VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

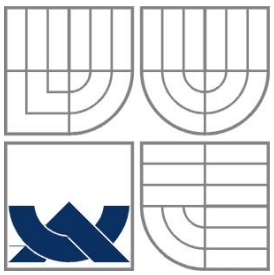
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZŠÍŘENÁ REALITA PRO PLATFORMU ANDROID

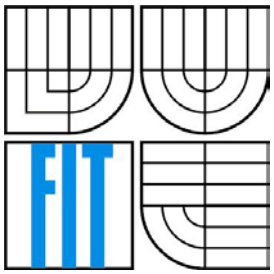
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PETR NOHEJL
AUTHOR

BRNO 2011



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

ROZŠÍŘENÁ REALITA PRO PLATFORMU ANDROID

AUGMENTED REALITY FOR ANDROID PLATFORM

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR NOHEJL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ LÁNÍK

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 zmiňuje detaily 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

Mobilní trh prochází velkými evolučními proměnami. Rychle se rozvíjející 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 průhledový 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.

V druhé kapitole jsou vysvětleny principy rozšířené reality a metody její interpretace. Ve třetí kapitole je popsána mobilní platforma Google Android, její architektura a základní principy důležité pro vývoj. Čtvrtá kapitola se zabývá grafickou knihovnou OpenGL ES, kterou používá navržená aplikace pro zobrazení virtuální scény. Pátá kapitola pojednává o návrhu frameworku rozšířené reality, na kterém je postavena samotná aplikace. V šesté kapitole jsou potom podrobně vysvětleny detaily implementace vybraných problémů. V sedmé kapitole je popsáno testování aplikace a způsob její distribuce. V závěrečné kapitole jsou nakonec zhodnoceny dosažené výsledky a možnosti vývoje aplikace v budoucnu.

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 vyvíjet.

[1] Mobilní telefony s pokročilými funkcemi

2 Rozšířená realita

„Rozšířená realita (augmented reality, dále jen AR) představuje mezistupeň mezi realitou skutečnou a virtuální. Rozšířená realita je doplněním obrazu skutečnosti o uměle doplněné obrazce či jiné informace. V současnosti je nejčastějším provedením rozšířené reality zobrazení skutečného obrazu na displeji a jeho doplnění o počítačem dodané informace.“ [9] Základním předpokladem rozšířené reality je, že reálný obraz a virtuální objekty se vykreslují v reálném čase.

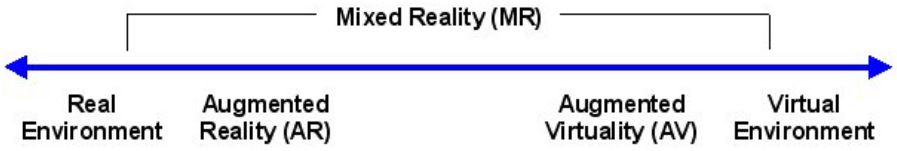
V této kapitole budou popsány základní principy rozšířené reality důležité pro návrh celého systému. Popíšeme si dva nejčastější způsoby realizace rozšířené reality na mobilních zařízeních. Představíme si hardwarová zařízení a senzory používané pro trasování virtuálních objektů, dále budou popsány možnosti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

2.1 Definice rozšířené reality

Virtuální i rozšířená realita [13] se zaměřují na vykreslování informací do 3D prostoru. Rozdíl mezi nimi je takový, že virtuální realita zobrazuje pouze umělé prostředí, zatímco rozšířená realita zobrazuje reálné prostředí doplněné umělými objekty či textovými informacemi. Definice rozšířené reality vychází z definice virtuální reality - „počítačem vytvořené prostředí, které je trojrozměrné, interaktivní a jímž je uživatel obklopen“. Paul Milgram a Fumio Kishino [11, 12] umísťují rozšířenou realitu na ose virtuálního kontinua mezi reálné a virtuální prostředí (viz obr. 2.1). Mezi nimi se nachází i rozšířená virtualita (augmented virtuality), která je definována jako umisťování obrazů reálných objektů do virtuálního prostředí. Rozšířená realita a rozšířená virtualita jsou souhrnně označovány jako smíšená realita (mixed 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 povzdálí 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.

Nejrozšířenější knihovnou pro vývoj aplikací s rozšířenou realitou využívající tuto metodu je ARToolKit [14]. Vyvinul ji Japonec Dr. Hirokazu Kato z Nara Institute of Science and Technology v roce 1999. Vydána byla jako Open source Univerzitou ve Washingtonu. Knihovna umožňuje rozpoznávání trasovacích značek, trasování pozice a rotace kamery metodami počítačového vidění, kalibraci kamery, tvorbu vlastních značek, podporuje knihovnu OpenGL pro zpracování 3D obrazu, je multiplatformní a má mnoho mutací včetně podpory platformy Android.

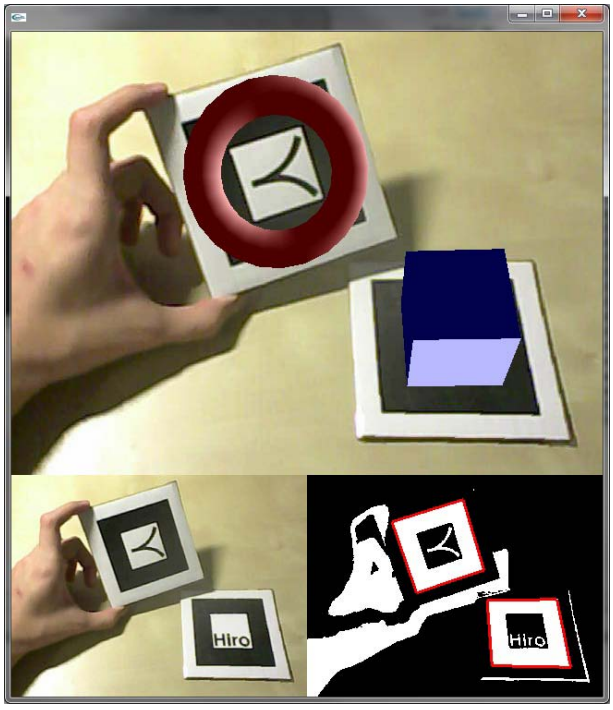
V následujícím textu budou popsány fáze zpracování obrazu v reálném čase pro knihovnu ARToolKit:

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 čtverhranné 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).
Nahoře 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.

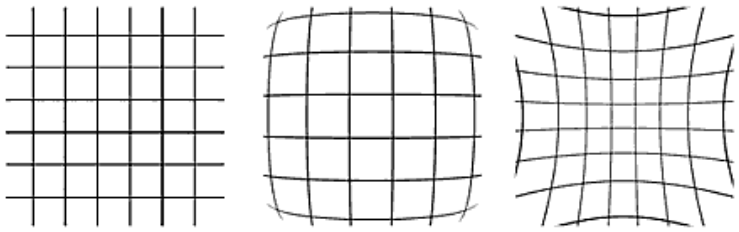
Obrázek 2.3: Aplikace AR, využívající polohové senzory [15].

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.

Kosmickou část tvoří družice obíhající v orbitální výšce 20200 kilometrů nad povrchem Země. Satelity obíhají Zemi na šesti polárních drahách v inklinaci[2] 55°. Doba oběhu je 11 hodin a 58 minut, rychlost pohybu je okolo 3.8 km/s. Družice jsou rozmístěny rovnoměrně a vždy je jich vidět alespoň šest, a to z jakéhokoliv místa na Zemi. Každý satelit vlastní vysílač, přijímač a velmi přesné atomové hodiny. Vysílač zasílá data uživatelům a řídícímu středisku. Přijímač získává data z řídícího střediska a podle nich pak řídí samotnou družici, např. koriguje oběžnou dráhu. Průměrná životnost družice je přibližně 10 let.

Řídící segment spravuje a řídí chod kosmického segmentu, tedy družic 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.

Uživatelskou část tvoří GPS přijímače, které používají běžní uživatelé. Přijímače jsou většinou zabudovány v různých přenosných zařízeních. Jejich součástí je anténa, která musí fungovat na stejné frekvenci jako družice.

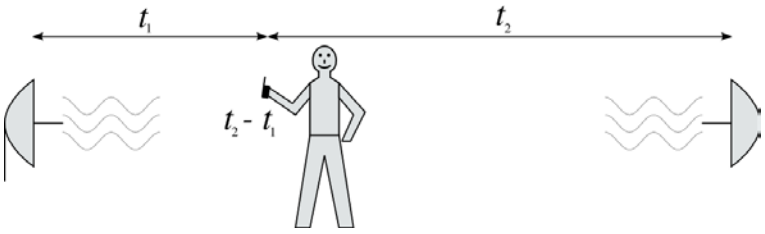
Každý satelit vysílá v podobě elektromagnetických vln informace o své pozici a času atomových hodin. Přijímač pak tyto informace zpracovává. Všechny družice vyšlou signál v přesně stanovený čas. K určení polohy uživatele se využívá časového rozdílu, s jakým zpožděním byl přijat signál od jednotlivých satelitů, tzv. TDOA (Time difference of arrival). Pro výpočet zeměpisné šířky a délky jsou zapotřebí informace alespoň ze tří satelitů, pro výpočet nadmořské výšky jsou potřeba alespoň čtyři satelity. Čím více satelitů přijímač využívá, tím je poloha přesnější. Pro správný příjem signálu je zapotřebí přímé viditelnosti oblohy. GPS nefunguje uvnitř budov, pod zemí, pod vodou, v husté zástavbě apod. GPS určuje polohu s přesností na metry. Princip určení polohy je znázorněn na obrázku 2.5.

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

Při zapnutí GPS přístroje musí nejdřív proběhnout inicializace. Až poté je možné navigovat. Inicializace trvá několik desítek sekund až minut. Dochází při ní k načítání informací o jednotlivých družicích. Načítají se i další potřebná data. Tyto informace se nazývají almanach.

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

[2] Sklon dráhy tělesa k základní rovině - rovníku



Obrázek 2.5: Určení polohy v jednorozměrném prostoru pomocí dvou satelitů [20].

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 $arctg\ x \cdot y^{-1}$, která určuje kurs vzhledem k ose *x*. Senzory magnetického pole, používané v mobilních zařízeních, měří magnetickou indukci v jednotkách Tesla.

Hlavní součástí kompasu je magnetoresistivní senzor vyrobený z feromagnetického materiálu. Ten je zmagnetizovaný a v případě nulového vnějšího vlivu magnetického pole má určitý odpor. Vnější magnetické pole však ovlivní původní směr zmagnetování a tím dojde ke změně odporu. Pak stačí měřit rozdíly v napětí při konstantním proudu. Aby bylo možné určit, z jaké strany působí magnetické pole, pokrývá se feromagnetický pruh řadou hliníkových pásků ve sklonu 45° vůči osám. Dále se využívá tzv. můstkového zapojení. Elektronický kompas také musí umět pracovat při různých teplotách prostředí, protože s rostoucí teplotou roste i odpor. V úvahu je nutno brát i možnost změny orientace původního zmagnetování senzoru silným magnetickým polem. Některé kompasy jsou schopné měřit i v nevodorovných polohách, tzv. tříosé kompasy [21].

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 $9{,}823\ m \cdot s^{-2}$. 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 tříelektrodového 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 ose 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. smartphony[3]. Tuto metodu 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.

Opticky průhledové náhlavní displeje jsou mnohem složitější. Nevyužívají kameru, ale polopropustné zrcátko, přes které pozorovatel sleduje reálnou scénu. Na zrcátko jsou potom promítány virtuální objekty. Pro správné zobrazení virtuálních objektů na odpovídajících místech je nutno provést kalibraci náhlavního displeje.

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 multimediálních 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[4] 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 listopadu roku 2007 Google oficiálně představil platformu Android a inicializoval vznik konsorcia Open Handset Alliance [2]. Hlavním cílem sdružení je prosazovat otevřené standardy pro mobilní zařízení. Zakladateli aliance byli výrobci telefonů, hardware, software, telekomunikační společnosti a další významné společnosti jako např. Google, Intel, Nvidia, HTC, LG, Motorola, Samsung, T-mobile a další. Prvním produktem sdružení byl právě systém Android a jeho vznik odstartoval velký boj konkurenčních společností na mobilním trhu.

V září roku 2008 byla oficiálně představena první verze operačního systému Android, založená na Linuxovém jádře verze 2.6. Systém byl vydán jako Open source pod licencí Apache 2.0 a GPL v2 včetně vývojového prostředí Android SDK (Software Development Toolkit). V říjnu téhož roku byl uveden první telefon s operačním systémem Android. Jednalo se o HTC Dream (označované též jako T-mobile G1).

První aplikace vznikly díky soutěži Android Developer Challenge, která měla přilákat nové vývojáře. Společnost Google do soutěže investovala desítky miliónů dolarů. Dosud proběhla dvě kola soutěže, první v roce 2008, další v následujícím roce 2009.

Společnost Google rovněž vytvořila online systém pro publikování aplikací, tzv. Android Market. Uživatelé si zde mohou stahovat dostupné aplikace zdarma nebo za poplatek. A vývojáři zase mohou své programy publikovat či prodávat.

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**

  První verze vydaná 23. září 2008.
- **Android 1.1**

  Vydána v březnu 2009. Hlavní změny: vylepšení uživatelského rozhraní, úprava některých vestavěných aplikací a podpora vyhledávání hlasem.
- **Android 1.5 (Cupcake)**

  Vydána v dubnu 2009. Hlavní změny: možnost nahrávání videa, podpora Bluetooth profilů, vylepšená softwarová klávesnice s automatickým dokončováním slov, podpora widgetů, kopírování 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)**

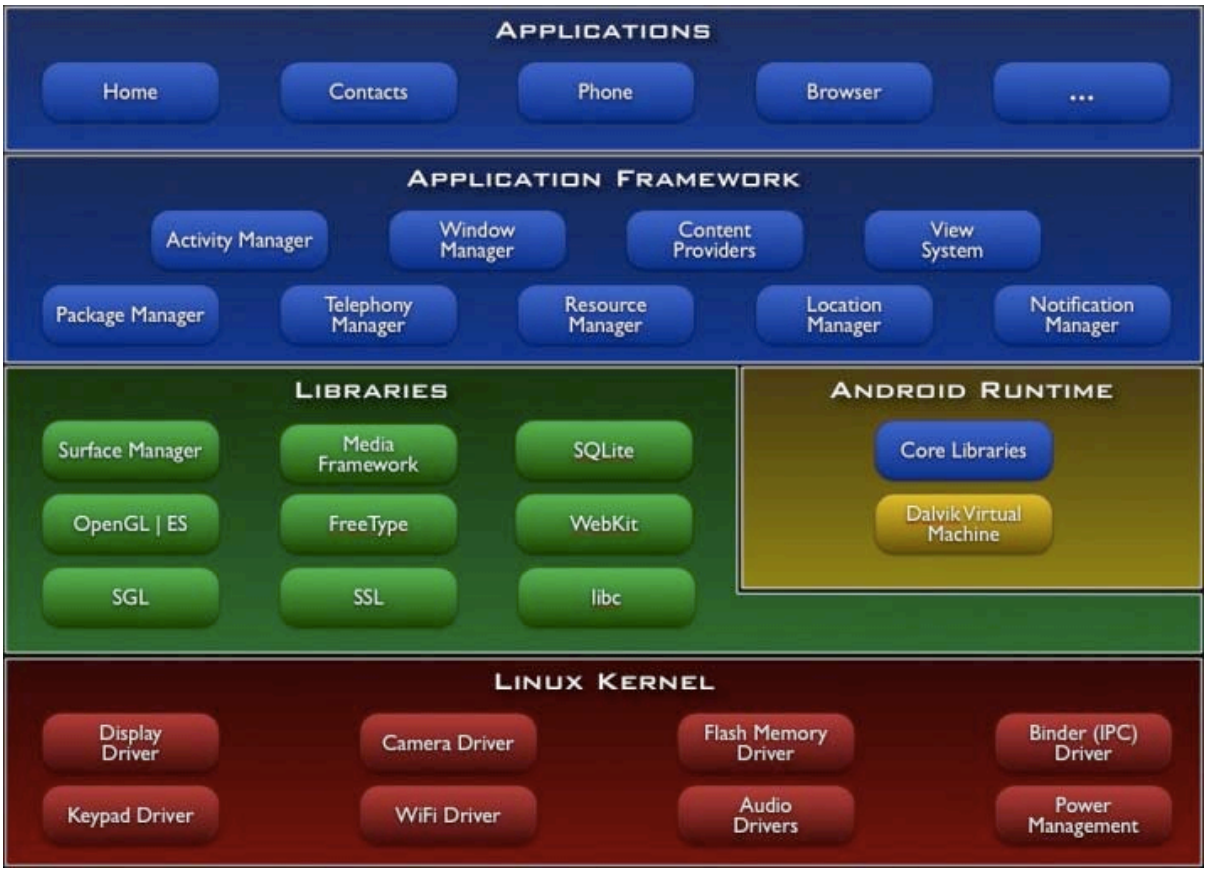
  Vydána v květnu 2010. Hlavní změny: možnost instalace aplikací na paměťovou kartu, optimalizace pro hardware, JIT kompilátor, podpora tetheringu a podpora OpenGL ES 2.0.
- **Android 2.3 (Gingerbread)**

  Vydána v prosinci 2010. Hlavní změny: podpora WebM, NFC, SIP protokolu, vylepšené uživatelské rozhraní a lepší podpora nativního kódu.

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].



Obrázek 3.1: Architektura platformy Android [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

Android runtime [4] obsahuje systémové knihovny a virtuální stroj Dalvik [5]. Každá aplikace běží v samostatném procesu a ve vlastní instanci virtuálního stroje. Díky tomu Android umožňuje běh více aplikací najednou. Operační systém se automaticky stará o uzavírání běžících aplikací v případě nedostatku systémových zdrojů.

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

Android obsahuje řadu knihoven [4] v jazyce C/C++. Tyto knihovny využívají různé komponenty systému Android a jsou dostupné programátorům přes aplikační rozhraní. Mezi hlavní knihovny patří:

- **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

Aplikační framework [4] poskytuje vývojářům široké možnosti využití vlastností systému pro vytváření vlastních aplikací. Programátoři mají plný přístup k API stejně jako systémové aplikace. Aplikační rozhraní bylo navrženo s ohledem na jednoduché a opakované použití komponent. Každá aplikace může poskytnout své schopnosti ostatním aplikacím, které je mohou využívat. Díky tomuto mechanizmu lze rovněž jednoduše nahrazovat existující aplikace za jiné alternativy. Mezi základní součásti aplikačního rozhraní patří:

- **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.

SDK (Software development kit) je běžná sada vývojových nástrojů. Zahrnuje: vývojové prostředí, emulátor, debugger, sadu knihoven, dokumentaci, ukázkové programy a další. SDK umožňuje vývoj aplikací v jazyce Java. Nevyužívá však žádného ze standardů jako např. Java SE nebo Java ME. Rovněž používá nestandardní virtuální stroj pro běh aplikací - Dalvik. SDK podporuje desktopové operační systémy Windows, GNU Linux a Mac OS, ve kterých lze vyvíjet. Oficiálně podporovaným vývojovým prostředím je Eclipse s využitím pluginu ADT (Android Development Tools).

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 manifest [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

Sdělení (*Intents*) [7] zajišťují vzájemnou komunikaci mezi jednotlivými komponentami v rámci aplikace, nebo v rámci celého systému. Umožňují přepínání mezi obrazovkami. Sdělení obsahuje typ operace (např. Main, View, Edit, Pick a další), která se má provést, a vstupní data definována formou URI (Uniform resource identifier). Příkladem sdělení může být odeslání e-mailové zprávy, volání telefonního čísla, nebo vyhledávání na webu.

Rozlišujeme implicitní a explicitní sdělení. Implicitní neobsahují, na rozdíl od explicitních, definici cíle. Zpravidla se používají pro komunikaci s jinými aplikacemi. Filtr sdělení (*Intent filter*) určuje typ implicitního sdělení, které je schopna aktivita zpracovat. Tyto filtry se definují v manifestu aplikace (viz kapitola 3.4.5). Například aplikace zobrazující mapy zpřístupní ve svém manifestu informaci, že umí zpracovat akci typu View pro geodata. Jiná aplikace potom zašle požadavek pro spuštění aktivity s určitým typem sdělení - např. zobrazení místa na mapě. Systém projde všechny aplikace a vybere aktivitu, která nejvíce vyhovuje danému typu sdělení - v našem příkladě tedy zobrazení bodu na mapě.

Přijímač broadcastů (*Broadcast intent receiver*) odchytává zprávy různých událostí v systému (např. příchozí SMS, vybitá baterie, zmáčknutí spouště fotoaparátu). Přijímače se obvykle definují v manifestu aplikace. Na základě přijetí zprávy lze potom provést určitou akci (např. zobrazení notifikace), nebo spustit danou aktivitu. Každá aplikace může posílat své vlastní broadcast 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:

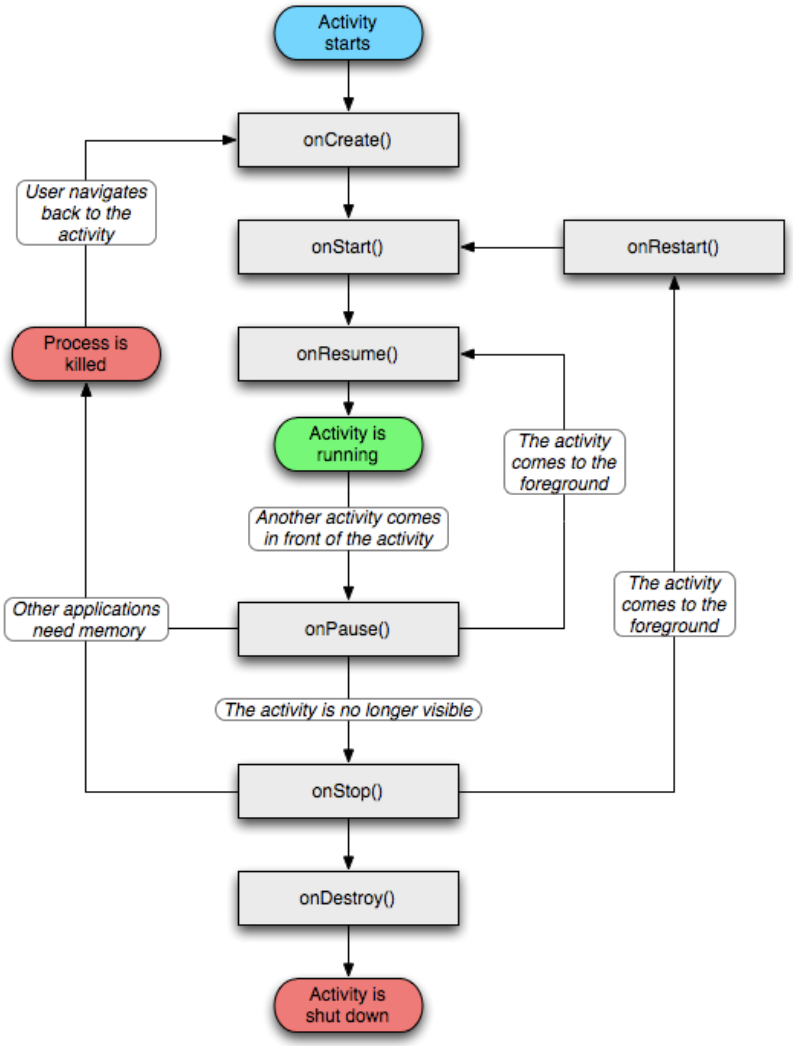
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.mypackage"
   android:versionCode="1"
   android:versionName="1.0">
   <application
      android:icon="@drawable/icon"
      android:label="@string/app_name">
      <activity 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*.

Každá aktivita má svůj životní cyklus [6, 7] a může se vyskytovat v několika stavech (viz obr. 3.2). Běžící aktivita (*running*) je aktivní, viditelná na popředí. Pozastavená aktivita (*paused*) je neaktivní, překrytá jinou aktivní. Zastavená aktivita (*stopped*) je uživateli skryta. Obě si však zachovávají svůj kontext i data. V případě nedostatku systémových zdrojů mohou být ukončeny (*destroyed*). Ukončování aktivit řídí systém automaticky podle určitých pravidel.

Služby mají také svůj životní cyklus podobný aktivitám.



Obrázek 3.2: Životní cyklus aktivity [6].

3.6 Uživatelské rozhraní

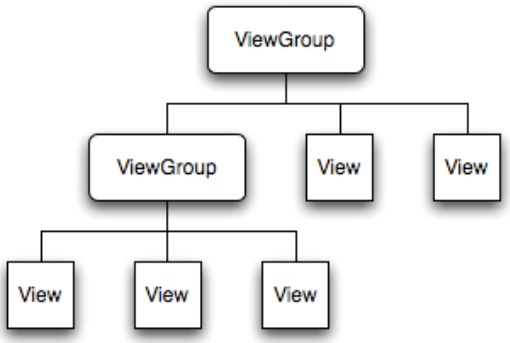
Uživatelské rozhraní [8] aplikace lze vytvářet přímo v programovém kódu nebo pomocí XML. V programovém kódu lze grafické rozhraní dynamicky měnit za běhu dle potřeby. Uživatelské rozhraní pomocí XML je naopak přehlednější, protože prvky rozhraní jsou zcela odděleny od kódu. V samotném kódu lze potom snadno přistupovat k těmto prvkům a pracovat s nimi.

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, zaškrtávací pole nebo seznamy. Systém poskytuje řadu těchto prvků, ale je možné vytvářet i vlastní.

Rozmístění prvků na obrazovce zajišťují tzv. *Layouty* (*ViewGroups*). Jsou to kontejnery pro jednotlivé prvky *Views*, které lze různým způsobem kombinovat v hierarchickém stromě. Každý XML soubor s uživatelským rozhraním obsahuje kořenový prvek typu *View* nebo *ViewGroup*. Mezi nejpoužívanější *Layouty* patří:

- **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.



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 platforem i programovacích jazyků včetně platformy Android a jazyku Java.

V této kapitole bude popsána grafická knihovna OpenGL ES, kterou využívá engine rozšířené reality, implementovaný v rámci této diplomové práce. Bude zde popsána struktura knihovny, její verze a podpora na platformě Android. Dále se budeme zabývat způsobem zobrazování 3D grafiky a zejména také geometrickými transformacemi, jejichž znalost je klíčová pro implementaci trasování kamery v rozšířené realitě.

4.1 Struktura knihovny

Rozhraní knihovny OpenGL je založeno na modelu klient-server. Díky tomu lze spustit program na jiném počítači než na tom, kde se příkazy vykonávají. Knihovna funguje jako stavový stroj a pomocí volání funkcí či procedur řídí množinu vykreslovacích operací. Těchto příkazů definuje OpenGL cca 250. Nepoužívá tedy metodiku objektově orientovaného programování.

Vykreslování obrazových dat se provádí do obrazových rámců (tzv. framebuffery). 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

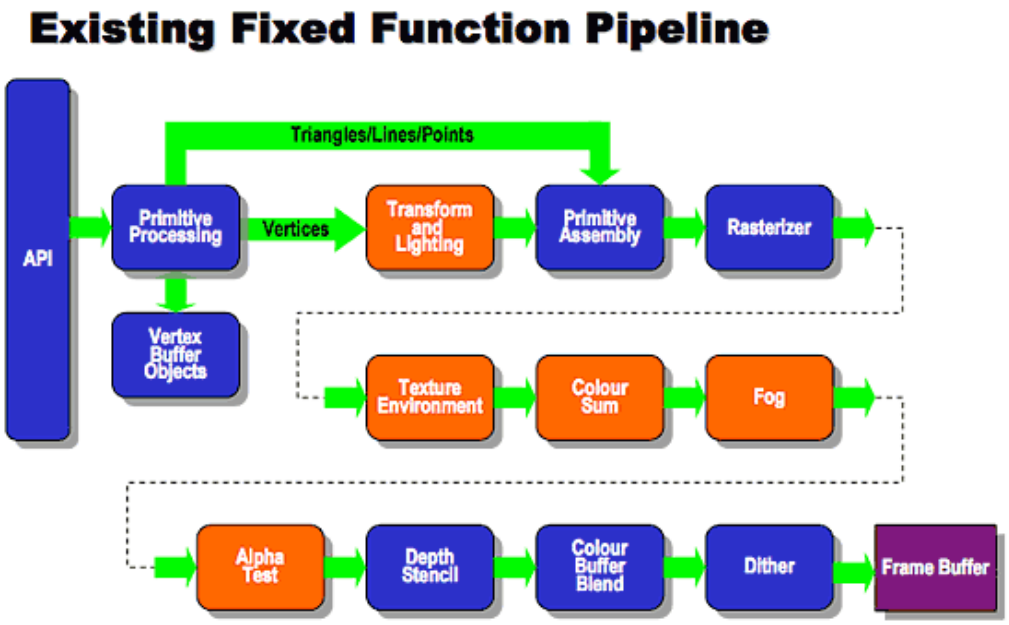
určena pro méně výkonné zařízení s nižší cenou, provádějící výpočty softwarově. Dále tato verze postrádá velkou řadu pokročilejších funkcí.

- **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[5] v OpenGL ES Shading Language. Programátor má plnou kontrolu nad renderovací pipeline[6] 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.



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

Sada vývojových nástrojů (SDK) plně podporuje knihovnu OpenGL ES pro vykreslování grafiky. Verze OpenGL ES 1.1 je podporována od verze platformy Android 1.6. Knihovna OpenGL ES 2.0 je potom dostupná od verze Android 2.2. V mnou implementované aplikaci byla využita verze OpenGL ES 1.0, která svými funkcemi postačuje pro vykreslování rozšířené reality.

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 *Renderer*, který běží ve vlastním vlákně, 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 souřadnice se provádí vedením rovnoběžek se souřadnicovými rovinami. Hodnoty souřadnice jednoho bodu jsou potom určeny protnutím dané rovnoběžky se souřadnou osou.

Grafické objekty je často potřeba transformovat. Transformace [32] jsou zpravidla aplikovány na všechny vrcholy transformovaného objektu. Pro účel transformací se zavádí tzv. homogenní souřadnice, které umožňují reprezentovat veškeré grafické operace jako násobení matic. Homogenní souřadnice mají v 3D prostoru podobu čtveřice, kterou tvoří kartézské souřadnice a váhový element ω. Zavádí se především kvůli operaci translace, kterou nelze běžně reprezentovat násobením matic 3x3 jako u operace rotace či změny měřítka (tzv. scaling).

Definice 4.3.1 *Bod P=[x,y,z] má homogenní souřadnice P_h=[ωX,ωY,ωZ,ω] právě tehdy, když platí:*

X = x/ω Y = y/ω Z = z/ω

kde ω 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:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d_z & 1 \end{pmatrix} \tag{4.1}$$

Změna měřítka:

$$\mathrm{S} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

Otočení:

$$R = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

Zkosení:

$$SH = \begin{pmatrix} 1 & S_{hy} & S_{hz} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

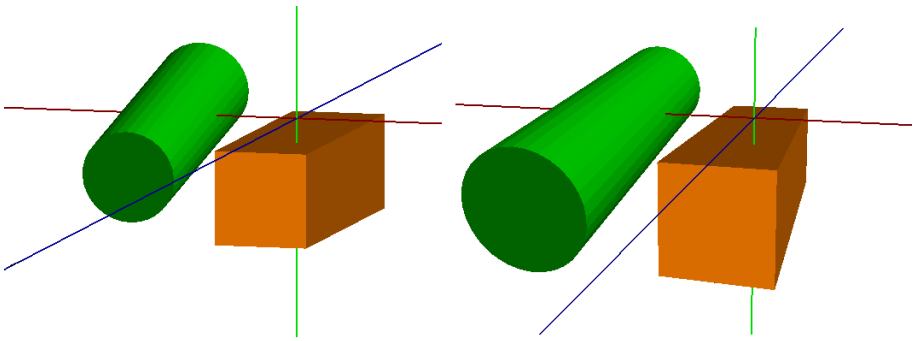
Mezi transformační operace patří také projekce scény. Projekce je zobrazení 3D objektů na 2D médium a lze ji vyjádřit pomocí matice 4x4. OpenGL ES umožňuje zobrazit scénu pomocí paralelní nebo perspektivní projekce (viz obr. 4.2). Scéna je určena specifickým objemem tělesa, které popisuje danou projekci. 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:

$$P_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

Perspektivní projekce:

$$P_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix} \tag{4.6}$$



Obrázek 4.2: Paralelní a perspektivní projekce.

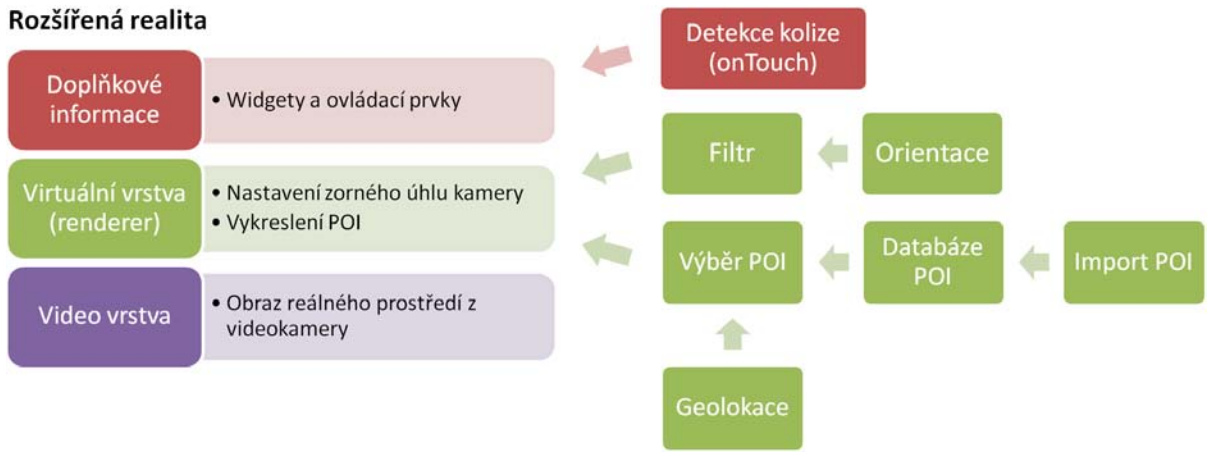
5 Návrh systému

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 blíže popsána v kapitole 2.2.2.

Systém rozšířené reality byl navržen jako samostatný framework[7], 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

Po nastudování metod rozšířené reality se zaměřením na polohové senzory byl navržen framework, který zajišťuje všechny potřebné funkce pro realizaci rozšířené reality. Při návrhu byl kladen důraz na obecnou použitelnost frameworku v nejrůznějších interaktivních programech a rovněž na použití zavedených standardů a formátů. Systém je určen pro vývojáře mobilních aplikací. Usnadňuje vývoj aplikací rozšířené reality s využitím polohových senzorů. Lze použít pro tvorbu nejrůznějších projektů zaměřených převážně na geolokaci objektů. Základní princip činnosti frameworku je zjednodušeně znázorněn na obrázku 5.1.



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

[7] 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 vykreslují 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[8]). 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.

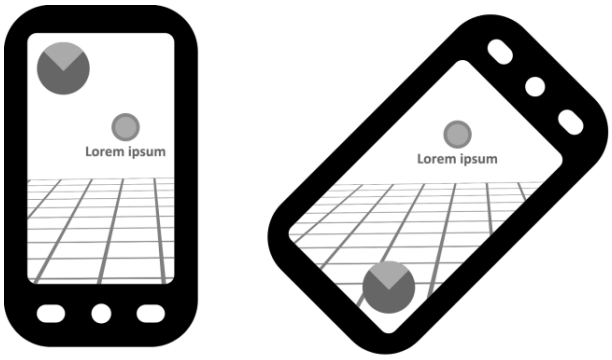
[8] Point of interest

Obrázek 5.3: Zobrazení detailu POI v aplikaci rozšířené reality.

5.1.2 Trasování virtuálních objektů

Trasování virtuálních objektů zajišťují polohové senzory telefonu. Pomocí akcelerometru a senzoru magnetického pole je počítána orientace pozorovatele (mobilního zařízení) v prostoru. Pro zajištění plynulejších přechodů při změně orientace je potřeba použít filtr, který vyhladí zašuměná data z polohových senzorů. Framework vypočítá transformační matici ze získaných dat a přepočítá ji do vhodného souřadného systému. Tato rotační matice je poté použita v 3D rendereru OpenGL pro nastavení zorného úhlu kamery ve virtuální vrstvě. Tímto způsobem se uživatel pomocí senzorů telefonu rozhlíží v 3D virtuálním světě, ve kterém jsou rozmístěny POI, odpovídající umístění v reálném světě. Poloha POI ve scéně je počítána z aktuální GPS lokace uživatele vzhledem k umístění objektů v reálném prostředí. Framework kromě rotační matice počítá i azimut (pro zobrazení kompasu) a svislou orientaci telefonu (pro zobrazení UI prvků a textu na displeji horizontálně zleva doprava vzhledem k uživateli). Všechny tři vypočítané informace o orientaci fungují při všech možných polohách natočení telefonu ve všech třech osách včetně portrait[9] a landscape[10] módu. Scéna virtuálního prostředí je tedy nezávislá na orientaci telefonu a vždy se zobrazuje relativně k prostředí reálnému. Jinak řečeno, rovina země ve virtuální scéně vždy odpovídá povrchu Země skutečného světa, pomineme-li nepatrné výchylky způsobené chybou ve výpočtu orientace či zpoždění filtru. Zobrazení scény při různých rotacích telefonu znázorňuje obrázek 5.4.

[9] Zobrazení na výšku
[10] Zobrazení na šířku



Obrázek 5.4: Přizpůsobení obrazu vůči natočení telefonu.

5.1.3 Souřadný systém

Ve virtuálním prostoru lze snadno určit světové strany. Framework používá kartézský souřadný systém. V počátku soustavy souřadnic stojí pozorovatel. V kladném směru osy Y leží sever, v záporném jih, v kladném směru osy X leží východ a v záporném západ. V kladném směru osy Z roste nadmořská výška. Při výpočtu souřadnic bodů zájmu je zohledněno i zakřivení Zemského povrchu.

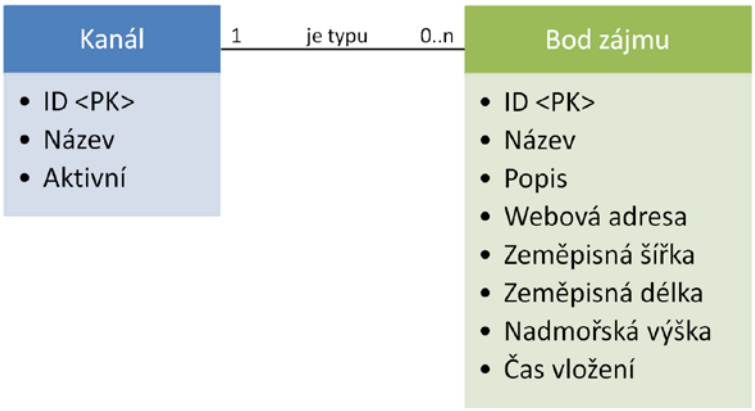
Důležitá je správná kalibrace videokamery, aby její zorný úhel odpovídal nastavení virtuální scény. Vzhledem k tomu, že převážná většina mobilních zařízení nepoužívá transfokátor a ohnisková vzdálenost objektivu je pevně daná, můžeme pro zjednodušení použít fixní zorný úhel pro nastavení perspektivní 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 příbuzné 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 obrázku 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 vykreslují 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 POI, maximální počet kanálů, jednotky vzdálenosti 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.

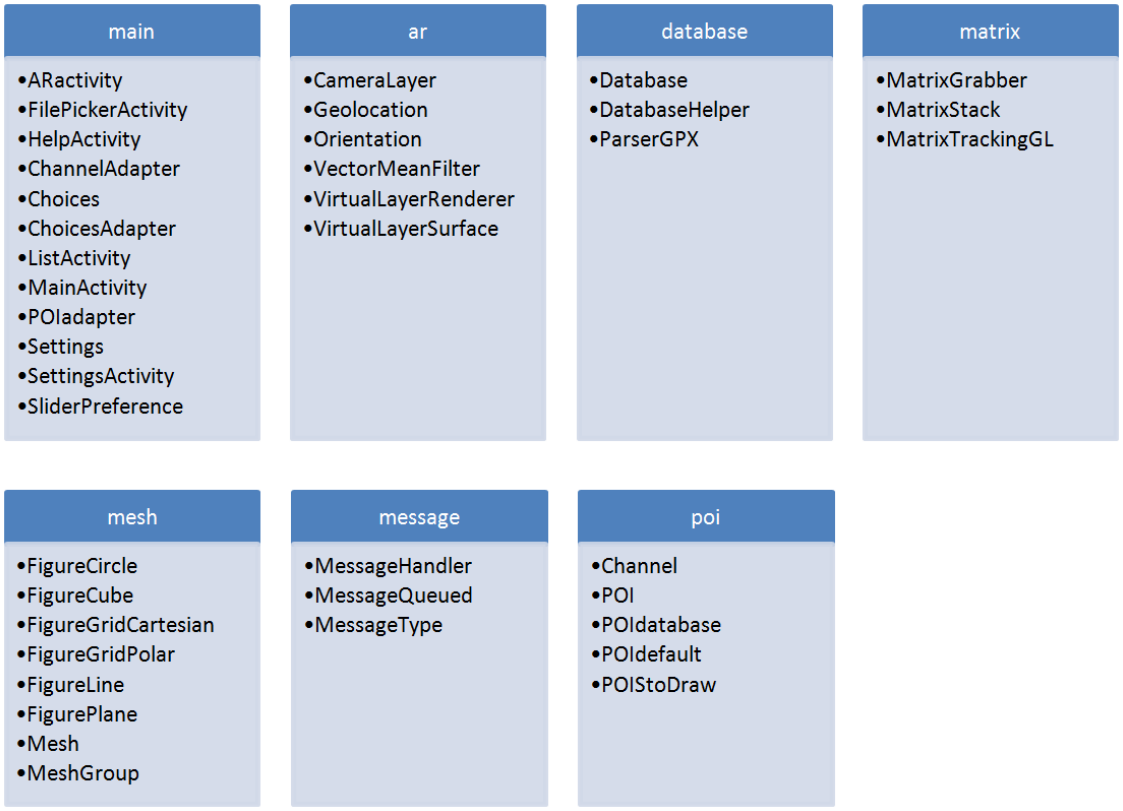
Více o letecké navigaci a jejím testování bude uvedeno v kapitole 7.

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

Celý systém je rozdělen do sedmi balíků, které tvoří jeden funkční celek (viz obr. 6.1). Aplikace byla navržena s ohledem na snadnou modifikovatelnost a rozšiřitelnost. Jednotlivé moduly aplikace jsou od sebe odděleny tak, aby je bylo možné snadno nahrazovat, rozšiřovat či upravovat. Moduly jsou na sobě nezávislé, takže při úpravách většinou není potřeba zasahovat do ostatních částí. Grafické uživatelské rozhraní aplikace je definováno v XML souborech a je zcela odděleno od kódu. Díky tomu jej lze snadno upravovat a měnit dle potřeby.



Obrázek 6.1: Rozdělení tříd do balíků.

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[11] 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 nenachází 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 vykreslují 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

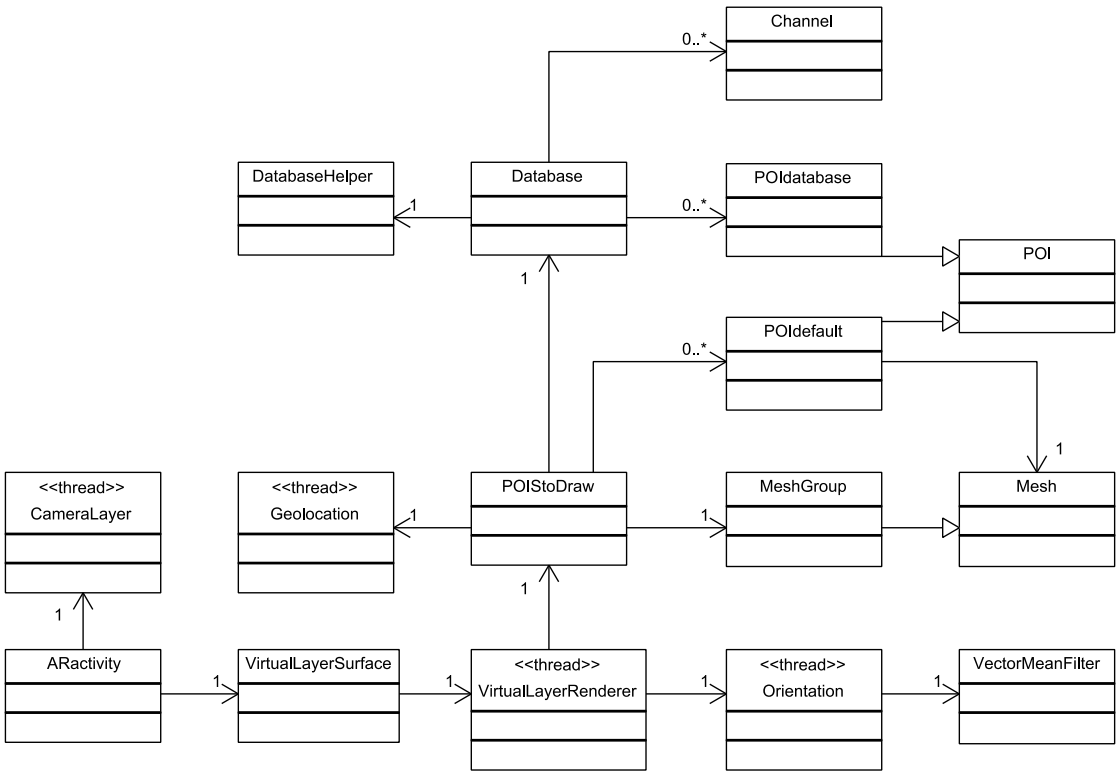
[11] Create, read, update, delete

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[12] 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 ukázáno na screenshotu[13] 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 nejspodnější vrstvě se vykresluje obraz z videokamery. Vykreslování běží ve vlastním vlákně. 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 vlákně a komunikuje s vnějším prostředím pomocí zasílání zpráv. Při vytvoření rendereru je potřeba nastavit transparenci vrstvy, aby nebyl překryt obraz z videokamery. Renderer používá verzi knihovny OpenGL ES 1.0, která poskytuje všechny potřebné funkce. Nejdříve se nastaví parametry rendereru scény 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. Tímto 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:

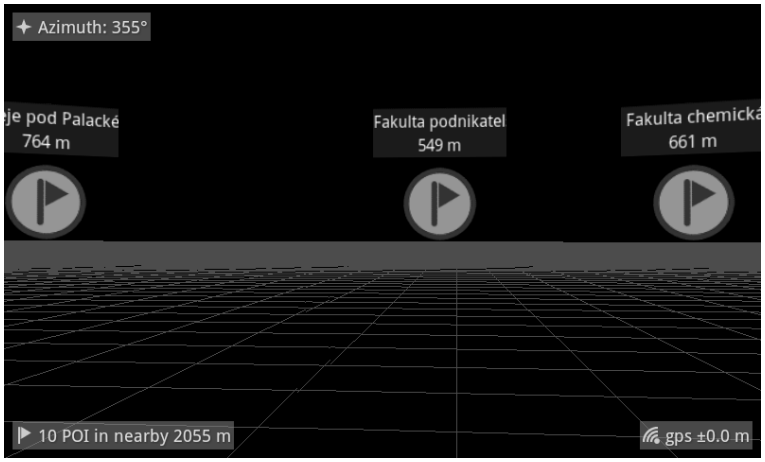
$$fov_w = \text{atan}\left(\frac{sensor\_width \cdot 0{,}5}{focal\_length}\right) \cdot 2$$
$$fov_h = \text{atan}\left(\frac{sensor\_height \cdot 0{,}5}{focal\_length}\right) \cdot 2 \quad (6.1)$$

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.

Vykreslování jednotlivých snímků obrazu probíhá v cyklu. V každém kroku se nejdříve vynulují buffery a transformační matice. Dále se nastaví rotační matice kamery, vypočítaná pomocí senzoru magnetického pole a akcelerometru. Tato matice určuje zorný úhel ve scéně odpovídající směru pohledu v reálném prostředí. Dochází pouze k nastavení rotace kamery. Její pozice je pevná a neměnná. Kamera scény je umístěna do počátku soustavy souřadnic. Následně můžeme vykreslit grafické objekty. Renderer vykresluje mřížku znázorňující rovinu země a dále grafické objekty bodů zájmu. Uživatel si může vybrat dva typy mřížky - kartézskou nebo polární. Zobrazení mřížky lze vypnout v nastavení aplikace. V případě režimu použití nadmořské výšky se mřížka nezobrazuje, jelikož by bylo obtížné získat informaci o nerovnosti terénu v daném místě. Mřížka tedy slouží pouze pro orientaci a znázorňuje rovný povrch země, nikoliv zvlnění terénu. Body zájmu jsou vykreslovány jako jeden celek uložený v objektu třídy *POIStoDraw*. Před samotným vykreslením je vždy potřeba zkontrolovat, zda není potřeba data obnovit. Více o vykreslování POI bude vysvětleno v kapitole 6.5.

Poslední nejsvrchnější vrstvou rozšířené reality jsou doplňkové informace a prvky uživatelského rozhraní. Na displeji se zobrazují informace o azimutu (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 vypíšou informace o názvu POI, názvu kanálu, do kterého přísluší, popisu POI, aktuální vzdálenosti, azimutu, 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 blíže vysvětlena v kapitole 6.7.



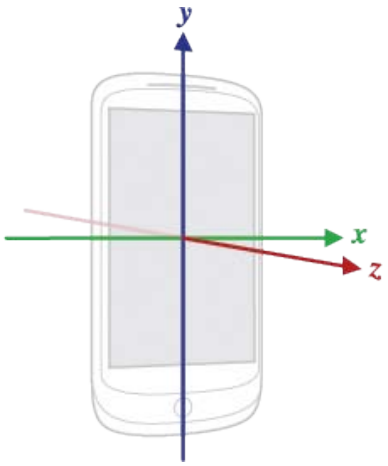
Obrázek 6.3: Pohled rozšířené reality v režimu bez nadmořské výšky s vypnutou videokamerou.

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

6.3 Výpočet orientace

Výpočet orientace je pro aplikaci rozšířené reality zaměřené na polohové senzory stěžejní. Je potřeba zjistit, jakým směrem je orientováno mobilní zařízení v reálném prostředí. Orientace telefonu se počítá v samostatném objektu ve vlastním vlákně. K výpočtu se používají data z akcelerometru a senzoru magnetického pole. Z nich se počítají tři informace: rotační matice určující zorný úhel, azimut a svislá orientace telefonu (náklon). Tyto informace se pravidelně obnovují s nejvyšší možnou frekvencí. Vypočítanou rotační matici používá renderer OpenGL (viz kapitola 6.2). Azimut se používá pro zobrazení kompasu. Svislá orientace zatím není v aplikaci využita, ale mohla by se uplatnit v budoucím rozšíření například pro zobrazení dialogového okna s detailem POI tak, aby okno směřovalo při všech možných způsobech natočení telefonu textem zleva doprava. Orientace okna by tedy byla přizpůsobitelná natočení telefonu.

Senzory na platformě Android používají vlastní souřadný systém [33], který je definován relativně k obrazovce mobilního zařízení ve výchozí 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 neprohazují. 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_s):

$$A_d = -\sum \frac{F_s}{m} \tag{6.2}$$

Naměřené zrychlení vždy ovlivňuje zejména gravitační síla:

$$A_d = -g - \sum \frac{F_s}{m} \tag{6.3}$$

Proto vrací akcelerometr v klidové poloze zrychlení o hodnotě $g = 9.81\ m \cdot s^{-2}$. Z tohoto poznatku plyne, že k výpočtu skutečného zrychlení musí být složka gravitační síly odstraněna. Toho lze dosáhnout pomocí filtru horní propusti. Naopak filtr dolní propusti lze použít k izolaci gravitační síly. 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 (μT).

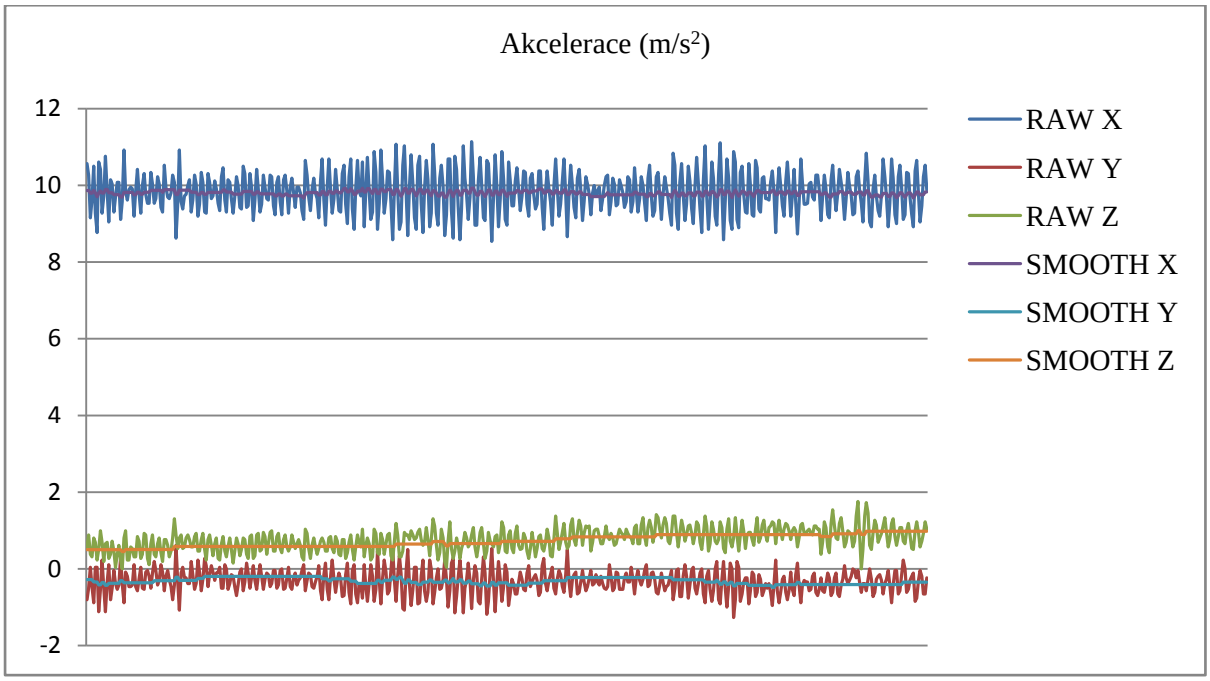
Data mohou být značně zkreslena v případě, kdy se zařízení nachází v silném magnetickém poli, je ve volném pádu nebo značně zrychluje. V těchto situacích většinou vrací senzory chybná data, jelikož nejsou schopná veličiny správně změřit.

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 srovnatelné 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.

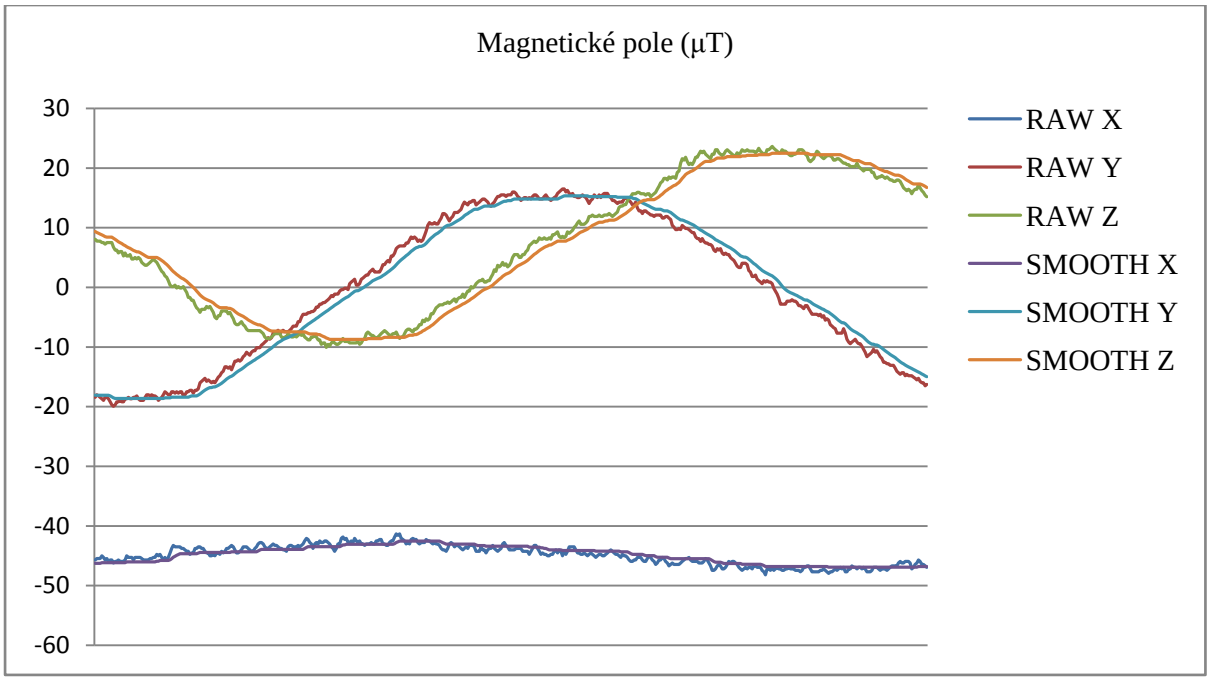
Výpočet nové hodnoty vyjadřuje následující vzorec:

$$outVector[i] = factor \cdot inVector[i] + (1 - factor) \cdot lastVector[i] \tag{6.4}$$

kde *factor* je desetinné číslo z intervalu $(0, 1\rangle$, 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 grafech 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.



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:

$$\begin{pmatrix} M_0 & M_1 & M_2 & 0 \\ M_4 & M_5 & M_6 & 0 \\ M_8 & M_9 & M_{10} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.5}$$

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. Nejdříve je však nutné ji přepočítat 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 vektorem:

$$sZVector = (0,0,1,1) \tag{6.6}$$

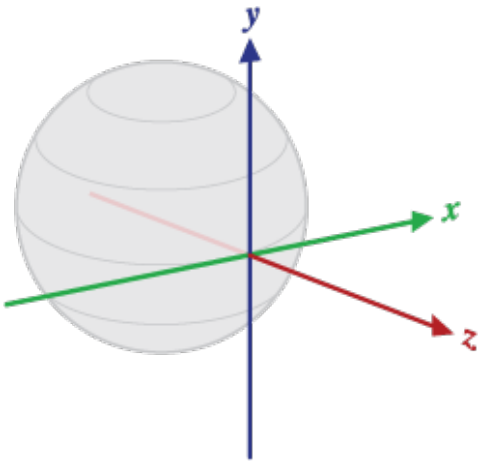
a získáme nový vektor *azimuthVector*. Výsledný azimut (ve stupních) se vypočítá pomocí vztahu:

$$azimuth = 180 + atan2(azimuthVector[0], azimuthVector[1]) \cdot RAD\_TO\_DEG \tag{6.7}$$

kde funkce *atan2(y,x)* vrací 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)* vrací 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 vektorem *sZVector*. Získáme nový vektor *orientationVector*, pomocí něhož vypočítáme výslednou orientaci dle následujícího vztahu:

$$orientation = -atan2(orientationVector[0], orientationVector[1]) \cdot RAD\_TO\_DEG \tag{6.8}$$



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[14] 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[15], který rovněž poskytuje API. Obdobně databázi s WiFi sítěmi poskytuje služba Wireless Geographic Logging Engine[16].

Instance třídy *Geolocation* vrací geografické informace o aktuální pozici telefonu. Některé informace v případě použití zdroje dat ze sítě nejsou k dispozici (viz tabulka 6.1). Informace se pravidelně obnovují vždy, když senzor zaznamená změnu pozice. Data ze senzorů zpracovává instance třídy *LocationListener*, která odchytává události. V případě odchycení události získáme přístup k novým datům ze senzoru daného typu.

	Geodata z GPS	Geodata ze sítě
Zeměpisná šířka (stupně)	●	●
Zeměpisná délka (stupně)	●	●
Časová známka (UTC v milisekundách)	●	●
Přesnost zaměření (metry)	●	●
Nadmořská výška (metry)	●	○
Azimut (stupně)	●	○
Rychlost (metry za sekundu)	●	○

Tabulka 6.1: Seznam dostupných geografických informací ze senzorů.

[14] Základnová převodní stanice sítě GSM
[15] Viz http://www.opencellid.org/
[16] Viz http://www.wigle.net/

Získaná data se používají při načítání bodů zájmu z databáze, aby se vybraly ty, které se nachází nejblíže 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 vykreslovaného objektu je potřeba vypočítat rozdíl pozice tohoto objektu a aktuální lokace zařízení pro osy *x* a *y*:

$$x = (poiLon - deviceLon) \cdot DEG\_TO\_METERS \cdot \cos(poiLat \cdot DEG\_TO\_RAD)$$
$$y = (poiLat - deviceLat) \cdot DEG\_TO\_METERS \tag{6.9}$$

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

$$RADIUS = 6378137$$
$$CIRCUMFERENCE = 2 \cdot \pi \cdot RADIUS$$
$$DEG\_TO\_METERS = \frac{CIRCUMFERENCE}{360} \tag{6.10}$$

6.5 Vykreslení bodů zájmu

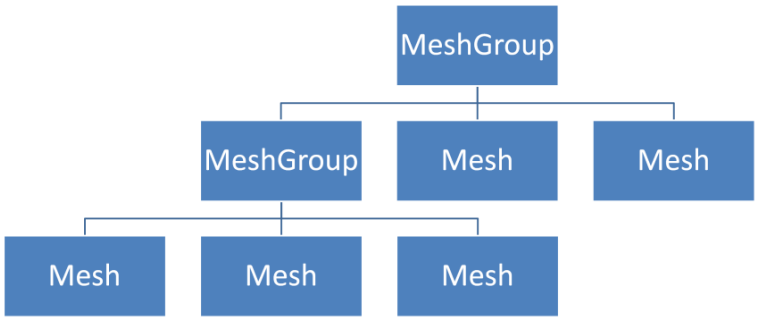
Vykreslování bodů zájmu zajišťuje renderer knihovny OpenGL, který byl blíže popsán v kapitole 6.2. Body zájmu určené k vykreslení jsou uloženy v instanci třídy *POIStoDraw*. V každém vykreslovacím cyklu se volá metoda *refresh* třídy *POIStoDraw*, která zajišťuje načtení POI z databáze a jejich aktualizaci. Při každém volání *refresh* se ověřuje, zda je skutečně potřeba tyto úkony provádět. Aktualizace probíhá ve vlastním synchronizovaném vlákně, aby bylo vykreslování v rendereru plynulé. Metoda je synchronizována tak, aby v jednom okamžiku nemohla pracovat dvě vlákna v kritické sekci se stejnými daty.

Načtení resp. aktualizace bodů zájmu z databáze proběhne vždy na začátku při spuštění rendereru 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 načtou nejbližší POI od aktuální pozice mobilního zařízení a uloží se do kolekce spolu s vytvořenými grafickými objekty, které se budou renderovat. Aktualizaci 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.



Obrázek 6.9: Stromová struktura grafických objektů.

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 blíže 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 mít 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.

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 nahraném 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[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 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.

Vytvoření databáze a přístup k ní umožňuje třída *DatabaseHelper*. Ta zpřístupňuje databázový objekt, jemuž lze pokládat standardní SQL dotazy. Databáze používá cizí klíče, které je nutné explicitně povolit. Obsahuje dvě tabulky. První ukládá informace o kanálech. Ty sdružují body zájmu do skupin. Jméno kanálu musí být unikátní. Pokud je kanál aktivní (booleanovská hodnota), budou se příslušné POI zobrazovat v rozšířené realitě. Druhá tabulka obsahuje informace o bodech zájmu. Název POI musí být unikátní. Každé POI patří do příslušného kanálu. Název, typ (kanál) a geografické souřadnice jsou povinné položky.

[17] Viz 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 od 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 vykreslují 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 vlákně. 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.

Detekce kliknutí (tzv. picking) na POI lze realizovat několika způsoby. Nejpoužívanější metody jsou color picking a ray picking. Color picking funguje na principu přidělování jednoznačné unikátní barvy každému objektu. Tyto barvy jsou uloženy ve speciálním bufferu, který se nevykresluje a slouží pouze k účelu detekce kolize. Pokud nastane událost kliknutí, zjistí se pomocí OpenGL funkce *glReadPixels* barva pixelu na daných souřadnicích. Podle specifické barvy se následně rozpozná objekt, na který uživatel kliknul. Druhou metodou detekce kolize je ray picking, který používá moje aplikace rozšířené reality.

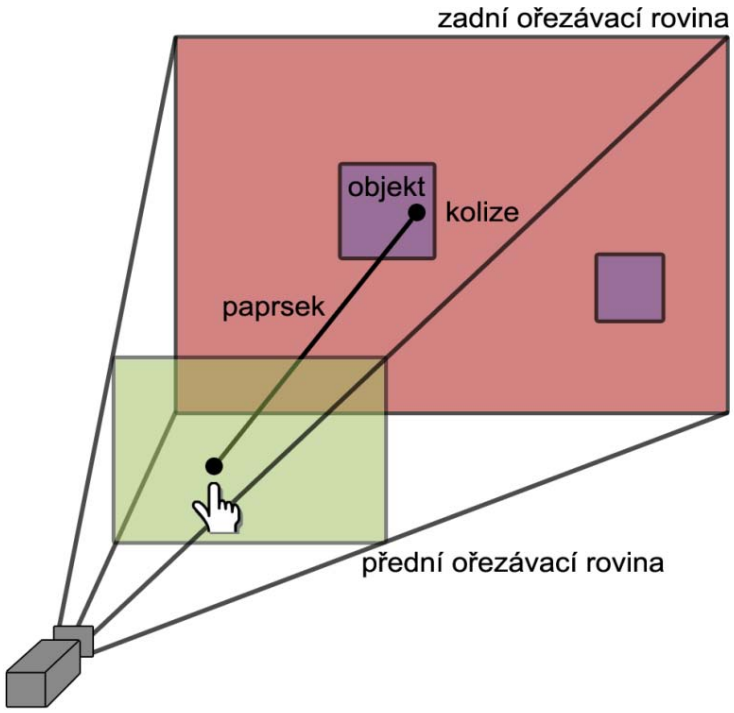
Metoda ray pickingu [36] funguje na principu trasování paprsku. Do 3D scény se vyšle paprsek a kontroluje se jeho průnik s danými objekty. Nejdříve je potřeba vypočítat směr šíření paprsku ve scéně. Startovní pozice paprsku je v počátku soustavy souřadnic a jeho směr je určen vektorem. Směr se vypočítá 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:

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

(6.11)

kde *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 vypočítanou 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.

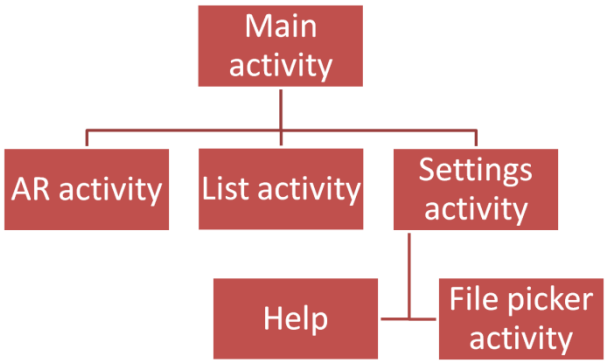


Obrázek 6.10: Ray picking pro detekci kliknutí na objekt.

6.8 Uživatelské rozhraní

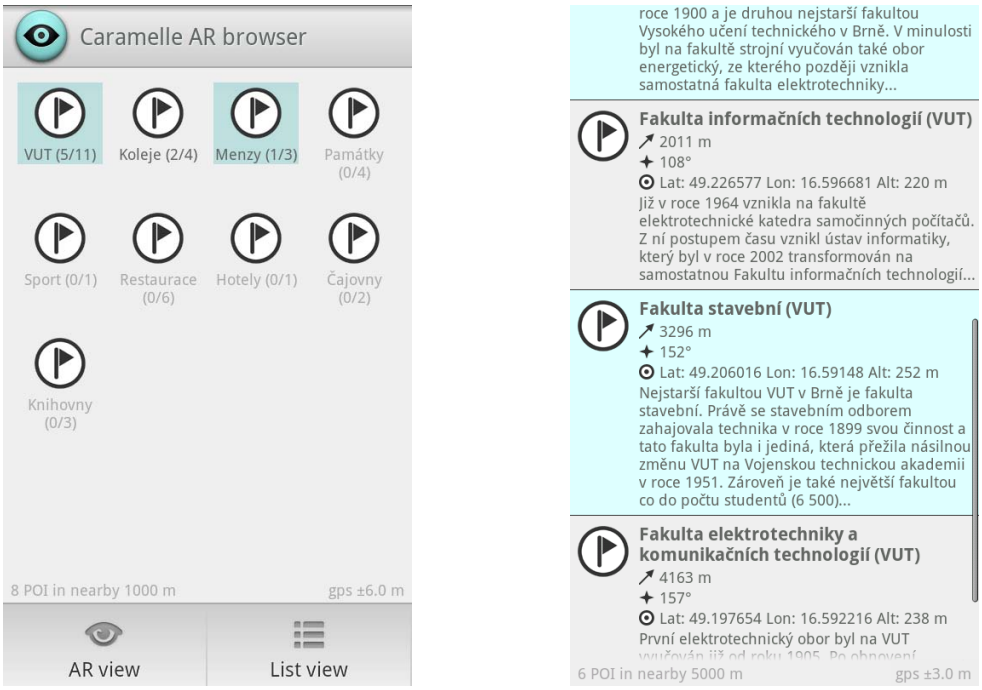
Uživatelské rozhraní aplikace bylo navrženo s ohledem na snadné a intuitivní ovládání. Po spuštění aplikace se inicializuje aktivita *MainActivity*, která zobrazuje hlavní menu aplikace (viz obr. 6.11a). Zde se nachází seznam všech kanálů s ikonou, názvem a počtem dostupných POI z aktuální lokace. Geolokace se načítá ve vlastním vlákně, jelikož zaměření pozice trvá určitý čas. Díky tomu je zobrazení hlavního menu plynulé. Kanály se v seznamu řadí podle počtu dostupných POI. Uživatel si vybere kanály, které ho zajímají a zvolí jeden z možných způsobů zobrazení POI. Výběr kanálů je možný ručně využitím dotykového displeje nebo pomocí joysticku. V kontextovém menu lze pomocí tlačítka vybrat všechny POI, případně je zrušit. Dále je možné pomocí tlačítka explicitně obnovit kanály z aktuální lokace.

V hlavní aktivitě se spouští další aktivity pro vizualizaci bodů zájmu nebo nastavení aplikace. Každá aktivita musí být deklarována v manifestu Android projektu. Vztahy mezi nimi jsou znázorněny na obrázku 6.14.



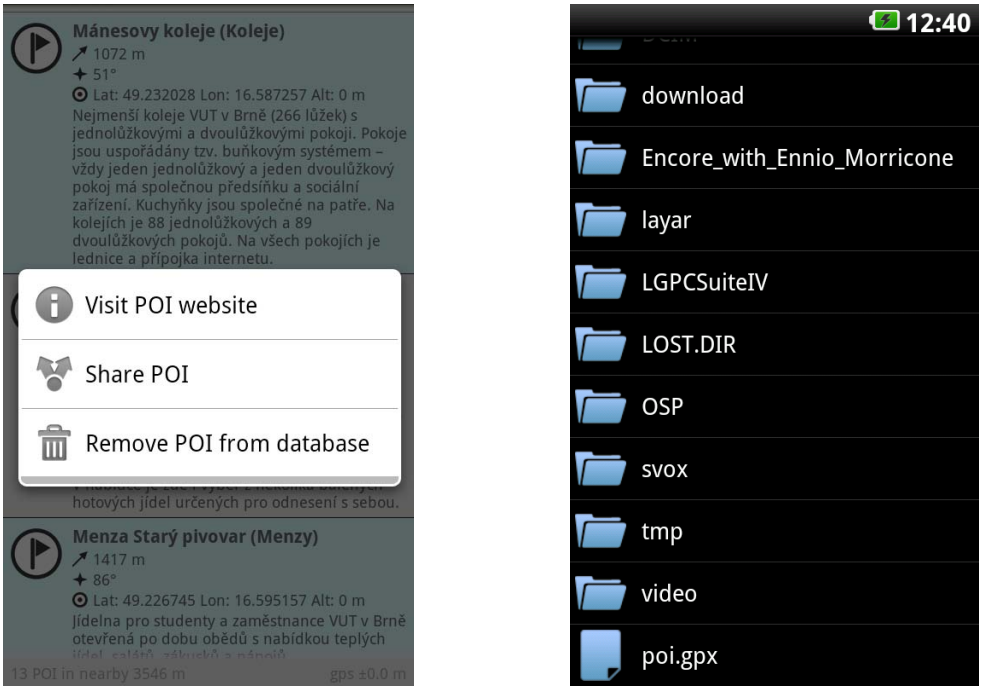
Obrázek 6.14: Diagram aktivit.

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.

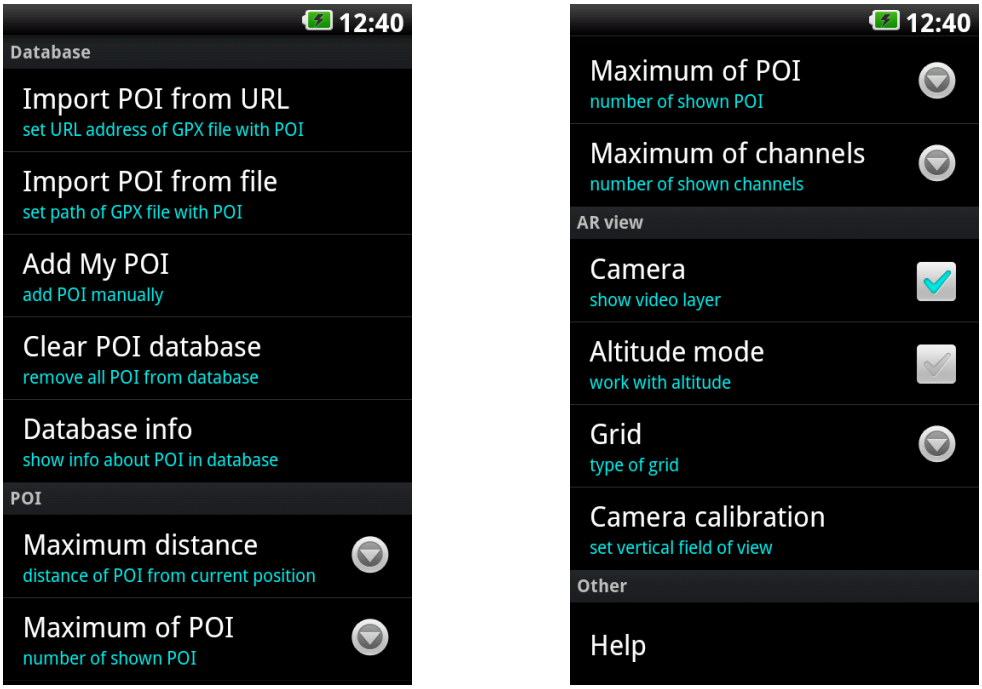
b) Vpravo zobrazení bodů zájmu v seznamu.



Obrázek 6.12: a) Vlevo kontextové menu seznamu bodů zájmu.

b) Vpravo výběr GPX souboru na SD kartě.

V kontextovém menu hlavní aktivity je dále k dispozici tlačítko pro zobrazení uživatelského nastavení aplikace. To spouští aktivitu *SettingsActivity* (viz obr. 13). V nastavení je možné importovat POI z URL adresy nebo ze souboru, přidat vlastní POI, smazat databázi a zobrazit informace o všech kanálech a bodech zájmu. Dále je zde možné nastavit maximální vzdálenost zobrazovaných POI, maximální počet kanálů a bodů zájmu. Uživatel rovněž může skrýt video vrstvu v pohledu rozšířené reality, vypnout režim nadmořské výšky, nastavit typ mřížky a v případě potřeby kalibrovat kameru. K dispozici je také nápověda k programu. Všechna uživatelská nastavení se ukládají ve sdíleném úložišti, které poskytuje platforma Android pro tento účel. Přístup k těmto informacím umožňuje třída *Settings*.



Obrázek 6.13: Uživatelské nastavení aplikace.

Grafické uživatelské rozhraní aplikace je definováno v XML souborech. Díky tomu lze snadno upravovat a přizpůsobovat. V XML jsou především definovány layouty, menu, grafické prvky, styly a různé hodnoty. Layouty specifikují rozvržení grafických prvků na displeji telefonu a určují tak vzhled dané aktivity. V souborech popisujících menu jsou uloženy dané položky včetně popisků a ikon. Soubory s grafickými prvky popisují např. barevné přechody (gradienty) nebo selektory. Selektory určují, jak bude grafický prvek vypadat při kliknutí, označení kurzorem apod. Případně mohou definovat vzhled lichých a sudých prvků např. v seznamu. Soubory s hodnotami ukládají seznam všech použitých barev, seznam všech řetězců použitých v programu a seznam výchozích hodnot uživatelského nastavení. Díky tomuto způsobu ukládání dat je snadno možné měnit design

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 sady 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 testovaní je snadno možné odhalit výkonnostní nedostatky aplikace.

7.1 Ladící nástroje

Během vývoje jsem použil několik užitečných utilit. Nezbytným nástrojem pro ladění Android aplikací je DDMS (Dalvik Debug Monitor Server). Ten umožňuje sledovat všechny události (tzv. logy) spuštěného emulátoru nebo připojeného telefonu v režimu ladění. Události mohou být informačního charakteru, určené pro ladění nebo se může jednat o výjimky a chyby v aplikaci. Pro vyšší přehlednost lze tyto logy filtrovat podle popisků (tzv. tagů) a sledovat tak např. jen události specifického modulu aplikace. DDMS dále umožňuje sledovat vytvořené procesy, vlákna a alokovanou paměť. Umí simulovat některé události telefonu jako např. příchozí hovor, SMS nebo GPS, procházet adresářovou strukturu úložiště v telefonu, vytvářet screenshoty a mnohé další funkce.

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.

Vývojové prostředí na platformě Android obsahuje speciální framework pro testování aplikací. Ten má několik nástrojů včetně podpory známého testovacího frameworku JUnit, který používá právě metody Unit testování. Na platformě Android se obvykle testují aktivity, služby a poskytovatelé obsahu. Ověřuje se jejich korektní zobrazení a chování v různých situacích, které mohou při provozu aplikace nastat. Pro systém Android dále existuje několik externích frameworků určených k Unit testování. Jedná se např. o Robolectric[18] nebo Robotium[19].

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.

Při vývoji jsem dále také provedl několik měření stability virtuálního obrazu (viz kapitola 6.3). Senzory vracely data s poměrně velkým šumem. Šum bylo potřeba redukovat. Z tohoto důvodu jsem implementoval filtr a následně změřil jeho vliv na hladký pohyb kamery a stabilizaci obrazu. Výsledky měření jsou znázorněny v grafech na obrázcích 6.6 a 6.7.

[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. příchozí telefonní hovor, SMS zpráva, spuštěný budík či jakákoliv 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, přenese 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 běží dál a tato událost na její běh nemá žádný vliv. Pouze se zobrazí běžná notifikace ve stavové liště.

7.4 Testování v terénu

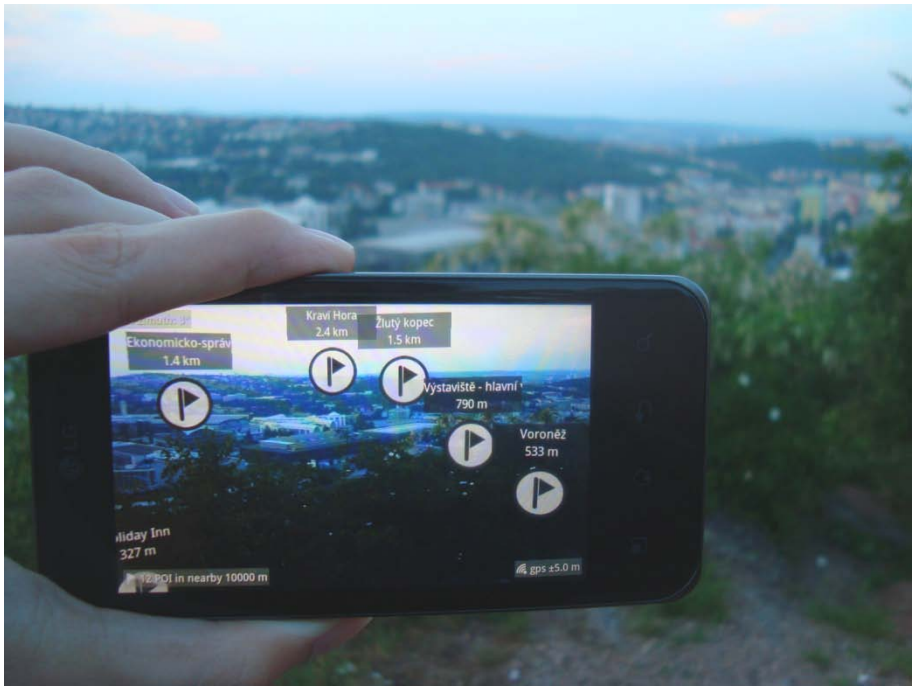
Cílem práce bylo vytvořit aplikaci rozšířené reality zaměřenou na leteckou navigaci. Nebyla možnost vytvořený systém testovat v letadle. Aplikaci jsem však vyzkoušel v terénu a to jak ve městě, tak na vyvýšeném místě na kopci, abych mohl alespoň částečně otestovat správnou funkčnost zobrazování nadmořské 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 OpenStreetMaps[20] a Google Maps[21]. Letecká data jsem získal z veřejně dostupné databáze letišť[22]. Tyto body zájmu jsou spolu s leteckými daty pro celou Českou Republiku uloženy v GPX souborech na přiloženém datovém médiu této diplomové práce.

Během testování aplikace rozšířené reality v terénu se podařilo ověřit, že body zájmu zobrazené na displeji telefonu skutečně odpovídají umístění reálných objektů. Zaměření nebylo v některých případech zcela přesné, ale pro účel navigace zcela dostačující. Výchylky v přesnosti pozice byly pravděpodobně způsobeny chybou ve výpočtu orientace, jež byla zapříčiněna nepřesným měřením. Senzory telefonu mohou být rušeny vnějšími vlivy jako např. magnetické pole, kovové předměty apod. Chyby ve výpočtu mohou být rovněž způsobeny špatnou kalibrací senzorů. Dalším potencionálním problémem může být nepřesný výpočet nadmořské výšky pomocí GPS. V terénu jsem dále zkoušel aktualizaci databáze bodů zájmu a obnovu jejich pozice v případě, že dojde ke změně geografické polohy uživatele. Ověřil jsem, že aktualizace probíhá správně podle nastavených kritérií, takže se pokaždé zobrazují nejbližší body zájmu v okolí.

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 kriteriem v terénu byla výdrž baterie. Spotřeba energie je při použití 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 nabité 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

Během vývoje jsem se seznámil s několika existujícími programy na platformě Android, které se zaměřují na rozšířenou realitu. Jednou z prvních aplikací rozšířené reality, která používá polohové senzory, byl program Layar. Ten nabízí několik stovek kanálů nazývaných vrstvy s velkým množstvím bodů zájmu získaných většinou z nejrůznějších online služeb (např. Wikipedia, Panoramio, Foursquare, Last.fm, Twitter, Flickr atd.). Uživatelé mají dokonce možnost vytvářet si vlastní vrstvy. Další poměrně rozšířenou aplikací je Wikitude. Její funkce jsou velmi podobné programu Layar. Wikitude umožňuje navíc zobrazovat více vrstev najednou, ovšem počet dostupných vrstev je mnohem menší. Oblíbenou aplikací je také Junaio. Ta v sobě kombinuje více typů zpracování rozšířené reality včetně metody počítačového vidění. Všechny zmíněné aplikace jsou dostupné na platformách Android a iPhone. Přínos aplikací je zejména v poznávání neznámých míst. Velkým omezením je nutnost připojení k internetu, což může být např. v zahraničí problém.

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 programy používající knihovnu ARToolKit a její mutace. Tyto aplikace jsem blíže nezkoumal, jelikož fungují na jiném principu, než řeší tato diplomová práce.

7.6 Publikace na Android Marketu

Každá nová aplikace, která dodržuje stanovené podmínky, může být publikována na Android Marketu, odkud je přístupná všem uživatelům. K tomu je potřeba se zaregistrovat jako vývojář a zaplatit určitý registrační poplatek. Potom se uživateli zpřístupní webové rozhraní, odkud je možné nahrávat, aktualizovat či spravovat vlastní aplikace. Ty lze nabízet zdarma či za poplatek. V České Republice však v současné době není možné aplikace prodávat. Každá zveřejněná aplikace musí být z hlediska bezpečnosti digitálně podepsaná. Při nahrávání uživatel stručně popíše, k jakému účelu daná aplikace slouží, přidá screenshoty a vybere státy, ve kterých bude přístupná. V uživatelském profilu lze sledovat zpětnou vazbu o užívání aplikace. K dispozici jsou jednoduché statistiky včetně počtu celkových a aktivních instalací.

Aplikace rozšířené reality implementovaná v rámci této diplomové práce je zdarma přístupná na Android Marketu[23] pod názvem Caramelle Augmented Reality.

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

8 Závěr

Cílem práce bylo nastudovat základní principy rozšířené reality na mobilních zařízeních a vytvořit systém, jenž bude za pomocí polohových senzorů zobrazovat geografické body zájmu. Mobilní telefon v této situaci funguje jako průhledový displej. Zachycuje pomocí videokamery reálný obraz světa, přes který se vykresluje rozšířená realita zobrazující na displeji body zájmu na odpovídajících místech. Navržený systém je určen pro mobilní platformu Google Android. V rámci této práce byl vyvinut framework rozšířené reality, který může nalézt uplatnění v nejrůznějších aplikacích. Na tomto frameworku byla následně postavena aplikace sloužící jako navigace či prostředek pro snadnější orientaci v terénu. Cílem práce bylo, aby byla aplikace schopna vizualizovat letecká data. Výsledná aplikace může sloužit nejen jako letecká navigace, ale například i jako turistický průvodce. Pro získání bodů zájmu jsem použil několik zdrojů a aplikaci jsem testoval v terénu, abych ověřil, zda je schopna uživatele správně navigovat. Cíl této práce byl tedy dosažen a zadání splněno.

Práce na projektu byla 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 a multimédií, 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 multimédií na FIT v Brně a přislíbena je i účast na konferenci Openmobility 2011 na Fakultě informatiky Masarykovy univerzity v Brně.

Vývoj projektu bude pokračovat i nadále. Především bych se chtěl zaměřit na vylepšování vytvořeného frameworku a použít jej v dalších aplikacích. Z hlediska celkové robustnosti projektu by 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

[1] Android (operating system). In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4 August 2007, last modified on 20 December 2010 [cit. 2010-12-20]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Android_%28operating_system%29>.

[2] Open Handset Alliance. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5 November 2007, last modified on 20 December 2010 [cit. 2010-12-20]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Open_Handset_Alliance>.

[3] KATYSOVAS, Tomas. *A first look at Google Android : Internet Technologies 2*. Bolzano, 2008. 28 s. Seminární práce. Free University of Bolzano. Dostupné z WWW: <http://www.kandroid.org/s2/doc/android.pdf>.

[4] *Android developers* [online]. 2008, 21 December 2010 [cit. 2010-12-21]. What is Android?. Dostupné z WWW: <http://developer.android.com/guide/basics/what-is-android.html>.

[5] Dalvik (software). In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 12 November 2007, last modified on 21 December 2010 [cit. 2010-12-21]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Dalvik_%28software%29>.

[6] *Android developers* [online]. 2008, 22 December 2010 [cit. 2010-12-22]. Activity. Dostupné z WWW: <http://developer.android.com/reference/android/app/Activity.html>.

[7] *Android developers* [online]. 2008, 22 December 2010 [cit. 2010-12-22]. Application Fundamentals. Dostupné z WWW: <http://developer.android.com/guide/topics/fundamentals.html>.

[8] *Android developers* [online]. 2008, 22 December 2010 [cit. 2010-12-22]. User Interface. Dostupné z WWW: <http://developer.android.com/guide/topics/ui/index.html>.

[9] ZANDL, Patrick. Příští fenomén: rozšířená realita je budoucnost webu. *Lupa* [online]. 22. 5. 2009, [cit. 2011-12-27]. Dostupný z WWW: <http://www.lupa.cz/clanky/rozsirena-realita-augmented-reality/>.

[10] MATĚNA, Lukáš. *Parametry systému pro rozšířenou virtuální realitu*. Brno, 2007. Diplomová práce. Masarykova univerzita, Fakulta informatiky.

[11] MILGRAM, Paul; TAKEMURA, Haruo; UTSUMI, Akira. Augmented Reality : A class of displays on the reality-virtuality continuum. *ATR Communication Systems Research Laboratories*. 1994. Dostupný také z WWW: <http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf>.

[12] Virtuality Continuum. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26 November 2006, last modified on 27 December 2010 [cit. 2011-12-27]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Virtuality_Continuum>.

[13] Augmented reality. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 15 September 2002, last modified on 27 December 2010 [cit. 2011-12-27]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Augmented_reality>.

[14] *ARToolKit* [online]. 1999 [cit. 2011-01-03]. ARToolKit Documentation. Dostupné z WWW: <http://www.hitl.washington.edu/artoolkit/documentation/>.

[15] *Area Cellphone* [online]. 20 May 2010 [cit. 2011-01-03]. Augmented Reality Brings Route Alive. Dostupné z WWW: <http://areacellphone.com/2010/05/best-android-navigation-app-this-month-augmented-reality-brings-route-alive/>.

[16] *Fifth Dimension Technologies* [online]. 1999-2005, last updated: 07/19/2007 [cit. 2011-01-05]. 5DT Head Mounted Display (HMD). Dostupné z WWW: <http://www.5dt.com/products/ihmd03.html>.

[17] KIRCHNER, Martin. *Robotik Medizin* [online]. Universität Karlsruhe : 01. Juli 2003 [cit. 2011-01-05]. Technische Realisierungen und medizinische Anwendungen von Head-Mounted-Displays. Dostupné z WWW: <http://www.robotikmedizin.de/robot_4.htm>. ISBN 3446227016.

[18] PIHAN, Roman. *Fotografování* [online]. 19.01.2006 [cit. 2011-01-08]. Objektivy, jak vybrat a používat - 2. Potíže objektivů. Dostupné z WWW: <http://www.fotografovani.cz/art/tech_vybirame_jak/rom_lenses2.html>.

[19] NOHEJL, Petr. *Navigace zaměřená na Geocaching*. Brno, 2009. Bakalářská práce. Vysoké učení technické, Fakulta informačních technologií.

[20] MARTÍNEK, Jan. *ABC Linuxu* [online]. 6. 9. 2006 [cit. 2011-01-08]. GPS a komunikační protokol NMEA. Dostupné z WWW: <http://www.abclinuxu.cz/clanky/ruzne/gps-a-komunikacni-protokol-nmea-1-princip-historie>.

[21] KURUC, Jiří. *Navigovat.cz* [online]. 16. 9. 2009 [cit. 2011-01-08]. Jak funguje elektronický kompas. Dostupné z WWW: <http://navigovat.mobilmania.cz/clanky/jak-funguje-elektronicky-kompas/sc-3-a-1314330>.

[22] *Magnetic Sensors : Honeywell Electronic Compassing Solutions* [online]. 2003 [cit. 2011-01-09]. Electronic Compasses for Personal Handheld Use, Auto Compasses, Aircraft Compasses, and Marine Compasses . Dostupné z WWW: <http://www.magneticsensors.com/landnav.html>.

[23] HUSÁK, Miroslav. *Akcelerometry*. Praha, 2005. České vysoké učení technické, Fakulta elektrotechnická. Dostupné z WWW: <http://www.micro.feld.cvut.cz/home/X34SES/prednasky/08%20Akcelerometry.pdf>.

[24] VOJÁČEK, Antonín. *HW.cz* [online]. 30. Duben 2007 [cit. 2011-01-09]. Jak pracují nové 3D MEMS akcelerometry Freescale. Dostupné z WWW: <http://hw.cz/Produkty/Nove-soucastky/ART1875-Jak-pracuji-nove-3D-MEMS-akcelerometry-Freescale-.html>.

[25] Accelerometer. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 22 September 2003 , last modified on 7 January 2011 [cit. 2011-01-09]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Accelerometer>.

[26] AZUMA, Ronald. A Survey of Augmented Reality. *Hughes Research Laboratories*. August 1997. Dostupný také z WWW: <http://www.cs.unc.edu/~azuma/ARpresence.pdf>.

[27] *Metaio : Augmented Solutions* [online]. 2010 [cit. 2011-01-09]. LEGO Group Brings its Augmented Reality Boxes to All Its Stores Worldwide. Dostupné z WWW: <http://www.metaio.com/media-press/press-release/press-release-lego-group-brings-its-augmented-reality-boxes-to-all-its-stores-worldwide/>.

[28] OpenGL. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16.1.2005, last modified on 8.11.2010 [cit. 2011-04-29]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/OpenGL>.

[29] OpenGL ES. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 12 December 2004, last modified on 31 March 2011 [cit. 2011-04-29]. Dostupné z WWW: <http://en.wikipedia.org/wiki/OpenGL_ES>.

[30] *Khronos Group* [online]. 2011 [cit. 2011-04-29]. OpenGL ES 2.X and the OpenGL ES Shading Language. Dostupné z WWW: <http://www.khronos.org/opengles/2_X/>.

[31] Coordinate system. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 7 September 2002, last modified on 27 April 2011 [cit. 2011-04-30]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Coordinate_system>.

[32] Grafické transformace. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1. 1. 2008, last modified on 22. 9. 2010 [cit. 2011-04-30]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Grafick%C3%A9_transformace>.

[33] *Android developers* [online]. 2008, 2 May 2011 [cit. 2011-05-03]. SensorEvent. Dostupné z WWW: <http://developer.android.com/reference/android/hardware/SensorEvent.html>.

[34] *Android developers* [online]. 2008, 2 May 2011 [cit. 2011-05-04]. SensorManager. Dostupné z WWW: <http://developer.android.com/reference/android/hardware/SensorManager.html>.

[35] FOSTER, Dan. *The GPS Exchange Format* [online]. 2002 [cit. 2011-05-19]. Dostupné z WWW: <http://www.topografix.com/gpx.asp>.

[36] KATIC, Steve. *The Code Project* [online]. 22. 4. 2009 [cit. 2011-05-19]. Moving 3D Objects with the Mouse using OpenGL. Dostupné z WWW: <http://www.codeproject.com/KB/openGL/TranslationController.aspx>.

[37] *Android developers* [online]. 16 May 2011 [cit. 2011-05-20]. Testing Fundamentals. Dostupné z WWW: <http://developer.android.com/guide/topics/testing/testing_android.html>.

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