BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Aplikace s vkládáním virtuálních předmětů do záběru kamery

Karel Popelka*

Abstrakt

Hlavním cílem projektu je vložení virtuálního objektu do záběru kamery tak, aby objekt působil co nejpřirozeněji ve scéně a zároveň budil dojem, že se ve scéně přímo nacházel. Řešením je vytvoření aplikace na platformách Windows 8.1 a Windows Phone 8.1 se zaměřením na uživatelské rozhraní. Aplikace byla vyvíjena iterativně a při vývoji byl kladen velký důraz na odezvy uživatelů. Výsledkem projektu je mobilní aplikace, která umožňuje vložení různých virtuálních předmětů do záběru kamery se stíny. Dále umožňuje efekty jako rozmazání stínu nebo nastavení jeho intenzity a možnosti nastavení vlastností světla.

Klíčová slova: Rozšířená realita – DirectX – Windows Phone – Uživatelské rozhraní – Vkládání virtuálních předmětů – Stíny – Osvětlení

Přiložené materiály: video s ukázkou použití aplikace

*xpopel15@stud.fit.vutbr.cz, *Fakulta informačních technologií VUT v Brně*

1. Úvod

Cílem projektu je vyvinout uživatelsky přívětivou a intuitivní aplikaci pro mobilní platformy Windows 8.1 a Windows Phone 8.1. Dále se projekt zaměřuje na technologie Microsoftu, kterými jsou DirectX, Media Foundation, COM objekty a na jejich praktické využití při vývoji. Cílem je také vyvinout aplikaci na základě moderních přístupů, mezi které patří paralelní programování, lambda funkce nebo automatické odvozování typů.

Projekt je založen na řadě experimentů, které zkoumaly lidské vnímání reálného světa. Výzvou je také propojení reálného světa s virtuálními objekty. Dále jsou řešeny otázky, jak správně matematicky popsat reálný svět tak, aby byl zachován jeho charakter.
V poslední části projektu se řeší vývoj uživatelského rozhraní a přirozené ovládání virtuálního objektu, například jeho setrvačnost při rotaci.

Vývoj projektu byl inspirován analýzou již existujících řešení. Protože je obor rozšířené reality poměrně mladý, neexistuje mnoho dostupných mobilních aplikací s podobnou tématikou. Přesto bylo analyzováno několik řešení a byly vybrané některé principy, které byly upraveny a přizpůsobeny aplikaci.

Ve výsledné aplikaci může uživatel vybrat virtuální objekt, který vloží před kameru v reálném čase i s jeho stínem. Scénu je možné pozastavit a objekt může být posouván, otáčen nebo zvětšován. Uživatel může také zvolit polohu slunce a natočení roviny, na kterou je stín promítán. Nakonec je možné nastavit ostatní vlastnosti scény tak, aby virtuální objekt působil přirozeným dojmem.

2. Potřebné teoretické základy pro projekt

V této kapitole jsou zmíněny základní teoretické znalosti z pohledu vývoje projektu.

Windows Runtime komponenty a Microsoft technologie

Windows Runtime (WinRT) komponenty jsou založeny na COM objektech a nativně podporují x86 a ARM architekturu. V projektu je vytvořená WinRT komponenta, která umožňuje vykreslování virtuálního objektu v C++/CX (component extensions) pomocí DirectX a spouštění v XAML.

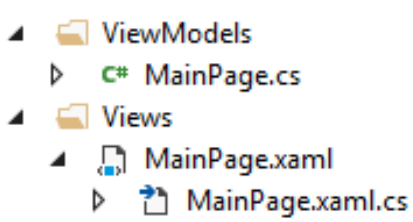
Dále je využit framework Media Foundation (MF), který umožňuje psát nízkoúrovňové komponenty do toku multimediálních dat od jejich načtení, až po jejich zobrazení. Je tedy možné psát své vlastní transformace (MF Transformation – MFT) nebo výsledné zobrazení medií, tzv. Media Sink. V projektu je framework Media Foundation využit pro získání náhledu z kamery pomocí komponenty Media Sink.

Uživatelské rozhraní v C# a návrhový vzor Model-View-ViewModel

V projektu jsou implementovány vlastní třídy pro podporu návrhového vzoru Model-View-ViewModel (MVVM) v C#. Tím je oddělen vývoj vzhledu od samotné logiky. Třídy jsou implementovány dle aktuálních programovacích metodologií Microsoftu.

Projekt obsahuje třídy *ViewBase* and *ViewModelBase*, které využívají reflexi v C# a díky tomu propojí vše dohromady automaticky. Dále jsou dostupné pomocné třídy pro navigační a notifikační servis.

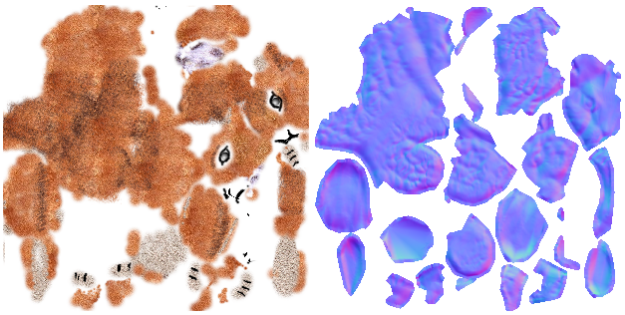
Stačí tedy vytvořit třídy reprezentující novou stránku se stejným jménem v patřičných složkách, které dědí z bázových tříd a všechno se automaticky na pozadí samo propojí. Poté je možné používat pomocné příkazy navigace z XAML nebo z C#. Výsledná souborová struktura MVVM je vidět na obrázku 1.



Obrázek 1. Souborová struktura návrhového vzoru MVVM při vytvoření MainPage stránky.

Normal mapping

Normal Mapping je technika na zlepšení vzhledu objektů s nízkým počtem trojúhelníků, která uspoří procesorový čas na úkor vyšší paměťové náročnosti. Je vytvořena normálová mapa, ve které jsou vypočítány normály v Tangent-Bitangent-Normal (TBN) prostoru. Výsledek je možné vidět na obrázku 2 s korespondující barevnou mapou.



Obrázek 2. Barevná mapa [1] (vlevo) a normálová mapa (vpravo).

V projektu je využit TBN prostor, aby bylo možné provádět libovolné afinní operace s objektem při korektním osvětlení. Každý bod je pak převeden do svého vlastního TBN prostoru. Výsledné pixely normálové mapy jsou v TBN prostoru daného bodu.

V projektu je normal mapping využit s phongovým osvětlovacím modelem [2]. Výsledek implementovaných metod na virtuálním objektu s nízkým počtem trojúhelníků lze vidět na obrázku 3.

Obrázek 3. Low-poly model bez normal mapping (vlevo) a s využitím normal mapping (vpravo).

Post processing

Cílem projektu je také vytvořit pěkný stín a nabídnout uživateli možnost jeho rozmazání nebo upravení intenzity. Z tohoto důvodu jsou implementovány post processing techniky, které vykreslí obraz do textury, upraví a znovu vykreslí. Na rozmazání stínu je využit Gaussian Blur [3] a na upravení intenzity alpha blending v DirectX 2D.

Koncept Device-Independent Pixels

V dnešní době je mnoho dostupných mobilních zařízení. Tato zařízení mají různá rozlišení i různě velký displej. Důraz byl proto kladen i na správné zobrazení aplikace na libovolném zařízení. Z těchto důvodů je nutné dodržovat Device-Independent Pixels (DIPs) koncept,

který umožňuje programovat nezávisle na rozlišení a velikosti displeje. V případě dodržování konceptu DIPs se programátor může vyhnout nesprávnému zobrazení layoutu, ořezávání UI elementů nebo nesprávnému výpočtu souřadnic myši.

V aplikaci jsou všechny souřadnice na obrazovce počítány v DIPs, které lze vypočítat z DPI na základě rovnice (1).

$$DIPs = pixels/(DPI/96.0) \quad (1)$$

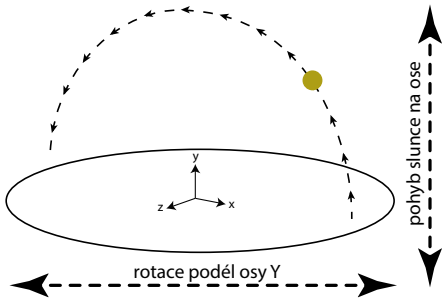
3. Návrh projektu

Na začátku kapitoly je popis analýzy již existujících řešení a závěr se věnuje iterativnímu vývoji vzhledu aplikace.

Analýza již existujících řešení

Před samotným začátkem projektu byla analyzována již existující řešení. Protože je rozšířená realita poměrně mladý obor a mobilní zařízení neměla doposud dostatečný výkon, neexistuje tolik aplikací se stejným tématem. Z tohoto důvodu byly analyzovány podobné aplikace a z těch byly vybírány využitelné principy do projektu.

Například ovládání objektů nebo stínů bylo inspirováno 3D programy na modelovaní například Blender, a je řešeno rozdílně pro ovládání plochy stínu a modelu. Nastavení slunce ve scéně bylo inspirováno aplikaci Sunrise Sunset pro Android. Dále byl tento princip upraven a přizpůsoben aplikaci. Výsledek je vidět na obrázku 4.

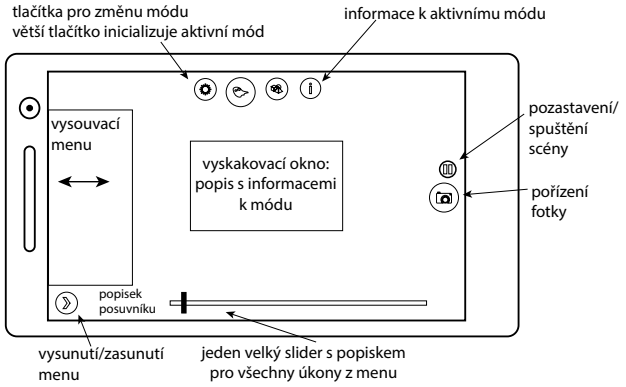


Obrázek 4. Základní princip nastavení pozice slunce ve scéně.

Iterativní vývoj návrhu

Nejdůležitější část při návrhu uživatelského rozhraní byla možnost nabídnout uživateli nastavení co nejvíce různých aspektů scény bez toho, aby byl uživatel zmatený.

Cílem bylo také nabídnout uživateli co nejvíce dostupné ovládací plochy a zároveň zobrazení co nejméně uživatelských prvků v jeden okamžik. Návrh probíhal iterativně a to tak, že bylo vytvořeno pro každou specifickou situaci více návrhů, které byly předloženy uživatelům a následně upravovány do finální podoby. Výsledný návrh je na obrázku 5.



Obrázek 5. Finální návrh UI.

Všechny texty, které jsou zobrazené, mají průhledný podklad, který je inverzní k barvě textu a rámeček v barvě textu. Animace uživatelského rozhraní jsou nelineární a nesymetrické. Díky tomu působí ovládání aplikace přirozeně.

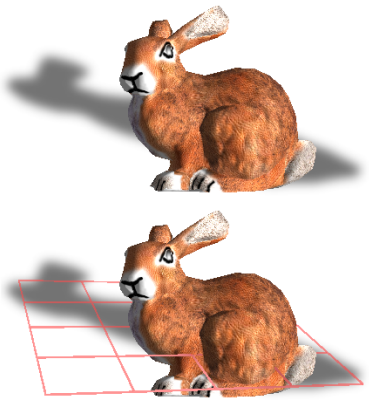
V dolní části se nachází posuvník s popiskem, který je určen pro všechna nastavení využívající posuvník. Posuvník při změně účelu zajede a znovu vyjede, aby byla naznačena změna kontextu.

V pravé části se nachází tlačítko pro pozastavení nebo spuštění náhledu z fotoaparátu a tlačítko pro vyfocení scény. Pokud je náhled z fotoaparátu pozastaven, tlačítko pro pořízení fotky je neaktivní.

Horní část obsahuje tlačítka pro změnu módu ovládání aplikace. Aktivní mód má zvětšené tlačítko tak, aby bylo jasně rozeznatelné od neaktivních módů. Vlevo se pak nachází tlačítko, které po stisknutí zobrazí vyskakovací menu s nápovědou k aktivnímu módu.

Ovládání virtuálního modelu má setrvačnost a reaguje jak na dotyková zařízení, tak i na vstup z myši. Je možné provádět klasické úkony jako je rotace, posun a zvětšování.

Ovládání stínu je řešeno pomocí mřížky, která naznačuje rovinu, na kterou je stín vržen. Pro uživatele bylo matoucí nastavovat rovinu stínu, kterou neviděli. Když se dostal stín do nekorektní polohy a občas i zmizel, uživatelé nevěděli, jak chybu opravit. Naproti tomu mřížka se zarovnává s reálnou hmatatelnou plochou a ovládání je mnohem jasnější. Lze pak nastavovat libovolnou polohu roviny a její vzdálenost od virtuálního objektu. Výsledek je vidět na obrázku 6.



Obrázek 6. Nesprávné nastavení stínu bez mřížky (nahoře) a s mřížkou (dole).

4. Implementace projektu

Projekt je implementován v jazycích C++/CX a C#/XAML. Vykreslování záběru kamery, virtuálního objektu a jeho stínu je řešeno v DirectX 11 jako WinRT komponenta, která dědí z UI elemantu SwapChainPanel a je načtena do XAML.

V komponentě WinRT je chod urychlen pomocí nových programovacích přístupů, jako je například paralelní načítání zdrojů aplikace. Celá aplikace pak vytvoří několik vláken. Například je vytvořeno speciální vlákno pro odchytávání vstupů nebo speciální vlákno pro vykreslování. Vše je pak synchronizováno přes události a kritické sekce. Díky tomu aplikace nikdy nebudí dojem zamrznutí, i když zrovna provádí složitější proces na pozadí.

Výsledná aplikace byla vytvořena dle výše uvedeného návrhu v kapitole 3 a výsledek je možné vidět na obrázku 7.

Načítání modelu

Aplikace umí načítat modely ve standardním binárním formátu Vertex Buffer Object (VBO), který v sobě obsahuje pole dat pro jednotlivé vrcholy. Data pro jeden vrchol obsahují pozici, normálu a souřadnice do textury.

Formát VBO může být vygenerován z klasického OBJ formátu, který podporuje řada 3D modelovacích programů. Podmínkou pro model tedy je, že musí obsahovat texturu s barvou modelu a volitelně normálovou texturu.

Načítání modelů z uživatelského rozhraní není prozatím implementováno, ale bude v dohledné době.

5. Závěr

Výsledkem projektu je aplikace, která umožňuje vložit virtuální objekt do záběru kamery. Aplikace je napsaná v DirectX 11 a zaměřená na platformu Windows 8.1 a Windows Phone 8.1. Uživatel může vložit virtuální objekt do reálné scény a nastavit vlastnosti virtuálního objektu tak, aby působil co nejpřirozenějším dojmem ve scéně.

Obrázek 7. Výsledný vzhled aplikace.

Osobním cíle bylo také vytvořit uživatelsky přívětivou aplikaci. Na základě reakcí testovacích uživatelů byl tento cíl splněn.

Poděkování

Chtěl bych poděkovat prof. Ing. Adamu Heroutovi, Ph.D. za kvalitní vedení práce a mnoho konstruktivních rad při řešení. Dále bych chtěl poděkovat rodičům a všem přátelům, kteří mě podporovali při vývoji aplikace.

Literatura

[1] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *ACM SIGGRAPH conference proceedings*, Jul 2002.

[2] Justin Stenning. *Direct3D Rendering Cookbook*. Packt Publishing, January 2014.

[3] Frank D. Luna. *Introduction to 3D GAME PROGRAMMING WITH DIRECTX® 11*. MERCURY LEARNING AND INFORMATION LLC, 2012.