Automatické generování propagačních předmětů pomocí skriptů

Bakalářská práce

Vedoucí práce:
Ing. Mgr. Jana Dannhoferová, Ph.D.

Jiří Krampol

Brno 2011
Děkuji vedoucí mé práce, Ing. Mgr. Janě Dannhoferové, Ph.D. za cenné rady a podporu při psaní této bakalářské práce. Dále pak panu Ing. Jiřímu Urbánkovi děkuji za to, že mi vyšel vstříc a ochotně se mnou konzultoval přínos této práce pro PR oddělení PEF.
Prohlašuji, že jsem tuto bakalářskou práci vypravoval samostatně s pomocí informačních zdrojů, které jsem uvedl v seznamu literatury.

V Brně dne 17. května 2011

__________________
Abstract


This bachelor thesis deals with automatic generation of three-dimensional models of promotional items in the program Rhinoceros by using the method of scripting. Describes the methodology of visualization and the creation of scripts generating the promotional items.

Keywords

Rhinoceros, scripting, promotional items, 3D models.

Abstrakt


Tato bakalářská práce se zabývá automatickým generováním trojrozměrných modelů propagačních předmětů v programu Rhinoceros využitím metody skriptování. Popisuje metodiku vizualizace a vytváření skriptů generujících zmíněné propagační předměty.

Klíčová slova

Rhinoceros, skriptování, propagační předměty, 3D modely.
# Obsah

1 Úvod a cíl práce .............................................. 9
   1.1 Úvod ................................................................ 9
   1.2 Cíl práce ...................................................... 9
   1.3 Metodika ...................................................... 9

2 Teoretická východiska ............................................ 11
   2.1 Úvod do 3D grafiky ........................................... 11
      2.1.1 Trojrozměrné modely .................................. 11
      2.1.2 Scéna ..................................................... 12
      2.1.3 Plochy .................................................... 12
      2.1.4 Vrcholy a hrany ........................................ 12
      2.1.5 Renderování ............................................. 12
      2.1.6 Materiál a textury ..................................... 12
   2.2 Skriptování .................................................... 13
      2.2.1 Skriptovací jazyk ....................................... 13
      2.2.2 Skripty ..................................................... 13
   2.3 Propagační předměty a podpora prodeje ....................... 13
      2.3.1 Cíle a nástroje podpory prodeje ...................... 13
      2.3.2 Podoba propagačních předmětů ....................... 13
      2.3.3 Podstata propagačních předmětů .................. 14
      2.3.4 Strategie výběru propagačních předmětů .......... 14
   2.4 Analýza nástrojů pro 3D modelování ................................ 15
      2.4.1 AutoDesk 3Ds Max ...................................... 15
      2.4.2 Blender ................................................... 17
      2.4.3 Google SketchUp ....................................... 19
      2.4.4 Rhinoceros .............................................. 21
      2.4.5 Zhodnocení ............................................. 23
   2.5 Skriptování v Rhinoceru .................................... 25
      2.5.1 Makra ..................................................... 25
3 Vlastní práce
    3.1 PR oddělení PEF Mendelu .................................................................33
    3.2 Výběr propagačních předmětů .............................................................33
    3.3 Modelování vybraných předmětů ..........................................................34
    3.4 Implementace skriptů ............................................................................34
        3.4.1 Skript modelující propisovací tužku.............................................35
        3.4.2 Část skriptu modelující podložku pod myš ..................................47
        3.4.3 Část skriptu modelující přívěsek na klíče ......................................49
        3.4.4 Část skriptu modelující deštník .....................................................49
    3.5 User Interface .........................................................................................51
        3.5.1 UI v prostředí Rhinoceros ...............................................................51
        3.5.2 UI samotné aplikace ......................................................................52
    3.6 Uživatelská příručka ............................................................................53

4 Závěr
    4.1 Přínos práce .........................................................................................55

5 Literatura .....................................................................................................56

A Skript uživatelského rozhraní .................................................................58
Seznam obrázků

Obr. 1: Rozhraní programu AutoDesk 3Ds Max 16
Obr. 2: Rozhraní programu Blender 18
Obr. 3: Rozhraní programu Google SketchUp 20
Obr. 4: Rozhraní programu Rhinoceros 21
Obr. 5: Struktura skriptu 27
Obr. 6: Struktura hlavního podprogramu 28
Obr. 7: Příkaz _LoadScript 30
Obr. 8: Příkaz _RunScript 30
Obr. 9: Konzolové okno 31
Obr. 10: Příkaz _Toolbar a Vlastnosti palety 31
Obr. 11: Editace tlačítka 32
Obr. 12: Tělo propisky 41
Obr. 13: Propiska 47
Obr. 14: Deštník 51
Obr. 15: Uživatelská rozhraní 52
Obr. 16: Ukázka finálních propagačních předmětů 53
Seznam tabulek

Tab. 1  Zhodnocení hodnotících kritérií  25
Tab. 2  Hodnotící kritéria  25
1 Úvod a cíl práce

1.1 Úvod

Počítačová grafika je v dnešní době součástí života každého z nás. Obklopuje nás téměř na každém kroku, aniž si to mnohdy sami uvědomujeme. Velký boom zažívá především 3D grafika, tedy namísto dvourozměrých grafických objektů, na které jsme zvyklí, se začínají stále více objevovat prvky trojrozměrné grafiky. Vizualizace předmětů ve 3D se tak stává čím dál tím více žádoucí a vyhledávánou. Základ této vizualizace tvoří 3D modelování, které nám umožňuje vytvářet trojrozměrné modely předmětů běžného života a přibližovat je tak ještě více reálně. Proces vytváření těchto 3D modelů je možné navíc díky skriptovacím jazykům plně zautomatizovat.

Trojrozměrná počítačová grafika je tak v současnosti dostupná opravdu každému. Nemusejí to tedy hned být velké společnosti, které jsou ochotny do zmíněné 3D vizualizace investovat obrovské prostředky, naopak i malé společnosti a firmy mohou perfektně tuto část počítačové grafiky využít např. pro lepší grafické ztvárnění a prezentaci svého logotypu, či reklamních předmětů. A právě reklamní resp. propagační předměty jsou nedílnou součástí podpory prodeje těchto konkrétních firem a společností, která vede k zvýšení povědomí o nich samotných, jednak prodeje jejich produktů.

V současnosti je možné na trhu využít mnoha reklamních agentur, které nabízejí nespočet druhů reklamních a propagačních předmětů. Málo kdy však, na základě nabídek těchto agentur, zákazník dostane ucelenou představu o tom, jak budou jeho propagační předměty vypadat ve skutečnosti.

Logickým vyústěním řešení tohoto problému je přinést zákazníkovi možnost vizualizace propagačních předmětů, s prvky identifikujícími jeho konkrétní firmu, a to za účelem získání jeho jasného představu o tom, jak budou propagační předměty vypadat, ještě předtím než se on sám rozhodne pro jejich vyhotovení.

1.2 Cíl práce

Cílem této práce, jak již bylo naznačeno v úvodu, je navrhnout proces automatického generování propagačních předmětů ve 3D a k tomuto účelu vhodně využít metody skriptování.

1.3 Metodika

Trojrozměrné modely propagačních předmětů budou generovány pomocí skriptů ve 3D modelovacím nástroji. Je tedy nutné, jednak správně zvolit konkrétní software, jednak si osvojit programovací jazyk, který tento software používá pro skriptování. Za tímto účelem bude provedena analýza 3D modelovacích nástrojů, která zhodnotí především možnosti vývoje skriptů v nich samotných a na základě které bude vybrán konkrétní software, který se použije pro tuto práci.
Ke tvorbě skriptů postačí, mimo prostředí vybraného programu, také jakýkoli textový editor, který je schopen zvýraznit syntaxi konkrétního skriptovacího jazyka.

Propagační předměty, které se mají pomocí skriptů automaticky generovat, budou vybrány především na základě konzultace s vedením PR oddělení fakulty. Zohledněn bude o význam modelování těchto konkrétních předmětů.

Po dokončení implementace skriptů bude vytvořena jednoduchá aplikace, která bude využívat externí přístup k vybranému 3D modeláři, tak aby co nejvíce usnadnila práci potencionálnímu uživateli s uvedenými skripty. Konkrétní software pro vývoj této aplikace bude vybrán opět na základě zvoleného 3D nástroje a jeho kompatibility s jinými programy.

Na závěr bude vytvořena uživatelská příručka pro potencionálního uživatele této aplikace. S největší pravděpodobností půjde o zaměstnance PR oddělení PEF Mendelu. Tato uživatelská příručka bude poskytovat informace o samotné aplikaci a dále pak o možnosti rozšíření a úpravy automaticky generovaných modelů propagačních předmětů.
2 Teoretická východiska

2.1 Úvod do 3D grafiky

S pojmem 3D grafika se již každý z nás určitě setkal. Žijeme v době, kdy na ni narážíme na každém kroku, ať už o tom víme nebo ne. Obecný pojem Grafika resp. Počítačová grafika je možné rozdělit na 2D a 3D grafiku. Zatímco 2D grafiku reprezentují vektory a bitmapy, 3D grafika používá tří-dimenzionální reprezentaci geometrických dat, uložených nejčastěji v počítači, pro účely provádění výpočtů a renderování 2D obrazů. Tyto obrazy mohou být uloženy pro pozdější použití, nebo mohou být okamžitě zobrazeny.

Přes tyto rozdíly 3D počítačová grafika spoléhá na mnoho stejných algoritmů jako 2D počítačová grafika. V programech určených pro počítačovou grafiku se tyto rozdíly smazávají. 2D aplikace mohou používat některé 3D techniky jako např. osvětlení a naopak 3D aplikace mohou používat např. 2D renderovací techniky k dosažení odpovídajících výsledků.

Trojrozměrná počítačová grafika bývá velmi často spojována s 3D modelováním. Pod pojmem 3D modelování je možné si představit definování tvaru prostorových objektů za pomoci datových struktur a algoritmů pro vytvoření a následnou manipulaci s 3D objekty. Produkt 3D modelování je 3D model, který lze definovat jako matematickou reprezentaci jakéhokoli trojrozměrného objektu.

2.1.1 Trojrozměrné modely

Trojrozměrné objekty resp. modely jsou oproti 2D grafice v trojrozměrném souřadnicovém systému [x, y, z]. 3D model vyplňuje jistou prostorovou oblast, která může a nemusí být souvislá. Hmota tělesa je ohraničena stěnami, které jsou vymezeny hranami, přičemž každá hraná začíná a končí ve vrcholu. Hrany jsou uspořádány do smyček, které vymezuju danou stěnu. Rozlišujeme tři základní typy modelů.


Dalším z řady modelů těles je Povrchový model, který přináší přímé informace o povrchu tělesa. Tedy o hranách a plochách, z kterých je těleso sestaveno. Umožňuje vytvoření tělesa tzv. bez objemu a poměrně snadné zobrazování. Potýká se však problémy právě při výpočtu svého objemu, či těžiště.

Posledním typem modelu je Objemový model, který pracuje přímo s vnitřním objemem, avšak neposkytuje informace přímo o povrchu tělesa. Zobrazování se tak může stát obtížnějším. Tento model pracuje se základní objemovou

---

1 Renderování je tvorba reálného obrazu na základě počítačového 3D modelu
2.1.2 Scéna

Je-li řeč o trojrozměrné počítačové grafice, pod pojmem scéna je možné si představit v podstatě 3D pracovní plochu, která je součástí každého 3D modelovacího nástroje.

Tato scéna obsahuje souřadné systémy jednotlivých objektů. V případě natačení scény se jednotlivé souřadné systémy objektů převádí na globální souřadný systém, ve kterém je otáčí a následně je zpětně převádí do původního souřadného systému. [9]

2.1.3 Plochy

Plochy jsou de facto stěny, ze kterých se jednotlivé 3D objekty skládají. Tyto plochy se skládají z polygonů, které jsou vždy ohraničeny minimálně třemi body. Proto jsou plochy také někdy označovány přímo jako polygony. [9]

2.1.4 Vrcholy a hrany

Jak již bylo zmíněno výše, každá plocha ve 3D je vymezena hranami, přičemž každá hrana začíná a končí ve vrcholu. Každou z těchto částí máme možnost samostatně měnit, ale až už upravujeme vlastnosti vrcholů, hran nebo ploch, upravují se zároveň parametry obou zbylých částí. [9]

2.1.5 Renderování

Renderování je dalším pojem, který se používá v rámci práce s 3D modely. Jde v podstatě o projekci reálného 2D obrazu z vytvořené 3D scény. Zjednodušeně řečeno, tento proces převádí 3D modely a aktuální scénu na 2D obraz, což je možné si představit jako vyfocení scény fotoaparátem, kdy je výsledkem bitmapový obrázek.

Renderování se používá také u videa, kde sice proces probíhá jiným způsobem, ale výsledkem je to samé, tedy realizace finálního produktu. [9]

2.1.6 Materiál a textury

Pod pojmem materiál je možné si představit soubor vlastností, ze kterého se skládá výsledný vzhled povrchu objektu. Při práci s 3D modelovacími programy je k dispozici mnoho vlastností, které mohou být materiálu nastaveny.

Textury jsou 2D obrázky, které obalují objekt a výrazně tak ovlivňují jeho vzhled. Objektu je možné přiřadit texturu jakékoli fotografie či obrázku a dodat mu tak realismický vzhled. [9]

---

2 Volumetric Picture Element – základní objemová jednotka
2.2 Skriptování

Pod pojmem skriptování je možné si jednoduše představit tvorbu skriptů pomocí skriptovacího jazyka.

2.2.1 Skriptovací jazyk

Skriptovací jazyk je interpretovaný programovací jazyk. Od klasického programovacího jazyka se liší především tím, že téměř nikdy nepoužívá nástroje, které používá programovací jazyk jemu podobný, jako je např. kompilátor. Skriptovací jazyky jsou navrženy především s ohledem na snadné ovládání jazyka a rychlé a pohodlné vývoj programů.

Velmi často jsou skriptovací jazyky používány k tvorbě plug-inů, které představují rozšíření funkcionality specifických programů, pro které jsou tyto plug-iny určeny. [12]

2.2.2 Skripty

Skripty lze označit jako sadu příkazů, uložených nejčastěji v textovém souboru, které ovládají určité části hardwaru, softwaru nebo operačního systému. Jde tedy de facto o program zapsaný v konkrétním skriptovacím jazyce, který nepotřebuje být překládán kompilátorem a namísto toho je vykonáván přímo aplikací.

Skripty jsou však odlišné od kódu aplikací, ve kterých jsou použity, a to především implementací jazyka, přístupovými právy, prostořem, který potřebují pro své uložení a obecně neschopností pracovat mimo kontext hostované aplikace. [12]

2.3 Propagační předměty a podpora prodeje

2.3.1 Cíle a nástroje podpory prodeje

Podpora prodeje nabývá v dnešní době stále více na významu a setkává se stále s větším zájmem. Pod tímto pojmem je možné si představit především užitečný nástroj pro získání pozornosti cílových skupin. Jedním z hlavních cílů podpory prodeje je tedy vyvolat pozitivní reakci spotřebitelů.

Za účelem dosažití nejen tohoto cíle je možné využít např. propagační předměty, které jsou jedním z hlavních nástrojů podpory prodeje. Při jejich výběru je však třeba brát v úvahu typ trhu, cíle podpory prodeje, konkurenci a samotnou nákladovou efektivitu. [14]

2.3.2 Podoba propagačních předmětů

Propagační předměty, reklamní zboží, reklamní předměty, propagační výrobky, propagační dárky nebo reklamní dárky, pod všemi těmito názvy se v podstatě skrývá jedno a to samé. Jde o předměty, které jsou prostředkem jak vhodně bu-
dovat dlouhodobé vztahy nejen se stávajícími, ale i novými klienty. Je dobré, když je předmět dlouhodobě používaný a pozitivně vnímaný příjemcem.

Tyto předměty jsou obvykle opatřeny potiskem nesoucím název, logo nebo slogan firmy a jsou rozdávány přirozeně za účelem propagace firmy, ať už v jejím sídle, na veletrzích či výstavách např. při zavádění produktu na trh, nebo dokonce jako součást větších Guerillových kampaní\(^3\). Reklamní předmět zákazníkovi neustále připomíná propagovanou firmu. Žádný jiný nástroj podpory prodeje nezůstává se zákazníkem po tak potenciálně dlouhou dobou.

Téměř všechny předměty běžného použití lze opatřit třeba jen např. firemními barvami a povšimnout klienta s jeho potřebami. Nejčastějšími propagacíními předměty se tak stávají různé přívěsky, plakáty, psací potřeby, hrnky, sklenice a kancelářské potřeby. Kategorii produktů, která má největší zastoupení mezi reklamním zbožím jsou předměty, které může člověk nosit na sobě. Jde tedy zejména o textil, po-

krývky hlavy, či dokonce šperky a ozdoby. Tyto předměty tvoří většinu reklamních předmětů tvoří většinu relativně malé a levné věci, ale výjimkou nejsou ani dražší a luxusní předměty, které mohou být určeny např. pro významné zákazníky. \([4]\)

### 2.3.3 Podstata propagačních předmětů

Principem v pozadí reklamních předmětů je opětování (reciprocation). Kdykoli někdo dostane dárek, má tendenci to oplatit. V byznysu lze tuto psychologickou výhodu využít několika způsoby. Na veletrzích a výstavách vytvářejí reklamní dárek pozitivní obraz firmy. Nicméně je nutné pečlivě dohlédnout na to, aby dárek přenášel zamýšlené sdělení. Jestliže například jeden prodejce nabízí reklamní plastové hrníčky a druhý hrníčky z porcelánu, návštěvníci si vytvoří zcela rozdílný dojem o obou prodejcích. Pohled na firmu rozdávající plastové hrníčky bude spíše negativní, což je jistě proti jejím záměrnům. \([10]\)

### 2.3.4 Strategie výběru propagačních předmětů

Jednou z nejlepších strategií při výběru propagačních předmětů je zajistit, aby to byly věci unikátní a aby komunikovaly firemní obchodní sdělení. Reklamní předměty mohou posílit nákupní rozhodnutí, protože zákazník může díky nim získat pocit, že se rozhodl správně.

Tyto předměty mohou posílit obchodní vztahy a pomoci vzbudit zájem nových potencionálních klientů. Obecně vzato, propagační předměty mohou být důležitým prostředkem posílení komunikace se všemi typy zákazníků. \([1]\)

---

\(^3\) Guerillové kampaně jsou nekonvenční formou propagace s cílem vzbuzení maximálního zájmu za použití omezeného rozpočtu.
2.4 Analýza nástrojů pro 3D modelování

Mají-li být výstupem této práce skripty, které budou automaticky generovat modely propagačních předmětů, je bezesporu nutné si správně zvolit software určený pro 3D modelování, který nabízí dostatečné funkce a možnosti pro vytváření těchto skriptů.

Rozhodl jsem se tedy porovnat několik nejdůstojnějších a nejpoužívanějších nástrojů určených pro 3D modelování a 3D grafiku obecně, z nichž dva jsou zástupci komerčního softwaru a dva z nich patří do rodiny Open Source, neboli Open-source software (OSS), tedy do skupiny programů s otevřeným zdrojovým kódem. Konkrétně se tedy bude jednat o programy AutoDesk 3Ds Max, také známý jako 3D Studio Max, Blender, Google SketchUp a Rhinoceros.

Za účelem provedení kvalitní analýzy jsem si zvolil čtyři hlavní a pro můj účel stěžejní hodnocení kritérií, kterým jsem subjektivně přisoudil rozdílnou váhu, a to sestupně ve stejném pořadí v jakém jsou uvedeny níže. Těmito hodnotícími kriterii tedy jsou:

- **Dostupnost programu**
  Zde budu posuzovat především to, jde-li o komerční software, nebo zdali se jedná o zástupce OSS. Pokud přijde o zástupce Shareware, budu zvažovat možnost využití např. školní či výukové licence a s tím spojené případné náklady.

- **Podporovaný skriptovací jazyk**
  Tedy ve kterém skriptovacím jazyce lze vyvíjet skripty pro konkrétní 3D modelovací software a jaká je dostupnost a kvalita dokumentace tohoto jazyka.

- **Rozšířenost programů**
  Zde budu hodnotit do jaké míry je daný software rozšířen a používán, s čímž souvisí i zmíněná dostupnost dokumentace programu.

- **Předchozí zkušenost**
  Pokud mi na základě hodnocení předchozích kritérií budou vycházet dva nebo více programů přibližně nestejno, přihlédnu i k předchozí zkušenosti s těmito modeláři.

2.4.1 AutoDesk 3Ds Max

AutoDesk 3Ds Max je profesionální program pro 3D grafiku, vizualizace a animace. Tento software bývá používán v postprodukci, při výrobě reklam, filmů a v televizním průmyslu např. pro architektonické a konstrukční vizualizace. Často také slouží k tvorbě grafiky do počítačových her atd.

Vytvářet skripty v tomto programu lze za pomocí jazyka MAXScript. Tento programovací jazyk byl navržen speciálně pro doplnění 3D Studia Max. Je objektově orientovaný stejně jako 3D Studio Max a má několik speciálních rysů a struktur, které odrážejí některé koncepy v uživatelském rozhraní 3D Studio
Maxu. Jedná se například o systém souřadnic, mód animace s automatickým klíčováním a přístup k objektům ve scéně přes hierarchické cesty, které jsou stejné jako hierarchie 3Ds Max objektů.

Obr. 1: Rozhraní programu AutoDesk 3Ds Max

MAXScript je ovšem natolik rozsáhlý, že umožňuje složité programové úlohy. Zahrnuje všechny důležité kontrolní struktury a datové typy jako například 3D vektory, algebru matic a prvky vyššího jazyka a rekurzivní funkce. MAXScript je přizpůsobený na práci s velkými skupinami objektů například při tvorbě komplexních procedurálních skupin, konstrukci náhodných hvězdokup, nebo umíšťování objektů v přesných numerických vzorech.

MAXScript podporuje I/O formátovaného textu, takže je možné jak číst v souborech, obsahujících např. názvy, detaily textur, nastavení výstupu atd. exportovaných z jiného softwaru, tak vytvořit soubor s nastavením přímo ze souborů scén 3D Studio Maxu.

Jazyk MAXScript je obecně snadno naučitelný. Jedná se totiž o speciální typ angličtiny, která je doplněná o část algebry běžící na pozadí. Syntaxe tohoto jazyka je dostatečně jednoduchá, a to i pro toho, kdo nikdy předtím neměl co dočinění s žádným programovacím jazykem. Samotný jazyk obsahuje minimum pravidel pro formátování. Nejde tedy o žádný pevně daný styl formátování, kte-
rý by se nesměl porušit. MAXScript je okamžitě spustitelný s volnými proměnnými a automatickou správou paměti pro jednodušší programování.


Specifickou věcí právě pro 3D Studio Max je tzv. MAXScript Listener, který k nalezení v té samé nabídce určené pro práci se skripty. Toto speciální okno je ve svém podobě příkazové řádce DOSu ve Windows. MAXScript Listener je též místem, kam se vypisují všechna chybová hlášení a upozornění.

Pokud je skript, tedy sekvence jednotlivých příkazů, vytvořený pomocí MAXScript Listeneru označen a přemístěn někam směrem na panel nástrojů v prostředí 3D Studio Maxu, automaticky se na tomto panelu vytvoří tlačítko, které funguje jako spoušť pro tento, právě vytvořený, skript. Vzhled a např. popisek takto vytvořeného tlačítka lze samozřejmě editovat podle vlastních představ.

MAXScript Listener ovšem nepředstavuje jedinou možnost, jak vytvářet skripty v tomto programu. Psát skripty je možné samozřejmě i mimo prostředí 3D Studio Maxu, a to za pomoci libovolného textového editoru. Potom co je tímto způsobem skript vytvořen, stačí jej uložit s příponou „.ms“ a otevřít pomocí nabídky MAXScript v 3D Studio Maxu. Další možností je takto externě vytvořený skript umístit do speciálního adresáře spolu s ostatními skripty, které jsou automaticky spouštěny vždy ihned po zapnutí tohoto 3D nástroje.

Důležitou součástí tohoto 3D modeláře je také nápověda, která je zdarma dostupná na internetu a je možné se k ní dostat přirozeně i za pomoci nabídky v prostředí 3D Studio Maxu. Další možností jak vyvolat nápovědu je stisknutí klávesy F1, tak jak to známe i z jiných aplikací. Toho lze velmi dobře využít při psaní konkrétních příkazů do MAXScript Listeneru, kdy se po vepsání klíčového slova a stisknutí zmíněné klávesy otevře nápověda právě pro tento konkrétní příkaz.

Samotný MAXScript nabízí také nesčetný počet funkcí určených pro manipulaci s vytvářenými objekty a mnohé další. Seznam všech těchto funkcí je dostupný kde jinde než v nápovědě MAXScriptu. [6]

2.4.2 Blender

Blender, který je dostupný na mnoha platformách a operačních systémech, nemůže patřit mezi profesionální 3D modelovací programy. Disponuje i mnoha dalšími možnostmi v oblasti tvorby animací, stínování, renderování a tvorby interaktivních aplikací. I přesto, že se jedná o Open Source produkt, množství nástrojů, které má k dispozici, ho řadí mezi ty kvalitnější programy v oblasti 3D grafiky.
Důkazem toho může být projekt Elephants Dream od vedoucího vývojáře Blenderu Tona Roosendaala, který se v roce 2005 rozhodl založit studio Orange, které mělo vytvořit krátký animovaný film pouze za pomoci Open Source technologií a ukázat tak světu, že to jde i bez drahých komerčních nástrojů určených pro vývoj podobných audiovizuálních děl. Do zmíněného studia byli najati ti nejlepší vývojáři Blenderu a výsledkem nebyl pouze samotný film, ale především obrovský pokrok, kterého nebylo dosaženo za celou dosavadní dobu vývoje tohoto 3D modelovacího nástroje. A protože šlo o Open Source projekt, jsou všechny pracovní soubory, stejně tak jako vlastní film, dostupné zdarma ke stažení na internetu.

Obr. 2: Rozhraní programu Blender

Během uvedeného projektu došlo tak např. k razantnímu rozšíření nabídky funkcí, které má Blender implementovány ve své základní verzi. Těmito funkcemi to však zdáleka nekončí. Značné rozšíření schopností programu je možné provést prostřednictvím dalších skriptů nebo tzv. plug-inů. Vytvářet tyto skripty lze ve zmíněném 3D nástroji mimo jiné za pomoci programovacího jazyka Python. Samotný Python je vhodné si, z oficiálních stránek, stáhnout a nainstalovat. A to především z toho důvodu, že obsahuje některé objekty a metody, které jsou nezbytné pro tvorbu zmíněných skriptů. Zároveň je však pravdou, že i samotný Blender obsahuje některé metody a objekty jako svou nedílnou součást a je možné se tak setkat se skripty, pro jejichž tvorbu není instalace programovacího jazyka Python vyžadována. Pokud ovšem uživatel ne-
chce zůstat pouze u základních skriptů resp. skriptů generujících základní a triviální předměty, je vysoce doporučováno zmíněnou instalací provést. Je nutné si ovšem dát pozor na úskalí s tím spojené a ujistit se, jakou verzi Pythonu podporuje používaná aktuální verze Blenderu. To lze snadno zjistit z konzolového okna Blenderu. Pokud je to potřeba, nabízí se možnost k samotnému Pythonu do instalovat i další moduly.

Konkrétní Python skripty lze psát v prostředí samotného Blenderu, stejně tak dobře jako například v externím textovém editoru, kdy se obyčejné textové soubory, obsahující skripty, pouze obohatí příponou „.py“ a následně v Blenderu otevřou. Samotná manipulace, načtení a spuštění skriptu v prostředí Blenderu je již pak velmi snadná a intuitivní. Pokud je skripv v pořádku, provede se po jeho spuštění všechno co má a tím celá akce vykonání skriptu končí. V opačném případě se na obrazovce vypíše informace o chybě a v konzolovém okně Blenderu je možné se o této chybě dozvědět podrobné informace.

Možnou nevýhodou může být fakt, že samotná instalace Pythonu zpravidla nestačí a po ní je možné, že skripy nebudou v prostředí Blenderu fungovat. Je třeba provést několik dalších kroků, jakými jsou např. modifikace systémových cest a souborů. Návod na kompletní instalaci je možné nalézt přímo na stránkách výrobce tohoto produktu. A jak je již z názvu patrné, jde o produkt společnosti Google, která si stojí za tím, že tento její modelovací software je tím nejlepším na trhu, a to především pro své intuitivní ovládání.

Tento 3D modelovací nástroj je mimo jiné velmi vhodný pro architekty, stavební a strojní inženýry, stejně tak dobře jako pro filmové tvůrce, vývojáře počítačových her a další odborníky pracující v podobných profesích. Samozřejmě je však dostupnost i pro úplné laiky. A jelikož se jedná o produkt společnosti Google, je tento software kompatibilní i s ostatními produkty této společnosti. Jedním z rozšíření je tzv. Google 3D Warehouse. Nejde o nic jiného než o webovou stránku, kam mají Google SketchUp vývojáři možnost umísťovat své modely. Tímto způsobem se tak mohou, uživatelem vytvořené modely, dostat do aplikace Google Earth. Pokud je například vytvořen model některé reálné existující budovy a při nařávání tohoto modelu na zmíněné webové stránky zatrhuje možnost „Google Earth Ready“, prochází tento model kontrolou několika speciálně určených zaměstnanců společnosti Google, které kontrolují, zda má
model opravdu všechny potřebné parametry. A pokud model touto kontrolou projde, je možné jej zanedlouho spatřit přímo v aplikaci Google Earth.

Obr. 3: Rozhraní programu Google SketchUp

Tak jako i ostatní 3D modelovací nástroje nabízí Google SketchUp možnost tvorby skriptů, a to již od své verze 4 pomocí tzv. Ruby skriptů, které jsou psány ve stejnojmenném programovacím jazyce. Vývojáři těchto skriptů poskytují své skripty zdarma ke stažení na mnoha tzv. SketchUp Ruby fórech. Tyto skripty lze psát jak externě, za pomoci oblíbeného textového editoru, tak přímo v prostředí Google SketchUp, jehož součástí je Ruby konzole. Do této konzole je možné libovolně vpísovat Ruby příkazy a metody, tak jak je tomu i u ostatních 3D modelovacích programů, které obsahují konzolové okno, či příkazový řádek.

Skripty vytvořené mimo prostředí programu je před jejich spuštěním nutné zařídit příponou „.rb“ a umístit je do složky Plugins, která se nachází v tom samém adresáři, kde je samotný Google SketchUp nainstalován. Do této složky Google SketchUp, vždy po svém spuštění, nahlíží a hledá zde skripty, které spočíteň se samotným programu spouští. Takto nahrávané skripty se po té nalézají v prostředí programu Google SketchUp pod záložkou Plugin. Takto vytvořené skripty se samozřejmě dají také spouštět voláním konkrétních funkcí ve zmíněném konzolovém okně tohoto 3D nástroje.

Samotný jazyk Ruby je interpretovaný skriptovací programovací jazyk, který je plně objektově orientovaný a díky své poměrně jednoduché syntaxi snadný k naučení. Zároveň je také však dostatečně výkonné na to, aby dokázal konkurovat známějším skriptovacím jazykům jakými jsou Python nebo Perl. Přes všechna jeho pozitiva, je tento programovací jazyk ovšem méně rozšířený např. oproti jazyku Perl a potýká se také s nedostatkem české dokumentace. [3]
2.4.4 Rhinoceros

Program Rhinoceros je dalším z řady komerčních nástrojů určených pro 3D modelování. Hojně se využívá v designu, ať už průmyslovém či grafickém, tak i např. v architektuře a šperkařství.

Tento 3D nástroj umožňuje vytvářet, editovat, analyzovat, dokumentovat, renderovat, animovat a překládat NURBS křivky, plochy a tělesa jakékoli složitosti, stupně nebo velikosti. Rhinoceros také podporuje polygonové sítě a mraky bodů.

Obr. 4: Rozhraní programu Rhinoceros

Tento program nabízí nesoutěžitelné 3D modelování. Disponuje nástroji, které lze nalézt pouze v mnohonásobně dražších programech a díky tomu umožňuje vy-modelovat jakýkoli tvar. Může se pochvátit extrémní přesností a je tak schopen vytvořit model čehokoli, bez ohledu na výslednou velikost tohoto modelu.

Samozřejmě je kompatibilita, Rhinoceros je schopen vyměňovat si soubory s ostatními grafickými programy. Dále pak umožňuje neomezené úpravy, vždy je možné posunout se v pracovním postupu zpět, či o úroveň dál. V neposlední řadě nelze opomenout možnost customizace, nebo-li přizpůsobení prostředí Rhinocera tak, aby odpovídalo specifickým potřebám konkrétního vývojáře. Kromě toho Rhinoceros nabízí 2D kreslení, anotace, ilustrace, organizaci velkých projektů, čtení ze souboru a opravy, přímé propojení s externími zařízeními.

Práce v tomto nástroji je rychle zvládnutelná a dobře naučitelná. Program nevyžaduje žádný zvláštní hardware a pracuje rychle dokonce i na obyčejném notebooku. Cena tohoto software je stejná jako cena ostatních programů pro

Skriptování v programu Rhinoceros bylo možné již ve verzi 1.0. V tomto případě šlo však o skriptování pouze ve formě posloupnosti příkazů. Tedy neo- hrané, staré, bez jakýchkoli vestavěných funkcí, ale přeci jenom dobře použitelné. Plnohodnotné skripty je možné v tomto programu vytvářet již od verze 2.0 pomocí Microsoftem vytvořených jazyků MS Visual Basic Scripts (VBScript) a MS Java Skript (JScript), což otevírá Rhinoceru širší možnosti do systému, k proměnným, objektům a jejich vlastnostem.

- VBScript
  Tento jazyk je jedním z nejsnadnějších a nejtolerantnějších jazyků. Je postaven na programovacím jazyku Visual Basic, událostmi řízeném programovacím jazyce taktéž od společnosti Microsoft. Pokud je člověk schopen programovat ve verzích jazyce a hledá skriptovací jazyk snadný na naučení, je pro něj VBScript jasným východiskem. Ovšem vytvářet skripty pomocí tohoto jazyka je schopen i ten, kdo předchozí zkušeností s tímto programovacím jazykem nemá.
  Malé skripty jazyka VBScript lze spouštět ve většině současných verzí operačního systému Windows, dokázali přistupovat k velkému množství programů uvnitř počítače a pracovat tak s daty uživatelů. Tímto způsobem bohužel zneužívali VBScripty někteří hackeri, kteří mají jazyk VBScript pro tyto vlastnosti v oblibě. Vznikl tak např. známý virus „ILoveYou“ a mnohé další.
  Jazyk VBScript je ovšem použitelný i v kladném smyslu slova, mimo jiné právě proto, že na jeho základech je postaven jazyk RhinoScript, s jehož pomocí jsme schopni vytvářet skripty v programu Rhinoceros. [8]

- JScript
  JScript je interpretovaný skriptovací jazyk založený na objektově orientovaném programování. Ačkoli tento jazyk neposkytuje tolik možností jako plnohodnotné objektově orientované programovací jazyky jako např. C++, JScript je dostatečně silným jazykem pro účely, které byly důvodem jeho vývoje.

---

4 Non-uniform rational basis spline (NURBS) je matematický model běžně používaný v počítačové grafice pro generování a reprezentování křivek a ploch, které nabízejí velkou flexibilitu a přesnost při manipulaci jak s analytickými tak s volnými tvary.
Jazyk JScript, jak by se mohlo z jeho názvu zdát, není žádnou zjednodušenou verzí jiného programovacího jazyka, přesto jsou, jak již bylo řečeno, jeho schopnosti omezené. Nelze v něm např. vyvíjet samostatně běžící aplikace a také postrádá vestavěnou podporu pro čtení nebo zápis souborů. Skripty napsané v tomto jazyce mohou být spouštěny pouze pomocí interpreta, jakým může být např. Active Server Pages (ASP), Internet Explorer nebo Windows Script Host.


A právě na zmiňěném jazyku VBScript je postavený skriptovací RhinoScript. Jazyk speciálně určený pro Rhinoceros, který je hojně využíván již od počátku vývoje skriptování v tomto programu.

Skripty vytvořené tímto skriptovacím jazykem je možné vytvářet jak externě, tak přímo v prostředí programu Rhinoceros, a to ať už pomocí kontextové nabídky RhinoScript a voláním funkcí a příkazů určených pro nahrání a spuštění skriptů, nebo použitím konzolového okna, které lze vytvořit pomocí stejné nabídky, a to příkazem pro editaci skriptu. Pokud se uživatel rozhodne psát skripty mimo prostředí Rhinocerosu, postačí mu k tomu obyčejný textový editor. Všechny Rhino VBScripty se totiž chovají naprosto stejně jako obyčejné textové soubory. Jedinečné na co nelze zapomenout, je uložit tyto soubory s příponou ".rvb".

Od verze Rhinoceros 5.0 je možné již využít také nově vytvořený skriptovací jazyk Rhino.Phyton, který si zakládá na své flexibilitě a jasné syntaxi. Není tomu ovšem tak, že by se tento jazyk snášel konkurz konkrétního jazyka RhinoScript. Nabízí pouze alternativní možnost vytváření skriptů v tomto 3D nástroji.

2.4.5 Zhodnocení

Všechny programy určené pro 3D modelování, které jsem hodnotil, mají mnoho společného, především co se týče možností skriptování. Všechny využívají skriptovacích jazyků, ať už speciálně vytvořených pro konkrétní software, nebo pouze převzetých odjinud. Tak jako tak, všechny tyto jazyky disponují dostatečně jednoduchou syntaxí a nejsou složité na naučení i pro ty, kdo se doposud s žádným programovacím jazykem nesetkali.

Dalším společným rysem těchto 3D nástrojů je fakt, že každý z nich obsahuje určitou variaci příkazového řádku, odkud je možné vytvářené skripty jednoduše spouštět. Ve skutečnosti se ve většině případů jedná o samostatně konzolové okno, které pro psaní a vývoj skriptů obecně představuje velmi mocný nástroj, kterého lze využít především při vytváření menších skriptů nebo také při testování dílčích částí rozsáhlejších skriptů.
Mám-li si ovšem vybrat pouze jeden z těchto nástrojů, který bude pro psaní skriptů nejvhodnější, je nutné se samozřejmě zaměřit především na jejich rozdíly resp. zhodnotit klady a zápory těchto programů v souvislosti s vývojem skriptů.

Pokud budu posuzovat klady a zápory programu AutoDesk 3Ds Max je nutné si uvědomit, že jde o zástupce shareware, tedy softwaru chráněného autorským právem, za který bude potencionální uživatel muset dříve či později zaplatit. Tato prvotní investice, konkrétně v případě tohoto programu značně nemalá, může být překážkou a tak i určitou nevýhodou při výběru. Přesto je tento program velmi hojně rozšířený nejen pro účely skriptování, což je výhodou při hledání pomoci, spojené s tímto programem, nejen na internetu. Je tomu tak jednoduše proto, že čím více lidí tento software používá, tím snadněji se dostává k informacím o něm. Další silnou stránkou tohoto nástroje je bezesporu samotný skriptovací jazyk MaxScript, speciálně vytvořený pro tento software, jeho autorem je John Wainwright, který sám vytvořil sérii instruktážních videí seminářů volně dostupných na internetu a není tak problém si tento jazyk velmi brzy osvojit.

Následují dva zástupci freeware softwaru, tedy programů s otevřeným zdrojovým kódem. Konkrétně jde o programy Blender a Google SketchUp. U těchto dvou nástrojů tak odpadá potenciální nevýhoda spojená s peněžní investicí, určenou na jejich pořízení. Nicméně oba tyto nástroje jsou velmi kvalitní a mohou tak směle konkurovat svým komerčním rivalům. Konkrétně pak software od firmy Google se navíc může pyšnit velmi intuitivním ovládáním, na kterém si značně zakládá a díky němuž je ještě snazší se s tímto nástrojem naučit pracovat.

Přesto, anebo právě proto, že jde o Open Source produkty, je možné zde narazit na určité nedokonalosti spojené přímo s konkrétními programovacími jazyky, které se v těchto nástrojích používají pro vývoj skriptů. V případě Blenderu jde o skriptovací jazyk Python, který sám o sobě nepředstavuje problém, ovšem komplikace mohou nastat při jeho instalaci, která je vyžadována pro vývoj složitějších skriptů resp. všech skriptů, které generují jakékoliv iné, než triviální objekty do 3D scény programu Blender. V některých případech dokonce však aní samotná instalace Pythonu nemusí stačit. V případě programu Google SketchUp je používán skriptovací jazyk Ruby. Nevýhodu zde představuje především to, že tento jazyk není v našich končinách příliš rozšířený. Potenciální uživatel se tak bude potýkat s nedostatkem české dokumentace, což ovšem samozřejmě nemusí být vyloženou překážkou pro každého.

Poslední v analýze je program Rhinoceros. Jde opět o zástupce shareware softwaru, což ovšem představuje asi jedinou nevýhodu toho 3D modeláře. Jeho stále rostoucí popularita je založena na jeho rozmanitosti a schopnosti importovat a exportovat do více jak třiceti formátů. Rhinoceros tak může mimo jiné velmi dobře posloužit jako převodník mezi programy zbyvající za designem. Velkou předností je speciálně navržený skriptovací jazyk RhinoScript, který tento program používá. RhinoScript je založený na jazyce MS Visual Basic a lze tak počítat s jeho velmi dobrou dokumentací. Další užitečně rozšíření programu...
Rhinoceros představují četné plug-iny, jakými jsou např. Flamingo, Maxwell, nebo Grasshopper.

Jak již bylo zmíněno výše, pro tuto analýzu jsem zvolil čtyři hodnotící kritéria, kterým jsem přidělil různou váhu. Pro každé kritérium jsem po té jednotlivé programy ohodnotil body v rozmezí od nuly do čtyř a následně tyto body násobil zminěnými váhami, viz Tab. 1.

Tab. 1  Zhodnocení hodnotících kritérií

<table>
<thead>
<tr>
<th>Program</th>
<th>Kritéria</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
<th>K4</th>
<th>Celkem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autodesk 3Ds Max</td>
<td>Váhy</td>
<td>4(x)</td>
<td>3(x)</td>
<td>2(x)</td>
<td>1(x)</td>
<td></td>
</tr>
<tr>
<td>Blender</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Google SketchUp</td>
<td></td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>Rhinoceros</td>
<td></td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>35</td>
</tr>
</tbody>
</table>

Tab. 2  Hodnotící kritéria

<table>
<thead>
<tr>
<th>K1</th>
<th>Dostupnost programu</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Skriptovací jazyk</td>
</tr>
<tr>
<td>K3</td>
<td>Rozšiřenost programu</td>
</tr>
<tr>
<td>K4</td>
<td>Předchozí zkušenost s programem</td>
</tr>
</tbody>
</table>

Na základě provedené analýzy a všech hodnocených kritérií jsem se rozhodl vybrat si program Rhinoceros. Chtěl jsem využít především rozsáhlé dokumentace jazyka MS VBScript a také značných předchozích zkušeností, jednak s tímto jazykem, ale především s modelováním ve zmíněném nástroji. Pro vývoj skriptů tedy budu využívat program Rhinoceros konkrétně ve verzi 4.0 Educational, která je plně kompatibilní s uvedeným skriptovacím jazykem RhinoScript a u kterého jsem muží školní licence.

2.5 Skriptování v Rhinoceru

Po té co jsem se si zvolil program Rhinoceros pro modelování zmíněných propagacích předmětů, je nutné si osvojit základy, nezbytné pro vývoj skriptů v tomto 3D nástroji.

2.5.1 Makra

Nejnížší forma programování v programu Rhinoceros je reprezentována právě pomocí maker. Jde o primitivní, avšak účinný způsob jak automatizovat procesy v prostředí nejen tohoto programu.

Makra tvoří základ pro skriptování, nicméně se bohužel stává, že makra bývají se skripty chybějící zaměňovány. Mluvím-li o makrech, nejedná se o nic jiného než o posloupnost příkazů, které jsou vykonávány, tak jak jdou za sebou jeden po druhém, tak jako v následujícím příkladě.
Následující makro nejprve přidá do 3D scény v Rhinu čtyři navazující úsečky, postupně je všechny označí a poté se spojí do jedné, čímž se z těchto úseček stane čtverec. Pokud je žádoucí tímto způsobem vytvářet čtverec opakovaně, pak má smysl si toto makro vytvořit a pro tento účel jej používat.

```plaintext
_SelNone
_Line w0,0,0 w10,0,0
_SelLast
_Line w10,0,0 w10,10,0
_SelLast
_Line w10,10,0 w0,10,0
_SelLast
_Line w0,10,0 w0,0,0
_SelLast
_Join
```

Podtržitka před názvy všech příkazů jsou zde kvůli české lokalizaci programu Rhinoceros. Podtržitka totiž přinutí Rhino používat anglické názvy jednotlivých příkazů, namísto například českých, japonských, nebo zkratk, které by se měly používat bez ohledu na lokalizaci a vlastní nastavení programu. Je však doporučováno používat právě anglické názvy příkazů, protože jsou při použití v Rhinocerosu všude na světě.

Neexistují žádné hranice a omezení pro to, jak složitý se makra mohou stát. Je možné přidávat a psát další příkazy jeden za druhým bez omezení, avšak nikdy nebude možné omezení, které by je omezovalo.

2.5.2 Skripty

Hlavní rozdíl mezi makry a skripty je ten, že makra jsou pouze statická. Naopak skripty neobsahují pouze fixní sekvence a mají tak kontrolu nad tím, který řádek jejich kódu se provede jako další. Toto řízení toku dat tak umožňuje přeskočit určité instrukce v těle skriptu, nebo naopak některé části kódu opakovat vícekrát. V makrech dalej není možné pracovat s proměnnými a neporadí se ani s matematickými operacemi. Někdy jsou tak marky rovněž označovány za starou metodu skriptování.

Tato omezení, která sebou makra přinášejí, vedly k vývoji skriptovacích jazyků. Skripty jsou něčím na půli cesty mezi makry a skutečnými zkompilovanými programy a plug-iny. Jak již bylo zmíněno výše, vytváří skripty v programu Rhinoceros je možné již od verze 2.0 díky speciálně pro tento účel navrženému jazyku Rhinoscript, který byl postaven na základech jazyka VBScript. Lze tedy očekávat, že to co platí v jazyce VBScript platí i pro jazyk Rhinoscript.

2.5.3 Struktura skriptu

Základní struktura skriptu je vždy stejná a je vyžadováno její dodržování, pokud je vyžadováno správné provedení skriptu.
Skripty vytvořené pomocí jazyka Rhinoscript se skládají ze čtyř základních částí.

Obr. 5: Struktura skriptu

1. **Option Explicit**

Tato část skriptu je pojmenována podle příkazu Option Explicit, který je její nedílnou součástí. Uvádět zde tento příkaz není povinné, ale vysoce se to doporučuje. Pokud je skript spouštěn v režimu Option Explicit, je nutné deklarovat a definovat všechny proměnné, se kterými se má pracovat ještě předtím, než je samotný skript začne používat. Pokud je příkaz Option Explicit vynechán jsou veškeré proměnné deklarovány kompilátorem. To se může na první pohled jevit jako dobrá věc, ale jakékoli odladčování chyb a hledání problémů v tomto režimu může samotnou práci se skripty velmi znepříjemnit.

V části Option Explicit je dobré uvést alespoň stručný popis skriptu, tedy co je jeho účelem, popř. kdo je jeho autorem a kdy a kde tento skript vznikl.

```vbnet
Option Explicit
'Script written by JK
'Copyrighted by JK
'2011, Brno
'Toto je pouze ukázkový skript
```

Kromě příkazu Option Explicit tato část také obsahuje hned několik tzv. komentářů. Komentáře jsou bloky textu, které jsou kompilátorem a interpretem ignorovány. Řádky resp. text za levým apostrofem se vůbec nevytisknou, neprovedou, zkrátka budou přecházeny. Tak jako je tomu i v jiných
programovacích jazykůch, pomocí komentářů je možné přidat do kódu určité doplňující informace, nebo naopak dočasně zakázat provádění některých řádek kódu.

Na tomto místě je dobré vždy uvést informace o aktuálním skriptu, jako např. jméno autora, verzi skriptu, popř. kdy tento skript vznikl a jaký je jeho účel. Část Option Explicit se může jevit jako zbytečná, ve chvíli kdy obsahuje pouze komentované řádky, ale je tomu právě naopak. Jak samotná část Option Explicit tak i komentáře jsou nedílnou součástí všech skriptů a konkrétně komentáře velmi usnadňují čitelnost a orientaci v těchto skriptech.

2. **Main subroutine**

Každý skript musí obsahovat alespoň jednu funkci, jinými slovy podprogram, který obsahuje hlavní kód skriptu. Nemusí to být velká funkce, může jít například o funkci, která pouze zavolá další funkce, ale přesto je velmi důležitá a to z toho důvodu, že definuje rozsah skriptu. Skript začne pracovat ve chvíli, kdy je tato funkce zavolána a svou práci skončí právě ve chvíli, kdy dojde na konec této funkce. Zkrátka bez hlavní funkce se žádný skript neobejde a i tato část skriptu má svoji specifickou strukturu.

**Obr. 6: Struktura hlavního podprogramu**

Nejprve je nutné program Rhinoceros upozornit na to, že je vytvářen nový podprogram. Toho je docíleno za pomocí klíčového slova Sub, za kterým bezprostředně následuje jméno nově vytvářené funkce. V tomto konkrétním případě jde o funkci Main.

Po této deklaraci podprogramu je nezbytné deklarovat také všechny proměnné, se kterými má skript pracovat. Tak jak bylo pro deklaraci samotného podprogramu použito klíčové slovo Sub, tak stejným způsobem se o deklaraci jednotlivých proměnných postará klíčové slovo Dim. Následuje kód tedy tzv. tělo tohoto hlavního podprogramu. Nakonec je vše zakončeno.
klíčovými slovy End Sub, za kterými již není nutné uvádět jméno podprogramu, tak jako při jeho deklaraci a které celý podprogram uzavírají.

3. Additional subroutines and functions

Většina jednoduchých skriptů musí obsahovat pouze jeden podprogram. Ale je samozřejmě možné definovat neomezené množství podprogramů resp. funkcí, jestliže jsou potřeba.

Dobrým návykem je také snažit se samotný hlavní podprogram rozdělit do více funkcí, což mimo jiné celý kód rozdělí do smysluplnějších celků a zpřehlední jej tak. Tyto další funkce, procedury, podprogramy aj. se vytvářejí naprosto stejným způsobem jako funkce hlavní.

4. Execution command

Execution command reprezentuje první příkaz, který je programu Rhinoce-
ros předán. Kdykoli je totiž skript v Rhinu zavolán a spuštěn, interpret si nejprve přečte celý obsah tohoto skriptu a zkontroluje, zda jsou deklarová-
né všechny použité proměnné, a zdali skript neobsahuje syntaktické chyby. Přesto samotný Rhinoceros není schopen automatického spuštění hlavního podprogramu. Namísto toho Rhino vykoná vše, co je napsáno mimo hrani-
ce zmíněného podprogramu.

Proto je nezbytné Rhino předat požadavek na to, aby se konkrétní podprogram či funkce začal vykonávat. Toho je možné docílit uvedením názvu hlavní funkce na konec skriptu. V tomto konkrétním případě tedy musí být na konci skriptu napsané slovo Main.

```vba
1 Option Explicit
2 'Script written by JK
3 'Copyrighted by JK
4 '2011, Brno
5 'Toto je pouze ukázkový skript
6
7 Sub Main()
8    Dim strMessage1
9    Dim intCount
10    '...
11
12 End Sub
13
14 Main
```

Toto je příklad jednoduchého skriptu, který má ovšem kompletní požadovanou strukturu. Na řádcích 1 až 5 se nachází již dobře známá část skriptu Option Ex-
plicit. Následuje deklarace hlavního podprogramu na řádku 7 a následně i dvou proměnných, jedné řetězcové a druhé číselné na řádcích 8 a 9. Na tomto místě je tělo podprogramu neúplné a patrně by dále pokračovalo. Na řádku 12 je hlavní funkce tohoto skriptu ukončena a následně na řádku 14 volána, tak jak bylo vy-
věšteno výše. [13]
2.5.4 Skripty v prostředí programu Rhinoceros

Existuje několik způsobů, jak v Rhinoceru spouštět skripty, z nichž každý má svoje vlastní výhody a nevýhody. Jedním z těchto způsobů je ukládat skripty jako externí soubory. Všechny skripty jsou ve skutečnosti uloženy jako obyčejné textové soubory s příponou „.rvb“ (RhinoVisualBasic) a je tedy možné, je editovat v libovolném textovém editoru.

Takto externě uložené skripty stačí pak již jen podle potřeby volat v prostředí programu Rhinoceros za pomocí příkazů k tomu určených, které je možné vyvolat i přes kontextovou nabídku pod názvem Rhinoscript, která se nachází v nabídce Nástroje. Jedná se o příkazy _LoadScript pro nahrání skriptu a příkaz _RunScript pro spuštění již jednou nahraného skriptu.

![Obr. 7: Příkaz _LoadScript](image1)

![Obr. 8: Příkaz _RunScript](image2)

Další možností je pro vývoj skriptů použít vestavěný editor, který umožňuje jednou vytvářet, ale i spouštět skripty. Tato vestavěná konzole je tak ideální především pro implementaci dílčích funkcí a odladčování drobných problémů. Tento vestavený editor lze vyvolat příkazem _EditScript a následně vytvořené skripty spustit pomocí tlačítka Run.
Třetím způsobem nahrání je vložení skriptu do tlačítka. Tento způsob je podobný prvnímu způsobu, ale přesto nepatrně vhodnější, protože přirozeně je jednoduše kliknout na tlačítko než neustále dokola nahrávat skripty přes různé nabídky a dialogová okna. V tomto případě je ovšem nutné nejdříve vytvořit lištu s tímto tlačítkem. Pomocí příkazu _Toolbar, anebo nabídek Nástroje a Rozvržení nástrojových palet se zobrazí dialogové okno, kde je nutné pomocí nabídky Paleta vytvořit novou paletu.

Po zvolení této nabídky se otevře další dialogové okno, kde se nabízí možnost upravit vlastnosti této nové palety. Zde je vhodné změnit alespoň název palety. Je možné ji pojmenovat např. Skripty, ostatní parametry mohou zůstat bez změny. Po potvrzení této nabídky se tímto způsobem nově vytvořená paleta přidá mezi ostatní, již existující palety, ale především se do scény Rhina přidá nové,
prozatím prázdné tlačítko, které lze umístit kamkoli do scény programu. Pro editaci zmíněného tlačítka je nutné najet kurzorem myši na toto tlačítko a podržet klávesu Shift. V tuto chvíli se zobrazí nápověda o tom, jaké akce lze s tlačítkem provádět. Stisknutím pravého tlačítka myši se zvolí možnost Upravit a po té se otevře další dialogové okno, ve kterém lze již měnit vlastnosti samotného tlačítka, jako např. vzhled a popisek tlačítka.

Obr. 11: Editace tlačítka

Stěžejnější jsou ovšem pole pro příkazy, které se vykonají po kliknutí na nově vytvořené tlačítko levým, či pravým tlačítkem myši. A tak je možné např. do pole „Příkazy pro levé tlačítko myši“ vložit příkaz _LoadScript, určený pro nahrání skriptu a bezprostředně za něj do dvojitéch uvozovek absolutní cestu k tomuto souboru. Do stejného pole je možno obdobným způsobem umístit i příkaz _RunScript společně s celým skriptem, který se má spustit. Potvrzením této nabídky se vytvoří již plně funkční tlačítko, které je schopné zavolat a spustit použitý skript, kdykoli na něj uživatel klikne. [2]
3 Vlastní práce

Na základě teoretických východisek a prostudované literatury bylo dosaženo dostatečných znalostí v oblasti skriptování ve 3D modelovacích nástrojích, zejména pak v modeláři Rhinoceros. Nyní již nic nebrání shrnutí teoretických znalostí a vlastní implementaci jednotlivých skriptů propagačních předmětů.

Tato kapitola tedy rozebere zamýšlený postup práce formou ukázky implementace dílčích částí některých skriptů, ale především kompletním rozbořem jednoho ze skriptů jako vzorového příkladu. Náhledy výsledných modelů zbylých skriptů budou umístěny v příloze této práce.

Na závěr pak informuje o uživatelské příručce, která obsahuje pokyny potřebné pro dokončení procesu, určené pro potenciálního uživatele těchto skriptů resp. celé aplikace.

3.1 PR oddělení PEF Mendelu

Praktický výstup této práce byl vytvářen především na základě konzultace s vedením PR oddělení Provozně ekonomické fakulty Mendelovy univerzity v Brně, které řeší otázku výběru a zhotovení propagačních předmětů každý rok.

V současné době celý tento proces funguje následujícím způsobem. Zmíněné PR oddělení se každoročně vybírá propagační předměty pro své tři hlavní cílové skupiny, kterými jsou studenti a zaměstnanci PEF a zástupci firem. Pro každou cílovou skupinu se vybírá jiný druh propagačního předmětu tak, aby byl co nejvíce využitelný, tedy nejlépe, jak dobře vypadající, tak praktický. Jakmile je vybrán konkrétní předmět začíná se řešit umístění loga a barevné kombinace. To je řešeno pouze na základě spekulací a konzultace s konkrétními agenturami, které tyto předměty vyrábí, což není ideální.

Pokud by tedy právě tato část procesu mohla být realizována vizuálně ve 3D, velmi by to, podle slov Ing. Jiřího Urbánka (vedoucího PR), tomuto oddělení ulehčilo práci. Následně by též bylo možné např. takto vizualizované propagační předměty dát k hlasování zminěným cílovým skupinám, aby si samy vybraly ty, které se jim nejvíce líbí.

3.2 Výběr propagačních předmětů

Hlavním kritériem výběru propagačních předmětů byl především význam jejich modelování do 3D scény, tedy vytváření 3D modelů těchto propagačních předmětů. Rovněž na základě konzultace s vedením PR oddělení fakulty bylo dosaženo závěru, že nemá příliš velký význam modelovat např. jednotlivé kusy oblečení.
Pro tyto propagační předměty spolehlivě postačí jejich 2D náhled například v podobě obrázku. Na základě této skutečnosti jsem se rozhodl vybrat tyto předměty:

- Hrnek
- Propiska
- Přívěsek na klíče
- Deštník
- USB flash disk
- Podložka pod myš
- Blok

### 3.3 Modelování vybraných předmětů

Před samotnou implementací skriptů jednotlivých propagačních předmětů je dobré mít ucelenou představu o tom, jakým způsobem je možné tyto předměty vymodelovat, a to konkrétně v prostředí programu Rhinoceros.

Těchto pracovních postupů existuje nespočetné množství, ale přesto je důležité vybrat si právě jeden z možných způsobů postupu, což následně velmi zjednoduší implementaci samotného skriptu, určeného pro vymodelování konkrétních objektů. Všechny vybrané propagační předměty jsem si tedy nejdříve vymodeloval bez použití skriptů, abych si ujasnil jakým způsobem budu postupovat při automatizaci těchto modelovacích procesů.

### 3.4 Implementace skriptů

Pro implementaci jednotlivých skriptů jsem používal předejším konzolové okno, které je součástí programu Rhinoceros. Takto vytvořené skripty jsem pak ukládal jako samostatné externí soubory. Za účelem usnadnění vývoje těchto skriptů jsem si vytvořil vlastní panel s tlačítkem v prostředí tohoto modeláře, které mi usnadňovalo volání a spouštění zmíněných externě uložených souborů se skripty.

Při vývoji jednotlivých skriptů jsem také hojně využíval nápovědu jazyka RhinoScript, která obsahuje výčet všech funkcí, které je možné v tomto jazyce použít. A přestože většina těchto funkcí a metod počítá se zásahem uživatele při modelování generovaných objektů, snažil jsem se, psát skripty takovým způsobem, abych tento zásah uživatele co nejvíce eliminoval a aby tak tyto skripty mohli používat i ten, kdo nemá žádné předchozí zkušenosti s modelováním v Rhinoceru, či jiných modelovacích nástrojích.
3.4.1 Skript modelující propisovací tužku

Jak již bylo zmíněno výše, jako vzorový příklad implementace byl vybrán jeden ze skriptů, konkrétně skript určený na vymodelování propisovací tužky.

Samotnou propisovací tužku, stejně tak jako její model, lze rozdělit na několik částí. Stejným způsobem jsem rozdělil i skript, určený k jejímu vymodelování, do jednotlivých podprogramů, které jsou na konci tohoto skriptu jeden po druhém volány. Jak již bylo zmíněno v kapitole 2.5.3, tento postup významně zpřehledňuje čitelnost skriptu, stejně tak jako komentáře uvnitř kódu.

Na začátku skriptu tak nemůže chybět část Option Explicit, informující o jeho základních vlastnostech.

```
1 Option Explicit
2 ' © Jiří Krampol 2011
3 ' Bakalářská práce - Automatické generování propagačních předmětů pomocí skriptů
4 ' Propiska
```

Takto byly uvozeny pochopitelně všechny vytvářené skripty. Po této části následuje již implementace jednotlivých podprogramů skriptu. Prvním z nich je podprogram s názvem Propiska. Tato část skriptu se postará o vymodelování hlavní částí, tedy těla propisky.

```
5 Sub Propiska()
6 'Telo propisky
7 8 Dim arrPlane, arrPoint, arrNextPoint, arrCylinder1
9 Dim strCircle1, strCircle2, strCircle3, strCone1, strCone2
10
11 'Nastavení aktivního pohledu a souradnic
12 Rhino.CurrentView("Perspective")
13 arrPlane = Rhino.WorldYZPlane
```

Na začátku všech podprogramů nesmí chybět deklarace a případná definice používaných proměnných, tak jak je tomu i u ostatních programovacích jazyků. Deklaraci jednotlivých proměnných je dobré rozdělit podle jejich datových typů a rovněž samotné názvy proměnných jsou uvozeny zkratkami těchto datových typů.

Jazyk RhinoScript nabízí řadu datových typů, které je možné použít. Ve většině případů si však vývojář vystačí s těmi základními, kterými jsou:

- **Long**
  V některých případech bývá označovaný jako Integer. Numerické proměnné tohoto datového typu jsou schopny na svých 32 bitech uložit jakékoli celé číslo.
- **Double**
  Tento datový typ, taktéž určený pro numerické proměnné, je rovněž nazý- 
ván Real, z čehož je patrné, že je schopen opět na svých 32 bitech uložit ja- 
kékolí reálné číslo.

- **Boolean**
  Proměnné tohoto logického datového typu nabývají dvou hodnot. V případě 
Rhinocerscriptu se jedná o hodnoty True, nebo False.

- **String**
  Do proměnných tohoto typu je možné ukládat libovolně dlouhý řetězec a 
díky operátoru ampersand také přidávat text do již deklarovaných proměn- 
ných typu String.
  Proměnná typu String je pro práci v Rhinoceru velmi důležitá. Všechny 
objekty v jeho scéně, se kterými se pracuje, mají svůj jednoznačný identifi- 
kátor, který je uložen právě jako řetězec znaků. Kdykoli je tak potřeba např. 
zejistit informace o určitěm objektu pomocí funkcí, které Rhino nabízí, je 
nutné znát zmíněné ID tohoto objektu.

- **Array**
  Proměnná tohoto datového typu je schopná uložit neomezené množství 
proměnných stejného typu do jedné proměnné.
  Tento datový typ je tak vhodný především pro zjednodušení práce. 
K jednotlivým záznamům pole lze poté přístupovat pomocí indexů.

Syntaxe jazyka RhinoScript samozřejmě umožňuje se všemi těmito proměnnými 
pracovat pomocí logických operátorů a funkcí, které jsou používány stejným 
způsobem tak jako například v matematice.

```bash
14 'Pridání kruhu do sceny
15 Rhino.AddCircle arrPlane, 2.0
16 strCircle1 = Rhino.FirstObject
17
19 arrPoint = Array(0,0,0)
20 arrNextPoint = Array(-35,0,0)
21 Rhino.MoveObject strCircle1, arrPoint, arrNextPoint
22
23 Rhino.selectObject(strCircle1)
24 Rhino.command("_ExtrudeCrvTapered _Cap=_Yes ÚhelÚkosu 10 -10")
25 strCone1 = Rhino.FirstObject
26
27 Rhino.selectObject(strCircle1)
28 Rhino.command("_Extrude _Cap=_Yes 60")
29 strCone2 = Rhino.FirstObject
30
31 'Smazání křivky kruhu
32 If Not IsNull(strCircle1) Then
33 Rhino.DeleteObject strCircle1
34 End If
```
Vlastní práce

Tělo modelované propisky, jak je z komentářů uvnitř kódu patrné, jsem vytvořil za pomocí kruhu. Tento malý kruh, o průměru 2 cm, po té postupně vytáhnu do opačných směrů a tím tak vytvořím tělo a špičku propisky. Nakonec ještě smažu již nepotřebnou křivku malého kruhu (viz řádek 33).

Jak je vidět, většina těchto jednotlivých fází modelovacího procesu se ve skriptech děje za pomocí metod. Tyto metody jsou de facto bloky kódu, které jsou zabudovány přímo do jazyka RhinoScript a které je možné použít při vývoji skriptů. Všechny metody jsou zde v podobě příkazů, které jsou schopné vykonat určitou část modelovacího postupu, jako například přidat objekt do scény Rhina (viz řádek 15), změnit velikost, nebo posunout tento objekt (viz řádek 21) apod. Bohužel není tomu tak, že by pro každý příkaz, na který je uživatel z modelování v programu Rhinoceros zvyklý, existovala odpovídající metoda v jazyce RhinoScript. Pokud tedy nastane tato situace, je možné využít metodu RhinoScriptu, která bere jako svůj parametr konkrétní příkazy, které lze používat při samotném modelování v Rhinoceru (viz řádky 24 a 28).

Mimo jiné tento jazyk obsahuje mnoho dalších užitečných metod určených například pro výpočty v oblasti vektorové matematiky atp. Kompletní seznam všech metod a funkcí jazyka RhinoScript je možné nalézt v jeho nápovědě.

```vba
37 'Barva
38 Dim intIndex
39
40 'Obarvení první části propisky
41 If Not IsNull(strConel) Then
42    intIndex = Rhino.ObjectMaterialIndex(strConel)
43    If (intIndex = -1) Then
44        intIndex = Rhino.AddMaterialToObject(strConel)
45        End If
46
47    If (intIndex > -1) Then
48        Rhino.MaterialColor intIndex, RGB(0,70,160)
49        End If
50    End If
51
52 'Obarvení druhé části propisky
53 If Not IsNull(strCon2) Then
54    intIndex = Rhino.ObjectMaterialIndex(strCon2)
55    If (intIndex = -1) Then
56        intIndex = Rhino.AddMaterialToObject(strCon2)
57        End If
58
59    If (intIndex > -1) Then
60        Rhino.MaterialColor intIndex, RGB(0,70,160)
61        End If
62 End If
```

Následuje část kódu, která nastaví oběma modelovaným částem propisovací tužky výchozí zbarvení. V tomto případě jde o modré zbarvení, hodnot
RGB(0, 70, 160), které identifikuje Provozně ekonomickou fakultu Mendelovy univerzity v Brně, a to na základě jejího jednotného vizuálního stylu.

Zde je možné se také poprvé, ve větší míře, setkat se zmíněným hlavním rozdílem mezi skripty a makry a tím je řízení toku dat, které je v tomto případě reprezentováno příkazem If Then Else. Tento příkaz je jedním ze základních podmíněných příkazů a jeho sémantika je stejná jako u ostatních programovacích jazyků. V případě RhinoScriptu se nesmí pouze zapomenout na uvedení klíčových slov End If na konci tohoto příkazu. Ve většině případů se tento příkaz používá například pro ověření nějaké počáteční podmínky předtím, než je vykonán určitý příkaz, nebo zavolána ně která z metod RhinoScriptu.

V tomto konkrétním případě se tedy nejprve kontrolová, zda objekt, který má být obarven vůbec existuje, resp. existuje-li jeho jednoznačný identifikátor, pod kterým je tento objekt uložen (viz řádek 41). Pokud ano, do pomocné pro měněně se uloží identifikátor materiálu zmíněného objektu a na řádku 43 se opět za pomoci podmíněného příkazu kontrolová, zda je materiál pro tento objekt již nastaven. Pokud ne, materiál se objektu přiřadí a přejde se již k samotnému obarvení objektu, které je zajištěno díky metodě volané na řádku 48.

V podstatě jde o to, že se nejprve kontrolová, zda má propiska již nastavenou nějakou barvu či texturu, pokud ne, nastaví se propisce informace o tom, že je jí možné nějakou barvu nastavit a po té se jí již konkrétní barva přiřadí.

```
63 'Spojeni dvou casti propisky
64 Rhino.UnselectAllObjects
65 Rhino.selectObject(strCone1)
66 Rhino.selectObject(strCone2)
67 Rhino.command("_Group")
68 Rhino.UnselectAllObjects
69
70 End Sub
```

Poslední část tohoto podprogramu se postará o spojení obou částí propisky do jedné. Nejprve se tedy odznačí všechny vybrané objekty ve scéně Rhina zavoláním k tomu určené metody na řádku 64. Následně se postupně vyberou obě části propisky a spojí se v jednu část pomocí příkazu na řádku 67.

Následují dva jednodušší a kratší podprogramy, které se postarají o vymodelování, jednak tuhy propisky, jednak tlačítka, kterým se propiska zapíná. Nejprve tedy ta část skriptu, která vymodeluje tuhu propisovací tužky.

```
71 Sub Tuha()
72 'Tuha Propisky
73 74 Dim arrStart, arrEnd
75 Dim strTuha
76 Dim intIndex
77 Dim dblValue
78 79 dblValue = 255.0
```
I v tomto podprogramu se nejdříve deklarují všechny proměnné, které jsou používány a je zde definována i jedna z proměnných, která se později použije při obarvení modelované tuhy. Po té se na konkrétní místo ve scéně Rhina přidá krátká 0,2 cm dlouhá úsečka, ze které je následně pomocí příkazu _Pipe, resp. „Potrubí“ v české verzi tohoto programu, vytvoří tuha propisky se zakulacenými rohy o počátečním i koncovém poloměru 0,15 cm viz řádek 93. Následně se opět smaže již nepotřebná původní úsečka.

Následuje opět část skriptu, která se postará o obarvení modelované části, tedy tuhy propisky. V tomto případě je aplikována barva RGB(192, 192, 192), která má simulovat kovový povrch. Na tomto místě se rovněž použije již dříve definovaná proměnná, která společně s metodou volanou na řádku 110 nastaví modelované tuze maximální možný odlesk, pro ještě větší iluzi kovového povrchu. Celý podprogram je pak ukončen klíčovými slovy End Sub, která nesmí být opomenuta.

```plaintext
81 arrStart = Array(-45,0,0)
82 arrEnd = Array(-45.2,0,0)
83
84 If IsArray(arrStart) Then
85   If IsArray(arrEnd) Then
86     Rhino.AddLine arrStart, arrEnd
87   End If
88 End If
89
90 strTuha = Rhino.FirstObject
91 Rhino.selectObject(strTuha)
92
93 Rhino.command("_pipe _Cap=_Round 0.15 0.15 _cancel")
94
95 'Smazání krivky tuhy
96 If Not IsNull(strTuha) Then
97   Rhino.DeleteObject strTuha
98 End If
99
100 'Obarvení tuhy propisky
101 strTuha = Rhino.FirstObject
102
103 If Not IsNull(strTuha) Then
104   intIndex = Rhino.ObjectMaterialIndex(strTuha)
105   If (intIndex = -1) Then
106     intIndex = Rhino.AddMaterialToObject(strTuha)
107   End If
108   If (intIndex > -1) Then
109     Rhino.MaterialColor intIndex, RGB(192,192,192)
110   End If
111 End If
112 End If
113 End Sub
```
Další částí celého skriptu je již zmíněný podprogram na vymodelování tlačítka, kterým se propiska zapíná. To je modelováno na základě stejného principu jako tuha propisky.

Jediným rozdílem je zde poloměr vytažené úsečky viz řádek 132 a po té barva, která je tlačítku nastavena. V tomto případě jde o barvu RGB(255, 255, 255), tedy barvu bílou.

Pokud se v tuto chvíli nakonec skriptu dodají klíčová slova jednotlivých podprogramů a skript se otestuje v prostředí programu Rhinoceros, bude výstupem tento výsledek, viz obrázk Tělo propisky.
V této fázi ovšem skript, ani propiska nejsou ještě hotovy. Je potřeba přidat tzv. drţátka na konec a na začátek propisky. O to se postarají následující dva pod-
programy.

Nejprve tedy ta část skriptu, která vymodeluje gumové drţátko u hrotu propisky, a to následujícím způsobem. Nejprve je do osy těla propisky, a to do její přední části, vloţena křivka malého krouţku o průměru 1 cm, který je po té tzv. obalen potrubím, a to stejným způsobem, který byl pouţit jiţ dříve. Tato operace je pak provedena několikrát po sobě, přičemţ je původní pomocný krouţek du-
plikován a vţdy posunut o 1 cm po ose jiţ vymodelovaného těla propisky.

156 Sub PropiskaDrzatko()
157 'Gumove drzatko propisky
158
159 Dim arrPlane, arrPoint, arrNextPoint
160 Dim strKrouzek
161 Dim lngCount
162 Dim intIndex
163 arrPlane = Rhino.WorldYZPlane
164 Rhino.AddCircle arrPlane, 1.0
165 strKrouzek = Rhino.FirstObject
166
167 'posunuti
168 arrPoint = Array(0,0,0)
169 arrNextPoint = Array(-35,0,0)
170 Rhino.MoveObject strKrouzek, arrPoint, arrNextPoint
Při pohledu na řádek 172 je zřejmé, že k opakování zmíněné operace je zde dosaženo díky dalšímu způsobu řízení toku dat, kterým je v tomto konkrétním případě For Next cyklus. Tento typ cyklu je používán, pokud je žádoucí provést určitý kus kódu opakovaně, tedy vykonat určité příkazy právě n-krát, kde n je fixní počet opakování.

V jazyce RhinoScript je možné navíc specifikovat hodnotu kroku, která je standardně nastavena na hodnotu 1. Pokud by ovšem byla uvedenámu cyklu na řádku 174 nastavena hodnota kroku např. 3 (volitelný parametr Step), příkazy uvnitř tohoto cyklu by se neprovedly patnáctkrát, tak jak je tomu nyní, ale pouze pětkrát.

I v případě této již známé části skriptu na obarvení vymodelovaného dílu propisky je použit cyklus For Next. Musí být totiž znám jednoznačný identifikátor každé části, kterou má být obarvena. Tento identifikátor je následně předán metodě na řádku 189. Bylo by tedy možné upravit předchozí For Next cyklus tak, aby v každém svém kroku ukládal zmíněný identifikátor například do pole. Já jsem se ovšem rozhodl znovu použít uvedený cyklus, který zpětně projde přesný
počet právě vytvořených objektů a s těmi pak dále pracuje. V tuto chvíli to však již není patnáct průchodu cyklem, nýbrž 30, protože se je nutné pamatovat i na pomocné křivky, které se vytvářely v každém z kroku předchozího cyklu.

Poslední podprogram skriptu a tedy i poslední část propisky, která zbývá vymodelovat je držátko propisky umístěné na její zadní části. Tento objekt je vymodelován za pomocí křivky elipsy, která je opět vytažena do opačných směrů, tak jak tomu bylo i u těla propisky. Následně je vymodelována ještě spojovací část mezi propiskou a samotným držátkem, a to opět podobným způsobem, který byl použit již dříve.

208 Sub PropiskaKonec()
209 'Držatko na boku propisky
210 Dim arrPlane, arrPoint, arrNextPoint, arrStartK, arrEndK
211 Dim strElipsa, strKonec, strPart1, strPart2, strPart3
212
213 arrPoint = Array(0,0,0)
214 arrNextPoint = Array(10,0,2.75)
215
216 arrPlane = Rhino.WorldYZPlane
217
218 Rhino.AddEllipse arrPlane, 1.0, 0.5
219 strElipsa = Rhino.FirstObject
220
221 Rhino.MoveObject strElipsa, arrPoint, arrNextPoint
222
223 Rhino.selectObject(strElipsa)
224 Rhino.command("_Extrude _Cap=_Yes 15")
225 strPart1 = Rhino.FirstObject
226
227 Rhino.selectObject(strElipsa)
228 Rhino.command("_ExtrudeCrvTapered _Cap=_Yes ÚhelÚkosu 10 -2.5")
229 strPart2 = Rhino.FirstObject
230
231 If Not IsNull(strElipsa) Then
232 Rhino.DeleteObject strElipsa
233 End If

Jak již bylo popsáno výše, nejprve je tedy na řádku 219 vytvořena pomocná křivka elipsy. Ta je po té pomocí metody volané na řádku 222 posunuta na patřičné místo, vzhledem k pozici již vymodelovaných částí propisky. Následuje již samotné vytažení této elipsy, jednak přímo a to na řádku 225, jednak s úkosem viz řádek 229. Nakonec je tak jako vždy ještě smazána již nepotřebná křivka elipsy.
V této části kódu je na konkrétním místě vytvořena úsečka za pomoci již dobře známého příkazu na řádku 242 a následně je z této úsečky vytvořeno těleso viz řádek 249. Úsečka je následně smazána a všechny tři části modelovaného držátka vybrány a spojeny do jednoho objektu.

263 'Barva
264 Dim intIndex
265
266 'Obarvení první casti
267 If Not IsNull(strPart1) Then
268 Rhino.selectObject(strPart1)
269 intIndex = Rhino.ObjectMaterialIndex(strPart1)
270 If (intIndex = -1) Then
271 intIndex = Rhino.AddMaterialToObject(strPart1)
272 End If
273
274 If (intIndex > -1) Then
275 Rhino.MaterialColor intIndex, RGB(255,255,255)
276 End If
277 End If
278
279 'Obarvení druhe casti
280 If Not IsNull(strPart2) Then
281 Rhino.selectObject(strPart2)
282 intIndex = Rhino.ObjectMaterialIndex(strPart2)
Ve zbývající části tohoto podprogramu jsou všechny tři vytvořené díly držátka propisky postupně stejným způsobem obarveny bílou barvou. Je možné si povšimnout, že před samotným obarvením je každá z těchto částí nejprve označena ve scéně Rhina viz řádky 268, 281 a 294. Toto zdánlivě zbytečné označování objektů, je zde proto, aby vedlo oko potencionálního uživatele této aplikace. Jednoduše řečeno, jde o to, aby při sledování toho jak se propiska modeluje, měl tento uživatel stále přehled o tom co se děje, resp. že modelování ještě není u konce.

Při pohledu na tuto poslední část skriptu je patrné, že by bylo pravděpodobně vhodnější přesunout opakující se úsek kódu do samostatné funkce, která by jako parametry přebírala jednoznačný identifikátor objektu a barvu, kterou má být tento objekt obarven. Jedno z možných řešení by tedy mohlo vypadat např. takto:
Function Obarvi(strObject, intRed, intGreen, intBlue)
'funkce ktera obarvi zvoleny objekt prislusnou barvou

Dim intIndex, intR, intG, intB
Dim strOb
Dim blnResult

blnResult = False
strOb = strObject
intR = intRed
intG = intGreen
intB = intBlue

If Not IsNull(strOb) Then
    Rhino.selectObject(strOb)
    intIndex = Rhino.ObjectMaterialIndex(strOb)
    If (intIndex = -1) Then
        intIndex = Rhino.AddMaterialToObject(strOb)
    End If
    If (intIndex > -1) Then
        Rhino.MaterialColor intIndex, RGB(intR, intG, intB)
        blnResult = True
    End If
End If

Obarvi = blnResult

End Function

Takto vytvořenou funkci je možné použít na všech místech, kde je potřeba obarvit modelovaný propagační předmět, či pouze jeho část. Jak je patrné, samotná funkce je typu Boolean, čehož se dá velmi dobře využít např. pro kontrolu správného vykonání funkce při jejím volání uvnitř jednotlivých podprogramů.

'Obarveni treti casti
Dim blnPart3

blnPart3 = Obarvi(strPart3, 255, 255, 255)
If Not blnPart3 Then
    Rhino.MessageBox "Obejkt se nepodarilo obarvit", 48
End If

Tento příklad ilustruje obarvení třetí části držátka propisky za použití nově vytvořené funkce. Navíc je zde zmíněná kontrola správného vykonání této funkce. Pokud tedy nastane situace, že se předmět neobarví, je o tom uživatel informován prostřednictvím chybového hlášení. Takto vytvořená funkce značně ušetří místo, ale především výrazně zpřehlední samotný kód.
Ať už je použito jakékoliv z těchto možných řešení, nesmí se zapomenout na konec skriptu uvést klíčová slova jednotlivých podprogramů, čímž se Rhinu de facto oznámí, které části kódu se mají vykonat viz řádky 308 až 312. Po té již nezbývá nic jiného, než celkový skript otestovat v prostředí programu Rhinoceros.

![Obr. 13: Propiska](image)

### 3.4.2 Část skriptu modelující podložku pod myš

V této části již nebude popisovaný celý skript, tak jak tomu bylo u modelu propagačního předmětu propisovací tužky. Budou zde uvedeny pouze některé zajímavé konstrukce a části kódu, které jsou součástí skriptu modelujícího právě tento propagační předmět.


Zajímavou je zde ta část skriptu, která generuje zmíněná dvě tělesa. Pro vytvoření těchto dvou těles sice RhinoScript má své metody, ty však počítají se zásahem uživatele, resp. v podstatě pouze suplují kliknutí uživatele na tlačítko pro vymodelování těchto těles. Z tohoto důvodu jsem si vytvořil metody resp. funkce vlastní, které zmíněný zásah uživatele eliminují. Konkrétně šlo o funkce GetEllipsoid() a GetBox().
Function GetEllipsoid()
  'Funkce která vrací elipsoid
  Dim dblWidth, dblDepth, dblHeight
  Dim strCenter, strPoint1, strPoint2
  Rhino.CurrentView("Top")
  SetLocale "en-us" 'resi problem desetinne carky a tecky
  dblWidth = 5
  dblDepth = 10
  dblHeight = 5
  strCenter = " 0,0," & dblHeight
  strPoint1 = " 0," & dblWidth & "," & dblHeight
  strPoint2 = " " & dblDepth & ",0," & dblHeight
  Rhino.Command ("_Ellipsoid" & strCenter & strPoint1 & strPoint2 & " 0,0,0")
  GetEllipsoid = Rhino.LastCreatedObject
End Function

Function GetBox()
  'Funkce která vrací kvádr
  Dim strBoxpoint1, strBoxpoint2, strBoxpoint3, strBoxpoint4
  strBoxpoint1 = "-10.5,-5.5,-1"
  strBoxpoint2 = " 21"
  strBoxpoint3 = " 11"
  strBoxpoint4 = " 6"
  Rhino.Command ("_Box" & strBoxpoint1 & strBoxpoint2 & strBoxpoint3 & strBoxpoint4)
  GetBox = Rhino.LastCreatedObject
End Function

Obě tyto funkce vrací jednoznačné identifikátory vymodelovaných těles, které jsou po té použity při jejich zmíněním rozdílu a obě také využívají jednu z vlastností datového typu String.

Jak již bylo řečeno výše, v RhinoScriptu lze do této řetězcové proměnné uložit libovolně dlouhý řetězec znaků. Dále je však možné, již do deklarováných proměnných typu String přidávat text, a to pomocí operátoru ampersand, tak jak je to patrné na řádcích 18 a 33. V RhinoScriptu je navíc možné díky ampersandu přidat do Stringu i hodnoty numerických proměnných tak jak je tomu na řádcích 14 až 16. Nikdy ne však naopak.
3.4.3 Část skriptu modelující přívěsek na klíče

Tento skript resp. model propagačního předmětu se skládá ze dvou částí. První částí je kroužek, který slouží k připevnění přívěsku ke klíčům, druhou částí je samotný přívěsek. Zajímavostí je zde právě zmíněná druhá část přívěšku na klíče, resp. její barevné řešení.

```
1  'Barva
2  Dim strObjectColor, intIndex
3  'Obarvení přívěsku
4  strObjectColor = Rhino.FirstObject
5  If Not IsNull(strObjectColor) Then
6     intIndex = Rhino.ObjectMaterialIndex(strObjectColor)
7     If (intIndex = -1) Then
8          intIndex = Rhino.AddMaterialToObject(strObjectColor)
9      End If
10     If (intIndex > -1) Then
11          Rhino.MaterialColor intIndex, RGB(0,70,160)
12          Rhino.MaterialTransparency intIndex, 0.50
13      End If
14  End If
15  If (intIndex > -1) Then
16      Rhino.MaterialColor intIndex, RGB(0,70,160)
17      Rhino.MaterialTransparency intIndex, 0.50
18  End If
```

Tak jako většina předmětů je tato část opatřena modrou barvou, symbolizující barvy fakulty, za pomoci již známé části skriptu na obarvení vybraného objektu. Na základě požadavků na vzhled tohoto předmětu je mu však ještě nastavena poloviční průhlednost pomocí metody na řádku 15, čehož není využíváno u žádného z modelovaných předmětů.

3.4.4 Část skriptu modelující deštník

Skript generující model deštníku je tím nejrozsáhlejším ze všech ostatních. Je rozdělen do tří částí, kterými jsou držátko, kostra deštníku a plachta, která tvoří samotný deštník. Přestože je tento skript tak obsáhlý, zajímavostí je u něj opět způsob barevného řešení, a to konkrétně způsob jakým je obarvena zmíněná plachta deštníku.

```
1  'Barva
2  Dim strColor, intIndex
3  Dim dblCount, dblResult
4  'Vyber a obarvení listu
5  strColor = Rhino.FirstObject
6  For dblCount=1 To 8
7      Rhino.selectObject(strColor)
8          If Not IsNull(strColor) Then
9              intIndex = Rhino.ObjectMaterialIndex(strColor)
10             If (intIndex = -1) Then
11                Rhino.MaterialColor intIndex, RGB(0,70,160)
12                Rhino.MaterialTransparency intIndex, 0.50
13          End If
14      End If
15  End If
```

Tak jako většina předmětů je tato část opatřena modrou barvou, symbolizující barvy fakulty, za pomoci již známé části skriptu na obarvení vybraného objektu. Na základě požadavků na vzhled tohoto předmětu je mu však ještě nastavena poloviční průhlednost pomocí metody na řádku 15, čehož není využíváno u žádného z modelovaných předmětů.
Plachta deštníků se skládá z osmi dílčích částí. Uvedená část skriptu, tedy všechny části jednu po druhé nejprve vybere a následně obarví. To vše za pomoci již známého For Next cyklu. Opět na základě specifikace jsou ovšem jednotlivé části plachty deštníku obarveny střídavě modrou a bílou barvou. Toho je dosaženo díky dvou pomocným proměnným a podmíněnému příkazu Case viz řádek 17 a 18.

V proměnné dblCount je uložena hodnota kroku, ve kterém se celý cyklus zrovna nachází. Druhá pomocná proměnná dblResult v sobě v každé iteraci zmíněného cyklu ukládá zbytek po celočíselném dělení uvedené proměnné dblCount viz řádek 17. Na následujícím řádku vstupuje proměnná dblResult do zmíněného podmíněného příkazu a pokud je hodnota v ní uložená rovna jedné, obarví se konkrétní část deštníku modrou barvou viz řádek 20. Pokud je však zbytek po celočíselném dělení roven nule, skript pokračuje vykonáním příkazů na řádku 22 a obarví konkrétní část deštníku bílou barvou.

Tohoto podmíněného příkazu je využito i ve skriptu generujícím jedno ze dvou částí uživatelského rozhraní celé aplikace viz níže. Výsledná podoba modelu deštníku vypadá nakonec takto:
3.5 User Interface

Uživatelské rozhraní neboli souhrn způsobů, jakými může uživatel ovlivnit chování, nejen počítačových programů, je nedílnou součástí i mnou vytvářených skriptů resp. celé aplikace.

Potencionální uživatel se tak setká jednak s uživatelským rozhraním uvnitř samotného Rhinocera a zároveň s externím uživatelským rozhraním samotné aplikace. Tato rozhraní jsou však velmi intuitivní a uživateli známá i z jiných počítačových programů. Nemusí mít tedy obavy z toho, že by musel způsobem, který mu není znám, ovlivňovat např. vykonání samotného skriptu. Tato část procesu je plně automatizována.

3.5.1 UI v prostředí Rhinocera

Tato část uživatelského rozhraní je realizována rovněž pomocí skriptu a jednotlivých metod jazyka Rhinoscript. Tento skript se jmenuje „UI.rvb“ a stará se hned o několik funkcí (viz Skript uživatelského rozhraní).

Po zavolání tohoto skriptu se nejprve označí a odstraní vše, co se doposud vyskytovalo ve scéně Rhina a nastaví se základní rozvržení čtyř pohledů, tak jako při prvním spuštění tohoto programu. Po té se již před uživatelem objeví jednoduchá nabídka, ve které má možnost zvolit, který z nabízených propagačních předmětů chce vymodelovat. Po vybrání jednoho z propagačních předmětů je uživatel upozorněn, kterou nabídku zvolil a dále se již vybraný předmět začne

Všechna zmíněná upozornění jsou ve formě malých pop-up oken s jedním tlačítkem „OK“. Tato upozornění samozřejmě obsahují i patřičný text, který vede uživatele k tomu, co má dělat. Skript i samotná činnost programu je v tuto chvíli pozastavena, a to až do okamžiku, kdy uživatel klikne na zmíněné tlačítko „OK“.

3.5.2 UI samotné aplikace

Druhou část uživatelského rozhraní reprezentuje UI samotné aplikace. Tuto jednoduchou aplikaci jsem vytvořil prostřednictvím programu Visual Basic 6.0 Enterprise Edition. Po spuštění této aplikace se spustí program Rhinoceros a před uživatelem se objeví malé dialogové okno s jedním tlačítkem „Spustit skript“. Po té, co uživatel kline na toto tlačítko, zavolá se zmíněný skript „UI.rvb“, který uživateli již dále vede v procesu generování propagačních předmětů, tak jak bylo popsáno výše.

Pokud si uživatel např. vygeneruje jeden z předmětů a následně si bude chtít vygenerovat další, stačí již pouze opět kliknut na tlačítko „Spustit skript“ a celý proces se opakuje znovu. Naopak po té, co uživatel zavře toto dialogové okno, ukončí se nejen samotná aplikace, ale i program Rhinoceros.

Obr. 15: Uživatelská rozhraní
3.6 Uživatelská příručka

Součástí této práce je i příručka určená pro potencionálního uživatele. Konkrétně může jít např. o zaměstnance PR oddělení Provozně ekonomické fakulty. Tato příručka informuje uživatele mimo jiné o všech potřebných krocích, které je nutné učinit proto, aby aplikace Propagační předměty správně pracovala.

Stěžejní, právě pro zmíněného zaměstnance, je ovšem druhá část této příručky, ve které se má možnost seznámit s tím, jak ovládat program Rhinoceros do takové míry, aby byl schopen na generované předměty např. nanášet jakékoli textury, které bude chtít. Může jít například o logo univerzity, které si bude, po přečtení této příručky, sám schopen nanést na konkrétní propagační předmět a dále pak měnit jeho velikost, či další vlastnosti atd už této textury, nebo samotného předmětu.

V závěru jej tato příručka ještě upozorní na to, jakým způsobem má možnost si zmíněný upravený předmět vyrenderovat a tím tak dokončit celý proces automatického generování těchto propagačních předmětů. Výsledné modely propagačních předmětů mohou po té vypadat například takto:
4 Závěr

Tato práce se zabývá možností automatizace procesu vytváření trojrozměrných modelů propagačních předmětů pomocí skriptování. Jejím cílem bylo tedy využít metody skriptování za účelem vizualizace a nabytí jasných představ o podobě těchto předmětů, před jejich samotnou výrobou.

V kapitolách 1 a 2 jsou shrnuta teoretická východiska, jednak o počítačové grafice, jednak o propagačních předmětech. Součástí tohoto oddílu je i analýza a zhodnocení programů určených pro 3D modelování a následný výběr jednoho z nich. Závěrečná část této kapitoly již obsahuje potřebné východisko informace, o skriptování ve vybraném 3D modeláři, nezbytné pro praktickou část (kapitola 3) a konkrétní výstup této práce.

V úvodu praktické části je ujasněn účel a metodika této práce. Dále se již práce zabývá samotným vývojem skriptů v konkrétním skriptovacím jazyce, což je podrobně popsáno na příkladu jednoho z modelovaných propagačních předmětů. Je zde tedy ukázáno jakým způsobem je možné využít metody skriptování ke generování 3D modelů v konkrétním 3D modelovacím nástroji. V závěru této kapitoly jsou po té dopodrobna popsány jednotlivé kroky vedoucí k dokončení procesu modelování těchto předmětů.

Výstupem práce je jednoduchá aplikace, která je schopná přistoupit k modeláři Rhinoceros a volat zde jednotlivé skripty generující propagační předměty, tak jak uživatel zadá. Součástí je i uživatelská příručka určená pro potencionálního uživatele této aplikace, která jej informuje, jednak o aplikaci samotné, jednak o dalších rozšiřujících možnostech práce se zmíněnými modely propagačních předmětů.

4.1 Přínos práce

Stěžejní je přínos práce pro PR oddělení Provozně ekonomické fakulty Mendelovy univerzity v Brně, pro které zmíněná aplikace a obecně automatizace procesu vizualizace propagačních předmětů znamená velkou pomoc a usnadnění práce. Přestože byla celá práce vytvářená právě pro zmíněné oddělení, uplatnění by mohla nalézt i mimo univerzitu.

Mimo tento hlavní přínos je práce přínosná také pro výukové účely. Práce může být použита jako příklad základních, tak i pokročilých předmětů spojených s počítačovou grafikou a 3D modelováním. Samotným přínosem je totiž již postup zvolený při modelování jednotlivých předmětů, který může sloužit jako inspirace, či názorná ukázka při výuce modelování v programu Rhinoceros.

A přestože je skriptování v těchto 3D modelářích využíváno obecně za jinými účely, než je modelování propagačních předmětů, může tato práce posloužit i jako jeden z příkladů využití této metody.
Literatura

Přílohy
A  Skript uživatelského rozhraní

Option Explicit
'
© Jiří Krampol 2011
' Bakalářská práce – Automatické generování propagačních předmětů pomocí skriptů
' UserInterface

'Smazání jakéhokoli objektu, který se nachází na začátku ve třechni Rhino
Rhino.UnselectAllObjects
Rhino.InvertSelectedObjects
Rhino.command('_Delete')

'Odstraní všechny objekty včetně přidaného osvětlení sceny
'Rhino.command('_SelAll')
'Rhino.command('_Delete')

'nastavení počátečního pohledu
Rhino.command('_4View')
Rhino.command('_4View')

Dim arrPredmety
Dim strResult

arrPredmety = Array("Hrnek","Propiska","Přívěsek na klíče","Deštník","USB flash disk","Podložka pod myš","Blok","Nic")

If IsArray(arrPredmety) Then
    strResult = Rhino.ListBox(arrPredmety, "Vyberte propagacní předmět")
    If Not IsNull(strResult) Then

        Select Case strResult
            Case "Hrnek"
                Rhino.MessageBox "Vybral jste hrnek", 64
                Rhino.command("!-LoadScript""C:sers\Uživatel\Documents\Škola\MENDELU\tretak\letni_sem\BP\skript z\hrnek.rvb""")
                Konec()
            Case "Propiska"
                Rhino.MessageBox "Vybral jste propisku", 64
                Rhino.command("!-LoadScript""C:sers\Uživatel\Documents\Škola\MENDELU\tretak\letni_sem\BP\skript z\propiska.rvb""")
                Konec()
            Case "Přívěsek na klíče"
                Rhino.MessageBox "Vybral jste přívěsek na klíče", 64
                Rhino.command("!-LoadScript""C:sers\Uživatel\Documents\Škola\MENDELU\tretak\letni_sem\BP\skript z\privesek.rvb""")
                Konec()
            Case "Deštník"
                Rhino.MessageBox "Vybral jste deštník", 64
Rhino.command("!-LoadScript
"C:\Users\Uživatel\Documents\Skola\MENDELU\tretak\letni_sem\BP\skript\destni.k.rvb"")

Konec()

Case "USB flash disk"
    Rhino.MessageBox "Vybral jste USB flash disk", 64
    Rhino.command("!-LoadScript
"C:\Users\Uživatel\Documents\Skola\MENDELU\tretak\letni_sem\BP\skript\USB.k.rvb"")
    Konec()

Case "Podložka pod myš"
    Rhino.MessageBox "Vybral jste podložku pod myš", 64
    Rhino.command("!-LoadScript
"C:\Users\Uživatel\Documents\Skola\MENDELU\tretak\letni_sem\BP\skript\podlozka.k.rvb"")
    Konec()

Case "Blok"
    Rhino.MessageBox "Vybral jste blok", 64
    Rhino.command("!-LoadScript
"C:\Users\Uživatel\Documents\Skola\MENDELU\tretak\letni_sem\BP\skript\blok.k.rvb"")
    Konec()

Case "Nic"
    Rhino.MessageBox "Nevybral jste žádný propagační předmět", 64
    'Case Else
    'DoWhateverItIsYouWouldNormallyDo()
    End Select

End If
End If

Function Konec()
    'nastaveni finalniho pohledu
    Dim blnResult, strTitle
    strTitle = "Perspective"
    blnResult = Rhino.IsViewMaximized("Perspective")
    If blnResult = False Then
        Rhino.MaximizeRestoreView strTitle
    End If
    Rhino.command("_RenderedViewport")

    'Zobrazeni vlastnosti objektu
    Rhino.InvertSelectedObjects
    Rhino.command("_PropertiesPage Materiál")

    'info print
    Rhino.Prompt "Pokud nyní v pravo na kartě 'Vlastnosti' vyberete možnost 'Základní nastavení', budete mít možnost upravit barvu, lesk, průhlednost, či texturu právě vygenerovaného objektu."
    Rhino.MessageBox "Pokud nyní v pravo na kartě 'Vlastnosti' vyberete možnost 'Základní nastavení', budete mít možnost upravit barvu, lesk, průhlednost, či texturu právě vygenerovaného objektu.", 64
    Rhino.MessageBox "Pro úpravu vlastností je nutné mít vybranou, ale spojí jednu část vygenerovaného objektu", 64
End Function