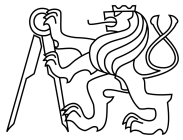
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Knihovna polohově zaměřené rozšířené reality pro systém Android
Student:	Bc. Daniel Průdek
Vedoucí:	Ing. David Sedláček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Navrhnětě a implementujte knihovnu pro systém Android, která bude umožňovat zobrazování bodů zájmu skrze pohled kamery - tzv. geo-rozšířenou realitu. Prozkoumejte a analyzujte existující řešení. Při samotné implementaci se soustřeďte na výkon (někdy je nutné zobrazit mnoho bodů, levnější zařízení disponují horším HW vybavením, apod.).
Navrhněte příjemné a intuitivní uživatelského rozhraní. Využijte všech dostupných senzorů zařízení pro dosažení co nejlepšího dojmu z vizualizace (zlepšení user experience). Práce má implementační charakter a jejím hlavním výstupem bude rozšiřitelná knihovna poskytující funkce geo rozšířené reality a funkční aplikace pro systém Android, která bude využívat tuto knihovnu a prezentovat její funkčnost. Součástí práce bude podrobná dokumentace popisující rozhraní knihovny a její použití.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 11. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Knihovna polohově zaměřené rozšířené reality pro systém Android

Bc. Daniel Průdek

Vedoucí práce: Ing. David Sedláček, Ph.D.

10. května 2016

Poděkování

Rád bych poděkoval vedoucímu práce, kterým byl Ing. David Sedláček, Ph.D. za jeho přístup a pomoc při realizaci této práce. Velký dík patří mé přítelkyni Šárce, která mě po celou dobu psaní podporovala a dodávala mi sebedůvěru. V neposlední řadě bych rád poděkoval všem přátelům a rodině, které jsem poslední dobou možná trochu zanedbával.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2016 .....

Odkaz na tuto práci

Průdek, Daniel. *Knihovna polohově zaměřené rozšířené reality pro systém Android*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá implementací Android knihovny polohově zaměřené rozšířené reality. Součástí práce je průzkum existujících řešení, návrh zaměřený na výkon a plynulost, implementace a uživatelské testování. Výsledkem je fungující aplikace zobrazující body zájmu skrze náhled fotoaparátu na zařízení se systémem Android.

Klíčová slova polohově zaměřená rozšířená realita, Android knihovna, senzory, body zájmu, rozšířená realita, geo-rozšířená realita, náhled fotoaparátu

Abstract

The aim of this thesis is to implement location-based augmented reality library for Android. There are four parts in this thesis — analysis of existing solutions, design focused on performance and smoothness, implementation and usability testing. The product of the thesis is working application displaying points of interest through camera view of the device with operating system Android.

Keywords location-based augmented reality, Android library, sensors, points of interest, augmented reality, geo-augmented reality, camera preview

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je někdy těžké orientovat se ve velkém množství informací, které obklopují občany takřka na každém kroku, a vybrat z nich ty relevantní a důvěryhodné. Tato diplomová práce má pomoci bojovat se zmíněným jevem poskytnutím aplikace pro mobilní operační systém Android, která usnadní schopnost orientace uživatelů převážně v okolí turistických destinací. Aplikace bude zobrazovat body zájmu (restaurace, památky apod.) v okolí uživatele skrze obrazovku zařízení a pohled kamery. Informace viditelné na obrazovce závisí na pozici pozorovatele vůči zvoleným bodům a orientaci zařízení. Tato technologie se nazývá polohově zaměřená rozšířená realita.

Motivací pro vznik zadání byl požadavek integrace rozšířené reality do turistické mobilní aplikace. Jak se ukázalo, hardware v některých zařízeních není dostatečně výkonný a data poskytovaná senzory nepřesná, a proto bylo nutné navrhnout sofistikované řešení vycházející z již existujících knihoven na trhu.

Práce nevysvětluje elementární pojmy související s vývojem mobilních aplikací pro operační systém Android, ani neslouží jako komplexní návod, jak psát aplikace pro tuto platformu. Mírně pokročilý vývojář by měl být schopen implementovat prezentované postupy, obecně však čtenář pro pochopení textu nepotřebuje žádné specifické znalosti.

Práce je strukturována do čtyř samostatných kapitol. V kapitole analýza je čtenář stručně seznámen se specifikami platformy Android a základy rozšířené reality. Dále se v této kapitole vyskytuje rešerše existujících řešení a rozbor možných postupů pro návrh knihovny. Kapitola návrh definuje operace jednotlivých částí knihovny, vybírá vhodné postupy a jejím výstupem je diagram tříd použitý pro realizaci. Následuje kapitola implementace, která se zabývá implementačními detaily vyvíjené knihovny. V poslední kapitole testování jsou popsány provedené postupy k ověření správného fungování aplikace a uživatelský test, na základě kterého bude vyhodnocena kvalita výsledné aplikace.

Cílem práce je navrhnout a naprogramovat knihovnu pro mobilní operační systém Android, která bude poskytovat polohově zaměřenou rozšířenou realitu umožňující jejím uživatelům usnadněnou orientaci v prostoru. Pří návrhu je důležité zaměřit se na výkon a kvalitu výsledného zobrazení. Součástí práce bude i rešerše existujících řešení, ukázková aplikace demonstrující využití knihovny a dokumentace k použití knihovny. Výstupní program by měl být otestován samotnými uživateli, aby se osvědčilo jeho možné uplatnění v praxi.

Kapitola 1

Analýza

1.1 Operační systém Android

Android je v dnešní době nejrozšířenější operační systém pro mobilní zařízení. Je založen na modifikovaném linuxovém jádru, zdrojové kódy tohoto systému jsou dostupné jako otevřený software[1] a výrobci zařízení mohou modifikovat téměř všechny jeho části. V dnešní době se tento systém rozmáhá nejen ve sféře mobilních telefonů a tabletů, ale i v tzv. nositelné elektronice[2], chytrých brýlích, televizích a dopravních prostředcích.

1.1.1 Historie

Za zrodem této platformy stála společnost Android Inc., která byla v roce 2005 koupena společností Google. Roku 2007 byla pod vedením Google založena Open Handset Alliace, což je uskupení více než 80 společností z oblastí telekomunikací, vývoje software a výroby hardware. Tato aliance následně představila a uvolnila zdrojové kódy k Android Open Source Project. [38], [41]

1.1.2 Podíl na trhu a verze

Poslední dva roky se podíl prodeje zařízení se systémem Android na trhu s chytrými telefony pohybuje okolo 80 %, což dokládá tabulka 1.1, která znázorňuje prodeje chytrých telefonů podle operačních systémů po kvartálech v letech 2014 a 2015.

Hlavním distribučním kanálem pro šíření aplikací na platformě Android je online obchod Google Play, dříve nazývaný Android Market. Tento obchod

[1] Software s dostupným zdrojovým kódem a volným licencováním [60]

[2] Miniaturní elektronická zařízení nošená pod, uvnitř nebo na oblečení. Zářízení může být oblečením samo o sobě [69]. Typickým příkladem jsou hodinky nebo chytré náramky monitorující tělesné funkce, pozici apod.

Tabulka 1.1: Prodej chytrých telefonů na základě OS po kvartálech v letech 2014 a 2015 podle Gartner, Inc.

Operační systém	1Q14	2Q14	3Q14	4Q14	1Q15	2Q15	3Q15	4Q15	Celkem
Android	227 549	243 484	254 354	279 058	265 012	271 010	298 797	325 394	2 164 658
iOS	43 062	35 345	38 187	74 832	60 177	48 086	46 062	71 526	417 277
Windows	7 580	8 095	9 033	10 425	8 271	8 198	5 874	4 395	61 871
Blackberry	1 714	2 044	2 420	1 734	1 325	1 153	977	907	12 274
Ostatní OS	1 731	1 417	1 310	1 287	1 269	1 229	1 134	887	10 264
Celkem	281 637	290 384	305 384	367 334	336 054	329 676	352 844	403 109	2 666 424
Podíl OS Android	80,80 %	83,85 %	83,29 %	75,97 %	78,86 %	82,20 %	84,68 %	80,72%	81,18%

Tabulka 1.2: Zastoupení verzí OS Android k 4.4.2016

Verze	Kódový název	API	Zastoupení
2.2	Froyo	8	0.1 %
2.3.3 - 2.3.7	Gingerbread	10	2.6 %
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2 %
4.1.x	Jelly Bean	16	7.8 %
4.2.x		17	10.5 %
4.3		18	10.5 %
4.4	KitKat	19	33.4 %
5.0	Lollipop	21	16.4 %
5.1		22	10.5 %
6.0	Marshmallow	23	4.6 %

Tabulka 1.3: Zastoupení fyzické velikosti a jemnosti displejů u zařízení OS Android k 4.4.2016

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Celkem
Small	2.4 %						2.4 %
Normal		4.9 %	0.1 %	41.5 %	23.9 %	14.9 %	85.3 %
Large	0.3 %	4.6 %	2.2 %	0.5 %	0.5 %		8.1 %
Xlarge		3.2 %		0.3 %	0.7 %		4.2 %
Celkem	2.7 %	12.7 %	2.3 %	42.3 %	25.1 %	14.9 %	

je předinstalovaný na zařízeních drtivé většiny výrobců a společnost Google pravidelně uveřejňuje porovnání počtů aktivních zařízení na základě různých kritérií, z čehož mohou těžit vývojáři při optimalizaci aplikací. [8]

Tabulka 1.2 zobrazuje rozdělení podle verzí a tabulka 1.3 ukazuje zastoupení velikostí a jemností displejů u zařízení s operačním systémem Android. Velikost displeje se v tabulce nachází na vodorovné ose a je vyjádřena příslušným anglickým názvem. Horizontální osa obsahuje zkratky pro jednotlivé jemnosti displeje seřazené od nejnižší hodnoty pixelů na palec (low density per inch) po nejvyšší.

1.1.3 Úskalí platformy

Největším problémem programátorů software pro Android je roztříštěnost tohoto systému. Průzkum firmy OpenSignal zaměřující se na vývoj mobilních aplikací zachytil na vzorku 682 000 stažení aplikace více než 24 000 různých zařízení od 1 294 různých výrobců [61]. Díky otevřenosti zdrojového kódu má každý výrobce možnost modifikovat systém dle svých představ a také uplatnit vlastní politiku na aktualizace verzí v zařízeních. Ze zmíněných možností přizpůsobení systému logicky vyplývá existence mnoho odlišných a specifických verzí systému Android.

Dalším, méně viditelným úskalím operačního systému Android je nutnost placení licenčních poplatků držitelům různých patentů, které systém využívá. Jednou ze společností profitujících z každého prodaného zařízení je Microsoft, který vlastní mj. i patent na práci se souborovým systémem FAT (File Allocation Table). Výše poplatků je individuální pro každého výrobce a podle serveru *howtogeek.com* se pohybuje od 5 do 15 dolarů za zařízení. Společnosti se ve většině případů raději dohodnou, než aby riskovali nákladnou soudní bitvu s nejasným výsledkem. [27], [41]

1.2 Rozšířená realita

Rozšířená realita[3], je podle anglické verze serveru Wikipedia „přímý nebo nepřímý pohled na fyzický svět, který byl rozšířen přidáním virtuální, počítačově vytvořené informace“ [39]. Tuto definici použil i Borko Furht v knize Handbook of Augmented Reality [35]. Alternativně lze virtuální realitu popsat jako „pojem používaný pro zobrazení reality a následného přidání digitálních prvků“ [2] nebo „integraci digitální informace do uživatelského prostředí v reálném čase“ [35], popř. jako „obohacený obraz nebo prostředí zobrazené na obrazovce či jiném displeji, který byl vytvořen překrytím prostředí skutečného světa počítačově generovanou vrstvou dat“ [29]. Všechny tyto definice jsou takřka shodné a vysvětlují virtuální realitu jako složení reálného světa s nějakým druhem počítačově vytvořené informace.

1.2.1 Rozdíl oproti virtuální realitě

Virtuální realita se dá za určitých okolností považovat za předchůdce té rozšířené. Není zde exaktní hranice, která tyto pojmy rozlišuje, ale za hlavní rozdíl lze považovat fakt, že virtuální realita nevyužívá vstupu z kamery zařízení. Na rozdíl od rozšířené reality se zobrazovaný svět skládá buď z animaci nebo předtočených scén. [70]

[3] V anglickém originálu augmented reality, často se v literatuře používá pouze zkratka AR

1.2.2 Rozdělení

Rozšířenou realitu lze rozlišovat podle prostředí, ve kterém se využívá, na vnitřní a venkovní. Nejčastěji se však rozděluje podle způsobu volby místa ve scéně, kde je umístěn počítačově generovaný obsah.

1.2.2.1 Marker-based AR

Tento druh rozšířené reality využívá tzv. značek, které umožňují vložení virtuálního objektu do fyzického světa [68]. Specifická značka identifikující virtuální objekt musí být v systému registrována a přiřazena k požadovanému objektu ještě před použitím. Značky by měly být unikátní, vhodné se jeví použití QR kódů nebo jiných dvourozměrných čárových kódů, které je možné snadno vygenerovat a umístit do nich dodatečné informace, kterými může být např. odkaz na zobrazovaný obsah. [35]

1.2.2.2 Markerless AR

Do této kategorie lze zařadit veškeré druhy rozšířené reality, které nevyužívají značky pro identifikaci pozice virtuálního objektu na obrazovce zařízení.

- **Rozpoznávání přirozených vlastností objektů** Používá techniky pro rozpoznání obrázků, 2D a 3D předmětů, vzorů či pohybů a na základě známých kombinací zobrazuje objekty pro rozšíření této reality. Příkladem algoritmu pro rozpoznávání obrazu je detekce rohů ve scéně. Rozdíl oproti rozšířené realitě používající značky je podle [35] ve využití přirozených vlastností prostředí. Autor se přiklání k tomuto rozdělení, přestože některé publikace tento přístup zařazují do stejné skupiny, která je zde nazvána marker-based.

- **Využití senzorů zařízení** Rozšiřující obsah aplikace je přidáván na základě údajů z libovolných senzorů komunikujících se zařízením. Těmito senzory je typicky GPS modul v kombinaci s určením orientace kamery zařízení, kterou lze spočítat např. z hodnot dvojice akcelerometr a magnetometr. Tento přístup používá právě polohově zaměřená rozšířená realita[4], kterou se zabývá tato práce.

- **Uživatelský nebo jiný vstup** Rozšířená realita je do scény generována na základě nějakého vnějšího vstupu, kterým může být například dotyk uživatele, data přijatá síťovým přenosem, zvukový vstup nebo přítomnost jiného zařízení v okolí.

[4]Polohově zaměřená rozšířená realita využívá aktuální polohy a orientace zařízení k výpočtům pozice generovaného obsahu na obrazovce

Obrázek 1.1: Výhled z kokpytu stíhacího letounu s rozšířenou realitou [52]

1.2.3 Oblasti využití

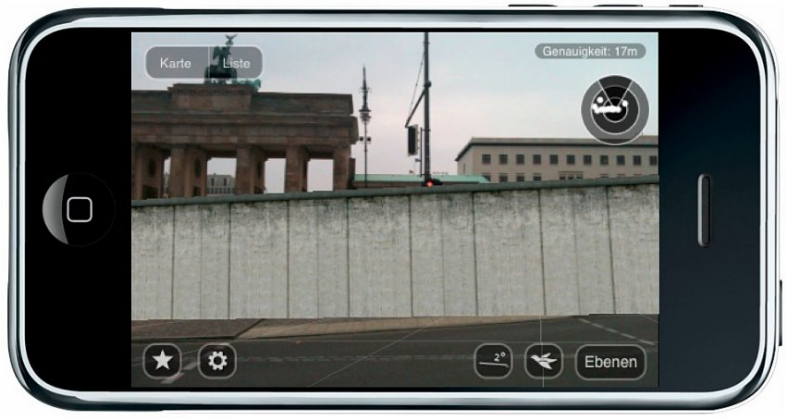
Možnosti využití rozšířené reality jsou takřka neomezené a s trochou fantazie lze této technologie využít téměř ve všech oblastech lidského života. Uvedeno je pouze několik pro autora nejzajímavějších a nejužitečnějších příkladů.

- **Armáda** V armádě se rozšířená realita využívá mj. při výcviku vojáků. Reálný svět je pomocí brýlí nebo promítání na sklo vozidla obohacen o modelové situace a takto se simulují bojové podmínky, při kterých vojáci rozhodují o řešení dané situace. Vojenské brýle pro noční vidění jsou také založeny na technologii rozšířené reality. [70]

  Velké množství vojenských operací záleží na geografii, a proto letectvo, námořnictvo i pozemní jednotky využívají rozšířené reality při navigaci [4]. Příkladem je pohled z kokpitu stíhacího letounu na obrázku 1.2.

- **Turismus** Polohově zaměřená rozšířená realita přímo vybízí k využití v oblasti cestovního ruchu. Při namíření kamery zařízení s aplikací rozšířené reality na nějaké zajímavé místo, lze zobrazit užitečné informace. U památek, které byly zničeny, lze zobrazit jejich původní vzhled přímo na místech, kde se dříve nacházely. Zajímavým a průkopnickým projektem z roku 2010 bylo zobrazení Berlínské zdi pomocí mobilní rozšířené reality viz. snímek obrazovky na obrázku 1.2. [71]

- **Televize** Časté využití technologií pro rozšířenou realitu lze pozorovat v televizních zábavních, sportovních a zpravodajských přenosech. Typickými příklady jsou animace zpráv či počasí v pozadí za moderátorem,



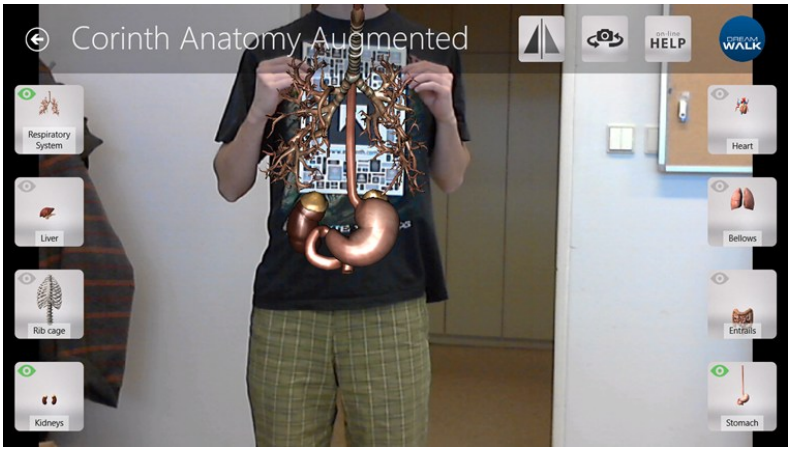
Obrázek 1.2: Zobrazení Berlínské zdi pomocí AR [71]

Obrázek 1.3: Rozdělení hřiště při živém utkání amerického fotbalu za pomoci přidaných grafických prvků [52]

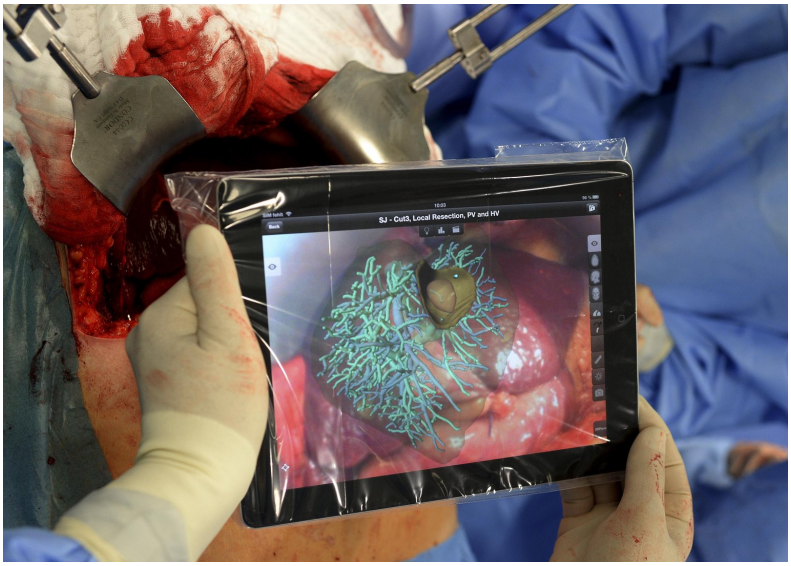
plovoucí otázky ve vědomostních kvízech či grafické pomůcky při analýze proběhlých situací v živém sportovním pořadu. Obrázek 1.3 ukazuje využití přidané grafiky při utkáních amerického fotbalu pro rozdělení území na zóny.

- **Vzdělání** Ve školách se dnes využívají interaktivní tabule a díky tabletům se objevují nové výukové možnosti. Výuku lze tedy inovovat nástroji s rozšířenou realitou pro efektivnější učení. Jedním z poskytovatelů výukových nástrojů rozšířené reality je česká společnost Corinth s.r.o.[5] vyvíjející aplikaci pro platformu Windows, která zobrazuje interaktivní trojrozměrné modely z oblastí geologie a biologie. Ukázka fungování aplikace s modelem listu je zobrazena na obrázku 1.4.

[5] http://www.ecorinth.com/



Obrázek 1.4: Vzdělávací aplikace Corinth zobrazující orgány v lidském těle [25]



Obrázek 1.5: Použití aplikace rozšířené reality zobrazující připravený model orgánu pacienta přímo při operaci [74]

- **Lékařství** Současným trendem v lékařství jsou operace za pomoci moderních technologií včetně rozšířené reality. Operujícímu doktorovi se mohou zobrazovat počítačem generované aktuální informace o pacientovi přímo při provádění operace [70].

  Na obrázku 1.5 může čtenář pozorovat reálné použití aplikace rozšířené reality, která pomáhá lokalizovat kritické struktury v těle pacienta, jako jsou tumory a cévy. Vše se děje pomocí zobrazení modelu připraveného před operací na základě provedených vyšetření. [74]

- **Herní průmysl** Na trhu se dnes objevují speciální brýle (např. Microsoft HoloLens[6]), které poskytují hráči unikátní zážitek pomocí virtuální nebo rozšířené reality.
- **Vozidla** Některá vozidla zobrazují informace o jízdě přímo na předním skle, ať už se jedná o aktuální rychlost, spotřebu paliva či navigaci na zadané místo.
- **Ostatní** Dalšími oblastmi, kde se používá rozšířená realita, jsou marketing, astronomie, filmový průmysl, překládání textů, architektura, zkoušení oblečení nebo návody na sestavení či opravení produktů [70]. Budoucnost AR může být v hologramech, videokonferencích nebo např. chytrých brýlích.

1.3 Rešerše existujících řešení

Autor práce provedl průzkum knihoven nabízejících funkce polohově zaměřené rozšířené reality určené k zobrazování bodů zájmu. Pro porovnávání kvality jednotlivých řešení byly použity telefony různých výrobců, ceny, stáří a verze operačního systému Android. Konkrétně se jednalo o Huawei P8 Lite (Android 5.0), Samsung Galaxy S III (Android 4.3) a Lenovo A2010 (Android 4.5). Pokud byly k dispozici, porovnávaly se referenční aplikace jednotlivých SDK (Software Development Kit), v opačném případě bylo nutné naprogramovat jednoduché zkušební aplikace využívající základních funkcí daných knihoven.

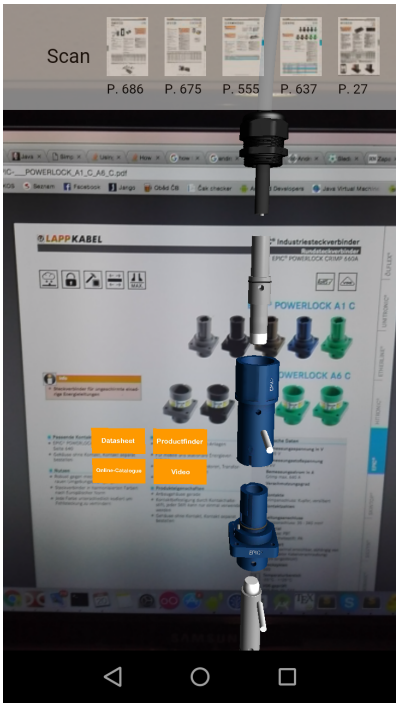
1.3.1 Dostupné vývojové sady pro Android

Níže je uvedeno shrnutí nejpoužívanějších knihoven pro Android, základní informace o nich a stručný popis implementace polohově zaměřené rozšířené reality. Pro ilustraci jsou připojeny snímky obrazovky zachycující aplikace používající popisované knihovny.

Wikitude SDK

Rakouská společnost Wikitude GmbH[7] se sídlem v Salzburgu je s více než 50 tisíci aktivními registrovanými vývojáři jedním z největších světových poskytovatelů technologií pro virtuální realitu. Současná verze vývojové sady nese název Wikitude SDK 5 a existuje pro Android, iOS, jako rozšíření do nástrojů Xamarin, Unity, Cordova, Titanium a v neposlední řadě podporuje vývoj aplikací pro chytré brýle Epson, Google Glass nebo Vuzix. Wikitude SDK nabízí mnohem více než polohově zaměřenou virtuální realitu. Dokáže rozpoznávat 2D i 3D objekty a dosazovat téměř jakékoliv objekty na značkou určená místa ve scéně.

[6] https://www.microsoft.com/microsoft-hololens
[7] `http://www.wikitude.com`



(a) Zobrazení animace na základě značky v časopisu aplikací LAPP GROUP AR[8] používající Wikitude SDK

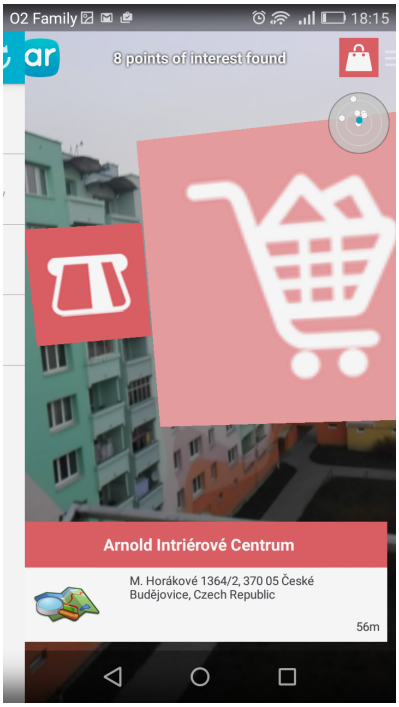
(b) Zobrazení bodů zájmu pomocí aplikace Wikitude

Obrázek 1.6: Ukázka použití Wikitude SDK

Pro účely této diplomové práce je důležitá pouze část SDK věnující se polohově zaměřené virtuální realitě. Jednotlivé části SDK nelze rozdělit a je nutné použít vždy celou knihovnu zabírající nezanedbatelných 12,6 MB. Z e-mailové komunikace s poskytovatelem vyplynulo, že rozdělení SDK neplánují. Wikitude SDK nabízí licencování závislé na aplikaci nebo časové předplatné služeb. Nejlevnější verze, která neumožňuje využívat některé pokročilé funkce, stojí 590€ za aplikaci na jedné platformě. Licence je platná po celou dobu životnosti aplikace, neposkytuje však podporu a aktualizace prostředí. Knihovnu lze bezplatně vyzkoušet po neomezenou dobu, na pozadí bude však vždy zobrazen vodoznak Wikitude. Z hlediska funkčnosti se aplikace jeví jako spolehlivá. Při testech na levnějších zařízení dosahovala dostatečné přesnosti zobrazení i výkonu při zvyšujícím se počtu bodů zájmu. Jediným zjištěným nedostatkem v této oblasti bylo řešení překryvu bodů nacházejících se za sebou. Na vzhledu značky je možné registrovat přítomnost více bodů a po kliknutí se zobrazí detail všech položek. V módu rozšířené reality je však vidět název pouze první z nich. Knihovna je rozšiřitelná a tento nedostatek je možné

[8] `http://www.lappkabel.com/`

(a) Zobrazení videa ve scéně na základě skenování obálky časopisu Top Gear za pomoci aplikace Layar



(b) Zobrazení bodů zájmu pomocí aplikace Layar

Obrázek 1.7: Ukázka použití Layar SDK

odstranit doplněním vlastní logiky zobrazování.

Obrázky 1.6 demonstrují použití frameworku Wikitude SDK v existujících aplikacích pro marker-based i location-based rozšířenou realitu.

Layar SDK

Layar[9] je společnost sídlící v Amsterdamu zabývající se tvorbou aplikací s rozšířenou realitou. Byla založena roku 2009 a od roku 2014 je součástí skupiny Blippar[10], která poskytuje profesionální řešení na míru právě v této oblasti. Jedním z produktů je vývojová sada Layar SDK, aktuálně ve verzi 8.4.4. Tato sada umožňuje vložit funkcionalitu virtuální reality do vlastní aplikace a existuje pro platformy iOS, Android a jako rozšíření frameworku PhoneGap. Layar SDK nabízí funkce zobrazování rozšířené reality, ať už se jedná o skenování značek či zobrazení na základě GPS souřadnic.

[9] `https://www.layar.com`
[10] `https://blippar.com`

Stejně jako u Wikitude je nutné zaplatit cenu celého SDK, v případě Layar SDK se jedná o 300€ měsíčně po dobu minimálně jednoho roku. K dispozici je zkušební verze na 30 dní. Prohlížení okolí v módu polohově zaměřené rozšířené reality je příjemné, zobrazují se ikony a vždy u jednoho zaměřeného bodu zájmu také detailní informace ve spodní části obrazovky. U zařízení s méně kvalitními senzory však často dochází k rychlým změnám zaměřeného bodu a je obtížné stihnout přečíst všechny detailní informace. Knihovna neposkytuje žádné řešení pro překrývající se body a někdy se dokonce zobrazují detailní informace k bodům, které nejsou viditelné. Aplikace Layar zobrazuje u blízkých bodů zájmu přehnaně velkou ikonu, což lze pozorovat na 1.7b. Knihovna je schopna plynule zobrazit desítky bodů zájmu a z hlediska výkonu je tedy naprosto dostačující.

Obrázky 1.7 demonstrují použití frameworku Layar SDK v existujících aplikacích pro marker-based i location-based rozšířenou realitu.

PanicAR

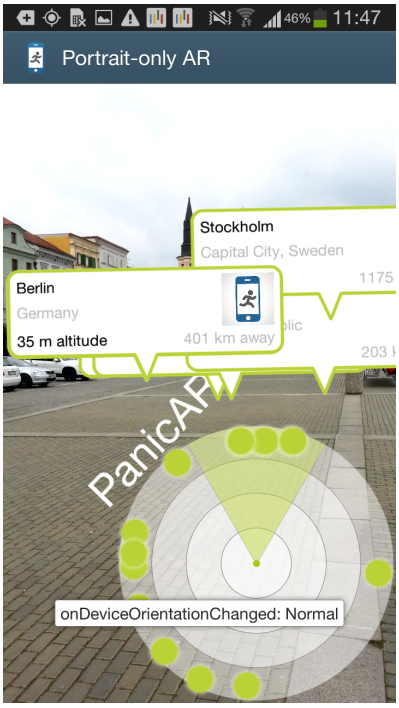
Německá společnost doPanic GmbH se sídlem v Regensburgu vyvíjí a prodává knihovnu, která poskytuje výhradně funkcionalitu polohově zaměřené rozšířené reality. Knihovna nese název PanicAR[11] a je k dispozici pro platformy iOS a Android. Současná verze pro Android má vývojové číslo 1.0.1562 a údajně existuje více než 60 aplikací používající tuto knihovnu.

Největší výhodou PanicAR je její jednoúčelovost, protože se zaměřuje pouze na polohovou virtuální realitu. Díky tomu je tato knihovna kompaktní a zabírá pouhých 200kB paměti. Integrace knihovny je jednoduchá a jednotlivé komponenty lze libovolně přizpůsobovat. PanicAR nabízí tři druhy licencí, z čehož dvě jsou účtovány zvlášť za každou aplikaci a platformu. Jedná se o neziskovou licenci za 99€ a komerční licenci za 899€. Poslední možností je zakoupení licence pro neomezené množství aplikací pro zvolenou platformu za 3499€. Zobrazení bodů zájmu je příjemné a aplikace fungovala plynule i na levnějších zařízeních. Překrývání bodů je řešeno náklonem zařízení a vystoupením zastíněných bodů. Originální je zvětšení radaru a jeho zobrazení přes celou obrazovku při položení zařízení do rovnovážné polohy.

Demo aplikaci nebylo možné spustit na Huawei P8 Lite kvůli ohlášené a již více než rok neřešené chybě. Poslední aktualizace knihovny proběhla před dvěma lety a poslední příspěvek na blogu je z data 31.7.2014. Je tedy pravděpodobné, že vývoj byl zastaven i kvůli nekomunikaci ze strany vývojářů.

Na obrázcích 1.8 lze pozorovat vzhled a řešení překryvů v ukázkové aplikaci PanicAR. Za zmínku stojí rozsáhlý radar umístěný v pravém dolním rohu aplikace.

[11] `http://panicar.dopanic.com/`



(a) Zobrazení bodů zájmu v ukázkové aplikaci PanicAR

(b) Řešení překrývání bodů v ukázkové aplikaci PanicAR

Obrázek 1.8: Ukázka použití PanicAR

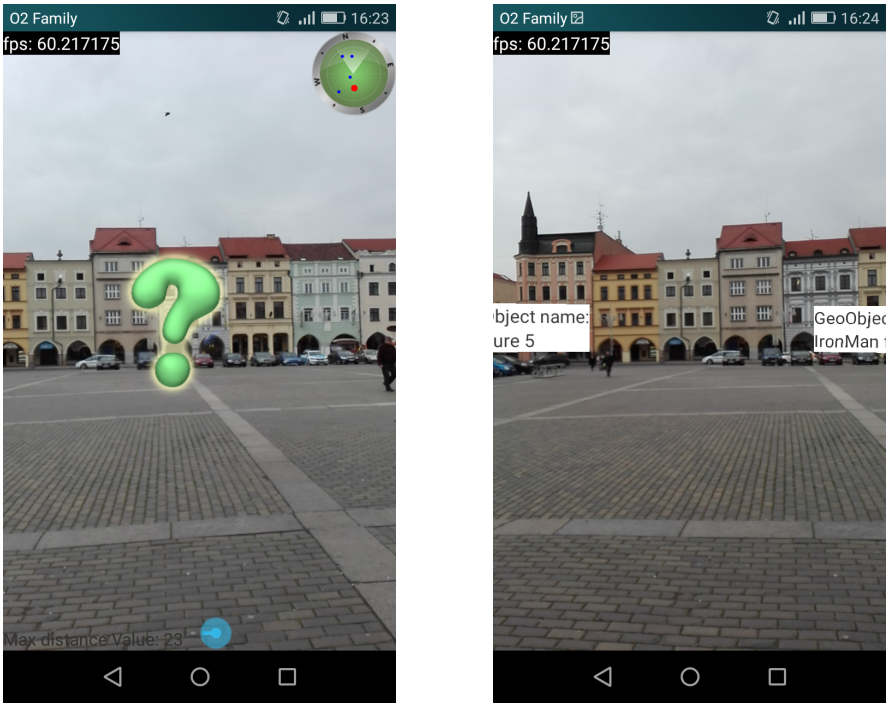
BeyondAR

BeyondAR[12] je projekt, jehož vývoj má na svědomí Joan Puig Sanz ze Španělska. Jedná se o knihovnu, která existuje pouze pro Android a umožňuje programovat aplikace s polohově zaměřenou rozšířenou realitou. Pro vylepšování a testování knihovny existuje BeyondAR Game[13], což je modulární hra využívající rozšířené reality.

Knihovnu lze používat při dodržení licence Apache 2.0 a zdrojové kódy jsou k dispozici. Zobrazení bodů zájmu je plynulé, občas však tyto body „plují“ po obrazovce a chvíli trvá než se ustálí na svém místě. Při prudších pohybech aplikaci trvá déle než se stabilizuje. Překrývání bodů není nijak řešeno. Poslední aktualizace knihovny proběhla zhruba před dvěma roky.

Na obrázcích 1.9 jsou zobrazeny snímky obrazovky z různých módů příkladové aplikace BeyondAR.

[12] `http://beyondar.com/`
[13] `https://play.google.com/store/apps/details?id=com.beyondar`



(a) Zobrazení ikony a radaru v ukázkové aplikaci BeyondAR

(b) Zobrazení bodů zájmu v ukázkové aplikaci BeyondAR

Obrázek 1.9: Ukázka použití příkladové aplikace BeyondAR

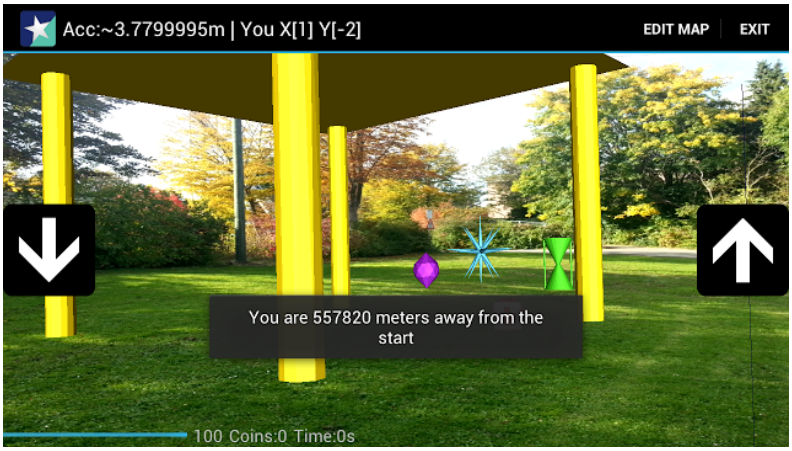
DroidAR

DroidAR[14] je framework pro Android od německé společnosti s názvem bitstars[15], který se soustředí na rozšířenou realitu. Je možné ho využít jak pro polohově zaměřené aplikace, tak i na zobrazování objektů na místě v obrazu, na kterém jsou detekovány určené značky.

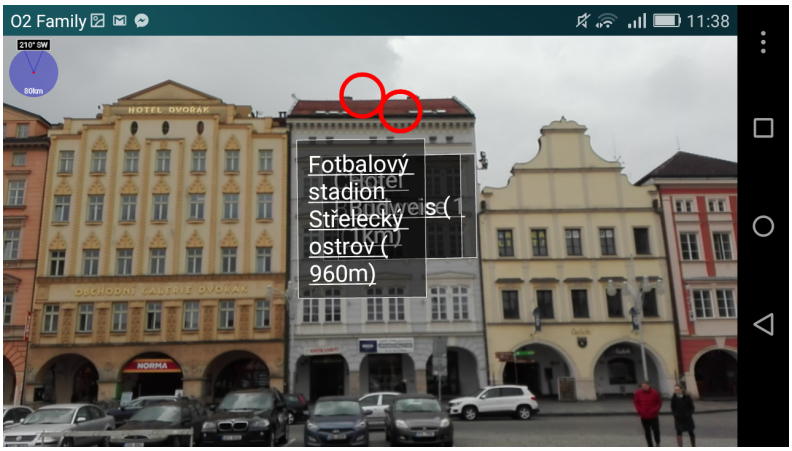
DroidAR poskytne základní funkcionalitu pro zobrazení bodů zájmu, ale nijak neřeší překrývání. K dispozici jsou zdrojové kódy a framework lze využívat pod GNU GPL licencí. Poslední úpravy kódu proběhly před více než třemi roky.

Obrázek 1.10 převzatý z [43] demonstruje použitelnost knihovny DroidAR dosazením 3D objektů na určené souřadnice.

[14] `http://droidar.blogspot.cz/`
[15] `http://bitstars.com/`



Obrázek 1.10: Aplikace využívající knihovnu DroidAR



Obrázek 1.11: Ukázka použití knihovny Mixare

Mixare

Mixare[16] je knihovna, za kterou stojí Daniele Gobbetti ze společnosti Peer GmbH[17] sídlící v Itálii. Tato knihovna poskytuje funkce rozšířené reality na základě pozice.

Zdrojové kódy projektu jsou k dispozici a celý projekt je licencován pod licencí GNU GPLv3. Stejně jako DroidAR i Mixare poskytuje pouze základní funkci zobrazení bodu zájmu.

Jak vypadá aplikace využívající tuto knihovnu lze vidět na obrázku 1.11.

[16] `http://www.mixare.org/`
[17] `http://www.peer.biz/en/`

Obrázek 1.12: Ukázka aplikace Junaio

Metaio / Junaio SDK

Metaio GmbH[18] byla německá společnost se sídlem v Mnichově zabývající se již od roku 2003 vývojem softwarových řešení pro rozšířenou realitu. Metaio poskytovalo licence na své SDK s pokročilými funkcemi rozšířené reality pro iOS, Android, Windows a rozšíření pro vývoj v Unity. Jedním z produktů postaveným na Metaio SDK byl mobilní prohlížeč rozšířené reality s názvem Junaio. Ten poskytoval pravděpodobně nejpokročilejší polohově zaměřenou rozšířenou realitu pro Android co se týče výkonu i pohodlnosti používání. V květnu roku 2015 byla však společnost Metaio koupena společností Apple, již dále nenabízí své SDK a aplikace byla dokonce stažena z obchodu Google Play.

Jak lze vidět na obrázku 1.12, aplikace Junaio používala zcela unikátní přístup zobrazování bodů zájmu, který byl uživateli shledán přehledný a intuitivní.

1.3.2 Zhodnocení existujících řešení

Knihovny zmíněné v předchozím odstavci byly otestovány podle různých kritérií a ohodnoceny na stupnici od 1 do 10, kde 10 je nejlepší skóre. Vynásobením získaných bodů a váhy jednotlivých kritérií bylo vypočítáno celkové hodnocení vhodnosti použití dané knihovny. Srovnání nebere v potaz ceny a licenční podmínky a zaměřuje se výhradně na funkčnost, vzhled a složitost integrace.

Metodika porovnání

Seznam kritérií, podle kterých byly vývojové sady porovnávány:

[18] `http://www.metaio.com/`

- Výkon (30 %)

  Hodnocení výkonu závisí na plynulosti aplikace při zvyšujícím se počtu bodů zájmu ve scéně. Tato část byla vyhodnocena jako nejdůležitější, a proto se podílí z téměř jedné třetiny na celkovém hodnocení.

- Orientace v informacích (20 %)

  Jedná se o celkový dojem a použitelnost aplikace pro její uživatele. Testování provádělo několik osob rozdílného věku, pohlaví a s různými zkušenostmi s používáním technologií.

- Řešení překryvů (10 %)

  Toto kritérium zohledňuje chování aplikace v situaci, kdy jsou některé body zájmu zastíněny jinými. Boduje se originalita, intuitivita a správná funkčnost. Některé SDK jsou rozšiřitelné a umožňují doimplementovat tuto funkcionalitu, to však na hodnocení nemá vliv.

- Reakce na otáčení zařízení (15 %)

  Hodnotí se reakce na změny dat ze senzorů. Testovány byly různě rychlé pohyby zařízením i nehybný stav. Měřítkem bylo jak plynule a za jakou dobu se bod zájmu ustálí na místě ve scéně, kde by měl být. Snížení hodnocení přinesla nestabilita bodů v situaci, kdy se se zařízením nehýbalo.

- Dokumentace a složitost použití (15 %)

  Hodnocena je orientace v dokumentaci, úroveň existujících návodů, příkladů k použití a náročnost implementování knihovny do aplikace. Komplexnější SDK mají obecně nevýhodu ve větším počtu funkcionalit k popisu a pochopení.

- Radar (10 %)

  Radar pomáhá uživateli ke snadnější orientaci v prostředí a je nedílnou součástí téměř všech knihoven polohově zaměřené rozšířené reality. Hodnotí se grafické zpracování, plynulost a správnost otáčení, umístění ve scéně a lze získat body za jedinečné nadstandardní funkce.

Výsledky porovnání

V tabulce 1.4 lze pozorovat bodové hodnocení jednotlivých knihoven. Podle předpokladů se na prvních čtyř místech umístila placená řešení. Další příčky obsadily knihovny s veřejně dostupným zdrojovým kódem. Aplikace Junaio u uživatelů triumfovala hlavně příjemným vzhledem, vhodným řešením překrývajících se bodů a jednoduchou orientací se v okolí. Knihovna Wikitude

[19] Celkové hodnocení ovlivnila absence dokumentace, která byla dopočítána poměrově z ostatních ukazatelů.

Tabulka 1.4: Porovnání existujících SDK posyktujících funkce polohově zaměřené rozšířené reality pro platformu Android

Název SDK	Celkem	Výkon	Orientace v informacích	Řešení překryvů	Reakce na otáčení zařízení	Dokumentace a složitost použití	Radar
Junaio	9[19]	9	10	8	9		8
Wikitude SDK	8	10	8	3	9	7	7
PanicAR	7,65	8	7	6	7	10	7
Layar SDK	6	8	6	1	8	2	8
BeyondAR	5,7	7	5	0	3	9	8
Mixare	4,75	6	4	0	5	8	2
DroidAR	4,05	5	3	0	5	8	0

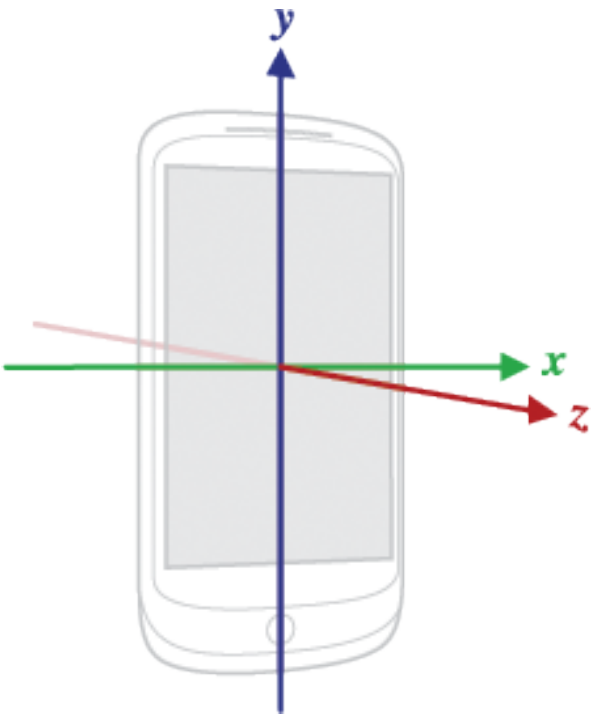
na druhém místě vyniká svou spolehlivostí a vysokým výkonem při vyšším množství bodů zájmu.

1.3.3 Závěr

Průzkumem existujících vývojových sad pro polohově zaměřenou rozšířenou realitu bylo zjištěno, že neexistuje uživatelsky příjemné, bez složitých úprav použitelné a cenově přijatelné řešení pro problematiku zobrazování bodů zájmu v turistických a městských průvodcích pro platformu Android. Velké množství knihoven bylo zrušeno, další jsou již delší dobu neaktivní a trh ovládají velké firmy Layar a Wikitude, které se však více soustředí na jiné druhy rozšířené reality. Nejpokročilejší aplikací zobrazující body zájmu v okolí na základě polohy uživatele bylo Junaio využívající SDK Metaio, které po akvizici Applem již není nadále dostupné. Každá knihovna byla schopna poskytnout základní funkcionalitu, a proto je možné se inspirovat zdrojovými kódy, vzhledem a chováním a zajímavými nestandardními funkcemi, jako je například zvětšení radaru přes celou obrazovku při natočení kamery směrem dolů.

1.4 Senzory

Zjištění orientace kamery zařízení je elementárním prvkem pro vyvíjenou knihovnu polohově zaměřené rozšířené reality. Ke zjištění, kam objektiv směřuje, je nutné využít dat ze senzorů. Kvůli velké diversitě zařízení s operačním systémem Android existuje mnoho různých senzorů, které někdy poskytují méně přesné informace. Je tedy vhodné pokusit se využít všech dostupných senzorů pro zlepšení uživatelského prožitku uživatelů. Senzory se dělí na hardwarové a virtuální. Virtuální senzory nejsou fyzická zařízení a zpravidla využívají vestavěných hardwarových senzorů. Podle charakteristik je lze rozdělit do tří kategorií.



Obrázek 1.13: Koordinační systém použitý pro senzory v zařízeních se systémem Android [21]

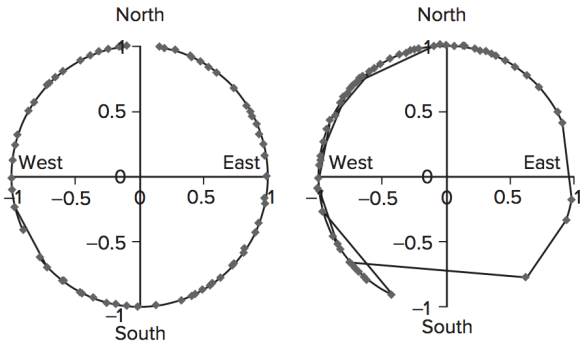
1.4.1 Pohybové senzory

Mezi hardwarové pohybové senzory se řadí akcelerometr[20] a gyroskop[21]. Další tři senzory mohou být jak hardwarové, tak softwarové. Jedná se o senzory měřící lineární zrychlení, gravitaci a rotační vektor. Pohybové senzory jsou užitečné, protože dokáží monitorovat pohyb zařízení jako je náklon, rotace či zatřesení. Obvykle poskytují naměřené hodnoty pro všechny tři osy zařízení. Definici os pro tyto senzory může čtenář pozorovat na obrázku 1.13. [15]

1.4.2 Poziční senzory

Poziční senzory jsou užitečné k měření fyzické pozice zařízení ve světě. Mezi tyto senzory patří GPS, senzor měřící geomagnetické pole, který může zjistit pozici zařízení k magnetickému severnímu pólu a softwarový senzor orientace, který je však zastaralý a místo něj se doporučuje používat funkci `getOrientation`. Při měření geomagnetického pole je nutné si uvědomit, že přítomnost větších kovových předmětů v blízkosti zařízení má negativní vliv

[20] Senzor měřící lineární zrychlení[3]
[21] Zařízení, které je schopné rozeznat svou orientaci v prostoru[44]



Obrázek 1.14: Rozdíl mezi měřením orientace bez a v přítomnosti kovového předmětu [55]

na přesnost [55], což lze vidět na obrázku 1.14. Platforma dále poskytuje senzor měřící vzdálenost přístroje od objektu. [17]

1.4.3 Senzory měřící okolní prostředí

Do této kategorie patří senzory měření teploty, světla, tlaku a vlhkosti. Praktické využití pro popisovaný typ knihovny může mít snad jen senzor tlaku, podle kterého lze určit nadmořská výška. Méně pravděpodobné využití má světelný senzor, který může např. ovládat automatické osvětlení pozorované scény. [16]

1.4.4 Přístup k senzorům

Pro přístup k senzorům se využívá systémová služba s názvem správce senzorů reprezentovaná třídou `SensorManager`, kterou získáme voláním funkce `getSystemService` s parametrem `Context.SENSOR_SERVICE`. Správce senzorů se můžeme dotazovat na výchozí senzor požadovaného typu pomocí metody `getDefaultSensor`, která vrací instanci typu `Sensor` s dodatečnými informacemi o senzoru a `null` v případě neexistence senzoru v zařízení. Senzorů stejného typu může být přítomno více, v praxi se zpravidla využívá právě ten, který je označen jako výchozí. [21]

Pro monitorování surových senzorových dat je třeba implementovat `SensorEventListener` funkcemi `onAcurateChanged` a `onSensorChanged`. První jmenovaná funkce je volána při změně přesnosti údajů poskytovaných zvoleným senzorem a druhá při poskytnutí nových aktuálně naměřených hodnot. Přesnost může být nízká, střední, vysoká nebo nedůvěryhodná. Při nevhodné přesnosti lze uživatele vyzvat k otočení zařízení ve všech jeho osách, čímž se spustí mechanismus automatické kalibrace integrované přímo v operačním systému. Samotný začátek monitorování zvoleného senzoru provádí správce senzorů skrze funkci `registerListener`, které je předána třída im-

Tabulka 1.5: Konstanty zpoždění senzorů

Název konstanty	Zpoždění	Vhodné použití
SENSOR_DELAY_NORMAL	200 ms	Změna orientace zařízení
SENSOR_DELAY_GAME	20 ms	Hry využívající senzory
SENSOR_DELAY_UI	60 ms	Dostačuje pro běžné překreslování obrazovky
SENSOR_DELAY_FASTEST	0 ms	Specializované aplikace, které existenčně závisí na datech ze senzorů

plementující `SensorEventListener`, instance senzoru a konstanta navrhující interval získávání hodnot senzoru viz tabulka 1.5. Lze také specifikovat vlastní navrhovanou absolutní hodnotu zpoždění, systém se však může rozhodnout poskytnout hodnoty pomaleji nebo častěji. [21], [55]

Při nutnosti získávat aktualizace ze senzorů častěji je možné využít JNI (Java Native interface)[22], není to však triviální [51].

1.4.5 Odstranění chyb

Při práci se senzory různé kvality, výrobců a prostředí dochází k rozlišným chybám v měření, které je třeba co nejvíce eliminovat. Existují dvě základní techniky, které se snaží odstranit nepřesnosti a výchylky ve výsledcích měření senzorů.

- **Filtry** Cílem filtrování je vyhlazení dat přicházejících ze senzorů tak, aby byly změny co nejvíce plynulé a přesné. Mezi filtry, které je možné použít, patří *Low-pass*, *High-pass*, *Bandpass* a *cut-of-frequency* filtr. [55]

- **Kombinace senzorů** Kombinováním dat z různých senzorů lze využít silných stránek každého z nich a eliminovat ty slabé.

1.4.6 Obecné rady pro práci se senzory

Níže je uvedeno několik obecných rad pro práci se senzory.

- Před použitím zkontrolovat přítomnost senzoru
- Vhodně volit intervaly doručování hodnot
- Neblokovat metodu `onSensorChanged`
- Nepoužívat zastaralé metody a typy senzorů
- Zrušit odposlech senzorů při přechodu aktivity do pozadí [21]

[22]Rozhraní umožňující volat funkce a knihovny napsané v jiných programovacích jazycích z prostředí jazyka Java

1.5 Náhled fotoaparátu

Nedílnou součástí aplikací rozšířené reality je tzv. *camera preview*, což lze přeložit jako náhled fotoaparátu. Fotoaparát dokáže zobrazit aktuální obraz okolního reálného světa v určitém úhlu na displeji zařízení. Tento náhled může sloužit jako základní vrstva pro rozšířenou realitu, na kterou je umisťován digitálně generovaný obsah.

1.5.1 Nastavitelné parametry náhledu

Fotoaparát v mobilním zařízení je přednostně určen k pořizování obrazových záznamů a při jeho používání pro nepřetržité pozorování okolí je třeba modifikovat několik parametrů.

Blesk/přisvětlovací dioda

Je nutné brát v potaz existenci blesku u většiny zařízení a rozhodnout se, jak s ním v rámci aplikace pracovat. Možnosti práce s bleskem jsou naznačeny v následujícím seznamu.

- **Trvale vypnutý nebo zapnutý** To, zdali je vhodné trvale zapnout nebo vypnout blesk, záleží hlavně na prostředí, kde se bude aplikace využívat. Pro opodstatněné případy nočního provozu může být vhodné přisvětlovací diodu trvale zapnout, ale je třeba také hledět na spotřebu baterie zařízení, která je se zapnutým bleskem vyšší.

- **Možnost zapnout** Umožnit uživateli zapnutí přisvětlovací diody je rozumným kompromisem, ale problematické může být umístění tlačítka na obrazovku, které znepřehlední aplikaci a zmenší prostor pro zobrazování obsahu. Alternativou je využít k zapnutí některé z hardwarových tlačítek zařízení, pokud existují.

- **Automatické vypínání a zapínání** Systém sám podle světelného čidla dokáže rozhodnout, kdy je správná chvíle scénu osvětlit a kdy ne. Tento přístup se však nedoporučuje z důvodu nedeterministického chování, které může rušivě působit na uživatele.

Zaostřování

Zaostřování záleží na vzdálenosti fotografovaného objektu. Ostrost znamená, že bod z reálné scény je zobrazen jako bod na výsledné fotografii. Bod považujeme za rozostřený, pokud se zobrazí jako kruh určitého průměru.[67] Android SDK poskytuje několik módů ostření, ze kterých je nutné vhodně zvolit z následujících možností. Příklad zaostřeného objektu na rozostřeném pozadí se nachází na obrázku 1.15.

Obrázek 1.15: Selektivní zaostření na blízký objekt [66]

- **Automatické** Tento mód automaticky zaostřuje podle vzdálenosti objektů, pro pozorovatele může být rozptylující neustálé rozostřování obrazu.
- **Průběžné pro video** Mód vhodný pro nahrávání videa, protože poskytuje plynulý přechod mezi jednotlivými zaostřeními.
- **Průběžné pro obrázky** Podobný mód jako předchozí, je však primárně určen pro fotografování a rychlost zaostřování je agresivnější než u videa.
- **Extended depth of field (EDOF)** Rozšířená hloubka pole kombinuje více snímků s různými hloubkami zaostření pro získání lepšího snímku [30].
- **Neměnné** Zaostřovací vzdálenost je nastavena na fixní, výrobcem zvolenou hodnotu a dále se nemění. Při pozorování objektů v různých vzdálenostech bude docházet k rozostřování obrazu.
- **Nekonečno** Zaostřovací vzdálenost je nastavena na nekonečno.
- **Makro** Mód, který je vhodný pro fotografování blízkých objektů.

Minimální a maximální FPS náhledu

Z podporovaných hodnot lze vybrat maximální a minimální počet rámců za sekundu, které se zobrazují při náhledu.

Další

Mezi další parametry, které je možno využít v souvislosti s fotoaparátem přenosného zařízení, patří volba scény a možnosti vyvažování bíle[23].

[23] Vyvážení bílé, neboli white balance je proces odstranění nerealistických zbarvení tak, že objekty bílé ve skutečnosti, budou bílé i na fotografii [26]

1.5.2 Rozměry náhledu a úhel kamery

Fotoaparáty v odlišných zařízeních se stejně jako typy obrazovek liší. Pro správné a pohodlné fungování polohově zaměřené rozšířené reality je třeba zobrazit náhled přes celou obrazovku, jejíž rozměr nemusí být vždy jedním z podporovaných rozlišení fotoaparátu v zařízení. Je proto třeba zvolit a vhodně transformovat některé z podporovaných rozlišení.

Dále je nutné zjistit úhel, který kamera zabírá, aby bylo možné správně zobrazovat body zájmu.

1.5.3 Android API pro přístup k fotoaparátu

Android SDK poskytuje nástroje pro přístup k fotoaparátu a vytvoření náhledu s požadovanými parametry. Součástí nového API (Application Programming Interface) je kompletně přepracovaný přístup, který nabízí více možností pro práci s fotoaparátem. Pro toto nové *Camera2 API* však neexistuje dostatečně komplexní návod k použití. Mnoho vývojářů ho odmítá používat a na fórech jej označuje za nedopracované. *Camera1 API* je podle oficiální dokumentace zastaralé, ale plně funkční i na zařízeních s novějšími verzemi systému.

Camera1 API

O otevření instance fotoaparátu se stará statická funkce `open` třídy `Camera`. Vlastnosti fotoaparátu lze získat voláním metody `getParameters` na instanci třídy `Camera` vrácenou zmíněnou funkcí `open`. Podle orientace zařízení lze nastavit také orientaci náhledu pomocí metody `setDisplayOrientation`. Mapování na zvolený `SurfaceView` probíhá předáním jeho `SurfaceHolder` jako argument metody `setPreviewDisplay` volané na instanci fotoaparátu. To se děje v implementaci abstraktní metody `surfaceCreated` rozhraní `SurfaceHolder.Callback`. Třída implementující toto rozhraní musí dále definovat metodu `surfaceChanged`, kde je vhodné spustit náhled voláním metody fotoaparátu s názvem `startPreview`. Pomocí `getHorizontalViewAngle` a `getVerticalAngle` je možné zjistit informace o rozsahu čočky fotoaparátu.

Camera2 API

Nová verze API pro fotoaparát poskytuje podporu na dvou úrovních oprávnění. Limitovaná úroveň poskytuje stejné funkce jako starší přístup, ale s čistším a efektivnějším rozhraním. Plný přístup k fotoaparátu pak umožňuje využití funkcí na nižší úrovni abstrakce.

Třída `CameraManager` se používá k procházení, dotazování a přístupu ke všem dostupným kamerám v zařízení (v současnosti je typická existence přední a zadní kamery). Identifikátor požadovaného fotoaparátu získáme pomocí

třídy `CameraManager` z listu vráceného metodou `getCameraIdList`. Ke zvolenému fotoaparátu se lze připojit pomocí metody `openCamera` s parametrem, kterým je funkce, jež je provedena při úspěšném připojení. Díky metodě `getCharacteristics` lze získat instanci třídy `CameraCharacteristics` pro zvolený fotoaparát a přistupovat k jeho vlastnostem.

Náhled se v novém API zobrazuje na `TextureView`.

1.6 Práce s vlákny

Správná práce s vlákny je stěžejní pro výkon aplikace, protože zařízení se systémem Android často disponují nevýkonným hardware. Zobrazování a přepočítávání většího počtu bodů zájmu jsou operace náročné na výpočetní výkon, a proto je vhodné využít vícevláknového modelu. Tato sekce seznamuje čtenáře se základními prvky vícevláknových aplikací poskytovanými Android SDK.

1.6.1 Běhové prostředí

Systém Android obsahuje modifikované Linuxové jádro přizpůsobené k efektivnímu a bezpečnému běhu více procesů. Programy napsané v jazyce Java jsou přeloženy do Java byte-code a následně do *Dalvik executable format* (DEX) [50]. Tento formát je poté spouštěn v běhovém prostředí ART (*Android runtime)*, které je od verze 5.0.0 plně kompatibilním nástupcem virtuálního stroje Dalvik. ART oproti Dalvik využívá kompilaci *ahead-of-time*[24], má vylepšený garbage collector[25] a bohatší možnosti ladění. [9]

1.6.2 Procesy a vlákna

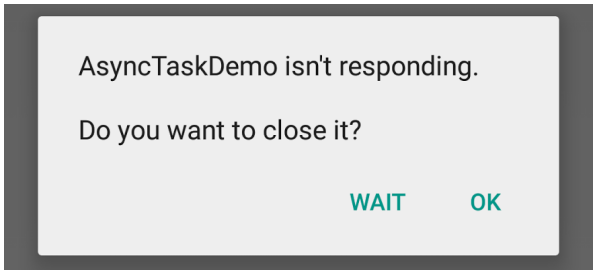
Každý proces běží nezávisle na ostatních a operační systém mu přiděluje malé množství procesorového času po krátkých intervalech. Ve výsledku to vypadá, že i jednoprocesorová zařízení dokáží pracovat s více než jednou aplikací zároveň. V rámci procesu může existovat více vláken, z nichž každé spouští sekvenčně své instrukce. Procesorový čas je rozdělen mezi vlákna obdobně jako mezi procesy. Zatímco proces nemůže přímo komunikovat s daty v adresním prostoru, vlákna mezi sebou mohou v rámci procesu komunikovat a sdílet data. [50]

1.6.3 Hlavní vlákno

Hlavní vlákno (anglicky *Main thread* nebo *UI thread (User Interface)*) je vytvořeno při startu aplikace a slouží k provádění operací, které manipulují s prvky uživatelského rozhraní, což je ostatním vláknům zakázáno. Pro jednoduché aplikace stačí využití hlavního vlákna, ale pro aplikace provádějící

[24] Při instalaci jsou DEX soubory aplikace zkompilovány přímo pro dané zařízení [9]

[25] Garbage collector (Správce paměti) má na starost úklid a přidělování paměti [64]



Obrázek 1.16: ANR (aplikace neodpovídá) dialog [24]

blokující[26] nebo výpočetně náročné operace tento přístup již stačit nemusí. [18]

1.6.4 Dialog zamrzlé aplikace

Operační systém Android monitoruje odezvu aplikace. Pokud aplikace nereaguje na uživatelský vstup po dobu pěti sekund, nastane událost ANR (*An Application Not Responding*) a systém vyvolá zobrazení dialogového okna (obr. 1.16) s textem upozorňujícím na zamrznutí aplikace a možností jejího násilného ukončení. [1]

1.6.5 Překreslování obrazovky

Pro oko uživatele je snadno postřehnutelné zpoždění již o velikosti 200 milisekund. OS Android se snaží překreslovat obrazovku shodně s obnovovací frekvencí displeje, zpravidla tedy 60 rámců za sekundu, což je pouze 16,67 ms na snímek [50]. Obecné doporučení pro vytváření plynulé a pro uživatele příjemné aplikace tedy zní provádět časově náročné operace v samostatných vláknech a neblokovat hlavní vlákno, které by mělo obsahovat pouze funkce pro překreslování obrazovky.

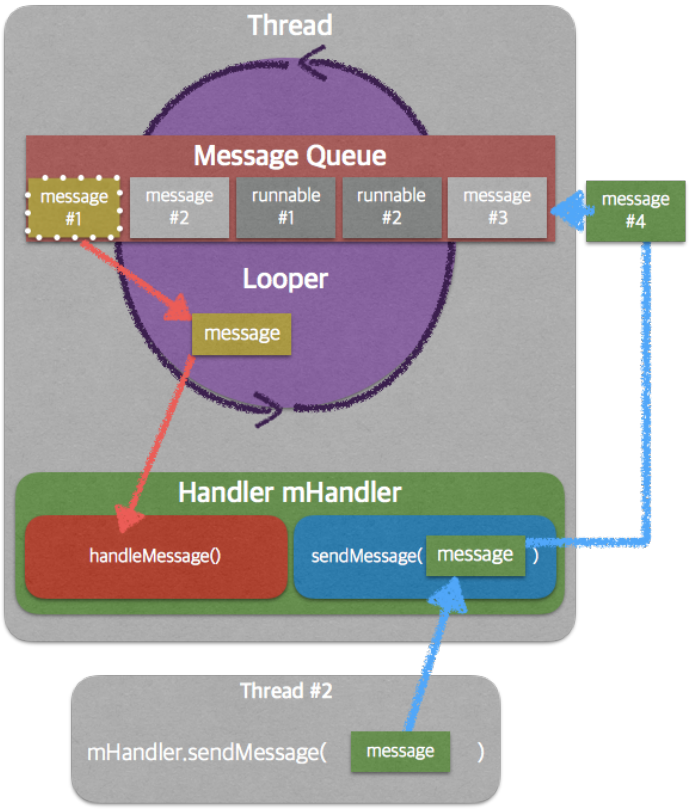
1.6.6 Handlery

Třída `Handler` pomáhá spravovat posílání zpráv mezi procesy a plánování v systému Android. Umožňuje vkládat do fronty úkoly, které mají běžet v jiném vláknu. Při vytvoření handleru dojde k jeho asociaci se třídou `Looper`, která obsluhuje frontu a produkuje instance objektů `Message` a `Runnable`. [1]

Třída `Looper` zná přidružené vlákno, obsahuje referenci na `Looper` hlavního vlákna a mj. poskytuje metody pro práci s frontou zpráv. `Looper` hlavního vlákna je nastaven automaticky běhovým prostředím Android. [12]

Pokud je vytvořena instance třídy `Handler` bez argumentu, je automaticky asociována s frontou aktuálně běžícího vlákna (typicky hlavního). Je

[26] Blokující operace nutí aplikaci čekat na výsledek než je možné pokračovat na další instrukci [56]. Typickým příkladem je manipulace se soubory či síťová komunikace.



Obrázek 1.17: Komunikace mezi vlákny za pomoci handleru [33]

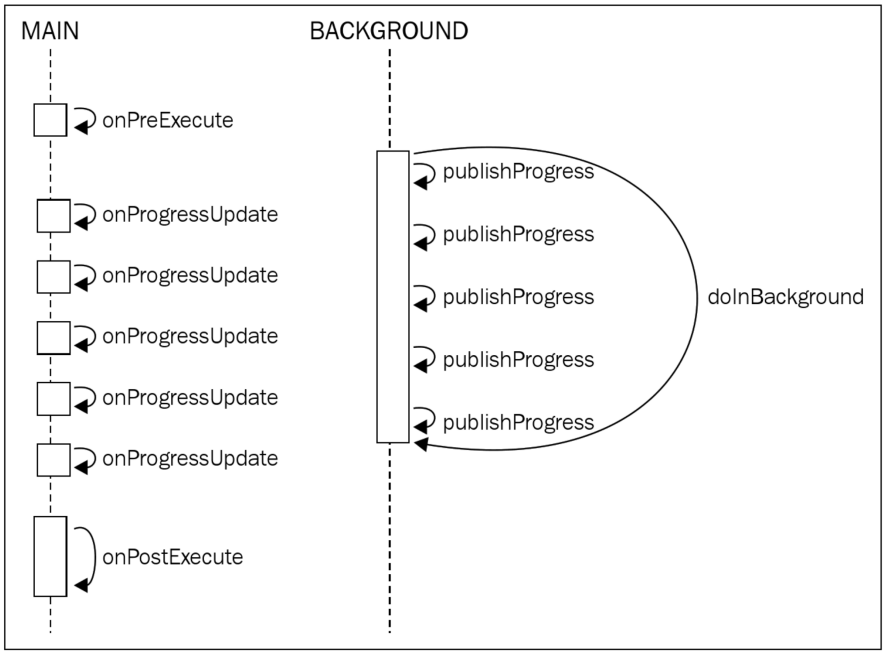
také nutné implementovat metodu `handleMessage`, která slouží ke zpracování zprávy ve frontě. Zprávu lze vložit do fronty voláním některé metody z rodiny `sendMessage`[27] na instanci třídy `Handler`. Třída `HandlerThread` umožňuje jednodušší práci s vlákny, je potomkem třídy `Thread` a poskytuje `Looper`. Obrázek 1.17 znázorňuje roli handleru v komunikaci mezi vlákny. [1]

Pro spuštění požadovaného kódu v hlavním vláknu aplikace z jiného vlákna je možné volat metodu `runOnUiThread` na instanci třídy `Activity`.

1.6.7 Asynchronní úkol

`AsyncTask` je konstrukt, který zapouzdřuje práci s vlákny a umožňuje vývojáři provést zvolenou operaci na pozadí, publikovat její průběh a výsledek použít k aktualizaci uživatelského rozhraní v hlavním vláknu aplikace. Použití je vhodné pro časově náročné a jednorázové operace informující uživatele o svém průběhu [50]. Pořadí a spouštěcí vlákno volaných metod instance třídy `AsyncTask` je nejlépe patrné ze sekvenčního diagramu na obrázku 1.18, z něhož

[27] Lze poslat prázdnou zprávu, prioritní zprávu a zprávu naplánovanou na určitý čas popř. po uplynutí specifikovaného časového intervalu [10]



Obrázek 1.18: Sekvence volání metod v AsyncTask [50]

také vyplývá, že veškeré náročné a blokující operace by měly být prováděny v metodě `doInBackground`, která je spouštěna mimo hlavní vlákno aplikace.

1.6.8 Synchronizace

Synchronizovaný blok v jazyku Java je blok kódu nebo metoda označený klíčovým slovem synchronize s právě jedním objektem, přes který se synchronizuje. V jednom okamžiku se může nacházet pouze jedno vlákno uvnitř synchronize bloku nad daným objektem. Ostatní vlákna jsou blokována, dokud jiné vlákno blok neopustí. Jedná se o první a nejjednoduší mechanizmus řešení konkurence vláken v Javě. [45]

Z uvedeného vyplývá, že je vhodné synchronizovat co nejkratší bloky kódu, aby docházelo k co nejmenšímu blokování vláken.

Klíčové slovo *volatile*, uvedené před proměnnou, označuje proměnnou dostupnou z více vláken a zajišťuje její načtení z hlavní paměti. Vždy je tedy zajištěna správná a aktuální hodnota oproti klasickým proměnným, které se mohou dočasně ukládat kvůli optimalizaci. [59]

Rozhraní `Lock`, neboli česky zámek, je další možností pro kontrolu přístupu ke sdíleným proměnným, kterou poskytuje jazyk Java. Jednou z výhod některých jeho implementací je funkce metody `TryLock`. Vlákno, které zavolá tuto metodu na instanci zámku, se ho tím pokusí zamknout. V případě, že se to nepovede, vlákno nečeká na jeho uvolnění, ale vrátí se zpět k provádění

svých instrukcí. Tento přístup může být zvláště výhodný pro synchronizaci hlavního vlákna, které nesmí být blokováno. [62]

V jazyku Java existují další a pokročilejší techniky pro zajištění správné práce se sdílenými proměnnými mezi vlákny, těmi se ale tato práce nezabývá. Autor se domnívá, že jednoduchá synchronizace pomocí klíčového slova synchronize bude pro účely zadání stačit. Více informací o pokročilých technikách synchronizace může čtenář získat v oficiální dokumentaci k balíčku java.util.concurent, v článku od Larse Vogela [48] nebo v některé z mnoha existujících publikací o programování v jazyku Java.

1.7 Zjištění polohy zařízení

Pro správné fungování aplikace s polohově zaměřenou rozšířenou realitou je nezbytná, jak již z názvu vyplývá, co nejpřesnější znalost aktuální pozice zařízení. Vysoká přesnost polohy a její časté aktualizace však mohou mít negativní dopad na výkon, baterii a velikost přenášených dat, a proto je třeba vhodně zvolit zdroj dat.

Výběr vhodné možnosti zjištění polohy je kompromisem mezi přesností a spotřebou baterie. Dále je třeba brát v potaz profil prostředí, ve kterém bude aplikace používána, a to především viditelnost na GPS satelity a hustotu WiFi přístupových bodů nebo GSM vysílacích věží v okolí.

1.7.1 Možnosti zjištění polohy

GPS

GPS (globální polohovací systém) je provozovány armádou USA a skládá se ze tří segmentů. Vesmírný segment je tvořen minimálně 24 aktivními satelity obíhajícími planetu. Kontrolní segment se skládá z pozemních radarových stanic monitorujících pozice satelitů a kontrolujících jejich chování. Uživatelský segment se skládá ze všech uživatelů, kteří vlastní přijímač GPS signálu. [32]

Udávaná přesnost GPS je v řádech jednotek a desítek metrů. Pro použití GPS je nezbytná přímá viditelnost minimálně čtyř družic, což značně omezuje použití uvnitř budov [47]. Tato technologie funguje přesněji mimo zástavbu, kde nedochází k odrazu signálu od budov [40].

WiFi triangulace

Technologie WiFi může být kromě připojení k Internetu využita i ke zjištění lokace. Pozice je vypočítána na základě znalosti polohy sítě s daným SSID (identifikátor bezdrátové WiFi sítě) a aktivního skenování síly signálu WiFi sítí v okolí zařízení. Tento způsob je vhodný pro použití ve městech a oblastech s velkou koncentrací WiFi přístupových bodů. Pro využití této funkce je potřeba mít zapnutý WiFi přijímač a aktivní internetové připojení, aby bylo

možné odeslat data k vyhodnocení na server. Přesnost se udává v desítkách metrů. [76], [40]

Triangulace pomocí GSM věží

Tato technika využívá síly GSM signálu a známé přesné polohy vysílacích věží pro zjištění lokace zařízení. GSM modul není typicky obsažen v některých typech zařízení, jako jsou např. levné tablety. Přesnost této techniky se obvykle udává ve stovkách metrů, je nenáročná na baterii a funguje všude, kde je pokrytí GPS signálu. Nejvyšší přesnosti dosahuje v oblastech s největší hustotou vysílacích věží. [40], [76], [49]

Přidělená IP adresa

V případě připojení k Internetu, je většině zařízení přidělena dynamická IP adresa, podle které lze identifikovat poskytovatele připojení a na základě vzorů rozdělování bloků IP adres přibližnou geografickou polohu [49].

1.7.2 Způsoby získávání polohy v Android

Oficiální dokumentace doporučuje použití knihovny *Google Play Services*, v opodstatněných případech (např. neexistence knihovny na cílovém zařízení nebo specifické podmínky používání aplikace) může být vhodnější využití *Android framework location API*.

Android framework location API

Získávání polohy se provádí za pomoci třídy `LocationManager`, která poskytuje přístup k aktuálním údajům o poloze a umožňuje aplikaci reagovat na její změnu. Pro získání polohy je nutné specifikovat způsob získávání polohy a k tomu slouží třída `ocationProvider`. Existují tři druhy poskytovatelů polohy — GPS, síťový a pasivní. GPS poskytovatel vrací data získaná GPS přijímačem, síťový využívá WiFi a GSM triangulaci a pasivní poskytuje polohu pouze v okamžiku, kdy o ní zažádá jiná aplikace, což se kladně odráží na výdrži baterie. Pro použití je nutné specifikovat obnovovací interval aktualizace polohy a minimální rozdílovou vzdálenost. Při praktickém využití je třeba přemýšlet o omezeních jednotlivých poskytovatelů a kombinovat více přístupů. Informace o aktualizované poloze obsahuje mj. zdroj dat, přesnost a čas získání, z čehož lze vycházet při rozhodování o přijetí vhodné hodnoty. [40], [54]

Google Play Services location API

Google Play Services je knihovna od společnosti Google, která je předinstalovaná na téměř všech zařízeních s obchodem Google Play. Knihovna umožňuje

Tabulka 1.6: Test nastavení priorit při získávání poloh pomocí Google Play Services

Priorita	Typický interval pro aktualizaci	Spotřeba baterie za hodinu (v %)	Přesnost
HIGH_ACCURACY	5 sekund	7.25%	∼10 metrů
BALANCED_POWER	20 sekund	0.6%	∼40 metrů
NO_POWER	N/A	zanedbatelná	∼1 míle

pohodlnou aktualizaci aplikací a poskytuje několik užitečných API, mezi které patří i služby zjišťování polohy.

Pro práci s lokační službou knihovny Google Play Services je nutné vytvořit pomocí třídy `GoogleApiClient.Builder` instanci objektu `GoogleApiClient` s parametrem nastaveným na `LocationServices.API`. K samotné poloze se přistupuje pomocí třídy `LocationServices.FusedLocationApi` a příslušných funkcí. Při dotazu na polohu se specifikuje pouze priorita a na bázi příslušných politik jsou vybráni vhodní poskytovatelé a aplikuje se algoritmus výběru. Je možné zvolit vysokou přesnost, nízkou či velmi nízkou spotřebu energie a v neposlední řadě vyvážení přesnosti a spotřeby baterie. Tabulka 1.6 byla uveřejněna na Google I/O konferenci v květnu roku 2013 a zobrazuje výsledky vícenásobného testování jednotlivých priorit na zařízení Galaxy Nexus [73]. [13]

Fused Location Provider pro přizpůsobování aktualizací polohy využívá tzv. rozpoznávání aktivity, což zde znamená snahu o zjištění, v jakém druhu pohybu se zařízení nachází. Výsledkem je pole obsahující objekty `DetectedActivity`, z nichž každý obsahuje důvěryhodnost, což je procentuální vyjádření pravděpodobnosti, že se zařízení nahází opravdu v takovém druhu pohybu. Aktivita může detekovat nepohyblivost zařízení, jízdu ve vozu, na kole, chůzi a běh. Dokáže také rozpoznat prudký vertikální pohyb zařízení. [73], [46]

Další poskytovanou službou je tzv. *Geofencing*, který umožňuje notifikovat aplikaci v případě, že zařízení vstoupí nebo opustí zvolenou oblast. Tento přístup je mnohem šetrnější k baterii zařízení. [73]

1.8 Datové struktury pro reprezentaci bodů zájmu

Bod zájmu je místo ve světě, které může být pro uživatele nějakým způsobem zajímavé. Musí obsahovat minimálně souřadnice a nějaký identifikátor pro uživatele, tím bývá nejčastěji název. Pro využití v aplikaci rozšířené reality se body zájmu mohou skládat ze dvou nebo tří souřadnic, záleží, zdali bude brána v potaz i nadmořská výška. Při volbě vhodné datové struktury pro uchovávání bodů zájmu je nutné zvážit vlastnosti uvedené v následujícím výčtu.

- Bodů zájmu může být mnoho
- Body zájmu jsou typicky načteny z persistentního úložiště pouze po startu aplikace resp. přenesení její aktivity do popředí
- Nejčastější operací je hledání všech bodů v určené oblasti nebo vyhledání určitého počtu nejbližších bodů v okolí stávající pozice
- Body mohou být reprezentovány dvěma nebo třemi souřadnicemi
- Není třeba často přidávat nové objekty
- Není třeba odebírat objekty

Mezi stromové datové struktury využívající se pro reprezentaci prostorových dat patří *KD-tree*, *Quad tree* a *R-tree* [42]. Podrobnější analýza těchto struktur je mimo rozsah této práce, ale zvídavým čtenářům, kteří by se o této problematice chtěli dozvědět více, se doporučuje přečíst desátou kapitolu knihy *Algorithms in a Nutshell* publikovanou nakladatelstvím O'Reilly Media.

1.9 Vykreslování 2D grafiky

Pro vykreslování vlastní dvourozměrné grafiky na obrazovku existují dvě následující možnosti.

1.9.1 Vytvoření View

Grafické prvky jsou součástí nějakého `View` objektu z `Layout` a o hierarchii vykreslování se stará operační systém. Toto je klasický přístup při definování vzhledu jednoduchých aplikací. [7]

1.9.2 Vykreslování přímo na Canvas

Grafické prvky se vykreslují přímo na `Canvas` a pro překreslení obrazovky se explicitně volá funkce `onDraw` s vytvořeným plátnem jako argumentem. Na `Canvas` lze kreslit funkcemi z rodiny `drawXXX`, která obsahuje funkce pro vykreslení obrázků, čar, cest, kruhů, obdélníků, oválů, bitmap a dalších tvarů a objektů. Tato možnost je vhodná pro aplikace, které se potřebují pravidelně a často překreslovat. Existuje více možností, jak vykreslovat přímo na plátno, které se dělí podle vlákna, ve kterém se volání provádí. [7]

Vytvoření vlastního View

Pro vykreslení na plátno v hlavním vláknu aplikace se používá přístup, který zahrnuje vytvoření vlastní třídy dědící z `View` a implementaci metody `onDraw`. Po následném zavolání funkce `invalidate` dojde k překreslení obrazovky. [7]

Využití SurfaceView

Třída `SurfaceView` umožňuje vykreslovat na `Canvas` z vedlejšího vlákna. K `SurfaceView` se nepřistupuje přímo, ale přes `SurfaceHandler` a je vhodné definovat vlákno starající se o překreslování přímo uvnitř třídy. [7]

1.10 Analýza požadavků

Zadáním je vytvořit knihovnu poskytující funkce polohově zaměřené rozšířené reality pro zařízení se systémem Android verze 4.0.3 a vyšší. Hlavní cílovou skupinou jsou uživatelé mobilních telefonů, aplikace by však měla rozumně fungovat i na tabletech. Architektura knihovny musí být navržena tak, aby ji bylo možné po přidání bodů zájmu jednoduše použít bez nutnosti větších zásahů a úprav. Zároveň musí splňovat níže zmíněné nároky na rozšiřitelnost.

- Možnost implementace vlastního zdroje dat s body zájmu
- Nastavení parametrů ovlivňujících poměr přesnosti a šetrnosti k baterii zařízení (intervaly senzorů, GPS)
- Možnost použití vlastního zdroje informací o poloze
- Možnost implementace vlastního vzhledu a chování viditelných bodů zájmu na obrazovce
- Možnost vložení vlastního filtru a nastavení počtu naměřených hodnot v historii
- Možnost omezení počtu zobrazitelných bodů

Při tvorbě knihovny by měl být kladen důraz na výkon a dosažení co největší přesnosti zobrazení pomocí úprav senzory naměřených dat, protože se předpokládá i používání v zařízeních se slabší hardwarovou výbavou. Používání aplikace by pro uživatele mělo být intuitivní a poskytovat snadnou orientaci v zobrazovaných informacích. Aplikace bude navržena tak, aby nedocházelo k zamrzání a zasekávání obrazu. Při návrhu vzhledu je vhodné se inspirovat některými prvky již existujících řešení.

Knihovna by neměla zabírat mnoho místa na disku zařízení.

Kapitola 2

Návrh

2.1 Vývojové nástroje a parametry

Pro vývoj knihovny byl zvolen v dnešní době nejrozšířenější přístup programování aplikací pro OS Android v podobě kombinace programovacího jazyku *Java* a *Android SDK*. V analýze nebyly zjištěny žádné důvody, které by naznačovaly nevhodnost použití této kombinace nástrojů. K usnadnění vývoje se autor rozhodl psát zdrojové kódy ve vývojovém prostředí *Android Studio*, a to z důvodů osobní zkušenosti s tímto nástrojem. Android Studio ve výchozím nastavení používá pro sestavování projektů automatizační nástroj *Gradle*. Ten funguje spolehlivě a je vhodný i pro vyvíjenou knihovnu, proto na tomto nastavení není třeba nic měnit.

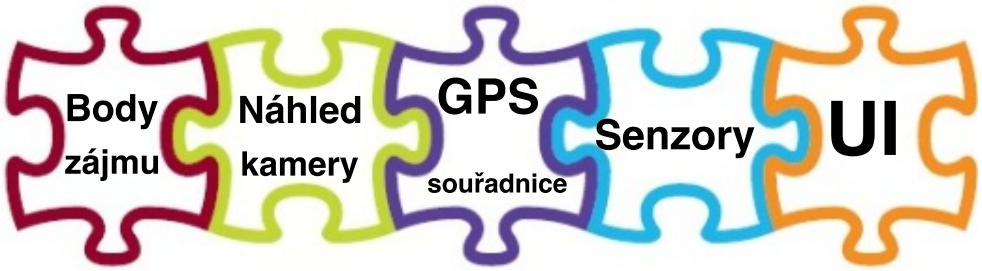
Návrh knihovny respektuje požadavky uvedené v sekci 1.10, a proto bude minimální podporovanou verzí operačního systému Android 4.0.3. Sestavování knihovny bude probíhat oproti poslední verzi vývojového prostředí, tedy Android SDK 23.

2.2 Operace jednotlivých částí knihovny AR

Aplikace rozšířené reality se skládá z několika elementárních součástí. Cílem této sekce je shrnout tyto části a pro každou z nich vyselektovat operace důležité pro návrh kostry vyvíjené knihovny. Díky tomuto shrnutí bude možné v další sekci stanovit závislosti a následně rozdělit aplikaci na více asynchronních segmentů, což je výhodné pro maximalizaci výkonu. Obrázek 2.1 vizualizuje, jak do sebe musí jednotlivé součásti aplikace rozšířené reality zapadat.

2.2.1 Body zájmu

Informace, které mají být zobrazeny, jsou základem aplikace rozšířené reality. Data jsou načtena z persistentního úložiště, zůstávají v paměti po celý běh aplikace a při ukončení jsou systémem automaticky odstraněna z paměti



Obrázek 2.1: Součásti knihovny rozšířené reality

(v případě přenesení aplikace do pozadí a její opětovné aktivace dojde k jejich znovunačtení).

Operace s body zájmu

- Načtení bodů zájmu do paměti aplikace

2.2.2 Pozice

Požadavkem na knihovnu je možnost poskytnutí vlastního zdroje lokačních dat, zároveň má však obsahovat funkční vestavěný mechanizmus pro zjištění pozice. Z těchto důvodů se návrh kostry knihovny nezabývá konkrétním způsobem získávání dat o lokaci, jeho inicializací apod. Pro tento návrh je podstatná jediná operace a tou je získání polohy.

Operace zjišťování aktuální pozice

- Schopnost poskytnout aktuální polohu

2.2.3 Senzory

Práce se senzory bude neměnnou součástí jádra knihovny a v rámci aplikace používající knihovnu bude možné konfigurovat pouze některá nastavení. Přístup k senzorům skrz Android SDK neumožňuje přímý dotaz na hodnotu [51], intervaly aktualizace mohou být navrženy, ale přesný okamžik určuje systém. Z tohoto důvodu je třeba uvažovat jak registraci odposlechu dat, tak moment, kdy bude dostupná alespoň jedna hodnota z každého potřebného senzoru. Pro uchování získaných dat a jejich historie bude třeba inicializovat vhodné datové struktury.

Operace spojené s získáváním dat ze senzorů

- Inicializace struktur pro uchovávání dat
- Registrace odposlouchávání hodnot
- Dostupnost hodnot ze všech potřebných senzorů

2.2.4 Náhled fotoaparátu

Náhled fotoaparátu je možné spustit okamžitě po přiřazení příslušného prvku UI, proto není nutné tyto operace rozdělovat. Z hlediska tvorby kostry knihovny je relevantní pouze okamžik spuštění náhledu.

Operace náhledu fotoaparátu

- Spuštění a zobrazení náhledu

2.2.5 Výpočty prováděné knihovnou

Pro správnou funkčnost knihovny je nutné nějakým způsobem zvolit body zájmu k zobrazení, spočítat aktuální orientaci zařízení, spočítat počáteční směr ke zvoleným bodům zájmu z pozice pozorovatele a na základě orientace zařízení zjistit jejich viditelnost a případnou pozici na obrazovce.

Výpočty

- Výběr bodů k zobrazení
- Přepočet směru k bodům
- Výpočet orientace zařízení
- Zjištění viditelnosti a výpočet pozice na obrazovce
- Výpočet pozice na radaru (volitelné)
- Zvolení vybraných bodů, které zobrazí více informací (volitelné)

2.2.6 Vykreslení informace na obrazovku

Vykreslování informací o bodech zájmu probíhá ve dvou fázích. Jedná se o vykreslení jednotlivých viditelných bodů ve směru kamery a překreslení radaru (pouze v případě jeho využití). Z hlediska této fáze návrhu lze obě tyto operace spojit do jedné.

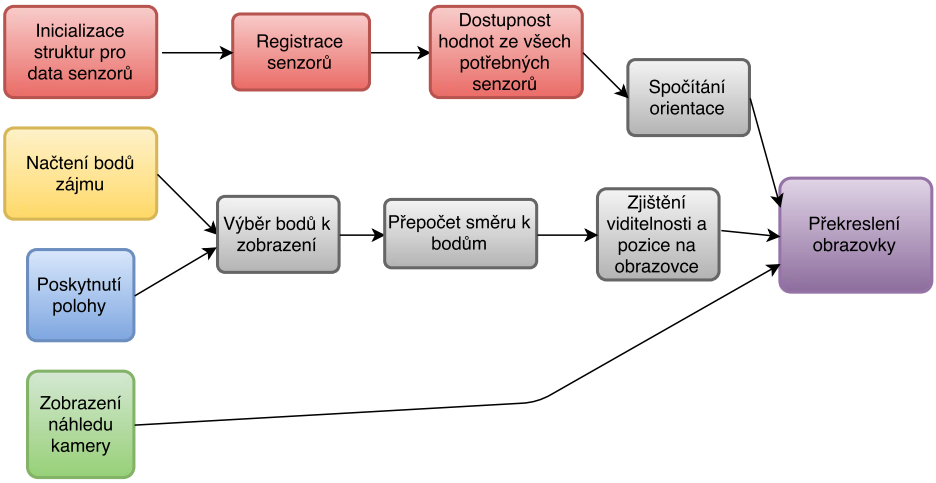
Operace s UI

- Překreslení obrazovky

2.3 Rozložení operací do vláken a synchronizace

2.3.1 Operace a jejich závislost

Diagram na obrázku 2.2 znázorňuje závislost operací zmíněných v sekci 2.2 a vychází se z něj pří návrhu rozložení komponent do vláken a určení pořadí počátečního volání operací popř. kontrol inicializace hodnot nezbytných pro start právě prováděné operace.



Obrázek 2.2: Závislost operací v knihovně rozšířené reality

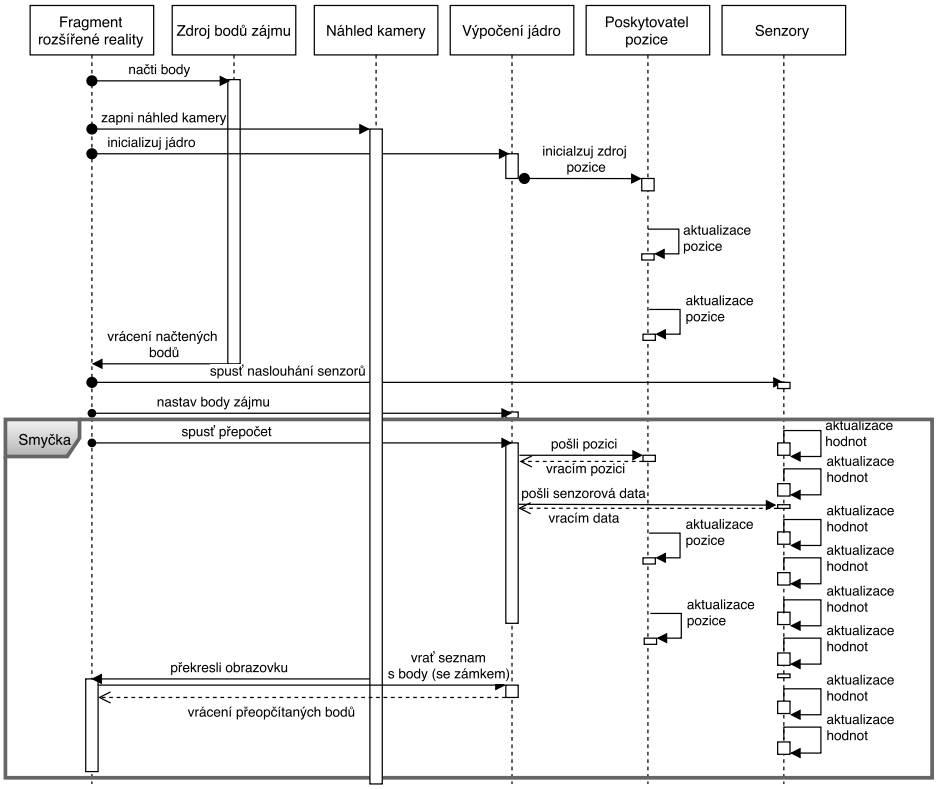
Tabulka 2.1: Rozdělení na sady operací

Název činnosti	Opakování	Hlavní vlákno	Závisí na počtu bodů	Odhad doby trvání
Načtení bodů zájmu	jednorázové	ne	ano	záleží na zdroji dat
Zobrazování náhledu kamery	kontinuální	ano/ne[28]	ne	nízká
Sběr dat ze senzorů	kontinuální	ne	ne	nízká
Získávání pozice	kontinuální	ne	ne	nízká
Veškeré výpočty pozic bodů na obrazovce (radaru, prioritních bodů)	jednorázové, opakující se	ne	ano	vysoká
Překreslení bodů a radaru na obrazovce	jednorázové, opakující se	ano	ano	střední

2.3.2 Definice činností

Přehledný souhrn činností nutných pro správný chod knihovny polohově zaměřené rozšířené reality včetně jejich relevantních vlastností znázorňuje tabulka 2.1, která vychází z diagramu 2.2. Knihovna byla rozdělena podle náročnosti, závislosti na vstupních bodech a druhu vykonávané práce na šest sad operací, které lze provádět nezávisle v různých vláknech.

Jen jedna sada operací manipuluje s uživatelským rozhraním aplikace (je navržena pro běh v hlavním vláknu), rozdělení tedy neporušuje zákaz manipulace s UI mimo hlavní vlákno, která je uvedena v podsekci 1.6.3. Také je



Obrázek 2.3: Sekvenční diagram navrhované komunikace mezi vlákny knihovny

v souladu s doporučením v téže podsekci a sada operací pro běh v UI vláknu aplikace je minimalizována pouze na ty činnosti, které jsou spojené s vykreslováním obsahu. Tímto je zaručena odezva a plynulost aplikace i při velkém množství zobrazovaných bodů.

2.3.3 Rozložení operací do vláken

Zjednodušený sekvenční diagram na obrázku 2.3 slouží k představě, jak by podle návrhu mělo vypadat rozložení aplikace do jednotlivých vláken a vzájemná komunikace mezi nimi. Pořadí prováděných operací respektuje závislosti zobrazené na obrázku 2.2. Z předávaných informací se vychází při návrhu synchronizačních postupů.

Načítání bodů zájmu

Načtení dat s body zájmu je jednorázová operace trvající delší dobu. Knihovna nemůže bez těchto dat dále pracovat — je tedy žádoucí upozornit uživatele a zobrazit průběh načítání. Jak vyplývá z analýzy v sekci 1.6, vhodným kon-

struktem pro tuto operaci je `AsyncTask`, který provádí jednorázovou blokující operaci na pozadí a zároveň skrze hlavní vlákno informuje uživatele o průběhu. S výsledkem je třeba pracovat, proto se implementuje jednoduché rozhraní, jehož metoda bude volána po dokončení operace na pozadí. Parametrem této metody bude nově načtená sada bodů zájmu. Tuto metodu bude možné přetížit a dodatečně filtrovat data.

Zobrazení náhledu kamery

Zobrazení náhledu kamery je kontinuální operace, která připravuje obraz na pozadí a při vykreslování nového snímku jej poskytuje hlavnímu vláknu. Probíhá nezávisle na všech ostatních operacích knihovny a jediným výstupem je obraz, proto není nutné provádět žádnou vlastní komunikaci s jinými vlákny.

V některých případech může být výhodné využít metodu `onFrameChanged` volanou při každém překreslení snímku i k vykreslení generovaných informací o bodech zájmu.

Sbírání dat ze senzorů

Sbírání dat ze senzorů probíhá v pravidelných intervalech na základě rozhodnutí systému a navržené hodnoty. Jediná akce, která je provedena po získání nové hodnoty, je její uložení do historie naměřených hodnot. V jiném vláknu je možné s touto historií pracovat, a proto je nutná jednoduchá synchronizace. Použití bloku `synchronize` v metodě `onSensorChanged` může znamenat porušení doporučení z podsekce 1.4.6, v tomto konkrétním případě však bude k synchronizovanému objektu přistupovat pouze výpočetní vlákno, které hodnotu pouze zkopíruje popř. provede nenáročné filtrování. Tento přístup byl navržen pro zjednodušení orientace ve zdrojovém kódu. Pokud bude v souvislosti s ním docházet k chybám při testování, bude nutné použít sofistikovanější řešení.

Poskytování údajů o poloze

Zadání požaduje, aby bylo možné rozšířit knihovnu o vlastního poskytovatele údajů o poloze zařízení. Jediným požadavkem na poskytovatele polohy je implementace příslušného rozhraní s metodou pro zjištění aktuální polohy. Při tvorbě vlastního poskytovatele je třeba se vyvarovat změny objektu s informacemi o poloze po jeho poskytnutí skrze rozhraní. Jednoduchým řešením je při aktualizaci polohy vytvářet vždy nový objekt.

Vykreslování UI

Překreslování uživatelského rozhraní aplikace musí probíhat v hlavním vlákně. Pro vykreslení je nezbytný seznam bodů[29] ve zvoleném okolí s předpočítanými hodnotami závislými na aktuální poloze a orientaci zařízení. S tímto seznamem však v době překreslování nesmí být manipulováno v jiném vláknu, a musí tedy být uzamčen. Výpočetní vlákno ale potřebuje aktualizovat hodnoty pro další vykreslení. Jednoduchý přístup za pomoci bloku `synchronize` zde není možný, protože by docházelo k blokování hlavního vlákna aplikace a byl by zdegradován vícevláknový model. Řešením je existence dvou seznamů[30] se zámky. Hlavní vlákno se pokusí zamknout první seznam a pokud se to nepodaří, bez prodlevy si zamkne druhý seznam, který v této situaci musí být volný, protože k seznamům se přistupuje pouze ze dvou zmíněných vláken. Po ukončení překreslování je zámek uvolněn. Stejný proces probíhá při přístupu k seznamům z výpočetního vlákna. Zmíněné řešení si klade za cíl předcházet zamrznutí aplikace a souběžně umožnit přepočítávání hodnot na jejím pozadí.

Provádění výpočtů

Veškeré výpočty nutné pro správné zobrazení informací na obrazovce budou probíhat v jednom vláknu, a to převážně kvůli své sekvenční povaze. Náročnost výpočtů se odvíjí od počtu bodů zájmu a složitosti algoritmů použitých k přípravě dat pro vykreslení.

2.4 Návrh jednotlivých částí knihovny

2.4.1 Základ

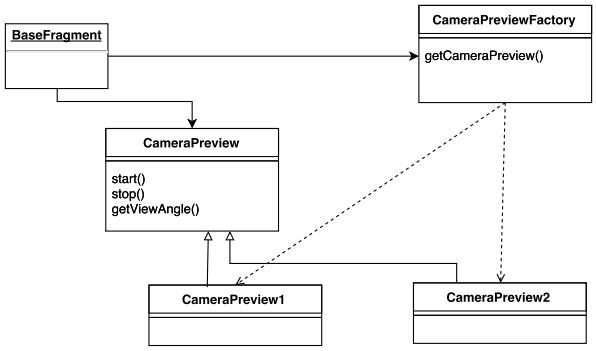
Primární scénář zakomponování vyvíjené knihovny do aplikace je velmi jednoduchý a zahrnuje pouze vložení příslušné aktivity nebo fragmentu rozšířené reality do kontextu aplikace. Autor se při návrhu knihovny přiklonil k použití fragmentu namísto aktivity, jelikož fragmenty poskytují rozsáhlejší možnosti vytváření dynamického uživatelského rozhraní [77]. Základním kamenem knihovny tedy bude fragment s abstraktními metodami pro vykreslení radaru a bodů zájmu, který bude zároveň implementovat rozhraní pro získávání pozice zařízení. Tento fragment bude dále v práci nazýván *základní fragment*.

2.4.2 Náhled kamery

Návrh implementace náhledu kamery musí respektovat existenci nového přístupu k fotoaparátu uvedeného v rámci Android API 21. Zároveň však, z dů-

[29] Pro optimalizační účely mohou existovat různé podseznamy s referencemi na stejné objekty (např. seznam viditelných bodů)

[30] Duplikace seznamů má za následek zvýšení paměťové náročnosti, které je pro většinu zařízení považováno za přijatelné.



Obrázek 2.4: Použití návrhového vzoru továrna v náhledu kamery

Tabulka 2.2: Nastavení parametrů kamery

Parametr	Hodnota
Zaostřovací mód	FOCUS_MODE_CONTINUOUS_VIDEO
Nastavení přisvětlovací diody	FLASH_MODE_OFF

vodu požadavku na minimální podporovanou verzi 15, je třeba do návrhu zakomponovat přístup původní. Autor shledal při návrhu vhodné odstínit tyto rozdíly společnou abstraktní nadtřídou a použitím návrhového vzoru továrna, popsaného v [36]. Návrh tříd ke zmíněnému konstruktu zachycuje obrázek 2.4.

Nastavení parametrů

V analytické části 1.5.1 bylo popsáno několik parametrů, které je vhodné modifikovat pro použití náhledu v souvislosti se zadáním. Zvolené implicitní nastavení těchto parametrů je uvedeno v tabulce 2.2.

Nejpravděpodobnější využití knihovny se předpokládá v oblasti turismu a stravování, konkrétně pro vyhledání památek či restaurací a hotelů v okolí uživatele. Z tohoto důvodu byl shledán jakýkoliv druh použití přisvětlovací diody zařízení jako nevhodný a zbytečně rozptylující uživatele.

Mód zaostřování byl zvolen na základě doporučení uvedeného v dokumentaci a provedeného testování, ve kterém se prokázalo, že mód průběžného zaostřování pro video je pro aplikaci AR opravdu nejvhodnější.

Autor se rozhodl neovlivňovat počet zobrazených rámců za sekundu ani žádné další parametry. V případě výskytu chyb nebo specifických požadavků uživatelů knihovny pro konkrétní využití je v budoucnu možné doplnit implementaci a možnost nastavení více parametrů.

Vhodný rozměru náhledu

Každá kamera podporuje určité velikosti náhledu. Tyto velikosti lze získat voláním metody `getSupportedPreviewSizes` nacházející se v třídě reprezen-

(a) Vertikální náhled (b) Horizontální náhled

Obrázek 2.5: Chybně implementovaný náhled

tující parametry fotoaparátu. Při programování je třeba zvolit tu nejvhodnější v závislosti na velikosti oblasti, ve které bude náhled zobrazován. Volba vhodných rozměrů se odvíjí od rozdílu poměru stran požadované a podporované velikosti. Při implementaci aplikace rozšířené reality je možné povolit určitou odchylku poměru, neboť drobné roztažení některých objektů má zanedbatelný vliv na výslednou funkčnost aplikace. Obrázek 2.5 zobrazuje chybně volenou velikost náhledu a její dopad v podobě nereálného pohledu na svět.

Úhel záběru

Pro výpočet bodů k zobrazení je nutné spolehlivě zjistit úhel záběru fotoaparátu, který se může u různých typů lišit. K této hodnotě bude přistupovat výpočetní vlákno a není vyloučeno, že i některé další. Proto bude abstraktní třída pro přístup k fotoaparátu obsahovat statickou metodu zjišťující právě úhel záběru fotoaparátu.

2.4.3 Načítání bodů zájmu

Datová struktura pro uchovávání bodů zájmu

Z požadavků na strukturu v analýze 1.10 byl, na základě porovnání vlastnosti struktur podle [42] a s přihlédnutím na možnost snadného přidání případné třetí dimenze (nadmořská výška), vybrán KD-tree. KD-tree umožní rychlé vyhledání okolních bodů zájmu při změně polohy zařízení i v případě rozsáhlých zdrojových dat.

Způsoby načítání bodů zájmu

Mechanismus pro načítání bodů není nezbytný a bylo by možné podmínit použití vyvíjené knihovny poskytnutím seznamu již načtených bodů. Autor však považuje za výhodné, aby byl proces načítání bodů obsažen přímo v knihovně, díky čemuž je možné provádět načítání paralelně s inicializací ostatních komponent a zobrazit uživateli informační dialog společně s náhledem kamery. Tento přístup také podpoří modularitu knihovny, protože bude možné pohodlně zahrnout uživatelsky vytvořené třídy pro načítání ze specifických zdrojů do nových verzí. V některých případech, kdy byla data načtena ještě před spuštěním rozšířené reality, je použití zmíněného přístupu zbytečné, a proto bude knihovna poskytovat alternativní přístup v podobě možnosti přímého předání seznamu bodů.

Návrh počítá s existencí abstraktní třídy `PoiLoader`, ze které bude možné dědit a implementací metody `loadPois` provést operaci načtení dat na pozadí. Argumentem této metody bude třída dědící z `PoiLoaderArguments` s parametry potřebnými pro načtení.

Ve výchozí implementaci první verze knihovny se počítá s existencí pouze jednoho mechanismu pro načítání, jehož zdrojem bude soubor ve formátu *Garmin CSV (Comma Separated Values) File Format*, který obsahuje čtyři sloupce oddělené čárkou s významy: poledník, rovnoběžka, název, popis a desetinnými čísly oddělenými tečkou [37].

2.4.4 Senzory

Návrh senzorové části knihovny počítá s použitím dvou základních senzorů pro zjištění orientace zařízení, kterými jsou akcelerometr a magnetometr. Zároveň je nutné uvažovat případné rozšíření na další senzory pro zlepšení přesnosti naměřených hodnot.

Naslouchání změnám

Proces naslouchání změn bude probíhat v jádru aplikace[31] a to standardním způsobem popsaným v analýze 1.4.4. Uživatel knihovny bude mít možnost

[31] Jádrem aplikace je myšlena třída starající se o sběr dat a spouštějící výpočty

nastavit hodnotu navrhující interval pro získávání hodnot.

Ukládání hodnot

Správné zobrazení rozšířené reality je existenčně závislé na kvalitě naměřených dat, a proto může být výhodné mezi překreslováním jednotlivých snímků nasbírat takových dat co nejvíce. K tomuto účelu slouží třída `SensorHistory`, která bude uchovávat historii naměřených senzorových hodnot pomocí kruhového pole o maximální délce specifikované v nastavení knihovny. Nastavení bude také obsahovat přepínač umožňující vynulování[32] již vyzvednutých hodnot z historie. Při implementaci třídy `SensorHistory` je nutné dbát na výkon a omezit vytváření nových objektů, protože hodnoty mohou být aktualizovány ve velmi krátkých intervalech.

Data senzorů se zpravidla skládají z pole o třech hodnotách. Nestačí uložit odkaz na pole, ale je třeba tyto hodnoty zkopírovat, aby nedošlo k jejich přepsání (systém může stejné objekty používat znovu [55]).

Zpracování hodnot

Data získaná mezi vykreslením jednotlivých snímků je nutné nějakým způsobem zpracovat a získat z nich výslednou hodnotu, která se použije ve výpočtech. K plynulému přechodu může sloužit i znalost hodnoty použité v předchozím kroku. Uživatel knihovny bude mít možnost implementovat vlastní přístup zpracování dat zděděním od abstraktní třídy `Filter` s metodou `filter` nebo bude moci využít vestavěných algoritmů. Návrh počítá s počáteční existencí jednoho přístupu, kterým bude zprůměrování historie v kombinaci s následným provedením *low-pass* filtru této hodnoty a hodnoty použité v předcházejícím překreslování obrazovky.

2.4.5 Aktuální pozice

Poskytování pozice zařízení patří mezi volitelně rozšiřitelné části knihovny. Její jedinou povinnou funkcí je poskytování aktuální pozice v reálném čase, a k tomu se při návrhu struktury nejvíce hodilo použití rozhraní. Poskytovatelem lokace může být jakákoliv třída, která implementuje toto rozhraní nazvané `PositionProviderInterface` obsahující metodu `getActualLocation` vracející pozici. Důležitou poznámkou k implementaci vycházející z vícevláknové architektury je zákaz manipulace s vráceným objektem. Při změně lokace tedy je nutné vytvořit nový objekt.

[32] Vynulování proběhne nastavením nultého prvku senzorových dat na zvolenou hodnotu, která patří mimo interval hodnot senzorů, aby nedocházelo ke zbytečnému vytváření nových objektů

Vestavěný poskytovatel lokace

Vyvíjená knihovna má obsahovat vlastního poskytovatele aktuální pozice a za tímto účelem byl zvolen přístup, který je v současnosti doporučován v oficiální dokumentaci [13] a novějších publikacích [75], [51]. Tímto přístupem je využití rozhraní pro práci s lokací v knihovně *Google Play Services*. Návody doporučují zakomponovat aktualizaci pozic z dostupných zdrojů do životního cyklu fragmentu a toto doporučení se autor rozhodl respektovat i pro vyvíjenou knihovnu. Pro zachování rozšiřitelnosti bude vytvořen abstraktní fragment, který bude potomkem základního fragmentu a definuje obsah metody `getActualLocation`.

2.4.6 Výpočty

Veškeré z výpočetních operací definovaných v sekci 2.2.5 budou implementovány v jedné metodě jádra aplikace. Jak bylo navrženo v 2.3.3, tato metoda bude spuštěna na pozadí ve vláknu k tomu uzpůsobeném. Zmíněné výpočty lze rozdělit do dvou kategorií. První z nich obsahuje výpočty prováděné pouze při změně polohy zařízení, zatímco operace z druhé kategorie proběhnou při každém volání funkce.

Výběr bodů k zobrazení

Tato operace zahrnuje selekci vybraných bodů v okolí uživatele. Tyto body budou zobrazeny na radaru (pokud se použije) a jejich aktuálně viditelná podmnožina bude zobrazena na displeji. Vybrané body slouží k vytvoření objektů typu `PoiAnnotation`, které obsahují doplňující proměnné na ukládání výsledků jednotlivých výpočtů. Pro výběr bodů byly uvažovány níže uvedené strategie[33].

- Počet nejbližších bodů
- Okolí zadané měrnou jednotkou
- Kombinace předchozích

Výhodou zvolení fixního počtu nejbližších bodů je zejména schopnost mít pod kontrolou zatížení aplikace, naopak nevýhodou je zobrazení potencionálně nechtěných bodů. Tuto nevýhodu lze mírně eliminovat viditelným odlišením bodů podle vzdáleností. Další drobnou nevýhodou oproti omezení vzdálenosti je naopak absence některých bodů, které by bylo vhodné zobrazit. Lze použít i algoritmy založené na kombinaci zmíněných přístupů, které mohou měnit své chování v závislosti na dalších faktorech, jako např. počtu bodů ve stejném směru nebo nadmořské výšce pozorovatele (uvažujeme-li její zahrnutí do knihovny). Pro účely této práce byla zvolena strategie omezení nejbližšího počtu bodů z důvodu snadné implementace a zajištění plynulosti aplikace.

[33] Strategie výběru závisí pouze na poloze bodů a nezabývá se filtrováním podle jiných vlastností

Obrázek 2.6: Rozdíl mezi počátečním a koncovým směrem k bodu [31]

Přepočet vzdálenosti a směru k bodům

Směr k určenému bodu (angl. *bearing*) nejkratší cestou po zemském povrchu není díky tvaru zeměkoule konstantní, ale v průběhu cesty se mění [28]. Rozdíl mezi počátečními a koncovými směry graficky znázorňuje obrázek 2.6. Vyvíjená knihovna bude v každém okamžiku dosazovat do výpočtů pouze počáteční směr.

Android SDK poskytuje ve třídě `Location` funkci `distanceBetween` pro výpočet přibližné vzdálenosti mezi dvěma body, která volitelně vrací i počáteční a koncové směry nejkratší cesty mezi těmito body. Autor se rozhodl využít zmíněnou funkci a abstrahovat tak od nízkoúrovňových výpočtů.

Výpočet orientace zařízení

Pro zjištění orientace zařízení poskytuje Android SDK statickou funkci `getOrientation` třídy `SensorManager`, která nahradila zastaralý přístup pomocí virtuálního senzoru označeného konstantou `Sensor.TYPE_ORIENTATION`. Funkce pro zjištění orientace potřebuje poskytnout rotační matici, kterou lze získat voláním statické funkce `getRotationMatrix` umístěné ve stejné třídě. Pro získání rotační matice jsou potřebné naměřené hodnoty z akcelerometru a magnetometru. Získanou rotační matici není možné použít přímo, neboť reprezentuje hodnoty ve výchozím koordinačním systému (viz obrázek 1.13), který je třeba transformovat tak, aby bylo možné zjistit odchylku od severního směru v ose fotoaparátu. Pro transformaci koordinačního systému rotační matice se používá statická funkce `remapCoordinateSystem`, kde osy jsou reprezentovány konstantami začínajícími na `AXIS_`. Zmíněná funkce i konstanty se opět nachází ve třídě `SensorManager` a podrobný popis je obsahem dokumentace Android SDK [20].

Zjištění viditelnosti a pozice na obrazovce

Za účelem zpřehlednění a drobného zrychlení aplikace je vhodné vytvořit seznam obsahující podmnožinu anotací viditelných bodů v závislosti na směru a

aktuální orientaci. Bod je shledán viditelným, pokud je minimální rozdíl mezi směrem fotoaparátu a počátečním směrem k vybranému bodu menší než polovinu zorného úhlu čočky fotoaparátu. Pro zjednodušení výpočtu jsou úhly normalizovány za pomoci vzorce $\alpha = (\alpha + 2\pi) \bmod 2\pi$, kde α je bod určený k normalizaci.

Pro body k zobrazení ve výsledném seznamu se poté provede přepočítání pozice na obrazovce. Vzorec pro výpočet horizontální souřadnice bodu zájmu na obrazovce má podobu

$$x = \frac{width}{2} + \left(sign \cdot \frac{\delta \cdot width}{\alpha} \right)$$

kde $width$ je šířka zobrazovacího prostoru v pixelech, α reprezentuje směr kamery a δ vyjadřuje minimální rozdíl mezi směrem kamery a počátečním směrem k bodu zájmu. Proměnná $sign$ nabývá hodnoty 1 v případě, že je minimální rozdíl mezi směrem kamery a počátečním směrem k bodu zájmu orientován po směru hodinových ručiček a hodnoty -1 v případě, že tomu tak není.

Vertikální souřadnice bude pro jednoduchost a přehlednost zadána fixně pro všechny body.

Výpočet pozice na radaru

Radar je užitečný pro orientaci pozorovatele v prostoru, mohou však existovat scénáře, ve kterých není jeho použití žádoucí např. z důvodu zmenšení volného prostoru na obrazovce. Při použití radaru a současném zobrazování dodatečných informací o vybraných bodech je vhodné, právě vybrané body na radaru zvýraznit, aby mohl uživatel lépe odhadnout rozdíly vzdáleností. Efektivní výpočet pozice na radaru se odvíjí od vzdálenosti pozorovatele a nejvzdálenějšího bodu ve vybraném okolí. Nákres na obrázku 2.7 popisuje význam jednotlivých proměnných vyskytujících se ve výpočtu a pravoúhlý trojúhelník, za pomoci kterého lze dopočítat souřadnice. Nejprve je vzdálenost k bodu přepočítána na pixely pomocí vzorce

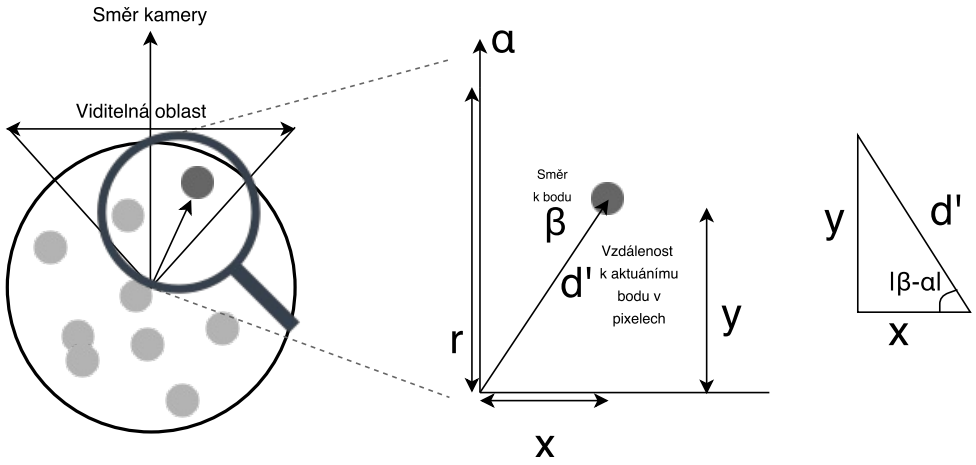
$$d' = \frac{d \cdot r}{dMax}$$

kde d je vzdálenost k bodu zájmu, r poloměr radaru a $dMax$ vzdálenost od nejvzdálenějšího bodu zájmu, který bude vykreslen na radaru. Poté můžeme snadno pomocí goniometrických funkcí dopočítat souřadnice x a y. Vzorce pro jejich výpočet jsou

$$x = \sin(\gamma \bmod \pi) \cdot d'$$

$$y = \sqrt{d'^2 - x^2}$$

kde γ značí úhel mezi mezi směrem kamery a počátečním směrem k bodu zájmu orientovaný proti směru hodinových ručiček.



Obrázek 2.7: Znázornění výpočtu pozice bodu zájmu na radaru

Tyto hodnoty jsou však pravdivé jen pro první kvadrant. Pokud se bod nachází v jiném kvadrantu, což lze zjistit z velikosti úhlu γ, je nutné zobrazený trojúhelník příslušně převrátit a do souřadnic dosadit správné hodnoty.

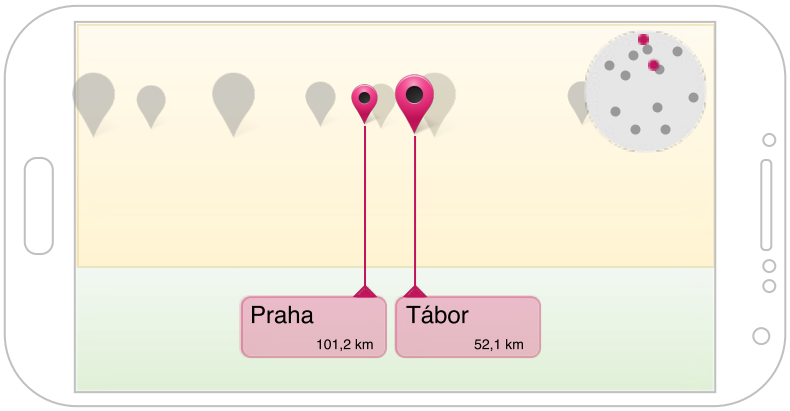
Zvolení vybraných bodů, které zobrazí více informací (volitelné)

Zobrazení dodatečných informací u bodů, které byly nějakým způsobem vyhodnoceny jako důležité, může mít kladný dopad na spokojenost uživatele. Dodatečné informace mohou obsahovat popis, vzdálenost, fotografii nebo nějakou informaci specifickou pro konkrétní aplikaci (např. vyhledávač restaurací pro milovníky piva může zobrazit druhy točených piv ve vybraných zařízeních).

Strategie pro výběr by neměla žádné body vynechávat a jednou za čas zobrazit i ty s nižší prioritou. Zároveň je třeba zajistit, aby ke změnám těchto bodů nedocházelo příliš často. Zajímavým vylepšením by mohlo být umožnění uživateli aplikace ovlivnit výběr (např. přidání maximální priority po kliknutí na zvolený bod). Návrh strategie pro vyvíjenou knihovnu tuto inovaci zatím nezahrnuje, výběr však závisí na několika faktorech. Těmito faktory jsou:

- Vzdálenost od středu obrazovky
- Vzdálenost od pozorovatele
- Prioritizace dlouho nevybraných bodů
- Náhoda[34]
- Dočasné přidání priority právě zobrazeným bodům

[34]Náhoda je chápána ve smyslu použití generátoru pseudonáhodných čísel.



Obrázek 2.8: Návrh vzhledu uživatelského rozhraní

2.4.7 Uživatelské rozhraní

Správný návrh uživatelského rozhraní je klíčový pro úspěch výsledného produktu u koncových uživatelů. Při jeho tvorbě se autor inspiroval pozitivně hodnocenými vlastnostmi knihoven zmíněných v sekci 1.3 a pokusil se vyvarovat jejich zjištěným nedostatkům.

Obrázek 2.8 zobrazuje návrh vzhledu uživatelského rozhraní výchozího nastavení knihovny. Návrh lze rozdělit do tří barevně oddělených oblastí.

- **Radar** V pravé horní části se nachází jednoduchý radar, který pomáhá uživateli orientovat se v prostoru. Radar by měl zvýrazňovat vybrané body a v sofistikovanějším provedení může obsahovat pomocnou mřížku a pozici oproti hlavním světovým stranám.

- **Zobrazení značek ve směru viditelných bodů** Žlutá oblast zobrazuje značky viditelných bodů. Návrh počítá s pětiúrovňovým škálováním ikon podle vzdálenosti bodů od pozorovatele. Jak již bylo zmíněno, v rámci této práce budou všechny značky ve stejné výšce. Lehce problémová se jeví kolize s radarem, kterou lze v případě negativních ohlasů uživatelů eliminovat zmenšením oblasti pro vykreslování značek nebo snížením vertikální pozice značek.

- **Doplňující informace o vybraných bodech** Zelená oblast ve spodní části displeje slouží k zobrazení dodatečných informací o vybraných bodech. Návrh prozatím počítá s mřížkou o jedné řádce a dvěma až třemi sloupci, ale v dalších verzích knihovny je možné tento prostor rozšířit.

2.5 Dokumentace

Součástí výsledné práce má být dle zadání i podrobná dokumentace. Tato sekce stručně shrnuje několik základních pravidel pro psaní dokumentace a popisuje prostředky a techniky použité k dokumentování vyvíjené knihovny.

2.5.1 Obecná doporučení

V následujícím seznamu je uvedeno několik základních vlastností, které by měla splňovat dokumentace k vyvíjenému programu.

- **Stručnost a věcnost** Dokumentace by měla být srozumitelná, k věci a měla by obsahovat minimum zbytečných informací [57].
- **Aktuálnost a pravdivost** Popisovaná fakta a postupy by měly být řádně vyzkoušeny a revidovány při změnách ve zdrojovém kódu [57].
- **Úplnost** Dokumentace by měla popisovat všechny části knihovny [63].
- **Psána přímo v kódu** Je vhodné psát dokumentaci přímo v kódu, ten by se v ideálním případě měl dokumentovat sám volbou srozumitelných názvů funkcí a proměnných [63].

2.5.2 Generovaná část dokumentace

Jedním z dokumentačních výstupů bude hypertextový dokument obsahující vazby mezi třídami, metodami a proměnnými knihovny. Tento dokument bude vygenerován pomocí programu Doxygen[35] přímo ze zdrojového kódu knihovny za použití komentářů v Javadoc notaci[36]. Dokument bude obsažen na CD (kompaktním disku) přiloženému k práci. Pro ilustraci vzhledu generovaného dokumentu čtenáři slouží obrázek 2.9.

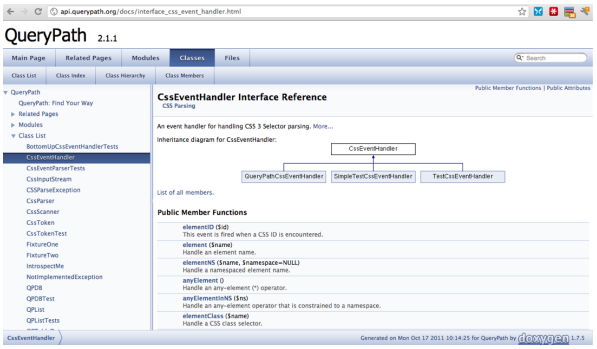
2.5.3 Manuál pro vývojáře

Důležitou součástí dokumentace je manuál pro vývojáře. Manuál k vyvíjené knihovně bude mít formu DokuWiki dokumentu[37] a jeho obsahem bude úvod a několik scénářů použití knihovny s popisem a praktickými příklady. Tento manuál bude součástí tištěné práce a přiloženého CD.

[35] `www.doxygen.org/`

[36] Specializovaný formát komentářů zdrojového kódu předurčený ke strojovému zpracování. Více informací lze nalézt např. na `http://download.java.net/jdk7u2/docs/technotes/tools/solaris/javadoc.html`

[37] `https://www.dokuwiki.org`



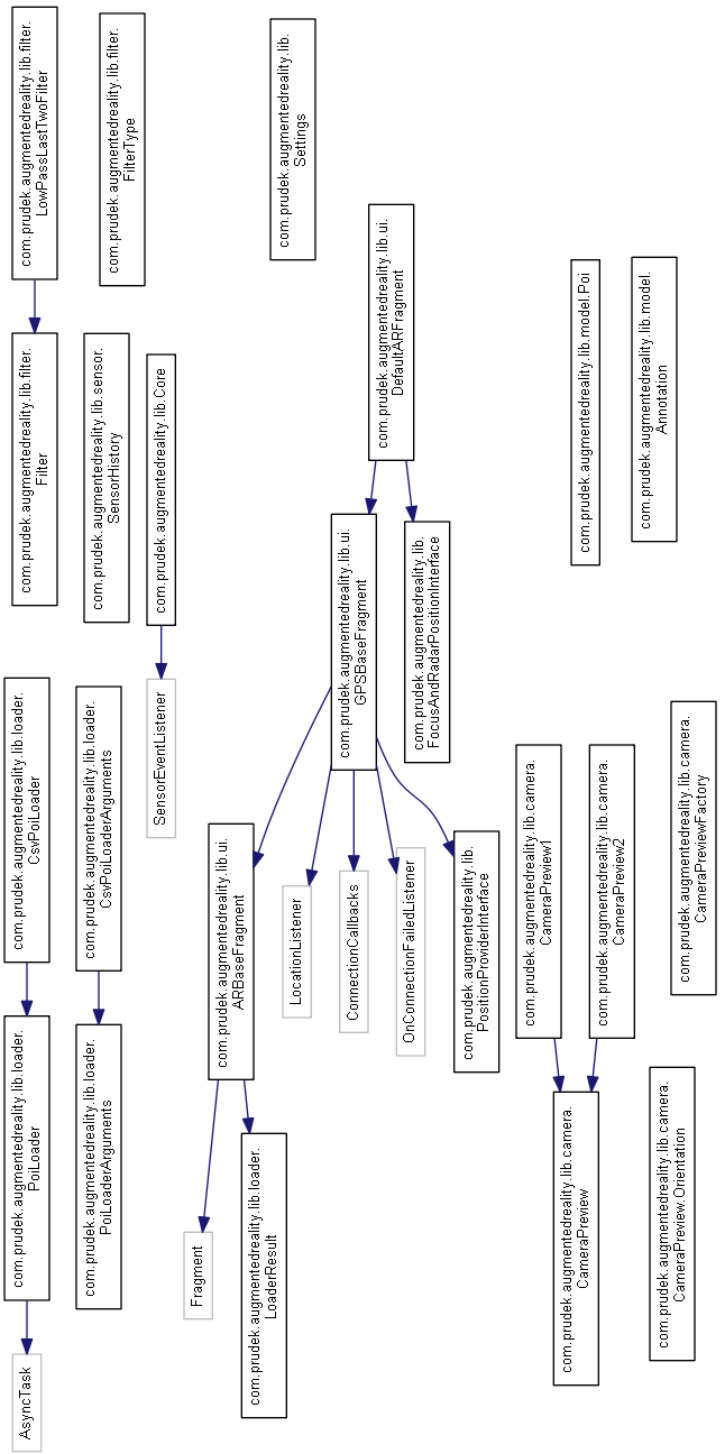
Obrázek 2.9: Vzhled dokumentace programem Doxygen [53]

2.6 Výstupy návrhu

Podařilo se navrhnout strukturu knihovny, na kterou by mělo být v budoucnu možné nabalovat další funkcionality, a vylepšovat tím kvalitu knihovny rozšířené reality. Návrh se zaměřil na efektivní rozdělení jednotlivých operací do vláken k dosažení vysoké efektivity a plynulého zobrazení informací na obrazovce. Výstupem návrhu jsou instrukce a diagramy popsané v této kapitole, které mají sloužit jako základ implementace knihovny.

Dále byl na základě návrhu vytvořen zjednodušený diagram tříd[38] (viz obrázek 2.10), který vychází z celé této kapitoly a graficky zobrazuje navržené třídy. Třídy mají anglické názvy snažící se co nejlépe popisovat jejich funkčnost, která byla vysvětlena v jednotlivých částech návrhu.

[38] Při návrhu byl diagram nakreslen ručně, tento konkrétní je vygenerován až naprogramování aplikace



Obrázek 2.10: Vizualizace návrhu tříd

Kapitola 3

Implementace

V této kapitole se autor zaměřuje na implementaci aplikace podle navržených postupů. Kapitola se nezabývá podrobným popisem programování jednotlivých částí popsaných v analýze a návrhu, ale pozastavuje se nad dříve nezmíněnými, nebo pro autora zajímavými, prvky.

3.1 Struktura

3.1.1 Projekt v programu Android Studio

Přestože se celá práce zabývá vývojem knihovny rozšířené reality, je nutné pamatovat i na ukázkovou aplikaci, která ji využívá. Výchozí struktura nově vytvořených projektů v programu Android Studio obsahuje existenci projektu, který obsahuje moduly. Ty mohou být aplikační, testovací, komunikující s vzdálenou službou a nebo právě knihovní. [14]

Tuto strukturu respektuje i projekt implementovaný v rámci této diplomové práce. Projekt obsahuje dva moduly. Knihovní modul s názvem `library`, který obsahuje knihovnu polohově zaměřené rozšířené reality a aplikační modul s názvem `demo` demonstrující fungování knihovny.

3.1.2 Rozdělení do balíčků

Pro zvýšení přehlednosti projektu byly jednotlivé třídy knihovny rozděleny do balíčků podle poskytované funkcionality. Již v průběhu vývoje byla znát časová úspora získaná díky tomuto kroku. Třída provádějící výpočty, nastavení, rozhraní pro poskytovatele pozice a rozhraní definující metody přepočítávání radaru a vybraných bodů nejsou umístěny do žádného balíčku. Následuje seznam a stručný popis jednotlivých balíčků.

- **camera** Balíček s třídami zajišťujícími náhled kamery
- **comparers** Třídy umožňující porovnávat a tedy i řadit body

- **debug** Obsahuje struktury pro pohodlnější a efektivnější ladění aplikace
- **filter** Třídy starající se o vyhlazení surových dat ze senzorů
- **loader** Obsahuje vše, co souvisí s načítáním bodů zájmu
- **model** Součástí balíčku jsou třídy reprezentující bod zájmu a anotaci (třída držící informace o umístění bodu na obrazovce)
- **sensor** Zahrnuje třídu pro uchovávání informací ze senzorů
- **ui** Fragmenty popř. aktivity knihovny
  - **view** Obsahuje vlastní vykreslitelné UI prvky (informační bublina, značka s čárou, radar)
- **utils** Zde se nachází různé pomocné funkce

3.2 Specifické součásti Android aplikace

3.2.1 Aplikační manifest

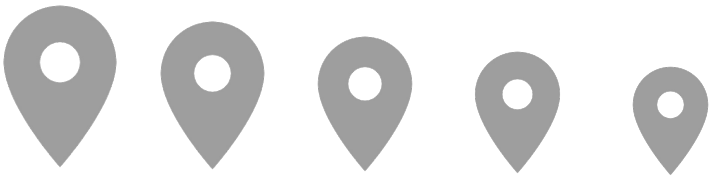
Aplikační manifest je soubor obsahující základní informace o aplikaci, kterými je například název balíčku pro distribuci, popis základních komponent, nutné součásti cílového zařízení nebo potřebná oprávnění [5]. U vyvíjené knihovny je velmi stručný. Deklaruje oprávnění `CAMERA` pro práci s kamerou a `ACCESS_FINE_LOCATION` pro přístup k přesné poloze zařízení. Dále obsahuje název balíčku a požaduje pro svůj běh přítomnost zadního fotoaparátu v zařízení.

3.2.2 Parametry sestavení

Sestavení proběhlo pro nejnovější verzi Android SDK, v době psání práce to byla verze s označením 23. Stejně tak byl zvolen nejnovější sestavovací nástroj s označením `23.0.3`. Byl použit plugin *com.android.library*, který zajišťuje, že je s výsledným programem nakládáno jako s knihovnou.

3.2.3 Resources

Ve složce *res* se nacházejí prostředky, se kterými je možné pracovat v rámci Android aplikace. Prostředky jsou oddělené od kódu a lze je specifikovat unikátně pro konkrétní konfigurace na základě rozdílného typu displeje, jazyku nebo např. orientace zařízení. Níže jsou popsány prostředky použité ve vyvíjené knihovně polohově zaměřené rozšířené reality podle jednotlivých adresářů.



Obrázek 3.1: Vzhled značky zobrazované v aplikaci ve směru bodu zájmu

Drawable

Adresář pro grafiku obsahuje ikony reprezentující body zájmu ve formátu *PNG* (Portable Network Graphics) s průhledným pozadím a šedým popředím. Tyto značky jsou škálované na pět různých velikostí a v aplikaci jsou zobrazovány podle vzdálenosti k bodu zájmu, který reprezentují. Vzhled značek a poměr velikostí lze pozorovat na obrázku 3.1.

Layout

Ve složce s grafickými rozloženími stránek nebo prvků (angl. *layouts*) se nachází pouze dva soubory, z nichž jeden reprezentuje vzhled bubliny s dodatečnou informací o bodu zájmu a druhý definuje rozvržení celého fragmentu. Tyto prvky jsou podrobněji popsány v sekci 3.3.4.

Raw

Adresář pro umístění libovolných souborů, které jsou potřeba pro běh aplikace. V tomto konkrétním případě obsahuje soubor ve formátu *CSV* se všemi testovacími zdrojovými body zájmu. Tento soubor je načten a parsován pomocí třídy `CSVLoader` implementované v rámci vyvíjené knihovny.

3.3 Podrobnosti implementace

3.3.1 Použité knihovny

V této sekci nalezne čtenář seznam knihoven použitých v aplikaci. Dále budou diskutovány důvody výběru konkrétních řešení a licenční podmínky. Základní knihovny programovacího jazyku Java a vývojová sada Android SDK nejsou zmíněny. Licenční podmínky pro užití Android SDK jsou k dispozici na [22] a jsou uzavírány se společností Google.

KD-tree

Autor neshledal nutné vytvářet vlastní implementaci návrhem zvolené datové struktury KD-tree pro uchovávání bodů zájmu, a proto se rozhodl využít

některé již existující knihovny. Byla vybrána knihovna od profesora Simona D. Levyho dostupná na jeho webových stránkách[39]. Knihovna je malá, srozumitelně dokumentovaná, na fórech označována jako prověřená a obsahuje všechny funkce, které byly vyžadovány. Z těchto důvodů byla upřednostněna před jinými, a to i přes poznámku autora o existenci pravděpodobně lepších implementací. Zvolená knihovna implementující KD-tree podléhá licenci GNU Lesser General Public License (LGPL), jejíž úplné znění lze dohledat na [34].

Google play services API

Přístup k aktuální pozici pomocí Google play services API byl detailně probrán v 1.7.2, a tento postup byl také použit v implementaci. Autor práce bohužel nebyl schopen dohledat detaily licencování k lokačnímu rozhraní knihovny.

3.3.2 Režim orientace

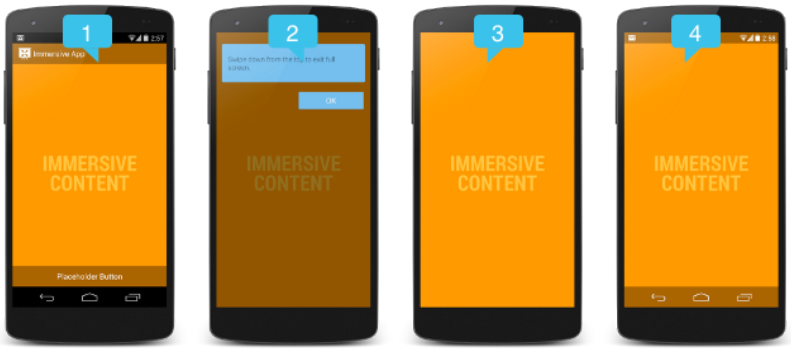
Zadání nespecifikuje v jakém režimu orientace má knihovna pracovat. Nabízí se hned několik řešení, které se liší mj. schopností reagovat na systémovou událost změny orientace, což však při povaze aplikace a dostupnosti dat ze senzorů nemusí být nutné.

- Pracovat pouze ve vertikální/horizontální poloze
- Podporovat obě možnosti bez reakce na změnu orientace zařízení (nutné specifikovat zvolený režim před zobrazením)
- Přizpůsobení aktuální orientaci telefonu
- Rotace vykreslovaným obsahem na základě dat ze senzorů

Po důkladném zvážení byl zvolen způsob, který obnáší podporu jak pro horizontální, tak pro vertikální polohu zařízení. Knihovna však nedokáže reagovat na systémovou změnu orientace, a proto je nutné při jejím použití v aktivitě nastavit pouze jeden povolený režim zobrazení. V praxi to znamená, že vývojář aplikace používající knihovnu rozhodne o vhodné orientaci, která bude fixní po celou dobu pozorovaní rozšířené reality. Podpora reakce na změnu orientace zařízení nebyla implementována z důvodu nízké priority, neboť autor této práce se domnívá, že změna orientace obrazovky během pozorování by mohla působit rušivě a mást uživatele.

Kvůli fungující podpoře bylo nutné implementovat pomocnou třídu `OrientationUtils`, která si je vědoma aktuální orientace a na jejím základě vrací příslušné hodnoty některých konstant a proměnných. Jako příklad lze uvést aktuální úhel kamery nebo velikost posunutí značky po vertikální ose.

[39] `http://home.wlu.edu/~levys/software/kd/`



Obrázek 3.2: Stavy režimu immersive full-screen [23]

3.3.3 Integrace Camera2 API

Při implementaci navrženého vzoru pro obsluhu náhledu kamery došlo ke komplikaci, jelikož se nepodařilo správně naprogramovat přístup pomocí nového *Camera2 API*, a to i přes existenci oficiálního příkladu na [6] a série videonávodů na [58]. Při pokusech o integraci rozhraní opakovaně vracelo nereálný obraz, a protože zastaralý přístup pomocí starého API fungoval korektně i na nových systémech, bylo rozhodnuto o neimplementování této části knihovny.

3.3.4 Vzhled obrazovky

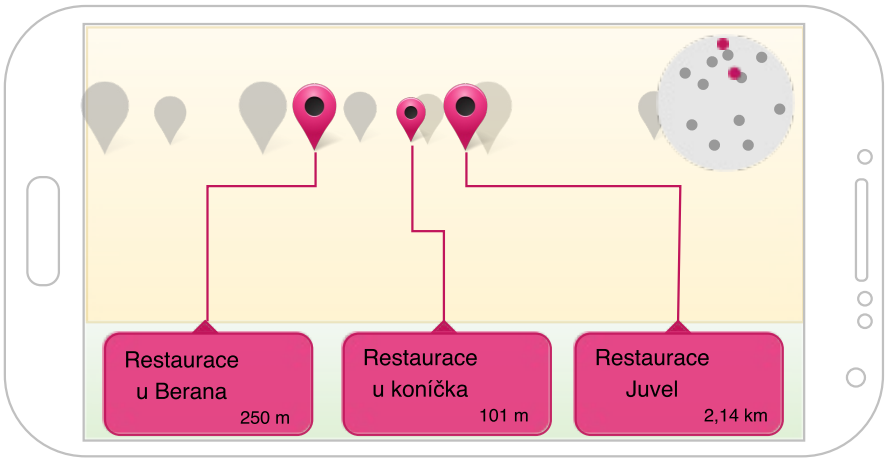
Immersive Full-Screen Mode

Z povahy aplikace vyplývá vhodnost maximalizace zobrazovací plochy, a proto bylo využito nové vlastnosti Android SDK s názvem *Immersive Full-Screen Mode*. Tento přístup funguje od verze API 19, na zařízeních se starším systémem se chová jako klasický celoobrazovkový mód. *Immersive Full-Screen Mode* je režim, který umožňuje využít opravdu celé plochy obrazovky. V tomto režimu jsou schovány všechny prvky uživatelského rozhraní systému včetně navigační lišty, kterou si však uživatel může zobrazit krátkým gestem směřovaným z kraje obrazovky. S prvním vstupem do tohoto módu se uživateli zobrazí i návod na jeho opuštění. [23]

Jednotlivé stavy popisovaného režimu je možné pozorovat na obrázku 3.2.

Zákaz vypnutí obrazovky

Aplikace nevyžaduje žádnou interakci s uživatelem, a proto je nutné zakázat operačnímu systému vypnutí obrazovky. To lze provést nastavením atributu `keepScreenOn` ve zvoleném kontejneru na hodnotu `true` [11].



Obrázek 3.3: Upravený návrh vzhledu aplikace

Rozložení prvků UI

Během realizace navrhovaného vzhledu z 2.4.7 došlo k jeho úpravě. Zde se autor pokusil použít zcela nový přístup propojení informační bubliny se značkou pomocí lomené čáry. Tento přístup je vyobrazen na obrázku 3.3. Ačkoliv přidaná lomená čára může působit trochu zmatečně, dovolí umístit bubliny na fixní pozice, čímž je možné využít celou šíři obrazovky pro zobrazení dodatečných informací.

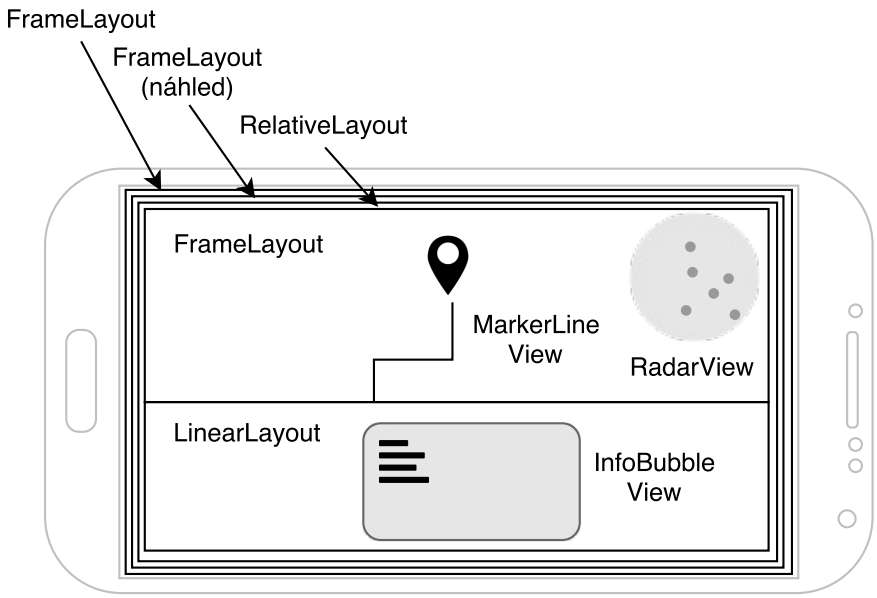
Obrázek 3.4 zobrazuje realizace navrhovaného rozložení prvků na obrazovce na základě tříd, které je reprezentují. Radar, značky a bubliny jsou implementovány pomocí tříd, které dědí z `View` a dokáží se tedy vykreslit na obrazovku.

3.3.4.1 Bitmap Cache

Pomocná statická metoda `getBitmapCache` třídy `BitmapUtils` vrací bitmapu značky požadované velikosti, která závisí na vzdálenosti bodu. Při prvním použití jsou do paměti načteny všechny dostupné obrázky těchto značek, což značně urychluje vykreslování snímku.

3.3.5 Runtime permissions

Počínaje nejnovější verzí operačního systému, kterou je Android 6.0 (API 23), dochází ke změně přístupu operačního systému k udělování povolení. To neprobíhá při instalaci aplikace, jak tomu bylo v předchozích verzích, ale uděluje se při běhu aplikace. Udělené povolení je možné kdykoliv odebrat, a proto je nutné při vývoji aplikací mířených na nové verze systému kontrolovat vlastnictví příslušného povolení vždy před voláním potencionálně nebezpečné funkce. [19]



Obrázek 3.4: Upravený návrh vzhledu aplikace

3.3.6 Algoritmus pro výběr označených bodů

Algoritmus, jenž rozhoduje, které body se mají zvýraznit, je velice důležitý pro správné fungování vyvíjené knihovny. Je třeba vhodně vybalancovat řadu situací, ke kterým může dojít. Řešení musí odhadnout, které body chce pozorovatel vidět, ale zároveň nesmí žádné úplně opomíjet. Předpokládá se, že uživatel preferuje místa, která jsou k němu blíže, a která se nachází uprostřed obrazovky. Zároveň je třeba předcházet zbytečně častým změnám vyznačených bodů, jelikož to může vést k dezorientaci pozorovatele a nečitelnosti doplňujících informací.

Vyvinutý algoritmus bere v potaz několik faktorů, podle kterých určuje prioritu zvolení daného bodu. Body s největší prioritou jsou vyznačeny a zůstávají vyznačeny po určený počet snímků (vydrží-li ve viditelné oblasti). Při změně polohy zařízení je přepočet proveden okamžitě. Faktory ovlivňující přidělenou prioritu jsou popsány v následujícím výčtu.

- **Vzdálenost od středu obrazovky** Zvolený počet bodů, nacházejících se nejblíže ke směru pohledu kamery, obdrží bonus, který je odstupňován podle vzdálenosti od středu obrazovky
- **Vzdálenost od pozorovatele** Body jsou rozděleny do pěti skupin podle vzdálenosti, a čím blíže jsou, tím dostávají větší bonusové ohodnocení na základě hodnoty příslušné proměnné v nastavení
- **Náhoda** Pro každý bod se generuje pseudonáhodné číslo v určeném rozsahu

- **Bonus za nezvolení** Bod dostane příslušné bonusové ohodnocení, pokud nebyl v minulém kroku algoritmu vybrán

3.4 Dokumentace ke knihovně

Součástí zadání práce je i požadavek na dokumentaci. Ta se skládá ze dvou částí manuálu a generované dokumentace.

3.4.1 Manuál pro vývojáře

Manuál je napsán stejně jako celá práce v českém jazyce. Obsahuje ukázky kódu doplněné popisem popř. diagramy a náčrty upřesňujícími význam parametrů. Hotový manuál je obsažen na přiloženém CD a jako příloha B

3.4.2 Generovaná dokumentace programem doxygen

Ačkoliv to není zvykem, rozhodl se autor napsat dokumentaci v českém jazyce, stejně jako vše ostatní v této práci. Ukázka použití a stručný popis jednotlivých tříd je již obsažen ve vývojářském manuálu. V rámci generované dokumentace byly popsány pouze části, ke kterým má vývojář používající knihovnu přístup. Dle autora není nutné detailně popisovat součásti jádra aplikace. Největší pozornost si zasloužily nastavitelné parametry knihovny, se kterými bude vývojář pracovat nejčastěji.

Výsledná dokumentace, spolu s vygenerovanými diagramy, se nachází na přiloženém CD.

Kapitola 4

Testování

V této kapitole se čtenář seznámí s postupy, které byly využity pro ověření správného fungování výsledné aplikace. Vzhledem k senzorové povaze knihovny nebylo vhodné testovat za požití emulátoru a musela být použita pouze reálná zařízení. Veškeré testy probíhaly se sadou dat o velikosti 10 815, která obsahovala restaurace, parkoviště, bankomaty, zastávky a další body zájmu stažené z [72]. V prvotní fázi aplikace při ladění výpočtů byla pro zjednodušení použita sada několika měst se souřadnicemi. Soubory jsou obsaženy ve zdrojovém adresáři *raw* vyvíjené knihovny.

4.1 Spuštění na různých modelech zařízení

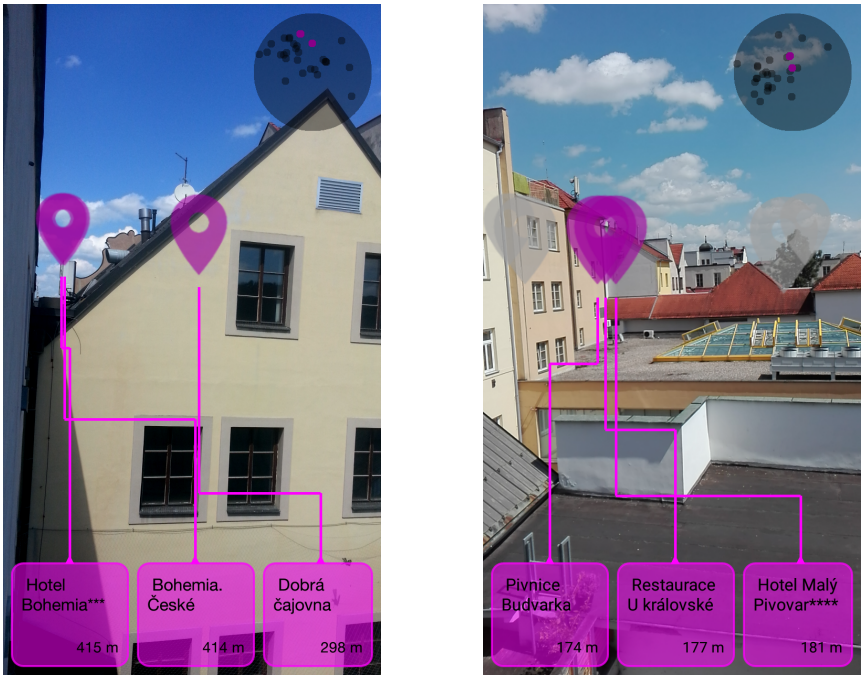
Testování knihovny probíhalo průběžně při jejím vývoji. K tomu byly používány dva mobilní telefony značek Samsung a Huawei s různými parametry a verzí systému. Na těchto zařízeních fungovala výsledná aplikace bez problému v obou podporovaných orientacích. Zařízení značky Samsung je nezanedbatelně starší a má slabší hardwarovou konfiguraci. Při testování s větším počtem bodů bylo překreslování znatelně pomalejší, což může být způsobeno neefektivním odebíráním a opětovným přidávání prvků UI. Řešením tohoto nedostatku by mohl být nějaký způsob recyklace prvků na obrazovce v další verzi knihovny.

Po dokončení měla být aplikace dodatečně otestována na tabletu značky Acer, který bohužel neobsahuje akcelerometr, a proto nebylo možné zobrazit body zájmu. Pozitivní na tom je, že aplikace se neukončila, ale zobrazila alespoň náhled. Naopak vhodnější by bylo zobrazení dialogu s popisem chyby.

Byl proveden také pokus o testování na zařízení LG Nexus s nejnovějším operačním systémem, který se však nezdařil. Následnou analýzou chyb bylo zjištěno, že se tak stalo pravděpodobně kvůli zastaralému způsobu přístupu ke kameře, který musel být použit kvůli nezdaru v implementaci popsaném v bloku 3.3.3.

Tabulka 4.1: Parametry zařízení určených k testování

Typ zařízení	Verze OS	Rozlišení obrazovky	Fyzická velikost obrazovky	Orientační cena[40]
Huawei P8 Lite	5.0	1280 x 720	5 palců	5 488 Kč
Samsung Galaxy S3	4.3	1280 x 720	4,8 palců	4 690 Kč
LG Nexus 5X	6.0	1920 x 1080	5,2 palce	9 890 Kč
Acer Iconia Tab 8	4.4	1280 x 800	8 palců	3 989 Kč



(a) Vzhled aplikace v telefonu Samsung

(b) Vzhled aplikace v telefonu Huawei

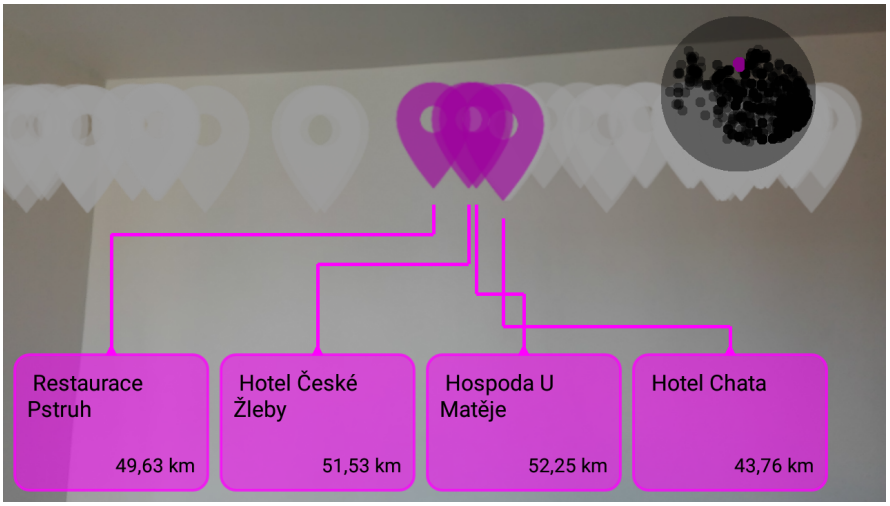
Obrázek 4.1: Porovnání vzhledu výsledné aplikace na odlišných zařízeních

Tabulka 4.1 popisuje rozdíly v parametrech zmíněných testovacích zařízení a na obrázku 4.1 lze pozorovat stejný vzhled aplikace ve vertikálním módu na zařízeních Samsung a Huawei se stejným rozlišením.

4.2 Test nastavení parametrů

Jelikož mnoho částí aplikace závisí na správné volbě parametrů, bylo věnováno mnoho času jejich vhodnému výběru pro výchozí konfiguraci knihovny. Jedi-

[40]Cena pochází z portálu CZC.cz a je aktuální ke dni 9.5.2016



Obrázek 4.2: Snímek obrazovky běžící aplikace s velkým množstvím zobrazených míst

ným výstupem z těchto testů je subjektivní názor autora na vhodné zvolení parametrů a z nich vycházející nakonfigurování knihovny použité pro uživatelské testování.

Za zmínku stojí provedené testování na zjištění maximálního možného množství zobrazených bodů. Ačkoliv to pravděpodobně nemá reálné využití, knihovna byla schopna přepočítávat a zobrazit velké množství bodů, což bylo ověřeno na konfiguraci s hodnotou 3 000[41]. Tento fakt poukazuje na správnost návrhu vícevláknového modelu knihovny. Důkazem toho tvrzení může být snímek obrazovky na obrázku 4.2 zobrazující aplikaci fungující pro 3 000 bodů zájmu.

V rámci testu bylo také zjištěno, že v praxi je rozumné udržet počet zobrazovaných bodů v řádech několika málo desítek. Při zvolení větších hodnot se již v informacích nelze téměř nijak orientovat a bylo by nutné implementovat rozdílnou strategii výběru zobrazených bodů.

4.3 Porovnání s konkurencí

Výsledná aplikace byla porovnána s konkurenty zmíněnými v sekci 1.3. Porovnání proběhlo v několika kategoriích. Pozitivním zjištěním je konkurenceschopnost knihovny v porovnání s mnohem komplexnějšími a placenými produkty.

[41] Docházelo k drobným, ale postřehnutelným zpožděním překreslování obrazovky

Výkon

Díky kvalitnímu rozvržení do vícevláknové architektury, předčí knihovna implementovaná v rámci této práce ve výkonnosti své konkurenty. Nemluvě o nedostižném Junaiu a velmi efektivně pracujícím Wikitude, jsou na tom ostatní analyzovaná řešení hůře a nejsou schopna zobrazit srovnatelné množství bodů zájmu..

Řešení překryvů

Řešení překryvů je závislé na konkrétním místě a účelu použití knihoven. Jelikož z existujících dokázaly řešit překryvy jen Junaio a svým netradičním způsobem Layar SDK, i nedokonalé řešení implementované v rámci této práce je nutno ocenit.

Práce se senzory (reakce na otáčení)

Jedná se rozhodně o část aplikace, ve které knihovna ztrácí oproti svým soupeřům. Autor však věří, že vylepšením používaného filtru dat a přidáním pokročilých heuristik lze knihovnu rozšířit a odstranit tak nedostatky v této oblasti.

Celkový dojem a použitelnost

Přestože aplikace nedosahuje nesporných kvalit popsané knihovny Junaoi, celkový dojem z jejího používáni byl rozhodně pozitivní. V porovnání s ostatními působí aplikace živě a prezentuje informace uživatelsky příjemným způsobem. Oproti konkurentům ztrácí ve stabilizaci pozorovaných bodů.

4.4 Uživatelské testování

Tato sekce popisuje proběhlé testování míry použitelnosti aplikace. Pod pojmem míra použitelnosti si lze představit „jednoduchost používání a naučení se práce s lidmi vytvořeným objektem“ [65].

4.4.1 Průběh testování

Pro účely této formy evaluace programu byl vytvořen list s pokyny a otázkami dostupný v příloze C. Pokyny obsahují dva testovací scénáře, 12 otázek hodnocených bodovou škálou, jednu otázku ano/ne a možnost připojit vlastní komentář k aplikaci.

Testování probíhalo za osobní přítomnosti autora, který pozoroval testující při jejich prvním kontaktu s aplikací a po dokončení scénářů vedl dialog o použitelnosti aplikace. Díky tomu bylo možné získat zajímavé poznatky.

4.4.2 Charakteristika testovaných osob a prostředí

Testování proběhlo na vzorku deseti osob. Ze zúčastněných byly tři ženy a sedm mužů ve věkovém rozmezí 16 – 54 let. Všichni oslovení byly české národnosti a dobrovolně souhlasili s účastí. Testování probíhalo na různých místech v Českých Budějovicích.

4.4.3 Výsledky testování

Otázky hodnocené body

Odpovědi jednotlivých respondentů na bodované otázky jsou zachyceny v příloze D. Vyšší hodnota vyjadřuje vyšší míru souhlasu s danou otázkou. Výsledné hodnoty se zpravidla liší, jelikož respondenti volili odlišnou přísnost hodnocení, zřejmě podle náročnosti jejich požadavků na aplikaci. U některých ukazatelů se však většina dotázaných shodla na obdobném hodnocení.

Graf na obrázku 4.3 vizualizuje bodování uživateli. Modře a tučné zvýrazněné sloupce vyznačují průměrné hodnocení. Pro představu jak hodnotili jednotliví uživatelé, jsou hodnoty propojeny (jiný důvod pro souvislost grafu není) a barevně odděleny. Zajímavostí je, že méně technicky zdatní uživatelé hodnotili aplikaci většinou pozitivněji.

Na otázku, zdali je vhodné zobrazit informační bubliny ve více řádcích odpovědělo 20 % uživatelů kladně.

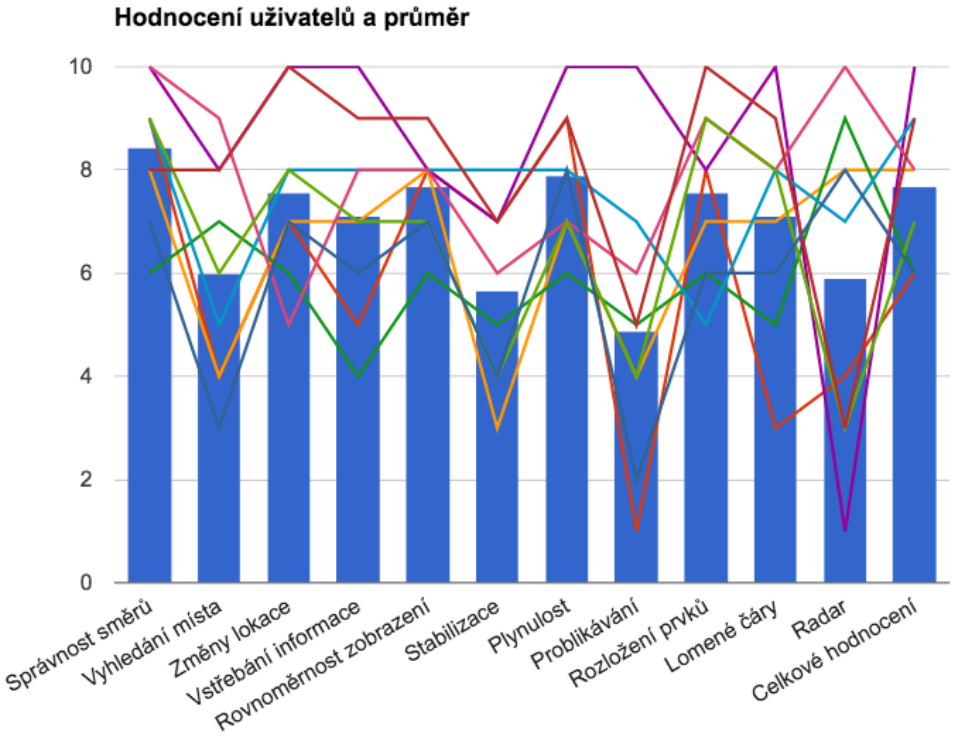
Vlastní poznámky k použití

Většina testujících uživatelů nevyužila možnost zapsat doplňující informace, a když ano, tak se odpovědi moc nelišily. Nejčastěji zmiňované nedostatky byly spojené s nepřesnými daty poskytovanými jednotlivými senzory. Konkrétně se jednalo o nemožnost zaměření se na konkrétní bod a někdy až příliš častou změnu doplňujících informací (při posouvání vyznačených bodů mimo viditelný úhel).

Diskuse

Diskuse byly překvapivě plodné, zvláště jedna, které se účastnili hned čtyři testeři. Uživatelé servírovali velké množství nápadů, z nichž většina byla specifická pro konkrétní použití. Aplikace byla přijímána pozitivně s nejčastějším využitím pro hledání vhodného restauračního zařízení na páteční večer.

Vzhled aplikace se převážně líbil, ale téměř každý tester přidal drobné připomínky. I když ne všichni by využili možnost přizpůsobení prvků uživatelského rozhraní, je určitě vhodné uvažovat o nějakém nastavení popř. návrhovém módu, kde by mohli uživatelé měnit některé parametry. Při zájmu ze strany vývojářů používajících knihovnu existuje možnost vytvoření návrhového programu pro jednodušší nastavení knihovny.



Obrázek 4.3: Vizualizace výsledků uživatelského testování

Nejvíce negativních ohlasů vyvolalo již zmíněné přeskakování informací a fakt, že se pozorovatel nemohl zaměřit na vybrané body. Byl navržen inovativní přístup umožňující dotykem „zmrazit“ aktuální obrazovku a následně vybrat body, u kterých bude zobrazena doplňková informace. Tím se dosáhne možnosti porovnávat fixovaný bod (nebo více bodů) s ostatními na obrazovce.

Další požadovanou funkcí ze strany uživatelů byla možnost zobrazit mapu se zvýrazněnými body zájmu, pro které jsou zrovna zobrazeny doplňující informace. Testující uživatelé se vyjádřili kladně k rozšíření o vertikální pohyb značek po obrazovce na základě natočení zařízení. Někteří by také uvítali prvek pro omezení zobrazovaných bodů maximální vzdáleností a možnost přidat oblíbené body zájmu.

4.5 Výstupy provedeného testováním

Níže uvedené seznamy obsahují výsledky provedeného testování a jeho zpracování v krátkých bodech. Poznatky jsou rozdělené do dvou kategorií, kterými jsou nedostatky a návrhy na vylepšení knihovny.

Nedostatky

- Aplikace nedokáže korektně uložit svůj stav a při jejím přenesení do pozadí je ukončena
- Nefunkčnost aplikace na Android 6.0 kvůli nefungujícímu novému přístupu ke kameře
- V knihovně chybí informační dialogy (např. neexistence potřebných senzorů)
- Při překreslování obrazovky nedochází k recyklaci prvků UI (projevuje se pomalejším překreslováním)
- Body nejsou stabilizované (autor doporučuje použití vyspělejšího filtru a více senzorů)

Návrhy na vylepšení

- Zobrazení radaru nebo mapy při vodorovné poloze zařízení
- Možnost zafixování aktuálně vyznačených bodů kliknutím na obrazovku
- Posouvání značky po vodorovné ose při náklonu telefonu
- Možnost zvolit oblíbené body zájmu
- Integrace tlačítka zobrazení mapy s právě pozorovanými body
- Možnost ovlivnit maximální vzdálenost zobrazení bodů
- Začlenění návrhářského módu (pro uživatele v rámci aplikace nebo pro vývojáře mimo ní)

Závěr

Hlavním cílem této diplomové práce bylo vytvoření knihovny pro operační systém Android poskytující funkci polohově zaměřené rozšířené reality. V rámci přípravných fází bylo třeba provést rešerši již existujících konkurenčních knihoven. Výstupem práce měla být kromě knihovny samotné, také testovací aplikace, která této knihovny využívá. Neméně důležitou součástí měla být vývojářská dokumentace. Výsledný program měl být navržen pro příjemné používání, a proto bylo vhodné provést uživatelské testování. Bylo požadováno zaměřit se na vysoký výkon knihovny a vytěžení maxima z práce se senzory.

Autor se v první kapitole práce zabýval analýzu informací potřebných k navržení knihovny včetně průzkumu existujících řešení a jejich porovnání. V další kapitole byl proveden návrh realizace se zaměřením na vytvoření výhodného rozvržení dílčích částí knihovny do vláken tak, aby bylo dosaženo co největšího výkonu. Navržené řešení bylo realizováno a současně byly popsány kritické části implementace. Následně byla vytvořena ukázková aplikace používající knihovnu a provedeny testy, včetně uživatelského testování s diskuzí. Poslední kapitola práce popisuje průběh tohoto testování a interpretuje zjištěné výsledky. V závěrečné fázi byla vygenerována dokumentace knihovny ve formě hypertextového dokumentu a sepsán manuál pro vývojáře.

Z výše uvedených informací plyne, že zadání bylo splněno v téměř celém rozsahu. Hlavním cílem bylo vytvořit fungující aplikaci, a to se podařilo. Nebyl splněn pouze požadavek na využití maxima senzorů, a to převážně kvůli jejich absenci v zařízeních, které měl autor k dispozici pro vývoj. Uživatelské testování proběhlo sice pouze se vzorkem deseti uživatelů, dle autora však poskytlo dostatek doplňujících informací k aplikaci. Při testování byly zjištěny drobné nedostatky knihovny, které ale nelze hodnotit jako závažné.

Diskuse s uživateli přinesla cenné návrhy na vylepšení výsledné aplikace a testy celkově odhalily některé nedostatky. Vývoj knihovny by měl zcela jistě pokračovat opravením chyby způsobující občasný pád aplikace při přechodu do pozadí, správnou implementací nového přístupu k fotoaparátu a přidáním informačních dialogů. Po vyřešení těchto nedostatků autor navrhuje vylepšit

práci se senzory. Prvotně by bylo vhodné pokusit se vyhladit nepřesné hodnoty pomocí sofistikovanějších filtrů a poté zakomponovat podporu pro více druhů senzorů a heuristiky na rozhodování o kvalitě měřených hodnot. Dalším v pořadí je zefektivnění vykreslování prvků uživatelského rozhraní za pomoci jejich recyklace.

Knihovnu je také možné postupně obohacovat o nové vlastnosti navržené uživateli v rámci testování. Mezi nejzajímavější patří možnost zafixovat vybrané body zájmu, zařazení tlačítka pro zobrazení mapy a přidat způsob, jakým by byl uživatel schopen ovlivnit vzdálenost zobrazovaných bodů. Mezi méně důležitá vylepšení patří zavedení oblíbených bodů, posouvání značek i po vertikální ose a možnost ovlivňování vzhledu uživatelského rozhraní přímo za běhu aplikace.

Literatura

[1] Ableson, W.; Sen, R.; King, C.: *Android in Action*. Running Series, Manning, 2012, ISBN 9781617290503.

[2] Affair advertising: *Co je AR - Rozšířená realita (Augmented Realita)*. 2014, [cit. 2016-04-18]. Dostupné z WWW: <http://www.rozsirenarealita.cz/>

[3] Allan, A.: *Basic Sensors in iOS: Programming the Accelerometer, Gyroscope, and More*. O'Reilly Media, 2011, ISBN 9781449315429, [cit. 2016-04-30].

[4] American Sentinel University: *Military GIS Operations Continue in Peace and War*. 2012, [cit. 2016-04-21]. Dostupné z WWW: <http://www.americansentinel.edu/blog/2012/08/06/military-gis-operations-continue-in-peace-and-war/>

[5] Android Developers: *App Manifest*. 2016, [cit. 2016-05-08]. Dostupné z WWW: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>

[6] Android Developers: *Camera2Basic*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <http://developer.android.com/samples/Camera2Basic/index.html>

[7] Android Developers: *Canvas and drawables*. 2016, [cit. 2016-05-01]. Dostupné z WWW: <http://developer.android.com/guide/topics/graphics/2d-graphics.html>

[8] Android Developers: *Dashboards*. 2016, [cit. 2016-04-17]. Dostupné z WWW: <http://developer.android.com/about/dashboards/index.html>

[9] Android Developers: *Dashboards*. 2016, [cit. 2016-04-23]. Dostupné z WWW: <http://developer.android.com/guide/practices/verifying-apps-art.html>

[10] Android Developers: *Handler*. 2016, [cit. 2016-04-26]. Dostupné z WWW: <http://developer.android.com/reference/android/os/Handler.html>

[11] Android Developers: *Keeping the Device Awake*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <http://developer.android.com/training/scheduling/wakelock.html>

[12] Android Developers: *Looper*. 2016, [cit. 2016-04-26]. Dostupné z WWW: <http://developer.android.com/reference/android/os/Looper.html>

[13] Android Developers: *Making Your App Location-Aware*. 2016, [cit. 2016-04-28]. Dostupné z WWW: <http://developer.android.com/training/location/index.html>

[14] Android Developers: *Managing Projects Overview*. 2016, [cit. 2016-05-08]. Dostupné z WWW: <http://developer.android.com/tools/projects/index.html>

[15] Android Developers: *Motion Sensors*. 2016, [cit. 2016-04-30]. Dostupné z WWW: <http://developer.android.com/guide/topics/sensors/sensors_motion.html>

[16] Android Developers: *Orientation Sensors*. 2016, [cit. 2016-04-30]. Dostupné z WWW: <http://developer.android.com/guide/topics/sensors/sensors_environment.html>

[17] Android Developers: *Postition Sensors*. 2016, [cit. 2016-04-30]. Dostupné z WWW: <http://developer.android.com/guide/topics/sensors/sensors_position.html>

[18] Android Developers: *Processes and Threads*. 2016, [cit. 2016-04-26]. Dostupné z WWW: <http://developer.android.com/guide/components/processes-and-threads.html>

[19] Android Developers: *Requesting Permissions at Run Time*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <http://developer.android.com/training/permissions/requesting.html>

[20] Android Developers: *SensorManager*. 2016, [cit. 2016-05-05]. Dostupné z WWW: <http://developer.android.com/reference/android/hardware/SensorManager.html>

[21] Android Developers: *Sensors Overview*. 2016, [cit. 2016-04-29]. Dostupné z WWW: <http://developer.android.com/guide/topics/sensors/sensors_overview.html>

[22] Android Developers: *Terms and Conditions*. 2016, [cit. 2016-05-08]. Dostupné z WWW: <http://developer.android.com/sdk/terms.html>

[23] Android Developers: *Using Immersive Full-Screen Mode*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <http://developer.android.com/training/system-ui/immersive.html>

[24] Aniket Thakur: *Using Async Tasks in Android*. 2015, [cit. 2016-04-26]. Dostupné z WWW: <https://opensourceforgeeks.wordpress.com/2015/11/10/using-async-tasks-in-android/>

[25] Brothersoft.com: *Corinth Micro Anatomy Augmented*. [cit. 2016-05-09]. Dostupné z WWW: <http://windows8.brothersoft.com/corinth_micro_anatomy_augmented-108307.html>

[26] Cambridge in Colour: *Understanding White Balance*. [cit. 2016-04-30]. Dostupné z WWW: <http://www.cambridgeincolour.com/tutorials/white-balance.htm>

[27] Chris Hoffman: *Why Microsoft Makes 5to15 From Every Android Device Sold*. 2014, [cit. 2016-04-17]. Dostupné z WWW: <http://www.howtogeek.com/183766/why-microsoft-makes-5-to-15-from-every-android-device-sold/>

[28] Chris Veness: *Calculate distance, bearing and more between Latitude/Longitude points*. 2015, [cit. 2016-05-08]. Dostupné z WWW: <http://www.movable-type.co.uk/scripts/latlong.html>

[29] Collins English Dictionary - Complete, Unabridged 10th Edition: Augmented reality | Define Augmented reality at Dictionary.com. [cit. 2016-04-18]. Dostupné z WWW: <http://www.dictionary.com/browse/augmented-reality>

[30] Cristobal, G.; Perrinet, L.; Keil, M.; aj.: *Biologically Inspired Computer Vision: Fundamentals and Applications*. Wiley, 2015, ISBN 9783527680498, [cit. 2016-04-27].

[31] Darryl Dennis: *Great Circle Route calculations (special considerations may apply)*. [cit. 2016-05-08]. Dostupné z WWW: <https://brilliant.org/problems/great-circle-route-calculations-special/>

[32] El-Rabbany, A.: *Introduction to GPS: The Global Positioning System*. Artech House mobile communications series, Artech House, 2002, ISBN 9781580531832, [cit. 2016-04-28].

[33] Eunjoo Im: *Thread, Looper, Handler*. 2015, [cit. 2016-04-26]. Dostupné z WWW: <https://realm.io/jp/news/android-thread-looper-handler>

[34] Free Software Foundation, Inc.: *GNU LESSER GENERAL PUBLIC LICENSE*. 2007, [cit. 2016-05-08]. Dostupné z WWW: <https://www.gnu.org/copyleft/lesser.html>

[35] Furht, B.: *Handbook of Augmented Reality*. SpringerLink : Bücher, Springer New York, 2011, ISBN 9781461400646.

[36] Gamma, E.; Helm, R.; Johnson, R.; aj.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994, ISBN 9780321700698, [cit. 2016-05-05].

[37] Garmin: *Creating Custom POI Files*. [cit. 2016-05-05]. Dostupné z WWW: <http://www8.garmin.com/products/poiloader/creating_custom_poi_files.jsp>

[38] Google Inc. and Open Handset Alliance: *Android Open Source Project*. 2016, [cit. 2016-04-18]. Dostupné z WWW: <http://source.android.com/>

[39] Google Inc. and Open Handset Alliance: *Augmented Reality — Wikipedia, The Free Encyclopedia*. 2016, [cit. 2016-04-18]. Dostupné z WWW: <https://en.wikipedia.org/wiki/Augmented_reality>

[40] Havryluk, M.; Čermák, O.; Havlíček, F.; aj.: *KNIHOVNY, LOKACE A SLUŽBY GOOGLE*. 2016, [cit. 2016-04-28]. Dostupné z WWW: <https://edux.fit.cvut.cz/courses/BI-AND/_media/lectures/11.zip>

[41] Havryluk, M.; Čermák, O.; Havlíček, F.; aj.: *PŘEDSTAVENÍ PLATFORMY ANDROID*. 2016, [cit. 2016-04-17]. Dostupné z WWW: <https://edux.fit.cvut.cz/courses/BI-AND/_media/lectures/01.pdf>

[42] Heineman, G.; Pollice, G.; Selkow, S.: *Algorithms in a Nutshell*. In a Nutshell (O'Reilly), O'Reilly Media, 2008, ISBN 9781449391133, [cit. 2016-04-29].

[43] Heinen, S.: *DroidAR Screenshots*. 2012, [cit. 2016-05-10]. Dostupné z WWW: <http://droidar.blogspot.cz/2012/10/droidar-screenshots.html>

[44] Hájek, L.; Hambálek, T.: *Gyroskopy*. 2010, [cit. 2016-04-30]. Dostupné z WWW: <http://fyzsem.fjfi.cvut.cz/2010-2011/Zima10/proc/gyroskopy.pdf>

[45] Jakob Jenkov: *Java Synchronized blocks*. 2014, [cit. 2016-04-26]. Dostupné z WWW: <http://tutorials.jenkov.com/java-concurrency/synchronized.html>

[46] Joyce Echessa: *Google Play Services for Location and Activity Recognition*. 2015, [cit. 2016-04-28]. Dostupné z WWW: <http://www.sitepoint.com/google-play-services-location-activity-recognition/>

[47] Lapiš, R.: *Srovnání navigačních systémů GPS, GLONASS, GALILEO, BEIDOU*. 2011, [cit. 2016-04-28]. Dostupné z WWW: <http://wiki.cs.vsb.cz/images/8/88/Gis_lap028.pdf>

[48] Lars Vogel: *Java concurrency (multi-threading) - Tutorial*. 2016, [cit. 2016-04-30]. Dostupné z WWW: <http://www.vogella.com/tutorials/JavaConcurrency/article.html>

[49] Lee, W.-M.: *Android Application Development Cookbook : 93 Recipes for Building Winning Apps (1)*. Wrox, 2012, ISBN 9781118177679, [cit. 2016-04-28].

[50] Liles, S.: *Asynchronous Android*. Community experience distilled, Packt Publishing, 2013, ISBN 9781783286881, [cit. 2016-04-23].

[51] MacLean, D.; Komatineni, S.; Allen, G.: *Pro Android 5*. Apress, 2015, ISBN 9781430246817, [cit. 2016-04-17].

[52] Marc Fischer: *Digital Content Creation for Seamless Augmented Reality*. 2015, [cit. 2016-04-21]. Dostupné z WWW: <https://www.vs.inf.ethz.ch/edu/FS2015/UCS/slides/MarcFischer_SeamlessAR.pdf>

[53] Matt Butcher: *Documenting PHP with Doxygen: The Pros and Cons*. 2012, [cit. 2016-05-07]. Dostupné z WWW: <http://technosophos.com/2012/02/01/documenting-php-doxygen-pros-and-cons.html>

[54] Meier, R.: *Professional Android 4 Application Development*. ITPro collection, Wiley, 2012, ISBN 9781118237229, [cit. 2016-04-18].

[55] Milette, G.; Stroud, A.: *Professional Android Sensor Programming*. ITPro collection, Wiley, 2012, ISBN 9781118240458, [cit. 2016-04-30].

[56] Morgillo, I.: *RxJava Essentials*. Community experience distilled, Packt Publishing, 2015, ISBN 9781784393571, [cit. 2016-04-26].

[57] Nette Foundation: *Psaní dokumentace*. 2016, [cit. 2016-05-07]. Dostupné z WWW: <https://nette.org/cs/writing>

[58] Nigel: *Camera2 API Video*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <https://www.nigeapptuts.com/android-camera2-api-looknfeel/>

[59] Oaks, S.; Wong, H.: *Java Threads*. Nutshell handbooks, O'Reilly Media, 2004, ISBN 9780596007829, [cit. 2016-04-30].

[60] Open Source Initiative: *The Open Source Definition (Annotated)*. [cit. 2016-04-18]. Dostupné z WWW: <https://opensource.org/osd-annotated>

[61] OpenSignal: *Android Fragmentation Visualized (August 2015)*. 2015, [cit. 2016-04-17]. Dostupné z WWW: <http://opensignal.com/reports/2015/08/android-fragmentation/>

[62] Oracle: *Lock (Java Platform SE 7)*. 2016, [cit. 2016-05-10]. Dostupné z WWW: <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Lock.html>

[63] Otec Fura: *Zamyšlení nad tvorbou programátorské dokumentace*. 2011, [cit. 2016-05-07]. Dostupné z WWW: <http://blog.novoj.net/2011/03/16/zamysleni-nad-tvorbou-programatorske-dokumentace/>

[64] Pecinovský, R.: *Myslíme objektově v jazyku Java 1.5*. 2004, [cit. 2014-05-14].

[65] Žikovský PhD., I. P.: *4. přednáška – Usability. Special Testing. Personas*. 2016, [cit. 2016-05-09]. Dostupné z WWW: <https://edux.fit.cvut.cz/archive/B151/MI-NUR/_media/lectures/x04-usability_special_testing_personas.pdf>

[66] Samsung: *Samsung Galaxy S5*. [cit. 2016-04-30]. Dostupné z WWW: <http://www.samsung.com/cz/galaxys5/features.html>

[67] Servisní středisko pro podporu e-learningu na MU: *Základy digitálních dokumentačních technik a možnosti jejich využití*. [cit. 2016-04-30]. Dostupné z WWW: <http://is.muni.cz/elportal/estud/prif/ps08/digitech/web/kapitola2/3.html>

[68] Shumaker, R.; Lackey, S.: *Virtual, Augmented and Mixed Reality: Designing and Developing Augmented and Virtual Environments: 6th International Conference, VAMR 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings*. Lecture Notes in Computer Science, Springer International Publishing, 2014, ISBN 9783319074580, [cit. 2016-04-19].

[69] Soegaard, M.; Dam, R.: *Encyclopedia of Human-Computer Interaction*. Interaction Design Foundation, 2013, [cit. 2016-04-17].

[70] Sood, R.: *Pro Android Augmented Reality*. Apressus Series, Apress, 2012, ISBN 9781430239451, [cit. 2016-04-17].

[71] SPIEGEL ONLINE : *Rebuilding the Berlin Wall with Augmented Reality*. 2010, [cit. 2016-04-19]. Dostupné z WWW: <http://www.spiegel.de/international/spiegel/when-science-fiction-becomes-reality-rebuilding-the-berlin-wall-with-augmented-reality-a-704970.html>

[72] s.r.o., C.: *POI pro GARMIN, TOMTOM, MIO a jiné GPS NAVIGACE / Czech points of interest*. 2016, [cit. 2016-05-10]. Dostupné z WWW: <poi.cz>

[73] Taras Kunyk: *Fused Location Provider*. 2013, [cit. 2016-04-28]. Dostupné z WWW: <http://blog.lemberg.co.uk/fused-location-provider>

[74] Tom Lewis: *Augmented reality iPad app used in liver surgery to aid tumour resection*. 2013, [cit. 2016-04-21]. Dostupné z WWW: <http://www.imedicalapps.com/2013/08/augmented-reality-ipad-app/>

[75] W, R.: *Learning Android Google Maps*. Packt Publishing, 2015, ISBN 9781849698870, [cit. 2016-05-05].

[76] Westfall, J.; Augusto, R.; Allen, G.: *Beginning Android Web Apps Development: Develop for Android using HTML5, CSS3, and JavaScript*. Apressus Series, Apress, 2012, ISBN 9781430239574, [cit. 2016-04-28].

[77] Wilson, J.: *Creating Dynamic UI with Android Fragments*. Packt Publishing, 2013, ISBN 9781783283101, [cit. 2016-05-05].

Příloha A

Seznam použitých zkratek

SDK Software Development Kit

DEX Dalvik Executable Format

UI User Interface

GPS Global Positioning System

2D Two-dimensional

3D Three-dimensional

FAT File Allocation Table

AR Augmented Reality

DEX Dalvik Executable Format

ANR Application is Not Responding

API Application User Interface

CSV Comma Separated Values

PNG Portable Network Graphics

LGPL Lesser General Public License

CD Compact Disk

Příloha B

Vývojářský manuál k použití knihovny

Manuál pro vývojáře

Zde se nachází manuál pro vývojáře využívající knihovny polohově zaměřené rozšířené reality implementované v rámci diplomové práce na FIT ČUT v Praze v roce 2016.

Popis a použití dílčích částí knihovny

Druhy fragmentů

Základním prvkem knihovny je fragment, který zobrazuje rozšířenou reality. Lze použít jeden ze tří níže uvedených.

ARBaseFragment

Abstraktní základní fragment, který zajišťuje funkci AR. Při dědění z tohoto fragmentu je třeba v metodě `onCreate()` volat `_core.setFocusAndRadar(classImplementingFocusAndRadarPositionInterface);`

GPSBaseFragment

Základní fragment implementující PositionProviderInterface a obsluhující zjišťování aktuální polohy pomocí Google Play Services API. Vhodné dědit v případě, že není nutné používat vlastního poskytovatele polohy, ale bude upravován vzhled AR.

DefaultARFragment

Hotový AR fragment implementující FocusAndRadarPositionInterface a poskytující algoritmy k vykreslení bodů na obrazovku a radar.

PositionProviderInterface

Rozhraní, které je nutné implementovat pro poskytování aktuální polohy. Definuje fuknce:

- `Location getActualLocation()`

FocusAndRadarPositionInterface

Rozhraní, které definuje metody volané při překreslování každého snímku. Metoda 'SetRadarPositions' je volána pouze, je-li v nastavení povoleno používat radar. Definuje fuknce:

- `void setFocusedAnnotations(List<Annotation> angleViewAnnotationsSortedFromFurthest, boolean gpsChanged)`
- `void setRadarPositions(List<Annotation> radarAnnotationsSortedFromFurthest, double azimuth)`

Loader

Abstraktní třída, která je potomkem AsyncTask a poskytuje wrapper na načítáním dat. Jediné, co je třeba při snaze napsat vlastní logiku pro načítání dat je implementace metody `loadPois`, která vrací data načtená do struktury KD-tree.

Příklad implementace načítání z CSV souboru.

```
    @Override
    protected KDTree<Poi> loadPois(PoiLoaderArguments... params) throws Exception {
        CsvPoiLoaderArguments csvArguments = ((CsvPoiLoaderArguments)params[0]);
        KDTree<Poi> pois = parseCSV(csvArguments.getCsvInputStream(),csvArguments.delimiter);
        return pois;
    }
```

Parametrem zmíněné metody je pole prvků `PoiLoaderArguments`. Dědením z této třídy si lze vytvořit vlastní objekt s potřebnými argumenty, které jsou předány do metody pro načítání bodů ve třídě `Loader`.

Příklad implementace potomka třídy `LoaderArguments` je jednoduchý:

```
public class CsvPoiLoaderArguments extends PoiLoaderArguments{

    private Context context;
    private int rawResourceId = -1;

    public CsvPoiLoaderArguments(Context ctx, int rawResourceId) {
        this.context = ctx;
```

```
        this.rawResourceId = rawResourceId;
    }

    public InputStream getCsvInputStream() throws Exception {
        if(rawResourceId != -1) {
         return context.getResources().openRawResource(rawResourceId);
    }
}
```

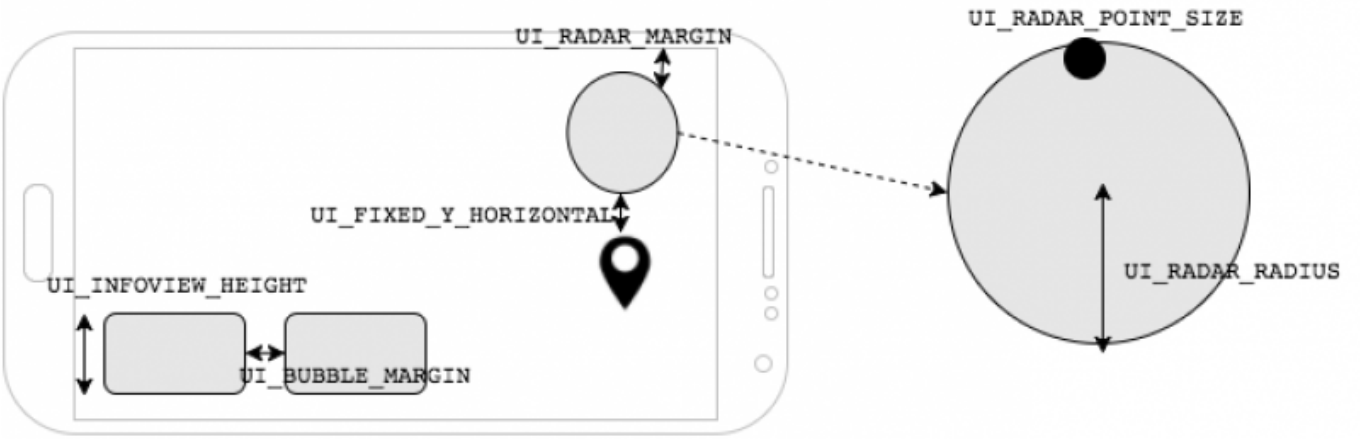
Filter

`Filter` je abstraktní třída definující jedinou metodu `public abstract float [] filter (SensorHistory sensorHistory, float [] lastFiltered)`, která zpracovává data naměřená senzory. Data jsou uložena v instanci třídy `SensorHistory` a lze k nim přistupovat skrze veřejnou metodu `public float [][] getDataFromNewestest()`. Parametr lastFiltered je hodnota vypočítaná při posledním proběhnutém výpočtu.

Nastavení

```
    //! Tag sloužící k vypisování informací do logu
    public static final String APP_TAG = "AR_library";

    //! Parametr určující, zdali má aplikace zobrazit radar
    public static boolean UI_ENABLE_RADAR = true;
    //! Počet snímků, po které zůstane zobrazena bublina pro daný bod
    public static int UI_STAY_FOCUSED_FRAMES = 40;
    //! Barva popředí prvků UI
    public static int UI_FOREGROUND_COLOR = Color.MAGENTA;
    //! Výška oblasti pro zobrazování bublin [px]
    public static int UI_INFOVIEW_HEIGHT = 200;
    //! Průhledná složka prvků UI <0,255>
    public static int UI_ALPHA = 160;
    //! Počet zobrazitelných bublin
    public static int UI_BUBBLE_NUMBER = 3;
    //! Velikost volného prostoru mezi bublinami [px]
    public static int UI_BUBBLE_MARGIN = 15;
    //! Odsazení značek od spodního okraje radaru při vertikální orientaci (může být záporné) [px]
    public static int UI_FIXED_Y_VERTICAL = 100;
    //! Odsazení značek od spodního okraje radaru při horizontální orientaci (může být záporné) [px]
    public static int UI_FIXED_Y_HORIZONTAL = -150;

    //! Krok odstupňování bonusové priority pro body uprostřed obrazovky
    public static int FOCUS_PRIV_MIDDLE_STEP = 15;
    //! Počet zvýhodněných bodů uprostřed obrazovky
    public static int FOCUS_PRIV_MIDDLE_COUNT = 6;
    //! Míra penalizace pro každý bod (K prioritě je připočítána výška zobrazené bitmapy, která závisí na vzdálenosté)
    public static int FOCUS_PRIV_BITMAP_PENALTY = 80;
    //! Horní mez generovaného náhodného čísla ovlivňujícího prioritu
    public static int FOCUS_PRIV_RANDOM_FACTOR = 100;
    //! Velikost zvýhodnění za nevybrání v jednom průchodu algoritmem
    public static int FOCUS_PRIV_UNSELECTED = 5;

    //! Poloměr radaru [px]
    public static int UI_RADAR_RADIUS = 110;
    //! Odsazení radaru od kraje obrazovky [px]
    public static int UI_RADAR_MARGIN = 20;
    //! Poloměr bodu radaru [px]
    public static int UI_RADAR_POINT_SIZE = 5;

    //! Je-li tento přepínač nastaven na true, aplikace použije na startu poslední známou lokaci (může vést k nepřesnsotem)
    public static boolean LOCATION_USE_LAST_POSITION = true;
    //! Interval aktualizace polohy [s]
    public static long LOCATION_INTERVAL = 15000;
    //! Nejrychlejší interval aktualizace polohy [s]
    public static long LOCATION_FASTEST_INTERVAL = 10000;
    //! Priorita použití při získávání aktualizací pozice [LocationRequest.PRIORITY_xxx]
    public static int LOCATION_PRIORITY = LocationRequest.PRIORITY_HIGH_ACCURACY;

    //! Počet bodů viditelných na radaru a zároveň maximální možný zobrazitelný počet
    public static int POI_SEE_POINTS = 30;

    //! Aktuální poskytovatel pozice (!!NUTNÉ NASTAVIT PŘI KAŽDÉM POUŽITÍ!!) [PositionProviderInterface]
    public static PositionProviderInterface POSITION_PROVIDER;
    //! Typ použitého filtru (FilterType.Custom značí použití nestandardního filtru, a ten je nutné nastavit) [FilterType]
    //! @see CUSTOM_FILTER
    public static FilterType FILTER_TYPE = FilterType.LOW_PASS_AVERAGE;
    //! Při zvolení typu filtru vlastní, musí být nastaven na instanci filtru [Filter]
    //! @see CUSTOM_FILTER
    public static Filter CUSTOM_FILTER;

    //! Interval navrhovaný pro aktualizaci hodnot senzorů [s]
    public static int SENSOR_DELAY = SensorManager.SENSOR_DELAY_FASTEST;
    //! Maximální počet hodnot v historii senzorových dat
    public static int SENSOR_DATA_HISTORY = 6;
    //! Určuje, zdali se histori senzorových dat po vyzvednutí resetuje
    public static boolean SENSOR_DATA_HISTORY_RESET = true;
```

Některé konstanty spojené se vzhledem UI jsou názorně zobrazeny na obrázku.



Příklad použití

AndroidManifest.xml

Základní použití knihovny je velmi jednoduché. Stačí v manifestu nastavit `android:screenOrientation` na hodnotu podle požadované orientace a zaregistrovat aktivitu, která bude zobrazovat AR.

```
<manifest package="com.prudek.augmentedreality"
          xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.prudek.augmentedreality.demo.MainActivity"
                  android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen"
                  android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

MainActivity.java

Aktivit, která chce zobrazit rozšířenou realitu jednoduše vytvoří nový Fragment pomocí jeho funkce `newInstance` a jako parametry zadá potomky tříd `PoiLoader` a `PoiLoaderArguments`, jejichž použití bylo vysvětleno výše.

Před samotným použitím fragmentu lze ovlivnit nastavení knihovny. Všechny hodnoty jsou ve výchozím nastavení zvoleny tak, aby nebylo nutné je nijak přenastavovat, a přesto fungovala AR. Jediné co je třeba je nastavit poskytovatele lokace voláním `Settings.POSITION_PROVIDER = classImplementingPositionProviderInterface` (pro případ, že by chtěl vývojář použít vlastní zdroj společně s výchozím fragmentem).

Che-li vývojář nastavit Immersive Full-Screen Mode, učiní tak přepsáním metody `onWindowFocusChanged` a doplněním kódu.

```
if (hasFocus) Settings.immersiveSticky(this);
```

```
public class MainActivity extends FragmentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_debug);

        CsvPoiLoader loader= new CsvPoiLoader(this);
        CsvPoiLoaderArguments args = new CsvPoiLoaderArguments(this, R.raw.poicz);

        DefaultARFragment fragment =  DefaultARFragment.newInstance(loader, args);

        Settings.POSITION_PROVIDER = fragment;
        Settings.POI_SEE_POINTS = 30;

        if (null == savedInstanceState) {
            getFragmentManager().beginTransaction()
                    .add(R.id.container, fragment)
                    .commit();
        }
    }
```

```
@Override
public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    if (hasFocus) Settings.immersiveSticky(this);
}
```

Autor

- Bc. Daniel Průdek 2016

„O chytré ženské je nouze. Konečně o chytré mužské zrovna tak.“ Jan Werich

Příloha C

Formulář pro testování míry použitelnosti aplikace

Proveďte scénáře podle pokynů a následně ohodnoťte práci s aplikací na stupnici od 1 do 10, kde 10 značí maximálně kladnou odpověď na otázku (odpovídá slovnímu vyjádření *rozhodně ano*). Pro dosažení nejlepších výsledků zapněte zjišťování polohy, WiFi a připojte se k Internetu (jakýmkoliv způsobem). Toto nastavení zajišťuje maximální přesnost polohy. Testování je prováděno anonymně, zapište prosím jen věk a pohlaví.

Scénáře

Správnost zobrazení bodů

Po dobu jedné minuty analyzujte okolí pomocí aplikace. Snažte se najít co nejvíce vám známých míst a zhodnoťte, zdali aplikace ukazuje místa ve správném směru.

Praktická užitečnost aplikace

Zvolte místo, o kterém víte, že je obsaženo v testovacích datech. Pokuste se toto místo pomocí aplikace najít a zobrazit informační bublinu. Zjistěte přesnou vzdálenost a zapište jí. Opakujte 5krát na stejném místě, vždy přibližně 3 vteřiny po zapsání. O kolik nejvíce a o kolik průměrně se hodnoty liší?

Dotazník pro uživatelské testování

- **Správnost směrů** Ukazuje aplikace správné směry?
- **Vyhledání místa** Lze snadno vyhledat konkrétní místo?
- **Změny lokace** Myslíte si, na základě zapsaných vzdáleností, že zjišťování aktuální polohy funguje správně?

- **Vstřebání informace** Lze si jednoduše spojit bod s bublinou a vstřebat informaci dříve než zmizí?
- **Rovnoměrnost zobrazení** Myslíte si, že se body mění rovnoměrně a jsou časem zobrazeny rozšiřující informace u všech? (abstrahujte od faktu, že se body uprostřed scény záměrně zobrazují častěji než ostatní)
- **Stabilizace** Jste spokojen/a se stabilitou obrazu?
- **Plynulost** Shledáváte aplikaci dostatečně plynulou? (hodnoťte pozitivně, pokud nedocházelo ke zpomalení či trhaným reakcím)
- **Problikávání** Jste spokojeni s prodlevou u změn aktuálně vyznačených? (k negativnímu hodnocení přispívá příliš častá změna vybraných bodů, díky čemuž dochází k problikávání a nečitelnosti informací)
- **Rozložení prvků** Hodnotíte kladně rozložení, barvu a pozici prvků na obrazovce?
- **Více řádek** Preferoval/a byste zobrazení bublin ve více řádcích?
- **Lomené čáry** Vnímáte pozitivně lomené čáry v aplikaci?
- **Radar** Pozoroval/a jste při používání ve velké míře radar?
- **Celkové hodnocení** Jaký je váš celkový dojem z aplikace?
- **Další** Stručně sepište vaše dojmy z používání aplikace a případné návrhy na vylepšení.

Děkuji za Váš čas věnovaný tomuto průzkumu.

Příloha D

Tabulka s výsledky uživatelského testování

Hodnocení výsledné aplikace uživateli v rámci testování se nachází na obrázku D.1.

Pohlaví	Věk	Správnost směrů	Vyhledání místa	Změny lokace	Vstřebání informace	Rovnoměrnost zobrazení	Stabilizace	Plynulost	Probliknávání	Rozložení prvků	Více řádek	Lomené čáry	Radar	Celkové hodnocení
muž	24	9	4	7	5	8	7	9	1	8	ne	3	4	6
muž	16	8	4	7	7	8	3	7	4	7	ne	7	8	8
muž	21	6	7	6	4	6	5	6	5	6	ano	5	9	6
žena	42	10	8	10	10	8	7	10	10	8	ne	10	1	10
muž	54	9	5	8	8	8	8	8	7	5	ne	8	7	9
muž	35	10	9	5	8	8	6	7	6	9	ne	8	10	8
žena	25	9	6	8	7	7	4	7	4	9	ne	8	3	7
žena	49	8	8	10	9	9	7	9	5	10	ne	9	3	9
muž	24	7	3	7	6	7	4	8	2	6	ano	6	8	6
Průměr	30% žen	8,44	6,00	7,56	7,11	7,67	5,67	7,89	4,89	7,56	80% ne	7,11	5,89	7,67

Obrázek D.1: Výsledky provedeného uživatelského testování

Příloha E

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresář se spustitelnou formou implementace
- `doc` ..... dokumentace ke knihovně
  - `manual` ..... vývojářský manál
    - `manual.pdf` ..... Vývojářský manuál ve formátu PDF
    - `manual.html` ..... Vývojářský manuál ve formátu DocuWiki
  - `doxygen` ..... vývojářský manál
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF