozřejmě editace záznamů a také mazání atributů i kompletních kategorií.

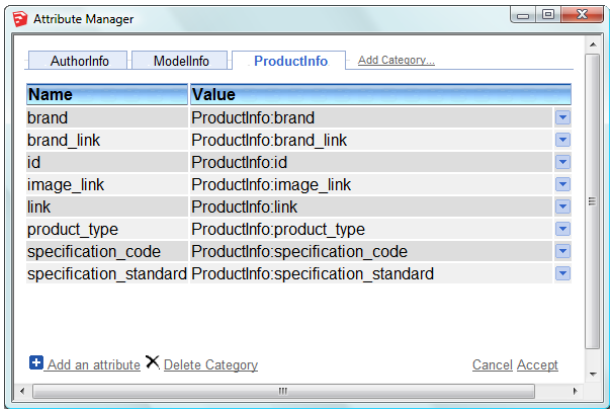
Pokud si zobrazíme atributy entity vytvořené v Attribute Manageru jiným způsobem (např. přímo z Ruby Console), zjistíme, že jsou atributy uloženy klasickým způsobem `AttributeDictionary/Name/Value` a vytvářené kategorie jsou tedy jen jinak nazvanými slovníky.

Výhodou Attribute Manageru je určitě pěkné grafické rozhraní, které je vytvořeno speciálně pro tento plugin. Užitečná může být i volba *Make standard categories*, pomocí které se vytvoří předpřipravené atributy, rozdělené do kategorií AuthorInfo, ModelInfo, ProductInfo. V případě, že je modelovaným objektem existující produkt a je-li třeba vytvořit jeho metadata, stačí již pouze vyplnit chybějící hodnoty jednotlivých atributů.

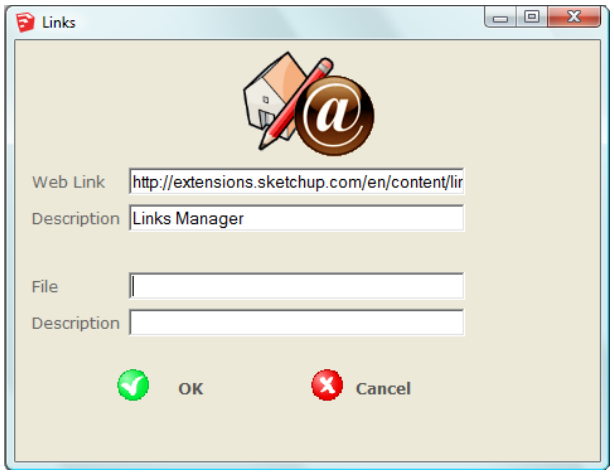
## 2.2 Links Manager

Autor pluginu Links Manager[3] vystupuje na webu Extension Warehouse pod uživatelským jménem *D.Bur*. Tento plugin využívá atributů k uložení odkazu na

[1] http://extensions.sketchup.com/en
[2] https://code.google.com/p/sketchupattributemanager/
[3] http://extensions.sketchup.com/en/content/links-manager




Obr. 2.1: Grafické rozhraní pluginu Attribute Manager



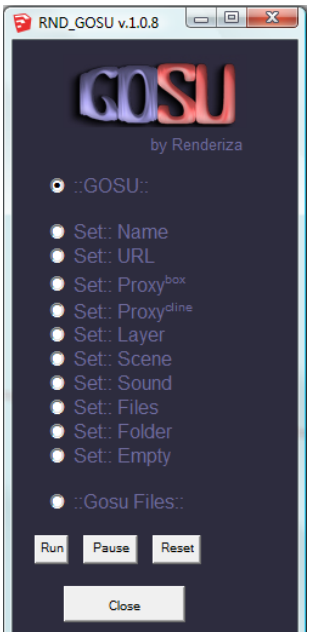

Obr. 2.2: Plugin Links Manager

webovou stránku („Web Link“) a na soubor uložený lokálně („File“) včetně popisů. Zdrojový kód pluginu je zakódován (`rbs`), ale jeho fungování lze odhadnout z uložených atributů.

Hlavní nástroje pluginu jsou dostupné z nového toolbaru (panelu nástrojů). Zadávací rozhraní je k dispozici po kliknutí na entitu levým tlačítkem myši. Se současně stisknutou klávesou `Ctrl` je potom možno přejít na odkaz. Bylo vyzkoušeno, že jak Web Link, tak odkaz na lokální soubor mohou obsahovat i relativní cestu a jako Web Link může být uložena i cesta k lokálnímu souboru.

Při zobrazení uložených atributů je patrné, že Links Manager ukládá všechny odkazy do slovníku `link_data`, pod klíče `FILE`, `FILE Description` resp. `URL`, `URL Description`

Pro zadávání odkazů využívá Links Manager rozhraní vytvořené pravděpodobně pomocí Ruby třídy `WebDialog`. Webové stránky jsou zobrazovány ve výchozím prohlížeči uživatele, lokální soubory potom v příslušném programu dle typu souboru.




Obr. 2.3: Rozhraní modulu GOSU

## 2.3 GOSU

Plugin GOSU[4], který vytvořil uživatel *Renderiza*, poskytuje podobné možnosti jako výše uvedený Links Manager, lze ho ale použít pouze pro komponenty. Pomocí menu programu (opět pravděpodobně Ruby `WebDialog`) je možno pro vybranou komponentu nastavit následující:

- Set:: Name

  Nastavení názvu komponenty (jedná se o položku okna *Entity Info – Name*).

- Set:: URL

  Nastavení s komponentou spojené URL (nastavení lokálního souboru s relativní cestou se pomocí této volby nepodařilo).

- Set:: Proxy

  Neslouží k nastavení informací, ale pro dočasné nahrazení komponenty jednodušším objektem. Dvojklikem se aplikuje na všechny výskyty dané komponenty.

- Set:: Scene

  Vybrání scény, na kterou SketchUp přejde po kliknutí na komponentu.

[4]http://extensions.sketchup.com/en/content/gosu

- Set:: Layer

  Vložení komponenty do zvolené vrstvy.

- Set:: Sound

  Přiřazení zvuku komponentě. Zvuk bude přehrán po kliknutí na komponentu přímo ve SketchUpu. Použít lze proto pouze audio soubory ve formátu `wav`.

- Set:: Files

  Asociování vybraného souboru. Může se jednat o video, obrázek, aplikaci...

- Set:: Folder

  Nastavení složky propojené s komponentou. Ve vybraném souboru je možno uchovávat například obrázky spojené s komponentou.

- Set:: Empty

  Vymaže všechny nastavené údaje.

Vzhledem k tomu, že se přiřazením údajů neplní atributy komponenty, lze předpokládat, že jsou informace ukládány jiným způsobem.

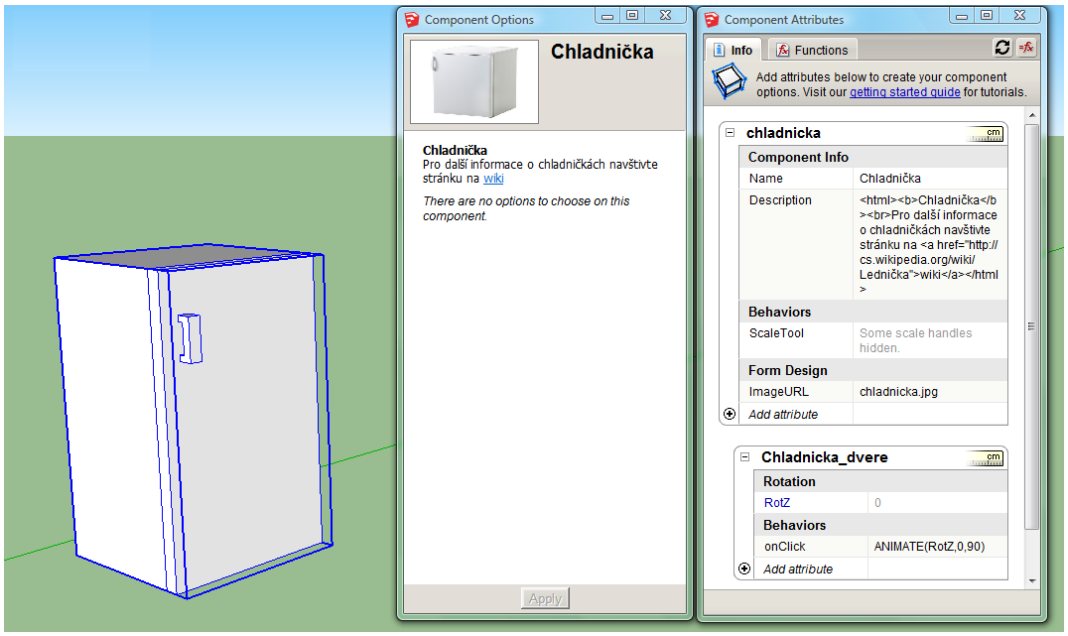
Při vybrání volby ::GOSU:: v rozhraní modulu lze potom nastavené údaje využít, tj. přejít na nastavenou URL, zobrazit asociovaný obrázek atd. Použití pouze jednoho rozhraní pro nastavování a prohlížení lze chápat jako nedostatek pluginu, neboť i uživatel, který si chce doplňující informace pouze prohlédnout, používá vždy stejné menu se všemi volbami jako ten, kdo atributy nastavil. Za nevýhodu je také možno považovat to, že lze uvedená nastavení provádět pouze pro komponenty.

## 2.4 Dynamické komponenty

Rozšíření *Dynamic Components*[5] je již standardní součástí aplikace SketchUp od verze 7 a bylo vytvořeno přímo vývojáři SketchUpu. Plně využívat dynamické komponenty lze ale pouze v placené Pro verzi programu. Ve verzi standardní lze pouze využívat jejich již nastavené vlastnosti.

Dynamické komponenty využívají speciálních atributů k několika účelům. Lze pomocí nich například vytvářet komponenty, které „inteligentně“ reagují na změnu měřítka (smart scaling – při prodloužení plotu přibývá sloupků, u schodů se se změnou velikosti mění počet stupňů...). Dále je možno nastavit tzv. *onClick* vlastnosti,

[5] http://extensions.sketchup.com/en/content/dynamic-components




Obr. 2.4: Dynamické komponenty

tedy akce, které se provedou při využití nového nástroje *Interact Tool*. Může se jednat např. o změnu barvy, otevírání a zavírání dveří, změnu polohy objektů a podobně.

Z hlediska informačního systému je potom nejzajímavější vlastností dynamických komponent možnost ukládat a zobrazovat popisné informace o komponentách. Jedná se o základní atributy jako je jméno a popis, přidávat lze ale i další vlastní atributy. Popis (Description) lze potom formátovat pomocí několika povolených HTML tagů [13]. Pro prohlížení atributů je k dispozici samostatný nástroj, kde se již popis zobrazí dle použitého formátování. Součástí popisu mohou být i hypertextové odkazy. Bylo zjištěno, že se může jednat skutečně pouze o odkazy na stránky umístěné na webu. Odkazy na soubory umístěné lokálně nelze pomocí tohoto nástroje vytvářet.

Pro ilustraci fungování dynamických komponent byl vytvořen jednoduchý model. Jelikož jsou dynamické komponenty v praxi hojně využívány v souvislosti s nábytkem a dalším vybavením bytů, jedná se o model chladničky. Celá chladnička je komponentou, uvnitř se potom nachází komponenta další – dveře lednice. Komponenta byla opatřena krátkým popisem, jehož součástí je i hypertextový odkaz. Tento popis se zobrazí při prohlížení vlastností komponenty (nástroj *Component Options*). V okně nástroje se zobrazí také obrázek, který byl komponentě přiřazen. Změna velikosti lednice byla omezena pouze na změnu výšky. Jiný rozměr nelze změnit. Dveře chladničky obsahují další atributy, které zajišťují *onClick* vlastnost – otevírání a zavírání dveří.

Pokud si zobrazíme atributy komponenty, získáme následující:

```
Dictionary Name/Attribute name/Attribute value

dynamic_attributes/has_movetool_behaviors/0.0
dynamic_attributes/hasbehaviors/1.0
dynamic_attributes/lengthunits/CENTIMETERS
dynamic_attributes/name/chladnicka
dynamic_attributes/description/<html><b>Chladnička</b>
<br>Pro další informace o chladničkách navštivte stránku na
<a href="http://cs.wikipedia.org/wiki/Lednička">wiki</a></html>
dynamic_attributes/imageurl/chladnicka.jpg
dynamic_attributes/name/Chladnička
dynamic_attributes/scaletool/123
_last_lenx/23.6220472440945
_last_leny/23.6220472440945
_last_lenz/39.3700787401575
```

Je tedy patrné, že atributy dynamických komponent se ukládají do slovníku `dynamic_attributes`. Zřejmé jsou také atributy `name` a `description`. Atribut `image url` obsahuje relativní cestu k obrázku v náhledu komponenty. V atributu `scaletool` je obsažena informace o blokování změn rozměru v některých směrech. Zajímavé jsou atributy `_last_len` mimo slovník `dynamic_attributes`, které zřejmě označují velikost obalového boxu komponenty (zobrazí se při vybrání komponenty) a to v palcích. V atributech dveří chladničky potom nalezneme:

```
Dictionary Name/Attribute name/Attribute value

dynamic_attributes/_has_movetool_behaviors/0.0
dynamic_attributes/_hasbehaviors/1.0
dynamic_attributes/_iscollapsed/false
dynamic_attributes/_lengthunits/CENTIMETERS
dynamic_attributes/_name/Chladnicka_dvere
dynamic_attributes/_onclick_state1/0
dynamic_attributes/_rotz_formula/
dynamic_attributes/_rotz_label/RotZ
dynamic_attributes/onclick/ANIMATE(RotZ,0,90)
dynamic_attributes/rotz /0.0
_last_lenx/3.93700787401575
_last_leny/23.6220472440945
_last_lenz/39.3700787401575
```

Přibyly zde tedy atributy popisující rotaci dveří chladničky kolem osy Z (tedy jejich „otevírání“). Hlavní je zde atribut `onclick`, který popisuje animaci otevírání mezi stavy s úhlem otevření 0 a 90 stupňů. U dveří otevřených se dotyčné atributy změní na:

```
dynamic_attributes/_onclick_state1/1
dynamic_attributes/_rotz_formula/
dynamic_attributes/_rotz_label/RotZ
dynamic_attributes/onclick/ANIMATE(RotZ,0,90)
dynamic_attributes/rotz/90.0
```

Stav `_onclick_state1` se tedy po otevření změnil z 0 na 1. Atribut `rotz` potom z 0 na 90 stupňů. Po zavření dveří se atributy pochopitelně vrátí do původního stavu.

Dynamické komponenty tedy pomocí speciálních atributů získávají další funkčnost. Využít se dají samozřejmě také pouze pro uchovávání informací o modelovaných objektech. V rámci formátování popisu lze odkazovat i na webové stránky. Chybí ale možnost odkazovat na lokální soubory spojené s komponentami (další obrázky, podrobnější model).

## 2.5 Museum/Gallery HTML Reference

Tento plugin[6] využívá externí HTML soubor k ovládání prohlížení modelu v programu SketchUp. Informace o modelovaných objektech tak nejsou uloženy v `skp` souboru (např. v atributech objektů) ale právě v souboru HTML. Pomocí tohoto souboru lze ovládat pozice kamery, viditelnost vrstev a přecházet mezi scénami ve SketchUpu (a také ovládat animace s dalším dodatečným pluginem). Referenční soubor tak může obsahovat popis, obrázky a odkazy na další informace jako běžná webová stránka, ale kromě toho i odkazy sloužící pro ovládání modelu.

Referenční soubor využívá standardních odkazů (anchor) pro:

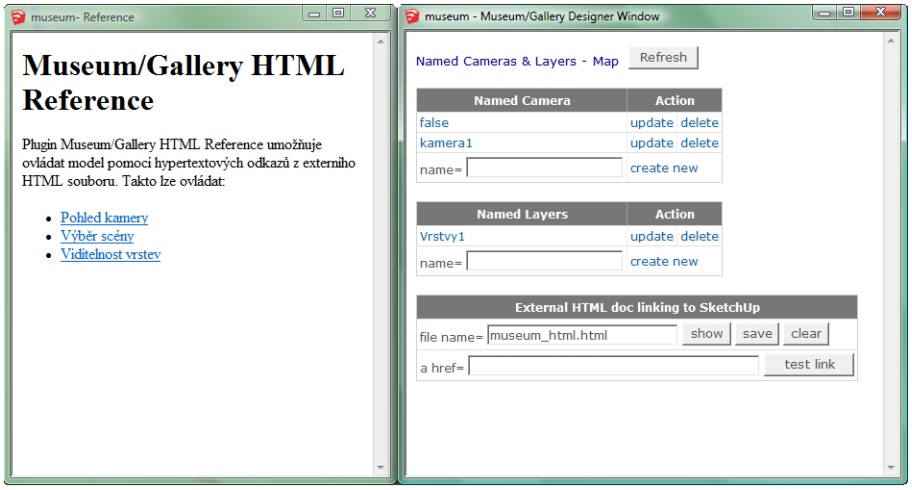
- Změnu pohledu kamery

  ```
  <a href="skp:retrieve_cam@kamera1">Pohled kamery</a>
  ```

- Výběr scény

  ```
  <a href="skp:select_scene@Scene 2">Výběr scény</a>
  ```

[6] https://sites.google.com/site/morisdov/




Obr. 2.5: Museum/Gallery HTML Reference – nastavení a referenční HTML soubor

- Změnu viditelnosti vrstev

```
<a href="skp:retrieve_layers@Vrstvy1">Viditelnost vrstev</a>
```

Tyto odkazy tedy může uživatel využít a zapsat do těla dokumentu. Proto, aby bylo možné ovládat přepínání mezi pohledy kamery (více pohledů v rámci jedné scény) a viditelnost vrstev, obsahuje plugin nástroj na ukládání okamžitého pohledu a stavu vrstev. K zobrazení referenčního souboru uvnitř aplikace SketchUp je použito ruby třídy `WebDialog`.

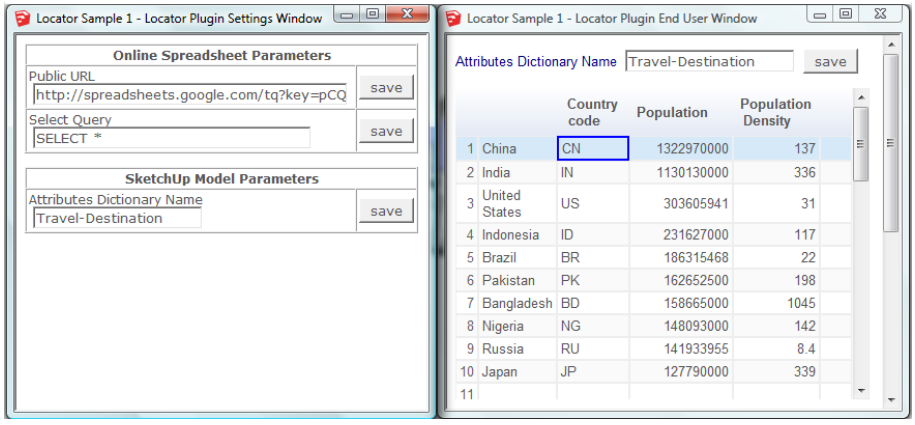
Stejně jako následující plugin General Locator toto rozšíření vytvořil Moris Papasmadov (pod přezdívkou MorisDov) [17]. Pro vyzkoušení funkcí pluginu je možno využít vzorových modelů, které jsou k dispozici ke stažení společně se souborem pluginu.

## 2.6 General Locator

Zajímavým rozšířením, které se bohužel zatím neukázalo jako úplně funkční, je plugin General Locator[7]. Ten by měl propojovat entity v modelu s online tabulkou Dokumentů Google[8].

Pro vytvoření nové tabulky je třeba vlastnit účet Google[9]. Po přihlášení se do služby se nové dokumenty vytvářejí na stránce *Disk*. Údaje o entitách jsou zapisovány do řádků v nové tabulce. První řádek musí obsahovat názvy sloupců a je nutno ho ukotvit (Zobrazit/Ukotvit řádky/Ukotvit 1 řádek) jako hlavičku. V nastavení sdílení

[7] https://sites.google.com/site/morisdov/
[8] http://www.google.com/google-d-s/intl/cs/tour1.html
[9] https://support.google.com/accounts/answer/27441?hl=cs?




Obr. 2.6: Okna pluginu General Locator (otevřen vzorový soubor)

dokumentu je nutno umožnit zobrazování dokumentu všem uživatelům s odkazem a odkaz na tabulku zkopírovat.

V okně nastavení pluginu (Settings Window) se nastaví minimálně URL tabulky a vyhledávací dotaz (výchozí je SELECT *). Tabulka a entity v modelu mohou být propojeny nejlehčeji pomocí názvu objektu (položka *Name* v okně *Entity Info*). Potom musí být jeden ze sloupců tabulky pojmenován právě „Name“. Další možností je propojení pomocí kódu, který je uložen jako atribut příslušné entity. V tabulce potom musí být sloupec odpovídající klíči atributu a v nastavení pluginu je nutno zapsat odpovídající atributový slovník.

Po propojení tabulky a objektů v modelu by mělo být možné v uživatelském okně (End User Window) kliknutím na řádky tabulky (vložená tabulka Google) vybrat zvolenou entitu. Ta je přidána do výběru SketchUpu (zvýrazněna) a přesune se na ní pohled.

Při testování programu bylo postupováno podle návodu [18] uvedeným postupem. Bohužel se nepodařilo docílit toho, aby byl první řádek tabulky pluginem interpretován jako hlavička (proto je první řádek tabulky ukotvován). Tento problém je popsán v návodu k pluginu, popsaná řešení ale problém neodstranila. Plugin tak nebyl schopen rozpoznat jednotlivé sloupce a propojení se nezdařilo. Funkčnost tak mohla být vyzkoušena pouze ve vzorovém souboru dostupném po stažení pluginu, výsledek se ale již nepodařilo zopakovat. Jinak by plugin představoval velice zajímavou možnost uchovávání informací o objektech modelu (v online souboru). Nevýhodou by byla pouze nutnost připojení k internetu a registrace poskytovatele informací (vytvoření účtu Google).

## 2.7 WalkAbout3d

WalkAbout3d[10] je samostatný program, který umožňuje procházení modelu vytvořeného v aplikaci SketchUp. K procházení modelu potom již není aplikace SketchUp zapotřebí. Protože ale formáty, které jsou exportovány programem WalkAbout3d, neumožňují přímou webovou prezentaci, byl program zařazen do této kapitoly. Vyvíjen je společností *Deliverance Software*. Dle domovských stránek používá program herní engine a zprostředkovává tak kvalitnější vjem, než poskytují běžné nástroje SketchUpu [20]. Procházení může fungovat i ve stereo módu, pro takové prohlížení modelu je potom nutno používat anaglyfické brýle. Pro vyzkoušení je k dispozici Trial verze programu, která poskytuje plnou funkčnost na 120 minut.

WalkAbout3d funguje jako samostatný program, pro snadný export modelů z aplikace SketchUp by měl sloužit speciální plugin. Ten během testování programu nefungoval, vzhledem k snadnému importu přímo ve WalkAbout3d to ale nepůsobilo žádné problémy.

Po spuštění programu je nejprve třeba vybrat soubor `skp`, který má být importován. Po načtení souboru je zpřístupněno další nastavení. Je zde možno nastavit výšku osoby (avatar), která bude modelem procházet, její rychlost a podobně. Dále lze nastavovat různé scénáře procházení (Scenarios) s předdefinovaným výchozím bodem, které mohou být mezi sebou propojeny tlačítky na obrazovce. Může se jednat o skutečné procházení nebo o hotové animace, do kterých nemůže uživatel dále zasahovat.

Exportovat hotový soubor pro procházení modelem lze do formátu `wpf`, který je vlastním formátem programu a lze ho prohlížet aplikací WalkAbout3d Viewer. Ta je sice zdarma k dispozici, přesto bude pro většinu uživatelů asi zajímavější exportovat model jako „Single Exe“. Tato funkce vytvoří samostatný spustitelný `exe` soubor, k jehož otevření již není potřeba dalšího prohlížeče.

Pomocí další funkce programu, která již souvisí s doplňujícími informacemi, lze propojovat jednotlivé modely dohromady pomocí odkazů a také odkazovat na webové stránky. K nastavení takovýchto propojení pomáhají i atributy prvků v programu SketchUp. U libovolného prvku stačí ve SketchUpu nastavit následující atributy:

```
Dictionary Name/Attribute name/Attribute value
WA3D/Link/
```

Pro jednoduché nastavení lze použít například výše uvedený Attribute Manager (2.1). Do hodnoty atributu *Link* by bylo možné přímo napsat odkaz na připojovaný

[10] http://www.walkabout3d.com/index.php

soubor, není to ale třeba. Okno pro další nastavení odkazu se totiž zobrazí automaticky v programu WalkAbout3d. V tomto okně lze potom jednoduše nastavit cestu k souboru (již exportovaný `wpf` soubor, který se po propojení stane součástí souboru výsledného). Místo odkazu na jiný model lze vložit také odkaz na webovou stránku obsahující další informace o modelované skutečnosti.

Obr. 2.7: WalkAbout3d – procházka modelu z pohledu třetí osoby

Procházení výsledného exportovaného modelu skutečně připomíná počítačovou hru. Zdmi a pevnými objekty nelze procházet, je možné stoupat po schodech, běhat, případně vyskočit i s odpovídajícími efekty. K dispozici je i pohled třetí osoby (Third-person view), kdy je v záběru kamery i jednoduše modelovaný avatar. Pokud se kurzor ocitne na objektu obsahujícím odkaz, změní se a je možno na odkazovaný soubor přejít. Návrat zpět je potom zajištěn tlačítkem v levém horním rohu obrazovky. Odkazování na webové stránky by mělo fungovat obdobně, během testování se ale neukázalo jako zcela funkční (i když vložený webový prohlížeč po ručním zadání adresy fungoval). Rovněž ovládání (zejm. přístup k ovládacímu panelu s možnostmi procházení) není zpočátku zcela intuitivní a tlačítko pro přechod zpět k hlavnímu modelu od modelu odkazovaného se podařilo najít až po delší době. Jinak ale WalkAbout3d zprostředkovává zajímavý pohled na model, který by ve SketchUpu nebyl možný (např. při uvážení složitosti pohybu uvnitř modelu ve SketchUpu).

# 3 Programovací jazyk Ruby

Jak již bylo řečeno výše, pro vytváření rozšíření aplikace SketchUp je využíváno rozhraní SketchUp Ruby API. Zásuvné moduly jsou tedy programovány v programovacím jazyce Ruby. V následující kapitole proto budou stručně popsány základní poznatky o tomto jazyku, které byly využity při programování vlastního pluginu.

## 3.1 Úvod

Ruby (angl. rubín) je interpretovaný skriptovací, objektově orientovaný programovací jazyk. Jeho autorem je Japonec Yukihiro Matsumoto, první verze byla uveřejněna v roce 1995 [22]. Jeho výhodou by měla být mimo jiné jednoduchá a snadno naučitelná syntaxe, což lze potvrdit i z vlastní zkušenosti. Jednou z výhod pro začínajícího programátora může být například i dynamické typování. U každé proměnné tedy není nutno předem specifikovat datový typ. Protože se jedná o interpretovaný jazyk, je pro spuštění programu nutný interpret (např. u SketchUp Ruby API je zdrojový kód interpretován programem v rámci aplikace SketchUp).

## 3.2 Třídy a metody

Jak již bylo řečeno výše, Ruby umožňuje objektově orientované programování (OOP – Object-oriented programming). Jedná se o skutečně objektově orientovaný jazyk. Vše, s čím je během programování manipulováno, je objekt a výsledkem operací jsou zase objekty [26]. Pro příklad je možné uvést například celá čísla (integer). I ta jsou v ruby objekty a to objekty třídy `Fixnum`. Třídou se rozumí šablona, pomocí níž jsou definovány datové typy. Jednotlivé objekty jsou potom *instancemi* tříd. Operace, které je možno s objekty provádět (metody), jsou definovány uvnitř třídy. Metody představují jediný způsob, jak lze s objektem komunikovat. V následujícím příkladu se vrátíme k celým číslům:

```
a = 5.to_s
```

Pomocí tohoto příkazu je do proměnné `a` vložen `string`, textový řetězec, který byl vytvořen z celého čísla 5. V tomto případě je tedy volána metoda `to_s`, která je zprávou instanci třídy `Fixnum`. Příjemce zprávy (receiver) je v Ruby oddělen od metody tečkou. Metoda `to_s` musí být definována uvnitř třídy `Fixnum`.

Vytvoření nové třídy v Ruby bude ilustrováno následujícím příkladem:

```
class Pes
```

```
        def initialize(plemeno,jmeno)
                @plemeno = plemeno
                @jmeno = jmeno
        end
end
```

V tomto příkladu byla, jak je patrné, vytvořena nová třída `Pes`. V rámci této třídy je definována pouze jedna metoda a to `initialize`. Pojmenování třídy a metody je v rámci zvyklostí – název třídy začíná velkým písmenem, název metody potom písmenem malým. Metoda `initialize` je speciální metodou v rámci Ruby. Při vytvoření nové instance třídy `Pes` je nejprve vytvořen neinicializovaný objekt a poté je zavolána metoda `initialize`, která inicializuje proměnné instance pomocí parametrů, které byly zadány při jejím vytvoření. Nový objekt třídy je vytvořen speciální metodou `new` (konstruktor), tedy:

```
pes1 = Pes.new("pudl","Ludvik")
```

Takto vytvořené třídy nejsou v Ruby nikdy uzavřené. Další metody by tedy bylo možno přidávat do již vytvořené třídy stejným způsobem, přičemž stávající metody by ve třídě zůstaly.

## 3.3 Proměnné

Výše bylo řečeno, že vše v Ruby je objektem. Jedinou výjimku v tomto případě tvoří proměnné. Ty nejsou v Ruby objekty ale pouze odkazy (referencemi) na ně. K podložení tohoto tvrzení může sloužit příklad, který je převzatý z [26]:

```
person1 = "Tim"
person2 = person1
person1[0] = 'J'
person1 #>>     "Jim"
person2 #>>     "Jim"
```

V tomto příkladu byla nejprve vytvořena proměnná `person1`, která je odkazem na objekt typu (třídy) `string`. Proměnné `person2` potom byla přiřazena proměnná `person1`. Ve výsledku tedy obě proměnné odkazují na stejný objekt a změna písmene (nultého znaku řetězce) provedená pro proměnnou `person1` potom změní celý objekt a projeví se i v hodnotě proměnné `person2`. Pokud bychom chtěli vytvořit nový objekt třídy `string`, bylo by nutno použít metodu `dup`, tedy:

```
person2 = person1.dup
```

Lokální proměnné existují pouze v rámci metody nebo třídy. Jejich názvy by měly začínat malým písmenem.

Proměnné instance (instance variables) začínají v Ruby znakem `@`. Jsou to tedy například proměnné `@plemeno` a `@jmeno`, které byly použity v příkladu v předchozí kapitole. Takové proměnné existují pouze v rámci jedné instance třídy a mohou pro každou instanci nabývat jiné hodnoty.

Proměnné třídy (class variables) se vyskytují pouze jednou v rámci jedné třídy (tedy v rámci všech instancí třídy). Jejich názvy začínají v Ruby znaky `@@`. Pokud bychom tedy chtěli například počítat všechny instance třídy `Pes`, mohli bychom tak učinit s pomocí proměnné třídy:

```
class Pes
        @@pocet = 0
        def initialize(plemeno,jmeno)
                @plemeno = plemeno
                @jmeno = jmeno
                @@pocet += 1
                puts @@pocet
        end
end
```

Vždy, když bude pomocí konstruktoru `new` vytvořena nová instance třídy Pes, bude k proměnné třídy `@@pocet` přičtena jednička a poté bude obsah proměnné vypsán.

Globální proměnné existují vždy pouze jedenkrát v rámci programu. Jejich názvy začínají znakem `$`.

## 3.4 Třída Array

Třída `Array` (pole) umožňuje uchovávat kolekci odkazů na objekty. Jedno pole může obsahovat reference na objekty různých typů. Vytvořit pole lze buď explicitně vytvořením nové instance třídy `Array` (`Array.new`) nebo pomocí přiřazení seznamu objektů:

```
array = [1,"a",72]
array[0] #>> 1
array[1] #>> "a"
array[2] #>> 72
array[-1] #>> 72
```

Přistupovat k obsahu pole lze pomocí operátoru `[]`, což je vlastně metoda třídy `Array`. Uvnitř složené závorky se napíše požadovaný index (začíná se od 0). Pokud je index zapsán se záporným znaménkem, počítá se pořadí odzadu (nyní ale od -1). Pro nastavování prvků pole se používá obdobně fungující operátor `[]=`. Přidávat prvky na konec pole lze potom buď metodou `push` nebo i přičtením nového pole pomocí operátoru `+`.

```
array = [1,2,3,4,5]
array.push(6) #>> [1, 2, 3, 4, 5, 6]
array = array + [6] #>> [1, 2, 3, 4, 5, 6]
```

## 3.5 Třída Hash

Hashe se nazývají také asociativní pole nebo slovníky. Prvky takových polí nejsou indexovány po sobě jdoucími celými čísly, ale jiným způsobem, například pomocí textových řetězců. Při ukládání hodnoty do asociativního pole je tedy nutno vždy zadat 2 objekty – klíč a hodnotu. Zadaná hodnota je potom vrácena pro příslušný klíč. Vytvoření hashe je uvedeno v příkladu níže.

```
h = Hash.new
h['jedna'] = 1
h['dva'] = 2
h #>> {"dva"=>2, "jedna"=>1}
```

## 3.6 SketchUp Ruby API

Pro tvorbu rozšíření aplikace SketchUp (dále také v 1.2) jsou v rámci SketchUp Ruby API předdefinovány některé třídy včetně potřebných metod. Pro představu o struktuře Ruby API může sloužit například diagram v [29].

Uživatel, který chce přidat další funkčnost do aplikace SketchUp, může samozřejmě využívat všech možností, které nabízí jazyk Ruby. Metody, které jsou definovány v rámci Ruby API, potom zajišťují komunikaci se SketchUpem. K dispozici tak jsou například metody, pomocí kterých lze otevírat modely, procházet entity modelů, editovat je atd. Dále je také možné přidávat nové nástroje, které zpřístupňují nové funkce SketchUpu dalším uživatelům v rámci jeho grafického rozhraní. Nástroje jsou uživatelem vytvářeny v rámci připraveného rozhraní (pravděpodobně) jako odvozené třídy ze třídy Ruby API `Tool`. Takové třídy potom dědí metody naprogramované v rámci Ruby API a uživatel může definovat metody další. U tříd

nástrojů jsou potom tyto metody volány při určitých událostech v rámci SketchUpu. Metoda `onLButtonDown` je tak například volána, pokud uživatel klikne levým tlačítkem myši.

V Ruby API jsou dále upraveny i některé standardní třídy Ruby tak, aby byly vhodné pro užití v rámci SketchUpu. Takovým případem jsou třídy `Array` a `String`. Třída `AttributeDictionary`, která slouží pro přidávání atributů objektům modelu (viz 1.3), se potom chová podobně jako asociativní pole, tedy třída `Hash`, neboť jsou zde v rámci slovníků uchovávány atributy systémem klíč/hodnota.

# 4 Popis modulu TIS

Zásuvný modul TIS vznikl s využitím poznatků o SketchUp Ruby API a po testování výše uvedených pluginů (viz 2). Cílem bylo vytvořit modul, který bude možné použít například pro nastavování a prohlížení dalších informací o zajímavých částech památkových objektů. U některých funkcí bylo možno využít poznatků z uvedeného testování pluginů a nechat se inspirovat jejich zajímavými vlastnostmi. Vzhledem k tomu, že jsou tyto pluginy k dispozici pouze v zašifrované podobě (`rbs` soubory), bylo ale vždy nutno navrhnout vlastní programové řešení.

## 4.1 Základní vlastnosti

Pro uchovávání dalších informací o entitách modelů je využíváno možnosti přidávat každému objektu modelu v programu SketchUp atributy. V kapitole o testovaných pluginech již byly popsány některé programy, které nastavování atributů umožňují. Dále lze samozřejmě nastavovat atributy i pomocí Ruby Console. To lze ale těžko nazvat uživatelsky přívětivým způsobem. Cílem návrhu pluginu je umožnit nejen atributy nastavit, ale také je přehlednou formou prohlížet. Vhodným modelem pro použití pluginu může být, jak bylo řečeno výše, například model památkového objektu. U každé části modelu je možno nastavit název, popis a také obrázek objektu ve skutečnosti. Například u soch se také může hodit možnost přidat odkaz na podrobněji zpracovaný model, který by třeba vzhledem ke své velikosti nemohl být součástí souboru modelu základního.

Program obsahuje tři základní nástroje, které může uživatel použít pro nastavení atributů/přidání dalších informací o objektech modelu a jejich zobrazení. Tyto nástroje jsou dostupné z nového panelu nástrojů, který se, je-li plugin správně nainstalován, zobrazí při spuštění aplikace SketchUp.

Prvním nástrojem na panelu je nástroj *Výběr*, pomocí kterého lze zobrazovat další informace o objektech. Nástroj *Tabulka* potom zobrazí souhrnný přehled objektů modelu s přidanými dalšími informacemi. Poslední nástroj na panelu, tedy nástroj *Nastavení*, je určen pro nastavování atributů.

Všechny nástroje jsou také dostupné z nového submenu *TIS* v rámci menu SketchUpu *Plugins*. Pouze z tohoto submenu je potom dostupný poslední nástroj *Přehled atributů*, který zobrazí všechny atributy objektu (tedy i ty nastavené jiným způsobem v rámci cizího slovníku) tak, jak jsou uloženy, tedy způsobem slovník/-klíč/hodnota. Tento nástroj je možno používat pro kontrolu nastavených atributů

a nepředpokládá se, že bude uživateli běžně spouštěn. Proto není dostupný z panelu nástrojů. Ze submenu *TIS* je kromě spuštění uvedených nástrojů možno také otevřít návod k pluginu ve formátu `pdf` (viz příloha A) a zobrazit základní informace o pluginu. Aby uživatel věděl, že pracuje se speciálními nástroji pluginu, mají nástroje *Výběr* a *Nastavení* vlastní kurzory.

Pomocí nástroje *Nastavení* je s využitím standardního rozhraní SketchUpu `input box` možno nastavit objektu název, přiřadit mu obrázek, samostatný detailnější model a popis. Dále je nastavován výchozí pohled kamery na objekt, který se využije v dalším nástroji *Tabulka*. Vše je nastavováno po kliknutí na vybraný objekt. Název je nastavován jako textový řetězec, který bude zobrazen v okně dostupném po vybrání objektu nástrojem *Výběr*. Pro obrázek, model a popis je nastavována pouze relativní cesta k samostatnému souboru obrázku, modelu `skp` a k textovému souboru s popisem objektu (relativní vzhledem k modelu, v rámci něhož jsou atributy nastavovány).

Nástroj *Výběr* umožňuje prohlížet informace prostřednictvím HTML stránky, vygenerované ze zadaných údajů v okně, které je vytvořeno třídou `WebDialog`. Při procházení modelu se pod kurzorem zvýrazňují objekty obsahující přidané atributy. Další informace se zobrazí po kliknutí na vybraný objekt. Se stisknutou klávesou `Control` je možno všechny objekty s dalšími informacemi zvýraznit.

Souhrnná tabulka, která je dostupná jako nástroj *Tabulka*, poskytuje přehled o objektech s nastavenými dalšími informacemi. Zobrazuje také, jaké informace jsou přiřazeny (je-li dostupný obrázek, popis, případně detailnější model). Jedná se opět o HTML stránku zobrazovanou pomocí `WebDialogu`. Součástí tabulky jsou i odkazy na jednotlivé objekty, kdy textem odkazu je název objektu. Po kliknutí na odkaz se na vybraný objekt přesune pohled kamery (proto je v nástroji *Nastavení* nastavován) a zobrazí se stránka s dalšími informacemi, stejně jako po kliknutí na objekt s aktivním nástrojem *Výběr*.

Přidávat další informace lze všem objektům modelu. Pokud je objektem komponenta, lze přidávat atributy i jednotlivým entitám v rámci komponenty (to platí stejně i pro skupiny). V souhrnném zobrazení jsou potom uvedeny všechny objekty s atributy v rámci modelu, pokud je otevřena některá komponenta pro editaci, tak jsou zobrazeny všechny objekty s dalšími informacemi v rámci komponenty. Přitom je vždy uveden nadřazený objekt pro danou entitu (tedy je-li přehled vytvořen v rámci hlavního modelu, komponenty atd.). Je tak umožněna vlastně volba různé podrobnosti pohledu na model (viz dále 4.7).

## 4.2 Panel nástrojů, menu, kurzory a WebDialogy

Aby bylo možné zpřístupnit naprogramované funkce uživateli, bylo nutno nejprve vytvořit nové položky v menu a dále také nový panel nástrojů. V ukázce kódu 4.1 uvedené níže je nejprve vytvořeno nové submenu v rámci menu *Plugins*. Do něj jsou potom vloženy odkazy na nové funkce, které plugin přidává. Pokud uživatel klikne na název funkce v menu, bude pomocí konstruktoru `new` vytvořena nová instance např. třídy `Vyber`. Pomocí metody `select_tool` aktivního modelu pak bude nástroj `Vyber` vybrán jako aktivní nástroj SketchUpu.

Ukázka kódu 4.1: Vytvoření nového submenu

```
22 #Nové submenu a položky v něm
23 nove_submenu = UI.menu("Plugins").add_submenu("TIS")
24 vyber = nove_submenu.add_item("Výběr") {
25 Sketchup.active_model.select_tool( Vyber.new )}
```

Tvorba nového panelu nástrojů začala vytvořením vlastního `Toolbaru`. Dále byly vytvořeny nové příkazy (`Command`), ty byly přidány na panel nástrojů a panel byl poté zobrazen, jak to je uvedeno v příkladu 4.2 níže. Ikony, které se zobrazí jako tlačítka na panelu nástrojů, byly vytvořeny v grafickém editoru GIMP[1].

Ukázka kódu 4.2: Nový panel nástrojů

```
60 panel_nastroju = UI::Toolbar.new "TIS"
61 nastaveni_command = UI::Command.new("Nastavení") {
62 Sketchup.active_model.select_tool( Nastaveni.new )}
63 nastaveni_command.large_icon = "nastaveni_24.png"
64 nastaveni_command.small_icon = "nastaveni.png"
65 nastaveni_command.tooltip = "Nastavení" #ve žlutém rámečku při vybírání nástroje
66 nastaveni_command.status_bar_text = "Nastavení atributů objektu" #ve stavovém řádku
67
68 panel_nastroju.add_separator #oddělovač na panelu nástrojů
69 panel_nastroju.add_item nastaveni_command #přidání tlačítka na panel nástrojů
70 panel_nastroju.show #zobrazení panelu nástrojů
```

Nové kurzory byly vytvořeny také v programu GIMP. Kurzory byly namalovány zvlášť pro funkce *Výběr* a *Nastavení*. *Tabulka* ze své podstaty kurzor nepotřebuje. Proměnná, do které je vložen odkaz na nový objekt kurzoru, bude použita pro volání kurzoru v rámci příslušného nástroje.

Ukázka kódu 4.3: Vytvoření nových kurzorů

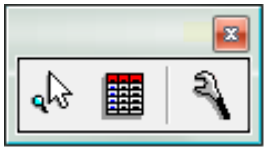
```
93 cursor_path = Sketchup.find_support_file("kurzor.png", "Plugins/TIS/") #soubor png
       kurzoru
94 @@cursor_vyber_id = UI.create_cursor(cursor_path, 10, 2) #cisla jsou souradnice
       vztazneho bodu (hotpointu) kurzoru
```

[1] http://www.gimp.org

Hned na začátku kódu programu byly též vytvořeny potřebné objekty třídy `WebDialog`. Bylo zjištěno, že se u dialogu, u kterého je umožněno měnit velikost, tato velikost ukládá, stejně jako poslední pozice okna. Při opětovném otevření dialogu se okno zobrazí v poslední známé konfiguraci. Proto, aby se `WebDialog` otevíral vždy se stejnými rozměry a na stejném místě, je nutno ještě jednou rozměry a pozici nastavit pomocí metod `set_size` a `set_position`. V rámci pluginu byly vytvořeny tři `WebDialogy` – pro zobrazení dalších informací, pro zobrazení tabulky a pro nastavení kamery. Třída `WebDialog` zobrazuje HTML stránky v okně v rámci programu SketchUp. Pro zobrazení stránky je na PC s OS Microsoft Windows využíván vždy prohlížeč Internet Explorer (bez ohledu na výchozí webový prohlížeč uživatele).

Ukázka kódu 4.4: Nový WebDialog

```
101 @@wd_podrobnosti=UI::WebDialog.new(
102 "Podrobnosti o objektu", true, "",  #název, scrollable?
103 360, 700, 700, 100, true )          #rozměry a pozice, resizable?
104 @@wd_podrobnosti.set_size(360,700)
105 @@wd_podrobnosti.set_position(700,100)
```



Obr. 4.1: Nový panel nástrojů – nástroje *Výběr*, *Tabulka* a *Nastavení*

## 4.3 Nástroj Výběr

Nástroj *Výběr* byl vytvořen pomocí Ruby API rozhraní `Tool`. Jedná se o novou třídu, která má nově definováno 5 metod. Metoda `onSetCursor` je použita pro nastavení nového kurzoru pro nástroj. Pokud se myš pohybuje, je volána metoda `onMouseMove`. V rámci té již probíhá výběr prvků modelu a je zjišťováno, má-li daný prvek přiřazen atributy. Pokud má a jsou-li to atributy přiřazené popisovaným pluginem, zvýrazní se prvek pomocí přidání do selekce programu SketchUp. Je nutno poznamenat, že byla hledána výhodnější možnost zvýrazňování prvků pod kurzorem myši, při které by objekt nemusel být přímo přidáván do selekce. Bohužel ale nebyla nalezena. Dále je přítomnost dalších informací v objektu zdůrazněna zprávou v rámečku vedle kurzoru myši a ve stavovém řádku. Pro vlastní vybírání objektů pod kurzorem myši je používána třída `PickHelper`. Nejprve je nutno aktivovat `pick_helper` v aktivním pohledu SketchUpu, poté je proveden vlastní výběr v místě

o daných souřadnicích v rámci pohledu metodou `do_pick`. Nakonec je použita metoda `best_picked`, která vrátí entitu modelu, která je programem považována za „nejlepší“ výběr, tedy takovou entitu, která by byla vybrána při použití standardního nástroje SketchUpu `SelectionTool`.

Ukázka kódu 4.5: Nástroj *Výběr*, metoda volaná při pohybu kurzoru myši

```
162 def onMouseMove(flags, x, y, view) #pokud se myš pohybuje
163 Sketchup.set_status_text "Ctrl + levé tlačítko = zvýraznění objektů s atributy",
    SB_PROMPT
164 Sketchup.active_model.selection.clear #vymazeme dosavadni vyber
165
166 #Výběr
167 ph = view.pick_helper
168 ph.do_pick x,y
169 best = ph.best_picked
170
171 if best != nil #pokud bylo něco vybráno
172         slovniky = best.attribute_dictionaries #všechny slovníky entity
173
174         if slovniky != nil #pokud má objekt přiřazeny atributy
175                 if slovniky["SlovnikTIS"] != nil #v našem slovníku
176                         slovnik = slovniky["SlovnikTIS"] #vybereme slovník
177                         view.tooltip = "Tento objekt obsahuje další informace" #
                            zobrazíme text vedle kurzoru
178                         Sketchup.set_status_text "Kliknutím na objekt získáte další
                             informace", SB_PROMPT #zobrazíme nápovědu ve stavovém
                            řádku
179                         Sketchup.active_model.selection.toggle(best) #přidání do
                            selekce, zvýraznění
180                 end #if slovniky["SlovnikTIS"] != nil
181         end #if slovniky != nil
182 end  #if best != nil
183
184 end #onMouseMove(flags, x, y, view)
```

Podobně jako v příkladu 4.5 vypadá i začátek definice metody `onLButtonDown`. Po kliknutí levým tlačítkem je nejprve rozhodováno, je-li zmáčknuta klávesa `Control`. To kontrolují další dvě metody třídy, tj. metody `onKeyDown` a `onKeyUp`. Dále byla definována proměnná, která je po zmáčknutí uvedené klávesy nastavena na hodnotu `true` a naopak po puštění klávesy na `false`. Při kliknutí se současně zmáčknutou klávesou `Control` se projdou všechny entity modelu (či komponenty nebo skupiny) a je zjišťováno, zda mají přiřazeny atributy. Pokud ano, jsou všechny zvýrazněny přidáním do selekce. Způsob procházení všech entit modelu či komponenty nebo skupiny bude popsán v kapitole, která popisuje nástroj *Tabulka* (viz 4.5).

Pokud uživatel kliknul na vybraný objekt a není zmáčknuta klávesa `Control`, je opět zjišťováno, zda jsou dostupné další informace o objektu. Dále je nutno určit, zda je k dispozici název objektu a cesta k obrázku, podrobnějšímu modelu a textovému

souboru s popisem. Nejsložitější situace nastává v případě modelu, a proto je uveden v příkladu 4.6 níže.

Ukázka kódu 4.6: Nástroj *Výběr*, klíč Model

```
220 if slovnik["Model"] != nil
221         plugin_path = File.dirname(__FILE__)
222         mod = File.dirname(Sketchup.active_model.path)+"/"+slovnik["Model"]
223         if File.file?(mod) == true #pokud existuje soubor s modelem
224                 model = '<a href="skp:soubor@'+slovnik["Model"]+'"><img src="'+
                        plugin_path+'/pokus_vyber_files/logo.png" alt="Obrazek" height
                        ="50px"  border="0"><br>Klikněte pro otevření samostatného
                        modelu</a><hr>'
225         else
226                 model = '<center><b><font color="red">Nenalezen soubor samostatného
                         modelu!</b></font></center><hr>'
227         end #if File.file?(mod) == true
228 else
229         model = ''
230 end #if slovnik["Model"] != nil
```

Jak je patrné z příkladu, nejprve je zjišťováno, zda je ve slovníku klíč `Model`. Pokud ano, je nalezena cesta k adresáři, v němž se nachází plugin. To bude potřeba k zobrazení ikony SketchUpu, která se nachází v adresáři s potřebnými soubory pluginu. Tato ikona bude použita jako součást odkazu na podrobnější model. Dále je do proměnné `mod` vložena cesta k modelu, která se skládá z cesty k aktivnímu modelu, v rámci kterého se uživatel nachází a z relativní cesty k modelu podrobnějšímu, která je hodnotou pro klíč `Model`.

Dále je třeba zkontrolovat, zda odkazovaný soubor existuje. Kontrola je provedena pomocí metody `file?`. Pokud je vrácena hodnota `true`, je jako textový řetězec vytvořena část výsledné HTML stránky, v níž budou uživateli zobrazeny další informace. Textový řetězec je poté vložen do proměnné `model`. Jedná se vlastně o hypertextový odkaz (anchor), který odkazuje na soubor podrobnějšího modelu.

Pokud soubor podrobnějšího modelu neexistuje (byl chybně zadán), bude příslušnou částí HTML stránky červená varovná hláška. Pokud není žádný podrobnější model k dispozici (tedy nebyl zadán vůbec), je v proměnné `model` pouze prázdný textový řetězec a v rámci výsledné HTML stránky se nic nezobrazí.

Po projití všech atributů objektu je nutno zjistit, zda se nejedná o komponentu či skupinu. Pokud ano, může tato komponenta/skupina obsahovat další objekty s atributy uvnitř. Proto je nutno projít všechny entity uvnitř komponenty/skupiny (opět viz 4.5). Pokud jsou nalezeny objekty s dalšími informacemi, je zjištěn jejich název. Pokud nejsou pojmenovány, uloží se řetězec 'nepojmenováno'. Do proměnné `objekty_uvnitr_text` je vložen textový řetězec, který informuje uživatele o názvech objektů s atributy v rámci komponenty či skupiny. Pokud nebyl žádný takový objekt nalezen, bude textový řetězec prázdný.

Základ HTML stránky, která se zobrazí ve `WebDialogu`, je také vytvořen jako textový řetězec v rámci skriptu pluginu. To umožňuje jednoduše měnit části stránky a reagovat tak na nastavené atributy. Kromě proměnné `model` ještě o vzhledu stránky podobným způsobem rozhodují proměnné `nazev` (zobrazí se na stránce jako nadpis), `obrazek` (zobrazí nastavený obrázek, který je zároveň odkazem na obr. ve skutečné velikosti) `objekty_uvnitr_text` (obsahuje informace o objektech s atributy uvnitř komponenty/skupiny) a `popis` (načte popis z textového souboru a zobrazí ho včetně případného HTML formátování). Pomocí metody `set_html` je provedeno nastavení stránky do příslušného `WebDialogu`. Pokud je již zobrazen, dojde pouze ke změně stránky a zobrazení nových informací, jinak ho je nutno zobrazit pomocí metody `show`.

Původním záměrem bylo použít v rámci vygenerované HTML stránky běžné odkazy, které odkazují na obrázek či model pomocí zadané relativní cesty. Po kliknutí na odkazy uživatelem by se obrázek otevřel v novém okně ve stejném prohlížeči, jaký je použit na pozadí `WebDialogu`, tedy v prohlížeči Internet Explorer. Model by byl také otevírán přes webový prohlížeč stejným způsobem jako při stahování souborů umístěných na webu. Protože Internet Explorer samozřejmě neumí `skp` soubor otevřít, byl by po „stažení“ pro otevření vybrán program SketchUp.

Pro uživatele může být při prohlížení obrázků samozřejmě příjemnější, když se tyto otevřou v programu, který má pro daný účel uživatel nastaven. Proto byly dosavadní běžné odkazy zaměněny za odkazy ve speciálním tvaru, které volají funkci uvnitř Ruby pluginu. Pomocí ní jsou potom soubory otevírány metodou `openURL`. Vlastní otevírání souborů tedy již není řešeno přímo prohlížečem, ale probíhá až v rámci SketchUpu. Uvedená metoda potom otevře vybraný soubor ve výchozím programu pro daný typ souboru. Obrázek je tak otevřen ve výchozím programu pro prohlížení bitmap s daným formátem, model je přímo otevřen ve SketchUpu. Komunikace Ruby a `WebDialogu` pomocí metody `add_action_callback` je dále popsána v další kapitole (viz 4.4).

Ukázka kódu 4.7: Základ HTML stránky vytvořený pro WebDialog nástroje *Výběr*

```
297 html= '<!DOCTYPE html>
298 <html>
299         <head>
300                 <meta http-equiv="Content-Language" content="cs">
301                 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
302         </head>
303         <body>'+nazev+'<center>'+
304         obrazek+'</center>'+
305         objekty_uvnitr_text+'<center>'+
306         model+'</center>'+
307         popis+
308         '</body>
```

```
309 </html>'
310
311 @@wd_podrobnosti.set_html(html)
312
313 if @@wd_podrobnosti.visible? == false
314         @@wd_podrobnosti.show()
315 end #if @@wd_podrobnosti.visible?
```

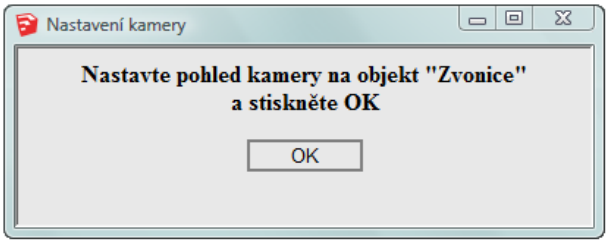
## 4.4 Nástroj Nastavení

Pomocí nástroje *Nastavení* lze, jak již bylo řečeno výše, nastavovat atributy objektů v modelu. Obrázek, podrobný model a popis jsou samostatnými soubory, které se nacházejí vedle souboru hlavního modelu tak, aby bylo možné k nim jednoduše zadat relativní cestu. Popis je textový soubor, který s ohledem na použité kódování pluginu *musí* být uložen v kódování Unicode UTF-8. To ale není problém zajistit ani v běžném Poznámkovém bloku (Notepad v OS Windows).

Nástroj je opět vytvořen jako nová třída. První tři metody této třídy, tedy metody `onSetCursor`, `activate` a `resume`, slouží k nastavení kurzoru a zobrazení nápovědy ve stavovém řádku při aktivaci nástroje, resp. při obnovení činnosti nástroje (pokud byl mezitím používán jiný nástroj, obvykle pro nastavení pohledu). Další metodou je `onLButtonDown`, tedy metoda, která je volána při kliknutí levým tlačítkem myši.

Poslední funkce `nastaveniKamery` je definována uvnitř metody `onLButtonDown`. Je použita pouze uvnitř této metody a je vytvořena jen proto, aby se nemusela dotyčná část kódu zbytečně opakovat. Použití kamery nastavené pomocí této funkce bude popsáno dále (viz 4.5). Nyní bude vysvětleno jen samotné nastavení kamery. Pokaždé, když uživatel provede nastavení nebo změnu atributů objektu, se zobrazí nový `WebDialog`. Uživatel dle pokynu v něm nastaví pohled na vybraný objekt pomocí běžných nástrojů SketchUpu. Po zmáčknutí tlačítka *OK* je kamera nastavena. `WebDialog` je v tomto případě používán pouze z toho důvodu, že běžná dialogová okna SketchUpu jsou modální, tedy neumožňují žádnou akci v rámci SketchUpu (tedy např. nastavení pohledu), dokud nejsou uzavřena.

Proto, aby dialog pro nastavení kamery fungoval, bylo poprvé v tomto pluginu využito metody `add_action_callback`. Tato metoda zprostředkuje komunikaci mezi Ruby API a HTML stránkou. Je tak vytvořena metoda, kterou může zavolat např. JavaScript uvnitř HTML stránky tak, že nastaví tzv. „fake URL“ do adresního řádku prohlížeče. Ta vypadá např. následovně `skp:nazev_callbacku@zprava`, kde `zprava` je textový řetězec, který je posílán Ruby (dle dokumentace max. 2038 znaků). Callback pro nastavení kamery tedy vypadá následovně:




Obr. 4.2: Okno zobrazené při nastavování pohledu kamery na objekt

Ukázka kódu 4.8: Nástroj *Nastavení*, callback pro nastavení kamery

```
420 @@wd_kamera.add_action_callback("kamera") do |js_wd, zprava|
421         kamera = Sketchup.active_model.active_view.camera
422         eye = kamera.eye
423         target = kamera.target
424         up = kamera.up
425         kamera_vektor = [eye,target,up]
426         best.set_attribute "SlovnikTIS","Kamera",kamera_vektor
427 
428         @@wd_kamera.close
429 end #@@wd_kamera.add_action_callback
```

Do proměnné `kamera` je tedy vložen odkaz na kameru, která je v daný okamžik v aktivním pohledu SketchUpu nastavena. Dále je nutno zjistit polohu kamery a polohu cíle pomocí metod `eye`, `target` a `up`. První dvě metody vrátí objekt třídy `Point3d`, třetí potom třídy `Vector3d`, tedy ve všech případech vlastně pole. Všechna tři pole jsou potom vložena jako vnořená do pole třetího, které je nastaveno jako hodnota atributu `Kamera`. S proměnnou `zprava`, do které je vložen obsah zprávy přijaté od JavaScriptu, není již třeba pracovat, protože jediná hodnota, která může být přijata, je v tomto případě `ok`. Část HTML stránky, která se zobrazí ve `WebDialogu`, bude vypadat následovně:

Ukázka kódu 4.9: Část WebDialogu pro nastavení kamery

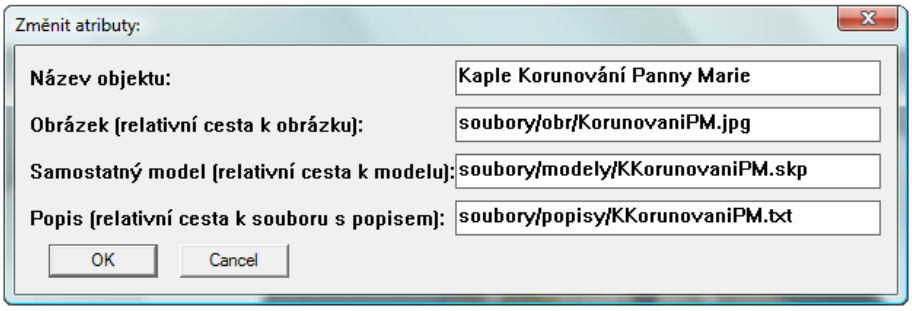
```
455 <button style="border: 2px solid gray;background-color: rgb(232,232,232)";
456 onclick="tlacitko()">OK</button>
457 
458 <script type="text/javascript">
459         function tlacitko() {window.location.href = "skp:kamera@ok"}
460 </script>
```

Ve výsledném dialogovém okně bude tedy pouze jedno tlačítko, které po stisknutí zavolá funkci JavaScriptu. Ta do adresního řádku vloží zprávu pro Ruby. Zpráva může být v tomto případě jakákoliv, kamera bude nastavena prostě po jejím obdržení.

Dále již k samotnému nastavení atributů. Protože i u nástroje *Nastavení* je nutno provést výběr objektu, je použito stejného postupu jako u nástroje *Výběr*. Po kliknutí levým tlačítkem na zvolený objekt se tak provede vybrání objektu. Dále

je opět nutno zjistit, zda má objekt přiřazeny atributy. Pokud ano, musí mít nastaven i výchozí pohled kamery. Ten je tedy získán z atributů nastavených dle popisu výše opačným postupem a poté nastaven. Dále se postupně zjišťuje, zda již má objekt přiřazen název, obrázek, model a popis. Tyto hodnoty budou zobrazeny jako výchozí v následujícím dialogu.

Pro nastavování atributů bylo nejprve zvažováno použití standardního okna SketchUpu, tedy rozhraní `UI::inputbox` s tím, že popis bude zadáván přímo do jednoho ze vstupních polí okna. Nebylo by tedy nutno ukládat samostatný soubor s popisem, i když uživatel by nejspíš stejně musel popis vytvářet mimo SketchUp v textovém souboru a pak ho teprve zkopírovat, protože vstupní řádek je pro psaní delších textů příliš nepřehledný. Bylo ale zjištěno, že se `inputbox` textu přizpůsobuje. Tedy že se vstupní pole „natahuje“ podle jeho výchozí hodnoty a s ním i celý dialog. Při opětovné editaci delšího popisu by bylo vstupní pole (s dlouhým textem v jednom řádku) a tím i celý `inputbox` tak dlouhý, že by se zdaleka nevešel na obrazovku. Editace by tak sice byla možná, ale pro uživatele jistě velmi nepříjemná.




Obr. 4.3: Dialogové okno pro nastavení dalších informací

Aby se předešlo problémům s `inputboxem`, byl zamýšlen jiný postup. Pro zadávání údajů byl vytvořen nový `WebDialog`, kde by se atributy zadávaly prostřednictvím HTML formuláře (tag `form`). Popis by tak bylo možné zadávat komfortněji například pomocí rozměrnějšího vstupního pole (tag `textarea`). Dále by na stránce byla tlačítka a JavaScript, který by umožnil předávání hodnot Ruby stejným způsobem, jaký byl předveden výše v dialogu pro nastavení kamery. Samotný `WebDialog` by mohl vypadat velmi podobně jako standardní `inputbox` SketchUpu s tlačítky OK a Cancel. Pokud by pro jeho zobrazení byla použita metoda `show_modal`, byly by také znemožněny další akce ve SketchUpu před vyplněním dialogu, což by v tomto případě mohlo být výhodné. Při testování však byl bohužel zjištěn problém v komunikaci mezi `WebDialogem` a Ruby. I když byly zdrojový kód pluginu i v něm generovaná HTML stránka ve stejném kódování, nebylo možné zaslat Ruby ve zprávě české znaky.

Protože se bez českých znaků zejména v popisu samozřejmě nebylo možno obejít, bylo nakonec rozhodnuto použít standardní dialog SketchUpu, ale popis uchovávat mimo SketchUp v samostatném souboru, jak již bylo výše uvedeno. To může být nakonec i příjemnější pro uživatele. Protože je celý plugin včetně v něm zapsaných koster HTML stránek v kódování Unicode UTF-8, musí být v tomto kódování uložen i textový soubor popisu. Do dialogu je tedy nakonec uživatelem vkládána pouze relativní cesta k příslušnému souboru, stejně jako tomu je u obrázku a samostatného modelu.

Ukázka kódu 4.10: Nástroj *Nastavení*, dialog pro nastavení atributů

```
524 prompts = ["Název objektu:","Obrázek (relativní cesta k obrázku):","Samostatný
    model (relativní cesta k modelu):","Popis (relativní cesta k souboru s popisem)
    :"]
525 defaults = [nazev,obrazek,model,popis]
526 input = UI.inputbox prompts, defaults, "Změnit atributy:"
527 if input[0] != "" && input[0] != slovnik["Nazev"] #pouze pokud změníme název
528         best.set_attribute "SlovnikTIS", "Nazev", input[0]
529 elseif input[0] == "" && slovnik["Nazev"] != nil #pouze pokud vymažeme název
530         slovnik.delete_key("Nazev")
531 end #if input[0]
```

V příkladu 4.10 nahoře je nejprve vytvořeno pole obsahující popisky, které budou uvedeny u jednotlivých vstupních polí dialogu. Do proměnné `defaults` je potom vloženo další pole se současnými hodnotami atributů, které budou zadány jako výchozí. Obsah vstupních polí `inputboxu` bude po zmáčknutí tlačítka *OK* uložen do pole, na které odkazuje proměnná `input`. Pokud bylo něco vloženo a příslušný atribut byl opravdu změněn, bude provedeno nové nastavení atributu. Pokud je vstupní pole prázdné a atribut přitom dříve existoval, je jasné, že uživatel jeho hodnotu vymazal. Je proto provedeno vymazání příslušného klíče. Stejně jako v případě názvu je samozřejmě postupováno i v případě obrázku, modelu a popisu.

Pokud jsou všechna pole prázdná, vymaže se celý slovník. Tak jsou vymazány i pro uživatele neviditelné atributy obsahující pohled kamery, které by již byly uchovávány zbytečně. Pokud ne, znamená to, že zůstal nějaký atribut uložen. Je proto zavolána funkce pro nastavení kamery a uživatel může toto nastavení změnit.

Pokud je ve chvíli vymazání atributů otevřena souhrnná tabulka, je nástroj *Tabulka* opět vybrán. Dosavadní přehled objektů s atributy je tedy obnoven a může okamžitě zobrazit provedené změny. Pokud dojde ke změně atributů, je obnovení tabulky řešeno až po nastavení pohledu kamery.

Ukázka kódu 4.11: Nástroj *Nastavení*, vymazání slovníku/nastavení kamery

```
551 #pokud vymažeme vsechny atributy ze slovníku, vymažeme i slovník
552 if input[0]=="" && input[1]=="" && input[2]=="" && input[3]==""
553         slovniky.delete 'SlovnikTIS'
```

```
554             if      @@wd_tabulka.visible? == true #pokud je vidět souhrnný přehled,
                    obnovíme tabulku
555                 Sketchup.active_model.select_tool(Tabulka.new)
556             end
557     else
558             #Nastavení kamery
559             nastaveniKamery(best)
560     end #if input[0]==""&&if input[1]==""&&input[2]==""&&input[3]==""
```

Pokud objekt zatím nemá žádné další informace přiřazeny (tedy jde o první nastavení a ne o změnu atributů), je situace obdobná. V `inputboxu` ale samozřejmě nebudou zobrazeny žádné výchozí hodnoty. Pokud uživatel vytvoří alespoň jeden z atributů, pokračuje se opět nastavením kamery. Pro případ, že by uživatel zavřel okno nastavení kamery nebo by na něj, vzhledem k tomu, že není modální, zapomněl a ponechal ho otevřené, nastaví se kamera nejprve automaticky podle aktuálního pohledu. Tak je vždy zajištěno nastavení alespoň nějakého výchozího pohledu na objekt. Kromě toho je pravděpodobné, že se uživatel při nastavování atributů objektu na objekt dívá a předtím na něj musel i kliknout a vybrat ho. Proto by se nemělo jednat o výchozí pohled zcela nesmyslný (tedy s neviditelným objektem zájmu). Další nastavení kamery probíhá již s otevřeným příslušným `WebDialogem` stejně jako v předchozím případě při změně atributů.

Na závěr rozboru nástroje *Nastavení* je nutné ještě napsat několik slov o formátování popisu pomocí HTML tagů. Vzhledem k tomu, že je popis textovým řetězcem, který bude vložen do kostry HTML stránky také vytvořené v rámci Ruby skriptu, je možné v něm používat jakékoliv HTML formátování. Je tak možné např. nastavovat písmo, vytvářet odstavce ale i přidávat do textu další odkazy. Vše, co bude do popisu zapsáno, by mělo být zobrazeno webovým prohlížečem v rámci `WebDialogu`. Použité HTML tagy nejsou uvnitř Ruby pluginu nijak kontrolovány. Vzhledem k tomu, že je model prohlížen v rámci SketchUpu pouze lokálně, nemělo by to představovat žádné bezpečnostní riziko. Je tedy pouze na nastavujícím uživateli, jaké formátování v rámci popisu použije a zda bude vytvořený popis s tímto formátováním funkční a vhodný pro následné zobrazování.

Během testování pluginu bylo dále zjištěno, že se HTML stránka vygenerovaná pluginem a zobrazená ve `WebDialogu` nachází v adresáři dočasných souborů na počítači uživatele. Při zobrazení tedy samozřejmě není stránka ve stejném adresáři jako model SketchUpu, ve kterém byla otevřena. Pokud tedy uživatel v rámci popisu použije hypertextové odkazy, které odkazují relativně na soubory umístěné vedle souboru modelu, ohlásí prohlížeč chybu, protože nemůže uvedené soubory najít. Tento problém byl vyřešen vytvořením nového callbacku `soubor` pro dialog nástroje

*Výběr*. Uživatel, který vytváří popis, může využít odkazu ve tvaru `<a href="skp:soubor@adresa">`, kde místo `adresa` zapíše relativní cestu k požadovanému souboru. Pomocí tohoto odkazu je tak pouze předána zpráva Ruby API. Předaný soubor je potom otevřen pomocí metody `UI.openURL` ve výchozím programu pro daný typ souboru. Takto lze odkazovat na všechny soubory, pro jejichž otevření bude mít uživatel, který si další informace prohlíží, vhodný program. Odkazy s absolutní adresou, které odkazují na soubory umístěné na webu, budou fungovat bez problémů, protože samozřejmě nezávisí na umístění HTML stránky ve `WebDialogu`.

## 4.5 Nástroj Tabulka

Nástroj *Tabulka* přehledně zobrazuje všechny objekty, které mají přiřazeny další informace. Jak již bylo řečeno výše, záleží vždy na tom, kde se uživatel v modelu nachází. Pokud není žádná komponenta ani skupina modelu otevřena pro editaci, zobrazí se v tabulce všechny objekty v rámci modelu. Objekty s atributy, které se nacházejí uvnitř komponenty nebo skupiny se zobrazí po otevření komponenty/skupiny pro editaci. Dále se pro lepší orientaci zobrazí i třída nadřazeného objektu a případně název, má-li ho nadřazený objekt přiřazen v rámci dalších informací vytvořených tímto pluginem.


Obr. 4.4: Okno s podrobnostmi o objektu a souhrnná tabulka

Nástroj *Tabulka* má definovánu pouze jednu metodu a to metodu `activate`. Ta je volána, pokud uživatel tento nástroj vybere. Po aktivaci nástroje je nutno nejprve zjistit, kde v modelu se uživatel nachází. To zajišťuje metoda `active_path`. Tato metoda vrátí pole, které obsahuje postupně všechny nadřazené objekty, v nichž se uživatel nachází, tedy ty, které předtím dvojklikem otevřel pro editaci. Tudíž pokud uživatel nejprve otevřel pro editaci komponentu a v ní potom skupinu, bude pole vypadat následovně: `[<Sketchup::ComponentInstance>,<Sketchup::Group>]` (ve skutečnosti samozřejmě včetně id instance komponenty a id skupiny).

Pokud je vrácena hodnota `nil`, nachází se uživatel na úrovni hlavního modelu. Proto budou procházeny všechny objekty hlavního modelu a objekty s atributy uvnitř komponent či skupin nebudou zobrazeny. Pokud je nějaké pole vráceno, nachází se uživatel uvnitř nějakého objektu. Dále proto program zjišťuje, zda je nadřazeným objektem komponenta či skupina. U skupin lze totiž rovnou procházet její entity stejně jako u hlavního modelu. U instance komponenty je ale nejprve potřeba najít její definici a pak teprve projít všechny entity této definice.

Pro procházení všech entit je vždy využita metoda `each`. Nejprve sice bylo snahou procházet všechny objekty bežným `for` cyklem, to ale bylo velmi pomalé a u větších modelů s velkým počtem elementárních entit docházelo i k zamrznutí aplikace SketchUp. Naproti tomu metoda `each` funguje i u velkých modelů dobře. Procházení objektů touto metodou bude ukázáno na nejsložitějším příkladě komponenty v příkladu 4.12.

Nejprve je nalezen název komponenty, pokud jí byl nějaký přidělen. Poté je nalezena definice komponenty a v rámci definice jsou metodou `each` procházeny všechny její entity. Název komponenty je případně uložen do textového řetězce, názvy jednotlivých entit s atributy jsou jako textové řetězce postupně přidávány do pole `nazvy`. Nebyl-li název komponenty nebo dílčího objektu zadán, uloží se řetězec 'nepojmenováno'. Pokud má entita přiřazen obrázek, model nebo popis, je do dalších vytvořených polí `obrazky`, `modely` a `popisy` vkládáno na dané místo vždy pouze „ANO“ nebo „NE“. Pole `objekty_kody` slouží pro ukládání odkazů na entity. Poslední pole s názvem `kamery` je potom určeno pro uložení parametrů kamery pro daný objekt. Obdobně je řešeno procházení objektů i v rámci modelu a skupiny.

Ukázka kódu 4.12: Nástroj *Tabulka*, procházení entit v rámci komponenty

```
706 pocet_urovni = Sketchup.active_model.active_path.length
707 nadrazena_trida = Sketchup.active_model.active_path[pocet_urovni-1]
708 if nadrazena_trida.class == Sketchup::ComponentInstance #pokud jsme v komponentě
709         nadrazeny_objekt_nazev = 'komponenty '
710         #zjistíme název nadřazené komponenty, má-li ho
711         slovniky_komponenty = nadrazena_trida.attribute_dictionaries #všechny
                slovníky komponenty
712         if slovniky_komponenty != nil #pokud má komponenta přiřazeny atributy
```

```
713                 if slovniky_komponenty["SlovnikTIS"] != nil   #v našem slovniku
714                         if slovniky_komponenty["SlovnikTIS"]["Nazev"] != nil
715                                 nadrazeny_objekt_nazev = nadrazeny_objekt_nazev +
                                        slovniky_komponenty["SlovnikTIS"]["Nazev"] #
                                        uložen nazev
716                         else
717                                 nadrazeny_objekt_nazev = nadrazeny_objekt_nazev + '
                                        nepojmenováno'
718                         end #slovniky_komponenty["SlovnikTIS"]["Nazev"] != nil
719                 else
720                         nadrazeny_objekt_nazev = nadrazeny_objekt_nazev + '
                                nepojmenováno'
721                 end #if slovniky_komponenty["SlovnikTIS"] != nil
722         end #if slovniky_komponenty != nil
723
724         definice_komponenty = nadrazena_trida.definition #definice komponenty
725         objekty_komponenty = definice_komponenty.entities
726         #projdeme všechny entity v rámci komponenty
727         objekty_komponenty.each{|objekt|
728                 slovniky = objekt.attribute_dictionaries #vsechny slovníky entity
729                 if slovniky != nil #pokud ma entita přiřazeny atributy
730
731                         if slovniky["SlovnikTIS"] != nil #v našem slovníku
732
733                                 slovnik = slovniky["SlovnikTIS"] #vybereme slovnik
734                                 objekty_kody = objekty_kody + [objekt]
735
736                                 if slovnik["Nazev"] != nil
737                                         nazvy= nazvy + [slovnik["Nazev"]]
738                                 else
739                                         nazvy = nazvy + ["nepojmenováno"]
740                                 end #if slovnik["Nazev"] != nil
741
742                                 if slovnik["Obrazek"] != nil
743                                         obrazky = obrazky + ["ANO"]
744                                 else
745                                         obrazky = obrazky + ["NE"]
746                                 end #if slovnik["Obrazek"] != nil
747
748                                 if slovnik["Model"] != nil
749                                         modely = modely + ["ANO"]
750                                 else
751                                         modely = modely + ["NE"]
752                                 end #if slovnik["Model"] != nil
753
754                                 if slovnik["Popis"] != nil
755                                         popisy = popisy + ["ANO"]
756                                 else
757                                         popisy = popisy + ["NE"]
758                                 end #if slovnik["Popis"] != nil
759
760                                 kamery = kamery+[slovnik["Kamera"]] #nastavení
                                        kamery
761
762                         end  #if slovniky["SlovnikTIS"] != nil
763                 end} # if slovniky != nil
```

Po projití všech objektů modelu či komponenty/skupiny je již možné přistoupit k vytvoření vlastní souhrnné tabulky. Protože má tato tabulka obsahovat odkazy, které umožní změnit pohled kamery tak, aby byl nastaven na vybraný objekt, tento objekt zvýraznit a zobrazit o něm další informace, je nutno nejprve vytvořit funkci pro `WebDialog` s tabulkou opět pomocí metody `add_action_callback`. Tato funkce získá od `WebDialogu` zprávu, která obsahuje pořadové číslo vybraného objektu v rámci všech objektů s atributy podle toho, jak byly výše nalezeny (tedy vlastně index v rámci pole). Poté je nastaven pohled kamery podle příslušného prvku pole s parametry kamery. Dále je zobrazeno okno s dalšími informacemi o objektu (stejné jako při vybrání entity nástrojem `Výběr`).

Ukázka kódu 4.13: Část callbacku pro vybírání objektů z tabulky

```
830 @@wd_tabulka.add_action_callback("zoom_vyber") do |js_wd, zprava|
831         poradi = eval(zprava) #String -> Fixnum
832         objekt = objekty_kody[poradi]
833         Sketchup.active_model.selection.clear
834         Sketchup.active_model.selection.add objekt
835         kamera_vektor = kamery[poradi]
836         eye = kamera_vektor[0]
837         target = kamera_vektor[1]
838         up = kamera_vektor[2]
839         kamera = Sketchup::Camera.new
840         kamera.set eye,target,up
841         Sketchup.active_model.active_view.camera=kamera
```

Vlastní tabulka je opět postupně vytvořena jako textový řetězec v rámci Ruby skriptu a poté je zaslána prohlížeči do `WebDialogu` jako HTML. Jednotlivé řádky tabulky jsou do textového řetězce postupně přidávány ve `for` cyklu s využitím polí připravených dle postupu výše. Název objektu v řádku tabulky je vždy součásti hypertextového odkazu. Po kliknutí na něj je nastavena zpráva s indexem objektu pro Ruby do adresního řádku prohlížeče. Ve veškerém textu zobrazeném v tabulce lze i jednoduše vyhledávat po stisknutí obvyklé klávesové zkratky `Ctrl+F`. Toto vyhledávání je zajištěno webovým prohlížečem na pozadí `WebDialogu`.

Ukázka kódu 4.14: Vytvoření souhrnné tabulky a zobrazení příslušného WebDialogu

```
976 tabulka = '<table border="1" bordercolor="White" cellpadding="5" cellspacing="0">'+
        '<tr bgcolor="#FA5858"><th>Název</th><th>Obrázek</th><th>Model</th><th>Popis</
    th></tr>'
977
978 pocet = objekty_kody.length - 1
979
980 for i in 0..pocet
981         nazev = nazvy[i]
982         obrazek = obrazky[i]
983         model = modely[i]
984         popis = popisy[i]
985         kod = objekty_kody[i]
```

```
986         tabulka = tabulka+'<tr bgcolor="rgb(184,184,184)" ><td><a href="skp:
                zoom_vyber@'+i.to_s+'">'+nazev+'</a></td><td>'+obrazek+'</td><td>'+
                model+'</td><td>'+popis+'</td></tr>'
987 end #for i in 0..pocet
988
989 tabulka = tabulka+'</table>'
990 wd_tabulka_html = '<!DOCTYPE html>
991         <html>
992                 <head>
993                  <meta http-equiv="Content-Language" content="cs">
994                  <meta http-equiv="Content-Type" content="text/html; charset=utf
                         -8">
995                     <style>
996                             a:hover     {color: #FA5858}
997                     </style>
998                 </head>
999                 <body>
1000                 <h3>Objekty s dalšími informacemi</h3>
1001                 <h4>V rámci: '+nadrazeny_objekt_nazev+'</h4>
1002                 Pro vyhledávání použijte Ctrl+F</br></br>
1003                 <center>'+tabulka+
1004                '</center>
1005                </body>
1006         </html>'
1007
1008 @@wd_tabulka.set_html(wd_tabulka_html)
1009 if @@wd_tabulka.visible? == false
1010         @@wd_tabulka.show()
1011 end #@@wd_tabulka.visible?
```

Protože velké modely obsahují značné množství entit a další informace bude mít přiřazeny jen malá část z nich, mohlo by se zdát procházení všech objektů modelu při každém aktivování nástroje *Tabulka* neefektivní. Aby nebylo nutné vždy procházet všechny entity, bylo by nutno někde uvnitř modelu uchovávat seznam všech entit, které mají nastavené atributy. Potom by stačilo pouze cyklem projít tyto entity a vygenerovat tabulku.

Vhodným místem pro umístění přehledu objektů s atributy se zdál být objekt `Model`, tedy přímo aktivní model v rámci SketchUpu. Tomu by bylo možno nastavit atributy stejně jako v případě nižších objektů. Hodnotou atributu by mohlo být například pole obsahující identifikátory všech objektů s dalšími informacemi. Bohužel bylo zjištěno, že identifikátor objektu, který je dostupný například metodou `entityID`, není stálý mezi jednotlivými session SketchUpu, tedy mezi jednotlivými otevřeními modelu. Proč je tomu tak je otázkou, nicméně to znamená, že je nutno při aktivaci nástroje *Tabulka* vždy procházet všechny entity modelu. To je naštěstí díky metodě `each` realizovatelné v rozumném čase.

Vzhledem k tomu, že může nastat situace, kdy má uživatel otevřenu souhrnnou tabulku a přitom otevře komponentu či skupinu pro editaci, je pro přehlednost vhodné, aby se v té chvíli tabulka přepsala a zobrazila objekty s atributy

uvnitř komponenty či skupiny. To je zajištěno přidáním dvou `Observer` objektů. Ty jsou vytvořeny jako třídy `ApplicationObserver` a `ActivePathObserver`. Metoda `onActivePathChanged` v rámci `ActivePathObserveru` sleduje, zda je nějaká komponenta nebo skupina otevřena pro editaci. Při změně, tedy při otevření či zavření je, pokud je otevřena, obnovena souhrnná tabulka. Aby bylo zajištěno, že bude `Observer` přidán i po vytvoření nového modelu nebo otevření stávajícího modelu aplikací v rámci jednoho okna SketchUpu, byl vytvořen `Application Observer`. Ten pomocí metod `onOpenModel` a `onNewModel` sleduje právě otevírání stávajícího nebo vytváření nového modelu a vždy do modelu přidává novou instanci `ActivePathObserveru`. Kromě toho také přepisuje souhrnnou tabulku, takže jsou hned vidět objekty s atributy v rámci otevíraného modelu. Nový model pak bude mít tabulku samozřejmě prázdnou. Protože `ApplicationObserver` nebude fungovat při otevření modelu v novém okně SketchUpu, je nutno `ActivePathObserver` přidat ještě jednou i přímo v rámci Ruby skriptu.

Ukázka kódu 4.15: Vytvoření observerů

```
122 #Observer sledující otevírání nových modelů
123 class ApplicationObserver
124         def onNewModel(model)
125                 if @@wd_tabulka.visible? == true
126                         Sketchup.active_model.select_tool(Tabulka.new)
127                         Sketchup.send_action "selectSelectionTool:"
128                 end #if @@wd_tabulka.visible?
129                 Sketchup.active_model.add_observer(ActivePathObserver.new)
130         end #onNewModel(model)
131
132         def onOpenModel(model)
133                 if @@wd_tabulka.visible? == true
134                         Sketchup.active_model.select_tool(Tabulka.new)
135                         Sketchup.send_action "selectSelectionTool:"
136                 end #if @@wd_tabulka.visible?
137                 Sketchup.active_model.add_observer(ActivePathObserver.new)
138         end #onOpenModel(model)
139 end #class ApplicationObserver
140
141 Sketchup.add_observer(ApplicationObserver.new)
142
143 #Observer sledující otevírání komponent a skupin pro editaci
144 class ActivePathObserver
145         def onActivePathChanged(model)
146         #pokud je videt souhrnny prehled, obnovime tabulku
147                 if @@wd_tabulka.visible? == true
148                         Sketchup.active_model.select_tool(Tabulka.new)
149                         Sketchup.send_action "selectSelectionTool:"
150                 end #if @@wd_tabulka.visible?
151         end #onActivePathChanged
152  end #class ActivePathObserver
153
154 Sketchup.active_model.add_observer(ActivePathObserver.new)
```

## 4.6 Nástroj Přehled atributů

Jak již bylo řečeno výše, *Přehled atributů* není součástí panelu nástrojů, ale je dostupný pouze z nově vytvořeného submenu. Jedná se o nástroj, který zobrazuje všechny atributy, které má vybraný objekt nastaveny. Uživatel si tak může prohlédnout i atributy nastavené jiným způsobem, například i atributy Dynamických komponent. Atributy jsou zobrazeny ve standardním okně SketchUpu jednoduchým způsobem v řádcích, kdy jsou od sebe slovník, klíč a hodnota atributu odděleny lomítkem.

Třída `PrehledAtributu` má 3 metody. První dvě, `activate` a `resume`, zobrazí nápovědu ve stavovém řádku při aktivaci a opětovné aktivaci nástroje. Metoda `onLButtonDown` funguje při kliknutí na vybraný objekt podobně jako stejnojmenná metoda u nástrojů *Nastavení* a *Výběr*. Nejprve je proveden vlastní výběr a zvýraznění objektu, poté je prozkoumáno, zda obsahuje atributy. Všechny slovníky vybrané entity se procházejí pomocí metody `each`. Stejnou metodou se v rámci každého slovníku procházejí i všechny klíče. Výsledek se zobrazí v okně `messagebox` `MB_MULTILINE`, které je určeno speciálně pro zobrazování víceřádkových zpráv.

Ukázka kódu 4.16: Zobrazení přehledu atributů po kliknutí na objekt

```
1016 def onLButtonDown(flags, x, y, view) #pri kliknuti levym tlacitkem
1017         Sketchup.active_model.selection.clear #Vymazeme dosavadni vyber
1018         #Vyber
1019         ph = view.pick_helper
1020         ph.do_pick x,y
1021         best = ph.best_picked
1022 
1023         if best != nil
1024                 Sketchup.active_model.selection.toggle(best)
1025                 slovniky = best.attribute_dictionaries
1026                 if slovniky != nil
1027                         zprava ='Dictionary Name / Attribute name / Attribute value
                                '+"\n"
1028                         slovniky.each{|slovnik|
1029                         slovnik.each{|nazev,hodnota|
1030                                 zprava = zprava + slovnik.name + ' / ' + nazev.to_s
                                        + ' / ' + hodnota.to_s + "\n"}
1031                         }
1032                         UI.messagebox zprava, MB_MULTILINE
1033                 else
1034                         UI.messagebox 'Objekt nemá přiřazeny žádné atributy', MB_OK
1035                 end #if slovniky != nil
1036         end #if best != nil
1037 end #onLButtonDown
```

## 4.7 Poznámka ke komponentám a skupinám

Jak již bylo patrné z popisu výše, bylo při navrhování pluginu nutno vyřešit problém nastavování atributů objektům uvnitř komponent (u skupin je samozřejmě situace obdobná, dále budou pro přehlednost zmiňovány pouze komponenty). Protože je atributy možno nastavit každé entitě modelu, je taky možné nastavit je entitám, které se nacházejí uvnitř komponenty. Přitom samozřejmě i celá komponenta může být součásti komponenty další atd. Entita s atributy tedy může být zanořena hluboko uvnitř struktury modelu.

Uživatel, který si model prohlíží, získává představu o tom, který objekt má uloženy další informace, zejména ze souhrnné tabulky a také na základě zvýrazňování jednotlivých objektů s atributy pod kurzorem. Pokud je ale entita uvnitř komponenty, není ji možné pomocí třídy `PickHelper` vybrat, protože se `PickHelperu` komponenta navenek jeví jako jeden objekt. Pomocí Ruby API bohužel není možné otevírat komponenty pro editaci. Je to zřejmě proto, že editovat entity komponenty pomocí Ruby lze i bez toho (bez ohledu na to, kde se uživatel nachází). Vybírání pomocí třídy `PickHelper` ale s uzavřenou komponentou možné není.

Pokud bychom chtěli opomíjet strukturu modelu a pod kurzorem zvýrazňovat entity s informacemi uvnitř komponenty, bylo by zřejmě nutné procházet všechny entity komponenty a sledovat, zda se nenachází na souřadnicích, kde uživatel kliknul a kde by byl výběr pomocí `PickHelperu` prováděn. Když k tomu uvážíme, že může být několik komponent vnořeno do sebe a že by teoreticky mohly mít objekty nastaveny atributy v několika úrovních, vyžadovalo by řešení problému možná až zbytečně komplexní přístup. Pokud by se již podařilo vybrat objekty ve všech úrovních, bylo by nutné uživateli nabídnout rozhodnutí, který objekt chce vybrat (tj. jestli komponentu nebo entitu uvnitř). Pokud by tak byl celý objekt uložen jako jedna velká komponenta s atributy, musel by uživatel pro zobrazení informací o objektech uvnitř takto volit vždy. Skutečný přínos pro uživatele je tedy sporný.

Další možností, jak řešit nastíněný problém, by bylo zakázat nastavování atributů objektům uvnitř komponent. To by sice bylo jednoduché, možná zbytečně by to ale znemožnilo nastavovat atributy objektům ve více úrovních. Proto bylo rozhodnuto umožnit nastavování atributů i entitám uvnitř komponent. Při prohlížení jsou ale objekty s atributy vždy procházeny buď v rámci modelu nebo v rámci komponenty. Pro nastavování a samozřejmě také prohlížení objektů je potom nutno využít i základní nástroj SketchUpu *Select* a otevřít komponentu pro editaci ručně (tedy dvojklikem na komponentu). Objekt, který má přiřazeny atributy, bude zvýrazněn pouze v rámci dané úrovně, tedy buď v rámci modelu nebo při prohlížení uvnitř komponenty.

Dále byla upravena souhrnná tabulka, jak již bylo popsáno výše, takže vždy zobrazuje entity v rámci úrovně, tedy i entity s atributy uvnitř komponenty, je-li tato otevřena pro editaci. Pokud je tabulka zobrazena, je při otevření komponenty obnovena a uživatel tedy ihned ví, jaké objekty s atributy komponenta obsahuje a uvnitř jaké komponenty se uživatel nachází. Nakonec byl pozměněn i dialog s podrobnostmi o modelu, který je zobrazován při kliknutí na objekt nástrojem *Výběr* nebo při vybrání objektu pomocí odkazu ze souhrnné tabulky. Ten nyní případně obsahuje i informace o tom, že se uvnitř vybrané komponenty nachází další entity s atributy. Uživateli jsou dále zobrazeny i názvy těchto entit a je mu sděleno, že pro zkoumání takových objektů je nutno otevřít komponentu pro editaci nástrojem *Select*. Aby se takto zobrazily entity s atributy v rámci komponenty, je nutné, aby měla nějaké další informace, například alespoň název, přiřazeny i samotná komponenta. Jinak by samozřejmě taková komponenta nebyla zvýrazněna při použití nástroje *Výběr* a nebyla by ani uvedena v souhrnné tabulce.

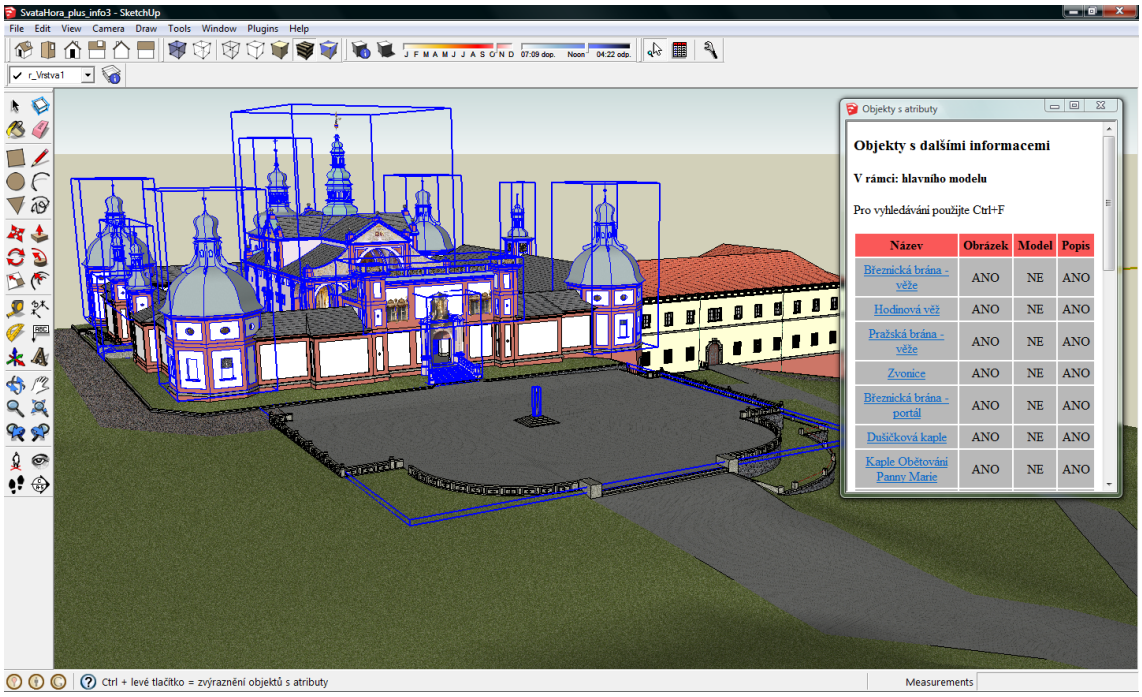
Uvedené vlastnosti pluginu tedy vlastně umožňují prohlížet model v několika úrovních podrobnosti. Pokud je celý model (tedy například celý památkový objekt) uložen jako komponenta, může mít jako celek přiřazeny další informace. Po otevření komponenty lze zobrazovat informace o menších celcích v rámci modelu, tedy například o jednotlivých budovách. Pokud jsou i budovy uloženy jako komponenty, lze je dále otevřít a prohlížet drobnější detaily budov atd. Aby ale prohlížení modelu takto fungovalo, je nutné model rozdělit na komponenty nejen z hlediska pohodlnějšího a efektivního modelování, ale také z hlediska logických celků s ohledem na pozdější prohlížení modelu. Nevýhodou také zůstává nutnost ručního otevírání komponent, které je dáno vlastnostmi Ruby API. Pohodlnější pro uživatele by samozřejmě bylo otevírání komponent např. přímo z okna s dalšími informacemi o objektu nebo ze souhrnné tabulky. Ten, kdo nastavuje další informace, ale není nijak nucen využívat uvedených vlastností pluginu. Pokud je model uspořádán tak, že jsou všechny objekty, které mají mít nastaveny atributy mimo komponenty nebo jsou přímo komponentami v rámci hlavního modelu, není samozřejmě nutné atributy přiřazovat objektům uvnitř komponent. Uživatel, který si informace prohlíží není potom vůbec nucen komponenty otevírat. To bude také pravděpodobně pro běžného uživatele nejpohodlnější. Pokud je naopak předpokládáno, že bude prohlížejícím zkušený uživatel SketchUpu, mohou být informace nastaveny do několika úrovní, což může být například u památkových objektů zajímavé.

Na závěr je také nutno poznamenat, že se komponenty, co se týká atributů, chovají dle pravidel SketchUpu. Každá instance komponenty je tedy samostatným

objektem a mohou ji být nastaveny vlastní atributy. Naopak entity uvnitř komponenty jsou v rámci modelu uchovávány pouze jednou (to je vlastně hlavní funkcí komponent). Pokud je tedy v modelu více instancí jedné komponenty a její entitě jsou nastaveny další informace, budou tyto atributy samozřejmě nastaveny v rámci všech instancí této komponenty. Uživatel tedy v tomto případě může zvážit použití skupin, kde k tomuto jevu pochopitelně nedochází.

## 4.8 Testování, vzorový model

Pro otestování funkčnosti vytvořeného zásuvného modulu byl použit 3D model poutního areálu Svatá Hora v Příbrami. Tento model je výsledkem diplomových prací tří studentů oboru Geodézie a kartografie – Marie Rajdlové, Michala Šatavy a Josefa Beníška[2]. Jedná se o poměrně rozsáhlý model, který obsahuje velké množství entit. Proto je tento model ideální pro otestování možností pluginu. Dalším důvodem výběru tohoto modelu pro demonstraci použití programu byla dostupnost údajů o jednotlivých objektech. K dispozici totiž byl text s podrobnostmi o jednotlivých částech areálu, který je používán jako podklad pro průvodce prohlídek Svaté Hory.




Obr. 4.5: Model Svaté Hory se zvýrazněnými objekty s dalšími informacemi

Před přidáním dalších informací o objektech modelu bylo nutno provést několik úprav. Některé entity modelu byly sloučeny do skupin či komponent jako například

[2]Více informací o 3D modelu Svaté Hory: http://geo3.fsv.cvut.cz/sh/index.html

rohové věže s kaplemi. Komponentu, která obsahovala terasu a baziliku, bylo naopak vhodné rozdělit do dvou částí. Protože v modelu nebyly vymodelovány jednotlivé otevřené kaple v ambitech, bylo nutno zobrazit je alespoň jednoduchou texturou. Tak byly získány plochy, pomocí kterých je možno odkazovat na další informace o těchto kaplích. Většina objektů, kterým byly přidány informace, se nachází přímo v úrovni hlavního modelu. Výjimkou je komponenta bazilika, která uvnitř obsahuje další objekty s informacemi – 3 kaple ve východní lodžii a Mariánskou věž. Na bazilice je tak možné vyzkoušet nastavování atributů objektům ve více úrovních a jejich následné prohlížení.

Jak již bylo řečeno, hlavním informačním podkladem byl text pro průvodce prohlídek v areálu. Pokud byly některé údaje nejasné nebo neúplné, byly použity informace dostupné na oficiálním webu Svaté Hory v sekci Informace/Průvodce po Svaté Hoře [31]. Dále byla využita i *Virtuální prohlídka* [32], dostupná taktéž na webových stránkách Svaté Hory, která sloužila hlavně jako pomůcka pro orientaci v areálu. K objektům byly přiřazeny také obrázky, využito bylo fotografií pořízených pro účely různých diplomových prací, které souvisejí s areálem Svaté Hory.

Plugin umožňuje přiřazovat objektům také odkaz na samostatný podrobnější model. V rámci diplomových prací, které souvisejí s areálem Svaté Hory, byly vytvářeny i 3D modely některých zdejších soch. Vzhledem k tomu, že v modelu nejsou nyní umístěny ani zjednodušené modely těchto soch, nebyly k dispozici objekty, pomocí kterých by bylo možné na podrobnější modely soch odkazovat. Proto byly na ukázku využity alespoň samostatné modely některých kaplí, jejichž vytvoření bylo jednou z částí diplomové práce Zuzany Kratinohové[3]. Tyto modely byly vygenerovány z fotografií v aplikaci Hypr3D[4] a byly k dispozici ke stažení ve formátu `dae`. Po jejich importu do aplikace SketchUp byly tyto modely uloženy do nativního formátu `skp`. Výstup z automatického zpracování fotografií ve webové aplikaci Hypr3D není zcela ideální, pro ukázku odkazu na samostatný model ale dostačuje.

Většina prací při nastavování atributů byla prováděna na PC s dvoujádrovým procesorem Intel Core 2 (2.00 GHz), 2 GB RAM, samostatnou grafickou kartou ATI 256 MB a operačním systémem Windows Vista SP2. Je možno říci, že tato sestava byla minimem pro pohodlné nastavování atributů u tak rozsáhlého a současně poměrně podrobného modelu, jakým je model areálu Svaté Hory. Otevírání modelu trvalo velmi dlouho, při rychlé změně pohledu a při ukládání modelu docházelo často k zamrzání aplikace SketchUp. Pro pohodlnější prohlížení modelu a hlavně

[3] Viz http://geo3.fsv.cvut.cz/dp/kratinohova/
[4] http://www.hypr3d.com/

pro nastavování je tedy nutno doporučit výkonnější počítač. Uvedené se ale týká celkově aplikace SketchUp. S popisovaným pluginem nebyly během testování z hlediska rychlosti problémy. V rámci pluginu jsou, jak bylo popsáno výše, často procházeny všechny entity modelu. S volbou vhodné metody (metoda `each` pro entity modelu resp. komponenty/skupiny) je toto i u velkého modelu možné. Procházení všech objektů běžným `for` cyklem by v tomto případě selhalo. Model Svaté Hory má v rámci hlavního modelu 111 314 entit.

Pro další vyzkoušení programu byly využity také jiné počítače. Jednalo se o PC s OS Windows 7 a dále dva počítače s Windows XP. Při testování OS Windows XP byl na obou počítačích při prohlížení atributů nalezen problém s otevíráním obrázků. Některé programy, které byly zkušebně nastaveny jako výchozí pro obrazové soubory, zde nefungují. Mezi nimi je bohužel i ve Windows XP standardní Prohlížeč obrázků a faxů. Při přidání odkazu na soubory jiného typu, jako jsou například tabulky ve formátu `xls` (MS Excel), projekty programu GIMP ve formátu `xcf` a další, fungují tyto bez problémů a otevřou se ve výchozím programu pro daný typ souboru. „Obyčejné“ obrazové formáty se ale neotevřou vůbec. To ale neplatí pro všechny programy, některé z testovaných totiž fungují. Jedním z takových programů je Internet Explorer. Ten sice není pro prohlížení obrázků nejvhodnější, na PC ho ale musí mít nainstalován každý uživatel popisovaného pluginu. Jinak by mu totiž nefungovaly ani žádné dialogy vytvořené pomocí třídy `WebDialog`. Při řešení problému byly hledány cesty, jak zjistit pomocí Ruby operační systém uživatele a jak případně zajistit otevírání obrázků v Internet Exploreru v případě použití Windows XP. Nalezeny byly zatím ale pouze možnosti zjištění druhu operačního systému, tedy zda používá uživatel OS Windows, Linux nebo Mac OS X. To ale v tomto případě samozřejmě nestačí, protože by bylo nutno zjistit i verzi OS. Jediným řešením je tedy zatím doporučit uživatelům s operačním systémem Windows XP, aby před použitím pluginu nastavili jako výchozí program pro otevírání obrazových souborů aplikaci Internet Explorer. Na počítačích s OS Windows Vista a Windows 7 probíhalo otevírání obrázků bez problémů ve všech testovaných výchozích programech.

Kromě výše popsaného prohlížení dalších informací bylo na všech počítačích zkoušeno také jejich nastavování. Zde proběhlo testování bez problémů. Zajímavé by také bylo vyzkoušet plugin v aplikaci SketchUp na počítači s operačním systémem Mac OS X. Funkčnost SketchUp Ruby API by totiž na počítačích Apple Mac měla být téměř stejná jako na PC. Takový počítač ale bohužel nebyl k dispozici. Pro OS

Linux neexistuje zatím standardní[5] verze programu SketchUp a proto plugin nebyl v tomto operačním systému testován.

Kromě různých operačních systémů byly také zkoušeny různé verze aplikace SketchUp. Plně funkční by plugin měl být od verze 7, doporučuji však použít raději minimálně verzi 8. V ještě starší verzi programu SketchUp, tedy ve verzi 6, již některé části pluginu nefungují. Důvodem je starší Ruby API v této verzi, které ještě neobsahuje všechny metody použité při programování pluginu. Příkladem může být souhrnná tabulka, která se ve verzi 6 nezobrazí. Ruby API ve verzi 6 totiž ještě neobsahuje metodu, pomocí které je zjišťováno, zda uživatel neotevřel nějakou komponentu či skupinu pro editaci. Vzhledem k tomu, že modely ze starších verzí aplikace SketchUp lze otevřít ve verzích novějších a opačně to není možné, je však otázkou, proč by měl uživatel vůbec používat starší verze. Kromě toho je práce s rozsáhlými modely v aplikaci SketchUp 6 výrazně pomalejší než ve verzích nových.

Obrázky, modely a textové soubory popisů, na které bylo při nastavování atributů odkazováno, byly uloženy do adresářů, které se nacházejí vedle vlastního souboru modelu Svaté Hory. Některým objektům byl přiřazen pouze jeden obrázek nastavením relativní cesty k němu do příslušného vstupního pole dialogu nástroje *Nastavení*. Tam, kde to bylo vhodné, byly vytvořeny odkazy na další lokálně umístěné obrázky pomocí speciálního odkazu popisovaného výše (tedy pomocí odkazu ve tvaru `<a href="skp:soubor@adresa">`). Nevýhodou použití tohoto odkazu, který je pouze zprávou pro Ruby API, je nemožnost zobrazení náhledu obrázku v odkazu. To je samozřejmě u běžných hypertextových odkazů, které zpracovává pouze webový prohlížeč, možné.

Všechny obrázky a případné podrobnější modely jsou umístěny lokálně vedle modelu hlavního. Velikost všech dat, které musí mít uživatel na svém počítači proto, aby si model se všemi informacemi prohlédl, je tedy značná. Komprimace obrázků by přitom byla samozřejmě zcela nevhodná, protože jedním z cílů pluginu je umožnit prohlížení detailů prvků, které jsou jinak v modelu zobrazeny často právě pomocí textur – značně komprimovaných obrázků. Otázkou tedy je, zda by nebylo vhodné uchovávat další obrázky v plném rozlišení, příp. i jiné soubory, na které je odkazováno v popisu, na webu. Odkazování na takové soubory by bylo řešeno pomocí standardního hypertextového odkazu v popisu. Potom by uživatel, který si model prohlíží, musel mít na svém počítači uloženy pouze obrázky a modely, které byly nastaveny přímo pomocí dialogu nástroje *Nastavení*. Pro prohlížení dalších obrázků, které by

[5] Používat SketchUp na počítači s operačním systémem Linux by mělo být možné pomocí aplikačního rozhraní Wine: http://wiki.winehq.org/Sketchup

byly dodatečně přiřazeny v popisu, by potom musel být připojen k internetu. To dnes už ale nelze pokládat za nevýhodu.

Na závěr bude ještě uvedena poznámka, která souvisí s testováním pluginu, týká se však celé aplikace SketchUp. V průběhu nastavování dalších informací objektům Svaté Hory bylo přistoupeno ke změně některých komponent modelu, jak již bylo řečeno výše. Vytvořeny byly komponenty nové, některé stávající byly naopak zrušeny, tedy rozbity na jednotlivé entity. Přitom bylo zjištěno, že se velikost souboru modelu výrazně zvětšila (z 66 MB na 76 MB). Uvedený jev byl způsoben tím, že veškeré definice komponent, tedy informace o tom, jak jsou komponenty složeny z jednotlivých entit, zůstávají v modelu i po smazání všech instancí komponent. Pokud bychom tedy měli model obsahující pouze jednu komponentu a tuto komponentu rozložili v rámci modelu na jednotlivé entity, zvětšila by se velikost modelu přibližně na dvojnásobek. V modelu by totiž byly uloženy všechny entity dvakrát – jednou jako samostatné viditelné objekty a podruhé jako neviditelná definice komponenty. Někdy může být uchovávání definic komponent, jejichž instance se již v modelu nevyskytují, vhodné. Pokud ale chceme zmenšit velikost souboru modelu a takové definice již nepotřebujeme, můžeme je vymazat. Pro jednoduché vymazání definic nepoužitých komponent slouží funkce *Purge Unused* (Window/Components/In Model/Purge Unused). Alternativou je použití stejnojmenné Ruby API metody, tedy: `Sketchup.active_model.definitions.purge_unused`. Například u modelu Svaté Hory se velikost po vymazání nepoužívaných definic komponent zmenšila z výše uvedených 76 MB na výsledných 55 MB. Toto zmenšení souboru modelu sice nezrychlí jeho vykreslování, protože jsou odstraněny pouze nezobrazované objekty, pro přenos dat mezi uživateli ale může být výhodné.

# 5 Možnosti exportu na web

V následující kapitole budou zkoumány možnosti exportu modelu, který byl vytvořen v aplikaci SketchUp, pro sdílení na webu. Nejprve bude popsán formát `swf` a zobrazování s využitím webového prohlížeče a jeho pluginu Adobe Flash Player. Více místa potom bude věnováno nové specifikaci HTML5 a možnosti prohlížet modely přímo ve webovém prohlížeči pomocí JavaScriptového rozhraní WebGL. Popsány budou také formáty, pomocí kterých lze model z programu SketchUp exportovat.

## 5.1 Adobe Flash a formát SWF

Formát `swf` (Shockwave Flash) slouží pro zobrazování animací, bitmapové a vektorové grafiky a videa. Prezentace v tomto formátu mohou obsahovat i interaktivní prvky, jako jsou například různá tlačítka [33]. Formát byl vyvinut společností FutureWave Software pro zobrazování animací. Od roku 1996 do roku 2005 byl formát vlastněn společností Macromedia. Od roku 2005 je potom formát i se společností Macromedia vlastnictvím firmy Adobe Systems [34].

Vytvářet soubory `swf` mohou samozřejmě některé produkty firmy Adobe. Jedná se v první řadě o program Adobe Flash Professional[1], což je editor vektorové grafiky, který může být využit například pro vytváření her či interaktivních prezentací na webu. K takovým účelům je možno v rámci editoru využít také skriptovací, objektově orientovaný programovací jazyk ActionScript. ActionScript vychází z jazyka ECMAScript, což je standardizovaná verze JavaScriptu (viz 5.2.2) [35]. Exportovat formát `swf` je ale možné i pomocí softwaru nepatřícího do vlastnictví firmy Adobe. Některé programy, které budou popsány v následujících kapitolách, export do tohoto formátu umožňují. Pro zobrazování formátu `swf` pomocí webového prohlížeče je nutno mít nainstalován plugin Adobe Flash Player[2]. Tento zásuvný modul je na rozdíl od aplikace Adobe Flash Professional k dispozici zdarma. Do webového prohlížeče Google Chrome je potom Adobe Flash Player přímo integrován [36].

[1] http://www.adobe.com/cz/products/flash.html
[2] http://get.adobe.com/cz/flashplayer/

## 5.2 HTML5, Canvas, WebGL

### 5.2.1 HTML5

HTML5 je nejnovější specifikací jazyka HTML (Hyper Text Markup Language), který je používán pro popis struktury webových stránek. V současné době ještě není tato specifikace oficiálně standardizována, schválení je předpokládáno do konce roku 2014. Většina moderních webových prohlížečů ale již nové prvky HTML5 podporuje. Nový standard by měl umožňovat jednodušší zápis HTML dokumentů a lepší možnosti strukturování stránek. Kromě toho lze v rámci HTML5 například přehrávat multimediální soubory přímo uvnitř webového prohlížeče bez nutnosti instalace dodatečných pluginů (nejčastěji Adobe Flash Player). Kvůli tomu byly také definovány nové tagy jako `<video>` nebo `<audio>` [37].

### 5.2.2 JavaScript

Protože bude v následujících oddílech zmiňováno použití JavaScriptu, zejména pro vykreslování grafiky ve webovém prohlížeči, bylo by vhodné krátce tento programovací jazyk charakterizovat. JavaScript je interpretovaný skriptovací, objektově orientovaný programovací jazyk. Jeho syntaxe se podobá syntaxi jazyků C, C++ či Java. Používá se při tvorbě webových stránek, programy se zapisují přímo do HTML kódu. Pomocí JavaScriptu se vytvářejí klientské skripty, programy jsou tedy spouštěny na počítači uživatele. Zapisovat skript do HTML je možné několika způsoby. První možností je zapsání skriptu mezi HTML tagy `<script>` a `</script>` do proudu dokumentu. Druhou možností je načtení externího skriptu (textový soubor s příponou .js) pomocí stejných tagů. Posledním způsobem je in-line zápis skriptu s využitím jiných tagů (např. `<button>`) [39].

### 5.2.3 Canvas

Canvas (angl. plátno) je dalším z nových elementů definovaných v HTML5. Jedná se o „bitmapové kreslící plátno závislé na rozlišení, které může být za běhu využíváno pro vykreslování grafů, herní grafiky a ostatních vizuálních prvků“. Je to tedy obdélníkový prostor, který slouží pro vykreslování grafiky pomocí JavaScriptu. Výše bylo psáno o podpoře HTML5 ve webových prohlížečích. Element canvas by měl být podporován prohlížečem Firefox od verze 3, prohlížečem Chrome také od verze 3 a prohlížečem Internet Explorer nativně od verze 9 [41]. Tyto starší verze sice nebyly testovány, ale v Google Chrome v. 30 a ve Firefoxu 24 byla grafika v canvasu

zobrazována bez problémů. V Internet Exploreru 9 se ale zatím canvas nepodařilo správně zobrazit.

Příklad jednoduché HTML stránky s elementem canvas je zobrazen níže.

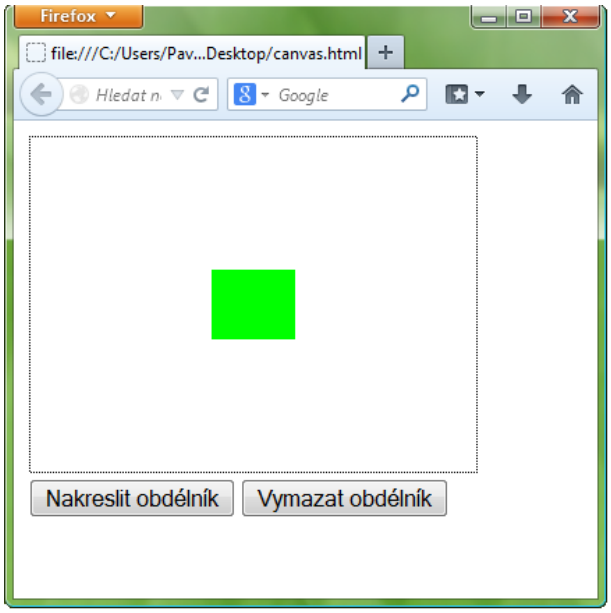
Ukázka kódu 5.1: HTML stránka s elementem canvas

```
<!DOCTYPE HTML>
<html lang="cs">
<head>
        <meta charset="windows-1250">
</head>
<body>
        <table><tr><td>
                <canvas
                        id="platno"
                        width="320"
                        height="240"
                        style="border:1px dotted;float:left">
                </canvas>
        </td><tr><td>
                <button
                        style='font-size:12pt'
                        onclick='kreslit()'
                >Nakreslit obdélník
                </button>
                <button
                        style='font-size:12pt'
                        onclick='mazat()'
                >Vymazat obdélník
                </button>
        </td></table>

        <script type='text/javascript'>
                var platno = document.getElementById("platno")
                var platno_context = platno.getContext("2d")
                platno_context.fillStyle="rgb(0,255,0)";
                function kreslit() {
                        platno_context.fillRect(platno.width/2-30,platno.height
                            /2-25,60,50)}
                function mazat() {
                        platno_context.clearRect(platno.width/2-30,platno.height
                            /2-25,60,50)}
        </script>
</body>
</html>
```

Jak bude taková HTML stránka vypadat v prohlížeči je ukázáno na obrázku 5.1. Na stránce se nachází jeden canvas element (s identifikátorem `platno`), který je ohraničen tagy `<canvas>` a `</canvas>`. Aby bylo „plátno“ viditelné, bylo ohraničeno okrajem s tečkovanou čarou. Dále jsou na stránce dvě tlačítka, pomocí kterých jsou volány JavaScriptové funkce. Nejdůležitější na stránce je tedy vlastní JavaScript.

Pomocí metody `getElementById` je nejprve canvas vyhledán pomocí identifikátoru `id` v rámci objektů stránky, tedy v rámci DOM (Document Object Model). Dále




Obr. 5.1: HTML stránka s elementem canvas

je pro něj vytvořeno 2D kreslící prostředí metodou `getContext`. Tato metoda vrací objekt, který teprve poskytuje metody pro kreslení v canvasu. Funkce `kreslit` vytvoří v rámci prostředí zelený obdélník ve středu canvasu, využívá metody `fillRect`. Druhá funkce `mazat`, která je volána druhým tlačítkem v rámci stránky, potom obdélník na stejných souřadnicích vymaže. K tomu je použita metoda `clearRect`.

V tomto příkladu byl tedy v rámci elementu canvas vytvořen velmi jednoduchý dvourozměrný obrazec. Pro nakreslení a smazání obrazce byly použity funkce JavaScriptu. Dále bude popsáno, jak v canvasu zobrazovat místo dvourozměrného i 3D obsah.

### 5.2.4 WebGL

WebGL je JavaScriptové rozhraní, které je navrženo pro vykreslování 2D i 3D grafiky prostřednictvím HTML elementu canvas. Taková grafika je potom zobrazována webovými prohlížeči s podporou WebGL bez potřeby dalších zásuvných modulů. Rozhraní je vyvíjeno konsorciem *Khronos group*. V rámci WebGL je přímo programován grafický procesor (GPU) počítače. Proto je každá WebGL aplikace vytvářena ve dvou jazycích. Prvním jazykem je JavaScript, který je používán pro popis zobrazovaných objektů, nastavování barev a načítání obrázků (textur). Druhým jazykem je potom GLSL (OpenGL Shading Language), nízkoúrovňový jazyk založený na jazyku C, který slouží pro ovládání GPU [42].

Tvary jsou ve WebGL popisovány pomocí trojúhelníků. Každý takový trojúhelník může být definován buď třemi vrcholy nebo jen vrcholem jedním s tím, že další

dva vrcholy tvoří stranu trojúhelníku předchozího. Ve druhém případě je tedy jako celý trojúhelník třemi vrcholy definován pouze trojúhelník první. Mimo trojúhelníků by bylo možné popisovat i jednotlivé body nebo linie. Každý vrchol (vertex) trojúhelníka je popisován prostorovými souřadnicemi x, y a z [43].

Prvním krokem vykreslovacího řetězce (rendering pipeline) WebGL je vytvoření polí vrcholů (vertex arrays). Ty obsahují informace o jednotlivých vrcholech, tedy pozici vrcholu v prostoru, informace o textuře, barvě a o tom, jak je vrchol ovlivněn osvětlením (normála vrcholu – vertex normal). Tato pole jsou vytvořena JavaScriptem. Podkladem pro jejich tvorbu může být zpracování souborů, které popisují 3D model (např. soubory ve formátu `obj`) nebo knihovny, které poskytují pole vrcholů pro geometrické útvary. Poslední možností je potom vytvoření zcela nových vrcholů přímo pomocí JavaScriptu.

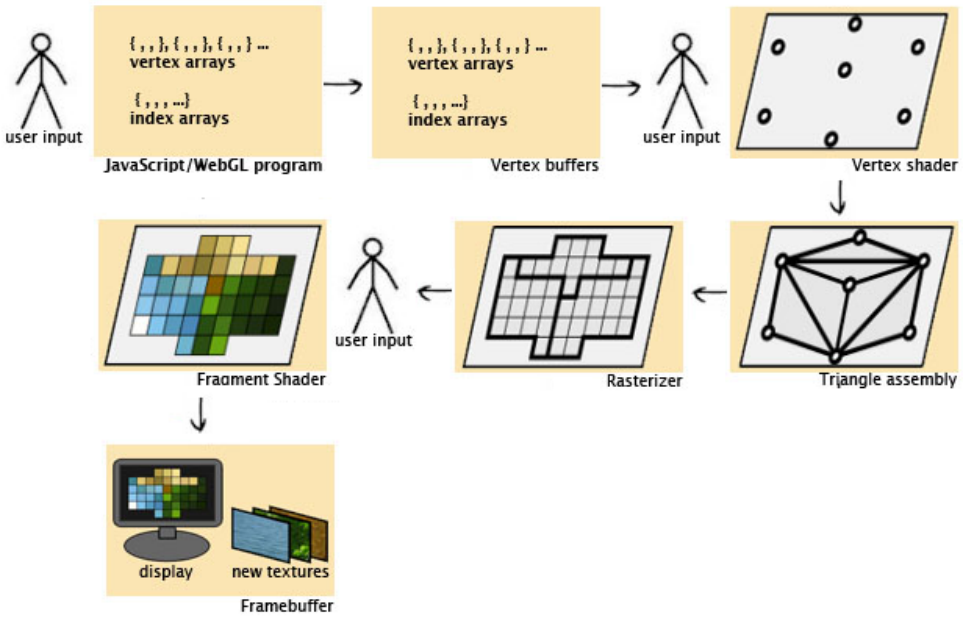
Do GPU jsou data ve formě polí vrcholů předávána uložená v *bufferech* (vertex buffers), tedy v částech paměti. Kromě polí vrcholů je poskytováno ještě další pole indexů (array of indices). Pomocí něho budou vrcholy později složeny do trojúhelníků.

Každý vrchol z vertex bufferu je v GPU zpracováván pomocí *vertex shaderu*. To je program, který vypočítá pozici, v níž se zobrazí vrchol s danými prostorovými souřadnicemi v rámci zobrazovací plochy (např. v rámci canvasu). Kromě toho může pro každý vrchol počítat další atributy jako barvu a texturu. Tento program je psán v jazyce GLSL. Uživatel si může buď napsat vertex shader vlastní nebo může využít vhodné JavaScriptové knihovny.

Dále jsou v GPU dle pořadí popsaného v poli indexů vytvořeny trojúhelníky. *Rasterizer* poté ořízne části, které jsou mimo obraz a rozbije zbývající viditelné části trojúhelníků do fragmentů velikosti pixelů. Další atributy pro vrcholy, které jsou vypočítány vertex shaderem jsou interpolovány do pixelů mezi vrcholy. Takto vytvořené fragmenty jsou zpracovávány *fragment shaderem*. Ten vrací hodnoty barvy a hloubky (depth value) pro každý pixel. V rámci fragment shaderu tedy probíhá také výpočet osvětlení a texturování. Fragment shader je program, který opět musí uživatel, pokud nepoužije vhodnou knihovnu, napsat v jazyce GLSL. Nakonec je pixel s hodnotami, vypočtenými fragment shaderem odeslán do *framebufferu*. Před vlastním zobrazením na obrazovce (zde tedy v rámci canvasu ve webovém prohlížeči) je provedeno odstranění všech objektů, které jsou zakryty (depth testing) [44].

V rámci jednoduchého popisu vykreslovacího řetězce WebGL byly několikrát zmíněny JavaScriptové WebGL knihovny. Ty může uživatel použít, pokud nechce sám přímo psát shadery, tedy programy ovládající GPU, v jazyce GLSL. Takovou knihovnou je třeba herní engine CopperLicht, který bude popsán dále (viz 6.4). Dalším

příkladem potom může být třeba jednoduchá a velmi využívaná knihovna Three.js[3]. Jak již bylo řečeno, pro zobrazování grafiky v canvasu s pomocí WebGL je nutný webový prohlížeč, který WebGL podporuje. Takovými prohlížeči jsou nyní například Firefox (od verze 4) a Chrome (od verze 9) [44]. Internet Explorer v současné době WebGL nepodporuje. To by se ale mělo změnit s verzí 11 [43].




Obr. 5.2: Vykreslovací řetězec WebGL (upraveno z [44])

## 5.3 Formáty pro přenos dat z programu SketchUp

Aby bylo možno prezentovat modely vytvořené pomocí programu SketchUp na webu s využitím WebGL (třeba pomocí aplikací uvedených v kapitole 6), je nutno exportovat model do vhodného 3D formátu. Vlastní formát SketchUpu `skp` totiž většinou není možno použít, což může souviset s tím, že se jedná o formát proprietární. Protože neplacená verze programu SketchUp (tedy SketchUp Make[4]) umožňuje export 3D modelu pouze do formátů `dae` a `kmz`, může být vhodné využít pro uložení modelu do dalších formátů osmihodinovou zkušební verzi placeného SketchUpu Pro[5]. Dále budou krátce popsány 3D formáty, které byly využity v rámci diplomové práce.

[3] http://threejs.org/
[4] http://www.sketchup.com/products/sketchup-make
[5] http://www.sketchup.com/download?sketchup=pro

Po exportu modelu do těchto formátů byly zkušební modely uploadovány do databáze Sketchfab (viz dále 6.3), případně importovány do editoru CopperCube (viz 6.4).

### 5.3.1 Wavefront OBJ

Formát `obj` byl vyvinut společností *Wavefront Technologies*. Jedná se otevřený, textový formát, který je čitelný pro člověka (human-readable) [48]. Jeho editace a prohlížení je tedy možné i v běžném textovém editoru. Po exportu modelu z programu SketchUp Pro získáme [49]:

1. Soubor s příponou `obj`, který obsahuje popis 3D geometrie, tedy:
   - Pozici každého vrcholu
   - Údaje o umístění textur
   - Normály vrcholů
   - Informace o tom, jak vrcholy vytvářejí mnohoúhelníky (plochy)
   - Odkaz na příslušný `mtl` soubor (viz níže)
2. Soubor `mtl` (Material template library), který popisuje vizuální stránku modelu. Ten obsahuje:
   - Informace o materiálech, které jsou aplikovány na plochy mnohoúhelníků
   - Údaje o texturách, průhlednosti a o chování materiálů při osvětlení
   - Odkazy na obrazové soubory, které byly využity pro tvorbu textur
3. Adresář s obrazovými soubory textur.

### 5.3.2 COLLADA DAE a KMZ

Formát COLLADA (COLLAborative Design Activity) [50] je vyvíjen konsorciem *Khronos Group*. Jedná se o výměnný formát, který je založen na značkovacím jazyce XML. Soubory s příponou `dae` (Digital Asset Exchange) jsou tedy samozřejmě čitelné pro člověka, jejich XML struktura je ale složitější než například textový formát souborů `obj`. Jak již bylo řečeno, export 3D modelu ve formátu `dae` podporuje i neplacená verze SketchUp Make.

Model ve výměnném formátu `dae` je používán i při exportu modelu do formátu `kmz`. Ten mohli uživatelé SketchUpu používat pro nahrávání modelů do vrstvy 3D

objektů v aplikaci Google Earth[6]. Soubor `kmz` je potom pouze přejmenovaným ZIP archivem. V něm se nachází adresář se souborem modelu ve formátu `dae` a složkou textur. Dále potom soubor ve formátu KML, který importuje COLLADA model a obsahuje informace o jeho umístění v rámci Google Earthu [52]. KML (Keyhole Markup Language) je opět aplikací značkovacího jazyka XML. Vzhledem k tomu, že Google ukončil 1. října 2013 přijímání uživatelských modelů pro Google Earth, nebude zřejmě uživateli export ze SketchUpu do formátu `kmz` příliš využíván [1].

### 5.3.3 VRML

VRML (Virtual Reality Modeling Language) je jazyk pro popis 3D objektů, který je standardizován normou ISO[7]. Soubory, které jsou vytvořeny v tomto jazyce mají příponu `wrl` (jsou nazývány worlds – světy) a jedná se o soubory textové. 3D modely jsou v tomto formátu popisovány s využitím stromové struktury, kdy jsou jednotlivé objekty hierarchicky uspořádány [53].

Přímo v rámci jazyka VRML by mělo být možné vytvářet i modely s aktivními objekty. Takové objekty potom mohou reagovat například na kliknutí myši, po kterém by bylo možné mj. otevřít webovou stránku. Pro prohlížení modelů ve formátu `wrl` na webu je ale nutný plugin pro webový prohlížeč (např. Cortona 3D Viewer[8]). V době, kdy se začíná rozšiřovat využívání HTML5 a WebGL se proto může zdát VRML poněkud zastaralé.

### 5.3.4 Autodesk 3DS

Formát `3ds` je formátem binárním a tedy pro člověka nečitelným. Vyvinut byl pro software společnosti *Autodesk* – 3D Studio. Tento software, který je nyní prodáván pod názvem *3ds Max*[9], slouží pro modelování 3D objektů, vytváření animací a renderování [55].

Zajímavostí je, že formát `3ds` lze otevřít pro čtení CAD softwarem *Microstation*[10], který je často využíván pro geodetické účely. Zkušebně byl model exportovaný ze SketchUpu otevřen ve studentské verzi Microstation PowerDraft XM Edition (verze 8). Model byl zobrazen poměrně správně, potíže nastaly pouze u několika textur, které nebyly zobrazeny nebo byly zaměněny za jiné. Tento formát by tak mohl být také využíván pro přenos dat z aplikace SketchUp do Microstationu, kdy

[6] http://www.google.cz/intl/cs/earth
[7] VRML97 je standardizován normou ISO/IEC 14772-1:1997
[8] http://www.cortona3d.com/cortona3dviewer
[9] http://www.autodesk.cz/products/autodesk-3ds-max/overview
[10] http://www.bentley.com/cs-CZ/Products/MicroStation/

by takto otevřený model šlo dále uložit do jiného formátu, například do nativního formátu Microstationu `dgn`. Nevýhodou může ovšem být, kromě potíží s texturami, také nepřehledné rozvrstvení modelu po importu do Microstationu. Kromě toho se také ze SketchUpu do formátu `3ds` nepodařilo správně exportovat hrany v modelu. Možnost další editace v Microstationu je tedy otázkou.

### 5.3.5 Formát FBX

Tento proprietární formát byl vytvořen společností *Kaydara* pro užití v rámci softwaru *Filmbox*, který sloužil pro animaci 3D postav. Uvedený program je nyní pod názvem *MotionBuilder* rozvíjen společností *Autodesk*. Formát `fbx` má dvě verze, textovou a binární [56]. Export ze SketchUpu probíhá do textové verze.

# 6 Aplikace pro export modelu na web

Po teoretickém popisu možností exportu 3D modelu na web budou popsány aplikace, které takový export umožňují. U těchto aplikací bude zkoumána samozřejmě hlavně možnost přidávat a zobrazovat další informace o objektech modelu.

## 6.1 Spread3D

Spread3D[1] je program, který umožňuje export modelu vytvořeného pomocí aplikace SketchUp do podoby webové stránky. Výsledný soubor tak lze prohlížet ve webovém prohlížeči s nainstalovaným Adobe Flash Playerem. K dispozici jsou tři verze programu: Pro, Lite a Free. Jediná verze Free je potom zdarma.

Export modelu prostřednictvím aplikace Spread3D je jednoduchý. Stačí pouze importovat soubor modelu ve formátu `skp` a poté ho exportovat jako webovou stránku.

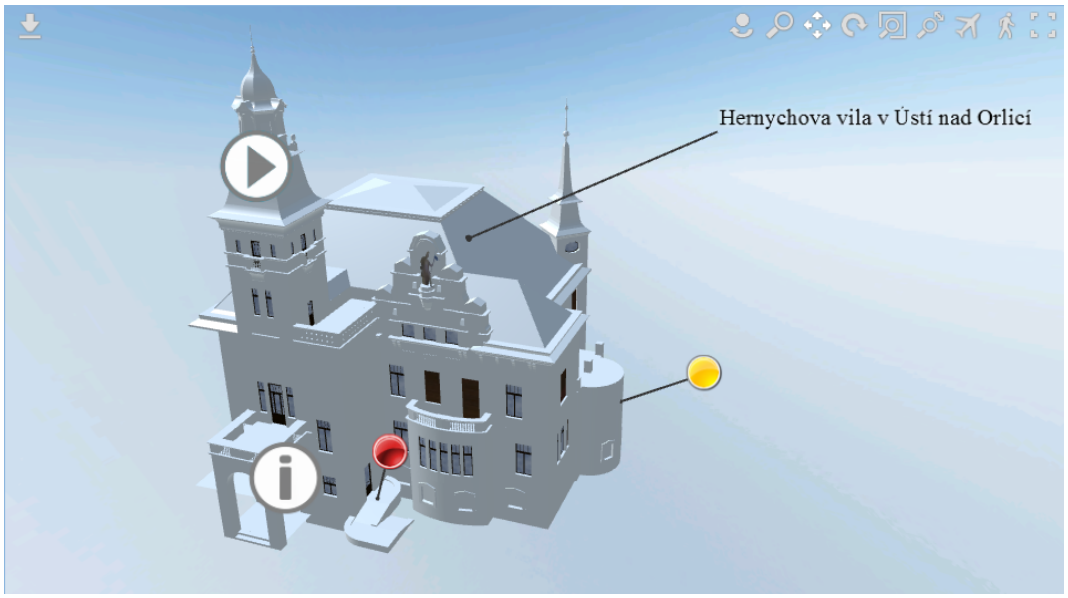
Pro vyzkoušení funkčnosti programu Spread3D Pro byla využita jeho trial verze. V této trial verzi je možné využívat všech funkcí programu, je ale znemožněno exportování výsledných projektů. V záložce *Model* oproti Free verzi programu přibyly další volby. Kromě nastavení zobrazení modelu a pohledů na něj zde tak lze do modelu přidávat 2D stromy a postavy, zvuky a vytvářet animace.

Pro potřeby informačního systému je potom nejdůležitější položka *Interactivity & Info*. Do modelu lze pomocí tohoto nástroje přidávat popisky (label) a tlačítka (button). Tlačítka potom mohou sloužit jako odkaz na webovou stránku, pro zobrazení obrázku nebo pro přehrání animace. Dále mohou tlačítka sloužit i pro přechod mezi jednotlivými náhledy na model.

Spread3D ve verzi Pro tedy v podstatě umožňuje zobrazovat další informace o modelovaných objektech. O uchovávání informací lze v tomto případě hovořit pouze u popisků. Velkou předností je ale jeho nezávislost na aplikaci SketchUp a možnost prohlížení výsledného modelu pomocí webového prohlížeče. Jedná se ale, jak již bylo řečeno o nejvyšší, placenou verzi programu. I verze Free představuje pěkný způsob webové prezentace modelu, ale už bez dalších přidaných informací a interaktivity. Na závěr je nutno také poznamenat, že se při testování programu u větších modelů několikrát vyskytl problém se správným zobrazováním některých ploch jak v náhledu, tak v exportovaném modelu.

Až po napsání tohoto textu bylo zjištěno, že vývoj softwaru Spread3D pokročil. Tento program tedy v podobě, v jaké byl popsán výše, již není dále rozvíjen a není

[1] http://www.spread3d.com




Obr. 6.1: Spread3D (screenshot z programu) – model doplněný dalšími informacemi

ani k dispozici ke stažení. Stejnojmenný nástupce je zatím (v době psaní tohoto dodatku) ve fázi soukromého beta testování [58]. Podle propagačních videí by mělo být v novém Spread3D možné přímo v rámci stránky s 3D modelem přidávat panely pro prohlížení dalších webových stránek, videí nebo pro zobrazení mapy. Tyto další informace jsou ale zřejmě vztaženy k celému modelu. Zda zůstanou ve Spread3D také výše popsané funkce zatím není jasné.

## 6.2 Hypercosm Teleporter

Program Hypercosm Teleporter[2] slouží pro export modelů a jejich zobrazování v rámci webové stránky. Pracuje tedy na podobném principu jako výše uvedený Spread3D. Pro export je zde ale využíván přímo program SketchUp, pro uskutečnění exportu je nutno mít nainstalovánu jeho placenou verzi SketchUp Pro. Pro zobrazování exportovaného modelu je potom nutný webový prohlížeč a taktéž nainstalovaný Hypercosm 3D Player. V současné době je k dispozici pouze produkt Hypercosm Teleporter Free, při jehož instalaci bude zároveň instalován i prohlížeč 3D Player.

Export pomocí programu probíhá z běžného menu aplikace SketchUp (File/-Export/3D Model). Pokud je Teleporter správně nainstalován, zobrazí se při ukládání jako jeden z výstupních formátů v políčku *Export type* možnost Hypercosm

[2]http://www.hypercosm.com/teleporter.html

(*.html). Po otevření souboru ve webovém prohlížeči se spustí plugin Hypercosm 3D Player a je možno model prohlížet.

Výhodou programu Teleporter je malá velikost výsledných souborů po exportu (např. 4 MB oproti 16 MB výstupu z programu Spread3D). Zřejmou nevýhodou je potom nutnost vlastnit pro export Pro verzi programu SketchUp a mít pro prohlížení ve webovém prohlížeči instalován nestandardní plugin. Na webových stránkách programu lze nalézt nástroj Hyperlink Tools, který byl součástí verze programu Teleporter Pro [59]. Ten zřejmě umožňoval přidávat do výsledného modelu odkazy na webové stránky a poznámky. Bohužel bylo zjištěno, že vývoj programu Hypercosm Teleporter již pravděpodobně neprobíhá a Pro verze již není k dispozici.

## 6.3 Sketchfab

Sketchfab[3] je webová služba, která slouží pro zveřejňování a sdílení 3D modelů. Protože je založena na WebGL, není pro zobrazování modelů potřeba žádný dodatečný plugin pro webový prohlížeč. Samozřejmostí je ale nutnost podpory WebGL ze strany prohlížeče. Služba by měla být obdobou serveru YouTube[4], kdy jsou ale místo videí sdíleny 3D modely. Stejně jako u YouTube videí je potom možno vkládat modely na vlastní webové stránky (prostřednictvím tagu `<iframe>`). Pro nahrávání modelů je nutno se nejprve registrovat. Alternativou k nové registraci je potom přihlášení pomocí účtu Google.

V rámci diplomové práce bylo zkoumáno, zda by prostřednictvím Sketchfabu nebylo možno přidávat další informace o zobrazovaných objektech, například alespoň takovým způsobem, jaký poskytuje program Spread3D (viz 6.1). Podobná možnost ale bohužel nebyla nalezena. Proto bylo alespoň vyzkoušeno přidávání složitějších modelů, které byl vytvořeny v aplikaci SketchUp, do databáze. Jako zkušební byl využit model Hernychovy vily v Ústí nad Orlicí, který byl vytvořen v rámci bakalářské práce [4] a dále upravován v rámci předmětu *Vizualizace a distribuce prostorových dat*. Jedná se o poměrně složitý model secesní vily, který obsahuje značné množství textur a přitom má ještě poměrně rozumnou velikost (6.57 MB). Pro vyzkoušení služby Sketchfab se tedy jevil jako vhodný.

Sketchfab podporuje import velkého množství 3D formátů, formát `skp`, který je používán programem SketchUp, mezi ně však nepatří. Pro jednoduchý export modelu z aplikace SketchUp lze ale stáhnout plugin Sketchfab Uploader. Ten je přístupný z menu SketchUpu *File* jako položka *Upload to Sketchfab*. V dialogu,

[3] https://sketchfab.com/
[4] http://www.youtube.com/

který se zobrazí po zvolení této položky, je nutno vložit minimálně název modelu a dále také osobní kód, který byl vytvořen při registraci do služby. Dále by už měl proběhnout upload modelu do databáze Sketchfabu. Pokus o export pomocí tohoto pluginu se ale nezdařil.

Po nevydařeném exportu pomocí pluginu v rámci SketchUpu bylo rozhodnuto pokusit se o nahrání modelu v některém z podporovaných formátů. Export do těchto formátů byl potom realizován pomocí osmihodinové zkušební verze programu SketchUp Pro. Nejprve byl na zkoušku uploadován model ve formátu `obj`. Protože ale nebyly textury modelu zobrazeny korektně, byly vyzkoušeny také další formáty, které umí aplikace SketchUp exportovat a naopak služba Sketchfab importovat. Jedná se dále o formáty `dae`, `kmz`, `wrl`, `3ds` a `fbx`. Kromě formátu `kmz` (který je vlastně sám archivem) bylo nutno model a textury nahrávat dohromady v jednom ZIP archivu. Bohužel s žádným formátem se nepodařilo dosáhnout příliš kvalitních výsledků a vždy byly některé textury poškozené, soubor ve formátu `wrl` se potom nepodařilo nahrát vůbec. Problém s texturami přitom pravděpodobně nastal až při zobrazování v rámci WebGL ve Sketchfabu. Model ve formátu `obj` byl totiž importován pro další pokusy do programu CopperCube (viz 6.4) a zde byl zobrazován bez problému. Na obr. 6.2 je možno porovnat výsledky po uploadu na server Sketchfab s modelem ve formátu `obj` exportovaným do `exe` souboru prostřednictvím CopperCube (a zobrazeným správně).

Z výše uvedeného je patrné, že služba Sketchfab umožňuje zajímavý způsob prezentace 3D modelů. Bohužel export z aplikace SketchUp neprobíhá úplně bez problémů. Také možnost propojení objektů modelu s dalšími informacemi zde v této chvíli neexistuje [61].

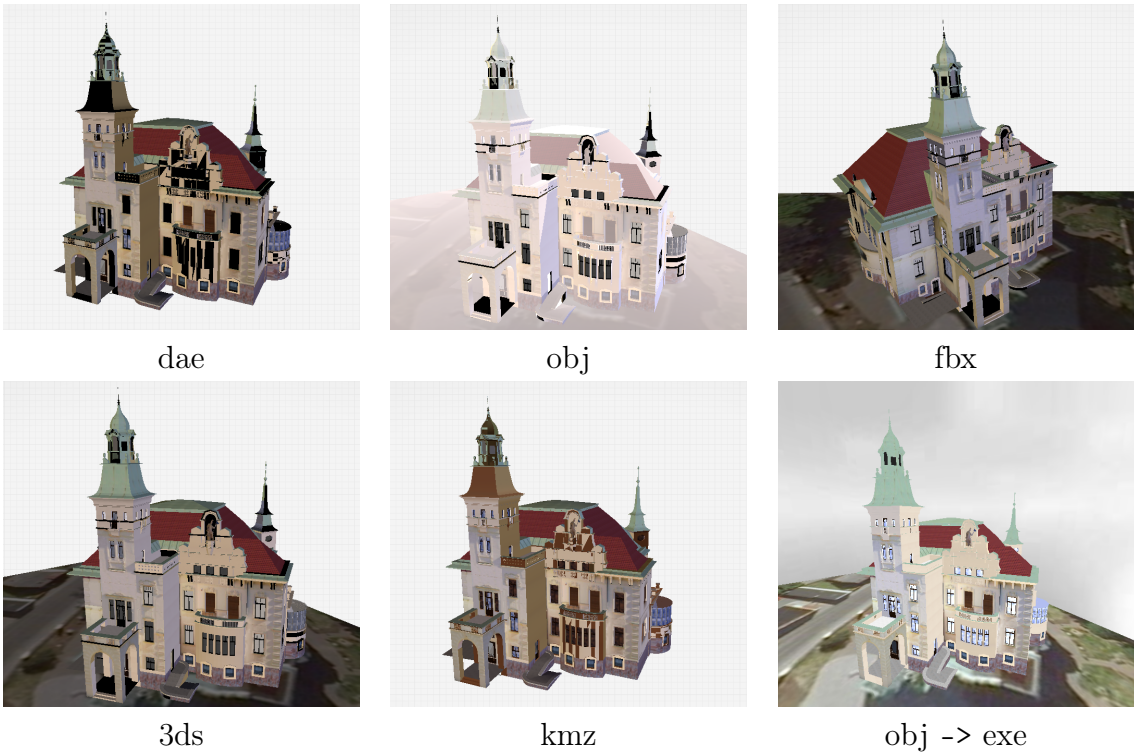
## 6.4 CopperCube 3D

### 6.4.1 Úvod

CopperCube[5] je editor, který je možno využít pro vytváření 3D her a dalších aplikací. Dle stránek výrobce se jednalo o první 3D editor sloužící pro vytváření interaktivních scén ve WebGL. Program využívá vlastní knihovnu CopperLicht. Jak knihovna, tak editor jsou vyvíjeny společností *Ambiera*.

CopperLicht je tedy WebGL knihovna a JavaScriptový 3D engine. Tato knihovna může být používána i samostatně bez editoru CopperCube. Knihovna je dostupná volně ke stažení, pro profesionální využití je ale možno zakoupit komerční licenci.

[5] http://www.ambiera.com/coppercube/




Obr. 6.2: Porovnání modelu v databázi Sketchfab s výstupem ve formě exe souboru

Ta stojí 99 euro na rok a jeden počítač. Uživatel se zakoupenou komerční licencí získá přístup ke zdrojovému kódu knihovny a také lepší podporu a aktualizace po dobu platnosti licence [62].

Editor CopperCube je placeným softwarem. Za jednu neomezenou licenci ve verzi Light zaplatí uživatel 149 euro. Pokročilejší verze Professional, která umožňuje vytvářet větší množství scén v dokumentu a obsahuje pokročilejší rozhraní, potom stojí 380 euro. Pro vyzkoušení funkcí je k dispozici demo verze s neomezenou funkčností, shodnou s Light verzí, na 14 dní. Tato verze (CopperCube 4 Demo) byla vyzkoušena pro účely diplomové práce. Kromě testované verze pro Windows lze zakoupit i CopperCube pro Mac OS X [63].

Program CopperCube se zdál být poměrně zajímavým nástrojem, který by mohl umožnit zobrazování modelů vytvořených v aplikaci SketchUp s využitím WebGL a canvasu. Proto mu bude na stránkách této práce věnováno více prostoru. Popsáno bude také testování tohoto programu jako prostředníka pro vytváření webových prezentací modelů s možností zobrazení dalších informací o jejich objektech.

### 6.4.2 Základní vlastnosti

CopperCube podporuje import souborů ve 22 formátech. Podpora vlastního formátu aplikace SketchUp sice není k dispozici, většinou ale fungoval bez problému přenos dat přes formát `obj`. Méně spolehlivým byl potom formát `dae`, kdy některé modely v tomto formátu odmítal editor importovat. Značnou výhodou formátu `dae` je ale na druhou stranu možnost exportovat ho i z bezplatné verze aplikace SketchUp Make. Model se do programu CopperCube importuje volbou *Import/Static 3D Mesh from File*. Takto importovaný model je potom v rámci editoru jedním objektem a zřejmě neexistuje další vhodná možnost jeho editace (kromě možnosti změny jednotlivých textur).

Exportovat výsledky z programu lze v pěti formátech. Nejzajímavější variantou je asi export pro zobrazování v rámci elementu canvas (tedy volba Publish as WebGL (.html)). Při tomto způsobu exportu je vytvořen adresář, který obsahuje model uložený do souboru `ccbjs` a textury. Dále je mimo adresář vytvořen HTML soubor, ve kterém je model zobrazen v canvasu s využitím JavaScriptu. Další možností publikace, která je vhodná pro web, je export do souboru `swf`. Takový soubor bude zobrazen ve webovém prohlížeči s využitím Adobe Flash Playeru. Možný je i export do samostatného `exe` souboru. Pro využití mimo PC lze exportovat do formátů `apk` (pro OS Android) a `app` (Mac OS X).

Kromě importu hotového modelu z jiného softwaru je možno v rámci editoru vytvářet také nové 3D objekty. Jedná se o jednoduché geometrické útvary jako jsou kvádr, koule, válec, kužel nebo rovina. Pro tvůrce her je také určitě zajímavý nástroj pro jednoduché vytváření chodeb a dalších interiérů, které jsou generovány z 2D mapy. Dále je možno do modelu umisťovat dvourozměrné objekty jako například stromy, editovat lze i „skybox“, tedy pozadí, které dotváří iluzi rozlehlého trojrozměrného prostoru. K dispozici je i nástroj pro tvorbu částicových systémů, který může ve hrách sloužit pro vytvoření efektů, jako je například kouř.

Pro nastavení pohledu na model lze využít různých kamer, které zohledňují jeho další využití. Základní kamera (Simple Camera) nemá žádné další vlastnosti a poskytuje pouze jeden pohled bez možnosti jeho změny. Dalšími možnostmi nastavení kamery jsou:

- Free Flying Camera

  Kamera, kterou lze volně „prolétávat“ model. Ovládána je pomocí šipek klávesnice a myši.

- First Person Shooter Camera

Podobná kamera jako v předchozím případě, pohyb je ale umožněn pouze po plochách, případně po schodech. Na kameru působí efekt „gravitace“, pomocí mezerníku lze také skákat. Je tak zajištěn vjem jako v počítačových hrách z pohledu první osoby.

- Third Person Camera

  Kamera, která sleduje objekt, jehož pohyb je řízen uživatelem. Uplatní se zejména u her z pohledu třetí osoby.

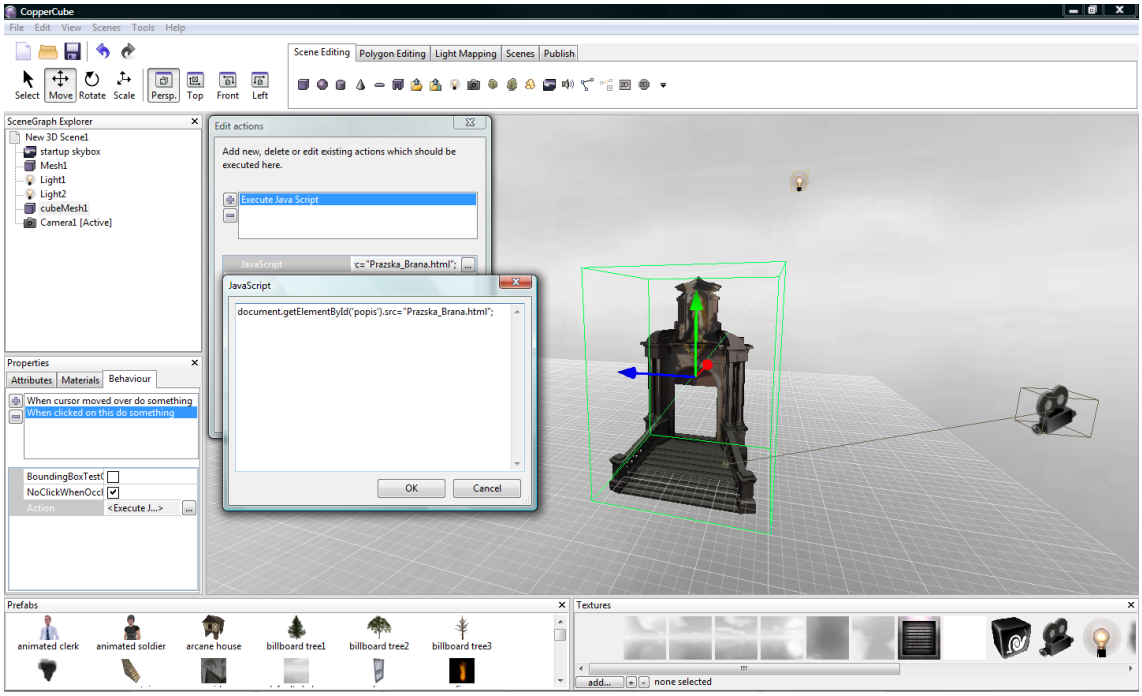
- Model Viewer Camera

  Tato kamera je ovládána myší a pohybuje se po sféře s daným poloměrem kolem sledovaného objektu. Nastavit lze i možnost zoomování.

- Panorama Camera

  Kamera, která se nepohybuje. Její pohled je ale možno ovládat myší.

Pokud budeme uvažovat například prohlížení modelu památkového objektu, bude asi nejvýhodnější možností Free Flying Camera nebo případně Model Viewer Camera. Pro procházení vnitřních prostor potom může být vhodná First Person Shooter Camera.




Obr. 6.3: Práce v prostředí editoru CopperCube

U každého objektu modelu lze v CopperCube provádět nastavení na třech záložkách. Záložka *Attributes* zpřístupňuje nastavení polohy a rozměrů objektu, jeho

natočení, viditelnost a podobně. V záložce *Materials* se provádí nastavení textur objektu. Nejzajímavější je potom záložka *Behaviour*. Na té lze nastavit chování objektu v rámci scény. Nejprve je možno nastavit pohyb objektu. Objekt se může pohybovat v kruhu, po určité trajektorii, může rotovat atd. Dále je možno zvolit chování objektu při kolizi s jiným objektem scény. K dispozici jsou také čistě herní volby chování, kdy může být objekt postavou v rámci hry. Tato postava může být řízena buď hráčem nebo počítačem s předdefinovanou umělou inteligencí (AI). Pro účely „pouhého" prohlížení objektů a zobrazování dalších informací o nich je asi nejzajímavější volba *Behaviors triggered by events*. V rámci této položky lze nastavovat chování objektů například v případě, že je na ně kliknuto myší nebo se přes ně kurzor myši pohybuje.

### 6.4.3 Testování funkcí

Do editoru CopperCube byly zkusmo importovány dva složitější modely – model Hernychovy vily a dále model Svaté Hory v Příbrami. Model Hernychovy vily se podařilo importovat s využitím formátu `obj` bez problémů včetně všech použitých textur. Proto byl vyzkoušen export tohoto modelu do `exe` souboru, do formátu `swf` a do formátu `ccbjs` pro zobrazování v canvasu. Export do samostatného `exe` souboru proběhl v pořádku a model bylo možno bez potíží prohlížet. Naproti tomu ve Flash Playeru (formát `swf`) se model zobrazil jako značně rozbitý a nepoužitelný. Pro zobrazování modelu v canvasu bylo nejprve nutno vyřešit problém s webovými prohlížeči, který bude popsán dále. Potom se již model zobrazil poměrně v pořádku, některé textury (v tomto případě textury oken) byly ale zaměněny za jiné. Také vykreslování probíhalo poměrně pomalu a pro nerušené plynulé prohlížení by zřejmě byl třeba výkonnější počítač. Model Svaté Hory se do editoru nepodařilo ani správně importovat, chybělo velké množství prvků modelu. Je otázkou, zde se jedná opravdu o problém importu do CopperCube nebo naopak o nesprávný export do `obj` ze SketchUpu Pro. Obojí může být způsobeno značnou podrobností a složitostí dotyčného modelu.

Jak již bylo řečeno, při pokusech se zobrazováním modelů v canvasu v rámci HTML stránky se objevily problémy s webovými prohlížeči. Google Chrome, který byl nejdříve použit, neumožňuje standardně z bezpečnostních důvodů otevírat lokální soubory. Tento problém byl popsán na stránkách výrobce editoru CopperCube [64]. Zde je také k dispozici postup k ošetření tohoto problému, který spočívá v doplnění parametru *–allow-file-access-from-files* do adresy cíle ve vlastnostech zástupce prohlížeče Chrome. Vytvořené HTML stránky je potom nutno zobrazovat v okně prohlížeče otevřeném pomocí tohoto zástupce. Tento postup ale nebyl plně funkční.

Proto byl pro další pokusy s lokálně umístěnými soubory používán prohlížeč Firefox. I ten ale bylo nutno nejprve nastavit. Nejprve bylo povoleno použití hardwarové akcelerace zaškrtnutím příslušné položky (Firefox/Možnosti/Rozšířené/Obecné/Použít hardwarovou akceleraci, je-li dostupná). Další možnosti nastavení bylo nutno otevřít napsáním příkazu `about:config` do adresního řádku prohlížeče. V tabulce, která byla zobrazena, bylo potom třeba dvojklikem na položku `webgl.force-enabled` změnit její hodnotu na `true` [65].

Pro demonstraci dalších funkcí programu CopperCube byl raději použit pouze jednoduchý model a to zejména kvůli pomalému vykreslování modelu v canvasu, které by při pokusech bylo na obtíž. Tento model představuje malou část areálu Svaté Hory v Příbrami, portál Pražské brány. Protože je tato část modelu vytvořena jako komponenta, bylo snadné ji oddělit od celku. Jedná se sice o velmi malý model, pro ukázku ale plně dostačuje. Cílem této ukázky bylo vytvořit jednoduchou HTML stránku, která bude obsahovat canvas se zobrazeným modelem. Po kliknutí na model v canvasu se potom zobrazí další stránka s informacemi o modelu.

Aby uživatel věděl, na jaký objekt může pro zobrazení informací kliknout, je vhodné, aby u takových objektů docházelo ke zvýraznění pod kurzorem myši. V našem jednoduchém příkladě je vlastně jen jeden objekt zájmu, tedy Pražská brána. Ve skutečnosti by ale samozřejmě byly zobrazovány rozsáhlejší modely, kde by bylo více objektů s dalšími informacemi. V CopperCube ale bude celý model importovaný ze SketchUpu vždy jen jedním objektem. Aby bylo možné zvýraznit pouze část modelu, byl okolo vybrané části vytvořen pomocný kvádr. Ten je normálně potažen průhlednou texturou a je tedy neviditelný. V záložce *Behaviour* mu byla nastavena vlastnost *Behaviors triggered by events/When cursor moved over do something/-Change a texture*. Pokud se kurzor ocitne nad kvádrem, změní se textura kvádru. Každá stěna kvádru je potom potažena průhlednou texturou s červeným viditelným ohraničením. Tím se vlastně zvýrazní zájmový objekt, který je uvnitř kvádru. Když kurzor opustí kvádr, změní se textura opět na zcela průhlednou. V případě, že by byl objekt zájmu pouze rovinný (třeba prvek na fasádě), je možné okolo něj místo celého kvádru vytvořit pouze plochu se stejnou texturou. Textury s průhledností ve formátu `png` byly vytvořeny v programu GIMP.

Podle toho, že se objekt zvýrazní, zjistí uživatel, že jsou o objektu dostupné další informace. Ty si dále může zobrazit kliknutím na objekt. To umožňuje vlastnost *When clicked on this do something*. Pro zobrazování textu přímo v náhledu modelu uvnitř canvasu jsou v CopperCube k dispozici speciální 2D objekty, které lze využít jako textová pole (2D overlay items). Zpřístupňovat delší popis tímto způsobem by ale nebylo příliš přehledné. Další možností je vytvořit vlastní HTML

stránku pro popisovaný objekt, kde se mohou objevit i obrázky, další odkazy atd. Přechod na takovou stránku zabezpečuje vlastnost *When clicked on this do something/Special/Open a website*. Navzdory tomu, co je uvedeno na stránkách výrobce, lze takto přejít i na webovou stránku umístěnou lokálně, na kterou je odkazováno pouze relativně.

Zobrazení HTML stránky, která je otevírána výše popsaným způsobem, bohužel zabrání prohlížeč v rámci blokování vyskakovacích oken. Uživatel samozřejmě může stránku přesto otevřít a uvedený problém by bylo možno řešit nastavením v rámci prohlížeče. Stejně ale není popsaný způsob úplně elegantní, protože uživatel musí pro zobrazení stránky s popisem přejít do nového okna nebo panelu prohlížeče. Proto by mohlo být vhodné zobrazovat stránku s popisem jako vnořenou vedle vlastního canvasu s modelem.

HTML stránka, kterou vygeneruje editor CopperCube, obsahuje pouze element canvas s náhledem na model. Proto bylo nutno tuto stránku nejprve upravit a přidat do ní plovoucí rám, ve kterém bude zobrazována stránka s popisem (tag `<iframe>`). Upravená stránka může v nejjednodušším případě vypadat např. následovně:

Ukázka kódu 6.1: Upravený HTML výstup z editoru CopperCube

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Language" content="cs">
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
        <script type="text/javascript" src="copperlichtdata/copperlicht.js"></
            script>
        <title>Svatá Hora v Příbrami</title>
</head>
<body>
        <div align="center">
        <h1>Svatá Hora v Příbrami</h1>
        <table>
                <tr><td><div align="center">
                        <canvas id="3darea" width="640" height="480" style="
                            background-color:#000000">
                        </canvas>
                </div>
                <script type="text/javascript">
                <!--
                                startCopperLichtFromFile('3darea', 'copperlichtdata
                                    /prazska_brana_ccb.ccbjs', 'Loading $PROGRESS$
                                    ...', 'Error: This browser does not support
                                    WebGL (or it is disabled).<br/>See <a href=\"
                                    http://www.ambiera.com/copperlicht/
                                    browsersupport.html\">here</a> for details.');
                -->
                </script>
                </td>
                <td>
                <iframe id="popis" width="400" height="480">
```

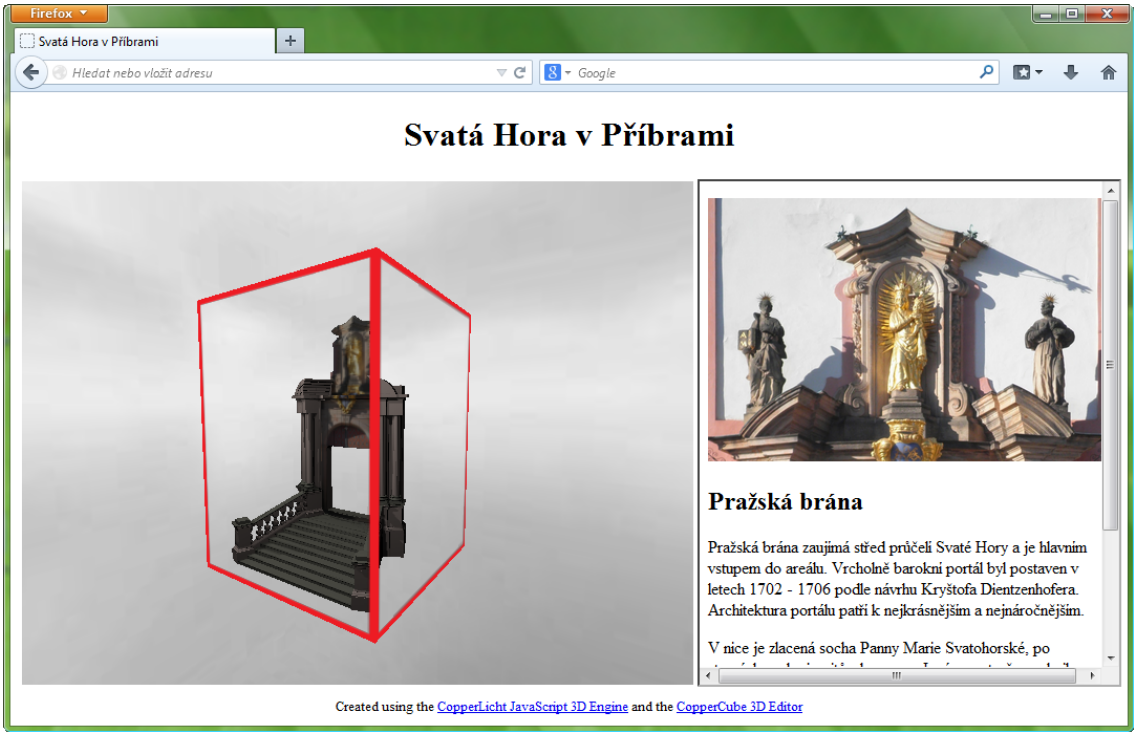
```
                </iframe>
                </td></tr>
        </table>
        <small>Created using the <a href="http://www.ambiera.com/copperlicht/index.
            html">CopperLicht JavaScript 3D Engine</a> and the <a href="http://www
            .ambiera.com/coppercube/index.html">CopperCube 3D Editor</a>
        </small>
        <div>
</body>
</html>
```

Část obsahu této stránky tedy byla vygenerována pomocí CopperCube. Je tedy vytvořen canvas s id `3darea` a poté je zavolána funkce `startCopperLichtFromFile`. Pomocí té je zobrazen model ve formátu `ccbjs`. V rámci této funkce je vytvořena i načítací obrazovka a text, který zobrazuje počet načtených textur. Dále je ošetřeno i chování v případě snahy o zobrazení modelu v prohlížeči, který nepodporuje WebGL.

K vygenerovanému HTML potom byl v textovém editoru přidán iframe, který se zobrazuje vedle canvasu s vlastním modelem. Tento iframe má id `popis` a dokud uživatel neklikne na objekt zájmu, zůstane prázdný. Nastavení zdroje iframu je řešeno pomocí JavaScriptu. Ten byl nastaven v CopperCube ve vlastnostech objektu, na který je možno kliknout, tedy v našem případě ve vlastnostech kvádru, který slouží jako obal objektu zájmu (zde Pražské brány). Nastavení proběhlo opět na záložce *Behaviour* a nastavena byla vlastnost *When clicked on this do something/Special/Execute Java Script*. Použitý JavaScript potom vypadá následovně: `document.getElementById('popis').src="Prazska_Brana.html"`. Takto je tedy po kliknutí na kvádr nastavena pro iframe cesta k HTML stránce s popisem.

### 6.4.4 Shrnutí

Jak vypadá výsledná HTML stránka je ukázáno na obr. 6.4. V rámci vnořené stránky s popisem je v tomto případě vložen i obrázek. Pokud by v modelu bylo více objektů s dalšími informacemi, nebyl by samozřejmě problém vytvořit pro každý z nich vlastní vnořené HTML stránky, na které by bylo odkazováno pomocí dalších pomocných objektů (tedy kvádrů, případně ploch, které po kliknutí aktivují JavaScript). Je možné říci, že uvedený způsob prezentování modelu by mohl být velmi zajímavý. Umožňuje totiž sdílet model na webu a přehlednou formou zobrazovat další informace o jednotlivých objektech. Jak vlastní model, tak i popis mohou být zobrazeny vedle sebe na jedné webové stránce a pro zobrazení modelu uživatelem




Obr. 6.4: Prezentace modelu v rámci HTML stránky s využitím editoru CopperCube

není třeba žádný doplněk prohlížeče. S využitím vhodných kamer lze pohodlně procházet i vnitřní prostory objektů. Pokud by byl výše uvedený postup rozšířen, mělo by dále být možné v rámci canvasu také přecházet mezi více uloženými modely.

Mezi nevýhody použití editoru CopperCube spadá zejména nutnost vytvářet pomocné prvky pro zvýraznění jednotlivých zájmových objektů a pro jejich vybírání. To je způsobeno tím, že je ze SketchUpu importovaný model v rámci CopperCube pouze jedním objektem (tedy jedním celkem). U rozsáhlých modelů to může být zdlouhavé. Další nevýhodou je nutnost alespoň základní znalosti HTML a JavaScriptu při vytváření takových prezentací. Ty jsou nutné i přesto, že je v rámci editoru CopperCube téměř vše možno nastavovat pouze klikáním v příslušných menu. Pokud bude zobrazován velký a podrobný model, může také nastat problém s jeho pomalým vykreslováním ve webovém prohlížeči, jak již bylo řečeno výše. Tento problém sice nenastane při exportu do samostatně spustitelného `exe` souboru, zde jsou ale zase omezeny možnosti přímé webové prezentace a zobrazování dalších informací mimo vlastní okno s modelem. Jako poslední, ale nezanedbatelnou nevýhodu je nutno uvést to, že je editor CopperCube placeným softwarem. Samotná knihovna CopperLicht je sice zdarma, její použití bez editoru ale vyžaduje samozřejmě mnohem větší znalost práce s JavaScriptem.

# Závěr

V této práci byly nejprve představeny možnosti rozšíření aplikace SketchUp o další funkce pomocí zásuvných modulů (pluginů), které jsou vytvořeny s využitím rozhraní SketchUp Ruby API. Popsány byly také atributy objektů modelu v aplikaci SketchUp a jejich správa pomocí tohoto rozhraní. Dále byly předvedeny některé zásuvné moduly, které umožňují propojovat model v aplikaci SketchUp s dalšími informacemi. Většina z nich využívá právě nastavování atributů jednotlivým objektům.

Nalezené moduly tedy umožňují propojit objekty ve SketchUpu s dalšími informacemi. S využitím jednoduchých funkcí je u pluginů Links Manager a GOSU možno odkazovat na soubory umístěné lokálně i na dokumenty na webu po kliknutí na vybraný objekt modelu. Protože nejpřehlednější možností pro zobrazování dalších informací je asi HTML stránka, předpokládá se od nastavujících uživatelů znalost jazyka HTML. Pluginy tedy slouží opravdu pouze pro vytvoření odkazů na ručně vytvořený dokument.

Stejná je situace u rozšíření Museum/Gallery HTML Reference. Tento plugin využívá HTML stránku jako zdroj informací i jako prostředek pro ovládání modelu (ovládání pohledu, viditelnosti vrstev). Zde ale chybí možnost zobrazovat informace po kliknutí na vybraný objekt. Asi nejzajímavějším pluginem je tedy oficiální rozšíření Dynamické komponenty. Jeho plnou funkčnost lze ale využít až s placenou verzí SketchUp Pro. Dále zde chybí možnost odkazovat na lokální soubory.

Aby bylo možné realizovat vlastní představu o fungování rozšíření pro nastavování a zobrazování dalších informací, byl tedy vytvořen vlastní plugin TIS. Informace o objektech jsou zde zobrazovány v rámci HTML stránky uvnitř dialogového okna SketchUpu. Plugin obsahuje dále nástroj pro přehled všech objektů s dalšími informacemi v modelu. Kromě toho byly přidány funkce umožňující nastavovat a poměrně přehledně prohlížet informace jak u komponent, tak u objektů uvnitř těchto komponent. Protože je HTML stránka s popisem generována v rámci pluginu uvnitř programu SketchUp, nemusí vlastně nastavující uživatel ovládat jazyk HTML. Jeho základní znalost lze ale využít pro formátování popisu jednotlivých objektů. I ve tvorbě webových stránek nezkušený uživatel může využít funkcí pluginu k přiřazení obrázku, podrobnějšího modelu a popisu.

Protože je dnes v hojné míře využíváno webové prezentace, byly poslední kapitoly věnovány exportu modelu na web. Lze předpokládat, že se v tomto směru budou rozvíjet hlavně aplikace založené na novém standardu HTML5 a na rozhraní WebGL, které nepotřebují pro prohlížení zásuvný modul prohlížeče. Zatím nebylo nalezeno moc programů, které by umožňovaly export na web s přidáním dalších informací

bez nutnosti důkladné znalosti programování v JavaScriptu. Z tohoto pohledu se jeví jako velmi zajímavý editor CopperCube. To je poměrně silný nástroj, s jehož pomocí je možné i bez většího programování vytvářet pěkné webové prezentace modelů. Prohlížení rozsáhlejších modelů je ale omezeno výkonem použitého počítače. Kromě toho je CopperCube placeným softwarem, což také snižuje jeho atraktivitu pro použití např. v akademických pracích. Webová služba Sketchfab umožňuje také prezentaci modelu s využitím HTML5 elementu canvas a WebGL. Přidávání dalších informací zde ale zatím není možné. Další testovaná aplikace – Spread3D v současné době prochází vývojem a lze zatím jen spekulovat o jejím fungování v nové verzi.

Program SketchUp se tedy s využitím výše uvedených možností stává základem jednoduchého informačního systému. V rámci vlastního zásuvného modulu TIS probíhá nastavování a zobrazování dalších informací přímo uvnitř programu SketchUp. Ve webových prezentacích, které jsou vytvořeny pomocí uvedených aplikací, je potom SketchUp základem pouze ve smyslu nástroje pro vytvoření 3D modelu. Další přiřazování informací probíhá pomocí jiných programů.

Zajímavé by bylo, kdyby bylo možno nastavovat další informace v rámci zásuvného modulu v programu SketchUp a model potom včetně nastavených informací přímo exportovat do formátu použitelného pro webovou prezentaci. To předpokládá vytvoření programu schopného provést export ze SketchUpu a hlavně také aplikaci, která bude umět s výsledkem na webu pracovat. Pro vytváření modulů, které slouží pro export z programu SketchUp by mělo sloužit rozhraní SketchUp Importer/Exporter Interface[6]. Prostudování exportu z aplikace SketchUp touto cestou je tedy možno doporučit pro další práci, která se bude podobným tématem zabývat. Zde navržený plugin TIS může v takovém případě sloužit také jako podklad při vytváření modulu pro nastavování atributů v aplikaci SketchUp před exportem pro prohlížení na webu.

Pro prezentaci výsledků diplomové práce byly vytvořeny jednoduché webové stránky. Na těchto stránkách je ke stažení zásuvný modul TIS včetně návodu a vzorového modelu. Dále je zde možno prohlížet zkušební výstupy z některých testovaných programů. Nakonec jsou uvedeny odkazy na stránky testovaných zásuvných modulů i na stránky samostatných aplikací. Webové stránky jsou k dispozici buď offline na CD v příloze práce nebo na adrese: http://geo3.fsv.cvut.cz/dp/tobias.

[6]http://www.sketchup.com/intl/en/developer/sdk_start.html

# Použité zdroje

[1] 3D Modeling for Google Earth & Maps. *Skupiny Google* [online]. [cit. 2013-10-30]. Dostupné z URL: <https://groups.google.com/forum/?hl=cs#!forum/3dwh>.

[2] SketchUp. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 3 November 2013 [cit. 2013-11-13]. Dostupné z URL: <http://en.wikipedia.org/wiki/SketchUp>.

[3] *SketchUp | 3D for Everyone* [online]. Trimble Navigation Limited, © 2013 [cit. 2013-11-13]. Dostupné z URL: <http://www.sketchup.com/>.

[4] TOBIÁŠ, Pavel. *3D model Hernychovy vily v Ústí nad Orlicí*. Praha, 2012. Bakalářská práce. ČVUT v Praze, Fakulta stavební, Katedra mapování a kartografie. Vedoucí práce Doc. Ing. Lena Halounová, CSc.

[5] Plugin. *Wikipedie, otevřená encyklopedie* [online]. naposledy editováno 13. 7. 2013 [cit. 2013-09-30]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Plugin>.

[6] LININGER, Scott. RBZs in SketchUp 8M2: Distribute Your Plugin as One File!. In: *SketchUp API Blog: Tips and Tricks for SketchUp Ruby API Developers*. [online]. December 1, 2011 [cit. 2013-09-30]. Dostupné z URL: <http://sketchupapi.blogspot.cz/2011/12/rbzs-in-sketchup-8m2-distribute-your.html>.

[7] Frequently Asked Questions (FAQ) SketchUp Ruby API. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-30]. Dostupné z URL: <http://www.sketchup.com/intl/en/developer/docs/faq.php>.

[8] Attribute Reporting SketchUp Ruby API. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-18]. Dostupné z URL: <http://www.sketchup.com/intl/en/developer/docs/tutorial_attrreporting.php>.

[9] SketchUp Ruby API: Developer Docs. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-22]. Dostupné z URL: <http://www.sketchup.com/intl/en/developer/index.php>.

[10] Sketchupattributemanager: Manage Google SketchUp Attributes. *Google Code* [online]. [cit. 2013-09-23]. Dostupné z URL: <https://code.google.com/p/sketchupattributemanager/>.

[11] Links Manager | SketchUp Extension Warehouse. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://extensions.sketchup.com/en/content/links-manager>.

[12] GOSU | SketchUp Extension Warehouse. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://extensions.sketchup.com/en/content/gosu>.

[13] Dynamic Components supported HTML tags | SketchUp Knowledge Base. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://help.sketchup.com/en/article/115374>.

[14] Dynamic Components Frequently Asked Questions | SketchUp Knowledge Base. *SketchUp | 3D for Everyone* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://help.sketchup.com/en/article/94885>.

[15] The SketchUp Show #56: Using Dynamic Component Appliances in the Kitchen - YouTube. *You Tube* [online]. Nahráno 10. 06. 2009 [cit. 2013-09-23]. Dostupné z URL: <http://www.youtube.com/watch?v=5vqS86TGhMQ>.

[16] Creating Dynamic Components -1 - 3 - YouTube. *You Tube* [online]. Nahráno 11. 11. 2008 [cit. 2013-09-23]. Dostupné z URL: <https://www.youtube.com/watch?v=fsBpIPnF31A>.

[17] *Morisdov* [online]. [cit. 2013-09-23]. Dostupné z URL: <https://sites.google.com/site/morisdov/home>.

[18] PAPASMADOV, Moris. *General Locator Tutorial*. 2010.

[19] PAPASMADOV, Moris. *Museum-Gallery Reference Tutorial*. 2010.

[20] Frequently Asked Questions. *WalkAbout3d* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://www.walkabout3d.com/faq.php>.

[21] *WalkAbout3d* [online]. [cit. 2013-09-23]. Dostupné z URL: <http://www.walkabout3d.com/index.php>.

[22] Ruby pro začátečníky. In: *AbcLinuxu.cz - Linux na stříbrném podnose* [online]. 8. 2. 2006 [cit. 2013-10-05]. Dostupné z URL: <http://www.abclinuxu.cz/clanky/programovani/ruby-pro-zacatecniky-1>.

[23] Ruby (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 8. 3. 2013 [cit. 2013-10-05]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Ruby_(programovac%C3%AD_jazyk)>.

[24] Ruby (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 15 October 2013 [cit. 2013-10-19]. Dostupné z URL: <http://en.wikipedia.org/wiki/Ruby_(programming_language)>.

[25] *Ruby-Doc.org: Documenting the Ruby Language* [online]. [cit. 2013-10-19]. Dostupné z URL: <http://ruby-doc.org/>.

[26] THOMAS, Dave a Andy HUNT. *Programming Ruby: The Pragmatic Programmer's Guide* [online]. © 2001 [cit. 2013-10-05]. Dostupné z URL: <http://www.ruby-doc.org/docs/ProgrammingRuby/>.

[27] SLAGELL, Mark. *Ruby User's Guide* [online]. (c) 2005-2008 [cit. 2013-10-05]. Dostupné z URL: <http://www.rubyist.net/ slagell/ruby/index.html>.

[28] *Code School - TryRuby* [online]. [cit. 2013-10-05]. Dostupné z URL: <http://tryruby.org/levels/1/challenges/0>.

[29] Objects Diagram SketchUp Ruby API. *SketchUp / 3D for Everyone* [online]. [cit. 2013-10-05]. Dostupné z URL: <http://www.sketchup.com/intl/en/developer/docs/diagram.php>.

[30] BAYER, Tomáš. *Úvod do OOP* [online]. [cit. 2013-10-19]. Dostupné z URL: <http://web.natur.cuni.cz/~bayertom/Prog2/prog2_1.pdf>.

[31] Průvodce po Svaté Hoře. *Svatá Hora* [online]. © 2003-2009 [cit. 2013-11-11]. Dostupné z URL: <http://svata-hora.cz/cz/32/pruvodce-po-svate-hore>.

[32] Průvodce po Svaté Hoře (Virtuální prohlídka). *Svatá Hora* [online]. © 2003-2009 [cit. 2013-11-11]. Dostupné z URL: <http://pruvodce.svata-hora.cz/>.

[33] SWF - Flash glossary | Adobe Developer Connection. *Adobe* [online]. © 2013 [cit. 2013-11-15]. Dostupné z URL: <http://www.adobe.com/devnet/flash/articles/concept_swf.html>.

[34] SWF. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 13. 9. 2013 [cit. 2013-11-15]. Dostupné z URL: <http://cs.wikipedia.org/wiki/SWF>.

[35] ActionScript. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 7 November 2013? [cit. 2013-11-15]. Dostupné z URL: <http://en.wikipedia.org/wiki/ActionScript>.

[36] Plugin Adobe Flash Player - Nápověda Chrome. *Prohlížeč Chrome* [online]. ©2013 [cit. 2013-11-15]. Dostupné z URL: <https://support.google.com/chrome/answer/108086>.

[37] HTML5. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2. 9. 2013 [cit. 2013-10-26]. Dostupné z URL: <http://cs.wikipedia.org/wiki/HTML5>.

[38] HTML5 Introduction. *W3Schools Online Web Tutorials* [online]. © 1999-2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.w3schools.com/html/html5_intro.asp>.

[39] JANOVSKÝ, Dušan. Javascript - návody, objekty, příklady. *Jak psát web, návod na html stránky* [online]. [cit. 2013-11-12]. Dostupné z URL: <http://www.jakpsatweb.cz/javascript/>.

[40] JavaScript. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 11 November 2013? [cit. 2013-11-12]. Dostupné z URL: <http://en.wikipedia.org/wiki/JavaScript>.

[41] PILGRIM, Mark. *Dive into HTML5* [online]. Martin Hassman et al. © MMIX–MMXI [cit. 2013-10-26]. Dostupné z URL: <http://kniha.html5.cz/index.html>.

[42] OpenGL Shading Language. *OpenGL: The Industry Standard for High Performance Graphics* [online]. © 1997 - 2013 [cit. 2013-10-28]. Dostupné z URL: <http://www.opengl.org/documentation/glsl/>.

[43] WebGL (Windows). *Internet Explorer Dev Center* [online]. © 2013 [cit. 2013-10-26]. Dostupné z URL: <http://msdn.microsoft.com/en-us/library/ie/bg182648(v=vs.85).aspx>.

[44] CABALLERO, Luz. An introduction to WebGL. *Dev.Opera: Distilled knowledge for web developers* [online]. October 13, 2011 [cit. 2013-10-28]. Dostupné z URL: <http://dev.opera.com/articles/view/an-introduction-to-webgl/>.

[45] WebGL. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 10. 10. 2013 [cit. 2013-10-26]. Dostupné z URL: <http://cs.wikipedia.org/wiki/WebGL>.

[46] ŽÁRA, Ondřej. Vytváříme Hello World pro WebGL. *Zdroják: O tvorbě webových stránek a aplikací* [online]. 15. 5. 2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.zdrojak.cz/clanky/vytvarime-hello-world-pro-webgl/>.

[47] TIŠNOVSKÝ, Pavel. Seriál Grafická knihovna OpenGL. In: *Root.cz: Informace nejen ze světa Linuxu* [online]. 2003 - 2004 [cit. 2013-10-28]. Dostupné z URL: <http://www.root.cz/serialy/graficka-knihovna-opengl/>.

[48] Wavefront .obj file. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 27 September 2013? [cit. 2013-10-30]. Dostupné z URL: <http://en.wikipedia.org/wiki/Wavefront_.obj_file>.

[49] CABALLERO, Luz. Porting 3D graphics to the web: WebGL intro part 2. *Dev.Opera: Distilled knowledge for web developers* [online]. November 24, 2011 [cit. 2013-10-30]. Dostupné z URL: <http://dev.opera.com/articles/view/porting-3d-graphics-to-the-web-webgl-intro-part-2/>.

[50] *COLLADA: Digital Asset and FX Exchange Schema* [online]. © 2013 [cit. 2013-10-30]. Dostupné z URL: <https://collada.org/>.

[51] COLLADA. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 29 October 2013? [cit. 2013-10-30]. Dostupné z URL: <http://en.wikipedia.org/wiki/COLLADA>.

[52] Models - Keyhole Markup Language. *Google Developers* [online]. February 24, 2012 [cit. 2013-10-30]. Dostupné z URL: <https://developers.google.com/kml/documentation/models>.

[53] TIŠNOVSKÝ, Pavel. VRML: jazyk pro popis virtuální reality. In: *Root.cz: Informace nejen ze světa Linuxu* [online]. 8. 11. 2007 [cit. 2013-10-30]. Dostupné z URL: <http://www.root.cz/clanky/vrml-jazyk-pro-popis-virtualni-reality/>.

[54] VRML. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 24 September 2013? [cit. 2013-10-30]. Dostupné z URL: <http://en.wikipedia.org/wiki/VRML>.

[55] .3ds. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 8 October 2013? [cit. 2013-10-30]. Dostupné z URL: <http://en.wikipedia.org/wiki/.3ds>.

[56] FBX binary file format specification. *Blender Code: Developer musings on Blender* [online]. Aug 10, 2013 [cit. 2013-10-30]. Dostupné z URL: <http://code.blender.org/>.

[57] *Spread3D* [online]. © 2013 [cit. 2013-09-23]. Dostupné z URL: <http://www.spread3d.com/>.

[58] Spread3D Forum. *Spread3D* [online]. [cit. 2013-11-01]. Dostupné z URL: <http://products.spread3d.com/forum>.

[59] Hyperlink Tools for SketchUp. *Hypercosm - Interactive, Web-Based, 3D Simulations* [online]. ©2011 [cit. 2013-09-23]. Dostupné z URL: <http://www.hypercosm.com/features/hyperlink_tools_for_sketchup.html>.

[60] Hypercosm Teleporter. *Hypercosm - Interactive, Web-Based, 3D Simulations* [online]. ©2011 [cit. 2013-09-23]. Dostupné z URL: <http://www.hypercosm.com/teleporter.html>.

[61] *Sketchfab | Publish, share and embed interactive 3D models* [online]. [cit. 2013-10-26]. Dostupné z URL: <https://sketchfab.com/>.

[62] CopperLicht License. *Ambiera e.U. Software Development* [online]. © 2009-2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.ambiera.com/copperlicht/license.html#commercial>.

[63] Buy CopperCube - order online. *Ambiera e.U. Software Development* [online]. © 2009-2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.ambiera.com/coppercube/buy.html>.

[64] CopperCube Documentation. *Ambiera e.U. Software Development* [online]. © 2009-2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.ambiera.com/coppercube/doc/index.html>.

[65] Enable or Disable WebGL in FireFox. *InfewBytes* [online]. © 2013 [cit. 2013-10-26]. Dostupné z URL: <http://www.infewbytes.com/?p=144>.

# Seznam obrázků

# Seznam příloh

# A Plugin TIS – návod

Následující kapitola je návodem k pluginu TIS. Tento návod, který je jinak k dispozici v rámci pluginu, byl upraven pro prohlížení v textu diplomové práce.

## A.1 Popis programu

Plugin TIS slouží pro nastavování a prohlížení dalších informací o objektech modelu v programu Trimble SketchUp. K tomu využívá možnost přidávat každému objektu v aplikaci SketchUp atributy. Program obsahuje čtyři nové nástroje. Nástroj *Nastavení* umožňuje nastavovat další informace o objektech (tedy přiřazovat název, obrázek, podrobnější model a popis). Jejich prohlížení je potom zajištěno nástrojem *Výběr*. Nástroj *Tabulka* zobrazí přehled všech objektů, kterým byly další informace přiřazeny. Nakonec nástroj *Přehled atributů* zobrazuje všechny atributy objektu a může sloužit pro kontrolu nastavených atributů, včetně atributů nastavených jiným způsobem (jiným pluginem).

## A.2 Systémové požadavky

Požadovaný výkon počítače závisí na velikosti a podrobnosti modelu. Minimální konfigurace pro prohlížení modelů rozsáhlejších objektů:

- **Operační systém:** (Windows XP SP3), Windows Vista SP2, Windows 7 [1]
- **Procesor:** Intel Core 2 (2.00 GHz)
- **Paměť:** 2 GB
- **Grafická karta (paměť):** 256 MB
- **Verze programu SketchUp:** 8, 2013

Pro správné fungování pluginu je nutno mít nainstalován prohlížeč Internet Explorer[2]. Plugin nepodporuje české znaky v cestě k souborům. Model, v rámci kterého budete přidávat další informace, uložte tak, aby v absolutní cestě k souboru nebyly žádné české znaky. Stejně postupujte při ukládání dalších souborů, na které bude v modelu odkazováno.

[1] Při použití OS Windows XP se mohou objevit problémy s otevíráním obrázků (viz A.6.1). Ve verzi aplikace SketchUp pro Mac OS X nebyl plugin testován.

[2] Plugin byl testován s verzí Internet Explorer 9.

## A.3 Instalace programu

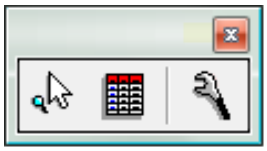
Komprimovaný soubor s příponou .rbz obsahuje hlavní soubor pluginu i všechny potřebné soubory pomocné. K instalaci využijte standardní nástroje aplikace SketchUp:

1. Otevřete okno *System Preferences* (Window/Preferences).
2. Na záložce *Extensions* vyberte volbu *Install Extension*.
3. Najděte soubor tis.rbz a stiskněte *Otevřít*.
4. Zmáčkněte *Ano* v dalším dialogovém okně.

Všechny potřebné soubory a složky pluginu budou takto zkopírovány do adresáře *Plugins* programu SketchUp. Na záložce *Extensions* můžete také volit, zda je plugin používán nebo ne. Pro aktivaci pluginu zaškrtněte příslušné políčko v přehledu nainstalovaných rozšíření.

## A.4 Menu a panel nástrojů

Všechny nástroje programu jsou dostupné z vlastního submenu (Plugins/TIS). Pro pohodlnější přístup k funkcím slouží panel nástrojů. Tento panel nástrojů se zobrazí automaticky po instalaci pluginu nebo po jeho aktivaci. Jinak je možno panel nástrojů zobrazit ze submenu *Toolbars* (View/Toolbars/TIS). Panel nástrojů obsahuje tři nástroje – *Výběr*, *Tabulka* a *Nastavení*. Pro přístup k nástroji *Přehled atributů* použijte submenu (Plugins/TIS/Přehled atributů).



Obr. A.1: Panel nástrojů – nástroje *Výběr*, *Tabulka* a *Nastavení*

## A.5 Nastavení dalších informací

Nástroj *Nastavení* umožňuje každému objektu modelu v programu SketchUp přiřadit název, obrázek, podrobnější model a popis. Obrázek, model ve formátu `skp` a popis ve formátu `txt` jsou samostatnými soubory a jsou uloženy vedle souboru modelu, v rámci kterého jsou nastavovány další informace.

Každému objektu je možno v dialogovém okně přiřadit jeden obrázek a jeden model. Pro odkazy na další soubory lze použít HTML formátování v rámci popisu.

Aby byly odkazy na přiřazené soubory funkční, musí být samozřejmě soubor modelu, v rámci kterého jsou další informace přiřazovány, uložen.

### A.5.1 Obrázek

Vybraný obrázek, který chcete přiřadit objektu modelu, uložte vedle souboru modelu (například do adresáře s dalšími obrázky). Otevírání obrázků bude zajištěno výchozím programem pro takový typ souborů. Funkční by tedy měly být všechny běžné obrazové formáty.

### A.5.2 Model

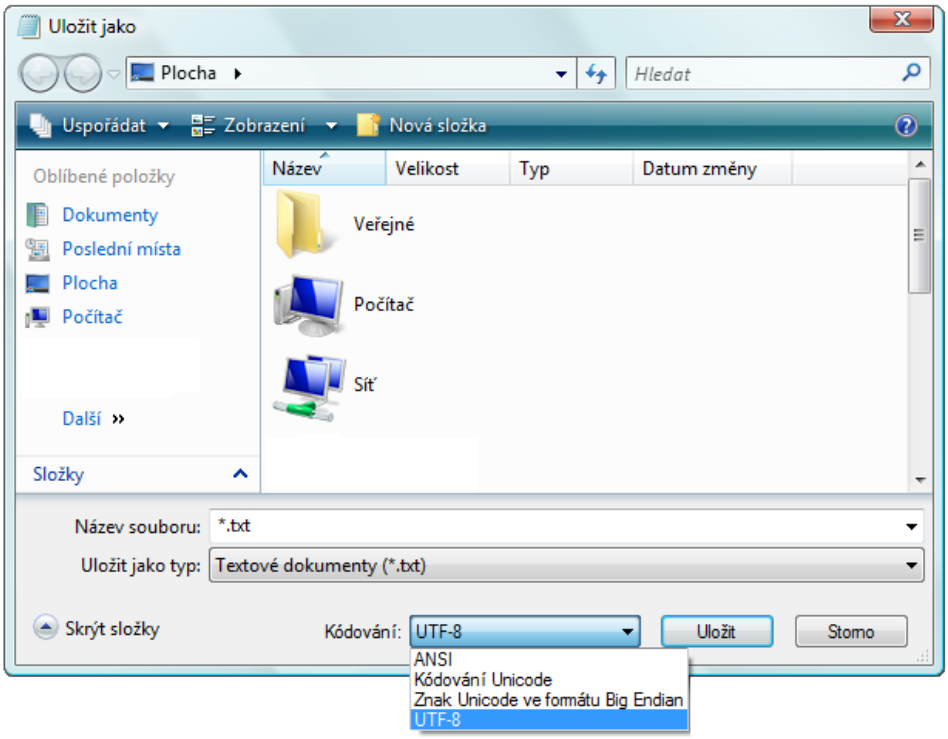
Podrobnější model ve formátu `skp` uložte vedle souboru hlavního modelu stejně jako v případě obrázku.

### A.5.3 Vytvoření popisu

1. Pro vytvoření souboru s popisem použijte vybraný textový editor.
2. Soubor uložte ve formátu `txt` v kódování Unicode UTF-8 (viz např. obr. A.2).
3. Pro formátování popisu můžete použít HTML tagy (nadpisy, odstavce...).
4. Pro odkazování na další soubory *na webu* použijte standardní HTML odkazy. Jako cíl odkazu nastavte vždy nové okno (`<a href="..." target="_blank"> Odkaz</a>`).
5. Pro odkazování na další soubory *umístěné lokálně* vedle souboru modelu použijte odkazy ve tvaru: `<a href="skp:soubor@adresa">Odkaz</a>`, kde místo `adresa` zapíšete relativní cestu k požadovanému souboru. Soubor se bude otevírat ve výchozím programu pro daný typ souboru. Zde naopak *nesmí* být cílem odkazu nové okno prohlížeče. Částí takovýchto odkazů nemůže být náhled obrázku.

### A.5.4 Použití nástroje Nastavení

1. Vyberte nástroj *Nastavení* (ze submenu nebo z panelu nástrojů).
2. Klikněte na vybraný objekt modelu.

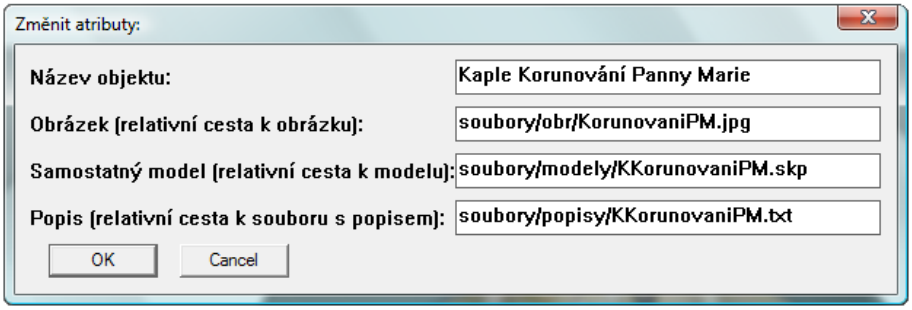



Obr. A.2: Uložení textového souboru v kódování UTF-8 (Poznámkový blok)

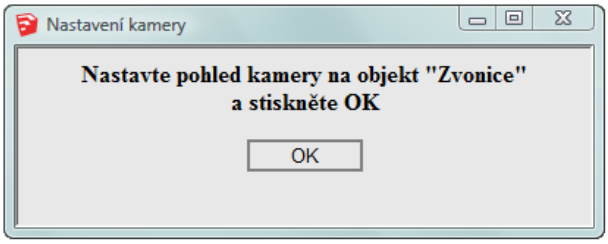
3. V dialogovém okně vyplňte příslušná vstupní pole (viz obr. A.3). V případě obrázku, modelu a popisu vložte relativní cestu k vytvořeným souborům. Pokud nějaký atribut nechcete přiřadit, nechte vstupní pole prázdné. Alespoň jeden atribut ale musí být nastaven. Stiskněte *OK*.

4. Objeví se dialog pro nastavení výchozího pohledu na objekt (viz obr. A.4). Pomocí běžných nástrojů aplikace SketchUp nastavte vhodný pohled a stiskněte *OK*.

5. Uložte soubor modelu, v rámci kterého provádíte nastavení, pokud jste tak již dříve neučinili.




Obr. A.3: Dialogové okno nástroje *Nastavení*




Obr. A.4: Dialog pro nastavení výchozího pohledu na objekt

### A.5.5 Změna a vymazání atributů

Změna a vymazání dalších informací se provádí obdobným způsobem jako jejich první nastavení:

1. Vyberte nástroj *Nastavení* (ze submenu nebo z panelu nástrojů).
2. Klikněte na vybraný objekt modelu.
3. V dialogovém okně se zobrazí dosavadní atributy jako výchozí hodnoty. Změňte obsah příslušných vstupních polí. Pro vymazání atributů vymažte obsah vstupních polí. Stiskněte *OK*.
4. Objeví se dialog pro nastavení výchozího pohledu na objekt. V případě potřeby změňte výchozí pohled na objekt a stiskněte *OK*.

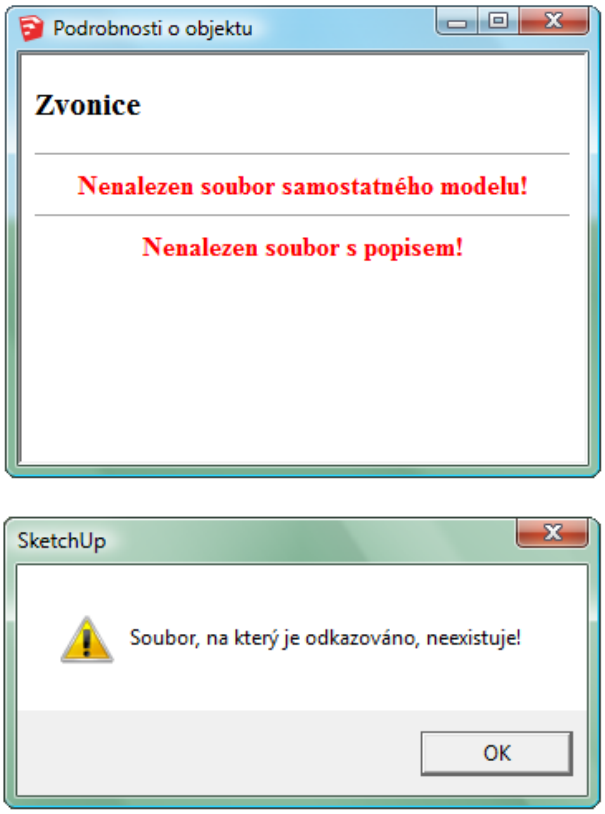
Pokud jste vymazali obsah všech vstupních polí, došlo k vymazání všech atributů i celého atributového slovníku objektu. V takovém případě již samozřejmě nebude pohled kamery nastavován.

### A.5.6 Kontrola správnosti nastavených atributů

Pro kontrolu správnosti nastavených dalších informací zobrazte okno *Podrobnosti o objektu*, buď pomocí nástroje *Výběr* (viz A.6.1) nebo pomocí odkazu ze souhrnné tabulky (viz A.6.2). Problémy, které se mohou vyskytnout:

- Pokud se místo popisu nebo odkazu na samostatný model objeví červený varovný text „Nenalezen soubor s popisem!“ nebo „Nenalezen soubor samostatného modelu!“ (viz obr. A.5), zkontrolujte správnost nastavené relativní cesty k příslušným souborům. Stejně postupujte, pokud se správně nezobrazí obrázek.
- Pokud se po kliknutí na odkaz na lokální soubor v popisu zobrazí varovná hláška „Soubor, na který je odkazováno, neexistuje!“ (viz obr. A.5), zkontrolujte správnost odkazu.

- Pokud se po kliknutí na odkaz na *lokální soubor* v popisu otevře prázdné okno aplikace Internet Explorer a neotevře se požadovaný soubor, máte pravděpodobně jako cíl v odkazu nastaveno nové okno (`target="_blank"`). Změňte cíl na `target="_self"` (nebo cíl vymažte).
- Pokud se po kliknutí na odkaz na *soubor na webu* otevře webová stránka přímo v okně *Podrobnosti o objektu*, nemáte nastaveno v odkazu jako cíl nové okno.




Obr. A.5: Chybové hlášky v případě nenalezení přiřazených souborů

## A.6 Zobrazení dalších informací

### A.6.1 Nástroj Výběr

První možností, jak zobrazit další informace o jednotlivých objektech, je použití nástroje *Výběr*:

1. Vyberte nástroj *Výběr* (ze submenu nebo z panelu nástrojů).

2. Objekty s dalšími informacemi se zvýrazňují pod kurzorem myši. Pro zvýraznění všech objektů s dalšími informacemi stiskněte Ctrl + levé tlačítko myši.

3. Pro zobrazení dalších informací klikněte na vybraný objekt. Pokud se objeví varovná hláška „Tento objekt neobsahuje další informace“, vybrali jste objekt bez přidaných atributů.

4. Při kliknutí na objekt, který další informace obsahuje, se zobrazí okno *Podrobnosti o objektu*. V tomto okně si můžete prohlédnout další informace o objektu. Pomocí odkazů lze otevřít přiřazený obrázek, model nebo případně i další soubory. Obrázek, model a soubory umístěné lokálně se zobrazí ve výchozím programu pro daný typ souboru. Webové stránky se při správném nastavení odkazu zobrazí v novém okně prohlížeče Internet Explorer.

Pokud používáte operační systém Windows XP, může se v závislosti na nastaveném výchozím programu objevit problém s otevíráním obrázků. Zkuste změnit výchozí program pro otevírání obrazových souborů. Při testování byl vždy funkční program Internet Explorer 9.


Obr. A.6: Okno s podrobnostmi o objektu a souhrnná tabulka

### A.6.2 Nástroj Tabulka

Nástroj *Tabulka* zobrazuje přehled všech objektů s dalšími informacemi. Souhrnná tabulka dále obsahuje odkazy pro zobrazení dalších informací o objektech modelu.

1. Vyberte nástroj *Tabulka* (ze submenu nebo z panelu nástrojů). Zobrazí se tabulka se všemi objekty s dalšími informacemi v rámci modelu (nebo v rámci komponenty/skupiny – viz A.7).

2. Názvy objektů slouží jako odkazy. Po kliknutí na odkaz se na vybraný objekt přesune pohled kamery (pohled nastaven v rámci nástroje *Nastavení* – A.5.4). Dále se zobrazí okno *Podrobnosti o objektu* stejně jako při použití nástroje *Výběr*.

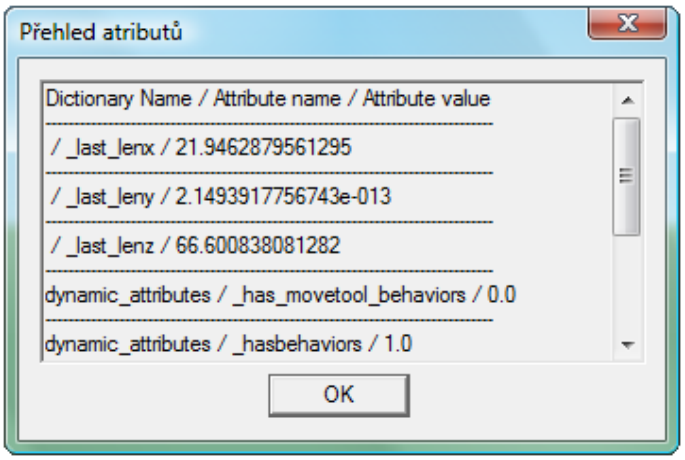
3. Vyhledávat v rámci tabulky lze po stisknutí klávesové zkratky Ctrl + F.

### A.6.3 Nástroj Přehled atributů

Nástroj *Přehled atributů* zobrazí všechny atributy, které má vybraný objekt přiřazeny. Může tedy sloužit i pro prohlížení atributů nastavených jiným způsobem.

1. Vyberte nástroj *Přehled atributů* (Plugins/TIS/Přehled atributů).

2. Klikněte na vybraný objekt. Zobrazí se všechny atributy objektu v jednoduché tabulce.




Obr. A.7: Okno nástroje *Přehled atributů*

## A.7 Komponenty a skupiny

Další informace je možno nastavovat i objektům uvnitř komponenty nebo skupiny. Nastavování i prohlížení informací u takových objektů je možné pouze pokud je komponenta či skupina otevřena pro editaci.

- Otevření komponenty/skupiny pro editaci provedete pomocí běžného postupu ve SketchUpu – dvojklikem na vybranou komponentu nástrojem *Select*. Využít můžete i volbu *Edit Component*/*Edit Group* z kontextového menu, které se zobrazí po kliknutí pravým tlačítkem na vybraný objekt.

- Uzavření komponenty či skupiny se provádí opět nástrojem *Select* – kliknutím levým tlačítkem mimo obalový obrazec komponenty/skupiny nebo volbou *Close Component*/*Close Group* z kontextového menu.

- Pokud je otevřena komponenta nebo skupina pro editaci, zobrazuje souhrnná tabulka (nástroj *Tabulka*) objekty s dalšími informacemi uvnitř otevřené komponenty/skupiny. Pokud jsou všechny komponenty a skupiny uzavřeny, zobrazuje tabulka objekty s dalšími informacemi v rámci hlavního modelu (nejsou tedy zobrazeny objekty uvnitř komponent nebo skupin).

- Informaci o tom, že se uvnitř vybrané komponenty nebo skupiny nachází objekty s dalšími informacemi, získáte z okna *Podrobnosti o objektu* (viz obr. A.6). Zde také případně zjistíte jejich názvy.

- Pokud komponenta/skupina obsahuje objekty s dalšími informacemi, je vhodné, aby sama měla nějaké informace přiřazeny. Jinak se nezvýrazní pod kurzorem, nezobrazí se u ní okno *Podrobnosti o objektu* a nebude ani uvedena v souhrnné tabulce. Objekty uvnitř by tedy zůstaly nepovšimnuty.

# B Obsah přiloženého CD

- Text diplomové práce ve formátu `pdf` – `tobias_dp.pdf`
- Plugin TIS pro SketchUp – složka `tis`
  - Plugin v komprimovaném formátu `rbz` – `tis.rbz`
  - Plugin v ZIP archivu – `tis.zip`
  - Návod k pluginu – `navod.pdf`
- Webové stránky prezentující výsledky práce – složka `webove_stranky`
  - Dostupné také na adrese: http://geo3.fsv.cvut.cz/dp/tobias
  - HTML soubory – `index.html`, `tis.html`, `vystupy.html`, `odkazy.html`, `kontakt.html`
  - Tapeta ve formátu `png` – `noisy_grid.png`[1]
  - Soubor kaskádového stylu – `styl.css`
  - Složka `obr` s použitými obrázky
  - Složka `text` s textem práce
    - Text ve formátu `pdf` – `text.pdf`
  - Složka `tis` s pluginem TIS a vzorovým modelem
    - Plugin v komprimovaném formátu `rbz` – `tis.rbz`
    - Plugin v ZIP archivu – `tis.zip`
    - Návod k pluginu – `navod.pdf`
    - Vzorový model Svaté Hory – `svata_hora.zip`
  - Složka `vystupy` s výstupy z testovaných programů
    - Výstupy z programu WalkAbout3d – `Hernychova_vila_wa.exe`, `svata_hora_wa.exe`
    - Výstupy z programu CopperCube
      - Model Pražské brány (`Svata_Hora.html`, `Prazska_brana.html`, `portal.jpg`, složka `copperlichtdata`)
      - Model Hernychovy vily (složka `hernychova_vila_cc`)

[1] http://subtlepatterns.com/