ROZŠÍŘENÁ REALITA PRO PLATFORMU ANDROID

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
Bc. PETR NOHEJL

BRNO 2011
ROZŠÍŘENÁ REALITA PRO PLATFORMU ANDROID
AUGMENTED REALITY FOR ANDROID PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE     Bc. PETR NOHEJL
AUTHOR

VEDOUCÍ PRÁCE     Ing. ALEŠ LÁNÍK
SUPERVISOR

BRNO 2011
Abstrakt
Tato diplomová práce se zabývá návrhem a realizací systému rozšířené reality na mobilní platformě Google Android s využitím polohových senzorů. Aplikace slouží jako navigace a zobrazuje geografické body zájmu. Práce se věnuje problematice rozšířené reality na mobilních zařízeních, návrhem vlastního frameworku a poté zmíňuje detaile implementace vybraných problémů. Nakonec je popsán způsob testování a zhodnoceny výsledky práce.

Abstract
This thesis describes design and implementation of augmented reality system for Android platform using location sensors. The application serves as a navigation and displays geographical points of interest. Thesis deals with augmented reality on mobile devices, describes design of own framework and mentions the details about implementation of selected problems. Finally, the results are evaluated.

Klíčová slova
rozšířená realita, Google Android, smartphone, senzory, akcelerometr, kompas, GPS, OpenGL ES

Keywords
augmented reality, Google Android, smartphone, sensors, accelerometer, compass, GPS, OpenGL ES

Citace
Nohejl Petr: Rozšířená realita pro platformu Android, diplomová práce, Brno, FIT VUT v Brně, 2011
Rozšířená realita pro platformu Android

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Aleše Láníka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

........................................
Petr Nohejl
24. 5. 2011

Poděkování

Dovoluji si poděkovat vedoucímu této práce Ing. Aleši Láníkovi za odbornou pomoc a cenné rady. Děkuji také svým rodičům za podporu během studia.
### Obsah

1. Úvod .......................................................... 3
2. Rozšířená realita ......................................... 4
   2.1 Definice rozšířené reality .......................... 4
   2.2 Trasování virtuálních objektů ..................... 5
   2.3 Snímání obrazu ........................................ 8
   2.4 Polohové senzory ...................................... 8
   2.5 Zobrazení rozšířené reality ......................... 12
   2.6 Využití rozšířené reality ........................... 13
3. Platforma Google Android ............................. 15
   3.1 Historie .................................................. 15
   3.2 Architektura ............................................ 16
   3.3 Vývoj .................................................... 19
   3.4 Struktura aplikace ..................................... 20
   3.5 Životní cyklus aplikace ............................... 22
   3.6 Uživatelské rozhraní ................................. 23
4. Grafická knihovna OpenGL ES ....................... 24
   4.1 Struktura knihovny ..................................... 24
   4.2 OpenGL ES na platformě Android .................. 26
   4.3 Zobrazení 3D grafiky a transformace ............ 26
5. Návrh systému .............................................. 29
   5.1 Framework rozšířené reality ....................... 29
   5.2 Letecká navigace ..................................... 33
6. Implementace aplikace .................................... 34
   6.1 Struktura aplikace ..................................... 34
   6.2 Zobrazení rozšířené reality ........................ 36
   6.3 Výpočet orientace ..................................... 39
   6.4 Výpočet geolokace .................................... 44
   6.5 Vykreslení bodů zájmu ............................... 45
   6.6 Databáze ................................................ 47
   6.7 Detekce kolizí .......................................... 48
   6.8 Uživatelské rozhraní .................................. 50
7. Vývoj a testování .......................................... 54
   7.1 Ladící nástroje ........................................ 54
   7.2 Unit testy ............................................... 55
7.3 Testování na platformě Android ................................................................. 55
7.4 Testování v terénu .................................................................................. 56
7.5 Porovnání s existujícími programy ....................................................... 57
7.6 Publikace na Android Marketu .............................................................. 58
8 Závěr ............................................................................................................ 59
1 Úvod

Mobilní trh prochází velkými evolučními proměnami. Rychle se rozbíjející mobilní technologie, nová technologická řešení a neustále se zvyšující výkon nabízí nové možnosti uplatnění. Mobilní telefony již neslouží výhradně pro komunikaci, ale jejich využití je mnohem širší. Výrobci se snaží o co největší expanzi tzv. chytrých telefonů a tomu přizpůsobují i aplikace, které by měly být jednoduché, účelné a snadno ovladatelné. Smartphony\(^1\) už nejsou určeny pouze pro technicky založené uživatele, ale snaží se oslovit širokou veřejnost.

Jednou z oblastí, kde dochází k velkému rozvoji a jejíž popularita vzrůstá, je problematika navigace. Většina moderních mobilních zařízení má k dispozici řadu pokročilých funkcí včetně podpory polohových senzorů (např. GPS, akcelerometr, kompas), s jejichž pomocí lze relativně přesně určit aktuální polohu uživatele. Využití těchto technologií lze aplikovat do oblasti rozšířené reality. Jde o pohled do reálného prostředí, který je doplněn prvky virtuální reality. S pomocí videokamery lze mobilní telefon efektivně využít jako přehledový displej s doplněním informací dodaných počítačem.

Tato diplomová práce se zabývá problematikou rozšířené reality na mobilních zařízeních. Snaží se analyzovat možnosti vizualizace rozšířené reality, na základě získaných poznatků specifikovat zadání a nabídnout rámcový návrh řešení problému. Navržený systém by měl realizovat rozšířenou realitu na platformě Google Android s využitím polohových senzorů. Aplikace by měla sloužit jako navigace a zobrazovat geografické body zájmu na displeji telefonu. Celý systém by měl být schopný zobrazit body zájmu zaměřené na letecká data. Základním předpokladem tedy je, aby uměl pracovat s nadmořskou výškou.


Práce si klade za cíl uvést čtenáře do problematiky použití rozšířené reality na mobilních zařízeních, osvětlit mu základní principy realizace rozšířené reality a navést či pomoci při řešení některých problémů, pokud by měl zájem takovou aplikaci vyvijet.

\(^1\) Mobilní telefony s pokročilými funkcemi
2 Rozšířená realita


V této kapitole budou popsány základní principy rozšířené reality důležité pro návrh celého systému. Popíšeme si dva nejčastější způsoby realizace rozšířené reality na mobilních zařízeních. Představíme si hardwarová zařízení a senzory používané pro trasování virtuálních objektů, dále budou popsány možnosti zobrazení rozšířené reality a nakonec její využití v praxi.

2.1 Definice rozšířené reality


Ronald Azuma definuje rozšířenou realitu jako systém se třemi základními vlastnostmi [10]:

1. Kombinuje reálné a virtuální.
2. Je interaktivní v reálném čase.
3. Je registrovaný ve 3D prostoru.

Obrázek 2.1: Znázornění osy virtuálního kontinua [12].
2.2 Trasování virtuálních objektů

Hlavním úkolem AR systému je správně umisťovat virtuální objekty do reálné scény (tzv. registrace). K tomu je potřeba správně najít a trasovat zorný úhel pozorovatele a jeho polohu v reálném prostředí. Pozorovatelem je v rozšířené realitě obvykle myšlena videokamera, která snímá obraz reálné scény, nebo také průhledový displej. Z polohy tohoto pozorovatele se vypočítávají transformace virtuálních objektů. Skutečný pozorovatel v pravém slova smyslu je potom divák, který sleduje výslednou scénu s rozšířenou realitou, a to buď z povzdáli na běžném displeji, nebo pomocí náhlavního displeje (viz kapitola 2.5). V rozšířené realitě se používají dva základní postupy pro získávání zorného úhlu pozorovatele:

- Pomocí trasovacích značek využitím počítačového vidění.
- Pomocí polohových senzorů.

2.2.1 AR využívající počítačové vidění

První metoda využívá videokameru a speciální trasovací značky, které jsou předem umístěny do scény. AR systém potřebuje znát přesnou pozici videokamery vůči značkám, aby byl schopen určit zorný bod, ze kterého se bude vykreslovat virtuální obraz. Jakmile je známa pozice videokamery, která se vypočte podle polohy a natočení značek, virtuální kamera může být umístěna do stejného bodu jako ta reálná a poté je možno vykreslit 3D modely přesně na místa značek (viz obr. 2.2). AR systém používá pro trasování pozice kamery algoritmy počítačového vidění, které jsou schopny rychle počítat polohu v reálném čase.


V následujícím textu budou popsány fáze zpracování obrazu v reálném čase pro knihovnu ARToolKit:

1. Videokamera sejme snímek a pošle jej počítači ke zpracování.
2. Program v počítači zpracuje snímek z videa, převede jej pomocí prahování na černo-bílý obraz a najde v něm všechny černé čtvercové tvary, které odpovídají vzorům trasovacích značek (viz obr. 2.2).
3. Pokud jsou nějaké značky detekovány, program vypočítá pozici a rotaci kamery vůči trasovacím značkám.
4. Jakmile je známa poloha kamery, virtuální 3D grafika může být vykreslena ze stejné pozice.
5. Virtuální grafika je vykreslena přes reálný snímek z videokamery a objekty se zobrazí přesně na místa trasovacích značek.

Metoda využívající počítačové vidění má několik základních omezení, které je potřeba brát v úvahu při návrhu aplikace:

- Virtuální objekty se objeví pouze tehdy, když jsou příslušné trasovací značky plně viditelné ze strany pozorovatele. Pokud je část značky zakryta, nelze ji rozpoznat.
- Správná detekce trasovacích značek ve videu je závislá na světelných podmínkách ve scéně. Při špatném osvětlení nemusí být značky v prahovaném obraze detekovány.
- Správná detekce značek je rovněž závislá na naklonění značek vůči pozorovateli. Čím více je značka odkloněna, tím hůře se rozpoznává.
- Při tvorbě vlastních značek je důležité zvolit vhodné symboly. Platí, že čím jednodušší, tím lepší a tím snadněji se správně rozpozná. Komplexnější vzory jsou z větší dálky hůře detekovány a můžou zkrátit rozlišovací vzdálenost až o polovinu.

Obrázek 2.2: Aplikace AR, využívající metodu počítačového vidění (ARToolKit).

Na příklad, výsledný obraz AR, vlevo dole reálná scéna, vpravo dole detekce trasovacích značek.
2.2.2 AR využívající polohové senzory

Druhá metoda využívá polohové senzory, které jsou schopny určit relativně přesnou polohu pozorovatele a jeho zorný úhel. Těmito senzory jsou např. GPS, kompas, akcelerometr nebo gyroskop (viz kapitola 2.4). Nejčastěji se používají ve spojení s tzv. Head mounted display (dále jen HMD) (viz kapitola 2.5). Základním předpokladem polohových senzorů v rozšířené realitě je, že jejich umístění se nemění vzhledem ke kameře, průhledovému displeji nebo HMD. Senzory jsou rovněž součástí různých moderních mobilních zařízení (viz obr. 2.3). Díky tomu se stávají snadno dostupné i pro běžné uživatele. Metodu rozšířené reality, využívající polohové senzory, používá i aplikace vytvořená v rámci této diplomové práce. V následujícím textu budou popsány fáze zobrazení AR využívající tuto metodu s HMD [10]:

1. Senzory zašlou počítači informace o poloze uživatele a směru, kterým je orientována hlava.
2. Generátor scény na základě polohy uživatele připraví k zobrazení virtuální objekty, které jsou v zorném poli uživatele.
3. Virtuální objekty se zobrazí na displeji.
4. Poloprůhledné zrcadlo odráží tyto objekty do zorného pole uživatele a tím vzniká dojem, že se nacházejí přímo ve scéně.

Obrázek 2.3: Aplikace AR, využívající polohové senzory [15].
2.3 Snímání obrazu

Digitální videokamera je jedním ze základních vstupních zařízení pro rozšířenou realitu. Zajišťuje získávání obrazových dat reálné scény v reálném čase. Zároveň může sloužit jako prostředek pro získání zorného úhlu pozorovatele v případě použití metod využívajících počítačové vidění (viz kapitola 2.2.1).

Videokamera se skládá z objektivu [18], který přenáší snímaný obraz na světlocitlivý senzor, a elektroniky, která zajišťuje čtení, úpravu a ukládání obrazové informace. Objektiv videokamery má několik základních parametrů, které mohou ovlivnit zobrazení scény.

Základním parametrem je ohnisková vzdálenost čočky, která určuje zorný úhel zachycené scény. Důležitým parametrem je také sférické zkreslení [18], které se projevuje jako monochromatická vada přenosu přímek. Vyskytuje se nejčastěji u širokoúhlých objektivů s malou ohniskovou vzdáleností. Sférické zkreslení se dá redukovat vhodnou kombinací dvou čoček, nebo softwarovou korekcí pomocí kalibrace kamery. Toto zkreslení je nejvíce nápadné na obrazcích s kolmými pravidelnými linkami nebo mřížkou. Obvykle roste směrem k okrajům snímku (viz obr. 2.4).

Obrázek 2.4: Sférické zkreslení (redukované, soudkovité, poduškovité) [18].

2.4 Polohové senzory

Polohové senzory jsou vstupním zařízením rozšířené reality. Používají se v aplikacích využívajících metody získávání zorného úhlu pozorovatele pomocí senzorů (viz kapitola 2.2.2). Často bývají součástí různých mobilních zařízení.
2.4.1 GPS

GPS (Global positioning system) [19, 20] je systém pro určování geografické polohy pomocí satelitních družic. Družice obíhají kolem Země a vysílají signály v podobě elektromagnetických vln. GPS lze rozdělit na tři části: kosmickou, řídící a uživatelskou.


Řídící segment spravuje a řídí chod kosmického segmentu, tedy družice na oběžných drahách. Úkolem je sledovat jejich činnost, zasílat jim příkazy, provádět manévry a v případě nějakých problémů má řídící část za úkol je řešit.


Všechny satelity vysílají informace na stejné frekvenci. Rozlišení jednotlivých satelitů se řeší pomocí pseudonáhodného šumu (Pseudo random noise), který je vysílán společně se signálem.


Přijímač GPS komunikuje s počítačem za pomoci protokolu NMEA. Protokol je textového formátu a udává informace o aktuální poloze, času, rychlosti, azimuthu, stavu družic atd. Data protokolu jsou prezentována formou tzv. vět.

---

2 Sklon dráhy tělesa k základní rovině - rovníku
2.4.2 Elektronický kompas

Elektronický kompas (senzor magnetického pole) [22] je senzor umožňující určit orientaci v terénu. Magnetické pole Země směřuje od jižního magnetického pólu k severnímu. Dvouosé magnetické kompasy měří horizontální vektor magnetického pole Země pomocí 2 senzorů v horizontální rovině a kolmo k sobě navzájem. Senzory měří magnetické pole na ose x a y. Orientaci potom udává funkce \( \arctg \frac{x}{y} \), která určuje kurs vzhledem k ose x. Senzory magnetického pole, používané v mobilních zařízeních, měří magnetickou indukci v jednotkách Tesla.


2.4.3 Akcelerometr

Akcelerometr [23, 25] je senzor, který měří zrychlení. V mobilních zařízeních se nejčastěji používají tzv. MEMS (Micro electro-mechanical systems) 3D akcelerometry, které mají velmi malé rozměry, zato však velké možnosti využití:

- Měření náklonu nebo natočení předmětů.
- Měření pohybu a zrychlení.
- Měření vibrací či nárazů.

Akcelerometr obvykle měří zrychlení v jednotkách přetížení g (Gravitational g-force), které představuje násobky normálního gravitačního zrychlení. 1 g odpovídá velikosti normálního gravitačního zrychlení na Zemi, tedy 9,823 \( m/s^2 \). Tabulka 2.1 znázorňuje příklady přetížení g.
Akcelerometr měří jak absolutní zrychlení vůči zemi, tak relativní zrychlení hmoty vůči pohybujícímu se předmětu.

Akcelerometr se skládá ze základny, pružně uložené setrvačné hmoty a tlumení. Technologie MEMS [24] je založena na proměnné kapacitě třielektrodového vzduchového kondenzátoru. Využívá závislost kapacity na vzdálenosti elektrod kondenzátoru. Kapacitní akcelerometr funguje na principu pohyblivé elektrody, jejíž pohyb je závislý na působícím zrychlení. Vychází se ze vztahu pro působení síly \( F \) při zrychlení \( a \) na hmotu \( m \), tedy \( F=m\cdot a \). 2D akcelerometry používají 2 základní stavební struktury vzájemně pootočené o 90°, které měří např. v osách \( X \) a \( Y \). 3D akcelerometry mají potom navíc výškově pohyblivou strukturu v osi \( Z \).

| Stání na Měsíci na jeho rovníku      | 0.1654 g |
| Stání na Zemi na úrovni moře         | 1 g |
| Zrychlení Bugatti Veyron z 0 na 100 km/h v 2.4 s | 1.18 g |
| Start vesmírného raketoplánu (maximum) | 3 g |
| Horská dráha                          | 3.5-6.3 g |
| Prudké brzdění Formule 1 (maximum)   | 5+ g |
| Akrobatické letadlo nebo stíhačka (maximum) | 9-12 g |
| Smrt nebo vážné poranění člověka      | >50 g |
| Raketová střela                       | 100 g |

Tabulka 2.1: Příklady přetížení v jednotkách g (Gravitational g-force) [25].
2.5 Zobrazení rozšířené reality

Zobrazovací displeje slouží jako výstupní zařízení pro rozšířenou realitu. Výslednou scénu rozšířené reality lze zobrazit dvěma základními způsoby [10, 13]:

- Pomocí běžného displeje, monitoru či projektoru.
- Pomocí Head mounted display (HMD).

První možnost realizace rozšířené reality nevyžaduje žádné speciální zařízení a je snadno dostupná všem běžným uživatelům. K realizaci stačí obyčejná kamera, zabudovaná v přenosném zařízení, nebo připojená pomocí USB, a běžný displej. Do této kategorie se řadí i různá mobilní zařízení (tzv. Handheld displays), např. smartphony³. Tuto metodou zobrazení využívá i aplikace vytvořená v rámci této diplomové práce.

Head mounted display je speciální zařízení, které je připevněno na hlavě uživatele (tzv. náhlavní displej). Součástí zařízení jsou speciální brýle s displejem zobrazujícím rozšířenou realitu. A dále kamera, která snímá reálné prostředí, kam se uživatel dívá. Některá zařízení jsou vybavena polohovými senzory. Náhlavní displeje (viz obr. 2.6) se dělí podle typu průhlednosti na:

- Video průhledové (Video see-through).
- Opticky průhledové (Optical see-through).

Video průhledové náhlavní displeje využívají videokameru. Obraz získaný z kamery je zpracován počítačem a doplněn virtuálními objekty. Uživateli je zobrazena výsledná scéna s rozšířenou realitou pomocí speciálních brýlí, vybavených dvěma malými displeji umístěnými těsně před očima pozorovatele.


Obrázek 2.6: Vlevo video průhledový náhlavní displej, vpravo opticky průhledový [16, 17].

³ Telefony s pokročilými funkcemi
2.6 Využití rozšířené reality

Rozšířená realita je poměrně mladou technologií, ale stále více nabývá na popularitě. Má velmi široké uplatnění v různých odvětvích [13, 26]:

- **Zábava**
  Rozšířená realita nachází uplatnění v herním průmyslu a obecně v zábavných a multimedialních aplikacích. Příkladem může být hra ARQuake, kde se hráči pohybují v reálném prostředí a bojují proti virtuálním nepřátelům.

- **Vzdělávání**
  Například interaktivní průvodce v muzeu, který lépe znázorňuje nebo doplňuje vystavované exponáty pomocí rozšířené reality. AR lze uplatnit i ve školách pro lepší a názornější vysvětlení probírané látky např. pomocí animací.

- **Reklama**
  Reklamní plakáty či billboardy, které propagují určitý produkt, mohou zobrazovat pomocí rozšířené reality některé přidané informace. Například v některých prodejnách se stavebnicemi Lego se nachází AR zařízení, které po přiložení krabice se stavebnicí zobrazuje zákazníkovi její 3D model (viz obr. 2.7).

- **Navigace**
  AR lze využít pro navigaci po budově, nebo i venku pro zobrazení významných míst (POI) v okolí. Uplatnění nalézá i v autonavigacích (viz obr. 2.3).

- **Vizualizace**
  Příkladem může být vizualizace návrhu konstrukce budovy, nebo návrh rozmístění nábytku v novém bytě apod.

- **Montáže**
  Rozšířenou realitu lze dobře využít v aplikacích sloužících jako průvodce pro vykonávání komplexních úloh v oblasti montáží a oprav. Například při opravě automobilu, instalaci součástek apod.

- **Armáda**
  Využití HMD pro zobrazování instrukcí, map, pozice nepřátel a dalších informací.

- **Medicína**
  Například vizualizace objektů získaných pomocí ultrazvuku či rentgenu, při operaci.
Obrázek 2.7: Zobrazení modelu stavebnice Lego pomocí rozšířené reality [27].
3 Platforma Google Android

V této kapitole bude představena mobilní platforma Google Android, pro kterou je navržena aplikace rozšířené reality. Bude zde popsána architektura platformy, dále bude podrobně prozkoumána struktura běžné aplikace, její životní cyklus a nakonec základy tvorby uživatelského rozhraní.

Android je nová a rychle se rozvíjející softwarová platforma založená na Linuxovém jádře. Tvoří ji operační systém, middleware a aplikace. Platforma Android je určena především pro smartphony, netbooky, tablety, navigace a další mobilní zařízení.

3.1 Historie

V červenci roku 2005 koupil Google malou společnost Android, Inc., která se zabývala tvorbou software pro mobilní zařízení. Pod vedením jednoho ze zakladatelů společnosti Andy Rubina započal vývoj mobilní platformy Android [1, 3]. Tímto krokem vstoupila společnost Google na trh mobilních technologií.


V současné době je k dispozici nejnovější verze Android 3.0 (Honeycomb), které předcházela řada vývojových verzí:

---

4 Software sloužící jako konverzní nebo spojovací vrstva
- **Android 1.0**

- **Android 1.1**

- **Android 1.5 (Cupcake)**
  Vydána v dubnu 2009. Hlavní změny: možnost nahrávání videoa, podpora Bluetooth profilů, vylepšená softwarová klávesnice s automatickým dokončováním slov, podpora widgetů, kopírovaní a vkládání textu a vylepšené uživatelské rozhraní.

- **Android 1.6 (Donut)**
  Vydána v září 2009. Hlavní změny: vylepšený Android Market, fotoaparát, galerie, Quick Search Box, podpora VPN, CDMA a syntézy řeči.

- **Android 2.0/2.1 (Eclair)**
  Vydána v říjnu 2009. Hlavní změny: optimalizována rychlost hardware, podpora různých rozlišení displeje, vylepšené uživatelské rozhraní, podpora HTML5, Bluetooth 2.1, Microsoft Exchange, podpora světelné diody a animované tapety.

- **Android 2.2 (Froyo)**
  Vydána v květnu 2010. Hlavní změny: možnost instalace aplikací na paměťovou kartu, optimalizace pro hardware, JIT kompilátor, podpora tetheringu a podpora OpenGL ES 2.0.

- **Android 2.3 (Gingerbread)**

### 3.2 Architektura

Android nabízí velké možnosti využití. S každou novou verzí systému přichází další rozšíření a nové funkce. Podporováno je několik bezdrátových technologií jako např. GSM, Bluetooth, WiFi, EDGE, 3G a další. Android umožňuje používat satelitní navigaci GPS a řadu dalších senzorů (např. kompas, akcelerometr, proximity senzor a další).

Architekturu platformy Android můžeme rozdělit do pěti vrstev (viz obr. 3.1) [1, 4].
3.2.1 Linuxové jádro

Na nejnižší vrstvě se nachází Linuxové jádro [4] verze 2.6, které zajišťuje řízení procesů, správu paměti, bezpečnost, síťové služby a spravuje systémové ovladače zařízení. Jádro tvoří abstraktní vrstvu mezi hardwarem a softwarem systému.

3.2.2 Android runtime


Virtuální stroj Dalvik byl vytvořen speciálně pro platformu Android. Byl navržen pro mobilní zařízení s ohledem na nízkou paměťovou náročnost a rychlost procesoru. Před spuštěním je většina aplikací, zkompilovaná Java kompilátorem, převedena na kompaktní formát Dalvik Executable.
3.2.3 Knihovny


- **Systémová knihovna C** - standardní knihovna jazyka C (libc), optimalizovaná pro vestavěná zařízení.
- **Multimediální knihovny** - pro přehrávání a nahrávání různých audio a video formátů (např. MP3, AAC, MPEG, JPEG, PNG a další).
- **Grafické knihovny** - pro vykreslování 2D a 3D grafiky (např. SGL, OpenGL ES).
- **FreeType** - pro vykreslování bitmapových a vektorových fontů.
- **LibWebCore** - moderní webový engine.
- **SQLite** - jednoduchý relační databázový engine.

3.2.4 Aplikační rozhraní


- **Views** - umožňují vytvářet grafické uživatelské rozhraní.
- **Content Providers** - poskytují přístup k informacím ostatních aplikací (např. kontakty).
- **Resource Manager** - zajišťuje přístup ke zdrojům (např. grafika, zvuk, řetězce).
- **Notification Manager** - umožňuje vytvářet vlastní notifikace ve stavové liště.
- **Activity Manager** - spravuje životní cyklus aplikace.

3.2.5 Aplikace

Na nejvyšší vrstvě jsou samotné aplikace [4], poskytující uživateli široké možnosti využití mobilního zařízení. Operační systém Android obsahuje několik výchozích aplikací (např. e-mailový klient, webový prohlížeč, kontakty, SMS program, kalendář, mapy a další). Všechny aplikace jsou vytvořeny v programovacím jazyce Java.
3.3 Vývoj

Aplikace pro systém Android lze vyvíjet dvěma způsoby - pomocí sady nástrojů NDK nebo SDK [1, 4].

NDK (Native development kit) je sada knihoven a vývojových nástrojů, která umožňuje programovat aplikace v nativním kódu pomocí jazyka C/C++. NDK je vhodné zvláště pro výpočetně náročné programy. Aplikace vytvořené v NDK jsou v některých případech rychlejší a efektivnější než běžné aplikace. V této práci se budeme zabývat vývojem pomocí sady SDK, jak je popsáno níže.


Součástí vývojové sady je emulátor simulující hardwarové zařízení s operačním systémem Android. Aplikace lze rovněž ladit i na fyzickém zařízení s OS Android, připojeném USB kabelem k počítači v režimu ladění. S pomocí nástrojů ADB (Android Debug Bridge) a DDMS (Dalvik Debug Monitor Server) je možné komplexně testovat aplikace, sledovat výstupy, výjimky, chyby, spravovat zařízení, procesy, procházet úložiště dat, ukládat screenshoty nebo testovat pokročilejší funkce zařízení jako např. GPS, telefonní hovory, SMS atd. Emulátor implicitně neumožňuje simulaci akcelerometru, kompasu či kamery. Tyto funkce se testují pomocí jiných speciálních nástrojů.
3.4 Struktura aplikace

Aplikace platformy Android se skládá z několika součástí, které jsou vzájemně propojeny. Tyto součásti umožňují běh aplikace a interakci s celým systémem. Patří mezi ně: aktivity, služby, sdělení, poskytovatelé obsahu a manifést [7].

3.4.1 Aktivity

Aktivita (Activity) [6] je základní komponenta celého programu. Odpovídá obvykle jedné obrazovce aplikace. Představuje uživatelské rozhraní poskytující funkce pro ovládání aplikace a zobrazení výstupních informací na displej. Aktivity k tomu využívají Views, umožňující vytvářet různé grafické a ovládací prvky (např. tlačítka, seznamy, menu, tabulky atd.).

3.4.2 Služby

Služba (Service) [7] umožňuje vykonávat určité akce probíhající na pozadí. Nemá k dispozici uživatelské rozhraní, ovládá se pomocí aktivity, která ho zpřístupňuje. Příkladem může být hudební přehrávač, který pomocí služby přehrává muziku na pozadí.

3.4.3 Sdělení


3.4.4 Poskytovatelé obsahu

Poskytovatelé obsahu (Content providers) [7] zajišťují sdílení dat mezi jednotlivými aplikacemi. Data se nejčastěji ukládají do SQLite databáze. Příkladem sdílených dat mohou být kontakty, historie telefonních hovorů, účty.

3.4.5 Manifest

Manifest [7] je soubor obsahující základní informace o programu. Informuje systém o možnostech, které mu může aplikace nabídnout. Každá aplikace má svůj vlastní manifest uložený ve formátu XML. Manifest obsahuje následující informace:

- Název balíčku, sloužící jako jednoznačný identifikátor aplikace v systému.
- Popis jednotlivých komponent aplikace (aktivity, služby, filtry sdělení, přijímači broadcastů a poskytovatelé obsahu).
- Povolení k využívání různých služeb systému (např. získávání pozice, přístup k internetu nebo použití videokamery).
- Deklarace povolení, které musí mít ostatní aplikace k používání této.
- Minimální verze Android API, kterou aplikace vyžaduje pro svůj běh.

V následujícím textu je ukázka manifestu, která obsahuje jednu základní aktivitu, povolení k používání kamery a definici minimální verze SDK:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mypackage"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name="MyActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-sdk android:minSdkVersion="8" />
</manifest>
```
3.5 Životní cyklus aplikace

Systém pracuje s aktivitami pomocí zásobníku. Všechny spuštěné aktivity v systému jsou uloženy na zásobníku a aktivní aktivita, se kterou se právě pracuje, je uložena na vrcholu zásobníku. Pokud se aktivita ukončí, je odstraněna z vrcholu zásobníku a aktivní se stává předchozí aktivita. Přepínání aktivit umožňuje hardwarové tlačítko Zpět.


Služby mají také svůj životní cyklus podobný aktivitám.

Obrázek 3.2: Životní cyklus aktivity [6].
3.6 Uživatelské rozhraní


Uživatelské rozhraní se skládá z objektů nazývaných Views. Tyto objekty jsou strukturovány v hierarchickém stromě (viz obr. 3.3). Views tvoří základ tzv. Widgets, elementárních prvků uživatelského rozhraní zajišťujících interakci s uživatelem. Příkladem takových prvků jsou tlačítka, textová pole, zaškrávací pole nebo seznamy. Systém poskytuje řadu těchto prvků, ale je možné vytvářet i vlastní.


- **FrameLayout** - základní Layout. Představuje prázdné místo na obrazovce, které může být doplněno jednoduchým objektem, např. obrázkem.
- **LinearLayout** - zarovnává všechny objekty v horizontálním nebo vertikálním směru.
- **TableLayout** - zarovnává objekty do řádků a sloupců tabulky.
- **RelativeLayout** - umožňuje rozmístit jednotlivé objekty na určité pozice vzhledem k ostatním.
- **AbsoluteLayout** - umožňuje rozmístit jednotlivé objekty na přesné pozice dané souřadnicemi x a y.

Pomocí XML lze vytvářet nejen Layouty, ale i další prvky jako např. menu, styly, barvy, gradienty, seznamy a další.

![Hierarchický strom prvků uživatelského rozhraní](image-url)

Obrázek 3.3: Hierarchický strom prvků uživatelského rozhraní [8].
4 Grafická knihovna OpenGL ES

OpenGL ES (Open Graphics Library for embedded systems) je odlehčená verze grafické knihovny OpenGL [28], zaměřená převážně na mobilní zařízení, tablety, navigace, herní konzole a jiné vestavěné systémy. Otevřený standard OpenGL specifikuje multiplatformní rozhraní (API) pro tvorbu počítačové grafiky. Spravuje jej konsorcium ARB (Architecture Review Board), jehož členy jsou významné společnosti jako např. NVIDIA, ATI, Microsoft atd. OpenGL je jedna z nejrozšířenějších grafických knihoven a podporuje velkou řadu platform i programovacích jazyků včetně platformy Android a jazyku Java.


4.1 Struktura knihovny


Vykreslování obrazových dat se provádí do obrazových rámčů (tzv. framebuffers). Základní grafická primitiva tvoří body, úsečky a polygony, které mohou být vykreslovány v různých režimech. Jednotlivá primitiva jsou tvořena pomocí vrcholů, které mají určité parametry (souřadnice vrcholu, texturovací souřadnice, barvy a normály).

Knihovna OpenGL ES, zaměřená na mobilní zařízení, obsahuje značně zjednodušené rozhraní s ohledem na výkonnostní limity cílových zařízení. Především je kladen velký důraz na nízkou paměťovou náročnost a s tím spojenou nízkou spotřebou elektrické energie z baterie.

V současné době je k dispozici nejnovější verze OpenGL ES 2.0, které předcházelo několik vývojových verzí [29]:

- **OpenGL ES 1.0**
  První verze vychází ze specifikace OpenGL 1.3. Hlavní odlišností verze OpenGL pro mobilní zařízení je, že nepoužívá ve své sémantice volání `glBegin` a `glEnd` pro vykreslování primitiv. Další výraznou změnou je použití fixed-point aritmetiky pro uložení souřadnic vrcholů (tedy čísla s fixní řádovou čárkou) z důvodu vyšší kompatibility s embedded procesory, které často postrádaly FPU (floating point unit). Tato verze je tedy
určena pro méně výkonné zařízení s nižší cenou, provádějící výpočty softwarově. Dále tato verze postrádá velkou řadu pokročilejších funkcí.

- **OpenGL ES 1.1**
  Tato verze vychází ze specifikace OpenGL 1.5 a přidává do původní verze několik nových funkcí, především podporu hardwarové akcelerace.

- **OpenGL ES 2.0**
  Nejnovější verze vychází ze specifikace OpenGL 2.0 a není zpětně kompatibilní s verzemi OpenGL ES 1.x. Byla vydána v březnu 2007 a přináší plně programovatelnou 3D grafiku s možností vytvářet vertex a fragment shadery v OpenGL ES Shading Language. Programátor má plnou kontrolu nad renderovací pipeline na rozdíl od verze 1.x, kde je pipeline fixní (viz obr. 4.1). V této verzi knihovny byly odstraněny všechny fixní funkce, které je možné nahradit shadery. Díky nim lze také vytvářet řadu nejrůznějších efektů. Tato verze knihovny je určena pro výkonnější zařízení s hardwarovou podporou 3D zobrazování.

**Existing Fixed Function Pipeline**

![Image](image)

Obrázek 4.1: Fixní pipeline v OpenGL ES 1.0 [30].

---

5 Specializovaný procesor pro zpracování grafických informací grafickou kartou
6 Zajišťuje zřetězené zpracování toku grafických dat
4.2 OpenGL ES na platformě Android


Jednotlivé funkce z OpenGL API zpřístupňuje v systému Android třída GLSurfaceView, která je součástí Android SDK. Tato třída implementuje rozhraní zajišťující vykreslování grafiky. V instanci této třídy se registruje Render, který běží ve vlastním vláknu, provádí volání procedur OpenGL API a stará se o vykreslování obrazu. Metoda onDrawFrame zajišťuje vykreslení jednoho snímku obrazu a volá se pokaždé, kdy je potřeba překreslit aktuální snímek. V této metodě se typicky volají funkce OpenGL API tak, jak to známe např. z PC verzí OpenGL programů. Tedy nejprve dojde k vynulování bufferů, vynulování transformační matice, nastavení pohledu kamery a nakonec vykreslení samotných grafických objektů.

4.3 Zobrazení 3D grafiky a transformace

Poloha grafických objektů ve scéně je určena souřadnicemi určité souřadné soustavy. OpenGL pracuje s 2D nebo 3D prostory a využívá Kartézskou soustavu souřadnic. Každý vrchol je určen dvojicí resp. trojicí skalárů, které určují pozici v daném prostoru. V mojí aplikaci používám 3D prostor, který se v rozšířené realitě připodobňuje k reálnému světu s definovanými zeměpisnými souřadnicemi a nadmořskou výškou.

Kartézská soustava souřadnic [31] je speciálním typem afinní soustavy, pro kterou platí, že souřadné osy jsou na sebe kolmé. Afinní soustava je tvořena třemi různoběžnými osami, které se protínají v jednom bodě - počátku soustavy souřadnic. Určení hodnot souřadnic se provádí vedením rovnoběžek se souřadnicovými rovinami. Hodnoty souřadnice jednoho bodu jsou potom určeny protnutím dané rovnoběžky se souřadnou osou.

Definice 4.3.1  Bod $P=[x,y,z]$ má homogenní souřadnice $P_h=[\omega x, \omega y, \omega z, \omega]$ právě tehdy, když platí:

$$X = x/\omega \quad Y = y/\omega \quad Z = z/\omega$$

kde $\omega$ je váha bodu.

Pro jednotlivé transformační operace se v 3D prostoru používají transformační matice o velikosti 4x4. Ve 2D prostoru potom o velikosti 3x3. Transformace lze skládat násobením matic, přičemž záleží na pořadí prováděných transformací, protože násobení matic není komutativní. Transformační matice pro základní operace jsou popsány níže.

**Posunutí:**

$$T = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
d_x & d_y & d_z & 1
\end{pmatrix}$$

(4.1)

**Změna měřítka:**

$$S = \begin{pmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$

(4.2)

**Otočení:**

$$R = \begin{pmatrix}
\cos \alpha & \sin \alpha & 0 & 0 \\
-\sin \alpha & \cos \alpha & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$

(4.3)

**Zkosení:**

$$SH = \begin{pmatrix}
1 & S_{hy} & S_{hz} & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$

(4.4)
Mezi transformační operace patří také projekce scény. Projekce je zobrazení 3D objektů na 2D médium a lze ji vyjádřit pomocí matice 4x4. OpenGL ES umožňuje zobrazit scénu pomocí paralelní nebo perspektivní projekce (viz obr. 4.2). Scéna je určena specifickým objemem tělesa, které popisuje danou projekci. V paralelní projekci je objemovým tělesem kvádr, zatímco v perspektivní projekci je to komolý jehlan, přičemž kamera směřuje na větší z protilehlých stěn.

Paralelní projekce:

\[
P_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\]

(4.5)

Perspektivní projekce:

\[
P_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix}
\]

(4.6)

Obrázek 4.2: Paralelní a perspektivní projekce.
5 Návrh systému

Cílem praktické části této diplomové práce je navrhnout a implementovat program realizující rozšířenou realitu pro podporu navigačního software. Program by měl vykreslit body zájmu do obrazu získaného kamerou s využitím polohových senzorů. Tato metoda rozšířené reality byla bližší popsána v kapitole 2.2.2.

Systém rozšířené reality byl navržen jako samostatný framework, na kterém je postavena samotná aplikace pro podporu letecké navigace. V této kapitole bude popsán základní princip fungování navrženého frameworku rozšířené reality a návrh programu navigace, která by měla být schopna vizualizovat letecká data včetně nadmořské výšky.

5.1 Framework rozšířené reality


Obrázek 5.1: Schematické znázornění základního principu činnosti frameworku rozšířené reality.

\[\text{Obrázek 5.1: Schematické znázornění základního principu činnosti frameworku rozšířené reality.}\]

\[^7\text{Softwarová struktura, sloužící jako podpora při vývoji jiných softwarových projektů}\]
5.1.1 Realizace rozšířené reality

Systém zajišťuje vykreslování rozšířené reality v reálném čase na displej mobilního telefonu a trasování virtuálních objektů za pomoci akcelerometru, senzoru magnetického pole a GPS. Obrazové informace se vykreslí do tří základních vrstev. První nejspodnější vrstva zobrazuje reálné prostředí zachycené pomocí videokamery mobilního zařízení. Druhá vrstva vykresluje virtuální objekty zpracované počítačem - body zájmu (dále jen POI). Třetí nejsvrchnější vrstva potom zobrazuje dodatkové informace jako např. azimut, GPS souřadnice, GPS signál, mapku a ovládací prvky aplikace. Ukázka návrhu aplikace je znázorněna na obrázku 5.2.

Virtuální vrstva, zobrazující body zájmu, je vykreslována v 3D prostoru. To umožňuje velmi snadnou manipulaci s POI ve virtuálním prostoru a rovněž využití možností 3D grafiky. Framework používá k renderování 3D scény knihovnu OpenGL ES. Body zájmu tak mohou být zobrazovány v podobě 2D ikon nebo 3D modelů. Po kliknutí na POI se zobrazí okno s detailními informacemi daného objektu (viz obrázek 5.3).

![Obrázek 5.2: Návrh aplikace rozšířené reality.](image)

8 Point of interest
Trasování virtuálních objektů

5.1.3 Souřadný systém


Důležitá je správná kalibrace videokamery, aby její zorný úhel odpovídal nastavení virtuální scény. Vzhledem k tomu, že převážná většina mobilních zařízení nepoužívá transfokátor a ohnisková vzdálenost objektivu je pevně daná, můžeme pro zjednodušení použít fixní zorný úhel pro nastavení perspektivní projekce scény. Zorné úhly získáme z parametrů použité videokamery. Perspektiva 3D prostředí je nastavena tak, aby 1 jednotka vzdálenosti ve virtuální vrstvě OpenGL odpovídala 1 metru ve skutečnosti. To umožňuje snadné měření vzdálenosti ve virtuálním prostoru a jednoduché přidávání bodů zájmu na odpovídající místo.

5.1.4 Uživatelské rozhraní a databáze

Grafické uživatelské rozhraní bylo navrženo s ohledem na jednoduché a intuitivní ovládání. Je snadno přizpůsobitelné potřebám programátora. Framework poskytuje několik základních widgetů zobrazujících kompas, mapku s nejbližšími body zájmu, údaje o poloze, regulátor vzdálenosti pro zobrazení POI a další.

Body zájmu si systém ukládá do lokální databáze SQLite. Databáze obsahuje informace o názvu POI, typu, popisu, webové adrese, geografických souřadnicích, nadmořské výšce a případně další volitelné informace. Jednotlivé body zájmu tvoří tzv. kanály, které sdružují určité příbuzně body zájmu. Např. v aplikaci letecké navigace mohou být použity kanály: letiště, města, hory, jezera apod. Framework umožňuje import POI ve standardizovaném formátu GPX. Import je možno provést z libovolných médií podle potřeb programátora (ze souboru na SD kartě, z webového serveru, apod.).
Systém pracuje pouze s body zájmu, které mají pevnou a neměnnou geografickou pozici. Strukturu databáze popisuje diagram na obrážku 5.5.

Obrázek 5.5: Entity-relationship diagram databáze.

5.2 Letecká navigace

Program letecké navigace používá navržený framework rozšířené reality popsaný výše. Aby bylo možné systém používat jako leteckou navigaci, musí umět pracovat i s nadmořskou výškou. Framework s tímto požadavkem počítá, ale je potřeba zajistit, aby všechny body zájmu měly definovanou správnou výšku, jelikož je tento parametr v GPX standardu nepovinný.

Po spuštění programu na mobilním zařízení se zobrazí úvodní obrazovka, kde jsou vypsány všechny dostupné kanály. Uživatel si vybere kanály, které ho zajímají, a zvolí si jeden ze způsobů zobrazení POI:

- Zobrazení rozšířené reality.
- Seznam bodů zájmu.

V pohledu rozšířené reality se vykreslí vybrané body zájmu z blízkého okolí. Další z možných zobrazení je detailní výpis všech informací o bodech zájmu v přehledném seznamu. V tomto seznamu má uživatel dále možnost zobrazit si webovou stránku zvoleného bodu zájmu, sdílet POI prostřednictvím sociálních sítí nebo e-mailu a případně také zvolený POI smazat z databáze. Program má také několik uživatelských nastavení jako maximální vzdálenost POI, maximální počet kanálů a další.

Výsledný program je použitelný nejen pro letectví, ale třeba i pro turistiku. Značně usnadňuje orientaci v terénu a na vyvýšených místech jako jsou rozhledny nebo vyhlídky, může také dobře posloužit jako doplněk turistického průvodce. V tomto případě je ovšem nutné do programu importovat turisticky zajímavé kanály. Samotný framework má pak mnohem širší možnosti využití.

Více o letecké navigaci a jejím testování bude uvedeno v kapitole 7.
6 Implementace aplikace

Tato kapitola se zabývá samotnou implementací frameworku rozšířené reality se zaměřením na polohové senzory a tvorbou aplikace letecké navigace. Bude zde popsána struktura celé aplikace, princip realizace zobrazení rozšířené reality a vykreslení bodů zájmu. Budeme se zabývat i problematikou výpočtu orientace a geolokace, použitím databáze pro ukládání informací a tvorbou uživatelského rozhraní.

6.1 Struktura aplikace


![Obrázek 6.1: Rozdělení tříd do balíků.](image-url)
Jednotlivé balíky mají následující funkce:

- **Main**
  V tomto balíku se nachází třídy definující uživatelské rozhraní a chování jeho prvků. Jsou zde všechny aktivity a třída ošetřující uživatelské nastavení aplikace.

- **AR**
  Tento balík implementuje samotnou rozšířenou realitu. Obsahuje metody pro zobrazení všech tří vrstev, které byly popsány v návrhu systému. Máme zde třídy pro získání obrazu z videokamery, výpočet orientace, výpočet geografické pozice a OpenGL renderer, který vykresluje obraz rozšířené reality na displej telefonu.

- **Database**
  Tento modul zajišťuje ukládání dat do databáze SQLite. Umožňuje provádět všechny potřebné CRUD operace nad daty. Rovněž obsahuje parser GPX formátu pro zpracování vstupních dat při importu.

- **Matrix**
  V tomto balíku je několik tříd pro práci s maticemi. Třídy používají knihovnu OpenGL ES. Zdrojové kódy byly staženy z ukázkových příkladů Android projektu, protože dané metody se nenacházejí ve standardní sadě vývojových nástrojů SDK a bylo potřeba je použít v mém projektu pro získání projekční matice.

- **Mesh**
  Součástí tohoto balíku jsou třídy, které se starají o grafickou reprezentaci bodů zájmu a dalších objektů v pohledu rozšířené reality. Jedná se o grafické modely a jejich seskupení, které se vykreslí v rendereru OpenGL.

- **Message**
  Tento balík zajišťuje komunikaci jednotlivých vláken mezi sebou. Komunikace funguje na principu zasílání zpráv.

- **POI**
  V tomto modulu jsou obsaženy třídy uchovávající informace o kanálech a bodech zájmu. Součástí je i třída zajišťující načtení a aktualizaci odpovídajících bodů zájmu, které jsou určeny k vykreslení v pohledu rozšířené reality.

Aplikace používá několik modulů mobilního zařízení pro získávání informací. Jedná se o videokameru, akcelerometr, senzor magnetického pole, GPS a internet. Přístup k těmto zdrojům dat je nutné povolit v manifestu projektu a uživatel ho musí schválit při instalaci aplikace z důvodu bezpečnosti. Případná absence některého z těchto modulů v zařízení může způsobit částečnou

---

11 Create, read, update, delete
či úplnou dysfunkci celého programu, jelikož aplikace rozšířené reality je přímo závislá na použití polohových senzorů. Tato situace by ovšem neměla nastat, protože všechny telefony se systémem Android, které jsou v současné době na trhu dostupné, podporují výše uvedené funkce.

6.2 Zobrazení rozšířené reality

Pohled rozšířené reality spouští aktivita ARactivity. V ní se inicializují základní objekty důležité pro běh rozšířené reality v reálném čase - orientace, geolokace a objekt, který uchovává body zájmu určené k vykreslení. Po spuštění aktivity se přepne zobrazení do režimu landscape fullscreen a vytvoří se tři vrstvy rozšířené reality - video vrstva, virtuální vrstva a vrstva s dodatečnými informacemi. Tyto vrstvy se nastaví v layoutu aktivity, přičemž se vzájemně překrývají v pořadí, které je uvedeno ve schématu na obrázku 5.1.

Vztahy mezi jednotlivými třídami pohledu rozšířené reality jsou znázorněny ve zjednodušeném diagramu tříd na obrázku 6.2. Zobrazení rozšířené reality je ukáznáno na screenshotu aplikace na obrázku 6.3.

Obrázek 6.2: Zjednodušený diagram tříd rozšířené reality.

12 Zobrazení programu na plnou velikost displeje
13 Ukázkový snímek obrazovky
Na nejpodnější vrstvě se vykresluje obraz z videokamery. Vykreslování běží ve vlastním vláknu. Při přerušení aktivity je důležité, aby byla instance videokamery uvolněna, protože se nejedná o sdílené zařízení.

Virtuální vrstva inicializuje renderer vykreslovací knihovny OpenGL ES. Renderer běží rovněž ve vlastním vláknu a komunikuje s vnějším prostředím pomocí zasílání zpráv. Při vytvoření renderoru je potřeba nastavit transparentní vrstvy, aby nebyl překryt obraz z videokamery. Renderer používá verzi knihovny OpenGL ES 1.0, která poskytuje všechny potřebné funkce. Nejdříve se nastávají parametry renderoru jako např. hladké stínování (smooth shading), testování hloubky (depth test), velikost okna (viewport) a nastavení perspektivy. Správné nastavení perspektivy scény je velmi důležité pro korektní zobrazení pozice bodů zájmu. Timto způsobem se kalibruje kamera tak, aby dané POI odpovídaly skutečné pozici v reálném prostředí. V nastavení perspektivy je potřeba definovat velikost zorného úhlu ve směru osy y, poměr stran, přední a zadní ořezávací rovinu scény. Zorný úhel (tzv. field of view) lze získat z parametrů videokamery daného zařízení. Vypočítá se pomocí následujícího vztahu:

\[
fovw = \text{atan} \left( \frac{\text{sensor width} \cdot 0,5}{\text{focal length}} \right) \cdot 2
\]

\[
fov_h = \text{atan} \left( \frac{\text{sensor height} \cdot 0,5}{\text{focal length}} \right) \cdot 2
\]

V případě potřeby je možné zorný úhel nastavit manuálně v uživatelském nastavení aplikace. Vzdálenost zadní ořezávací roviny je závislá na uživatelském nastavení - maximální vzdálenosti POI.


Poslední nejsvětelnější vrstvou rozšířené reality jsou doplňkové informace a prvky uživatelského rozhraní. Na displeji se zobrazují informace o azimuthu (ve stupních), zdroji
geografických dat (GPS nebo BTS), přesnosti zaměření geografické pozice (v metrech), počtu zobrazených bodů zájmu a maximální vzdálenosti, do které se POI vykreslují (viz obr. 6.3). Vrstva dále zajišťuje zobrazení detailních informací o POI v případě, že na něj uživatel klikne. V dialogovém okně se vypíší informace o názvu POI, názvu kanálu, do kterého přísluší, popisu POI, aktuální vzdálenosti, azimuthu, souřadnicích a nadmořské výšce (viz obr. 6.4). Uživatel má rovněž možnost navštívit webové stránky příslušného bodu zájmu, pokud jsou k dispozici. Dále může také sdílet informace o bodu zájmu se svými přáteli na sociálních sítích jako např. Twitter či Facebook, nebo zaslat informace o POI prostřednictvím e-mailu. Detekce kliknutí na bod zájmu bude bližší vysvětlena v kapitole 6.7.

Obrázek 6.3: Pohled rozšířené reality v režimu bez nadmořské výšky s vypnutou videokamerou.

Obrázek 6.4: Dialogové okno s detailními informacemi o daném bodu zájmu.
6.3 Výpočet orientace


Senzory na platformě Android používají vlastní souřadný systém [33], který je definován relativně k obrazovce mobilního zařízení v východní pozici. Osa x je horizontální a směřuje zleva doprava. Osa y je vertikální a směřuje zdola nahoru. Osa z směřuje odzadu dopředu. Souřadnice, nacházející se za obrazovkou mají zápornou hodnotu z. Při změně orientace obrazovky z landscape nebo portrait režimu se osy nepohybovají. Souřadný systém je znázorněn na obrázku 6.5.

Obrázek 6.5: Souřadný systém zařízení pro SensorEvent API [33].

Data ze senzorů zpracovává instance třídy SensorEventListener, která odchytává události. V případě odchycení události získáme přístup k novým datům ze senzoru daného typu. Dále můžeme zjistit přesnost výpočtu a čas, ve kterém byla data zpracována. Při každé nové události se přepočítávají všechny tři výsledné informace o orientaci, aby byly vždy aktuální.
Akcelerometr [33] vrací naměřené zrychlení na všech třech osách v jednotkách \( m \cdot s^{-2} \). Tento senzor měří zrychlení vzhledem k danému zařízení \( (A_d) \). Měření je závislé na silách, jež působí na samotný senzor \( (F_i) \):

\[
A_d = - \sum \frac{F_i}{m}
\]

(6.2)

Naměřené zrychlení vždy ovlivňuje zejména gravitační síla:

\[
A_d = -g - \sum \frac{F_i}{m}
\]

(6.3)

Proto vrací akcelerometr v klidové poloze zrychlení o hodnotě \( g = 9.81 \text{ m} \cdot \text{s}^{-2} \). Z tohoto poznatku plyne, že k výpočtu skutečného zrychlení musí být složka gravitační sily odstraněna. Toho lze dosáhnout pomocí filtru horní propusti. Naopak filtr dolní propusti lze použít k izolaci gravitační sily. Pro naše potřeby není z těchto důvodů nutné data upravovat, jelikož je pro výpočet výsledné rotační matice použita knihovní funkce, která má na vstupu čistá data získaná ze senzorů.

Senzor magnetického pole měří velikost magnetické indukce okolního prostředí na všech třech osách v jednotkách mikro-Tesla (\( \mu T \)).

Data mohou být značně zkreslena v případě, kdy se zařízení nachází v silném magnetickém poli, je ve volném pádu nebo značně zrychluje. V těchto situacích většinou vračí senzory chybná data, jelikož nejsou schopná veličiny správně změřit.

Senzory obvykle vrací data s poměrně velkým šumem, který je způsoben vnějšími vlivy a chybou ve výpočtu. Šum je proto potřeba redukovat, aby bylo výsledné zobrazení virtuální vrstvy plynulé, stabilní a nedocházelo ke chvění či otřesům kamery ve scéně. Původně byl v aplikaci pro odstranění šumu použit Kalmanův filtr. Při testování však bylo zjištěno, že výsledky Kalmanova filtru jsou přibližně srovnatelně s jednodušším Mean filtrem, který funguje na principu průměrování. Kalmanův filtr je výpočetně náročnější, a proto byl nakonec v aplikaci nahrazen Mean filtrem s prahováním, který dosahuje poměrně dobrých výsledků. Oba senzory používají vlastní instanci Mean filtru. Vstupem je vektor hodnot získaných ze senzoru a výstupem je vektor hodnot s redukovaným šumem. Filtr si vždy ukládá hodnoty z předchozího výpočtu a s jejich pomocí počítá nové hodnoty. Použití hodnot z předchozího výpočtu má za následek určité časové zpoždění filtru.
Výpočet nové hodnoty vyjadřuje následující vzorec:

\[ outVector[i] = factor \cdot inVector[i] + (1 - factor) \cdot lastVector[i] \]

(6.4)

kde \( factor \) je desetinné číslo z intervalu \((0, 1)\), které vyjadřuje míru vlivu na výsledek aktuální vstupní hodnoty oproti předchozí vypočítané hodnotě. Jinými slovy toto číslo ovlivňuje míru vyhlazení zašuměných dat. Pokud je \( factor \) roven 1, filtrování je vypnuto. Čím více se \( factor \) blíží 0, tím je zpoždění filtru větší. Filtr dále využívá prahování, které odstraňuje nepatrné chvění. Pokud je rozdíl nové a předchozí hodnoty menší než zvolený práh, nová hodnota se zahodí a použije se hodnota z předchozího výpočtu. Redukci šumu dat z akcelerometru a magnetometru pomocí Mean filtru znázorňují grafy na obrázcích 6.6 a 6.7. V grafu 6.6 je dobře patrné časové zpoždění filtru. Testovací data uvedená v graftech byla získána experimentálně při otáčení telefonu po směru hodinových ručiček v landscape režimu. K měření byl použit telefon HTC Desire.

![Obrázek 6.6: Redukce šumu dat získaných z akcelerometru.](image-url)
Obrázek 6.7: Redukce šumu dat získaných ze senzoru magnetického pole.

Poté co získáme data ze senzorů a redukujeme šum pomocí filtru, můžeme vypočítat výsledné informace o orientaci. Požadovanou rotační matici získáme voláním knihovní funkce `getRotationMatrix` z třídy SensorManager [34]. Tato metoda přijímá na vstupu data z akcelerometru a magnetometru. Transformuje souřadný systém zařízení (viz obr. 6.5) na světový souřadný systém (viz obr. 6.8) a na výstup vrací transformační matici o rozměru 4x4. Matice je uložena v jednorozměrném poli a její podoba je následující:

\[
\begin{bmatrix}
M_0 & M_1 & M_2 & 0 \\
M_4 & M_5 & M_6 & 0 \\
M_8 & M_9 & M_{10} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(6.5)

Osa x je definována ve světových souřadnicích jako vektorový součin \( y \times z \). Je tečná k zemi v bodě, kde se zařízení aktuálně nachází a směřuje na východ. Osa y je tečná k zemi v bodě, kde se zařízení aktuálně nachází a směřuje na sever. Osa z směřuje k nebi a je kolmá k zemi. Získaná matice může být použita v knihovně OpenGL. Nejdríve je však nutné ji přepočíst tak, aby souřadnice odpovídaly souřadnému systému OpenGL. To provedeme pomocí knihovní funkce `remapCoordinateSystem`. V našem případě, kdy pokaždé pracujeme v landscape režimu, je potřeba přemapovat souřadnici zařízení x na světovou y souřadnici a souřadnici zařízení y na -x světovou souřadnici. Výslednou
matici je nyní možné použít v rendereru OpenGL pro nastavení pohledu kamery ve scéně použitím funkce `glLoadMatrixf`.

Azimut a svislá orientace se vypočítají z rotační matice využitím goniometrické funkce arcus tangens. Při výpočtu azimutu nejdříve invertujeme výslednou rotační matici. Následně ji vynásobíme vektorom:

$$sZVector = (0,0,1,1)$$

(6.6)

a získáme nový vektor **azimuthVector**. Výsledný azimut (ve stupnících) se vypočítá pomocí vztahu:

$$azimuth = 180 + \text{atan2}(azimuthVector[0], azimuthVector[1]) \cdot RAD_TO_DEG$$

(6.7)

kde funkce `atan2(y,x)` vráci nejbližší dvojitou aproximaci funkce arcus tangens $y/x$ v rozsahu $(-\pi,\pi)$. Pro dvě libovolná reálná čísla $x$, $y$, která se nerovnají nule, platí, že funkce `atan2(y,x)` vráci úhel v radiánech mezi pozitivní osou $x$ roviny a bodem $(x,y)$ ležícím v této rovině. Konstanta `RAD_TO_DEG` převádí radiány na stupně a je rovna hodnotě $180/\pi$.

Při výpočtu svislé orientace nejdříve vynásobíme rotační matici vektorom $sZVector$. Získáme nový vektor **orientationVector**, pomocí něhož vypočítáme výslednou orientaci dle následujícího vztahu:

$$orientation = -\text{atan2}(orientationVector[0], orientationVector[1]) \cdot RAD_TO_DEG$$

(6.8)

Obrázek 6.8: Světový souřadný systém [34].
6.4 Výpočet geolokace

Výpočet geolokace je důležitý pro správné rozmístění bodů zájmu na odpovídající souřadnice ve scéně. Geolokace se počítá ve vlastním vlákně a používá data z GPS. Pokud nejsou informace z GPS dostupné, lokace se určuje ze sítě využitím informací o dostupnosti BTS stanic a přístupových bodů sítě WiFi. Pozice získaná ze sítě je však značně nepřesná (stovky až tisíce metrů), a proto je pro použití v rozšířené realitě nevhodná. Lokace z GPS senzoru je mnohem přesnější (jednotky až desítky metrů), a proto se doporučuje její použití v aplikaci. Tyto dva uvedené zdroje geografických dat mají určité výhody a nevýhody. GPS dosahuje poměrně vysoké přesnosti, ovšem na úkor velké spotřeby elektrické energie. Další velkou nevýhodou je, že GPS přijímač vyžaduje přímý vizuální kontakt se satelity. V interiérech nebo jiným způsobem stíněných místech bude signál velmi slabý či nulový. Geodata získávaná ze sítě jsou zase závislá na GSM modulu a dostupnosti WiFi sítí. Poloha je značně nepřesná, jelikož dochází k průměrování pozice podle signálu z dostupných sítí. Poloha vysílače dané sítě je uložena v databázi. Jde o pouhý odhad pozice. Databázi s BTS stanicemi poskytuje např. Open source projekt OpenCellID, který rovněž poskytuje API. Obdobně databázi s WiFi sítěmi poskytuje služba Wireless Geographic Logging Engine.


<table>
<thead>
<tr>
<th>Informace</th>
<th>Geodata z GPS</th>
<th>Geodata ze sítě</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zeměpisná šířka (stupně)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Zeměpisná délka (stupně)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Časová známka (UTC v milisekundách)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Přesnost zaměření (metry)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Nadmořská výška (metry)</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Azimut (stupně)</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Rychlost (metry za sekundu)</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

Tabulka 6.1: Seznam dostupných geografických informací ze senzorů.

14 Základnová převodní stanice sítě GSM
16 Viz [http://www.wigle.net/](http://www.wigle.net/)
Získaná data se používají při načítání bodů zájmu z databáze, aby se vybraly ty, které se nachází nejbližší aktuální pozici. Další využití je při výpočtu umístění bodů zájmu ve 3D scéně. V OpenGL odpovídá jedna jednotka velikosti jednomu metru. Pozice zařízení se nachází v počátku soustavy souřadnic 3D scény. Pro nastavení správných souřadnic vykresleného objektu je potřeba vypočítat rozdíl pozice tohoto objektu a aktuální lokace zařízení pro osy $x$ a $y$:

$$x = (\text{poiLon} - \text{deviceLon}) \cdot \text{DEG\_TO\_METERS} \cdot \cos(\text{poiLat} \cdot \text{DEG\_TO\_RAD})$$

$$y = (\text{poiLat} - \text{deviceLat}) \cdot \text{DEG\_TO\_METERS}$$

Výsledný rozdíl ve stupních je potřeba převést na metry, abychom získali požadované vykreslovací souřadnice. Při výpočtu je zohledněno zakřivení Zemského povrchu:

$$RADIUS = 6378137$$
$$CIRCUMFERENCE = 2 \cdot \pi \cdot RADIUS$$
$$DEG\_TO\_METERS = \frac{CIRCUMFERENCE}{360}$$

(6.10)

### 6.5 Vykreslení bodů zájmu


Načtení resp. aktualizace bodů zájmu z databáze proběhne vždy na začátku při spuštění renderera a pak vždy, když se uživatel přesune z místa poslední aktualizace o určitou vzdálenost a zároveň uběhne určitý čas. Implicitně je nastavena minimální vzdálenost 100 metrů a čas 45 sekund. Z databáze se načítají nejbližší POI od aktuální pozice mobilního zařízení a uloží se do kolekce spolu s vytvořenými grafickými objekty, které se budou renderovat. Aktualizací databáze lze provést i explicitně pomocí tlačítka „Refresh“ v kontextovém menu.

Pozice grafických objektů bodů zájmu ve scéně je potřeba také obnovovat, stejně tak jako dynamické textury, do kterých se vykresluje informace o aktuální vzdálenosti POI (viz níže). Tato aktualizace probíhá nezávisle na načítání dat z databáze a s mnohem vyšší frekvencí. Standardně je
nastavena minimální vzdálenost přesunu od posledního místa aktualizace na 20 metrů a časová prodleva je 15 sekund. Při aktualizaci dochází k přepočítání pozice, rotace a velikosti grafických objektů všech POI. Rovněž dochází k nastavení nové textury. Pozice objektu se vypočítá pomocí geolokace (viz kapitola 6.4).

Informace o jednom bodu zájmu jsou uloženy v instanci datové třídy POIdefault, která je potomkem abstraktní třídy POI. Objekt uchovává všechny potřebné informace uložené v databázi a dále také grafickou reprezentaci daného POI - instanci třídy Mesh. Tato třída obecně představuje jeden grafický prvek a všechny grafické objekty použité v aplikaci jsou od ní odvozené. Jedná se o třídy: FigureLine (úsečka), FigureCircle (kruh), FigurePlane (rovina), FigureCube (krychle), FigureGridCartesian (kartézská mřížka), FigureGridPolar (polární mřížka). Tyto grafické objekty se mohou sdružovat do skupin (viz obr. 6.9) - instance třídy MeshGroup. Tímto způsobem mohou tvořit stromovou strukturu. V rendereru jsou poté všechny objekty rekurzivně vykreslovány jako jeden celek. Třída Mesh obsahuje metody pro vykreslení objektu. Tyto metody používají volání knihovny OpenGL. Body zájmu se vykreslují od nejvzdálenějšího po nejbližší. Dále třída obsahuje metody pro práci s texturou a transformacemi objektu. Samotný 3D model potom tvoří seznam vrcholů, indexů a texturovacích souřadnic. Objekt třídy POIStoDraw uchovává v kolekcích všechny načtené body zájmu z databáze a jejich grafické modely.

Grafický objekt třídy POIdefault představuje jednoduchý čtverec s částečně transparentní texturou. Ten se vždy natáčí směrem k pozorovateli. Velikost objektu se nastavuje podle vzdálenosti od pozorovatele tak, aby byla na displeji vždy stejně velká, přičemž vzdálenější objekty jsou o něco menší než ty umístěné bliže k pozorovateli. Velikost se takto přepočítává, aby bylo možné pokaždé vidět i vzdálené POI, které by podle zákona perspektivy byly normálně příliš malé a nerozeznatelné. Textura grafického objektu obsahuje ikonu a dále text s názvem POI a vzdáleností. Textura se načítá z PNG souboru. Její rozměry musí být velikost mocniny čísla 2. Toto omezení určuje knihovna OpenGL ES. Text je do textury dokreslen dynamicky pomocí knihovní funkce. OpenGL ES standardně nepodporuje zobrazování fontů, a proto byl zvolen tento způsob zobrazení textu.

Obrázek 6.9: Stromová struktura grafických objektů.
Framework je možné rozšířit o další funkce a způsoby zobrazení bodů zájmu. Je možné si vytvořit vlastní grafickou reprezentaci Mesh a plně využít možností 3D zobrazení. Pro vytvoření vlastní modifikace bodu zájmu stačí implementovat potomka třídy POI a definovat si vlastní vzhled a funkce.

6.6 Databáze

Body zájmu se ukládají do lokální databáze SQLite. Import dat lze provádět v nastavení aplikace z GPX souboru nahaněm na paměťové kartě telefonu nebo dostupném přes URL adresu. Při výběru souboru z paměťové karty se zobrazí jednoduchý prohlížeč souborů (tzv. picker). Uživatel může procházet všechny adresáře na kartě a vybrat požadovaný soubor (viz obrázek 6.12b).

GPX (GPS Exchange Format) [35] je standardizovaný formát pro ukládání geografických dat. Používá XML schéma\(^\text{17}\). Svou strukturou je vhodný pro ukládání a sdílení bodů zájmu pro rozšířenou realitu. GPX formát má poměrně široké možnosti použití a kromě ukládání význačných geografických bodů umí zpracovávat i cesty a další geodata. Aplikace importuje z GPX souboru pouze body zájmu a několik potřebných informací. Používané schéma GPX souboru v aplikaci je následující:

```xml
<?xml version="1.0"?>
<gpx version="1.1" creator="ExpertGPS 1.1 - http://www.topografix.com">
  <wpt lat="49.152838" lon="16.688654">
    <ele>237</ele>
    <name>Letiště Tuřany</name>
    <desc><![CDATA[Mezinárodní letiště Brno-Tuřany.]]></desc>
    <link>http://www.airport-brno.cz</link>
    <type><![CDATA[Letiště]]></type>
  </wpt>
</gpx>
```

Při importu dat je potřeba parsovat GPX soubor. Načtené body zájmu se uloží do objektů třídy POIdatabase, která je potomkem třídy POI. Následně se uloží do databáze.


\(^{17}\) Viz [http://www.topografix.com/gpx.asp](http://www.topografix.com/gpx.asp)
Všechny potřebné CRUD operace nad daty se provádí v instanci třídy Database. Při načítání POI z databáze se berou jen ty, které mají aktivní kanál. Body zájmu se řadí do nejbližšího k nevzdálenějšímu, přičemž se kontroluje maximální počet zobrazovaných POI a maximální vzdálenost od aktuální pozice zařízení. Hodnota těchto maxim je uložena v uživatelském nastavení (viz kapitola 6.8). Při importu dat z GPX souboru se body zájmu ukládají do příslušných kanálů. Odpovídající kanál je určen položkou type v GPX souboru. Pokud daný kanál v databázi neexistuje, automaticky se vytvoří. Pokud není položka type nastavena, POI se uloží do kanálu „Default“. Body zájmu je možné přidávat i ručně v nastavení aplikace. Potom se POI uloží do kanálu „My POI“. 

6.7 Detekce kolizí

Všechna mobilní zařízení se systémem Android používají dotykový displej, jehož prostřednictvím se ovládají aplikace. V pohledu rozšířené reality se zobrazují grafické objekty, které reprezentují určité body zájmu. V mé aplikaci se vykreslí jednoduché 2D ikony s textovým popiskem. Po kliknutí na ikonu se zobrazí okno s detailním popisem POI, jak bylo popsáno v kapitole 6.2. Detekce kliknutí na ikonu je netriviální, protože ikony jsou umístěny v 3D prostoru, zatímco displej telefonu je dvourozměrný.

Třída VirtualLayerSurface odchytává události kliknutí na displeji telefonu ve vlastním vláknu. Pokud je událost kliknutí zachycena, objekt zašle zprávu rendereru, ten událost zpracuje a pokud je detekováno kliknutí na POI, zašle objektu zpět odpověď s informací o daném bodu zájmu. Instance třídy VirtualLayerSurface potom zobrazí dialogové okno.


Metoda ray picking v [36] funguje na principu trasování paprsku. Do 3D scény se vyšle paprsek a kontroluje se jeho průnik s danými objekty. Nejjdříve je potřeba vypočítat směr šíření paprsku ve scéně. Startovní pozice paprsku je v počátku soustavy souřadnic a jeho směr je určen vektorem. Směr se vypočítá pomocí knihovní funkce gluUnProject, která mapuje souřadnice displeje na objektové souřadnice ve 3D. Tato funkce používá k výpočtu projekční matici, modelovou matici, rozměry displeje a souřadnice místa kliknutí.
Pokud známe směr paprsku, vypočítáme délku jednoho kroku šíření paprsku na jednotlivých osách podle následujícího vztahu:

\[
rayDeltaX = ray[0] \cdot coef \\
rayDeltaY = ray[1] \cdot coef \\
rayDeltaZ = ray[2] \cdot coef \\
coef = \frac{STEP^2}{\sqrt{(ray[0])^2 + (ray[1])^2}}
\]

(6.11)

dle \( ray \) je vektor určující směr paprsku, \( ray[0] \) a \( ray[1] \) se nesmí rovnat nule, \( STEP \) je konstanta značící požadovanou délku kroku v jednotkách OpenGL (v aplikaci jsem použil hodnotu 10) a \( rayDelta \) jsou proměnné určující vypočitanou délku jednoho kroku na dané souřadnici. Šíření paprsku probíhá v cyklu, dokud nedojde ke kolizi s objektem nebo zadní ořezávací rovinou. V každém kroku se kontroluje kolize paprsku s obalovým objektem daného grafického modelu bodu zájmu. Pokud je kolize detekována, renderer zašle zprávu příslušnému objektu, který kolizi zpracuje. Princip ray pickingu je znázorněn na obrázku 6.10.

![Obrázek 6.10: Ray picking pro detekci kliknutí na objekt.](image-url)
6.8 Uživatelské rozhraní

Uživatelské rozhraní aplikace bylo navrženo s ohledem na snadné a intuitivní ovládání. Po spuštění aplikace se inicializuje aktivita `MainActivity`, která zobrazuje hlavní menu aplikace (viz obr. 6.11a). Zde se nachází seznam všech kanálů s ikonou, názvem a počtem dostupných POI z aktuální lokace.


![Diagram aktivit](image.png)

Obrázek 6.14: Diagram aktivit.

První možný způsob zobrazení POI je pohled rozšířené reality, který se spouští aktivitou `ARactivity` (viz kapitola 6.2). Zde se vykresluje rozšířená realita s aktuálně dostupnými body zájmu (viz obr. 6.3). Dalším způsobem zobrazení je textový seznam, který představuje aktivita `ListActivity`. V ní se vypisují detailní informace o všech dostupných bodech zájmu (viz obr. 6.11b). Po kliknutí na vybraný bod zájmu se zobrazí kontextové menu s možnostmi navštívení webové stránky, sdílení POI nebo odstranění z databáze (viz obr. 6.12a).
Obrázek 6.11: a) Vlevo hlavní menu aplikace.

b) Vpravo zobrazení bodů zájmu v seznamu.


b) Vpravo výběr GPX souboru na SD kartě.


aplikace a jeho barvy. Jednoduchým způsobem je také možné přidat nové jazykové mutace programu nebo upravovat stávající.

Každá grafická ikona v aplikaci má tři různé rozměry. Mobilní telefony s displejem o vysokém rozlišení (hdpi) používají standardní rozměry ikon 72 × 72, střední displeje (mdpi) 48 × 48 a ty nejmenší (ldpi) potom 36 × 36. Obecně platí, že rozměry hdpi jsou rovny 1,5 násobku mdpi a ldpi jsou rovny 0,75 násobku mdpi. Většina ikon použitých v aplikaci byla stažena z oficiálního repozitáře Android projektu.
7 Vývoj a testování

Navržený systém byl vytvořen pro platformu Google Android s využitím standardní sad o vývojových nástrojů SDK. Použil jsem vývojové API level 8, které odpovídá verzi operačního systému Android 2.2 Froyo. Zdrojový kód byl napsán v jazyce Java a jako vývojové prostředí jsem použil program Eclipse s využitím pluginu ADT (Android Development Tools).

Testování systému probíhalo průběžně po celou dobu vývoje a jeho výsledky se promítaly do úprav v samotné implementaci. Aplikace byla vyvíjena a testována přímo na mobilních telefonech HTC Desire a LG P-990 s OS Android 2.2 Froyo v režimu ladění pomocí připojeného USB kabelu. Tento způsob testování je efektivnější než použití emulátoru. Důvodem je mnohem vyšší rychlost instalace aplikace do zařízení, její běh a především možnost přímého využití senzorů v telefonu pro testování. Emulátor systému Android standardně nepodporuje senzory jako např. akcelerometr nebo GPS. Simulace videokamery je rovněž problematická. Ačkoliv lze tyto senzory a zařízení pomocí speciálních nástrojů nasimulovat, použití reálného zařízení je pro účely vývoje a testování mnohem jednodušší, výhodnější a rychlejší eventualitou. Díky použití reálných zařízení při testování je snadno možné odhalit výkonnostní nedostatky aplikace.

7.1 Ladící nástroje


Při vývoji jsem dále použil standardní debugger vývojového prostředí Eclipse, s jehož pomocí lze sledovat stav paměti, krokovat spuštěný program, sledovat logy a případně nalézt potencionální chyby v implementaci aplikace.

Zajímavým a užitečným nástrojem je program Monkey, který se zaměřuje na testování uživatelského rozhraní. Vytváří pseudonáhodné akce jako např. stisknutí tlačítka, kliknutí, gesto a další systémové události. Program tak testuje stabilitu vyvíjené aplikace (tzv. stress test).
7.2 Unit testy

Při testování některých funkcí aplikace jsem použil několik jednoduchých Unit testů, pomocí nichž jsem ověřoval např. správné překreslení obrazovky a zachování kontextu aktivity při změně orientace telefonu, korektní chování při updatu aplikace či její stabilitu v případě, že nejsou dostupné zdroje informací jako internet, GPS, videokamera apod. Dále jsem také ověřoval správné zobrazení grafického uživatelského rozhraní na zařízeních s rozdílnými velikostmi rozlišení displeje. K tomu byl využit emulátor systému Android.

Unit testy [37] obecně zahrnují nástroje a metody, jejímž cílem je ověření správné funkčnosti dílčích částí (jednotek) zdrojového kódu aplikace. Testovací jednotkou bývá v objektově orientovaných programech obvykle třída, jejíž součástí jsou testované metody. Během testování se ověřuje správná funkčnost jednotlivých případů (částí), které by měly být na sobě nezávislé. Tyto části jsou od ostatních oddělené a z toho důvodu je často potřeba vytvořit pomocné objekty (tzv. mock objekty), které simulují předpokládaný kontext, v němž daná část pracuje.


7.3 Testování na platformě Android

Během vývoje nastal problém, jakým způsobem otestovat funkčnost aplikace rozšířené reality v místnosti, kde není GPS signál. Pro tento účel jsem použil aplikaci My Fake Location, která dokáže jednoduše simulovat GPS lokaci v telefonu.

Velmi důležitým aspektem pro správné zobrazení rozšířené reality je kalibrace kamery. V nastavení perspektivy OpenGL se nastavuje zorný úhel kamery. Tento úhel lze zjistit pomocí knihovní funkce. Při vývoji jsem experimentálně ověřoval, zda daný úhel skutečně odpovídá zornému úhlu videokamery použitého mobilního zařízení, aby bylo zobrazení rozšířené reality co nejpřesnější.


18 Viz http://pivotal.github.com/robolectric/
19 Viz http://code.google.com/p/robotium/
Součástí testování aplikace je také ověření správného chování v případě, že je běh programu přerušen nějakou vnější událostí. Příkladem takové události může být např. příchod telefonní hovor, SMS zpráva, spuštěný budík či jakákoli událost vyvolaná jiným programem v telefonu. Některé tyto události můžeme vyvolat uměle za účelem testování v ladícím nástroji DDMS, případně použitím programu telnet. V případě telefonního hovoru se aplikace pozastaví, přeneše se na pozadí a uvolní se všechny sdílené zdroje. Po ukončení hovoru se opět spustí. Pokud přijde SMS zpráva nebo nastane jiná událost, která nespouští novou aktivitu na popředí, aplikace běži dál a tato událost na její běh nemá žádný vliv. Pouze se zobrazí běžná notifikace ve stavové liště.

7.4 Testování v terénu

Cílem práce bylo vytvořit aplikaci rozšířené reality zaměřenou na leteckou navigaci. Nebyla možnost vytvořený systém testovat v letadle. Aplikaci jsem však vyzkoušel v terénu a to jak ve městě, tak na vyvýšeném místě na kopci, abych mohl alespoň částečně otestovat správnou funkčnost zobrazování nadmořské výšky u bodů zájmu (viz obr. 7.1). Pro účel testování jsem vytvořil sadu POI s význačnými místy v okolí města Brna, jako jsou památky, známé budovy apod. Pro získání geodat jsem použil projekt OpenStreetMaps20 a Google Maps21. Letecká data jsem získal z veřejně dostupné databáze letišť22. Tyto body zájmu jsou spolu s leteckými daty pro celou Českou Republiku uloženy v GPX souborech na přiloženém datovém médiu této diplomové práce.


Při testování jsem také zkoušel použít namísto GPS souřadnic geografická data získaná pomocí BTS nebo WiFi sítě. Jejich značná nepřesnost byla patrná na první pohled. Využití dat ze sítě není pro navigaci vhodné a může posloužit pouze jako hrubý odhad. Dalším testovacím kritériem v terénu byla výdrž baterie. Spotřeba energie je v případě aplikace vyšší než obvykle, jelikož zařízení používá

20 Viz http://www.openstreetmap.org/
21 Viz http://maps.google.com/
22 Viz http://www.aerobaze.cz/gps/
řadu senzorů, GPS a videokameru. Při plně nabitě baterii je aplikace schopna běžet na průměrném mobilním zařízení pouze několik hodin.

Obrázek 7.1: Testování aplikace v terénu. Tato ukázková fotografie byla pořízena z Červeného kopce v Brně.

7.5 Porovnání s existujícími programy


Setkal jsem se i s několika frameworky, které usnadňují vývoj aplikací rozšířené reality se zaměřením na polohové senzory. Jeden takový framework jsem potom implementoval v rámci této diplomové práce. Za zmínku stojí Mixare engine, který jsem vyzkoušel a porovnal s vlastní aplikací.
Mixare získává body zájmu z online služeb podobně jako výše zmíněné aplikace. Dokáže rovněž pracovat s nadmořskou výškou. K vykreslování ovšem nepoužívá knihovnu OpenGL.

Mezi hlavní výhody mnou vytvořené aplikace oproti ostatním patří především zohlednění nadmořské výšky, možnost použití libovolných bodů zájmu, vykreslování virtuální vrstvy v 3D prostoru pomocí knihovny OpenGL a nezávislost na internetovém připojení. Existuje i několik aplikací, které zpracovávají rozšířenou realitu jinými metodami než pomocí polohových senzorů. Často jde o prostory používané hraní ARToolKit a její mutace. Tyto aplikace jevim bliže nezkoumal, jelikož fungují na jiném principu, než řeší tato diplomová práce.

7.6 Publikace na Android Marketu


Aplikace rozšířené reality implementovaná v rámci této diplomové práce je zdarma přístupná na Android Marketu23 pod názvem Caramelle Augmented Reality.

23 Aplikace je dostupná na URL adrese: https://market.android.com/details?id=net.jestrab.caramelle
8 Závěr


Příspěvek ke vývoji projektu byl pro mě velkým přínosem. Podařilo se mi vytvořit použitelnou mobilní aplikaci a publikovat ji. Zdokonalil jsem se v programování v jazyce Java, ve vývoji mobilních aplikací na platformě Android a nabyl mnoho nových zkušeností zvláště pak v oblasti rozšířené reality a práci s grafickou knihovnou OpenGL ES. V praxi jsem mohl využít nejen vědomosti dlouho nabývané v oblasti počítačové grafiky, ale také z oboru zpracování signálů (pro redukci šumu signálu ze senzorů). Výsledky práce jsem úspěšně prezentoval na konferenci Student EEICT 2011, kterou pořádala Fakulta elektrotechniky a komunikačních technologií společně s Fakultou informačních technologií v Brně. Projekt získal první místo v kategorii posterové sekce. Dále jsem práci prezentoval na soutěži Media Contest (nejlepší multimediální demo) pořádanou Ústavem počítačové grafiky a multimedií na FIT v Brně a příslušná je i účast na konferenci Openmobility 2011 na Fakultě informatiky Masarykovy univerzity v Brně.

Vývoj projektu bude pokračovat i nadále. Především bych chtěl zaměřit na vylepšování vytvořeného frameworku a použit jej v dalších aplikacích. Z hlediska celkové robustnosti projektu bylo vhodné v budoucnu některé části aplikace znovu analyzovat a pokusit se navrhnout efektivnější řešení problému s využitím zkušeností z předchozí implementace. Projekt byl od počátku vyvíjen s cílem vytvořit plně funkční, životaschopnou a v praxi použitelnou aplikaci, jež bude přístupná široké veřejnosti. Díky tomu by neměl být problém najít další motivaci k postupnému vylepšování aplikace k dokonalosti.

V budoucnu bych se zaměřil především na vylepšení následujících funkcí:

- Vizualizace bodů zájmu na mapě.
- Vylepšení grafického uživatelského rozhraní především v pohledu rozšířené reality.
- Vytvoření grafického kompasu, přehledné mapky bodů zájmu apod.

Vývoj projektu bude pokračovat i nadále. Především bych chtěl zaměřit na vylepšování vytvořeného frameworku a použit jej v dalších aplikacích. Z hlediska celkové robustnosti projektu bylo vhodné v budoucnu některé části aplikace znovu analyzovat a pokusit se navrhnout efektivnější řešení problému s využitím zkušeností z předchozí implementace. Projekt byl od počátku vyvíjen s cílem vytvořit plně funkční, životaschopnou a v praxi použitelnou aplikaci, jež bude přístupná široké veřejnosti. Díky tomu by neměl být problém najít další motivaci k postupnému vylepšování aplikace k dokonalosti.

V budoucnu bych se zaměřil především na vylepšení následujících funkcí:

- Vizualizace bodů zájmu na mapě.
- Vylepšení grafického uživatelského rozhraní především v pohledu rozšířené reality.
- Vytvoření grafického kompasu, přehledné mapky bodů zájmu apod.
• Přizpůsobení zobrazení grafických prvků natočení telefonu.
• Podpora více formátů pro import bodů zájmu (KML, WPT, JSON apod.)
• Vytvoření dynamických kanálů, které budou zobrazovat body zájmu z různých webových služeb jako např. Wikipedia, Foursquare, Last.fm atd. Díky tomu bude použití aplikace ještě snadnější a nebude přímo nutné importovat body zájmu manuálně.
Literatura


Příloha A

Obsah CD

- `/bin/` - instalační soubor aplikace
- `/doc/` - text technické zprávy a návod k použití aplikace
- `/gpx/` - testovací data s vybranými body zájmy
- `/prezentace/` - plakát a demonstrační video
- `/src/` - zdrojové soubory Android projektu