

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHOLOGÍÍ

## *BAKALÁŘSKÁ PRÁCE*



Brno, 2005

Radek Kubíček

## Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením MgA. Roberta Chudého.

Další cenné informace mi poskytli:

- Milan Špaček
- Jiří Hnídek

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal, stejně jako osoby, od nichž jsem získal potřebné informace.

v Brně dne 26. dubna 2005

---

### **Abstrakt:**

Tato bakalářská práce se zabývá tvorbou interaktivní aplikace pro real-time průchod virtuální budovou. Pro její vypracování byl zvolen open source 3D modelační program Blender, převážně jeho součást GameEngine pro tvorbu real-time aplikací a her. Součástí práce je tvorba univerzálního engine pro snadný import modelu vytvořeného jinou aplikací do Blenderu, nastavení vlastností a parametrů modelu a možnost okamžitého interaktivního virtuálního průchodu v něm. Tvorba tohoto engine spočívá v napsání skriptu spouštěného přímo v prostředí Blenderu. Skript vytvoří grafické uživatelské rozhraní a provádí ho s potřebnými funkcemi.

### **Klíčová slova:**

počítačová grafika, interaktivní aplikace, real-time zobrazení, virtuální realita, průchod virtuální budovou, Blender, Blender GameEngine, stereoskopické zobrazení, skript, Python, GUI, grafické uživatelské rozhraní, 3D model, avatar, radiozita

### **Abstract:**

This work engages in creation of interactive application for the virtual real-time building model walkthrough. 3D modelling program Blender, especially its part GameEngine for games and real-time content creating, was selected for its elaborating. Part of this work is to create general-purpose engine for easy external model import into Blender, adjust its parameters and properties and enable possibility of immediate interactive model walkthrough. Creating of the engine consists in writing the script as a part of Blender, which makes possible to run it directly from the Blender world. The script creates graphics user interface which connects with necessary functions and abilities.

### **Key words:**

computer graphics, interactive application, real-time displaying, virtual reality, virtual building walk-through, Blender, Blender GameEngine, stereoscopic visualisation, script, Python, GUI, graphic user interface, 3D model, avatar, radiosity

# Obsah

<b>1</b>	<b>Úvod</b>	<b>3</b>
1.1	Stručný obsah kapitol . . . . .	3
<b>2</b>	<b>Požadavky a současný stav</b>	<b>5</b>
2.1	Současný stav oblasti real-time vizualizace . . . . .	5
<b>3</b>	<b>Program Blender</b>	<b>7</b>
3.1	Jak ho získat . . . . .	7
3.1.1	Stabilní verze . . . . .	7
3.1.2	Vývojové a ostatní verze . . . . .	7
3.2	Součásti programu Blender . . . . .	8
3.3	Zobrazovací schopnosti Blenderu . . . . .	12
<b>4</b>	<b>Tvorba pomocí Blender GameEngine</b>	<b>13</b>
4.1	Logic Bricks . . . . .	13
4.1.1	Vlastnosti a proměnné objektu . . . . .	13
4.1.2	Senzory (Sensors) . . . . .	14
4.1.3	Ovladače (Controllers) . . . . .	14
4.1.4	Aktivní prvky (Actuators) . . . . .	14
4.2	Možnosti skriptování . . . . .	15
4.3	Nástroj GameEngine pro detekci kolizí . . . . .	15
4.4	Nástroje GameEngine pro řešení pohybu . . . . .	16
4.5	Shrnutí tvorby v Blender GameEngine . . . . .	16
<b>5</b>	<b>Blender Walkthrough Engine</b>	<b>17</b>
5.1	Příprava a návrh . . . . .	17
5.2	Implementace interaktivní části . . . . .	18
5.2.1	Základ šablony avatara . . . . .	18
5.2.2	Úprava interaktivní šablony . . . . .	19
5.2.3	Možnost použití jiné šablony . . . . .	20
5.3	Podmínky a doporučení pro importovaný model . . . . .	22
5.4	Tvorba engine . . . . .	23
5.4.1	Spuštění skriptu . . . . .	23
5.4.2	Import modelu a nastavení kalibračního objektu . . . . .	24
5.4.3	Konfigurace naimportovaných objektů . . . . .	26
5.4.4	Výpočet a aplikace radiozity . . . . .	29
5.4.5	Nastavení real-time režimu a uložení projektu . . . . .	31

<b>6</b>	<b>Možnosti prezentace vytvořených dat</b>	<b>34</b>
6.1	Zobrazení pomocí webového prohlížeče . . . . .	34
6.1.1	Zjištěné problémy se zobrazováním . . . . .	35
6.2	Využití přehrávače BlenderPlayer . . . . .	36
6.3	Export do spustitelné aplikace . . . . .	36
6.4	Potíže při tvorbě a přehrávání real-time obsahu . . . . .	37
6.5	Rozšiřující možnosti . . . . .	38
6.5.1	Stereoskopické zobrazení . . . . .	38
6.5.2	Zabalení textur . . . . .	39
6.5.3	Zprůhledňování dveří . . . . .	39
<b>7</b>	<b>Závěr</b>	<b>41</b>
7.1	Návrhy pro pokračování práce . . . . .	41
<b>8</b>	<b>Slovníček pojmů</b>	<b>43</b>
<b>9</b>	<b>Přílohy</b>	<b>45</b>
9.1	Návod k použití Blender Walkthrough Engine . . . . .	45
9.1.1	Potřebné ovládání Blenderu . . . . .	45
9.1.2	Import modelu (obr. 5.2) . . . . .	46
9.1.3	Nastavení scény (obr. 5.2) . . . . .	46
9.1.4	Vlastnosti importovaných objektů (obr. 5.3) . . . . .	46
9.1.5	Nastavení radiozity (obr. 5.4) . . . . .	47
9.1.6	Parametry real-time režimu (obr. 5.5) . . . . .	47
9.1.7	Uložení výsledků (obr. 5.5) . . . . .	47
9.2	Obrázky . . . . .	48

# Kapitola 1

## Úvod

Počítačová grafika je již od počátků vzniku prvních grafických adaptérů oborem, který se neustále velmi rychle zdokonaluje a rozvíjí, a silně tak ovlivňuje technologický pokrok v oblasti počítačů. Posledních pár let se objevuje stále rostoucí zájem o virtuální realitu, real-time zobrazení a 3D stereoskopické zobrazení, ať již formou prezentací, zábavy nebo stále oblíbenějších 3D animovaných filmů.

Vytvořit trojrozměrný model lze mnoha způsoby. Dá se třeba využít různých 3D modelačních programů, naprogramovat jej pomocí některé z mnoha grafických knihoven, z nichž jsou některé prohlášeny za standard (OpenGL, DirectX, SDL a mnohé další) nebo vytvořit iluzi 3D prostoru různými optickými efekty (blending<sup>1</sup>, stereoskopické zobrazení<sup>2</sup> (viz část 6.5.1) a další).

V současné situaci, kdy je dostupná spousta 3D modelačních programů široké veřejnosti, je vytvoření modelu otázkou pár okamžiků. Ale možností převedení těchto modelů do interaktivního virtuálního zobrazení již zdaleka tolik není.

Za interaktivní aplikaci lze považovat takovou, ve které má uživatel možnost volného pohybu a ovlivňování děje zásahem do jejího běhu na základě svých požadavků. Výborným příkladem interaktivní aplikace jsou např. počítačové hry, převážně 3D akční hry z prvního pohledu – pohledu ovládané postavy. Některé programy sice dokáží vygenerovat průlet modelem podle různých trajektorií. Takovou aplikaci však nelze nazvat interaktivní, protože je uživatel omezen pouze na sledování předdefinované animace a nemá možnost její běh ani děj nijak ovlivnit.

Existuje sice spousta prostředí, grafických engine nebo konverzních nástrojů, které toto dokáží, ale většina z nich neobsahuje buď všechny potřebné funkce, jednoduchost a uživatelskou vstřícnost ovládání nebo alespoň přijatelnou cenu pro obvyčejného uživatele. Navíc některé neumožňují přímou tvorbu interaktivní aplikace, ale pouze výše zmíněné předdefinované animace. Jiné jsou pro laiky takřka nedostupné kvůli nutnosti vše naprogramovat.

### 1.1 Stručný obsah kapitol

První část pojednává o požadavcích a úkolech této práce, všeobecných problémech, současném stavu v několika zvolených projektech věnujících se virtuální nebo real-time grafice a aktuálních možnostech této oblasti.

---

<sup>1</sup>Používá se ve spoustě grafických efektů, převážně však ke zprůhlednění objektů. Při dobrém zvládnutí této metody lze nasimulovat trojrozměrný prostor. Více o blendingu např. na URL <http://www.root.cz/clanky/opengl-28-blending/>

<sup>2</sup>Technologie používaná různými zábavními komplexy a kiny – v České republice se s ní lze setkat např. v kině OskarIMAX 3D

V další kapitole se seznámíme s programem Blender jako celkem, zjistíme jeho vlastnosti, a součásti a povíme si, jak ho získat pro libovolnou platformu.

Následující část obsahuje popis tvorby v modulu Blender GameEngine, a to jak prostřednictvím rozhraní samotného Blenderu, tak pomocí skriptovacího jazyka Python. S tím je prostředí Blenderu kompletně provázáno pomocí Blender Python API a představuje mocný prostředek při tvorbě real-time obsahu. Jsou zde též zhodnoceny možnosti GameEngine, prostředky pro řešení základních interaktivních problémů, např. kolizí nebo pohybu, a potenciál Blenderu pro real-time zobrazování.

Další část popisuje proces tvorby uživatelského rozhraní engine. Jedná se o sadu skriptů, kompletně vytvořených pomocí prostředků nabízených Python API rozhraním Blenderu. Povíme si o implementaci jednotlivých částí rozhraní, jeho funkcích, možnostech jako celku a některých implementačních detailech.

V poslední kapitole se dozvíme o možnostech prezentace výsledků prostřednictvím webového rozhraní či samostatných spustitelných binárních aplikací. Budou zde zmíněna různá rozšíření, jejichž podrobnější popis by již byl mimo rozsah této práce nebo ještě nejsou plně implementována a mohla by se objevit v některé z příštích verzí. Též se zde dočteme o nedostacích a chybách, které se během práce vyskytly, a zjistíme zhodnocení výsledků práce. Nalezneme zde i návrhy pro její pokračování nebo možná rozšíření.



## Kapitola 2

# Požadavky a současný stav

Hlavním požadavkem je vytvořit funkční, interaktivní real-time aplikaci pro průchod virtuálním modelem budovy. Tvorba by měla probíhat co nejintuitivněji pro obvyčejného uživatele bez znalosti jakéhokoliv programovacího jazyka a nejlépe ve volně dostupné aplikaci. Úkolem tedy bylo najít freeware, open source nebo jiný snadno dostupný program, do jehož portfolia funkcí spadá import modelu budovy, provedení určitých akcí s tímto modelem a umožnění uživateli interaktivního průchodu v reálném čase.

Cílem této práce je seznámení s možnostmi interaktivního virtuálního průchodu modelem budovy a jedním ze zástupců této oblasti, 3D grafickým programem Blender. Ten spojuje modelační software, herní engine pro tvorbu real-time aplikací s podporou zvuku, video a audio editor a spoustu dalších užitečných modulů do jedné aplikace. Úkolem práce je zjistit potenciál Blender GameEngine v real-time oblasti, popsat různá úskalí tvorby v tomto modulu a dále vytvořit demonstrační aplikaci pro virtuální průchod naimportovaným modelem budovy. Dále je potřeba naimplementovat uživatelské rozhraní pro snadný import a nastavení vlastností.

Práce si též klade za cíl přiblížení virtuálního zobrazení a software Blender širšímu okruhu uživatelů a jejich seznámení s možností tvorby 3D her a real-time prezentací v software, který je zdarma, multiplatformní, aktivně vyvíjen a se silnou uživatelskou a vývojářskou komunitou a jehož instalační balíček nedosahuje ani 10MB velikosti.

Výsledkem této práce je již zmíněný engine pro import modelu uloženého v některém z podporovaných formátů do Blenderu, nastavení jeho parametrů a interaktivní průchod tímto modelem. Výsledný engine jsem pojmenoval jako **Blender Walkthrough Engine**.

### 2.1 Současný stav oblasti real-time vizualizace

Real-time grafika patří do oblasti počítačové grafiky generované, popř. zobrazované v reálném čase. Nejčastěji se s ní lze setkat při současném použití s virtuální realitou (VR). Cílem VR je poskytnout uživateli iluzi, že se nachází v jiném, umělém prostředí, nazývaném virtuální svět, scéna či prostředí. Dosažení pocitu přítomnosti uživatele ve virtuálním světě se docílí ovlivněním lidských smyslů, nejčastěji zraku a sluchu, vzácněji pak hmatu [3, str. 523].

Možnosti současných počítačů jsou pro real-time grafiku a VR více než dostačující. Na trhu je spousta aplikací, zaměřených pouze na tuto oblast, jiné kombinují možnost tvorby ve 2D a 3D prostoru a následnou konverzi těchto výsledků do VR nebo real-time zobrazení. Nicméně tato oblast se stále bouřlivě vyvíjí, protože ještě nebyla zbavena všech "dětských nemocí". Každým dnem vznikají nové projekty, jsou vylepšovány ty stávající a objevovány nové technologie. Počítačová grafika je tak více přístupná širší veřejnosti.

Vznikají nové technologie více přístupné obyčejným lidem, nové produkty jsou uživatelsky přívětivější a komplexnější. Budují se komplexy, které se specializují i na stereoskopické zobrazení (např. multiplexová kina IMAX 3D<sup>1</sup>). Zde se trojrozměrný film na filmové plátno promítá prostřednictvím dvou polarizovaných čoček (jedna vertikálně, druhá horizontálně), které na plátno odděleně promítají filmové pásy pro pravé a levé oko. Diváci mají na očích polarizované brýle, které jsou vybaveny čočkami stejného polárního zasměrování pro pravé a levé oko jako projektor. Pravé a levé oko tak dostávají pouze obraz pro ně určený a uvnitř pozorovatelova mozku se pak skládá iluze trojrozměrné reality.

Druhou variantou stereoskopických pohledů je, že jsou obrazy pro levé a pravé oko promítány odděleně a pravidelně se střídají. Tím se však musí zdvojnásobit počet zobrazování snímků při stejném framerate. Tento způsob VR je možné dekódovat pomocí helem pro VR (*HMD, Head Mounted Display*), kdy je obraz pro každé oko přenášen zvlášť a před každým okem je malá obrazovka. Obrazy jsou tedy odděleny prostorem, ale mohou být zobrazeny ve stejném čase. Více o způsobech stereoskopie je popsáno v [3, str. 529].

Existuje mnoho rozličných programů, modulů a grafických engine, v nichž lze dosáhnout interaktivního real-time průchodu v požadovaném modelu. Některé z nich jsou ale určeny pouze pro uživatele se znalostí některého programovacího jazyka, jiné zase finančně nedostupné. Zde bych rád stručně zmínil několik aplikací zastupujících jednotlivé skupiny.

- **Ogre3D** – grafický engine. Zástupce "programovací skupiny". Obsahuje širokou škálu funkcí, v nichž nechybí ani částicové efekty, práce s texturami a možnost importu modelu z externího souboru. Je multiplatformní, zdarma a stále vyvíjen. Jeho umístění v žebříčku oblíbenosti grafických engine je zpravidla na první příčce.

Mezi největší "nevýhody" patří již zmíněná nutnost vše naprogramovat. Jeho domovská stránka je k nalezení na<sup>2</sup>, kde lze engine stáhnout ve formě SDK do nejrozšířenějších vývojových prostředí. Též se zde nalézá množství různých zásuvných modulů zvyšujících jeho schopnosti a funkčnost.

- **TurboSquid Rtre** – interaktivní renderer, dodávaný jako plug-in do programů 3DS Max a Autodesk VIZ. Umožňuje vytvářet real-time fotorealistické 3D prezentace formou virtuálních procházek včetně zvuku. Modul podporuje real-time rendering statických 3D scén s rozlišením až 65 000 x 65 000 bodů, stejně jako nefotorealistické (umělecké) styly renderingu, napodobující např. malbu nebo kresbu. Jeho cena je ale příliš vysoká i pro většinu větších korporací. V USA lze pořídit za 1295\$. Tento modul je navíc použitelný pouze na platformách podporovaných výše zmíněnými aplikacemi. Více informací o projektu lze nalézt na stránkách projektu<sup>3</sup>.

- **FloorPlan 3D** – zástupce komplexnějších řešení. Jedná se o program typu CAD, zaměřený spíše pro stavební použití. Obsahuje zabudovaný modul pro okamžitou interaktivní real-time procházku vytvořenou budovou, včetně prohlídky exteriéru i interiéru. Letmé seznámení s tímto programem a jeho umístění ve srovnávacím testu snadno dostupných CAD aplikací můžeme nalézt v [4, str. 132]. Cena programu je přijatelná, i s českou lokalizací stojí 1999 Kč.

Bohužel, ani tento CAD systém nezvládá vše. Potíže mu činí např. složité nebo rozsáhlejší modely. Navíc je určen pouze pro platformu Windows. Také virtuální průchod modelem je umožněn pouze při spuštění aplikaci. Domovská stránka programu se nachází na webu výrobce<sup>4</sup>.

<sup>1</sup><http://www.oskarimax.cz>

<sup>2</sup><http://www.ogre3d.org>

<sup>3</sup><http://www.turbosquid.com>

<sup>4</sup><http://www.imsisoft.com>

## Kapitola 3

# Program Blender

Blender je jako 3D modelační nástroj velice dobře použitelný. Má sice docela nestandardní ovládání, na které se ale dá časem zvyknout, a tvorba je v něm potom velice rychlá a snadná. Obsahuje spoustu voleb a nástrojů. Některé se ve většině ostatních 3D modelačních aplikací nenacházejí a musejí být použity externí aplikace.

### 3.1 Jak ho získat

Vzhledem ke své open source politice je program Blender volně dostupný každému uživateli. Možností jeho získání je spousta, dokonce si uživatel může zvolit verzi, kterou chce používat. Může pracovat ve stabilní verzi, v některé ze starších verzí nebo sáhnout po jedné z vývojových verzí.

Dále existují různé projekty vznikající ze stabilních či vývojových verzí Blenderu a implementující nové užitečné funkce. Tyto verze však bývají zpětně kompatibilní, takže uživatel nepřijde o možnost přenosu souborů mezi různými verzemi.

Všechny níže zmíněné verze se kromě dále uvedených oficiálních zdrojů nachází ke stažení též na adrese <http://blendertestbuilds.kidb.de/>.

#### 3.1.1 Stabilní verze

Stabilní verzi Blenderu lze stáhnout z mnoha serverů po celém internetu, nejlépe však přímo z jeho domovské stránky <http://www.blender3d.org> pomocí odkazu *Download*. Kompletní přehled všech verzí lze nalézt na adrese <http://download.blender.org/release/>. Zde je možné získat aktuální i všechny starší stabilní verze, stejně jako webový modul či externí raytracer yafray.

#### 3.1.2 Vývojové a ostatní verze

Vzhledem k popularitě této aplikace a poměrně dlouhému vývojovému cyklu lze používat i některou z vývojových verzí, popř. verzi, která je vyvíjena jinou organizací či firmou. Tyto verze obvykle obsahují funkce navíc, které se ale časem synchronizují s hlavní vývojovou větví Blenderu, takže o ně nebudou ochuzeni ani uživatelé používající stabilní verzi.

- **Vývojové verze**

V současné době existují dvě vývojové větve Blenderu. Tzv. *bf-blender*, což je větev, z níž později vznikne stabilní verze a *Tuhopuu*, v současnosti s pořadovým číslem 3. Do ní jsou implementovány nové a neozkoušené funkce. Pokud se projeví jako užitečné a stabilní, jsou přesunuty do *bf-blender* větve.

Obě verze lze získat na vývojářském diskuzním fóru na adrese <http://www.blender.org> a v menu zvolit *Developers forum*. Zde se potom v sekci *Testing builds* nacházejí aktuální buildy obou verzí pro nejpoužívanější platformy (Windows, Linux, Mac OsX).

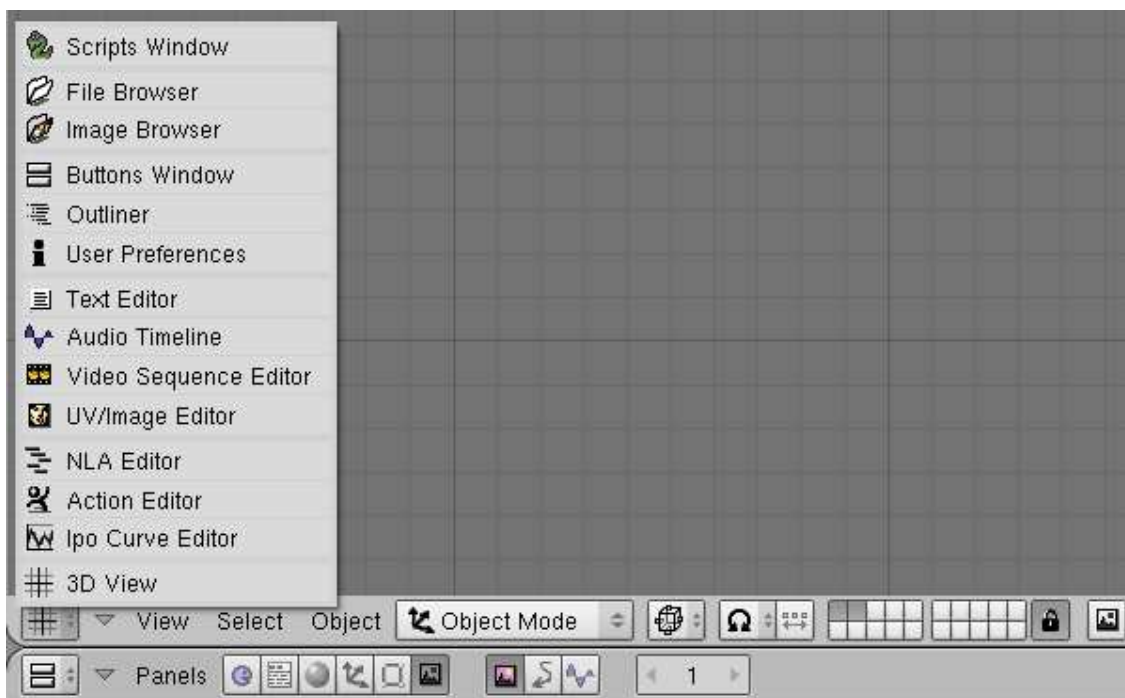
- **Ostatní verze Blenderu**

Asi nejznámější odnoží Blenderu je jeho verze od německé firmy Instinctive<sup>1</sup>, pojmenovaná *Instinctive Blender*. Jsou v ní implementovány některé funkce navíc (např. cloth simulator, vylepšený LOD – level of detail – a simulace vln) a snaží se skloubit nejnovější funkce a vlastnosti poslední stabilní a vývojové verze s grafickým rozhraním starší verze Blender Publisher, na nějž je spousta uživatelů stále navyklá a nepřechází kvůli tomu na novější verzi.

Aktuální i všechny předchozí verze jsou volně ke stažení na adrese webu výrobce<sup>2</sup>. Zde se také nachází popis všech vylepšení a ukázkové soubory demonstrující největší vylepšení.

## 3.2 Součásti programu Blender

Program Blender se skládá z více součástí. Ty všechny dohromady tvoří komplexní řešení pro tvorbu 3D modelů, animací, real-time obsahu, zahrnují jednoduchou práci se střihem zvuku a videa a mnoho dalších akcí. Menu obsahující všechny moduly, nejdůležitější součásti 3D okna a zbytek modulů, které jsou součástí **Buttons Window**, zobrazuje obr. 3.1.



Obrázek 3.1: Součásti Blenderu

- **Scripts Window**

Modul pro práci se skripty. Ty jsou rozděleny do jednotlivých sekcí a lze je odtud jednoduše

<sup>1</sup><http://www.instinctive.de>

<sup>2</sup><http://blender.instinctive.de/downloads/release/>

vyvolat. Po přidání nového skriptu je potřeba zavolat položku menu *Update Menus* a ten bude poté zahrnut do příslušného submenu.

- **File Browser**

Vyvolá dialog pro otevření souboru. V něm lze soubory volit stiskem prostředního tlačítka myši, popř. výběrem levého tlačítka myši a stiskem klávesy `Enter`. Tento modul by však měl být volán pouze pomocí skriptu, popř. tlačítka rozhraní Blenderu, jinak není funkční.

- **Image Browser**

Stejně jako předchozí možnost, pouze slouží k prohlížení a načítání obrázků.

- **Buttons Window**

Jedna ze stěžejních součástí Blenderu. Obsahuje funkční tlačítka, panely pro nastavení textur, světel, materiálů, vlastnosti objektu, výpočet radiozity a mnoho dalších. Pomocí této části lze ovládat většinu modelačních funkcí. Při implicitním sezení se nachází ve spodní části. Jeho podmoduly budou popsány později.

- **Outliner**

Topologický graf scény. Lze zobrazit buď jeho starší verzi, kdy jsou jednotlivé objekty a jejich součásti zobrazeny jako bloky propojené logickými spoji do grafu, nebo novější verzi, ve které jsou již seřazeny do přehledného stromu. Zjistíme z něj parentaci objektů, jejich materiály, IPO křivky a ostatní data nacházející se ve scéně.

- **User Preferences**

Hlavní menu, kterým lze ovládat práci s obsahem a aktuálním sezením. Dále je zde obsaženo nastavení lokalizace, fontů, témat vzhledu a ostatních uživatelských nastavení. Při implicitním sezení se nachází ve formě lišty úplně nahoře.

- **Text Editor**

Textový editor, který je přímo součástí Blenderu. Slouží ke psaní nejen textových poznámek, ale též skriptů, které lze přímo z editoru spouštět pomocí položky menu *File -> Run Python Script*. Ve vývojové verzi Tuhopuu obsahuje i zvýrazňování syntaxe. Samozřejmě lze při psaní skriptů využívat funkcí Blender Python API [6] nebo ostatních funkcí jazyka Python, je-li v počítači nainstalován a správně nakonfigurován.

- **Audio Timeline**

Jednoduchý lineární audio editor. Umožňuje načtení zvuku, jeho jednoduchou editaci, volbu sekvence a její přiřazení určité části animace. Jeho další část se nachází na liště modulu **Buttons Window**. Ve spojení s video editorem tvoří mocné stříhové studio, s jehož pomocí lze vytvářet vysoce kvalitní animace.

- **Video Sequence Editor**

Video editor, sloužící k tvorbě a úpravě animací. Mezi jeho schopnosti patří práce se sledem obrázků, videí, lze do něj dokonce vložit kompletní scénu sezení Blenderu a použít různé předdefinované efekty (zprůhlednění, prolnutí a mnoho dalších).

- **UV/Image Editor**

Editor obrázků, s jehož pomocí lze otexturovat objekty v tzv. *UV Face Select Mode*, kdy se na jednotlivé strany (faces) objektu mapuje požadovaná část textury. Tento postup je znám jako UV Mapping. K dispozici je i jednoduchý UV Unwrapper, který se pokusí vyhodnotit jednotlivé strany objektu a přiřadit jim příslušné části textury automaticky.

- **NLA Editor**

Editor pro tvorbu nelineárních animací. Lze v něm pomocí bloků prolínat různé animace a efekty. Používá se například pro tvorbu plynulých akcí, kde se musí zesynchronizovat IPO křivky, pohyb a střih, nejčastěji s metodou klíčování.

- **Action Editor**

Modul používaný pro tvorbu pohybu, kdy se pomocí kosterního systému (armature) vytváří pohyb po framech a zbytek je dopočítáván – tzv. metoda klíčování. Pokud je v pojmenování kostí určitý řád, **Action Editor** dokáže zrcadlit pohyb pro levou i pravou stranu.

- **IPO Curve Editor**

Editor IPO křivek, pomocí nichž se dá specifikovat pozice, rotace a nastavení jednotlivých barev objektů. Dále se používají k nastavení vlastností materiálů, textur, vertexů prolínání mezi snímky animace atd. IPO křivky jsou jedním z nejkompexnějších a nejmocnějších nástrojů. Používají se též v klíčování.

- **3D View**

Hlavní okno pro modelaci. Obsahuje mřížku, 3D kurzor a jsou do něj ”vykreslovány” všechny vložené objekty. Lze různě natáčet, měnit pohledy, měnit přiblížení a zobrazované informace.

Vedle rozvinutého menu se na obr. 3.1 nachází dvě lišty. Horní náleží k panelu **3D View** a spodní k **Buttons Window**. Na liště panelu **3D View** se za sebou nacházejí tyto ovládací prvky:

- **Menu**

V menu se nalézají všechny funkce dostupné v příslušném panelu. Každý panel má své specifické menu.

- **Pracovní mód**

Tímto tlačítkem se přepíná pracovní mód. Lze pracovat v následujících režimech.

- *Object Mode* – práce s objekty jako celkem.
- *Edit Mode* – práce s jednotlivými vertexy, hranami a stranami.
- *UV Face Select* – texturovací mód, ve kterém se vybírají jednotlivé faces a lze jim přiřadit požadovanou část textury.
- *Vertex Paint* – barvení vertexů. Těm lze přiřadit různé barvy, které se poté na hranách a stranách interpolují.
- *Texture Paint* – barvení textur. Stejně jako v předchozím případě, pouze s tím rozdílem, že lze kreslit na textury.
- *Pose Mode* – zakostňovací mód. Objeví se v menu pouze v případě, že pracujeme s kostním systémem (Armatures).

- **Mód zobrazení**

Zde lze zvolit způsob, jak budou objekty ve 3D okně zobrazeny. Na výběr je mezi

- *Bounding Box* – objekty budou zobrazeny pouze jako jejich drátěná obálka, která celý objekt zahrnuje.
- *Wireframe* – objekty budou zobrazeny jako vertexy a strany, které spolu tvoří drátěný model.

- *Solid* – z objektu jsou zobrazeny stěny, není však zahrnuto obarvení vertexů, stínování ani otexturování.
  - *Shaded* – stejné jako *Solid*, pouze jsou zahrnuty i stíny.
  - *Textured* – zobrazení stěn, zahrnující i otexturování a stínování.
- **Používaný pivot**  
Následujícím tlačítkem je možno přepínat centrálního pivota, podle něhož budou aplikovány veškeré polohové a rotační operace a operace změny velikosti. Mezi volbami je střed objektové obálky, střed objektu, 3D kurzor a střed shluku objektů. Tlačítko napravo slouží k přepínání, jestli se budou pohybovat celé objekty nebo pouze jejich středy. Při rotačních operacích můžeme pivota chápat jako osu otáčení.
  - **Zobrazené vrstvy**  
Stejně jako v každém kvalitním modelačním programu i v Blenderu lze pracovat ve vrstvách. Dvacet "čtverečků" představuje vrstvy a zobrazeny jsou všechny stisknuté.
  - **Zamknutí kamery a vrstev**  
Dalším tlačítkem lze uzamknout kameru a vrstvy k aktuální scéně.
  - **Rychlý render**  
Poslední tlačítkem se vyrenderuje aktuální pohled. Nastavení renderu je stejné jako poslední použité.

Panel **Buttons Window** je rozdělen na více podpanelů (modulů). Ty lze přepínat pomocí lišty. Tlačítka vlevo slouží k výběru požadovaného podpanelu, ta uprostřed upřesňují volbu a u každého podpanelu jsou jiná. Poslední ovládací prvek, obsahující číslici 1, zobrazuje aktuální frame. **Buttons Window** obsahuje v levé části tlačítek následující součásti:

- **Logic (GameEngine)**  
Modul použitý k implementaci engine. Slouží ke tvorbě real-time obsahu. Podrobnější popis jeho použití se nalézá v kapitole 4.
- **Scripts**  
V tomto podpanelu lze nastavit skripty, které se budou spouštět po provedení požadované akce. Skripty lze specifikovat pro celou scénu, objekty, okolní svět a materiály. Máme možnost nastavit skript vykonaný po načtení nebo uložení souboru, jeho spuštění nebo např. po změně snímku.
- **Shading**  
Podpanel obsahující nastavení materiálů, textur, výpočtu radiozity a činností s ní spojené a též nastavení okolního světa. Zde se nachází jedna důležitá položka pro tvorbu real-time obsahu – gravitační konstanta. Také je v něm obsažena konfigurace osvětlení.
- **Object**  
Nastavení objektu. Vlastnosti animace, vrstvy ve kterých se nachází, zobrazené informace ve 3D okně a též zde lze specifikovat různé efekty, např. vlny (waves) nebo částicové efekty (particles).
- **Editing**  
Volby pro editační mód a editaci objektu. Lze zde nastavit jméno objektu a příslušného meshe, přiřazené materiály, vytvářet skupiny vertexů pro tvorbu zakostění a další potřebná nastavení.

Dále se zde po vstupu do editačního módu nacházejí různé nástroje pro tvorbu a deformaci meshe, jako Screw, Spin nebo Spin Dup, pomocí nichž lze vytvářet šroubovitě nebo jinak pravidelné objekty. Najdeme zde i nastavení pívota objektu.

- **Scene**

Vlastnosti scény a renderu. Obsahuje možnosti ovlivnění výstupu, jako jméno výstupního souboru, použitý způsob renderingu a renderovací engine a formát výstupního renderu. Též se zde nachází nastavení animace, jejího formátu, popř. použitý kodek.

### 3.3 Zobrazovací schopnosti Blenderu

Blender interně používá spoustu nejznámějších a kvalitních algoritmů pro výpočet osvětlení a radiozity, detekci kolizí a mnoho dalších akcí. Zde zmíním jen některé důležitější algoritmy a principy zobrazení, ty zbývající lze dohledat ve zdrojích zmíněných na konci této části.

Stínování implicitně používá Gouraudova modelu, protože má mnohem kvalitnější výstup než konstantní model a není tak výpočetně náročný jako Phongův model. Algoritmy stínování využitelné pro spekulární (zrcadlový) odraz jsou CookTorr, Blinn a Toon, simulující stínování užívané animovanými filmy (cartoon). Difuzní odraz implicitně vyžívá Lambertův model, další možné jsou Oren-Nayar a již zmíněný Toon model.

Pomocí osvětlení typu Spot lze zobrazovat stíny – ty mohou být klasické, jemné (soft) a lze použít i Ray, popř. Buffered Shadows. Osvětlení může být i volumetrické, s jehož pomocí lze vytvářet Halo efekty.

Při výpočtu radiozity máme na výběr mezi konstantním nebo Gouraudovým modelem stínování. Algoritmy aplikované při výpočtu radiozity jsou jedny z nejkvalitnějších, zohledňujeme-li rychlost výpočtu.

Dokáže zobrazit průhledné objekty, jejichž průhlednost lze definovat různými způsoby. Jedním z možných způsobů je texturová maska, kdy černá barva udává maximálně průhledný a bílá barva neprůhledný pixel. Tuto masku lze kombinovat s modelem různými způsoby, např. sečtením, odečtením a jinými operacemi. Též lze využít alfa položky materiálu, jejíž hodnotu lze zadat pomocí IPO křivek.

Při práci s texturami lze využít metod mipmappingu, displacement, bump-mapping a environmentálních textur, popř. UV mapování, pro nějž je vytvořen i jednoduchý Unwrapper. Problémy mu nečiní ani obarvené vertexy (colour vertex) či textury (texture painting).

Během real-time zobrazení je aktivována funkce ořezávacích rovin, kdy jsou objekty ležící mimo rozsah zorného pole a blízkého okolí dočasně zneviditelněny kvůli rychlejšímu a méně paměťově náročnému výpočtu scény. Real-time režim dokáže zobrazit texturované modely a obarvené vertexy ať již formou přímého barvení nebo radiozitivním výpočtem. Též dokáže zobrazit stínované modely a aplikovat nastavení světla ve scéně.

Algoritmus pro detekci kolizí využívá hierarchii obálek. Jejich tvar lze určit pro každý objekt zvlášť. Tento algoritmus však při složitých scénách může vykazovat snížení snímkové frekvence. Proto je používán i algoritmus časově závislých kolizí. Popis a princip těchto algoritmů nalezneme např. v [3].

Pro render lze využít interní renderer, popř. některý z externích raytraycerů (např. Yafray, pro nějž má Blender přímo zabudovanou podporu) pro dokonalější fotorealistický výsledek.

Popis algoritmů používaných programem Blender a jejich použití lze nalézt v [1] a na internetových stránkách<sup>3</sup> a<sup>4</sup>. Podrobněji se o zmíněných algoritmech můžeme dočíst např. v [3].

<sup>3</sup><http://www.blender.org>

<sup>4</sup><http://www.blender3d.org>



## Kapitola 4

# Tvorba pomocí Blender GameEngine

Blender GameEngine je modul pro tvorbu interaktivního real-time obsahu implementovaný přímo do prostředí Blenderu. Tvorbu v něm je možné rozdělit do dvou na sobě nezávislých částí. Ty se ale dokonale doplňují a při společném použití tvoří velice mocný prostředek k vytváření real-time interaktivního obsahu.

### 4.1 Logic Bricks

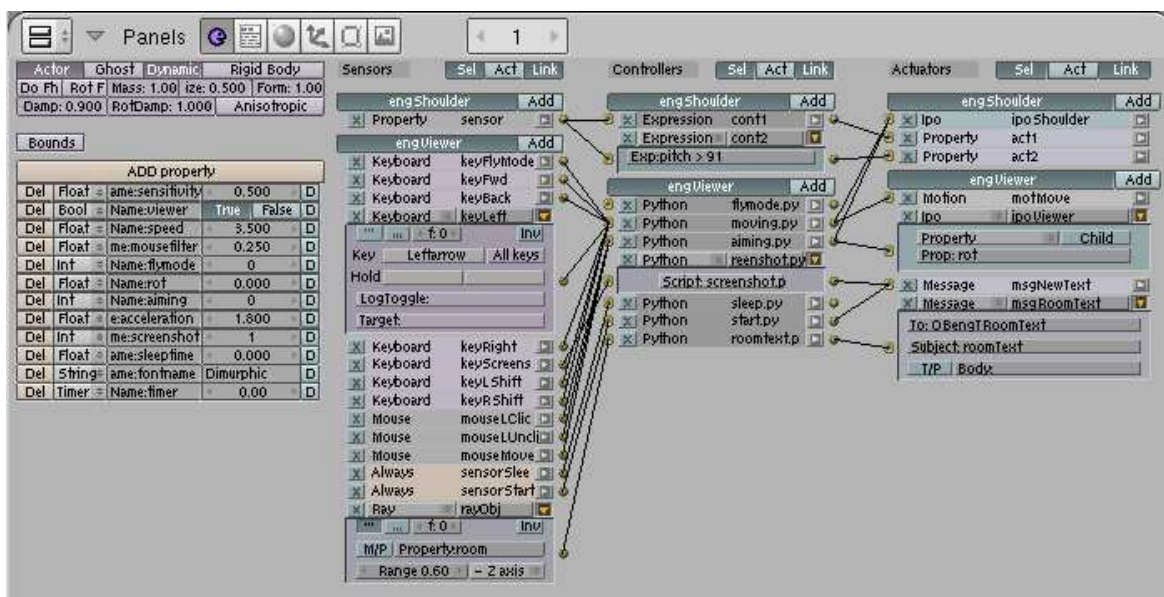
Lze v něm pracovat pomocí uživatelského rozhraní Blenderu, kdy se veškerý real-time obsah, aktivní prvky a reakce na události dá nadefinovat pomocí tzv. Logic Bricks – logických bloků (obr. 4.1). Tento modul lze rozdělit na 4 hlavní části (sloupce), které si nyní postupně popíšeme.

Logické cesty se vytvářejí pomocí kliknutí levým tlačítkem myši na kruh vedle počátečního bloku a následném tažení až do kruhu vedle požadovaného cílového bloku. Zde tlačítko myši pustíme a měla by se objevit linka propojující zvolené bloky. Propojovat můžeme pouze sousední bloky – nelze propojit Senzor a Aktivní prvek. Již vytvořenou cestu můžeme zrušit najetím myši nad požadovanou linku, kterou si přejeme zrušit a stiskem kláves `Del` nebo `X`.

#### 4.1.1 Vlastnosti a proměnné objektu

Úplně vlevo nahoře se nachází blok ovládacích prvků, pomocí nichž lze nadefinovat vlastnosti objektu. Možnosti interakce s okolím, jestli se pro něj mají detekovat kolize, případně zda je nehmotný (tzv. Ghost mode). Dále jeho odrazivost od stěn a podlahy, vektor tření a další nastavení týkající se pohybu a detekce kolizí.

Pod tímto blokem je vidět blok pro specifikaci proměnných patřících k objektu. Ty mohou být 5 různých typů – časovač, číslo s plovoucí desetinnou čárkou, celé číslo, řetězec a logická hodnota – a lze je využít jak pro uchovávání dat během práce v Blenderu, tak v rámci real-time zobrazení. Proměnné lze vytvářet staticky přímo v prostředí pomocí tlačítka **ADD Property** nebo dynamicky pomocí skriptů. To je ale umožněno pouze skriptům běžícím mimo real-time režim – tam lze totiž volat funkce pouze z Python Game Logic API [7], nebo externích skriptů, k nimž je nastavena cesta. Proměnnými lze též určit identifikaci objektů a tu poté využívat pomocí senzorů pro řešení kolizí (viz část 4.3).



Obrázek 4.1: Logic Bricks

### 4.1.2 Senzory (Sensors)

Tlačítka ve druhém sloupci se nazývají senzory. Používají se k detekci událostí. Lze si vybrat ze spousty typů, od senzoru klávesnice, myši, obecného generátoru impulzů, či radaru až např. po detektor vzdálenosti od objektu, kolize s objektem nebo detekčního paprsku vyslaného podél požadované osy.

Pomocí senzorů se definuje událost, na kterou se má reagovat a podrobně specifikují její podmínky. Pokud je tato událost během real-time běhu zachycena a jsou splněny všechny podmínky, senzor vyšle pulz zvolené orientace (pozitivní, negativní), který může být vyslán pouze jednou nebo opakovaně v daném intervalu.

Impulz se po vyslání přesune po své "logické cestě" do specifikace ovladače impulzu.

### 4.1.3 Ovladače (Controllers)

Ovladači upřesňujeme, jakým způsobem mají být impulzy zaslané senzory aplikovány. Lze zvolit mezi logickým součtem (OR) a součinem (AND), výrazem (Expression) a skriptem Pythonu (Python). Pomocí logických operátorů lze spojit více impulzů do jednoho celku a ten se potom vyvolá pouze při jejich odpovídající kombinaci. Výrazovým ovladačem lze naopak poslat impulz ze senzoru do aktivního prvku pouze tehdy, kdy daný výraz nabývá platnosti. Posledním ovladačem lze po obdržení impulzu ze senzoru zavolat interpret jazyka Python na zvolený skript, který se poté kompletně provede.

Jestliže jsou splněny všechny podmínky ovladače, impulz se přesune do poslední části své logické cesty, samotného aktivního prvku. Ten provede požadovanou akci, a to buď na základě impulzu příslušné orientace nebo pomocí API volání z ovládacího skriptu.

### 4.1.4 Aktivní prvky (Actuators)

Aktivní prvek je zakončením logické trasy impulzu. Zde se specifikuje požadovaná akce, která bude při úspěšných předchozích podmínkách provedena. Lze opět zvolit mezi spoustou typů, od prvků

pohybových, IPO volání, prvků ovládajících real-time sezení, objekt, scénu, posílání zpráv až po ty, týkající se práce se zvukem, CD-ROM a další.

Jestliže dojde impuls až do některého z aktivních prvků (nebo pokud byl tento prvek aktivován pomocí skriptu), dojde k vykonání specifikované akce.

## 4.2 Možnosti skriptování

Druhou možností tvorby v GameEngine je využití interpretovaného jazyka Python a skriptů. Ty lze tvořit též jako nedílnou součást obsahu modulem **Text Editor** nebo nalinkovat externí skripty.

Při psaní se dá využívat bohatého Blender GameEngine Python API, jehož dokumentace je volně ke stažení [5]. Využívá se pro tvorbu interaktivního pohybu, detekci objektů, práci s objekty a mnoho dalších akcí nezbytných ke tvorbě kvalitního real-time obsahu.

Blender GameEngine Python API je rozděleno na tři moduly – **GameLogic**, ten obsahuje vše, co se týká logických bloků a objektů ve scéně, **Rasterizer** obsahující funkce ohledně zobrazování, jako uložení snímku obrazovky, získání a úprava rozměrů okna a nastavení rozteče eye-separation při stereoskopickém zobrazení a **GameKeys**, který obsahuje symbolické názvy všech možných kláves a konstant.

Skripty mohou vykonávat téměř vše, co jde specifikovat logickými cestami, kromě vytváření nových logických bloků. Ty bohužel musí být vytvořeny vždy procesem návrhu mimo real-time režim. Skripty lze zavolat jen pomocí ovladače Python a jsou vždy provedeny kompletně celé – pokud některý ze skriptů obsahuje čekání nebo nekonečný cyklus, aplikace se zpomalí, respektive zacyklí. Během real-time běhu se v nich též nesmí vyskytnout příkaz `print`. Viz část 6.4.

## 4.3 Nástroj GameEngine pro detekci kolizí

Blender GameEngine přímo v sobě obsahuje jednoduchý, ale mocný nástroj pro detekci kolizí. Pro každý objekt lze pomocí ovládacích prvků specifikovat, zda se pro něj mají zjišťovat kolize, jestli má pevnou polohu nebo je pohyblivý, popřípadě jeho průhlednost a nehmotnost. Pro detekci kolizí lze určit tvar obálky objektu, která může mít tvar např. krychle, koule, jehlan, dokonce i polyhedru.

Při detekci kolizí se též aplikuje nastavení vektoru tření, který lze nastavit pro každou osu zvlášť. Dále je v real-time módu používána gravitační konstanta. Její nastavení můžeme nalézt ve World části panelu **Buttons Window**. Gravitaci lze též měnit dynamicky skriptem pomocí následující funkce.

```
GameLogic.setGravity(hodnota_gravitacni_konstanty)
```

Možnosti detekce kolizí, které většinou nebyly aplikovány v tomto projektu, ale jsou v modulu obsaženy, se týkají senzorů. Některé typy senzorů jsou určeny pro kolizní interakci. Jedná se o následující senzory:

- **Touch** – vyšle impuls, jestliže se avatar dostane do kontaktu se zadaným materiálem.
- **Collision** – indukuje impuls, pokud se avatar dostane do kolize s objektem obsahujícím zadanou proměnnou.
- **Near** – vygeneruje impuls při přiblížení avatara do požadované vzdálenosti k objektu obsahujícímu zadanou proměnnou.
- **Radar** – vyšle impuls, když se v daném poli radaru, specifikovaném pomocí úhlu rozevření a hloubky kuželu, nachází objekt s požadovanou proměnnou. Tento senzor býval hojně využíván

při minimalizaci počtu objektů ve scéně, kdy jsou viditelné pouze ty v dosahu radaru. Ve verzi 2.36 je již ořezávání scény naimplementované přímo jako součást Blenderu a je aplikováno automaticky.

- Ray – vyšle paprsek zadaným směrem a do požadované vzdálenosti. Pokud narazí na objekt se zadanou proměnnou, generuje impuls. Tento senzor je využíván např. při akčních hrách na pozici zaměřovače. Jako jediný byl použit i v této práci na rozpoznání, jestli se avatar nachází v místnosti (viz část 5.4.3).

## 4.4 Nástroje GameEngine pro řešení pohybu

Pohyb lze řešit oběma uvedenými způsoby tvorby. Chceme-li vytvořit interaktivní pohyb pomocí Logic Bricks, stačí použít senzor Keyboard, respektive Mouse a propojit ho s aktivním prvkem Motion nebo Constraint. Tímto způsobem lze sice snadno vyřešit pohyb, ten ale bude vždy pevné velikosti (rychlosti) a při velkém množství možností pohybu může být příliš obtížné orientovat se ve spoustě logických cest.

Mnohem mocnějším nástrojem pro řešení pohybu je využití skriptů. Stačí propojit všechny pohybové senzory a ovladač příslušného skriptu. V něm poté vše kontrolovat a nastavovat. Nakonec lze skriptem zavolat jediný připojený aktivní prvek, jehož vlastnosti se nastaví rovnou pomocí skriptu a prvek pouze odešle informaci do real-time prostředí.

Tímto způsobem je řešen pohyb i v Blender Walkthrough Engine. Zde se o něj starají dva spolupracující skripty, kdy jeden obsluhuje události myši a druhý události klávesnice.

## 4.5 Shrnutí tvorby v Blender GameEngine

Jak můžeme vidět, tvorba real-time obsahu v GameEngine je velice snadná a promyšlená a práce v něm jde rychle od ruky. Na druhou stranu, materiálů popisujících tento modul je příliš málo na to, aby bylo umožněno v něm již od začátku pracovat naplno. Nabízí totiž ohromnou škálu možností.

Pomocí GameEngine lze vytvořit real-time obsah s otexturovanými modely, radiozitou, colour vertexy. Obsah interaktivní i neinteraktivní. Jeho potenciál je obrovský vzhledem k obtížnosti tvorby v něm. Obsahuje funkce pro automatickou detekci a řešení kolizí, možnosti pohybu jsou zde též téměř neomezené. Jediné omezení je při použití skriptů, kdy musí být užíváno pouze GameEngine Python API nebo externích skriptů, jejichž cesta musí být správně přilinkována. Nelze používat ostatní skripty uložené v .blend souboru.

Existují i knihy zabývající se touto částí Blenderu, ale u nás jsou nedostupné a navíc se povětšinou zabývají GameEngine verze 2.25, ve které jistě spousta začátečníků tvořit nebude. Také je volně ke stažení popis Blender GameEngine Python API a spousta tutoriálů a ukázkových aplikací [5].

Vzhledem ke své snadné dostupnosti a intuitivnímu ovládaní by si tento herní engine mohl získat spoustu příznivců, o čemž svědčí i poměrně vysoká návštěvnost diskuzního fóra na českém serveru uživatelů Blenderu<sup>1</sup>.

---

<sup>1</sup><http://www.blender3d.cz>

## Kapitola 5

# Blender Walkthrough Engine

Jedním z klíčových úkolů práce je vytvoření interaktivní real-time aplikace, která bude umožňovat průchod libovolným modelem budovy (ten však musí splňovat určité podmínky, o nichž se dočteme v části 5.3 na straně 22). Implementace se tak rozděluje na dvě části – vytvoření interaktivních prvků a práce se samotným modelem. Implementace mohla proběhnout mnoha způsoby, např. naprogramováním pomocí některé z grafických knihoven (OpenGL, DirectX). Tento způsob by se ale nakonec mohl projevit jako příliš náročný, zdlouhavý a navíc se nejedná o způsob, který by byl adaptivní. Interaktivní část by sice mohla zůstat neměnná, ale každý další model by musel být znovu naprogramován.

Další cestou mohlo být vytvoření modelu v některé z modelačních aplikací. To je docela snadné i pro ne příliš zdatné uživatele. Ale v těchto aplikacích lze většinou vytvořit jen statický model bez jediného náznaku interakce. Dalším problémem tedy je, jak tento statický model dostat do real-time prezentace.

### 5.1 Příprava a návrh

Zvolil jsem možnost využití modulu Blender GameEngine. Jak jsme se dozvěděli v předchozí kapitole, je tvorba v něm snadná i pro ne příliš zdatného uživatele, pokud nevyžaduje pokročilejší funkce, které by se musel naučit. Též umožňuje tvorbu univerzálních skriptů a je provázán s jedním z nejmočnějších interpretovaných jazyků, jazykem Python.

Dalším zjednodušením pro cílového uživatele, které jsem zvolil, je napsání skriptu, po jehož spuštění se zobrazí uživatelské rozhraní (dále jen GUI), které umožňuje každý krok přípravy real-time obsahu jednoduše "naklikat".

Ještě předtím, než se pustíme do samotného procesu tvorby, je nutno si rozvrhnout, jak bude celý engine nakonec vypadat. Měl by být uživatelsky přívětivý a obsahovat všechny potřebné funkce. Mohou se v něm však vyskytovat i některé doplňkové funkce pro pokročilejšího uživatele. Ty ale nesmí působit rušivě pro uživatele neznalého, který je nebude využívat.

Měl jsem na výběr mezi různými způsoby implementace rozhraní engine. Prvním návrhem bylo jeho vytvoření s využitím jednoho z mnoha grafických toolkitů (lze jmenovat např. GTK+, Qt, FLTK a spoustu dalších). Tento návrh se však projevil jako absolutně nevyhovující, protože by zde byla nulová nebo velmi nízká možnost provázání s funkcemi Blenderu. Ty jsou však potřebné při manipulaci s modelem.

Další z možností bylo vytvořit skript přímo v Blenderu, který načte konfigurační soubory, v nichž se nachází nastavení scény, modelu, materiálů pro výpočet radiozity a vše potřebné, aplikuje je na importovaný model a výsledný produkt uloží do specifikovaného adresáře.

To by sice umožňovalo jednoduše spustit kompletní proces jedním krokem, což pro uživatele na jednu stranu znamená výhodu, že nemusí do tohoto procesu nijak zasahovat, ale na druhou stranu je to nevýhodou pro ty, kdo chtějí mít nad celým procesem kontrolu. Navíc uživatelé Windows nejsou příliš navyklí na používání konfiguračních souborů, tudíž by pro ně byla tato cesta konfigurace poněkud nezvyklá. Formát konfiguračních souborů také nebyl příliš jednoduchý, jednalo se o způsob zápisu datového typu slovník (dictionary) jazyka Python.

Proto z této fáze práce vznikl poslední návrh, podle něhož jsem vypracoval finální řešení. Vytvořit GUI pomocí téhož skriptu, do něhož by byla veškerá funkčnost naimplementována pod jednotlivé grafické prvky a uživatel by komunikoval s činností procesu tvorby aplikace pomocí grafického rozhraní. Tím sice odpadá výhoda provedení celé akce naráz, ale uživatel tím získává možnost neustálé kontroly činnosti programu.

Naštěstí Blender Python API obsahuje podporu pro tvorbu grafických ovládacích prvků, takže celá tvorba spočívala v naprogramování tohoto GUI s využitím jazyka Python a Blender Python API.

Nejprve je ale třeba vytvořit interaktivní část aplikace, která bude zajišťovat pohyb uživatele, reakce na jeho požadavky a zbývající interaktivní prvky. Až poté se budu věnovat tvorbě samotného GUI.

## 5.2 Implementace interaktivní části

Základem celé práce je vytvoření interaktivní části obsahu. Celá tato část je vytvořena pomocí Blender GameEngine. Na internetu lze nalézt spoustu šablon implementujících virtuálního avatara. Já jsem použil jednu přímo z domovských stránek Blenderu a upravil ji podle potřeb. Tuto šablonu jsem zvolil z toho důvodu, že je v ní naimplementován mód volného pohybu (fly mode), který může sloužit např. k rychlému přesunu mezi převýšenými místy nebo po vypadnutí z modelu, důsledkem jeho špatného vytvoření, k návratu zpět. Též je naimplementována relativně snadným způsobem.

Celá interaktivní šablona představuje objekt avatara. Jedná se o reprezentaci uživatele ve virtuálním prostoru. Tato postava je jeho dvojníkem, pomyslnou bytostí, která se pohybuje ve virtuálním světě a její chování představuje akce uživatele. Uživatel vidí na obrazovce to, co avatar ve virtuálním světě.

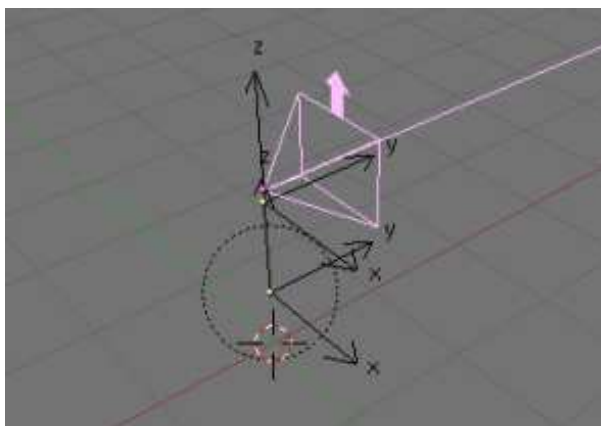
Avatar má určité rozměry, které mu brání v průchodu příliš úzkým či nízkým prostorem. V jednoulázevské aplikaci není avatara vidět, jeho obraz není důležitý. Jsou na něj též aplikovány základní fyzikální zákony, zejména při pohybu, kdy na něj působí zemská přitažlivost a jsou detekovány kolize s objekty po nichž se avatar pohybuje či do nich naráží. Opakem je mód létání, kdy gravitační konstanta neexistuje a na avatara tedy nepůsobí přitažlivost. Jeho pohyb je tedy prováděn nejčastěji ve směru pohledu.

Simulace fyzikálních zákonů – přitažlivost a hmotný charakter objektů – však rapidně zvyšuje výpočetní náročnost. Více informací lze nalézt v [3, str. 527-529].

### 5.2.1 Základ šablony avatara

Šablona avatara obsahuje objekt dotýkající se podlahy v odrazové obálce, nejlépe kulové, aby se mohl snadno pohybovat po rovině i do schodů. Dále objekt symbolizující rameno avatara, podél kterého lze pohled směřovat nahoru a dolů a nakonec objekt kamery symbolizující hlavu, potažmo oko avatara (obr. 5.1).

Všechny objekty jsou spojeny (parented) k sobě, aby se při pohybu spodní styčné části přesunuly i všechny ostatní. Dále jsou použity skripty obstarávající pohyb pomocí klávesnice a myši. Použitou šablonu bylo třeba upravit přímo pro potřeby této práce.



Obrázek 5.1: Šablona pro průchod

### 5.2.2 Úprava interaktivní šablony

Jednou z nutných úprav bylo přejmenování objektů tak, aby se shodná jména vyskytla v importovaných modelech s co nejmenší pravděpodobností. Dále bylo potřeba poupravit vlastnosti styčného objektu, jeho velikost, vektor tření, hmotný charakter a odrazivost od stěn. To lze provést pomocí tlačítek nastavení objektu (viz 4.1.1).

Dále jsem do scény přidal objekt `engProperties` typu `Empty` (aby neovlivňoval zobrazení). Pomocí něj je uskutečněna komunikace se skriptem a implementuje přidání funkce.

Největší změny jsem provedl v ovládacích skriptech. Ty byly upraveny pro pohyb pomocí kurzorových kláves a byl do nich přidán mód rychlejšího pohybu pomocí kláves `Shift`. Další funkce, které byly implementovány navíc jsou uvedeny v následujícím výčtu.

- **Úprava způsobu navigace pomocí myši**

Původní způsob navigace nebyl příliš intuitivní, protože se pro natočení pohledu muselo stisknout levé tlačítko myši a po ukončení natáčení ho uvolnit. Upravil jsem tedy způsob rotace tak, aby se co nejvíce přibližoval počítačovému hram. Pohled se nyní natáčí automaticky podle pohybu myši a není potřeba mít stisknuto žádné tlačítko.

- **Uložení snímku obrazovky**

Klávesou `S` lze během real-time režimu uložit snímek aktuálního obrazu do adresáře, odkud je aplikace spuštěna. Snímky jsou pojmenovány jako `screenshotXXXX.png`, kde `XXXX` je pořadové číslo snímku doplněné zleva nulami. Potíže při vytváření snímku jsou popsány v části 6.4.

- **Snížení zátěže procesoru**

Nastavením parametru `Sleep time` ve Walkthrough Engine (viz sekce 5.4.5) lze snížit zátěž procesoru během real-time režimu pomocí čekacích cyklů. Tato volba ale současně sníží počet snímků za vteřinu a tím plynulost celé prezentace.

- **Informativní texty**

Po provedení některých akcí se na obrazovku vypíše informativní nápis. Je možno je použít i pro vypisování textu příslušejícímu místnosti, ve které se avatar právě nachází. Text místnosti lze specifikovat pomocí GUI.

- **Dynamická změna eye-separation**

Přehráváme-li s podporou stereoskopického zobrazení, je někdy potřeba upravit rozteč oddělených obrazů. Proto jsem přidal podporu změny této rozteče během přehrávání. Další informace nalezneme v sekci 6.5.1.

- **Ukončení pomocí ESC**

Přidal jsem i možnost okamžitého ukončení přehrávání obsahu pomocí stisku klávesy ESC.

### 5.2.3 Možnost použití jiné šablony

Není příliš obtížné vytvořit vlastní real-time šablonu, která by implementovala některé nestandardní, ale potřebné funkce, popř. ty nepoužívané by opomíjela. Také na internetu lze takových šablon nalézt nepřehledné množství. Proto je do projektu zahrnuta možnost použít jinou real-time šablonu, která by uživateli více vyhovovala. Ta musí splňovat následující podmínky, aby nebyl chod skriptu přerušován různými chybami.

- **Jméno styčného objektu**

Při použití nové šablony je třeba skript seznámit se jmény důležitých objektů. Na začátku skriptu se v oddělené sekci nalézají specifikace těchto jmen. Šablona musí obsahovat minimálně jeden objekt, který bude představovat styčnou plochu s podlahou. Ten by se měl buď jmenovat `engViewer`, nebo musí být jeho jméno přiřazeno proměnné `avatarName`.

- **Proměnné pro bezchybný chod skriptu**

Engine využívá různých stavových proměnných, které ovlivňují jeho správný chod. Všechny tyto proměnné jsou obsaženy v objektu pojmenovaném `engProperties`. Je doporučeno tento objekt ponechat ve scéně i při použití jiné šablony avatara. Implementuje totiž některé doplňkové funkce. Ty by jinak musely být dodatečně doimplementovány.

Pokud se i přesto rozhodneme tento objekt odstranit, musí se všechny potřebné proměnné nacházet i v nové šabloně. Mohou být přiřazeny buď samostatnému objektu (nejlépe typu *Empty*), nebo mohou být součástí některého z objektů obsažených v šabloně. Všechny však musí patřit stejnému objektu. Jedná se o následující proměnné (jména i hodnoty jsou case sensitive):

- `mlx1Name` – typ `String`, výchozí hodnota `"=None1x1="`.
- `resizeRatio` – typ `Float`, výchozí hodnota `0.0`.
- `resizeSmaller` – typ `Int`, výchozí hodnota `1`.
- `modelFile` – typ `String`, výchozí hodnota `" "`.
- `MODEL_IMPORTED` – typ `Int`, výchozí hodnota `0`.
- `OBJ_1X1_SELECTED` – typ `Int`, výchozí hodnota `0`.
- `MESHES_COLLECTED` – typ `Int`, výchozí hodnota `0`.
- `RAD_CALCULATED` – typ `Int`, výchozí hodnota `0`.

Jméno tohoto objektu nakonec musíme přiřadit do proměnné `propertiesName`, popř. ho stačí přejmenovat na původní `engProperties`.

- **Proměnné styčného objektu**

Objekt zmíněný v prvním bodě musí též obsahovat některé potřebné proměnné. Šablona je nemusí nijak využívat, ale je nutná jejich existence ve správném objektu.



- `speed` – typ `Float`, výchozí hodnota `3.5`.
- `acceleration` – typ `Float`, výchozí hodnota `1.8`.
- `sleeptime` – typ `Float`, výchozí hodnota `0.0`.

Ještě je třeba, aby obsahoval proměnnou typu `Bool` a hodnotou `True`, jejíž jméno bude sloužit jako identifikátor avatara. Ten je potřebný při tvorbě zprůhledňování dveří, viz část 6.5.3.

#### • Použití informativních textů

Pokud jsou i v nové šabloně použity informativní texty, musí v ní být obsaženy textury fontů a ty správně namapovány na zobrazovací objekty. Konfigurační proměnnou `useFonts` je třeba nastavit na `1` a hlavní objekt musí obsahovat následující proměnnou:

- `fontname` – typ `String`, výchozí hodnota jméno požadovaného fontu.

Do seznamu `strFonts` v konfigurační sekci skriptu musí být též vypsána jména používaných fontů. První položka ale musí zůstat prázdná.

Je doporučeno použít fonty a objekty, na které se vypisují texty, z původní šablony nebo alespoň nastavit proměnnou `showFonts` na `0`. Jinak by bylo nutné vytvořit nové ukázky fontů. To však v této zprávě není popsáno.

Pokud texty používány nejsou, nastavíme konfigurační proměnné `useFonts` a `showFonts` na `0`. Jinak bude skript generovat chyby.

#### • Vyplnění seznamu objektů a textur

Dále je potřeba vyplnit seznam všech objektů obsažených v šabloně kvůli možnosti zjištění těch naimportovaných. To je provedeno vyplněním pole `initObjs` v konfigurační sekci.

Je též doporučeno umístit objekty související s informativními texty do druhé vrstvy pro jejich pozdější snadné zneviditelnění, bude-li to třeba.

#### • Povolení obrázků v GUI

Přejeme-li si zobrazit logo v GUI, případně ukázky fontů, musíme nastavit proměnnou `useImages` na `1`. Tyto obrázky obsadí ohromné množství paměti RAM a není doporučeno jejich zobrazení na počítačích s jejím malým množstvím.

Pro správné vykreslení obrázků též musíme dodávat spolu s Blender Walkthrough Engine adresář `engine`. V něm jsou v souboru `images.py` uloženy všechny potřebné funkce pro práci s obrázky a samotné obrázky převedené do formátu souřadnic.

Pro zobrazení loga je v konfigurační sekci volba `showLogo`, zobrazení ukázek fontů můžeme ovlivnit pomocí proměnné `showFonts`.

#### • Zbývající nastavení

Dále je nutno vyplnit další dvě pole – pole obrázků obsažených v šabloně (fonty, textury. . .) a materiálů. Jedná se o pole `initImgs` a `initMats`. Pokud nebudou tyto informace vyplněny správně, nelze od skriptu očekávat korektní chování.

#### • Názvy IPO křivek

Protože je ve většině šablon naimplementováno otáčení a naklánění avatara pomocí IPO křivek, je nutné vložit do konfigurační proměnné `avatarIpo` jméno IPO křivky styčného objektu, pomocí níž lze avatarem natáčet do stran, popř. tuto křivku pojmenovat jako `ipoViewer`.

V konfigurační části se též nachází jméno IPO křivky specifikující průhlednost dveří (proměnná `alphaIpo`). Více o jejím využití se dočteme v části 6.5.3.

Poslední konfigurační proměnná `recalcNormals` určuje, jestli se po naimportování modelu má Blender pokusit přepočítat normály objektů směrem mříčím ven. Nicméně, pokud jsou normálové vektory umístěny špatným směrem již v importovaném modelu, nelze od této volby očekávat 100% výsledky.

### 5.3 Podmínky a doporučení pro importovaný model

Vzhledem k tomu, že bude model importován pomocí skriptu a budou nastavovány jeho vlastnosti, navrhnul jsem podmínky a doporučení, která by měl model splňovat. Tyto podmínky však nejsou dogmatem. Při jejich nedodržení ovšem nelze očekávat správnou funkčnost engine nebo výsledného real-time obsahu.

- **Krychle o rozměrech 1x1**

V modelu by se měla nacházet krychle o velikosti 1x1, tak aby tato velikost odpovídala reálným rozměrům 1x1 metr. Podle této krychle bude také změněna velikost importovaného modelu a může být na její pozici snadno posunut interaktivní avatar. Po importu modelu tato krychle nemusí působit rušivě, lze totiž pomocí engine zneviditelnit popř. ji lze zachovat, pokud má být nedílnou součástí modelu.

Jesliže se tato krychle v modelu nenachází, engine bude funkční, pouze musí uživatel změnit velikost modelu a počáteční pozici avatara ručně.

- **Výška schodů**

Avatar se dokáže vypořádat s pohybem do a ze schodů. Ty by ale měly dosahovat standardní výšky 20-30 cm (poměrově k jednotkové krychli). Do vyšších schodů se již avatar nemusí dostat, nižší mu samozřejmě nečiní žádné potíže.

- **Šířka dveří**

Pokud se v modelu vyskytují dveře nebo jiná propojovací místa, jejich šířka by měla dosahovat minimálně 0,75 – 1 metru. Jinak totiž hrozí, že jimi avatar nebude moci projít. Také by tyto dveře měly být reprezentovány samostatným objektem.

- **Formát výsledného modelu**

Importovaný model musí být uložen v některém z následujících formátů podporovaných engine. Je zde též popsáno, která data lze v daném formátu naimportovat.

- *.3ds* – formát souborů programu 3D Studio MAX. Spolehlivě importuje objekty, jména objektů, UV mapování textur a materiály. Též je zde podpora pro obarvené vertexy (vertex colours). Tento formát je používán jako primární a byly na něm testovány všechny funkce.
- *.obj* – soubor programu Wavefront Maya. Importuje správně objekty, materiály a mírně upravená jména objektů. Problém mu nečiní ani textury a UV mapování, stejně jako obarvené vertexy.

Dále bych rád zmínil určitá doporučení, která by měl model dodržet pro snadnější a příjemnější import a následné zpracování importovaného modelu. Dodržení těchto doporučení později usnadní práci s modelem a zvýší možnosti nastavení.

- **Vizuální stránka**

Je lepší, pokud je model správně otexturovaný, protože s ním lze pomocí engine provádět více operací. Textury musí být dodány současně s modelem a je třeba dát pozor na velikost písmen v názvech souborů s texturami, obzvláště na systémech Linux.

Model může sice mít pouze přiřazené materiály, ze kterých lze v engine spočítat radiozitu, ale toto řešení má spoustu omezení (viz část 6.4 na str. 37). Výsledné modely doporučuji importovat otexturované, popř. radiozitu používat na jednodušší modely, u nichž není potřeba nic nastavovat.

- **Struktura modelu**

Toto doporučení se týká převážně otexturovaných modelů, protože při výpočtu radiozity jsou všechny zahrnuté objekty nahrazeny jedním výsledným.

Model by měl být členěn pomocí objektů na jednotlivé místnosti, dveře a ostatní objekty. Pro každý objekt lze totiž pomocí engine specifikovat jeho typ – dveře, místnost nebo skleněný objekt – čehož je poté využíváno během prezentace výsledků. Pokud je model dobře členěný, jeho nastavení tak bude mnohem jednodušší a navíc tím uživatel získá větší možnosti.

- **Cesty k texturám**

Jestliže model obsahuje textury, měly by se nacházet nejlépe ve stejném adresáři nebo maximálně v podadresáři první úrovně. Jejich cesta by též měla být specifikována pomocí relativního adresování. Je to kvůli ukládání výsledného projektu. Jinak by totiž engine nemusel textury najít nebo uložit na správnou pozici.

## 5.4 Tvorba engine

Během tvorby grafického rozhraní jsem rozdělil funkce Blender Walkthrough Engine do 4 skupin podle jejich pole působnosti a předpokládaného pořadí použití. V první skupině jsou funkce pro práci s externím souborem modelu, jeho naimportování a nastavení scény pomocí zvoleného kalibračního objektu. Dále se zde nachází funkce pro nastavení počáteční pozice a rotace avatara.

Do druhé skupiny spadají funkce pro nastavení vlastností jednotlivých naimportovaných objektů. Nalézá se zde specifikace typu objektu a jeho vizuálních parametrů.

Třetí skupina slouží k výpočtu radiozity. Vzhledem ke komplexnosti celého procesu je tato skupina nejpočetnější a obsahuje nejvíce ovládacích prvků.

Do poslední skupiny jsem zařadil funkce sloužící k nastavení vlastností real-time sezení a uložení celého projektu a všech potřebných souborů do oddělené složky.

V celém skriptu jsou bohatě využívány funkce Blender Python API [6] a je kompletně napsán v jazyce Python. V následujících sekcích bych rád popsal jednotlivé skupiny funkcí rozmístěné do příslušných záložek uživatelského rozhraní skriptu. Zmíním též některé implementační detaily uživatelských prvků. Tento popis neslouží jako uživatelský manuál pro práci s grafickým rozhraním engine, ten je k nalezení v přílohách na straně 45.

### 5.4.1 Spuštění skriptu

Po spuštění Blender Walkthrough Engine skriptu je ještě před vykreslením samotného GUI provedeno několik akcí.

- Pokud je v GUI povoleno použití obrázků, tak se ze všeho nejdříve načtou do paměti všechny povolené obrázky – logo, případně ukázky použitých fontů.

- Poté se naplní slovník všech objektů ve scéně, které nejsou přímou součástí real-time šablony – nebyly specifikovány v konfigurační části, viz část 5.2.3. Tento slovník je dále používán prvky pro práci s objekty a je při každé změně aktualizován.
- Dále skript zjistí, jestli již byl proveden import modelu. Pokud ano, nastaví ovládací prvky do takové polohy, ve které se nacházely při ukončení předchozího sezení. To se týká všech nastavení objektů, polohy atd. Neobnovuje se pouze seznam objektů, na něž měla být aplikována radiozita. Místo toho je vytvořen nový seznam z vyselektovaných objektů ve scéně.
- Následujícím krokem je nastavení rotace avatara podle jeho IPO křivky.
- Na závěr se spustí samotné grafické rozhraní skládající se ze čtyř záložek zastupujících jednotlivé tematické skupiny funkcí.

Ke každé záložce náleží samostatná logovací oblast, do níž jsou vypisovány informační a chybové hlášky skriptu. Jedná se o černou oblast ve spodní části GUI. Je navržena tak, aby se zobrazovaly vždy nejnovější zprávy. Pokud je v logu větší počet zpráv, než je oblast schopna zobrazit, lze mezi nimi rolovat pomocí kláves se šipkou nahoru a dolů.

Tyto logy lze též promazávat pomocí tlačítek úplně dole. Stiskem tlačítka **Current** jsou vymazány všechny logovací zprávy na aktivní záložce, zatímco tlačítkem **All** lze odstranit logovací zprávy ze všech záložek. Pomocí **Exit** lze ukončit činnost skriptu.

Spousta ovládacích prvků skriptu supluje standardní tlačítka Blenderu, která by ale musel jinak uživatel hledat v různých modulech. Zde tvoří uspořádaný celek, navíc jsou zde zastoupena jen ta nejnutnější a k některým jsou doimplementovány další funkce.

### 5.4.2 Import modelu a nastavení kalibračního objektu

První záložka (obr. 5.2) obsahuje ovládací prvky sloužící k naimportování modelu do scény a specifikování jednotkového objektu v něm. Dále zde lze měnit velikost modelu oběma směry, přemísťovat avatara a nátačet jeho pohled v rozmezí  $-180^\circ$  až  $180^\circ$ .

- **Browse**

Vyvolá standardní dialog Blenderu pro otevření souboru. Toho je dosaženo pomocí volání fce Python API

```
Window.FileSelector(loadModelFile, 'Select model file...')
```

`loadModelFile` je zde callback funkce zajišťující akci se zvoleným souborem.

- **Import model**

Vytvoří instanci třídy `Importer` naimplementované ve stejném skriptu, které předá jako parametr název vybraného souboru. Ta podle přípony rozezná typ souboru a pokud je podporován, spustí správnou importovací funkci. Je nutné dodávat spolu s tímto skriptem i adresář `import`, v němž se nacházejí potřebné importovací skripty.

Proběhne-li import modelu v pořádku, tlačítko se stiskne a zaktivují se i všechna ostatní.

- **Select imported 1x1 model object:**

V tomto seznamu je nutno zvolit kalibrační objekt. Podle něj se poté aplikují všechna další nastavení.



Obrázek 5.2: Záložka 1: Import modelu a kalibrační objekt

- **Invisible 1x1**

Podle stavu tlačítka zapíná neviditelnost a kolize u zvoleného kalibračního objektu. Toho je dosaženo ovládáním příznaků `Blender.NMesh.FaceModes[ 'INVISIBLE' ]` a `Blender.NMesh.FaceModes[ 'DYNAMIC' ]` kalibračního objektu.

- **Resize model**

Zobrazí relevantní část GUI pro úpravu velikosti modelu. Patří do ní následující ovládací prvky (obr. 5.2 vpravo):

- **Ratio:** – nový poměr velikosti původního naimportovaného modelu a modelu s upravenou velikostí. Změny nejsou ihned aplikovány.
- **Auto** – automaticky změní velikost modelu v poměru ku zvolenému kalibračnímu objektu. Střídavě mění velikost podle kalibračního objektu a vrací ji zpět na původní.
- **Small** – pokud je tlačítko stisknuto, velikost importovaného modelu se tímto poměrem zmenší, jinak je zvětšena.
- **Apply new** – aplikuje nové nastavení velikosti. Volá funkci

```
resizeScene(sldRatio.val)
```

Ta provádí následující akci – není-li zvolen kalibrační objekt, ukončí se. Jinak zjistí souřadnice `BoundingBox` tohoto objektu. Není-li funkci předán parametr, automaticky změní velikost modelu podle aktuálního poměru ke kalibračnímu objektu. Je-li parametr zadán, aplikuje se nové nastavení velikosti. Model se změní zpět na původní velikost a poté je na něj aplikován nový poměr velikosti.

- **Move avatar**

Zviditelňuje část GUI sloužící k přesunu avatara na novou počáteční pozici. Ovládací prvky dostupné v této části:

- **X, Y, Z**

Změna pozice na příslušné souřadnici. Bohužel, kliknutí na okraj číselných tlačítek není funkční, je nutné stisknout levé tlačítko myši na požadovaném okraji a tažením měnit pozici. Přesnější hodnoty můžeme zadat kliknutím na číselnou hodnotu uvnitř tlačítka a napsáním na klávesnici. Změny pozice je dosaženo voláním následující funkce:

```
engViewer.setLocation(numX.val, numY.val, numZ.val)
```

- **Position of 1x1**

Přesune objekt avatara na pozici kalibračního objektu. Zjistí `BorderBox` tohoto objektu, spočítá jeho stereometrický střed a přesune do něj střed styčného objektu avatara.

- **Original position**

Vrátí avatara na původní pozici. Volá API funkci

```
engViewer.setLocation(0.0, 0.0, 0.5)
```

- **Resize avatar**

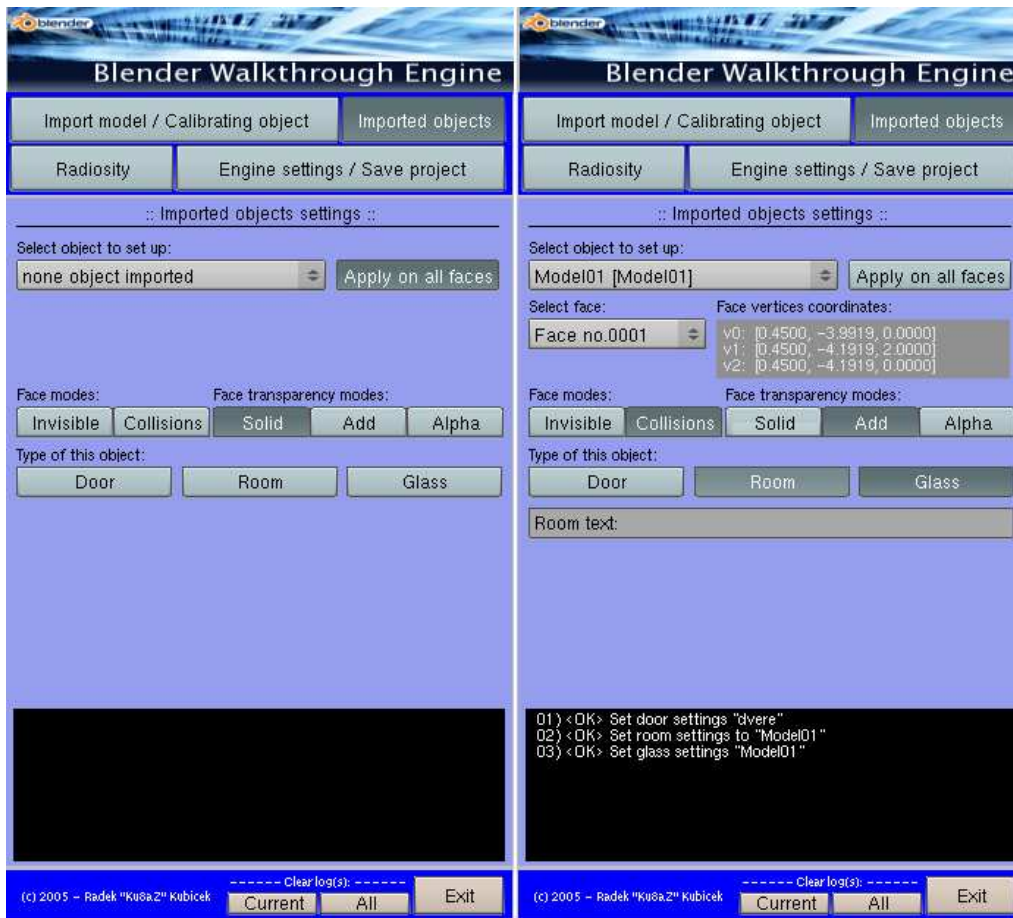
Zobrazuje příslušnou část GUI k otočení avatara. Nachází se zde ovládací prvek **Rotation angle**. Tím lze natočit avatara v rozsahu  $-180^\circ$  až  $180^\circ$ . Změna je aplikována okamžitě a je jí dosaženo změnou polohy bodů IPO křivky pomocí následujícího kódu:

```
ipo = Blender.Ipo.Get(avatarIpo) # (1)
rot = ipo.getCurve('RotZ') # (2)
pts = rot.getPoints() # (3)
pts[0].setPoints([pts[0].pt[0], sldRotate.val/10]) # (4)
pts[1].setPoints([pts[1].pt[0], (sldRotate.val/10)+18.0]) # (5)
rot.Recalc() # (6)
engViewer.RotZ = (sldRotate.val*PI)/180 # (7)
engViewer.getData().update() # (8)
```

Krokem (1) nalezneme objekt IPO křivky. V kroku (2) získáme z IPO objektu křivku pro rotaci kolem osy Z a krokem (3) její body (pro rotační křivky jsou souřadnice na ose Y násobky  $10^\circ$ ). Příkazy (4) a (5) zajistí přesun bodů křivky na požadované hodnoty, kdy se do spodního bodu uloží požadovaná počáteční hodnota a do druhého bodu hodnota o  $180^\circ$  vyšší. V (6) je přepočítána křivka a pomocí příkazu (7) se provede natočení avatara ve 3D okně. Poslední příkaz (8) aktualizuje změny v objektu avatara.

### 5.4.3 Konfigurace naimportovaných objektů

Součástí druhé záložky (obr. 5.3) jsou prvky, jejichž funkce se vztahuje k nastavení objektů ve scéně. Lze zde zprůhledňovat objekty, zapínat a vypínat jejich kolize a nastavovat typy objektů.



Obrázek 5.3: Záložka 2: Nastavení objektů

- **Select object to set up:**

Rozbalovací seznam sloužící k vybrání objektu, jehož nastavení hodláme měnit. Objekt je vyselektován ve scéně a ostatní ovládací prvky změny své hodnoty podle jeho aktuálního stavu.

- **Apply on all faces:**

Pokud je toto tlačítko aktivní, změny jsou aplikovány na celý objekt. Po jeho zneaktivnění (obr. 5.3 vpravo) se objeví další rozbalovací seznam **Select face:**, s jehož pomocí lze zvolit face, na který chceme změny aplikovat. Jeho transformované souřadnice se zobrazí v šedém poli napravo. Možným vylepšením výběru face by mohlo být jeho vyselektování ve scéně, souřadnice totiž nejsou příliš intuitivní, ale nenalezl jsem API funkci, která by toto umožňovala. Vyselektování zvolené face a odselektování všech ostatních je provedeno následujícími příkazy.

```
SEL = Blender.NMesh.FaceFlags[ 'SELECT' ]
for f in mesh.faces:
    f.flag &= ~SEL
mesh.faces[mnuFaces.val-1].flag |= SEL
```

- **Invisible**

Ovládá příznak neviditelnosti objektu. Toho je dosaženo API voláním

```
f.mode |= Blender.NMesh.FaceModes[ 'INVISIBLE' ]
```

pro zneviditelnění, případně

```
f.mode &= ~Blender.NMesh.FaceModes[ 'INVISIBLE' ]
```

pro opětovné zviditelnění. Objekt se ale stále zúčastňuje výpočtu kolizí.

- **Collisions**

Ovládá příznak detekce kolizí zvoleného objektu. Při neaktivním tlačítku kolize vypíná pomocí

```
f.mode &= ~Blender.NMesh.FaceModes[ 'DYNAMIC' ]
```

Pokud je tlačítko stisknuto, vytvoří se kolem objektu kolizní obálka a je nastaven příznak detekce kolizí.

```
f.mode |= Blender.NMesh.FaceModes[ 'DYNAMIC' ]
```

Následující tři prvky slouží k přepínání způsobu zobrazení vybraného objektu.

- **Solid**

Pevné zobrazení. Vše je viditelné tak, jak je vytvořeno a namapováno texturou, popř. vytvořeno výpočtem radiozity.

- **Add**

Vytváří průhlednost podle textury. Černá barva je maximálně průhledná, bílá neprůhledná. Používá se při nastavení skleněných objektů. Podle jejich namapované textury je vytvořen průhledný objekt.

- **Alpha**

Vytváří průhlednost pomocí barevné složky Alpha, kterou lze měnit pomocí IPO křivek. Tato volba slouží pro plynulé zneviditelnění dveří, pokud se k nim avatar přiblížil a opětovné zviditelnění po oddálení. Více v části 6.5.3.

Pomocí spodního bloku tlačítek lze nastavit objekt na některý z možných předdefinovaných typů. Toto nastavení je vždy aplikováno na celý objekt.

- **Door**

Zvolený objekt nastaví jako dveře – vypne u něj kolize. Pokud bychom chtěli naimplementovat zprůhledňování dveří, dozvíme se postup v části 6.5.3. Chceme-li objekt dveří pouze zprůhlednit, stačí přepnout typ zobrazení ze **Solid** na **Add**.

- **Room**

Nastaví typ objektu na místnost. Pokud jsou používány informativní nápisy, po aktivaci této volby se zobrazí další ovládací prvek **Room text:** (obr. 5.3 vpravo). Do tohoto textového pole můžeme vepsat text, který je poté zobrazen během real-time režimu, během avatarovy přítomnosti v této místnosti.

- **Glass**

Objekt zobrazí jako skleněný – aplikuje na něj zprůhlednění na základě textury. Detekci kolizí ponechává aktivní. Jen v případě, že by byl stejný objekt identifikován též jako dveře, je detekce kolizí deaktivována.





Obrázek 5.4: Záložka 3: Výpočet a aplikace radiozity

### 5.4.4 Výpočet a aplikace radiozity

Na třetí záložce (obr. 5.4) můžeme nalézt prvky pro práci s výpočtem a aplikací radiozity. Lze s jejich pomocí specifikovat objekty aktivně se podílející na výpočtu, nastavit parametry radiozity a spustit samotný výpočet. Nakonec umožňují výsledek radiozity aplikovat na zvolené objekty.

Radiozita je metoda pro dosažení fotorealistické věrnosti scény. Dovolí z fyzikálního hlediska korektně umožňovat šíření světelných paprsků scénou. Základní radiozitní algoritmus vychází ze zákona zachování energie a předpokládá, že přenos světelného záření mezi objekty probíhá v energeticky uzavřené scéně. Objekty jsou popsány ploškovou hraniční reprezentací.

Postup při zobrazování scény metodou radiozity lze rozdělit na dvě části – nejprve je vyhodnoceno šíření světla z plošných světelných zdrojů (v programu Blender reprezentovaných jako materiály objektů vyzařující a přijímající intenzitu světla) a jeho odrazy na povrchu těles. Výsledkem tohoto výpočtu je ohodnocení ploch hodnotami osvětlení, které vyjadřují množství difúzně odraženého světla pro každou plošku. Tyto hodnoty nezávisí na poloze pozorovatele, jsou pouze vlastností scény. Více o radiozitě je popsáno v [3, část 15.10].

Algoritmy a principy používané aplikací Blender jsou popsány v [1, část Radiosity].

- **Select object for radiosity settings:**

Rolovací seznam všech objektů, které se mohou podílet na procesu výpočtu.

- **Add all**

Přidá všechny objekty kromě kalibračního do seznamu objektů podílejících se na výpočtu radiozity. Kalibrační objekt není zahrnut z důvodu jeho pozdějšího zviditelnění a zapnutí kolizí po aplikaci výsledků radiozity.

- **Rem. all**

Vyprázdní seznam objektů podílejících se na výpočtu. Toho je dosaženo jednoduchým příkazem

```
radList = {}
```

- **Add to rad. calc.**

Pokud je tlačítko stisknuto, je zvolený objekt zahrnut do výpočtu, jinak je z něj odebrán.

- **Select material for radiosity settings:**

Seznam všech materiálů ve scéně aplikovaných na některý z objektů. Pro každý materiál lze nastavit jeho vlastnosti potřebné pro výpočet radiozity. Po výběru materiálu jsou automaticky zaktualizovány hodnoty **Emit** a **Ambient**.

- **Emit**

Vyzařování materiálu. Určuje hodnotu, jak bude příslušný materiál ovlivňovat výpočet vyzařováním světla.

- **Ambient**

Množství přijímaného difuzního světla. Čím vyšší hodnota, tím více bude materiál ovlivněn ostatními.

- **Objects for radiosity calc:**

Výpis všech objektů zvolených pro výpočet radiozity. Pokud jejich počet přesáhne velikost výpisového rámce, lze s ním rolovat pomocí kolečka myši. Výpis je implementován pomocí následujícího kódu:

```
keys = radList.keys()
for i in range(radCount):
    if (i+radPos) < len(radList):
        glRasterPos2d(205, posTop-90-(i*11))
        Text("%s" % radList[keys[i+radPos]]['m'], 'small')
    else:
        break
```

Ten zajistí, že je vypsáno pouze radCount položek se začátkem v radPos. Zbytek je přeskočen. radPos se automaticky mění během scrollování seznamu.

Ve spodní části této záložky se nachází dvě sekce. V jedné se nastavují konfigurační faktory výpočtu, ve druhé se spouští samotný výpočet a aplikují výsledky. Všechny ovládací prvky z těchto sekcí se nacházejí i v podpanelu **Radiosity** modulu **Buttons Window**. Nejprve si popíšeme první sekci – **Radiosity render** (obr. 5.4 vlevo).

- **Hemires**

Velikost "polokrychle" umístěné uprostřed záplaty. Na každé straně obsahuje miniaturní obraz okolního prostředí. V každém pixelu těchto obrazů je barevně zakódována jistá ploška, pro kterou je spočítáno množství energie. Jedná se o zjednodušení a optimalizaci radiozitivní rovnice. Nastavení správného rozlišení těchto polokrychlí je důležité pro odstranění aliasových artefaktů.

- **Max. iterations**

Maximální počet iterací průběhu kalkulace. Po dosažení konečného výsledků v nižším počtu kroků se toto nastavení neaplikuje, jinak je po překročení specifikovaného maximálního počtu iterací výpočet okamžitě ukončen.

- **Mult**

Násobitel energie vyzařované energetickými ploškami.

- **Gamma**

Nastavení kontrastu energetických plošek.

Druhá sekce, nacházející se pod tlačítkem **Radiosity tool** (obr. 5.4 vpravo) slouží ke sloučení všech objektů zahrnutých do výpočtu, spuštění výpočtu a následné aplikaci výsledku.

- **Collect meshes**

”Spojí” všechny zúčastněné objekty do jednoho a ten rozdělí na plošky (záplaty – patches), pro něž bude počítána radiozita a budou vyzařovat, popř. přijímat difuzní světlo.

- **Start calculating**

Spustí proces kalkulace radiozity. Aplikuje nastavení specifikovaná v sekci **Radiosity render** a začne počítat radiozitu. Volá API funkci

```
Blender.Scene.GetCurrent().getRadiosityContext().go()
```

- **Replace meshes**

Až je radiozita dopočítána, musí se aplikovat výsledek na stávající objekty. Toho je dosaženo pomocí tohoto tlačítka, které volá funkci

```
Blender.Scene.GetCurrent().getRadiosityContext().replaceMeshes()
```

Všechny objekty zúčastněné ve výpočtu jsou odstraněny a nahrazeny jediným, výsledným objektem s aplikovaným ohodnocením jednotlivých plošek množstvím difuzního světla.

- **Free radio data**

Po dokončení nebo pro zrušení procesu výpočtu radiozity je třeba provést uvolnění kontextu radiozity.

```
Blender.Scene.GetCurrent().getRadiosityContext().freeData()
```

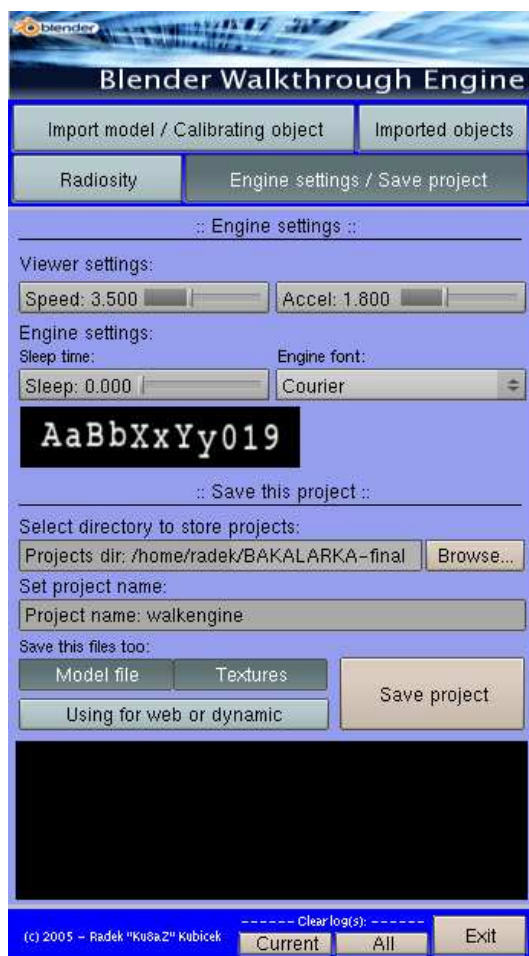
Jestliže bylo provedeno **Collect meshes**, je zrušeno rozdělení objektů na plošky a můžeme změnit nastavení výpočtu a zúčastněné objekty. Byla-li již dokončena kalkulace radiozity, je uvolněna paměť obsazená tímto výpočtem a odstraněn radiozitní kontext aktivní scény.

### 5.4.5 Nastavení real-time režimu a uložení projektu

Na poslední záložce (obr. 5.5) se nalézají ovládací prvky, jejichž funkce se týká nastavení real-time režimu a uložení kompletního výsledného projektu se všemi potřebnými soubory do zadaného adresáře.

- **Speed**

Tento posuvník slouží k zadání rychlosti pohybu avatara. Ta je uložena jako proměnná styčného objektu a je jí využíváno ve skriptu obsluhujícím pohyb během real-time režimu.



Obrázek 5.5: Záložka 4: Real-time nastavení a uložení projektu

- **Accel**

Hodnota zrychlení avatara při stlačené klávese **Shift** během pohybu. Jedná se o násobek rychlosti. Stejně jako **Speed** je uložena jako proměnná styčného objektu a posuvník mění pouze její hodnotu.

- **Sleep**

Doba v milisekundách, po kterou bude procesor nečinný mezi jednotlivými snímky. Sníží se tak vytížení procesoru, ale na druhou stranu klesne framerate a tím i plynulost přehrávání. Pro tuto funkci je vytvořen samostatný skript `sleep.py`, který je volán v real-time režimu. Obsahuje následující kód:

```
import time, GameLogic
sleep_time=GameLogic.getCurrentController().getOwner().sleep_time
if sleep_time > 0.0:
    time.sleep(sleep_time)
```

Ten zajistí, že jsou čekací cykly použity pouze, pokud je proměnná `sleep_time` nastavena na kladnou hodnotu.

- **Select directory to store projects:**

Cesta k adresáři pro uložení projektu. Tlačítkem **Browse** lze nalézt adresář vybráním některého ze souborů, které se v něm nachází. Callback funkce automaticky osekne název souboru a ponechá pouze adresářovou cestu. Samozřejmě lze vepsat požadovanou cestu i ručně přímo do textového pole.

- **Set project name:**

Jméno projektu. Ve výše zvoleném adresáři se vytvoří podadresář pojmenovaný podle jména projektu a do něj budou uloženy veškeré požadované soubory.

- **Model file**

Určuje, jestli do adresáře s projektem bude zkopírován i externí soubor s importovaným modelem.

- **Textures**

Je-li aktivní, do adresáře s projektem budou nakopírovány též veškeré potřebné textury použité v modelu.

- **Using for web or dynamic**

Přepíná viditelnost vrstev. Je-li aktivní, pouze první vrstva je viditelná. Jinak jsou viditelné dvě vrstvy. Pokud je projekt prezentován pomocí webového modulu (část 6.1) nebo pomocí BlenderPlayeru (část 6.2 a 6.3) a současně obsahuje radiozitu, nejsou ve stabilní verzi Blenderu funkční informativní texty. Ty se nacházejí ve vrstvě 2, stačí tedy pomocí příkazu

```
Blender.Window.ViewLayer([1])
```

tuto vrstvu zneviditelnit a vše je poté prezentováno správně.

- **Save project**

Vytvoří cílovou adresářovou cestu a nakopíruje do adresáře s projektem všechny potřebné a požadované soubory. Nakonec uloží i aktuální .blend soubor pomocí API volání

```
Blender.Save(Blender.sys.join(path, projname + ".blend"), 1)
```

Pokud je v konfigurační sekci povoleno použití fontů, zobrazí se na této záložce i ovládací prvek **Engine font**, pomocí něhož je možné nastavit font použitý pro informativní nápisy. Jsou-li povoleny i obrázky náhledů fontů, zobrazí se tímto prvkem též náhled použitého fontu.

## Kapitola 6

# Možnosti prezentace vytvořených dat

Výstupem Blenderu nemusí být pouze vyrenderovaný obrázek nebo animace. Nás zajímá spíše forma prezentace vytvořeného real-time obsahu. Existuje možnost exportovat real-time obsah do samostatně spustitelné aplikace. Vždy je též možné všechna vytvořená, importovaná nebo jinou cestou vložená data, nacházející se v aktuálním sezení, uložit do *.blend* souboru. Tento datový soubor obsahuje vše potřebné a je nativním formátem pro výměnu dat aplikace Blender. Navíc neslouží pouze pro uchovávání dat, ale velice snadno z něj můžeme spustit připravenou real-time scénu pomocí přehrávače **BlenderPlayer**. Ten je standardně dodáván jako součást balíku Blender.

Tohoto způsobu přehrávání real-time obsahu je využíváno i v pomocném plug-inu (modulu) do webových prohlížečů, díky němuž snadno zobrazíme real-time obsah nacházející se v *.blend* souboru přímo v okně webového prohlížeče. Je třeba si však uvědomit, že tato možnost prezentace není bez omezení. Právě naopak, jsme při ní omezeni více než při ostatních způsobech. Viz část 6.1.1.

Způsob spuštění pomocí přehrávače BlenderPlayer je častěji používaný. Nemá taková omezení, lze nastavovat různé parametry, např. velikost okna, použití zvuků a dokonce typ stereoskopického zobrazení.

Při exportování obsahu do spustitelné aplikace jsou též určitá omezení (stejná jako při využití přehrávače), o kterých se zmíním v sekci 6.4. Tato forma prezentace je nejlépe využitelná pro ty, kdo nemají ve svém počítači program Blender.

### 6.1 Zobrazení pomocí webového prohlížeče

Real-time obsah lze za pomoci speciálního modulu (Blender Web Plug-in) zobrazit přímo v okně internetového prohlížeče. Tento modul, který je volně ke stažení, bohužel není od doby, co se Blender stal open-source projektem, nadále vyvíjen. A protože vývoj GameEngine za ten čas ušel obrovský kus cesty, nefungují v tomto modulu některé aplikace na 100%. Výjimkám a různým problémům se budu věnovat v části 6.1.1.

Integrace modulu do prohlížeče je velice jednoduchá. Na jeho domovské stránce<sup>1</sup> stačí zvolit požadovaný prohlížeč a poté kliknout na tlačítko pro instalaci modulu. Ten se do něj nainstaluje automaticky. Tato starší verze podporuje MS Internet Explorer (musí být povolena technologie ActiveX) a Netscape Navigator.

Pro prohlížeče rodiny Mozilla lze použít také, ale způsob instalace je nepatrně odlišný. Je třeba modul stáhnout<sup>2</sup>, potom musíme změnit příponu staženého souboru s modulem z **.jar** na **.zip** a

<sup>1</sup>[http://www.blender3d.org/cms/3D\\_web\\_plug-in.15.0.html](http://www.blender3d.org/cms/3D_web_plug-in.15.0.html)

<sup>2</sup><http://download.blender.org/release/>

rozbalit obsah archivu do adresáře s ostatními moduly. V některých verzích těchto prohlížečů stačí též pouze kliknout na instalační tlačítko a modul se nainstaluje automaticky. Tento způsob je však spíše sporadický.

Též vzniká nová verze, jejíž domovská stránka se nachází na<sup>3</sup>, ale protože je ve velice ranném stádiu vývoje, není jeho výstup vždy perfektní. Podrobnosti popisují v části 6.1.1.

Po nainstalování modulu se můžeme pustit do tvorby webové stránky sloužící k vlastní prezentaci. Na domovském webu uživatelů Blenderu je přístupná šablona<sup>4</sup>, po jejímž vložení do kódu stránky se bude volat rozhraní modulu. Jsou zde též podrobněji popsány možnosti nastavení, předávání parametrů real-time aplikaci, způsoby interakce HTML kódu s modulem, změna úvodní animace během načítání souboru s real-time obsahem a další potřebné informace.

### 6.1.1 Zjištěné problémy se zobrazováním

Během používání webového modulu se může vyskytnout některý z následujících zjištěných problémů, popř. jiný, na který jsem nenarazil při přípravě této práce. Dále zde přibližuji i některé nedostatky tohoto modulu.

- **(Ne)multiplatformnost**

Původní modul byl sice vyvíjen jako multiplatformní, ale pod Linuxem se mi ho nepodařilo zprovoznit v žádném z dostupných prohlížečů, protože Mozilla/Firefox nedokáží příliš spolupracovat se starým formátem používaným prohlížečem Netscape. Pod Windows však modul v prohlížeči Mozilla Firefox zprovoznit lze. Je tedy použitelný pouze na platformě Windows, naštěstí ve většině prohlížečů.

- **Verze Blenderu**

Dalším problémem je verze Blenderu. Modul ve starší verzi je totiž stavěn pro Blender Publisher (verze 2.25), který je považován za přelomový právě v oblasti GameEngine a modul byl dokončen v době, kdy byl aktuální a placený.

Tato verze Blenderu však neobsahuje téměř žádné pokročilejší GameEngine Python API, používá jinou verzi BGL a chybí jí některé funkce implementované až v pozdějších verzích. Dalším problémem může být i její rozdílné uživatelské rozhraní, což sice není problém přímo web modulu, ale uživateli navyklému na novější verzi 2.3x pravděpodobně způsobí nemalé potíže při tvorbě real-time obsahu.

- **Zprávy**

V GameEngine jsou používány pro komunikaci mezi objekty zprávy (Messages). Webový modul však tyto zprávy nedokáže interpretovat, nebudou tedy fungovat akce s nimi přidružené. Na to je při tvorbě obsahu dávat pozor a předem se vyvarovat návrhu s použitím zpráv. V real-time obsahu vytvořeném pomocí Blender Walkthrough Engine se to projeví tím způsobem, že nebudou funkční informativní texty a je potřeba zakázat jejich zobrazování (část 5.4.5).

- **Problémy novější verze Mozilla/Mozilla Firefox modulu**

Novější verze modulu se vyznačuje tím, že lze snadno integrovat do prohlížečů rodiny Mozilla a měla by implementovat některé z nových funkcí GameEngine. Ale protože není dostatečně dlouho vyvíjena, obsahuje stále některé závažné nedostatky a chyby:

- *Chybná interpretace událostí myši* – myš se chová podivně a je velice těžké se s její pomocí pohybovat a natáčet.

---

<sup>3</sup><http://www.neeneene.de/blender/plugin/>

<sup>4</sup>[http://www.blender3d.org/Education/index\\_old.php?sub=TutorialEmbedplugin](http://www.blender3d.org/Education/index_old.php?sub=TutorialEmbedplugin)

- *Pomalejší pohyb* – pokud v real-time aplikaci pohybujeme nějakou postavou či virtuálním avatarem, je pohyb znatelně pomalejší než ve starší verzi modulu. To může být pravděpodobně způsobeno použitím novější verze tickrate implementované ve vývojových verzích Blenderu, kde se podobný problém též vyskytuje.
- *Zhoršení grafické kvality zobrazení* – real-time aplikace po grafické stránce vykazuje mnohem více artefaktů než ve starší verzi modulu. Může to být způsobeno samotným webovým modulem, verzí Blenderu či špatným namapováním textur, ale tato "chyba" se projevuje vždy jen u této novější verze modulu.
- *Nestabilita* – občas se stane, že se modul chová nestabilně. Většinou po delší době činnosti. Ale jak již bylo zmíněno, je to způsobeno tím, že se jedná o rannou verzi a chyba jistě bude s příští verzí opravena. Nestabilním chováním je myšleno pozdní a často chybné zareagování na vstupní událost a zamrzání prohlížeče.

## 6.2 Využití přehrávače BlenderPlayer

Pokud máme všechna potřebná data uložena v .blend souboru a rádi bychom jejich obsah prezentovali bez spuštění Blenderu nebo použití omezených schopností webového plug-inu, lze využít aplikaci BlenderPlayer dodávanou společně s aplikací Blender.

Práce s přehrávačem je velice snadná. Stačí mu předat jako parametr jméno .blend souboru s požadovaným real-time obsahem, který přehrávač automaticky spustí. Navíc obsahuje spoustu nepovinných parametrů, s jejichž pomocí lze ovládat přehrávání. Jejich podrobnější přehled se dozvíme po provedení příkazu `blenderplayer -h`. Nachází se zde parametry pro nastavení velikosti okna (daná pozice a rozlišení nebo možnost zobrazení přes celou obrazovku), spuštění stereoskopického zobrazování a jeho typ nebo nastavení chování samotného přehrávače. Lze zapnout zobrazení ladicích proměnných, počtu snímků za vteřinu (fps) nebo zapnout či vypnout některé funkce. Tím můžeme dosáhnout vyšší rychlosti zobrazování na úkor kvality.

Po spuštění s parametrem `-c` vypisuje všechna upozornění a varování na konzoli (příkazovou řádku na systémech Windows). Tento parametr je obzvláště důležitý na platformě Windows, pokud se real-time obsah nechová, jak by měl. Zde totiž BlenderPlayer implicitně nevypisuje žádné zprávy, dokud není předán tento parametr.

Ani tento způsob prezentace není bez nedostatků. O problémech, které se mohou vyskytnout se dočteme v části 6.4.

## 6.3 Export do spustitelné aplikace

Tento způsob se od možnosti přehrávání liší pouze v tom, že je při něm vytvořena binární spustitelná aplikace, do níž je zakomponován BlenderPlayer. Ten tedy na cílovém počítači nemusí být vůbec přítomen. Výsledný soubor lze spouštět se stejnými parametry jako BlenderPlayer.

Samotný export je možno provést jediným způsobem. Pomocí hlavního menu Blenderu *File* → *Save Runtime*. Ve verzi pro Windows se v menu *File* nachází i položka *Save Dynamic Runtime...*, která tam ale přetrvává z předchozích verzí a v současnosti pouze vygeneruje chybu.

Po exportu se vytvoří požadovaný soubor, do něhož je automaticky zakompilován BlenderPlayer. Pro jeho spuštění jsou tedy potřebné jen 2 dynamické knihovny nacházející se v adresáři s Blenderem – **Python23.dll** a **SDL.dll**, popř. v linuxové verzi s koncovkou **.so**. Tyto knihovny je nutné distribuovat současně s vygenerovanou aplikací.



Také je potřeba dodat s aplikací veškeré soubory s texturami, jinak bude model během přehrávání neotexturován. Existuje možnost zahrnout všechny použité textury do .blend souboru, aby nemusely být dodávány s aplikací, ale tuto akci je třeba provést ručně (viz část 6.5).

## 6.4 Potíže při tvorbě a přehrávání real-time obsahu

V této části si povíme, jaká úskalí a chyby můžeme očekávat během tvorby, ale i přehrávání real-time obsahu. Problémům s webovým modulem se věnuje část 6.1.1, zde je tedy již nebudeme rozvádět.

Vzhledem k tomu, že je dynamický obsah spustitelného souboru interpretován pomocí integrovaného BlenderPlayeru, je množina problémů a nedostatků společná pro oba způsoby prezentace.

- **Spuštění GUI a real-time současně v rozhraní Blenderu**

Tato chyba se netýká přímo přehrávání real-time obsahu pomocí BlenderPlayeru. Může se objevit při práci s engine, je-li spuštěno rozhraní skriptu a současně stiskem klávesy P nad oknem **3D View**, popř. pomocí položky menu *Game -> Start Game*. Po ukončení real-time režimu je vyvolána chyba a GUI se ukončí. Poté musí být znovu spuštěno kombinací kláves **Alt+P** nad oknem **Text Editor** nebo pomocí položky jeho menu *File -> Run Script*. Nepodařilo se mi zjistit, čím je tato chyba způsobena a vylimínovat tak její výskyt.

- **Omezení při použití radiozity**

Vzhledem k obtížnosti procesu výpočtu radiozity má tento způsob zobrazení jistá omezení. Jedním z těch hlavních je nemožnost pozdějšího nastavení jednotlivých objektů, protože po úspěšné kalkulaci radiozity jsou veškeré zahrnuté objekty nahrazeny pouze jedním, obsahujícím výsledek výpočtu. Nelze tedy nastavit různé typy průhlednosti, průchozí strany a objekty, popř. texty jednotlivých místností.

Jedním z řešení může být otexturování modelu a nepoužívání radiozity, dalším např. rozdělení modelu na více úseků a vyřešení radiozity pro tyto úseky. Potom bude možno provést nastavení u více objektů, ale výsledek pravděpodobně nebude vypadat přesně jak by měl.

Dalším problémem radiozity je, že pokud do výpočtu zahrneme i objekt s nastaveným typem průhlednosti **Alpha**, při přehrávání ve stabilní verzi Blenderu se po chvíli kompletně celý model přepne do téhož způsobu zobrazení. Ve vývojových verzích se již tato chyba neprojevovala.

Též se může objevit i následující "chyba". Pokud se po aplikaci výsledků radiozity objeví varování, že počet materiálů výsledného modelu překročil 16, je to způsobeno tím, že byl některý materiál použit ve více objektech zúčastněných výpočtu. Proces nahrazování objektů výsledným prochází jednotlivé objekty a přiřazuje do výsledku materiály postupně. Může se tedy stát, že ve výsledku bude některý materiál zahrnut vícekrát, zatímco další v něm vůbec nebude.

- **Zobrazení informativních textů a radiozity současně**

Pokud je v real-time obsahu použit model, s vypočítanou a aplikovanou radiozitou a současně jsou použity informativní texty, během přehrávání není vidět nic jiného než právě tyto texty a barva okolního světa.

Tato "chyba" se objevuje ve všech stabilních verzích, při použití vývojové verze (část 3.1) je však již vše přehráno správně.

Dalším možným řešením je přepnutí viditelných vrstev pouze na první vrstvu (viz část 5.4.5). Protože jsou tyto texty uloženy ve druhé vrstvě, ve výsledku prezentace se neuplatní a vše je prezentováno správně.

- **Uložení snímku obrazovky při změně velikosti okna**

Jestliže uživatel během přehrávání změní velikost okna přehrávače a rozhodne se poté uložit snímek obrazovky, ten se neuloží správně, ale s rozhozeným barevným spektrem a nepřesnou strukturou. Při uložení snímku v okně o původní velikosti je vše provedeno správně.

Tato chyba se opět objevuje ve všech stabilních verzích, při použití novější, vývojové verze je snímek uložen v pořádku.

- **Použití příkazu `print` během real-time**

Tento příkaz v některém ze zpracovávaných skriptů je přímou zkázou celé aplikace, protože se skript zacyklí a aplikace se začne chovat nepředvídatelně nebo bude neovladatelná. Je to jedna z vůbec nejčastějších chyb, na kterou ale uživatelé nejčastěji zapomínají. Vyskytuje se u všech real-time výstupů, včetně přehrávání pomocí web plug-inu.

- **Použití parametru `-c` při přehrávání obsahu s chybou**

Je-li v prezentovaném obsahu zanesena chyba, většinou se přehrávání nechová správně, ale toto chování nijak neovlivní rychlost chodu aplikace. Pokud je ale při přehrávání takového obsahu předán BlenderPlayeru parametr `-c`, chybová hlášení se neustále vypisují do okna konzole a rychlost běhu aplikace se rapidně sníží.

Toto však není chyba, ale předpokládané a korektní chování. Zde ho uvádím pouze pro případ, že by uživatel nedokázal zjistit důvod zpomalení přehrávání.

## 6.5 Rozšiřující možnosti

Je nutno zmínit i použití následujících rozšiřujících možností. Ty lze využít během přehrávání nebo šíření výsledné aplikace. Všechny tyto přidané možnosti vyžadují alespoň drobný zásah uživatele a nejsou přímo zabudovány do GUI ovládacího skriptu.

### 6.5.1 Stereoskopické zobrazení

Podpora pro stereoskopické zobrazení, jehož princip je popsán v sekci 2.1 a v [3], je přímo zaimplementována do BlenderPlayeru již od verze 2.25. Podpora pro nejlépe využitelné zobrazení typu anaglyf, kdy jsou používány speciálně polarizované brýle, ale existuje až od BlenderPlayeru verze 2.35.

Při zobrazení anaglyf se k optickému oddělení dvou obrazů používají filtry. Nejjednodušší je použití barevných filtrů, např. obraz pro levé oko v odstínech červené a pro pravé oko v odstínech modré barvy. Oba obrazy můžeme smíchat a zobrazovat najednou. Stereoskopické brýle mají barevné filtry, které propustí jen správné odstíny barvy ([3, str. 530]).

Lze samozřejmě využít i ostatních typů zobrazení. Přehled všech aktuálně podporovaných získáme pomocí příkazu `blenderplayer -h`. Současnou verzí 2.36 jsou podporovány tyto typy:

- `hwpageflip` – využití HW podpory střídání obrazu.
- `syncdoubling` – obrazy pro každé oko jsou zobrazeny pod sebou (obr. 9.2).
- `sidebyside` – obrazy pro každé oko jsou zobrazeny vedle sebe (obr. 9.3).
- `anaglyph` – překrývající se obrazy pro jednotlivé oči odlišené barevným odstínem (obr. 9.4).
- `vinterlace` – podobné jako `anaglyph`, pouze využívá techniky šrafování místo různých barevných odstínů (obr. 9.5).

Real-time obsah lze spustit ve stereoskopickém zobrazení pouze pomocí BlenderPlayeru. Tato volba tedy nelze použít ve webovém modulu.

Přehrávání s podporou stereo-zobrazení spustíme jako `blenderplayer -s <stereomód>` nebo vyexportované spustitelné aplikaci předáme parametr `-s <stereomód> blend_soubor`. Pokud tedy vyžadujeme spouštět stereoskopický mód pomocí vygenerované aplikace, musíme současně dodat i datový `.blend` soubor.

Pro stereomód `anaglyph` je ve standardní šabloně (popř. v jakékoli jiné šabloně, v níž byl ponechán objekt `engProperties`) naimplementována možnost ovládání rozteče jednotlivých polarizovaných obrazů. Klávesy `+ a -` na numerické klávesnici korigují vzdálenost obrazů o menší krok, klávesy `* a /` na numerické klávesnici korigují vzdálenost o větší krok.

### 6.5.2 Zabalení textur

Importujeme-li otexturovaný model a přejeme si výsledek dále šířit, máme v podstatě dvě možnosti. Uložíme výsledný projekt i se soubory textur a vše budeme distribuovat společně, nebo zabalíme textury do `.blend` souboru. Poté nemusí být soubory s texturami dodávány s tímto `.blend` souborem, ale textury se stanou jeho součástí.

Bohužel, Blender Python API neobsahuje funkci pro zabalení textur, takže je nutné tento krok provést ručně. Není to ale vůbec obtížné.

Přepneme si některý z panelů pomocí výběru modulů na **UV/Image Editor**. Zde se na liště nachází rolovací seznam všech textur obsažených ve scéně (obr. 6.1). Volíme zde tedy postupně naimportované textury a u těch, které chceme distribuovat jako součást `.blend` souboru (popř. vyexportované aplikace) zaktivujeme tlačítko **Pack Texture** (s ikonou balíčku). Po dalším uložení projektu již budou zvolené



Obrázek 6.1: Lišta modulu UV/Image Editor

textury nedílnou součástí `.blend` souboru stejně jako případně vygenerované spustitelné aplikace. Textury, které nebyly zabaleny, musí být i nadále distribuovány s vlastním obsahem.

### 6.5.3 Zprůhledňování dveří

V současném stavu projektu je "otevírání" dveří řešeno způsobem, že jsou pouze průchozí. Jistě by bylo možné je otevírat klasickým způsobem, což by ale bylo téměř prakticky nerealizovatelné, protože se musí specifikovat strana, na níž se nachází panty, přesunout na ni pivot objektu a dále vytvořit logické bloky s požadovaným pohybem dveří, což by se neobešlo bez zásahu uživatele. Navíc existuje spousta typů dveří a mnoho způsobů otevírání, engine by tedy musel obsahovat velké množství různých IPO křivek specifikujících pohyb různých dveří.

Dalším možným způsobem je jejich plynulé zneviditelnění během přibližování se k nim a zviditelnění při oddálení. Tento způsob je velice snadno realizovatelný, bohužel nelze pomocí skriptů vytvářet logické bloky. Vyžaduje tedy jednoduchý zásah uživatele.

Je třeba identifikovat všechny objekty, které budou ve výsledném modelu zastupovat dveře. To je provedeno pomocí tlačítka **Door** na druhé záložce Blender Walkthrough Engine (část 5.4.3). Na stejné záložce je poté nutno přepnout typ zobrazení ze **Solid** na **Alpha**. Též je nutné, aby se střed (pivot) dveří nacházel v jejich opravdovém středu, jinak bude výsledek nevyzpytatelný. Případné přesunutí pivotu do středu objektu lze provést na panelu **Buttons Window** v sekci **Editing** tlačítkem **Centre New**.

Pro každý objekt dveří musíme po nastavení jejich identifikace vytvořit logickou cestu kolizního senzoru. Přepneme se tedy na panelu **Buttons Window** do sekce **Logic** – ikonka s modrou hlavou. Zde je třeba provést následující:

- *Vytvoření senzoru* – tlačítkem **Add** ve sloupci Sensors vytvoříme nový senzor. Změníme jeho typ na Near, položku Property nastavíme na identifikační proměnnou avatara (ve standardní šabloně pojmenovanou avatar, viz část 5.2.3), Dist nastavíme na vzdálenost hranice od dveří, kdy se již má objekt zneviditelnit a Reset na vzdálenost, po jejíž dosažení se objekt znovu zviditelní.
- *Vytvoření ovladače* – tlačítkem **Add** ve sloupci Controllers vytvoříme nový ovladač. Ponecháme jeho typ na AND a propojíme se senzorem.
- *Vytvoření aktivního prvku* – tlačítkem **Add** ve sloupci Actuators vytvoříme nový aktivní prvek. Jeho typ nastavíme pomocí rolovacího seznamu na Ipo a v seznamu o řádek níže zvolíme typ Flipper. Hodnoty počátečního a koncového frame Sta a End nastavíme na 1 a 40.

Výsledek našeho snažení by měl vypadat jako na obrázku 6.2. Po této úpravě budou všechny ob-



Obrázek 6.2: Logická cesta zprůhlednění dveří

jekty představující dveře, u nichž byla provedena, zneviditelněny při přiblížení se na specifikovanou vzdálenost Dist a opětovně zviditelněny po oddálení se o vzdálenost Reset od středu dveří.

# Kapitola 7

## Závěr

Cílem této práce bylo vytvoření univerzálního engine, který by umožňoval interaktivní průchod naimportovaným modelem budovy. Ten byl vypracován pomocí programu Blender. Jeho implementace prošla mnoha etapami, mezi nimiž byl i kompletně nový návrh a následné celkové přepracování.

O tvorbě a použití tohoto engine, stejně jako základních informacích o použité aplikaci pojednává tato technická zpráva. Tu lze použít nejen jako stručnou referenční příručku k Blender GameEngine, ale i jako uživatelský manuál k vytvořenému engine.

Projekt zdaleka neobsahuje vše, co je možno naimplementovat a co umožňuje použitá aplikace, ale je plně funkční a většině uživatelům může sloužit k plné spokojenosti. Během implementace jsem přišel na některé funkce, které by ještě mohl obsahovat a zesílil tím svou funkčnost, ale ty již v rámci vypracování projektu nebyly nezbytně nutné. Lze je chápat např. jako návrhy do budoucna pro možné vylepšení této práce.

### 7.1 Návrhy pro pokračování práce

V této části bych rád zmínil některé implementační body, na něž jsem v průběhu řešení narazil, ale nepovažoval jsem je za důležité k dokončení projektu. Dále zde jsou uvedena možná vylepšení, která nejsou současnými schopnostmi Blenderu uskutečnitelná, popř. byla mimo rozsah a zadání práce.

- **Zvuky**

Vhledem k tomu, že Blender i jeho GameEngine obsahují komunikační vrstvu pro grafickou a zvukovou knihovnu SDL, lze do real-time obsahu zahrnout i podporu zvuků. V tomto projektu by bylo nejlepší použít zvuky pro chůzi, otevírání dveří (které se ve skutečnosti neotevírají, ale jsou průchozí), popř. některé další zvukové efekty. Jejich přidání není obtížné a zároveň dokáží navodit správnou atmosféru.

- **Podpora pro více formátů**

V současné době jsou podporovány pouze dva formáty souborů, ze kterých lze naimportovat model. Dalším vylepšením by mohlo být naimplementování podpory pro více formátů souborů s modely.

- **Importér pro X3D a VRML2**

Toto vylepšení souvisí i s předchozím bodem. Protože jsou X3D a VRML2 standardizované formáty pro výměnu grafických dat na webu, jsou velice dobře zdokumentovány a jedná se o textový, popř. XML popis scény [3, str. 530]. Bylo by tedy vhodné naimplementovat importéry

pro tyto formáty. Většina modelačních programů ve svých aktuálních verzích dokáže exportovat data do těchto formátů, jejich podpora by tedy mohla být vítána.

- **Real-time stínování**

Tato možnost stínování není současnou verzí Blenderu podporována. Zde jsou stíny spočítány pouze staticky a poté jednoduše připojeny k modelu. Do budoucna by ale mohlo jít počítat stíny a radiozitu dynamicky a aplikovat je na objekty v reálném čase. Poslední verze Tuhopuu by již jednoduchou podporu dynamického stínování měla obsahovat, bohužel se mi ho nepodařilo zprovoznit.

- **Širší možnosti nastavení skla**

Dalším přídatkem by mohla být možnost měnit texturu skla přímo pomocí engine. Dalo by se tak vytvářet sklo o různé průhlednosti a hustotě, což by uživatel mohl přímo ovlivňovat texturou s alfa maskou.

## Kapitola 8

# Slovníček pojmů

**BGL** – Blender OpenGL. Rozhraní Blenderu implementované kompletně v OpenGL, což zajišťuje rychlost a multiplatformnost grafického rozhraní. Tento modul obsluhuje volání OpenGL příkazů ve skriptech.

**Blender Python API** – API rozhraní Blenderu pro interpret jazyka Python. Najde široké uplatnění při tvorbě různých skriptů a modulů běžících v prostředí Blenderu. Umožňuje práci s veškerými moduly Blenderu (kromě Game Engine), práci s objekty, vertexy, IPO křivkami, .blend soubory, světly, texturami a dalšími objekty uloženými v .blend souborech.

**Blender GameEngine Python API** – API rozhraní modulu GameEngine pro interpret jazyka Python. Používá se při tvorbě real-time aplikací, převážně her. Umožňuje reagovat na události nebo různé události vytvářet a volat. Je používáno pouze vnitřním rozhraním Blenderu během spuštěné real-time aplikace.

# Literatura

- [1] Findseiss, F., Heizer, A., McKay, R., Oppel, J., Roosendaal, T., Selleri, S., Veldhuizen, B., Wartmann, C.: Blender Documentation, 2003. Dokument dostupný na URL <http://download.blender.org/documentation/BlenderManual.pdf.zip>
- [2] Beazley, M. David: Python Essential Reference, 2001. New Riders Press.  
Český překlad: Zavadil, Petr RNDr., 2002. Praha, Neocortex
- [3] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika, druhé, přepracované a rozšířené vydání, 2004. Brno, Computer Press

## **Periodika:**

- [4] CHIP – Magazín informačních technologií, ročník 15, číslo Duben/2005

## **Internetové zdroje:**

- [5] <http://www.blender.org/modules/documentation/>
- [6] <http://www.blender.org/modules/documentation/236PythonDoc/index.html>
- [7] [http://www.blender.org/modules/documentation/pydoc\\_gameengine/PyDoc-Gameengine-2.34/index.html](http://www.blender.org/modules/documentation/pydoc_gameengine/PyDoc-Gameengine-2.34/index.html)
- [8] <http://www.oskarimax.cz>

Všechny zde uvedené URL zdrojů byly na těchto internetových adresách dostupné v dubnu 2005.



## Kapitola 9

# Přílohy

Přiložené CD médium obsahuje následující položky:

- **Blender Walkthrough Engine** – výsledný skript, který byl cílem této práce. Nalézá se v adresáři `BlenderWalkEngine` jako soubor `WalkEngine.blend`. Pouze skript je k nalezení ve stejném adresáři jako soubor `WalkEngine.py`.
- **Tato zpráva** – elektronická forma této zprávy je uložena v adresáři `Documentation` v souboru `TechnickaZprava.pdf`.
- **Zdrojové kódy technické zprávy** – zdrojové kódy této zprávy se nalézají ve stejném adresáři. Patří jim podadresář `Sources`. Technická zpráva je vytvořena pomocí typografického systému  $\text{\LaTeX}$ .
- **Poslední verze Blenderu** – adresář `Blender` obsahuje všechny poslední verze Blenderu, které byly dostupné v čase dokončení tohoto projektu. Jsou rozděleny podle použitého systému do adresářů `Windows` a `Linux`.
- **Ukázkové modely** – z adresáře `Misc/Models` můžeme použít ukázkové modely, na kterých byla testována funkčnost engine.
- **Fonty** – v adresáři `Misc/Fonts` se nacházejí některé z fontů, které lze použít v Blenderu. Ty musí být správně namapovány.

V přílohách dále nalezneme návod k použití Blender Walkthrough Engine a některé zajímavé snímky obrazovky.

### 9.1 Návod k použití Blender Walkthrough Engine

Přestože je práce s Blender Walkthrough Engine poměrně jednoduchá a přímočará, rozhodl jsem se sepsat stručný návod k jeho použití.

#### 9.1.1 Potřebné ovládání Blenderu

Veškeré klávesy a události myši odchytává v Blenderu panel, nad kterým se právě nachází kurzor myši. Na to je třeba dávat pozor, budeme-li chtít s některým z panelů provádět nějakou akci.

Jestliže bude místo GUI zobrazen jen prázdný panel **Scripts Window**, přepneme se do modulu **Text Editor**, zvolíme skript `WalkEngine.py` a spustíme ho znovu.

Potřebujeme-li posouvat obsahem některého panelu, můžeme to provést pomocí stisknutého prostředního tlačítka myši. Stejně tak lze pomocí prostředního tlačítka a stisknuté klávesy CTRL měnit velikost obsahu. Nedisponujeme-li myší se třemi tlačítky, lze posouvat obsahem v horizontálním směru pomocí kláves PgUp a PgDown, ve směru vertikálním stejnými klávesy se současně stisknutou klávesou Shift, popř. měnit velikost obsahu panelu se současně stisknutou klávesou CTRL.

### 9.1.2 Import modelu (obr. 5.2)

Máme-li vytvořen model budovy, pomocí tlačítka **Browse** tento soubor nalezneme. Popř. můžeme vepsat cestu k souboru do textového pole **Model path**. Stiskneme tlačítko **Import model**.

Jestliže proběhl import modelu v pořádku, tlačítko se zůstane stisknuté a zpřístupní se další ovládací prvky.

### 9.1.3 Nastavení scény (obr. 5.2)

V rolovacím seznamu zvolíme kalibrační objekt (krychli 1x1). Pokud tento objekt nemá být součástí scény – slouží pouze ke kalibraci scény – stiskem tlačítka **Invisible 1x1** ho zneviditelníme.

Chceme-li změnit velikost naimportovaného modelu, aktivujeme tlačítko **Resize model**. Objeví se nové ovládací prvky. Zde pomocí tlačítka **Auto** můžeme nechat vypočítat poměr velikosti modelu ku kalibračnímu objektu automaticky. Rozměry lze poté upřesnit pomocí posuvníku **Ratio**, tlačítka **Small**, kterým lze určit směr změny velikosti a poté stiskem **Apply new**. Tím aplikujeme nový poměr velikosti na model.

Dále posuneme avatara na požadovanou pozici, pokud nám jeho aktuální poloha nevyhovuje. Stiskem **Move avatar** zobrazíme relevantní část GUI. Číselnými tlačítky **X**, **Y**, **Z** můžeme ručně měnit polohu avatara, tlačítkem **Position of 1x1** přesuneme avatara doprostřed kalibračního objektu a pomocí **Default position** ho vrátíme zpět na počáteční pozici.

Přejeme-li si avatara natočit do jiného směru, poslouží k tomu tlačítko **Rotate avatar**. Po jeho aktivaci se objeví posuvník **Rotation angle**. Tím lze avatara natočit v rozmezí -180° až 180°. Tento krok vypisuje na konzoli ladící zprávy Blenderu – neznamenají chybnou funkčnost Blender Walkthrough Engine. Po změně úhlu se avatar automaticky natočí do požadovaného směru.

Pokud velikost scény a pozice avatara vyhovuje našim požadavkům, máme na výběr z těchto možností. Pracujeme-li s otexturovaným modelem, který lze dobře nastavit, můžeme přejít k nastavení jednotlivých naimportovaných objektů (část 9.1.4). Jestliže pracujeme s neotexturovaným modelem, v němž je třeba počítat radiozitu, přejdeme k nastavení radiozity (část 9.1.5).

### 9.1.4 Vlastnosti importovaných objektů (obr. 5.3)

Postupně všechny objekty, které chceme nastavit, vybíráme z rolovacího seznamu a přiřazujeme jim požadované parametry. Chceme-li nastavovat parametry celého objektu současně, necháme aktivované tlačítko **Apply on all faces**. Při nastavení jednotlivých faces toto tlačítko deaktivujeme a z nového rolovacího menu volíme faces, na které se má nastavení aplikovat.

Jestliže se jedná o objekt symbolizující dveře, aktivujeme u něj tlačítko **Door**. Chceme-li použít postupné zprůhledňování tohoto objektu jako reakci na přiblížení, nalezneme postup v části 6.5.3.

Pro objekt místnosti stiskneme **Room**. Objeví se textové pole, do nějž můžeme zadat popisek, který se objeví na obrazovce během přítomnosti avatara v tomto objektu.

Skleněné objekty vytvoříme aktivací tlačítka **Glass**.

Chceme-li změnit různé zobrazovací vlastnosti objektu, použijeme tlačítka **Invisible**, **Collisions**, **Solid**, **Add** a **Alpha**. Popis jejich funkcí lze nalézt v části 5.4.3.

Jestliže máme nastavené požadované vlastnosti, přejdeme k určení parametrů real-time režimu (část 9.1.6).

### 9.1.5 Nastavení radiozity (obr. 5.4)

Pomocí tlačítek **Add all**, **Remove all** a **Add to rad. calc.** vybereme objekty, které se budou účastnit výpočtu radiozity. Po výběru materiálu z rolovacího seznamu pomocí **Emit** určíme jeho intenzitu vyzařování a pomocí **Amb.** intenzitu přijímaného difuzního světla.

V části **Radiosity render** nastavíme vlastnosti procesu radiozity, podrobnosti se můžeme dočíst v části 5.4.4, a v panelu **Radiosity tool** nejdříve sloučíme objekty pomocí **Collect meshes**, spustíme výpočet stiskem **Start calculating** a po dokončení výpočtu (lze ukončit též klávesou **ESC**) nahradíme použité objekty výsledkem pomocí **Replace meshes**. Nakonec musíme uvolnit radiozitivní kontext tlačítkem **Free radio data**. To lze provést i před samotným výpočtem, pokud chceme ještě upřesnit objekty, pro něž se bude radiozita počítat.

Pokud bychom potřebovali natavit proces radiozity podrobněji, lze to provést na podpanelu **Radiosity buttons** části **Shading** modulu **Buttons Window**.

Po dokončení procesu radiozity můžeme nastavit parametry výsledného objektu (část 9.1.4) nebo se přesunout ke konfiguraci real-time režimu (část 9.1.6).

### 9.1.6 Parametry real-time režimu (obr. 5.5)

Rychlost pohybu avatara můžeme ovlivnit dvěma způsoby. Rychlost klasického pohybu můžeme změnit nastavením příslušné hodnoty posuvníku **Speed**, rychlost pohybu simulujícího běh (stisknutá klávesa **Shift** během pohybu) pomocí posuvníku **Accel**.

Pokud chceme vložit mezi snímky čekací smyčky, nastavíme **Sleep** na hodnotu různou od 0. Čím vyšší hodnota, tím méně snímků bude za vteřinu zobrazeno.

Používáme-li informativní nápisy, můžeme změnit použitý font pomocí rolovacího seznamu **Engine font**.

Máme-li nastaveno vše potřebné, přejdeme do poslední etapy práce, uložení výsledků (část 9.1.7).

### 9.1.7 Uložení výsledků (obr. 5.5)

Do pole **Projects dir** zadáme adresář, kam se má výsledný projekt uložit, buď přímým napsáním nebo pomocí tlačítka **Browse**, kdy bude po vybrání souboru odseknut název souboru a použita adresářová cesta.

V poli **Project name** specifikujeme název projektu. Bude tak pojmenován podadresář, do kterého budou uloženy všechny potřebné soubory.

Pomocí **Model file** určíme, zda chceme k cílovému projektu uložit i zdrojový soubor s modelem, pomocí **Textures** zase, jestli se mají uložit i použité textury. To je potřeba, pokud jsme textury nezabalili do projektového souboru, viz část 6.5.2.

Jestliže máme v projektu spočítanou radiozitu, používáme současně i informativní texty a chceme výsledek prezentovat pomocí stabilní verze Blenderu (2.36), musíme tlačítkem **Using for web or dynamic** zakázat zobrazení textů.

Posledním krokem je uložení specifikovaných souborů do projektového adresáře pomocí tlačítka **Save project**.

## 9.2 Obrázky

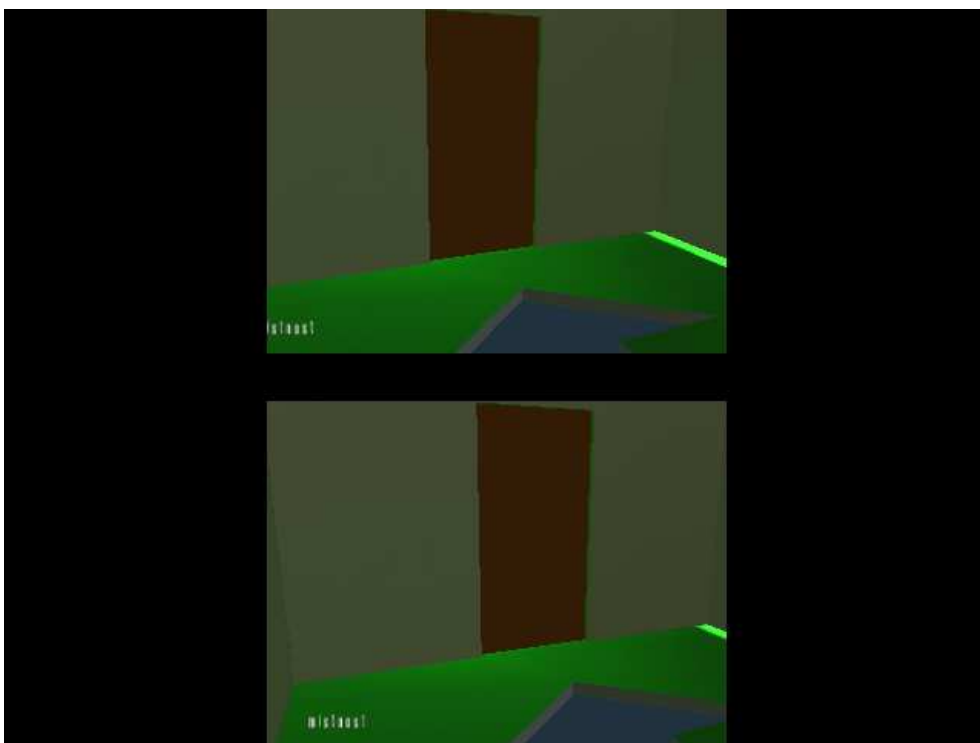
Následují obrázky důležitých prvků nebo zajímavých snímků obrazovky.

```

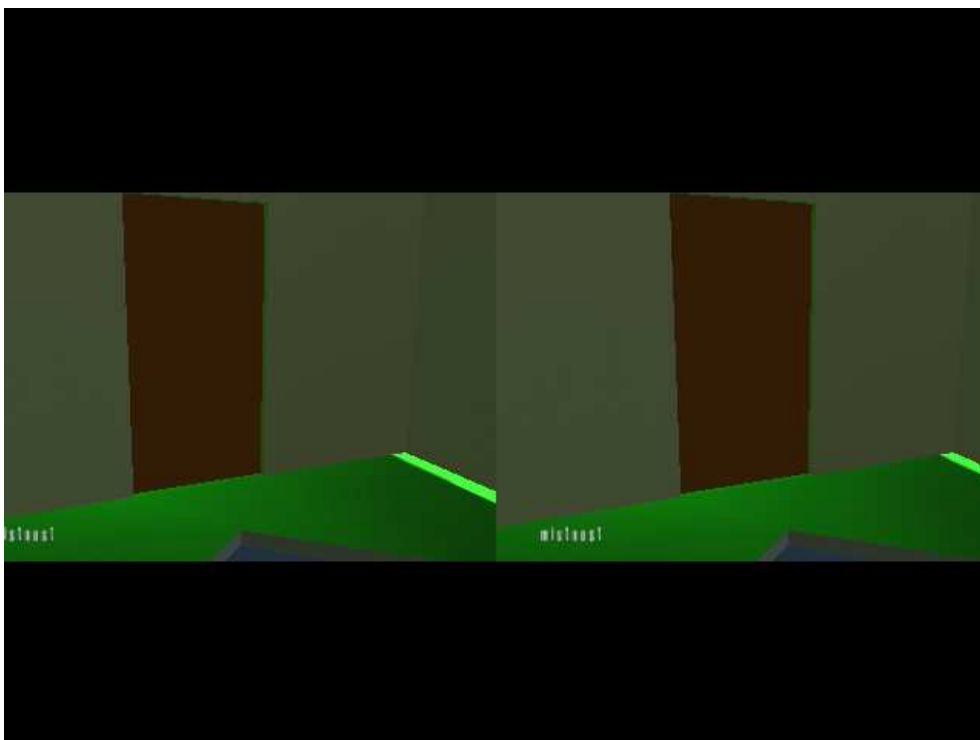
31
32 #
33 ###
34 #####
35 #####
36 #####
37 #####
38 #####
39 # SPECIFICATIONS OF ENGINE CAPABILITIES - CONFIGURATION PART
40 #####
41
42 # use images - huge memory eater!!
43 useImages = 1
44
45 # show logo - show Blender Walkthrough Logo
46 showLogo = 1
47 showFonts = 1
48
49 # engine has fonts
50 useFonts = 1
51
52 # font names, first item has to be '!'
53 strFonts = ['!', 'Benegraphic', 'Courier', 'Dimurphic', 'Hatten']
54
55 # recalculate normals of imported objects
56 recalcNormals = 1
57
58 # avatar object name
59 avatarName = 'engViewer'
60 propertiesName = 'engProperties'
61
62 # avatar's IPO curve
63 avatarIpo = 'ipoViewer'
64 # alpha IPO curve - door's invisibility - doesn't work, yet!
65 alphaIpo = 'ipoAlpha'
66
67 # initial objects, which are part of walkthrough template
68 initObjs = ['engViewer', 'engShoulder', 'engCamera', 'engProperties']
69 # font objects
70 initObjs += ['engIBenegraphic', 'engTCourier', 'engTDimurphic', 'engTHatten', \
71             'engTCopyright', 'engTRoomText']
72
73 # initial images, which are part of engine
74 initImgs = ['fnBenegraphic.tga', 'fnCourier.tga', 'fnDimurphic.tga', 'fnHatten.tga']
75
76 # initial materials
77 initMats = ['engWalkthrough']
78
79 #####
80 # END OF CONFIGURATION PART
81 #####
82 #####
83 #####
84 #####
85 #####
86 ###
87 #

```

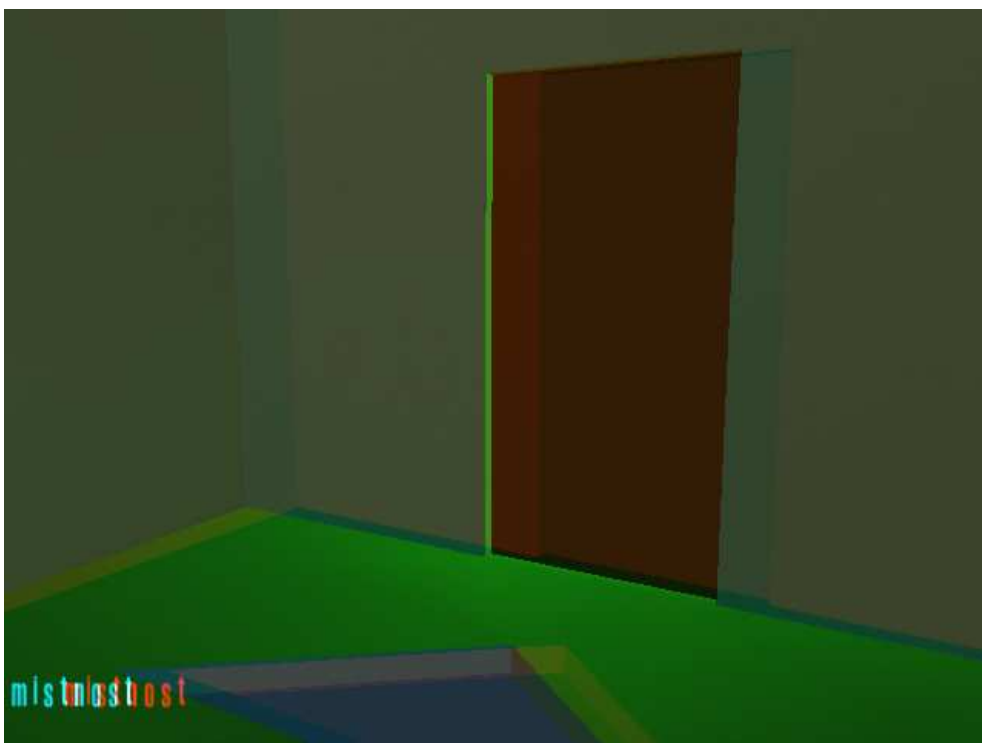
Obrázek 9.1: Konfigurační sekce skriptu. Zde je potřeba přenastavit proměnné při použití jiné šablony avatara. Více podrobností je popsáno v části 5.2.3.



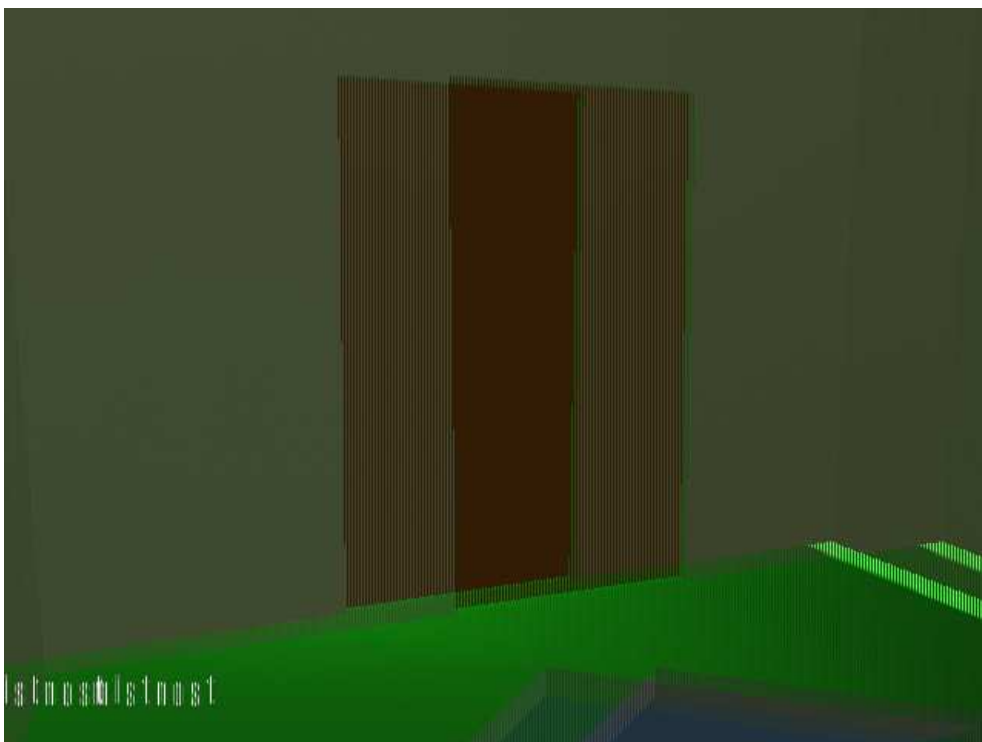
Obrázek 9.2: Stereoskopické zobrazení – syncdoubbling



Obrázek 9.3: Stereoskopické zobrazení – sidebyside



Obrázek 9.4: Stereoskopické zobrazení – anaglyph



Obrázek 9.5: Stereoskopické zobrazení – vinterlace