VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE BUDOV POMOCÍ ROZŠÍŘENÉ REALITY PRO PLATFORMU ANDROID

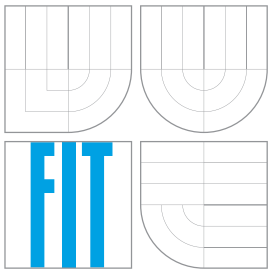
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

IVO HLAVNIČKA

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE BUDOV POMOCÍ ROZŠÍŘENÉ REALITY PRO PLATFORMU ANDROID

AUGMENTED REALITY BUILDINGS VISUALISATION FOR ANDROID PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — IVO HLAVNIČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. LÁNÍK ALEŠ
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá implementací rozšířené reality pro platformu android a její praktickému využití v architektuře, a to především k vykreslování návrhu budov do reálného prostředí. Zabývá se problémy získávání polohy a orientace zařízení, filtrování těchto údajů a přepočtu globálních souřadnic do kartézského souřadného systému.

Abstract

This thesis focuses on implentation of an Augmented Reality for the Android Platform and its practical usage in architecture, mainly for rendering of building designs into real enviroment. It deals with problems of acquiring location and orientation of device, filtering these values and conversion of global coordinates into the Cartesian coordinate system.

Klíčová slova

Android, rozšířená realita, vizualizace, architektura, java, obj, 3D grafika, geolokace.

Keywords

Android, augmented reality, visualization, architecture, java, obj, 3D graphics, geolocation.

Citace

Ivo Hlavnička: Vizualizace budov pomocí rozšířené
reality pro platformu Android, bakalářská práce, Brno, FIT VUT v Brně, 2013

Vizualizace budov pomocí rozšířené reality pro platformu Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Aleše Láníka, Ing.

.....
Ivo Hlavnička
14. května 2013

Poděkování

Děkuji tímto panu ing. Aleši Láníkovi za odborné vedení a pomoc při řešení problémů a ing. Petrovi Nohejlovi za odbornou konzultaci.

© Ivo Hlavnička, 2013.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **2**

2 Platforma Android **3**
2.1 Historie . . . 3
2.2 Klíčové koncepty . . . 3
2.3 Struktura aplikace . . . 5
2.4 Opengl ES . . . 7
2.5 Knihovny třetích stran . . . 10
2.6 Hardwarové komponenty . . . 10

3 Rozšířená realita **13**
3.1 Rozdělení . . . 13
3.2 Historie . . . 13
3.3 Použití . . . 14

4 Návrh aplikace **17**
4.1 Název aplikace . . . 17
4.2 Funkce aplikace . . . 17
4.3 Uživatelské rozhraní . . . 17
4.4 Cílová skupina . . . 18

5 Implementace **19**
5.1 Senzory . . . 19
5.2 Geolokace . . . 20
5.3 Ukládání dat . . . 25
5.4 Nastavení scény a zobrazení modelu budovy . . . 27

6 Testování **29**
6.1 Sony xperia mini pro . . . 29
6.2 Asus Transformer . . . 30

7 Závěr a možnosti rozšíření **31**

A Obsah CD **34**

B Plakát **35**

Kapitola 1

Úvod

Android je poměrně mladá platforma. Od doby svého představení společností Google, které proběhlo v roce 2007 [17], se tato platforma rozšířila do celého světa a bez jakýchkoli problémů velmi rychle přebrala světový trh. Již mnohem dříve existovala celá řada operačních systémů pro mobilní zařízení: Symbian OS, iOs, BlackBerry OS nebo Windows Mobile. Žádný se však nedočkal tak masového rozšíření, jako Android.

Ať už stabilní linuxové jádro, otevřenost zdrojového kódu nebo široká škála podporovaného hardwaru, důvodů k tomuto rozšíření může být mnoho. Mobilní trh dlouho postrádal platformu, která by svou otevřeností a modifikovatelností vyhověla i těm nejnáročnějším výrobcům a zdá se, že Android tuto funkci plní naprosto dokonale.

Od doby vzniku se mobilní telefony staly pro jejich majitele něčím mnohem větším, než jen prostředkem ke komunikaci. Diář, budík, kalkulačka a fotoaparát, to všechno jsou funkce, které již dávno nejsou nadstandardem. Po nástupu mobilních operačních systémů a následnému prudkému vzrůstu počtu vyvinutých aplikací téměř neexistuje odvětví, do kterého by tyto programy nezasahovaly.

Jedním z těchto odvětví, o němž si dovolím říci, že je stále na experimentální úrovni, je rozšířená realita. Jakožto zařízení, oplývající jak kvalitním fotoaparátem, tak mnohými senzory, akcelerometry a GPS modulem, jsou mobilní telefony a tablety velmi vhodným médiem pro vývoj a testování aplikací založených na principech rozšířené reality. Jejich potenciál je vskutku obrovský. Koneckonců – kdo by si nechtěl zjednodušit orientaci v cizím městě, zahrát si počítačovou hru ve skutečném světě nebo si prohlédnout návrh svého vysněného domu přímo tam, kde jej chce postavit?

Mobilní platformu a její vývojáře čeká ještě dlouhá cesta, než rozšířená realita dosáhne svého vrcholu. Technologie, které jsou pro to nezbytné, buď ještě nebyly vynalezeny nebo jejich rozšíření mezi běžný lid brání jisté překážky. Přesto však existuje celá řada aplikací, které využívají současný potenciál mobilních zařízení do poslední kapky a odhaluje tím nedostatky, na které by se výrobci v následujících letech měli zaměřit. Nezbývá než doufat, že pokrok půjde kupředu stejně rychle, jako naše očekávání.

Kapitola 2

Platforma Android

Android je operační systém postavený na Linuxovém jádře, primárně určený pro mobilní telefony a tablety. Jeho největší předností je pro uživatele velký počet aplikací, kterými mohou své zařízení rozšířit, pro vývojáře pak velmi promyšlený, snadno rozšiřitelný, objektový návrh systému založený na komponentách a vysoká technická úroveň systému. Díky důmyslnému použití již zavedených technologií, kterými jsou např. OpenGL ES 2.0 a SQLite má Android velice strmou křivku učení a v začátcích si vývojář vystačí se znalostí objektového přístupu a jazyka Java nebo jiného objektově orientovaného jazyka (např. C#).

2.1 Historie

Společnost Android, Inc. byla založena v říjnu roku 2003 Andy Rubinem, Richem Minerem, Nickem Searsem a Chrisem Whitem. Přestože je známo, že se tato společnost zabývala vývojem mobilního operačního systému, přesnější informace o aktivitách společnosti nejsou známy, jelikož firma pracovala v utajení. V srpnu roku 2005 společnost odkoupila firma Google a převzala tak vývoj operačního systému Android.

Prvním prodávaným mobilním telefonem obsahujícím tento systém byl HTC Dream [17], který byl vydán v roce 2008 a obsahoval systém verze 1.0.

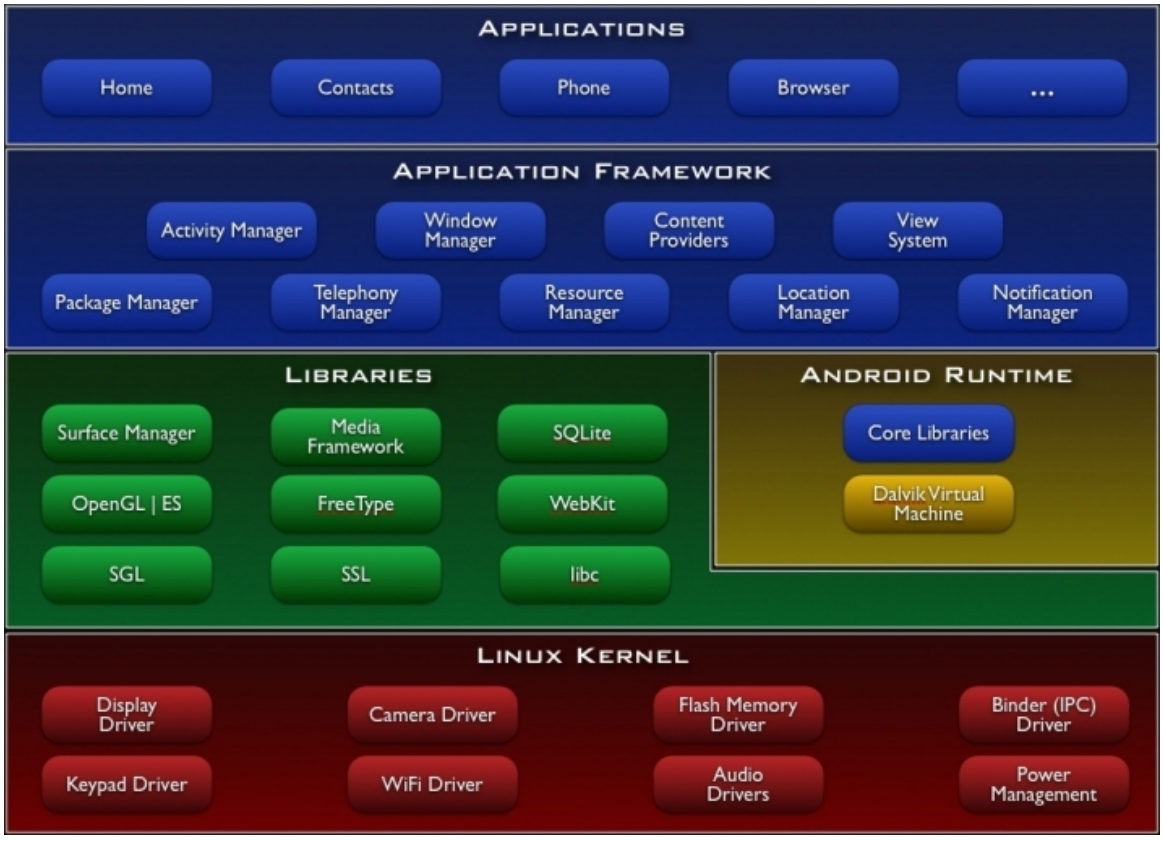
Od svého představení se tato platforma dočkala mnoha vylepšení, v současné době je nejnovější stabilní verze 4.2.2 Jelly Bean, která byla vydána 11. února 2013. Android je tedy stále aktivní a velmi rychle se rozvíjející projekt, jehož potenciál může převýšit snad jen jeho oblíbenost.

2.2 Klíčové koncepty

Tento systém vychází převážně z již existujících principů. To umožnilo jeho bleskový rozvoj a oblíbenost u vývojářů, kteří tak mohou vycházet z jejich dosavadních znalostí a zkušeností.

2.2.1 Linuxové jádro

Android je postaven na modifikovaném monolitickém jádře systému Linux, konkrétně na verzi 2.6. To poskytuje operačnímu systému vrstvu hardwarové abstrakce a umožňuje mu rozšíření na velkou řadu platform, které jsou touto vrstvou podporovány. Velmi časté je např. využití procesorů ARM nebo MIPS, které jsou pro svou nízkou spotřebu a dostačující výkon ideální pro použití v mobilních zařízeních a jsou Linuxovým jádrem podporovány.



Obrázek 2.1: Architektura platformy Google Android.

Android používá linuxové jádro pro správu paměti, procesů, síťových a jiných služeb operačního systému, a není přímo přístupné programátorovi [2].

2.2.2 Jazyk Java a XML

Java je konkurentní, třídní, objektově orientovaný programovací jazyk obecného užití [24], který je vytvořen tak, aby měl co nejméně implementačních závislostí a je tedy velmi vhodný pro multiplatformní programování. Kód napsaný v Javě se překládá do bytekódu a následně se spouští na takzvané JVM (Java Virtual Machine), která bytekód provádí.

XML je značkovací jazyk, který byl vyvinut konsorciem W3C, a jeho primárním účelem je výměna dat mezi různými aplikacemi [22]. V platformě Android slouží XML pro popis uživatelského rozhraní aplikace, ukládání textových hodnot, barev a jiných konstant, které je v systému třeba rychle a snadno měnit.

2.2.3 Dalvik VM

Dalvik Virtual Machine je implementace JVM optimalizovaná pro nízké paměťové nároky, umožňuje spustit více instancí VM a spolupracuje se skrytým Linuxovým jádrem kvůli bezpečnosti a oddělení procesů [2].

2.2.4 Nativní knihovny

Systém Android obsahuje velké množství knihoven, které umožňují snadnou práci s nejdůležitějšími souborovými formáty a poskytují rozhraní pro ukládání dat, zobrazování 2D/3D grafiky, prohlížení webových stránek a podobně. Mezi tyto knihovny patří např.:

- **OpenGL ES** – API pro zobrazování 2D/3D grafiky (více v kapitole 2.4)
- **SQLite** – Softwarová knihovna zprostředkovávající přístup k databázi
- **Kodeky** – Knihovny pro dekódování multimédií

2.3 Struktura aplikace

2.3.1 Adresářová struktura

Aplikace pro operační systém Android mají pevně danou adresářovou strukturu. Pomocí názvu složek lze dokonce definovat odlišné zdroje grafiky a textů pro různé jazykové mutace, velikosti displeje nebo verze systému. Základní struktura má následující formát:

- `src/` Zdrojové kódy aplikace
- `bin/` Výstupní složka pro uložení výsledku překladu aplikace.
- `gen/` Soubory `.java` automaticky generované vývojovým prostředím
- `assets/` Prázdná složka. Umožňuje ukládat zdroje dostupné při běhu programu s nezměněným jménem
- `res/` Texty, soubory `xml` a multimédia. V podsložkách je možné uložit více verzí pro různou lokalizaci systému, různé velikosti displejů apod.

- `libs/` Obsahuje soukromé knihovny
- `AndroidManifest.xml` viz kapitola 2.3.2
- `project.properties` Nastavení projektu, jako je např. cíl překladu. Ve vývojovém prostředí eclipse se dá upravit kliknutím pravým tlačítkem na projekt a zvolením *Properties*
- `local.properties` Stejné jako `proect.properties`, ale může specifikovat různá nastavení pro jednotlivé počítače
- `ant.properties` Nastavitelné vlastnosti překladacího systému.

2.3.2 AndroidManifest.xml

Každá aplikace musí mít soubor `AndroidManifest.xml` ve své kořenové struktuře. Ten dodává operačnímu systému elementární informace, které jsou bezpodmínečně nutné k jejímu spuštění. V souboru jsou specifikovány následující informace [7]:

- Specifikuje jméno aplikace
- Popisuje komponenty aplikace (aktivity, služby, poskytovatele obsahu a podobné), ze kterých se skládá.
- Určuje, ke kterým hardwarovým nebo softwarovým komponentám musí mít aplikace přístup pro její správnou funkci
- Deklaruje rozhraní jednotlivých aktivit sloužící k interakci s jinými aplikacemi
- Uvádí minimální verzi API, pro který je aplikace určena
- Definuje knihovny, vůči kterým musí být aplikace linkována

2.3.3 Aktivita

Aktivity jsou základním stavebním prvkem téměř každé aplikace pro android. Reprezentují právě jednu činnost, kterou uživatel aplikace může vykonat. Všechny aktivity jsou potomky třídy `Activity` a mají svůj životní cyklus (viz obrázek 2.2).

U aktivity můžeme rozlišit čtyři základní stavy [4]:

1. **Aktivní** – Označuje stav aplikace, kdy provádí činnost, je na popředí.
2. **Pozastavena** – Pokud je aplikace stále viditelná, ale její běh je pozastaven (např. když je aplikace aplikace částečně překryta jinou), přejde aktivita do tohoto stavu. Všechny procesy nadále probíhají, v extrémních případech však může být aplikace ukončena.
3. **Zastavena** – Když je aplikace posunuta do pozadí a je zcela překryta jinou aplikací, přechází do tohoto stavu. Stav aplikace je i nadále uchováván, často ale dojde k ukončení aplikace, pokud je potřeba uvolnit paměť.

4. **Neaktivní** – Stav, který nastává poté, co operační systém aplikaci ukončí. Obecně jsou dva způsoby, jak systém aplikaci ukončuje – zasláním požadavku o ukončení, nebo násilným ukončením. V obou případech musí dojít ke kompletní reinicializaci aplikace, pokud je znovu spuštěna.

Při přechodu mezi těmito stavy jsou volány metody, které provádějí akce nutné k jeho uskutečnění. Mezi tyto metody patří `onStart()`, `onPause()`, `onCreate()`, ...

2.3.4 Fragment

Fragmenty jsou části uživatelského rozhraní nebo chování, které umožňují zapouzdřit požadovanou funkčnost do samostatné třídy (potomka třídy `Fragment`) a následně ji využít v aktivitě. To podporuje přehlednost a znovupoužitelnost kódu.

Díky fragmentům je také možné vytvářet složitější uživatelská rozhraní „složením“ několika fragmentů do jedné aktivity (ilustrováno na obrázku 2.3).

Stejně jako aktivity, i Fragmenty mají svůj životní cyklus. Základní metody, jako jsou např. `onCreate`, `onStart`, `onPause`, atd. jsou shodné s životním cyklem aktivity, obsahuje však další, specifické metody které jsou zaměřeny na interakci s aktivitou a generování rozhraní [5].

2.4 Opengl ES

Opengl ES 1.0 je odnož OpenGL API verze 1.3, která je určena pro vestavěné systémy (ES – Embedded Systems), jako jsou např. mobilní telefony, tablety, herní konzole a jiná specializovaná zařízení [18].

Základní myšlenkou při vytváření OpenGL ES bylo odstranění veškerých nadbytečných implementací funkcí při zachování jediného, nejefektivnějšího způsobu řešení problému. V původním API bylo např. vícero možností, jak specifikovat geometrii objektu. Při vytváření portu pro vestavěné systémy byla ponechána pouze nejefektivnější metoda – specifikace pomocí pole vertexů[1] [10].

Tohoto cíle se však vývojáři snažili dosáhnout při zachování co nejvyšší úrovně kompatibility s původním API. Aplikace vyvíjené v podmnožině OpenGL pro vestavěné systémy by tedy měly být podle této myšlenky plně kompatibilní se standardem OpenGL ES. Toho ale v některých případech nebylo dosaženo [10].

2.4.1 Verze 1.0

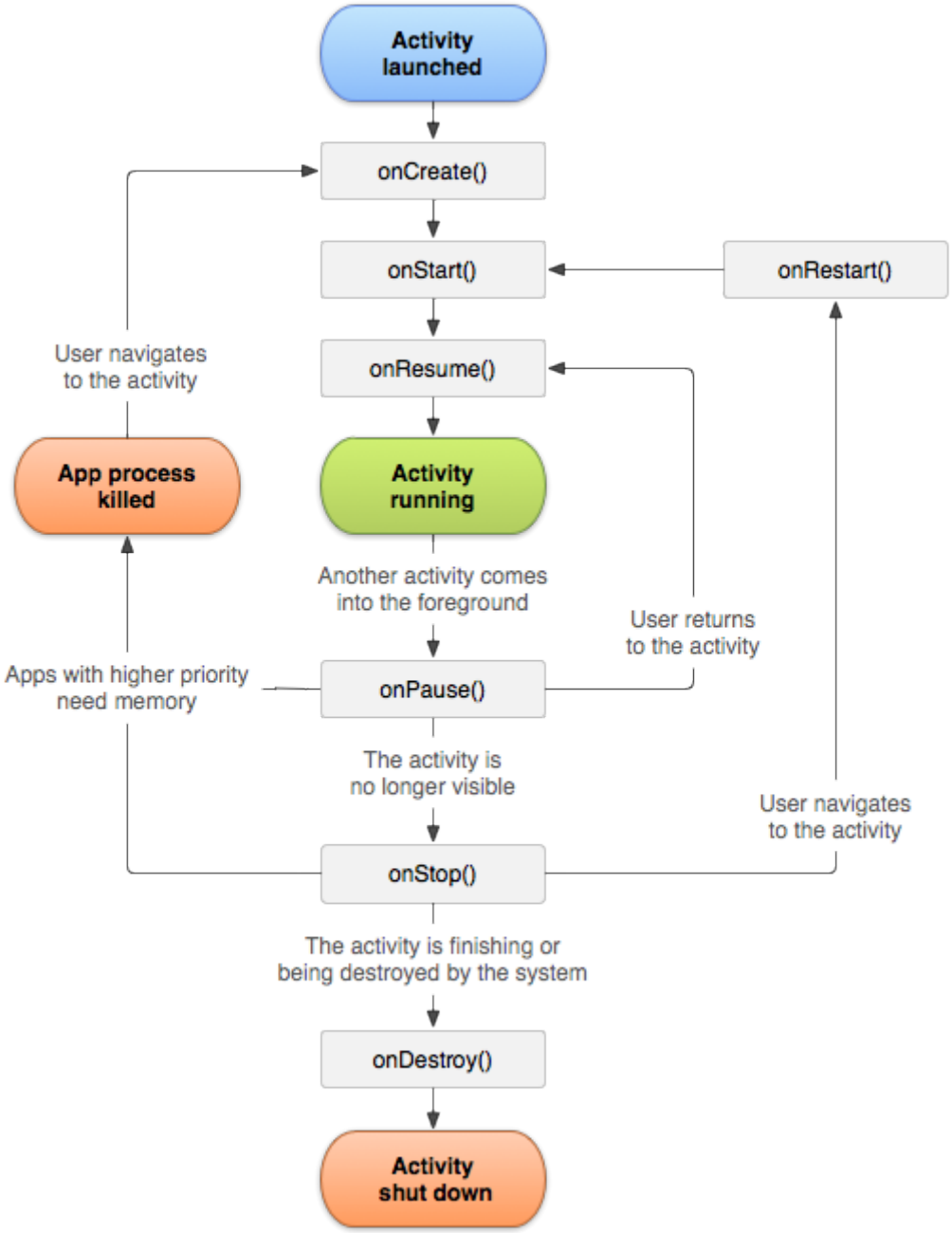
Tato knihovna implementuje velkou část funkcionality OpenGL 1.3 a přináší některá rozšíření, jako např. lepší podpora vestavěných zařízení často postrádajících FPU[2]

Mnoho funkcionalit původního API[3] bylo v tomto standardu také odebráno. Jedná se především o složitější vykreslovací funkce (nahraditelné primitivum polygon, některé parametry materiálů a mnohé další) [14].

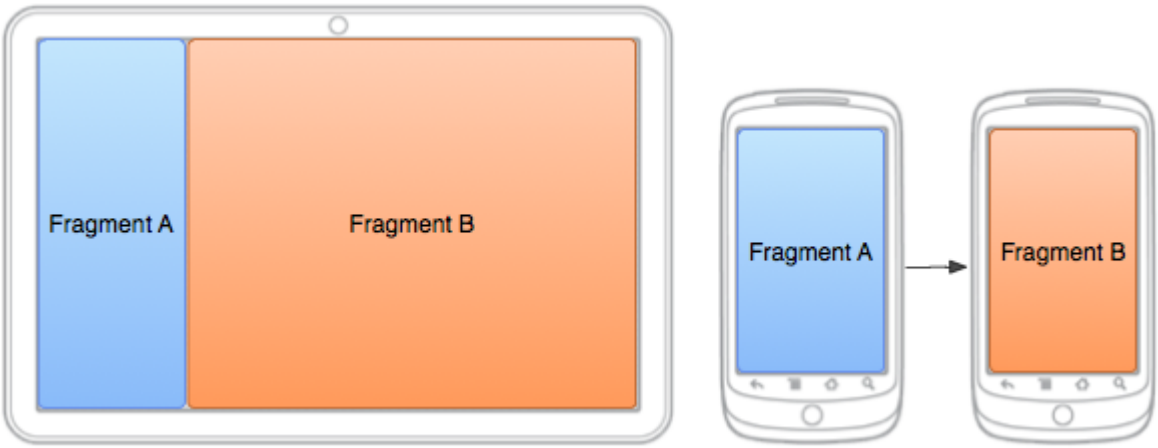
[1] Vertex – druh bodu popisující rohy, hrany nebo průniky

[2] Floating point unit – Matematický koprocesor akcelerující výpočty s plovoucí řádkovou čárkou

[3] Application programming interface – Knihovna obsahující specifikaci funkcí, datových struktur a dalších prvků definujících komunikační rozhraní aplikace [19]



Obrázek 2.2: Životní cyklus aktivity [4]



Obrázek 2.3: Princip pokročilého uživatelského rozhraní implementovaného pomocí fragmentů

2.4.2 Verze 1.1

Tato verze přidává podporu multitextur, automatické generování mipmap[4], lepší kontrolu vykreslování bodů a jiná vylepšení [14].

2.4.3 Verze 2.0

Verze 2.0 není zpětně kompatibilní s předchozími. Je založena na OpenGL 2.0, ale přináší jisté změny. Eliminuje téměř všechny fixní funkce renderovací pipeline[5] a umožňuje tak programátorovi plný přístup k jejímu chování. Další významnou změnou je také jiný způsob specifikace materiálů a parametrů světla z fixních funkcí na shadery[6].

2.4.4 Verze 3.0

V současné době nejnovější implementace OpenGL ES byla vydána v srpnu roku 2012 [14]. Jedná se o standard zpětně kompatibilní s předchozí verzí 2.0. Umožňuje tedy inkrementálně přidat do aplikací novou vizuální funkcionalitu bez změny původního kódu. Nová funkcionalita zahrnuje:

- Vylepšení renderovací pipeline o akceleraci vizuálních efektů
- Komprese textur ETC2/EAC, eliminující potřebu různých textur pro různé platformy
- Nová verze jazyka pro programování shaderů s plnou podporou 32-bitových operací s plovoucí řádovou čárkou
- Výrazně zlepšená funkcionalita texturování včetně garantované podpory floating-point textur[7], 3D textur, hloubkových textur a dalších

[4] Mipmapy jsou předvypočítané kolekce obrázků sloužící ke zvýšení vykreslovacího výkonu a redukování aliasingu [16].

[5] „Zajišťuje zřetězené zpracování grafických informací grafickou kartou“ [11]

[6] Počítačový program spouštěný na grafickém procesoru [15]

[7] Typ textur ukládající informace o materiálu pomocí čísel s pohyblivou řádovou čárkou. Používá tedy rozsah hodnot od $-\infty$ do ∞

2.5 Knihovny třetích stran

V projektu jsem použil dvě knihovny třetích stran, které mi umožnily jednoduše a efektivně implementovat některé části projektu. Všechny uvedené knihovny jsou šířeny pod licencí Apache verze 2.0 (dostupná na `http://www.apache.org/licenses/LICENSE-2.0`)

2.5.1 Rajawali framework

Rajawali je 3D engine pro platformu Android, založený na OpenGL ES 2.0. Je vhodný pro vytváření 3D aplikací i živých tapet (live wallpapers) a velmi zjednodušuje vykreslování 2D/3D grafiky v aplikacích. Podporuje i zobrazení okna rendereru ve fragmentu, což mi posloužilo k vytvoření výše zmíněného jednoduchého 3D editoru. Mezi funkce tohoto frameworku patří:

- Importování grafických souborů typu `obj`, `md2`, `3ds`, `fbx`, ...
- Všesměrové a směrové světla
- Mlha
- Široká škála materiálů
- Skybox[8]
- Vlastní shadery
- 2D/3D renderer s vlastním životním cyklem
- Quaterniony
- Kamery s perspektivní a paralelní projekcí
- Primitiva: čára, krychle, koule, částice, plocha
- Částicové systémy

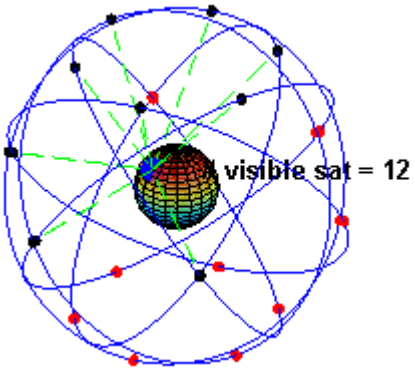
2.5.2 CWAC LoaderEx

Umožňuje vytvářet a využívat `Loadery`[9] bez použití `ContentProvideru`[10]. Místo něj využívá funkce databázového helperu.

2.6 Hardwarové komponenty

Dnešní zařízení se systémem Android jsou vybaveny různými typy senzorů, které mají široké oblasti využití. Mobilní telefony a tablety dnes většinou obsahují GPS modul ke zjištění zeměpisných souřadnic, akcelerometr sloužící k detekci pohybu nebo magnetometr měřící magnetické pole, ve kterém se zařízení nachází. Při vytváření rozšířené reality je bezpodmínečně nutné využít některých z těchto senzorů.

[8] Jedná se o texturovanou krychli ohraničující celou scénu a vytvářející tak okolní prostředí
[9] Třídy sloužící k jednoduchému načítání dat z databáze bez využití hlavního vlákna aplikace [8]
[10] Prostředek umožňující sdílet data mezi aplikacemi



Obrázek 2.4: Rozmístění GPS satelitů na zemské orbitě [20]

2.6.1 Fotoaparát

Komponenta sloužící k digitalizaci obrazu, v některých zařízeních bývá zabudován více než jeden. Podle umístění na zařízení jej můžeme rozlišit na typ přední (sloužící primárně k uskutečňování videohovorů) a zadní (sloužící k pořizování snímků). Zřídka mají zařízení pouze přední fotoaparát, příkladem může být tablet Google Nexus 7.

Zařízení obsahující pouze přední fotoaparát nejsou vhodné pro aplikace rozšířené reality. Absence zadní kamery znemožňuje přirozený průhled kamery – pro správné zobrazení scény je nutné stát zády ke sledovanému objektu.

2.6.2 GPS modul

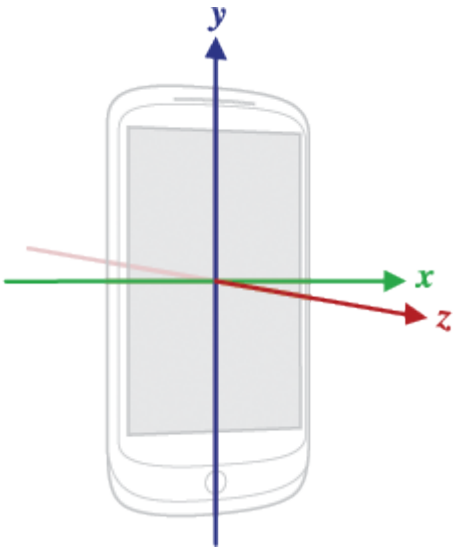
GPS[11] je systém satelitní navigace, který poskytuje informace o poloze přijímače kdekoliv na Zemi, pokud má zařízení nestíněný přístup ke čtyřem nebo více satelitům na oběžné dráze Země. Vesmírný segment tohoto systému tvoří 24 až 32 satelitů (rozmístění znázorněno na obrázku 2.4), které svým signálem pokrývají celý zemský povrch. GPS je založen na principu zachytávání signálu družic zařízením a následném výpočtu světových souřadnic (a případně výšky) ze zpoždění signálu, který se v závislosti na vzdálenosti od družic mírně liší [20].

Přesné určení polohy zařízení je jedním z klíčových kroků k vytvoření rozšířené reality. Kromě GPS modulu podporují systémy android ještě jeden způsob určení polohy – pomocí internetové sítě. Tento způsob však nemá dostatečnou přesnost a neposkytuje informace o nadmořské výšce (v případě použití tohoto způsobu by bylo nutno použít webovou službu poskytující informace o nadmořské výšce na zadaných souřadnicích – např. Google Elevation API, popsanou v sekci 5.2.3), a proto pro tuto aplikaci není vhodný.

2.6.3 Akcelerometr

Akcelerometr je komponenta sloužící k měření okamžitého zrychlení (v $m \times s^{-1}$), které působí na zařízení ve fyzických osách X, Y a Z (viz obrázek 2.5). V zařízeních Android zahrnuje měření i gravitační složku [6].

[11] Global Positioning System [20].



Obrázek 2.5: Relativní systém souřadnic použitý v systémech Android [6]

2.6.4 Magnetometr

Magnetometr měří sílu magnetického pole, ve kterém se zařízení nachází. Protože jsou hodnoty v osách X, Y a Z (viz obrázek 2.5) měřeny zvlášť, lze podle nich lze určit orientaci zařízení vzhledem k magnetickým pólům Země (funkce kompasu). Senzor provádí měření v jednotkách μT, ale s hodnotami v tomto tvaru se většinou výpočty neprovádějí přímo.

Kapitola 3

Rozšířená realita

Rozšířená realita (AR – Augmented Reality) je pojem označující smísení skutečného světa s virtuálními prvky v reálném čase. Narozdíl od virtuální reality, založené na principu nahrazení skutečného světa světem počítačovým, se okolní využívá jako nosné médium virtuálních prvků.

3.1 Rozdělení

Obecně se rozlišuje několik druhů AR, které se liší rozdílným způsobem umisťování virtuálních prvků a druhů senzorů, které jsou pro získání těchto informací použity:

- **Rozšířená realita využívající geolokaci a orientaci** – Tento typ využívá umístění zařízení a jeho orientaci v prostoru k zjištění umístění tzv. POI[1] nebo jiných objektů. Ty pak vykreslí na překryvnou vrstvu většinou ve formě značek znázorňujících umístění určitých reálných těles ve skutečném světě. Tento způsob se často využívá pro aplikace zjednodušující orientaci v prostoru (obrázek 3.3).
- **Počítačové vidění** – U počítačového vidění je princip založen na vytvoření virtuálního objektu na místě již existujícího reálného objektu, přičemž vůbec nemusí být brán zřetel na umístění nebo orientaci zařízení. Ve většině příkladů se jedná o jenoznačně identifikovatelný vzor, kterým může být např. čárový kód, QR kód[2] a podobně (obrázek 3.1).

3.2 Historie

Přestože se teprve v nedávné době dočkala rozšířená realita masového rozšíření mezi běžné uživatele, její principy a návrhy byly rozvíjeny již v začátcích vývoje počítačových věd. Mnoho jejích základních konceptů bylo zobrazeno např. ve filmech The Terminator (1984) nebo RoboCop (1987).

Dnešní pojem „Augmented Reality“ byl poprvé užit v roce 1990 Tomem Caudellem, zakladatelem společnosti Boeing, když on a jeho spolupracovníci na počátku devadesátých let pracovali na vývoji náhlavního displeje, umožňujícího technikům sestavovat složité rozvody za pomoci AR vykreslované na desku plošných spojů, kterou bylo třeba k systému

[1] Points of Interest – body zájmu

[2] Dvourozměrný, strojově čitelný kód užívaný k uložení malého množství dat na tisknutelné médium.

Obrázek 3.1: Technologie počítačového vidění

připojit. Převedením mnoha složitých diagramů do elektronické podoby byl tento proces značně zjednodušen [9].

3.3 Použití

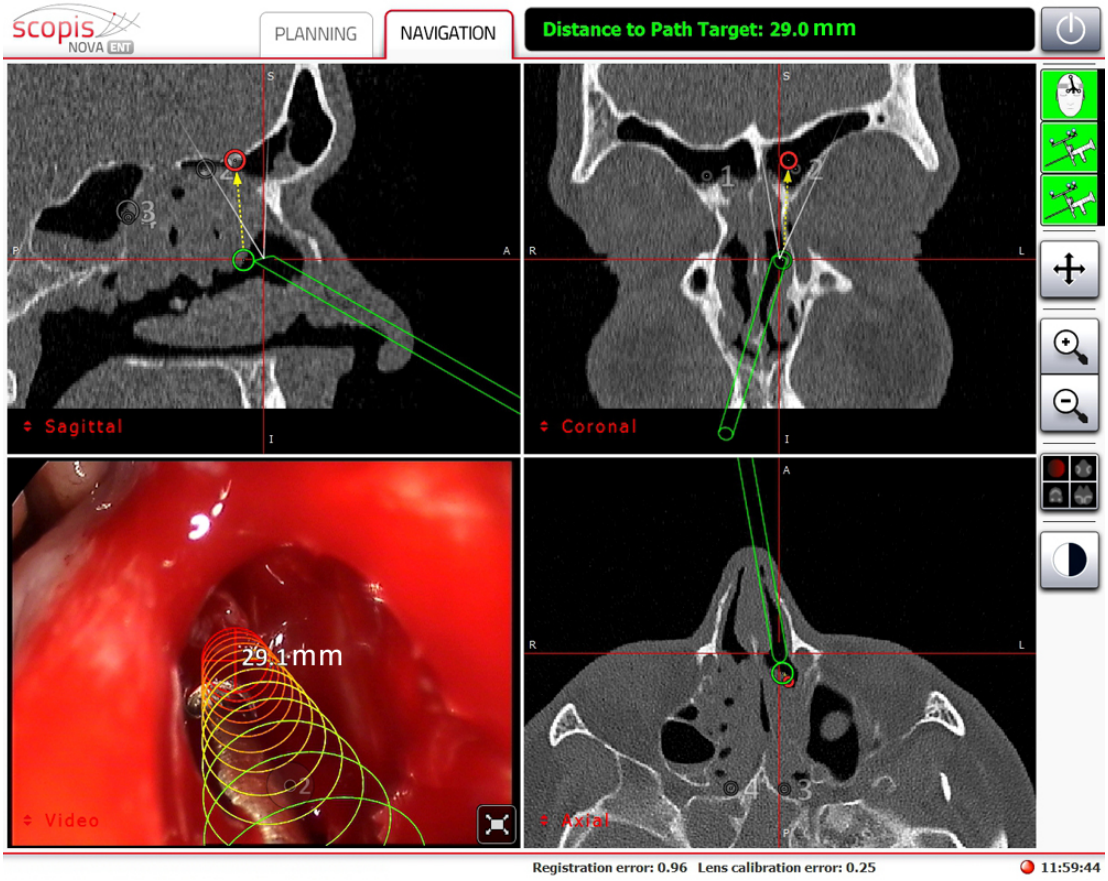
Zařízení schopna zobrazit rozšířenou realitu jsou cenově dostupná, proto rozvoji tohoto odvětví aplikací nic nestojí v cestě. Někteří výrobci mobilních zařízení naopak podporují jejich vývoj vytvářením hardwaru, který rozšířenou realitu přímo podporuje (Google glasses, přední skla automobilů vyšší třídy atd.).

3.3.1 Reklama

Dovolím si tvrdit, že nejrozšířenějším odvětvím použití rozšířené reality je v současné době reklama. Ať už se jedná o vizualizaci stavebnic LEGO®nebo animace použité v Guinnessově knize rekordů, je tento způsob reklamy velmi moderní a pro spotřebitele atraktivní.

3.3.2 Zdravotnictví

Bezesporu nejužitečnějším odvětvím využití rozšířené reality je zdravotnictví. Zprostředkovává lékařům průhled do lidského těla, aniž by bylo nutno přistoupit k mnohdy destruktivním zákrokům. Může také napomáhat při vpichu jehly, lékař může vidět „skrz“ tělo a vést vpich přesněji. (ilustrační obrázek využití AR ve zdravotnictví, viz obrázek 3.2)



Obrázek 3.2: Chirurgická endoskopie využívající principů rozšířené reality [1]

Obrázek 3.3: Rozšířená realita používající geolokaci a orientaci [12]

3.3.3 Aplikace usnadňující orientaci

Tento typ aplikací umožňuje uživateli snadnou orientaci v prostoru. Využívá principu rozmístění virtuálních značek do průhledu fotoaparátu. Tyto značky mohou nést textová či obrazová data a slouží uživateli k identifikaci a získání informací o místě, na kterém se značka nachází (viz obrázek 3.3).

3.3.4 Hry

Stejně jako drtivá většina cenově dostupných technologií na trhu i rozšířená realita si našla své místo v herním průmyslu. Poskytuje hráčům zcela novou herní platformu, mohou hrát virtuální postavy v reálném světě. Hry využívající těchto principů lze již v dnešní době nalézt téměř na všech mobilních operačních systémech.

Kapitola 4

Návrh aplikace

4.1 Název aplikace

Můj program jsem se rozhodl nazvat „Architectures“. Jedná se o kombinaci anglických pojmů Architecture Resources (prostředky pro architekturu), tento název jsem zvolil s ohledem na funkci a cílovou skupinu uživatelů.

4.2 Funkce aplikace

Aplikace je primárně zaměřená na vizualizaci trojrozměrných modelů budov. Jako formát pro jejich ukládání jsem zvolil otevřený formát `.obj`, právě díky jeho otevřenosti, jednoduchosti a dostupnosti. Uživatel má možnost uložit si své vlastní modely v tomto formátu do svého zařízení a vložit je do virtuální scény.

Data o jednotlivých výskytech modelu ve scéně se ukládají do databáze, je tedy možno jeden model využít ve scéně vícekrát. Díky jednoduchému editoru, který je součástí aplikace, lze různým výskytům modelu ve scéně nastavit odlišnou velikost a orientaci.

4.3 Uživatelské rozhraní

Aplikaci jsem navrhoval s ohledem na jednoduchost a intuitivnost ovládání, ale vzhledem ke skutečnosti, že vytváření 3D modelů budov vyžaduje pokročilou počítačovou gramotnost, nemusí na něj být kladen takový důraz.

Uživatelské rozhraní se skládá ze dvou základních prvků. Jedním z nich je zobrazení scény s prvky rozšířené reality, v druhém může uživatel přidávat, upravovat, nebo mazat jednotlivé instance modelů, respektive zapínat a vypínat jejich viditelnost na scéně. Hrubý nástin uživatelského rozhraní je možno vidět na obrázku 4.1.

4.3.1 Zobrazení scény

Je navržen jako pohled bez aktivních ovládacích prvků. Jeho inicializace může trvat poměrně dlouhou dobu, probíhá při ní totiž načítání všech viditelných modelů ze souborů.

4.3.2 Přidávání a editace prvků scény

Jak jsem již zmiňoval výše, součástí aplikace je i jednoduchý 3D editor, který umožňuje upravit rotaci a velikost modelu předtím, než je přidán do scény. To může částečně kompen-

Obrázek 4.1: Návrh uživatelského rozhraní – zobrazení virtuální scény (vlevo) a správa budov(vpravo)

zovat rozdíl měřítka mřížky 3D editoru a aplikace (při exportu modelu v editoru blender je však měřítko zachováno).

Důležitou součástí přidávání prvků je také správné určení jeho globálních souřadnic pro přesné umístění modelu ve scéně. Souřadnice se dají specifikovat dvěma způsoby – ručně, nebo výběrem pozice na mapě. Protože však Google Maps Android API v2 neposkytuje informace o nadmořské výšce, je potřeba získávat tyto data nadmořskou výšku v zadaném bodě jinak (viz sekce 5.2.3).

4.4 Cílová skupina

Aplikace je konstruována primárně pro architekty, kteří chtějí jednoduchým způsobem ověřit, zda jejich návrhy nenarušují svým vzhledem okolní krajinu. Vzhledem ke skutečnosti, že tato skupina není příliš početná, jedná se spíše o aplikaci s úzkým profilem uživatelů.

Stále se rozšiřující skupina grafiků, kteří se podílejí na vytváření 3D budov pro aplikaci Google Earth by však také pro tuto aplikaci mohla nalézt uplatnění. Google Sketchup, který je k tomuto účelu primárně používán, totiž export do formátu `.obj` podporuje.

Kapitola 5

Implementace

5.1 Senzory

K určení orientace zařízení v prostoru je nutno použít dvou typů senzorů. Akcelerometr k určení směru působení gravitace na zařízení a magnetometru zjišťujícího, kterým směrem leží magnetické póly Země. Z těchto dvou informací lze získat poměrně přesné informace o natočení zařízení v reálné scéně, které jsou následně použity k nastavení rotační matice kamery.

5.1.1 Filtrování dat ze senzorů

Protože jsou data získávaná ze senzoru ovlivněny výrazným šumem, je bezpodmínečně nutné je před použitím filtrovat. Výběrem nejlepšího filtru pro omezení rozptylu hodnot můžeme diametrálně zlepšit výsledný dojem z aplikace. Při vývoji jsem testoval a porovnával z hlediska uživatelského pocitu následující typy filtrů:

- **Prahový filtr** – Nová hodnota ze senzoru je použita jen tehdy, když se od předchozí hodnoty liší o více, než je hodnota prahu. Tímto způsobem lze efektivně odstínit vibrace scény při klidovém stavu zařízení, není však vhodný pro filtraci při změnách hodnot. Ukázka funkce filtru uvedena na obrázku 5.1(a).

- **Aritmetický průměr** – Vypočítá aritmetický průměr ze současné a několika minulých hodnot ($\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$) [23]. Při vyšším počtu vzorků, ze kterých je hodnota počítána, způsobuje zpomalené reakce systému na pohyb, není proto zcela vhodný. Při kombinaci aritmetického průměru s prahovým filtrem jsem však dosáhl uspokojivých výsledků. Jediným parametrem, který se dá u filtru tohoto typu změnit, je počet předchozích hodnot, ze kterých je aritmetický průměr počítán. Ukázka funkce filtru uvedena na obrázku 5.1(b).

- **Dolní propusť** – Jedná se o implementaci elektronického filtru, který je konstruován tak, aby propouštěl pouze nižší frekvence [21]. Zjednodušeně se dá říci, že váha vzorku se zvyšuje s uplynulým časem. Tento druh filtru má velmi dobré výsledky a je použit i v některých AR frameworcích. Implementace dolní propusti, kterou jsem testoval, však byla frekvenčně závislá a proto nebyla pro můj projekt příliš vhodná. Jako proměnný parametr přijímá takzvaný „smoothing“. Ten určuje míru vyhlazení filtru. Ukázka funkce filtru uvedena na obrázku 5.2(a).

- **Exponenciální klouzavý průměr** – Nejlepší výsledky filtrování vykazuje exponenciální klouzavý průměr. Jedná se o vážený průměr, váha jednotlivých vzorků klesá exponenciálně s pořadím (teoreticky nikdy nedosáhne nuly). Tento typ filtru je použit např. v aplikaci Stardroid (Google Sky Map). Jeho chování lze ovlivnit parametrem α, který s narůstající hodnotou snižuje strmost exponenciální váhové křivky. Ukázka funkce filtru uvedena na obrázku 5.2(b).

5.1.2 Získání rotační matice

K získání rotační matice pro nastavení orientace kamery ve scéně jsem využil nativní funkce android frameworku `getRotationMatrix()`. Ta jako argumenty přijímá hodnoty akcelerometru a magnetometru, jejím výsledkem je pak rotační matice zařízení, která je přímo aplikovatelná na kameru ve scéně.

Protože souřadný systém výsledné rotační matice není uzpůsoben pro použití v landscape režimu (kdy je zařízení orientováno na šířku), bylo nutné přemapovat systém souřadnic. K tomu jsem využil funkci `remapCoordinateSystem()` a převedl tak systém souřadnic z globální osy Y na osu zařízení $-X$.

5.2 Geolokace

V systémech android rozeznáváme dva základní přístupy k získávání pozice zařízení – pomocí GPS modulu, nebo takzvané „Určování polohy Google“. Z důvodu podpory co největšího spektra zařízení jsem v aplikaci použil oba tyto přístupy. Upřednostňována je ale kvůli nesporným výhodám metoda získávání pomocí GPS, v případě nedostupnosti signálu nebo jinému problému je zdroj polohy automaticky přepnut na druhý způsob.

5.2.1 Lokace pomocí GPS

Lokace pomocí modulu GPS má oproti druhé alternativě zásadní výhodu – jako součást výsledku vrací i informace o aktuální nadmořské výšce zařízení[1], která je pro správnou funkci aplikace klíčová.

5.2.2 Lokace pomocí síťových dat

Získává data z databáze s informacemi o vlastnících IP adres. Přestože je tato metoda velmi rychlá, její přesnost je velice nízká a pravdivost poskytnutých dat zpochybnitelná. Pokud se totiž zařízení nachází za překladačem adres NAT, nezíská tato metoda pozici koncového zařízení, ale pozici daného překladače.

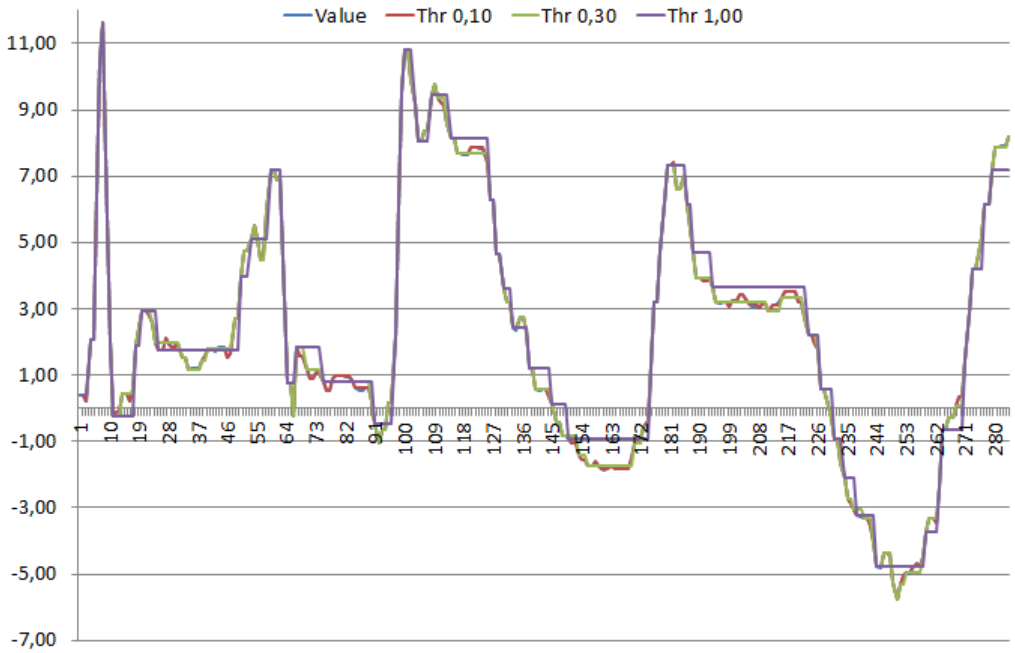
Protože tento systém nedodává informace o nadmořské výšce zařízení, je tento údaj získáván pomocí Google Elevation API (viz 5.2.3).

5.2.3 Google Elevation API

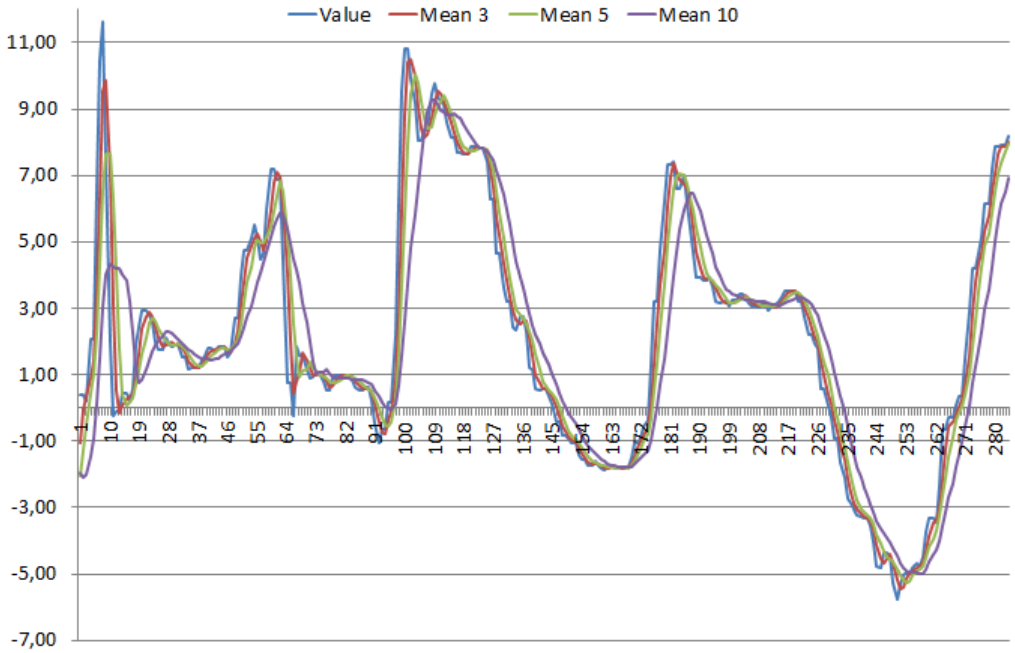
Jedná se o webovou službu, která poskytuje informace o nadmořské výšce povrchu země v zadaném bodě. Parametry jsou službě zadány metodou GET[2], výsledek je dodáván buď

[1] Pokud je zařízení v nestíněném dosahu čtyř nebo více satelitů

[2] Jako součást adresy požadavku

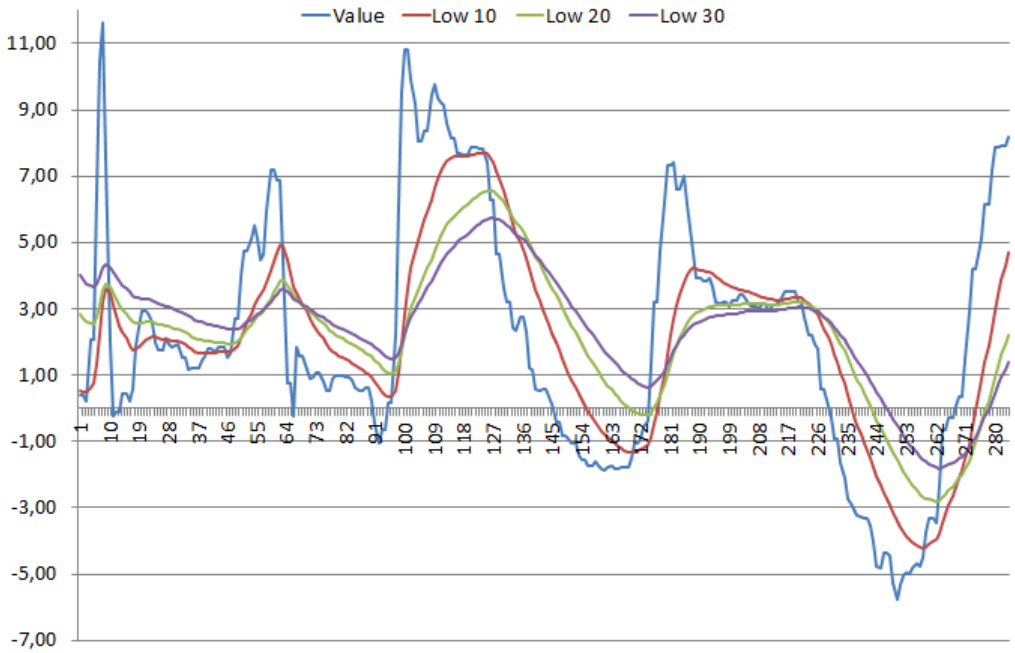


(a) Prahový filtr

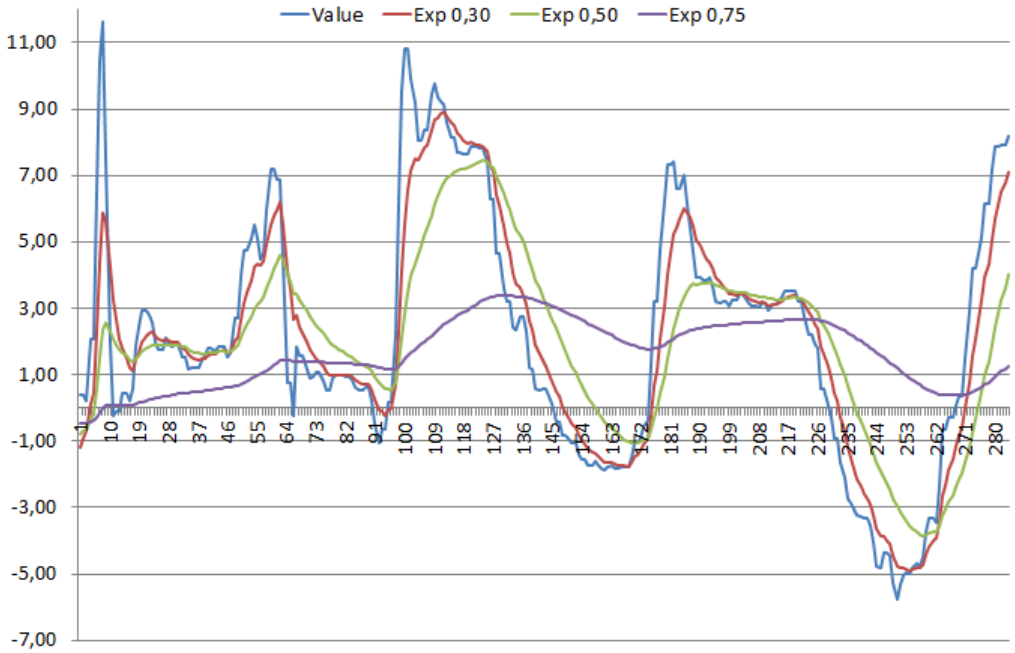


(b) Aritmetický průměr

Obrázek 5.1: Ukázka prahového filtru a aritmetického průměru.

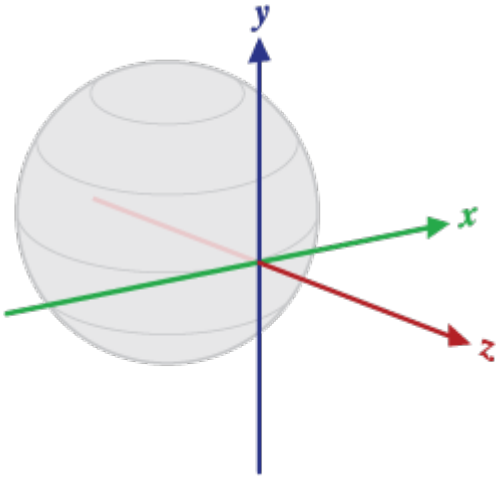


(a) Dolní propusť



(b) Exponenciální klouzavý průměr

Obrázek 5.2: Ukázka dolní propusti a exponenciálního klouzavého průměru



Obrázek 5.3: Systém souřadnic rotační matice zařízení

ve formátu XML, nebo JSON. Protože android frameworku obsahuje velice jednoduchou a efektivní implementaci XML parseru, zvolil jsem tento formát.

Požadavky lze tomuto rozhraní odesílat jak pomocí protokolu HTTP[3], tak i chráněnou verzí HTTPS. S přihlédnutím ke skutečnosti, že odesílaná data mohou obsahovat citlivá data uživatele jsem se rozhodl použít chráněnou verzi protokolu.

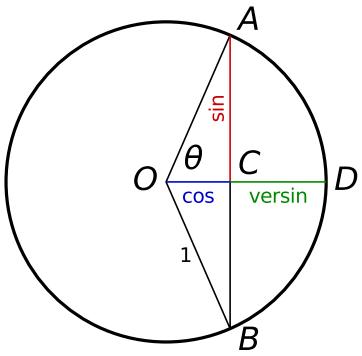
Ukázka odpovědi z Google Elevation API (formát XML):

```
<?xml version="1.0" encoding="UTF-8"?>
<ElevationResponse>
 <status>OK</status>
 <result>
  <location>
   <lat>40.7147280</lat>
   <lng>-73.9986720</lng>
  </location>
  <elevation>8.8836946</elevation>
  <resolution>76.3516159</resolution>
 </result>
</ElevationResponse>
```

V odpovědi jsou obsaženy hodnoty z požadavku (zeměpisná šířka a délka) a další informace:

- **Status** – Stavový kód požadavku, nabývá hodnoty *OK* v případě úspěšného vykonání požadavku, jiných hodnot v případě neúspěchu (kompletní seznam stavových kódů je k dispozici oficiálních stránkách Google Elevation API [3])
- **Elevation** – Výsledná nadmořská výška v metrech
- **Resolution** – Označuje maximální vzdálenost bodů, ze kterých je interpolována nadmořská výška (v případě, že přesná informace pro zadaný bod neexistuje)

[3]Hypertext Transfer Protocol



Obrázek 5.4: Sinus, kosinus a versin úhlu na jednotkové kružnici

5.2.4 Souřadnicové systémy

Informace o umístění budov ve světě i aktuální pozice zařízení jsou získávány a uchovávány jako zeměpisná šířka a zeměpisná délka v souřadincovém systému země. Před použitím těchto souřadnic je nutné převést je do kartézského souřadného systému, aby bylo možno je použít v OpenGL.

Vzhledem k tomu, že potřebuji získat relativní pozici modelu vzhledem ke kameře, zvolil jsem následující postup:

1. Získání vzdálenosti mezi pozicí kamery a modelem
2. Získání úhlu, který svírá osa X (mířící na sever) s vektorem směřujícím k modelu (takzvaný „bearing“)
3. Převod z polárních souřadnic na kartézské

Pro získání vzdáleností jsem použil metodu zvanou „Haversine formula“, pomocí niž lze vypočítat vzdálenost dvou bodů na sférickém modelu země. Nepočítá tedy se zploštěním země na pólech, její maximální chyba ale činí přibližně 0.3%, což je přesnost pro tuto aplikaci více než dostačující. Pomocí této metody lze vzdálenost mezi dvěma body získat následujícím způsobem [25], [13]:

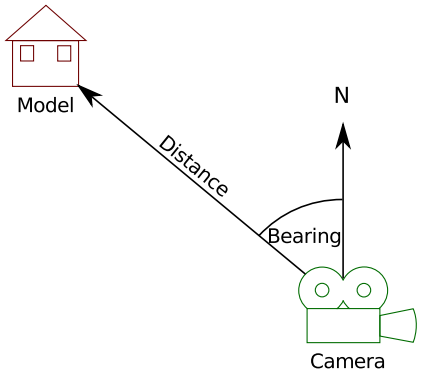
$$\operatorname{haversin}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2} \tag{5.1}$$

$$\operatorname{haversin}\left(\frac{d}{r}\right) = \operatorname{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\operatorname{haversin}(\lambda_2 - \lambda_1) \tag{5.2}$$

$$d = r\,\operatorname{haversin}^{-1}(h) = 2r\arcsin\left(\sqrt{h}\right) \tag{5.3}$$

Haversine je definován rovnicí 5.1 a rovná se polovině hodnoty versinu. Na obrázku 5.4 lze vidět zobrazení versinu na jednotkové kružnici.

Pomocí rovnice 5.2 vypočítám haversin podílu vzdálenosti a poloměru země, ze kterého následně podle rovnice 5.3 získám vzdálenost dvou bodů určených zeměpisnými souřadnicemi (v metrech).



Obrázek 5.5: Ilustrace výsledku získaného z výpočtu haversine a bearingu

Tento postup ovšem vypočítá pouze vzdálenost – absolutní hodnotu směru umístění modelu ve scéně. Pro správnou funkci rozšířené reality je ale potřeba určit, ve kterém směru se budova nachází a vhodně změnit znaménka výsledné vzdálenosti.

K tomuto účelu jsem výpočítal hodnotu bearingu, úhlu, který určuje odklon směrového vektoru k modelu od severu. Výpočet se reliazuje pomocí vztahu [13]:

$$\theta = \operatorname{atan2}(\sin(\lambda_2 - \lambda_1)\cos(\phi_2), \cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\lambda_2 - \lambda_1)) \tag{5.4}$$

Funkce atan2 je goniometrická funkce arcustangens s dvěmi parametry. Účel použití dvou argumentů je získání informací o znaménkách vstupu, aby bylo možné získat správný kvadrant vypočítaného úhlu, což při použití jednoargumentové varianty není možné.

Hodnoty funkce pro jednotlivé vstupy jsou uvedeny v rovinci 5.5 [26].

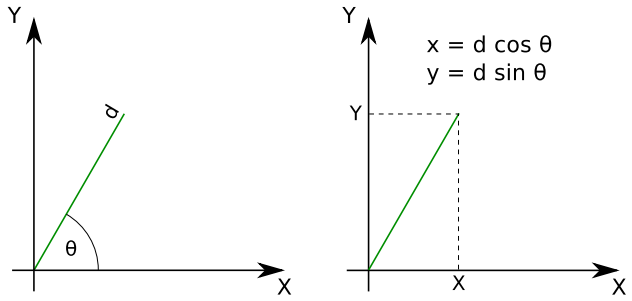
$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases} \tag{5.5}$$

Po získání bearingu je k dispozici úplná informace o umístění v polárním souřadnicovém systému. Vztah sloužící pro převedení do kartézské soustavy lze vidět na obrázku 5.6.

5.3 Ukládání dat

5.3.1 Soubory modelů

Tato aplikace podporuje pouze modely ve formátu `.obj`. Vzhledem k omezení Rajawali frameworku, popsaném v oddíle 2.5.1, musí model splňovat následující pravidla:



Obrázek 5.6: Ukázka rozdílu polárního (vlevo) a kartézského (vpravo) souřadnicového systému

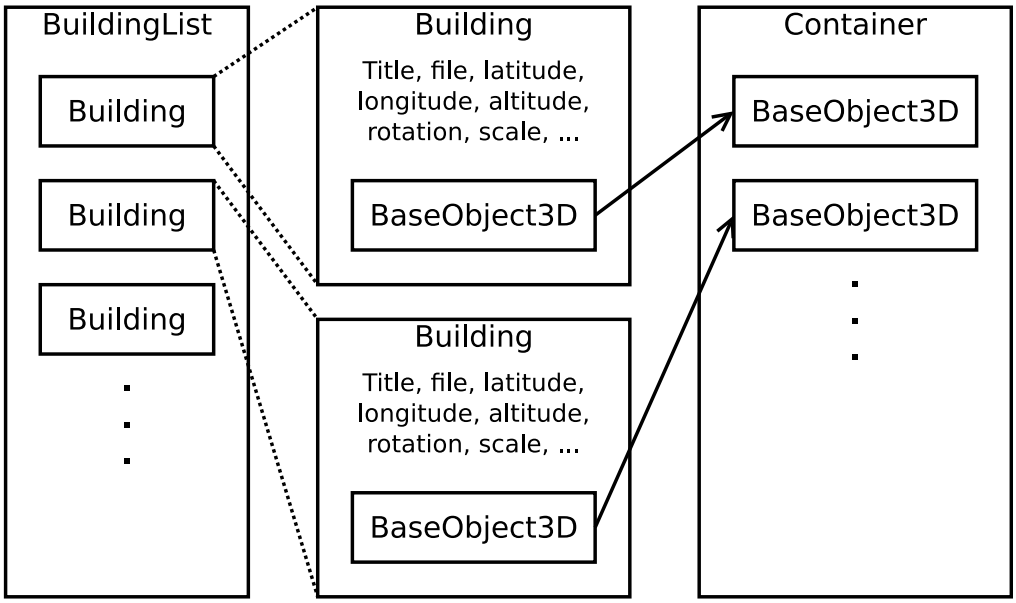
- **Normály** – Model musí být exportován s normálami. V případě, že normály nejsou specifikovány, nemůže být model správně zpracován a může dojít k chybě aplikace. Toto chování způsobují interní funkce frameworku.
- **Jmenné konvence** – Protože je framework primárně určen k parsování souborů z resources[4], dodržuje i v případě načítání z SD karty stejné jmenné konvence. Všechny soubory modelu tedy musí mít příponu oddělenou podtržítkem (_) místo tečkou (.).
- **Struktura povrchu** – Pro správné zobrazení modelu je nutné, aby se jejich povrch skládal pouze z trojúhelníků. Parser zpracovává pouze první tři informace o vertexech pro každý polygon a zbytek ignoruje. Nedodržení tohoto pravidla způsobuje při zobrazení „díry“ v modelu.

5.3.2 Databáze SQLite

Protože se ke každému modelu váže množina metadat, jako jsou např. pozice ve scéně, rotace nebo název, musí se tyto informace uchovávat ve vhodném úložišti. Vzhledem k povaze dat jsem zvolil zabudované databázové úložiště SQLite. Databázi tvoří jediná tabulka s následující strukturou:

Sloupec	Popis	Příklad
title	Název budovy	Návrh 1
file	Cesta k modelu	house_obj
latitude	Zeměpisná šířka	12.2135131
longitude	Zeměpisná délka	-10.5468451
elevation	Nadmořská výška	232.486
height	Korekce výšky modelu	1.3
rotation	Rotace modelu	90
scale	Měřítko modelu	0.3
visible	Viditelný ve scéně	1

[4]Soubory obsahující multimédia, XML kódy a jiné zdroje, které jsou předzpracovány v době překladu a připojeny k aplikaci.



Obrázek 5.7: Správa modelů scény. Metainformace jsou uloženy v poli `BuildingList`, ze kterého vedou odkazy do kontejneru (container). Ten je předán rendereru ke zobrazení.

5.4 Nastavení scény a zobrazení modelu budovy

Pro zjednodušení a sjednocení vytváření vlastních rendererů jsem vytvořil abstraktní třídu `ARenderer`, ze které jsou zděděny všechny renderery v aplikaci. Tato třída rozšiřuje funkčnost původní třídy `RajawaliRenderer` o implementaci následujících metod:

- Přidává callback volaný při inicializaci scény (ve skutečnosti je funkce `onInitComplete()` volána před prvním vykreslením snímku ve funkci `onDrawFrame()`).
- Metoda `setSurfaceTransparency()` umožňuje zapnout nebo vypnout průhlednost pozadí rendereru. Tato metoda je nutná při překrytí průhledu kamery rendererem.
- Definuje metody pro výchozí nastavení scény, jako jsou např. `set3PointLightning()`, `drawAxis()`, `drawGrid()`.
- Přidává podporu pro zobrazení loadingu v nadřazené aktivitě. Metody `showLoading()` a `hideLoading()` umožňují skrýt, respektive zobrazit načítací dialog. Ke správné funkčnosti však musí mít Aktivita implementující rozhraní ILoading.
- Mění způsob přidání modelů do scény tak, aby bylo možné provádět transformace celé scény. Místo `List<BaseObject3D>` se objekty ukládají do jednoho rodičovského `BaseObject3D` objektu, se kterým se následně dají provádět transformace.
- Umožňuje přidávat modely společně s metainformacemi. K tomu slouží třída zvaná `BuildingList`

Aplikace obsahuje dva základní renderery – `PreviewRenderer` sloužící k zobrazení náhledu 3D modelu před přidáním do scény, a `ViewSceneRenderer`, který má za úkol vykreslit všechny viditelné modely z databáze do virtuálního prostředí kolem uživatele.

V případě prvního zmíněného existuje pouze jeden model, který je do scény přidán při jeho inicializaci. V případě, že uživatel zvolí z nabídky jiný soubor, původní model je odstraněn ze scény a proběhne načítání a vkládání modelu nového.

Složitější situace nastává v případě `ViewSceneRenderer`u. Při inicializaci jsou do scény vloženy všechny viditelné modely z databáze, kvůli změně pozice zařízení je však nutno k nim později přistupovat. K tomuto účelu slouží instance zmíněné třídy `BuildingList`. Ilustraci řešení je možno vidět na obrázku 5.7

5.4.1 Aplikace rotační matice

Tento krok byl díky použitému frameworku jednoduchou záležitostí. Třída `Camera` z frameworku Rajawali (viz sekce 2.5.1) totiž implementuje metodu `setRotationMatrix()`, která nastaví rotaci kamery z předané rotační matice. Tímto způsobem bylo možno použít matici získanou funkcí `getRotationMatrix` přímo.

Kapitola 6

Testování

V průběhu vývoje jsem pracoval na různých typech zařízení pro dosažení maximální kompatibilty a podpory rozdílných typů hardwaru.

Jedinou testovanou hodnotou, jejíž výsledky mohly být objektivně porovnány, je doba parsování modelu z SD karty. Tato operace je zároveň nejnáročnější na výpočetní čas procesoru a délka načítání scény je z drtivé většiny tvořena právě načítáním viditelných modelů ze souboru a jejich následným parsováním.

6.1 Sony xperia mini pro

6.1.1 Technické specifikace

Parametr	Hodnota
Paměť RAM	512MB
Velikost displaye	3 palce
Rozlišení displaye	480 x 320
Typ procesoru	ARMv7 Processor rev 2 (v71)
Frekvence procesoru	1GHz
Počet jader	1
Operační systém	CyanogenMod custom ROM
Verze systému	4.04: Ice CreamSandwich
Verze jádra	2.6.32.9-FXP

6.1.2 Naměřené hodnoty

Model	Počet polygonů	Velikost souboru	Čas parsování
house_obj	2 507	190 689 B	**2.748 s**
atst_obj	6 434	497 189 B	**8.655 s**
box_obj	49 152	2 140 204 B	**37.615 s**

6.2 Asus Transformer

6.2.1 Technické specifikace

Parametr	Hodnota
Paměť RAM	1GB
Velikost displaye	10.1 palce
Rozlišení displaye	1280 x 800
Typ procesoru	NVIDIA® Tegra™ 2
Frekvence procesoru	1GHz
Počet jader	2
Operační systém	Android
Verze systému	4.03: Ice CreamSandwich
Verze jádra	2.6.39.4

6.2.2 Naměřené hodnoty

Model	Počet polygonů	Velikost souboru	Čas parsování
house_obj	2 507	190 689 B	**0.94 s**
atst_obj	6 434	497 189 B	**3.24 s**
box_obj	49 152	2 140 204 B	**15.071 s**

Kapitola 7

Závěr a možnosti rozšíření

Cílem této aplikace bylo zprostředkovat uživateli digitálně rozšířený obraz poskytující ilustraci zasazení budovy do reálného světa.

Při vytváření této aplikace jsem se postupně seznámil s programovacím jazykem Java, operačním systémem Android a principem implementace systémů rozšířené reality. Vývoj aplikace jsem prováděl na různých zařízeních s různou hardwarovou výbavou a snažil jsem se, aby všechna zařízení výslednou aplikaci podporovala.

V průběhu jsem využil různých knihoven sloužících pro vykreslování scény a přístup k databázi. To mi umožnilo soustředit se na problematiku rozšířené reality a nezabývat se elementárními problémy.

Aplikce splňuje zadání dle očekávání. Umožňuje zobrazení rozšířené reality, jednoduchou a přehlednou správu budov a podporuje všechna testovaná zařízení se systémem Android 4.0 a vyšší, které mají k dispozici zadní kameru. Na zařízeních s nižším výkonem procesoru funguje aplikace pomaleji (zvláště kvůli vysokým procesorovým nárokům parsování modelů), její funkčnost však není nikterak omezena.

Přestože aplikace svým způsobem plní svůj účel, vhodným rozšířením by bylo možné její funkčnost vylepšit. Jednou z možností je přidání podpory textur modelů, úprava osvětlovacího modelu, aby lépe reflektoval slunce v reálném světě, nebo třeba sychnronizace snímkování rendereru a fotoaparátu. Tyto změny by pak měly příznivý vliv na věrnost zobrazení rozšířené reality.

V případě většího rozšíření aplikace by zřejmě bylo nutné zefektivnit parsování modelů z SD karty, např. pomocí serializace modelu, nebo zavedení vlastního formátu souborů podporovaného konkrétním programem pro práci s 3D grafikou (napíklad `c4d`, `.3ds`). Tím by bylo také možné přidat do scény modely obsahující animaci.

Použití Google Elevation API je také poměrně problematické. Tento zdroj nadmořské výšky je totiž limitován počtem přístupů a navíc vyžaduje aktivní připojení k internetu. Bylo by tedy vhodné získávat data z rychlejšího, spolehlivějšího a neomezeného zdroje. Tím by bylo rovněž možné implementovat ořezávání spodní strany modelu podle terénu, ve kterém se nachází a stínění modelu terénem, což by u současného zdroje dat bylo problematické.

Literatura

[1] Bichlmeier, C.: Endoscope Augmentation by Scopis GmbH. [online], 2013-04-09 [cit. 2013-04-17].
URL http://medicalaugmentedreality.com/2013/04/endoscope-augmentation-by-scopis-gmbh/

[2] Burnette, E.: *Hello, Android: Introducing Google's Mobile Development Platform.* Raleigh, N.C.: Pragmatic Programmers, 2010, ISBN 978-193-4356-562.

[3] Inc., G.: The Google Elevation API. [online], 2013-03-14 [cit. 2013-04-16].
URL https://developers.google.com/maps/documentation/elevation/

[4] Inc., G.: Activity. [online], 2013-05-01 [cit. 2013-05-08].
URL http://developer.android.com/reference/android/app/Activity.html

[5] Inc., G.: Fragment. [online], 2013-05-01 [cit. 2013-05-08].
URL http://developer.android.com/reference/android/app/Fragment.html

[6] Inc., G.: Sensors Overview. [online], [cit. 2013-04-16].
URL
http://developer.android.com/guide/topics/sensors/sensors_overview.html

[7] Inc., G.: The AndroidManifest.xml File. [online], [cit. 2013-05-08].
URL
http://developer.android.com/guide/topics/manifest/manifest-intro.html

[8] Inc., G.: Loaders. [online], [cit. 2013-05-08].
URL http://developer.android.com/guide/components/loaders.html

[9] Mullen, T. K.: *Prototyping augmented reality.* Indianapolis, IN: Wiley Pubishing, Inc., 2011, ISBN 11-180-3663-8.

[10] Munshi, A.: *The OpenGL ES 2.0 programming guide.* Upper Saddle River: Addison-Wesley, 2009, ISBN 978-0-321-50279-7.

[11] Nohejl, P.: *Rozšířená realita pro platformu android.* Diplomová práce, FIT VUT v Brně, Brno, 2011.

[12] Prindle, D.: Best Augmented Reality Apps. [online], 2013-01-29 [cit. 2013-04-16].
URL http://www.digitaltrends.com/mobile/best-augmented-reality-apps/

[13] Veness, C.: Calculate distance, bearing and more between Latitude/Longitude points. [online], [cit. 2013-04-16].
URL http://www.movable-type.co.uk/scripts/latlong.html

[14] Wikipedia: OpenGL ES. [online], 2013-03-15 [cit. 2013-04-15].
URL http://en.wikipedia.org/wiki/OpenGL_ES

[15] Wikipedia: Shader. [online], 2013-03-18 [cit. 2013-04-16].
URL http://en.wikipedia.org/wiki/Shader

[16] Wikipedia: Mipmap. [online], 2013-04-10 [cit. 2013-04-16].
URL http://en.wikipedia.org/wiki/Mipmap

[17] Wikipedia: Android (operating system). [online], 2013-04-12 [cit. 2013-04-14].
URL http://en.wikipedia.org/wiki/Android_(operating_system)

[18] Wikipedia: Embedded system. [online], 2013-04-14 [cit. 2013-04-15].
URL http://en.wikipedia.org/wiki/Embedded_systems

[19] Wikipedia: Application programming interface. [online], 2013-04-15 [cit. 2013-04-16].
URL http://en.wikipedia.org/wiki/Application_programming_interface

[20] Wikipedia: Global Positioning System. [online], 2013-04-15 [cit. 2013-04-16].
URL http://en.wikipedia.org/wiki/Global_Positioning_System

[21] Wikipedia: Low-pass filter. [online], 2013-04-18 [cit. 2013-05-03].
URL http://en.wikipedia.org/wiki/Low-pass_filter

[22] Wikipedia: XML. [online], 2013-04-27 [cit. 2013-05-08].
URL http://en.wikipedia.org/wiki/XML

[23] Wikipedia: Mean. [online], 2013-04-28 [cit. 2013-05-03].
URL http://en.wikipedia.org/wiki/Mean

[24] Wikipedia: Java (programming language). [online], 2013-05-03 [cit. 2013-05-08].
URL http://en.wikipedia.org/wiki/Java_%28programming_language%29

[25] Wikipedia: Haversine formula. [online], 2013-05-06 [cit. 2013-05-08].
URL http://en.wikipedia.org/wiki/Haversine_formula

[26] Wikipedia: atan2. [online], 2013-05-06 [cit. 2013-05-09].
URL http://en.wikipedia.org/wiki/Atan2

Příloha A

Obsah CD

- `/apk` – Soubor `.apk` připravený k instalaci
- `/doc` – Vygenerovaná dokumentace k programu
- `/excel` – Implementace rovnic a grafy v excelu
- `/models` – Ukázkové modely
- `/src` – Zdrojové kódy aplikace
- `README` – Soubor popisující postup instalace aplikace

Příloha B

Plakát