Univerzita Hradec Králové
Fakulta informatiky a managementu
Název katedry

Implementace vybrané grafické knihovny pro geolokační mobilní hru

Diplomová práce

Autor: Bc. Leon Vojtěch
Studijní obor: Aplikovaná informatika a matematika

Vedoucí práce: Ing. Pavel Kříž, Ph.D.

Hradec Králové Srpen 2019

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 11.8.2019 Leon Vojtěch

Poděkování:

Děkuji vedoucímu diplomové práce Ing. Pavlovi Křížovi Ph.D. za metodické vedení práce, nápady a užitečné rady, které mi dával v průběhu tvorby této práce. Také děkuji všem, kteří mě v průběhu tvorby práce podporovali.

Anotace

Tato práce se zabývá implementací grafického frameworku pro mobilní hru, ve které hráči chodí po budově FIM a vykonávají akce, během kterých jsou získávány informace o okolních bezdrátových sítích. Výsledná aplikace umožňuje určit polohu hráče a zobrazit polohu ostatních aktivních hráčů. Vykonávané akce využívají framework pro rozšířenou realitu.

Obsah této práce je rozdělen na několik částí. První část se zabývá rešerší existujících frameworků a knihoven, které je možné pro tuto práci použít. V druhé části je provedena analýza předešlého řešení, které se zabývá podobnou tématikou a následně popis požadovaných funkcí aplikace. V další části je popsán návrh a implementace daných funkcí. Na závěr je popsáno testování výsledné aplikace a zhodnoceny výsledky aplikace.

Annotation

Title: Implementation of Particular Graphics Library into a Geolocation Mobile Game

This thesis deals with implementation of graphical framework for mobile game, in which players walk around FIM building and perform actions during which information about surrounding wireless networks is gathered. The resulting application is able to determine the position of the player and display the position of other active players. The performed actions use the augmented reality framework.

The content of this thesis is divided into several parts. The first part deals with the research of existing frameworks and libraries that can be used for this thesis. The second part analyzes the previous solution, which deals with a similar topic and then it describes the required functions of the application. The next part describes the design and implementation of the functions. At the end the testing of the resulting application is described, and the results of the application are evaluated.

Obsah

Seznam obrázků

Seznam ukázek kódů

Seznam zkratek a pojmů

AAR – Android Archive povinně obsahující např. AndroidManifest

APK – Android Package, formát používaný pro distribuci a instalaci Android aplikací

BLE – Bluetooth Low Energy, neboli iBeacon, zařízení vysílající Bluetooth signály

Bluetooth – standard pro bezdrátovou komunikaci

FBX – Filmbox, formát pro ukládání 3D objektů a videí

Fingerprint – otisk okolních Wi-Fi a Bluetooth signálů

GPS – globální polohový systém provozovaný Ministerstvem obrany Spojených států amerických

HTML – Hypertext Markup Language, jazyk pro tvorbu webových stránek

JAR – Java Archive, obsahuje převážně Java třídy, textové soubory a obrázky

JSON – JavaScript Object Notation, datový formát, dobře čitelný lidmi i stroji

PNG – Portable Network Graphics, grafický formát pro rastrovou grafiku

SVG – Scalable Vector Graphics, formát pro ukládání vektorové grafiky

Wi-Fi – skupina standardů popisující bezdrátovou komunikaci v počítačových sítích

XML – Extensible Markup Language, jazyk určený pro výměnu dat

1 Úvod

V dnešní době chytrých zařízení není problém určit přibližnou polohu uživatele pomocí systému GPS. Tento systém využívá k určení polohy signál z družic. Problém u GPS nastává ve chvíli, kdy není přímá viditelnost na družice. Budovy, doly, tunely, ale i husté lesy blokují signál natolik, že určení polohy není možné a je tak nutné využít jiných metod. Jak ale určit polohu v budově, kde není možné použít GPS? Dnes se již v každé budově, kde se denně nachází stovky lidí, nachází vysílače Wi-Fi, které vysílají signály do okolí. Pokud je vysílačů dostatek a existuje databáze s otisky okolních vysílačů v konkrétních místech, lze určit polohu zařízení i v budově.

Tvorba takové databáze je ale časově náročná, proto vznikají aplikace, které posílají uživatele na konkrétní místa, kde je nutné dané otisky vytvořit. Tímto problémem se zabývala i práce na téma Gamifikace sběru fingerprintů bezdrátových sítí. (Vojtěch, 2016) Autor této práce vytvořil aplikaci Game of Flags pro chytrá zařízení s operačním systémem Android, kde mezi sebou válčily 2 frakce o nadvládu nad vlajkami. Aplikace zobrazovala mapu s polohou vlajek a při zabírání vlajky byla zobrazena animace. Během zabírání byly získány informace o síle signálů vysílačů a výsledný otisk poslán na server.

V této práci bude původní hra re-implementována. Důraz bude kladen na interaktivní mapu, na které budou zobrazeny herní předměty. Během útočení budou nadále získávány otisky, které budou ukládány na server. Aplikace také umožní zobrazit polohu uživatele na mapě pomocí vybraného algoritmu.

Her využívajících polohu zařízení vznikla celá řada. Mezi nejznámější tituly patří Ingress a Pokemon GO od firmy Niantic. Obě hry využívají upravené mapy světa, na které zobrazují herní objekty. Využívají klasické určování polohy a lze je tedy hrát pouze ve venkovních prostorech. Pokemon GO se stalo oblíbeným i díky rozšířené realitě, kdy je možné při dostatečném přiblížení k místu na mapě daného pokémona v reálném světě zobrazit a chytit ho do pokéballu.

2 Cíl práce

Cílem této práce je provést rešerši existujících grafických knihoven a frameworků pro Android. Zaměřit se na takové, které by umožnily vývoj hry ve stylu Pokémon Go nebo Ingress, kde je hlavním prvkem herní mapa s polohou hráče a dalšími herními předměty. S pomocí vybraného řešení (případě více) re-implementovat a rozšířit původní hru Game of Flags určenou pro sběr rádiových fingerprintů. Výsledná aplikace se bude skládat z části mobilní, která umožní určit polohu hráče a získávat dané fingerprinty a z části serverové, kam se budou fingerprinty ukládat.

3 Přehled existujících frameworků

Tato kapitola se zabývá rešerší existujících frameworků. Cílem je popsat takové, které umožní tvorbu hry pro mobilní platformu Android, kde je hlavním prvkem mapa, na kterou lze umístit objekty a s kterými je možné interagovat. Zároveň je důležité, aby framework podporoval způsob získávání dat z dostupných Wi-Fi a Bluetooth senzorů.

3.1 LibGDX

LibGDX je jedním z nejznámějších open-source multiplatformních frameworků pro tvorbu převážně 2D her. Vývoj probíhá v jazyce Java. Většina aplikací určených pro více platforem obsahuje hlavní část kódu, která je pro všechny platformy stejná. Aby se dala tato část vyvíjet efektivně, podporuje LibGDX při testování na desktopu využití vlastnosti Java Virtual Machine (JVM) nazývané Code Hotswapping, díky které jsou změny v kódu ihned viditelné v aplikaci. (Zechner, 2013)

Pro vytvoření projektu s LibGDX je možné využít nástroj, který automaticky vygeneruje strukturu projektu. Po nastavení základních informací o projektu a vybrání platforem, na které bude aplikace vyvíjena, lze vybrat i rozšíření projektu, například fyziku pro chování objektů nebo umělou inteligenci. Rozšíření lze dodávat i dodatečně. Pro editaci projektu umožňuje LibGDX použití všech oblíbených vývojových prostředí. Mezi nejznámější patří Intellij IDEA. (Skupnik, 2018)

3.1.1 Historie

Vývoj frameworku začal v roce 2009 pod názvem Android Effects (AFX). Byl určen pouze pro vývoj na platformu Android, ale nahrávání změn přímo do zařízení bylo zdlouhavé, proto byl AFX upraven pro fungování na desktopu. V roce 2010 byly zdrojové kódy nahrány na Google Code pod licencí GNU Lesser General Public License (LGPL) a název změněn na LibGDX. Po přidání podpory Box2D (engine přidávající fyziku) se komunita začala rozrůstat a na její žádost byla licence změněna na Apache License 2.0. Dalším důležitým rokem se stal rok 2012, kdy přibyla podpora iOS, HTML5 s WebGL a především přesun kódů na Github. Postupně se

dolaďovali funkce, přesunula se wiki na Github a v roce 2014 byla oficiálně vydána verze 1.0. (Zechner, 2014)

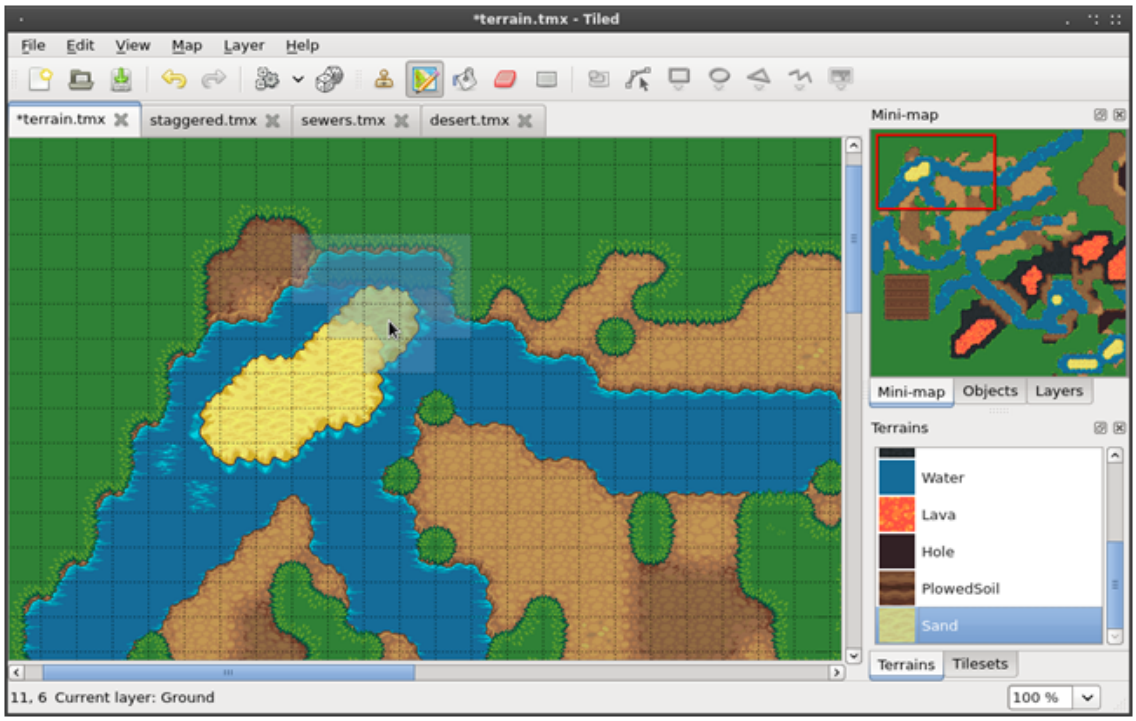
Vývoj frameworku probíhá dodnes, a ačkoliv novinky na oficiální stránce LibGDX uvádějí poslední verzi 1.9.8 z roku 2017, na Githubu lze dohledat aktuální verzi 1.9.9 vydanou koncem roku 2018.

3.1.2 Mapy a interakce s objekty

LibGDX podporuje všechny známé formáty 2D map. Toho je docíleno především díky tomu, že každá mapa obsahuje vrstvy, které obsahují objekty. Mezi nejznámější zástupce 2D map patří Tiled mapy.

3.1.2.1 Tiled Mapy

Tiled mapy jsou nejznámějšími a nejpoužívanějšími 2D mapami ve všech frameworcích a herních enginech. Nejpoužívanějším programem pro tvorbu map je Tiled.



Obr. 1 - Ukázka programu Tiled, zdroj: www.mapeditor.org

Tento program je zdarma dostupný a je určen pro Windows, Linux i Mac OS X. Použití je jednoduché, uživatel si nadefinuje terén, případně vybírá podobu

dlaždice přímo z tilesetu. Tileset je obrázek rozdělený na vybraný počet dlaždic. Program umožňuje vytvářet vrstvy a přidávat objekty, se kterými je poté možné interagovat. Mezi nejznámější použití patří vytvoření neviditelné kolizní vrstvy, ve které jsou vytvořeny objekty v oblastech kolizí a následně je v daném frameworku při pohybu objektu kolize ověřena. (Woitaschek, 2017)

Problém nastává, pokud je potřeba zjistit objekt pod myší nebo prstem na mobilním zařízení. LibGDX obsahuje pouze funkce na zjištění objektu ve vrstvě pomocí jména objektu nebo pomocí pozice objektu v seznamu objektů. Pro vytvoření interaktivního objektu by tedy bylo nutné vytvořit vlastní funkci.

3.1.2.2 3D objekty

Ačkoliv je LibGDX určeno převážně pro 2D tvorbu, obsahuje od roku 2013 možnost tvorby i 3D her. Jelikož je LibGDX pouze framework, neobsahuje žádný grafický editor pro usnadnění tvorby 3D objektů. Objekty lze tvořit přímo psaním kódu, ale to je vhodné jen pro základní objekty. Naštěstí lze použít objekty vytvořené v externích aplikacích. Vhodným programem pro tvorbu 3D modelů včetně animací je Blender. Blender je open-source program vyvíjený od roku 1995. Je dostupný pro všechny známé desktopové systémy. Umožňuje exportovat modely do několika formátů, oficiální dokumentace LibGDX doporučuje exportovat výsledné modely včetně animací do formátu FBX. Tento formát ale není možné přímo použít ve vytvořené aplikaci, je nutné jej nejprve změnit do podporovaného formátu. LibGDX podporuje 2 druhy formátů objektů. Prvním je G3DJ, ve kterém jsou data uložena v JSON formátu a druhým je G3DB, který má data uložena v binárním kódu. Tyto formáty ale mohou obsahovat maximálně 32767 vrcholů, takže zobrazení detailního objektu není možné. (SeanFelipe, 2019)

3.1.3 Získávání dat ze senzorů

LibGDX obsahuje metody pro práci se senzory, které se nejčastěji u her používají. Jedná se o akcelerometr, kompas, vibrátor a gyroskop. Pro získání dat z Wi-Fi a Bluetooth senzorů neexistují žádné připravené funkce. Vzhledem k tomu, že se tato práce zabývá tvorbou aplikace na mobilní zařízení s operačním systémem

Android, lze využít návody určené přímo pro Android a naimplementovat získávání dat přímo v LibGDX, bez nutnosti použití jiných pluginů.

3.1.4 Souhrn

LibGDX je zdarma dostupný framework pro vývoj her v Javě na různá zařízení. Je vyvíjen komunitou lidí, do které se může každý zapojit. Zaměřuje se především na tvorbu 2D her, pro které je možné tvořit mapy v externích editorech a následně s nimi jednoduše pracovat. Existuje i podpora 3D, která s určitými omezeními umožňuje používání objektů vytvořených v jiných aplikacích. Získávání dat ze senzorů na mobilních zařízeních s operačním systémem Android sice není implementováno, nicméně díky tomu, že vývoj probíhá v Javě, lze danou logiku jednoduše vytvořit.

3.2 Unity

Unity, někdy nazýváno Unity3D, je multiplatformní herní engine vyvíjený společností Unity Technologies. Jak již název napovídá, je primárně určen pro vývoj 3D her a aplikací, ale umožňuje vývoj i 2D her. Při vytváření nového projektu lze vybrat, zdali bude výsledná aplikace 2D nebo 3D. Následně je grafické rozhraní uzpůsobeno dané volbě. Ačkoliv je možné vytvořit jednoduché aplikace pouze za použití grafického editoru, vyvíjet složitější logiku aplikace lze v jazyku C# v předinstalovaném Visual Studiu od firmy Microsoft. (Unity, 2019a)

3.2.1 Historie

Vývoj Unity začal v roce 2002 kdy se Nicholas Francis a Joachim Ante, oba pracující na vlastním herním enginu pro Mac OS X, rozhodli pro vývoj společného enginu. Následně se k nim přidal David Helgason. Svůj tým pojmenovali Over the Edge Entertainment. (John, 2019) Po dvou letech vývoje vydali první hru s názvem Gooballs, která jim pomohla vyladit chyby v uživatelském prostředí. V roce 2005 byla vydána první verze Unity. Následovalo přidání podpory pro Microsoft Windows a internetové prohlížeče. V roce 2007 byla vydána verze Unity 2.0, která přidala mnoho nových funkcí, například engine pro vývoj terénu. Zlomovým rokem se stal rok 2008, kdy společnost Apple vytvořila pro své chytré mobilní telefony obchod

s aplikacemi. Unity se stalo prvním enginem umožňujícím vytvářet aplikace pro iPhone. To zvýšilo zájem lidí. (Brodkin, 2013) V roce 2010 byla vydána verze Unity 3.0 s rozšiřujícími grafickými možnostmi pro stolní počítače. V roce 2012 následovala verze 4.0 přidávající podporu DirectX 11 a Adobe Flash. Verze 5.0 byla vydána v roce 2015. Následně došlo ke změně číslování verzí a od roku 2017 jsou verze číslovány podle roku. K červenci 2019 je aktuální verze Unity 2019.1.

Ačkoliv je Unity napsáno v jazyku C++, pro programování se využívá jazyk C#. V minulosti byly také využívány jazyky JavaScript a Boo. (Brodkin, 2013) Pro vývoj kódu bylo využíváno upravené open-source prostředí MonoDevelop-Unity. To se ale změnilo verzí Unity 2018.1, kdy bylo MonoDevelop-Unity nahrazeno Visual Studiem. (Unity, 2019a)

3.2.2 Licence

Unity nabízí 3 druhy licencí v závislosti na potřebách uživatele a ziscích. Unity Personal je základní verze Unity, která je zdarma dostupná. Umožňuje vývoj na všechny platformy, ale roční zisk uživatele nebo společnosti nesmí překročit hranici 100 tisíc dolarů. Zároveň není možné vypnout úvodní logo Unity, ale je možné provést drobné úpravy, např. v animaci.

Druhou verzí je Unity Plus, která stojí 35 dolarů měsíčně. Tato verze oproti verzi Personal umožňuje mít roční zisky do 200 tisíc dolarů. Úvodní logo je možné zcela vypnout, případně upravit na logo vlastní. Velkou výhodou je přístup k online lekcím s experty. Tato licence přidává analytické nástroje pro zjištění chování uživatelů v aplikaci. Zároveň nabízí 25 GB místa v cloudu pro nahrání projektů.

Poslední verzí je Unity Pro, která stojí 125 dolarů měsíčně. Tato verze nemá žádné limity zisků uživatele nebo společnosti. Oproti verzi Plus má plnou podporu specialistů zodpovídajících otázky v reálném čase. Tuto verzi využívají přední společnosti na vývoj her. Mezi nejznámější patří Ubisoft nebo Electronic Arts.

Unity nevyžaduje žádný podíl ze zisku a práva k projektu vždy patří autorovi. Pokud se uživatel nebo společnost rozhodnou ukončit předplatné Unity, nesmí nadále vyvíjet již vytvořené aplikace ve verzi Personal. Pokud ale platí licenci alespoň 2 roky, lze si aktuální verzi Unity ponechat a po ukončení předplatného

získá ještě další 3 aktualizace. Ztratí ale výhody, které v daném předplatném měl. (Downie, 2016)

3.2.3 Asset store

Asset store je prostor ke sdílení materiálů a vylepšení pro Unity. Nabízí více jak 50 tisíc položek, které je možné vyhledávat podle názvu nebo kategorií. (Ewald, 2019) Položky stojí obvykle několik dolarů, ale je možné dohledat i položky, které jsou zdarma ke stažení. Po zakoupení jsou položky uloženy v profilu uživatele a lze je přímo přidat do aplikace.

3.2.4 Mapy a interakce s objekty

Ačkoliv je Unity zaměřeno na vývoj 3D her a aplikací, lze v něm vyvíjet i 2D hry. V roce 2017 byl přidán vlastní editor pro tvorbu tile map, který funguje na stejném principu jako program Tiled. Unity nemá podporu 2D map vytvořených v externích programech, ale v Asset store je možné najít rozšíření do unity, které umožní přidat vytvořené mapy do Unity, včetně převodu vrstev a tilesetů do editoru.

3.2.4.1 3D objekty a terén

Unity umožňuje vytvářet jednoduché 3D objekty přímo v editoru, ale upravit lze pouze měřítko objektu podle jednotlivých os. Mezi základní objekty patří například kostka, koule nebo terén. Pro tvorbu složitějších 3D objektů je tedy nutné využít externích programů. Doporučeným programem je již zmiňovaný Blender. Pokud je tento program nainstalován, umožňuje Unity automaticky převádět objekty z Blenderu, včetně animací.

Pro vybírání objektů lze využít metodu raycasting. Raycasting, neboli metoda vržení paprsku, funguje na jednoduchém principu vržení paprsku určitým směrem. Pokud paprsek zasáhne objekt, je vykonána určená akce. Pokud se v projektu nachází více objektů, lze je rozdělit do vrstev. Poté je možné vybírat objekty jen z dané vrstvy. (Unity, 2019b)

Pro tvorbu terénu má Unity vlastní editační nástroj. Tento nástroj obsahuje 5 hlavních možností pro editaci terénu. První možností je přidávat sousední terény. Druhá možnost umožňuje měnit členitost terénu a přidávat texturu. Třetí možností

je přidávání stromů. Unity nemá žádné předpřipravené modely stromů, proto je nutné modely nejdříve přidat do projektu. Čtvrtá možnost slouží pro přidávání detailů do terénu. Zde je také nutné předem připravit texturu trávy nebo menší objekty, které budou přidávány do terénu. Poslední možností je nastavení terénu. Například lze nastavit rychlost a sílu větru, velikost terénu nebo vzdálenost od kamery, do jaké budou objekty vykreslovány. Toto nastavení společně s nastavením rozlišení detailu má velký vliv na zatížení zařízení. (Unity, 2019c)

3.2.5 Získávání dat ze senzorů

Unity, podobně jako LibGDX, má podporu herních senzorů. Pro získání informací od okolních Wi-Fi a Bluetooth vysílačů je nutné využít jiný způsob. Existují 2 možnosti, jak data získat. První možností je vytvoření JAR (Java Archive) nebo AAR (Android Archive) pluginu, v kterém bude získávání implementováno. Následně je možné volat vytvořené Java třídy z C# kódu. Druhou možností je vytvořit kód pro získání dat přímo v C#. To ale není vhodné pro složitější logiku, protože se kód stává méně přehledným.

Pro volání Java tříd a objektů existují v Unity třídy AndroidJavaClass a AndroidJavaObject. AndroidJavaClass je reprezentací java.lang.Class a AndroidJavaObject je reprezentací java.lang.Object. (Agrawal, 2018)

```
private void ShowToast() {
        AndroidJavaClass toastClass = new
             AndroidJavaClass("android.widget.Toast");
        object[] toastParams = new object[3];
        AndroidJavaClass unityActivity = new
             AndroidJavaClass("com.unity3d.player.UnityPlayer");
        toastParams[0] =
unityActivity.GetStatic<AndroidJavaObject>("currentActivity");
        toastParams[1] = "This is a Toast";
        toastParams[2] = toastClass.GetStatic<int>("LENGTH_LONG");
        AndroidJavaObject toastObject =
toastClass.CallStatic<AndroidJavaObject>("makeText", toastParams);
        toastObject.Call("show");
    }
```

Ukázka kódu 1 - C#, zobrazení toastu v Android aplikaci, zdroj: medium.com

Použití AndroidJavaClass a AndroidJavaObject pro zobrazení jednoduchého textu pomocí toastu je znázorněno v ukázce kódu 1. Stejný kód napsaný v Javě je zobrazen v ukázce kódu 2.

```
private void showToast()
{
    Context context = getApplicationContext();
    CharSequence text = "This is a toast";
    int duration = Toast.LENGTH_SHORT;

    Toast toast = Toast.makeText(context, text, duration);
    toast.show();
}
```

Ukázka kódu 2 – Java, zobrazení toastu, zdroj: medium.com

3.2.6 Kompilace a sestavení aplikace na Android

Unity umožňuje po nadefinování základních informací o projektu a nastavení build parametrů vytvořit Android application package (APK). APK slouží k instalaci aplikací na operační systém Android. Pokud je zařízení s tímto operačním systémem připojeno k počítači, je možné nahrát výslednou aplikaci přímo do zařízení, bez nutnosti ruční instalace. (Unity, 2019d)

APK ale není nutné vytvořit přímo v Unity. Existuje možnost vyexportování projektu do vybrané složky. Následně je možné projekt otevřít v oblíbeném vývojovém prostředí, například v již zmíněné Intellij IDEA. Poté lze provést úpravy projektu a následně vytvořit APK a nainstalovat aplikaci do zařízení. (Joe, 2019)

3.2.7 Souhrn

Unity je multiplatformní herní engine umožňující vývoj 2D i 3D aplikací. Má 3 druhy licencí, které se liší především podporou od profesionálů a limity možných zisků z aplikací. Vývoj probíhá v grafickém prostředí doplněném skripty v jazyce C#. Unity má vlastní obchod s doplňky a materiály, které jsou vyvíjeny ostatními uživateli. Pro vývoj 2D her má vlastní systém tvorby tile map, ale lze v obchodě nalézt i doplněk pro použití map vytvořených v externích aplikacích. Unity umožňuje import 3D modelů různých formátů. Pro tvorbu 3D terénu obsahuje grafické prostředí vlastní nástroj. Získávání dat z Wi-Fi a Bluetooth senzorů není implementováno, ale lze si vytvořit vlastní skript pro získávání dat pomocí AndroidJavaClass a AndroidJavaObject, případně vytvořit Android plugin a získávat data voláním metod z pluginu.

4 Analýza serverové a mobilní aplikace

Cílem této práce je re-implementovat původní hru Game of Flags v grafickém frameworku nebo enginu a rozšířit ji o nové funkcionality. V této kapitole bude tedy provedena analýza předešlé aplikace a následně analýza nové aplikace.

4.1 Analýza předešlé aplikace

Aplikace Game of Flags vznikla v roce 2016 jako výsledek bakalářské práce na téma Gamifikace sběru fingerprintů bezdrátových sítí. Cílem této práce bylo vytvořit aplikaci, která posílala uživatele na předem určená místa, kde byly získány informace o okolních bezdrátových sítích. (Vojtěch, 2016) Aplikace byla rozdělena na mobilní a serverovou část.

4.1.1 Analýza serverové části

Server byl provozován v cloudovém prostředí OpenShift od společnosti Red Hat, který nabízel možnost vytvoření až 3 projektů zdarma. Projekty bývaly různého typu, nejčastěji se jednalo o webový server a databázi. Projekt využil kombinaci PHP s MySQL databází.

Pro webový server byl zvolen framework Nette. Důvodem bylo zajištění ochrany před možnými destrukčními útoky na databázi. Ačkoliv framework funguje na principu Model-View-Presenter, v projektu byl vytvořen pouze jediný Presenter zajišťující komunikaci mezi mobilní aplikací a databází. Pro ověření totožnosti uživatele byl využit Google Identity Toolkit (dále Gitkit). Gitkit sloužil pro přihlášení jak do webových, tak mobilních aplikací. Umožňoval jak klasické přihlášení pomocí e-mailu, jména a hesla, tak pomocí účtů od různých společností, například Google nebo Facebook. (Google, 2019a)

Databáze se skládala ze 4 tabulek – player, fraction, scan a flag. Tabulka player obsahovala token pro ověření hráčovy totožnosti, jméno hráče, odkaz na frakci, datum poslední změny frakce a počet zabraných vlajek. V tabulce fraction bylo uvedeno jméno a popis frakce. Tabulka scan obsahovala vytvořený otisk bezdrátových sítí, včetně času vytvoření, kdo otisk vytvořil, u jaké vlajky a jaké frakci vlajka patřila. Poslední tabulka obsahovala název vlajky, čas poslední zabrání, odkaz na hráče, který vlajku zabral, pozice vlajky, popis, QR kód a celkový počet zabrání.

4.1.2 Analýza mobilní části

Mobilní aplikace byla rozdělena na 4 části podle vytvořených aktivit. Jednalo se o přihlašování, mapu, snímání a nastavení.

4.1.2.1 Přihlašování

Jak již bylo zmíněno v serverové části, pro přihlašování byl vybrán Gitkit. Gitkit pro každého uživatele vytvořil unikátní token, kterým bylo možné se přihlásit. Aby nebylo nutné se při každém zapnutí aplikace znovu přihlašovat, byl token uložen pomocí rozhraní SharedPreferences, které ukládá data do souboru. Po přihlášení bylo ověřeno v databázi, zdali uživatel již existoval a bylo mu posláno jméno pro ověření. Pokud jméno odpovídalo základnímu jménu, jedno se o nového uživatele a musel si změnit jméno. Uživatel byl také při registraci automaticky přidělen do méně aktivní frakce.

4.1.2.2 Mapa a stav hry

Po přihlášení byla uživateli zobrazena mapa s vlajkami, jeho počet zabraných vlajek a bodový stav frakcí. Podoba této aktivity je zobrazena na obrázku mapy předešlé aplikace. Vybraná frakce byla zvýrazněna tečkou v barvě frakce. Zvolené patro bylo uloženo, takže po načtení této aktivity bylo automaticky vybráno.

Mapa byla zobrazena pomocí WebView, které umožňuje zobrazit webový obsah, ale nelze použít jako plnohodnotný prohlížeč. Velikost rozměrů obrázků pater se lišila, proto byla pro každé patro vytvořena jednoduchá HTML stránka obsahující SVG element s pozadím daného patra, který umožňuje vytvářet jednoduché grafické objekty. WebView umožňuje využít i JavaScript, pomocí kterého je vyřešena komunikace mezi aplikací a HTML stránkou. Oboustranná komunikace byla využita pro zobrazení bodů s polohou vlajek a následně pro zobrazení informací o vybrané vlajce. Informacemi byly název vlajky, frakce, které vlajka patřila, kdo a kdy ji zabral a jaký je stav vlajky. Stav oznamoval, zdali je možné danou vlajku zabrat, případně kdy bude zabrání možné.

Z této aktivity bylo možné zapnout aktivitu nastavení přes ozubené kolo a aktivitu snímání přes tlačítko Zaber vlajku.

Obr. 2- Mapa předešlé aplikace, zdroj: theses.cz

4.1.2.3 Aktivita snímání a ověření pozice

Cílem této práce bylo zajistit získávání dat z okolních Wi-Fi a Bluetooth vysílačů. Samotný sběr dat probíhá během animace zabírání vlajky. Animace byla vytvořena snižováním a zvyšováním viditelností dvou obrázků s vlajkami. Po dokončení snímání byl otisk uložen do lokální SQLite databáze. Jednalo se o zjednodušenou verzi SQL. Vedle informací o dostupných sítích byly uloženy i informace o vlajce, času pořízení a zařízení, včetně informací z dalších senzorů. Po úspěšném poslání na server byl lokální otisk vymazán.

Pro ověření pozice byly zvoleny QR kódy. Aby se zabránilo načítání cizích kódů, byly po zapnutí dané aktivity staženy obsahy všech kódů označujících pozici vlajky. Následně byl po naskenování kódu ověřen stav dané vlajky. Stavy existovaly 4 a určovaly, zdali uživatel může danou vlajku zabrat. Sloužily především k zamezení zabírání vlajky stejným uživatelem, případně k okamžité výměně mezi více hráči za účelem zisku bodů.

Pro ověření hráčovy pozice při zabírání vlajky byl využit také krokoměr, ale kvůli malé citlivosti byl vypnut. Z tohoto důvodu byla neustále ověřována viditelnost QR kódu. Pokud se uživatel vzdálil, zabírání bylo zrušeno a uživatel byl upozorněn. Po úspěšném zabrání vlajky byl uživatel přesměrován zpět na aktivitu s mapou.

4.1.2.4 Nastavení

Poslední aktivitou bylo nastavení. V nastavení bylo možné vypnout nebo zapnout notifikace, změnit frakci nebo se odhlásit. Po nainstalování aplikace byly notifikace vypnuty, ale pokud si je hráč zapnul, zůstaly aktivní i po restartu zařízení.

Aplikace využívala 2 druhy notifikací. První druh využíval polohu uživatele a zobrazoval notifikace pouze pokud se uživatel přiblížil k vybrané budově. Druhým druhem byly časové notifikace. Ve stanovenou dobu se ověřila dostupnost okolních Wi-Fi vysílačů a pokud se zjistila shoda s uloženými adresami vysílačů, byl zjištěn bodový stav frakcí. Pokud hráčova frakce prohrávala, byla notifikace zobrazena.

Další funkcí v nastavení byla změna frakce. Změnu bylo možné provést vždy minimálně týden po poslední změně frakce. Po změně frakce nebyl hráči vynulován bodový stav. Toto chování bylo vytvořeno z důvodu soutěživosti hráčů a prevence proti nadvládě jedné frakce.

Poslední funkcí bylo odhlášení z aplikace. Po odhlášení byl smazán token ze SharedPreferences a uživatel byl přesunut na přihlašovací aktivitu.

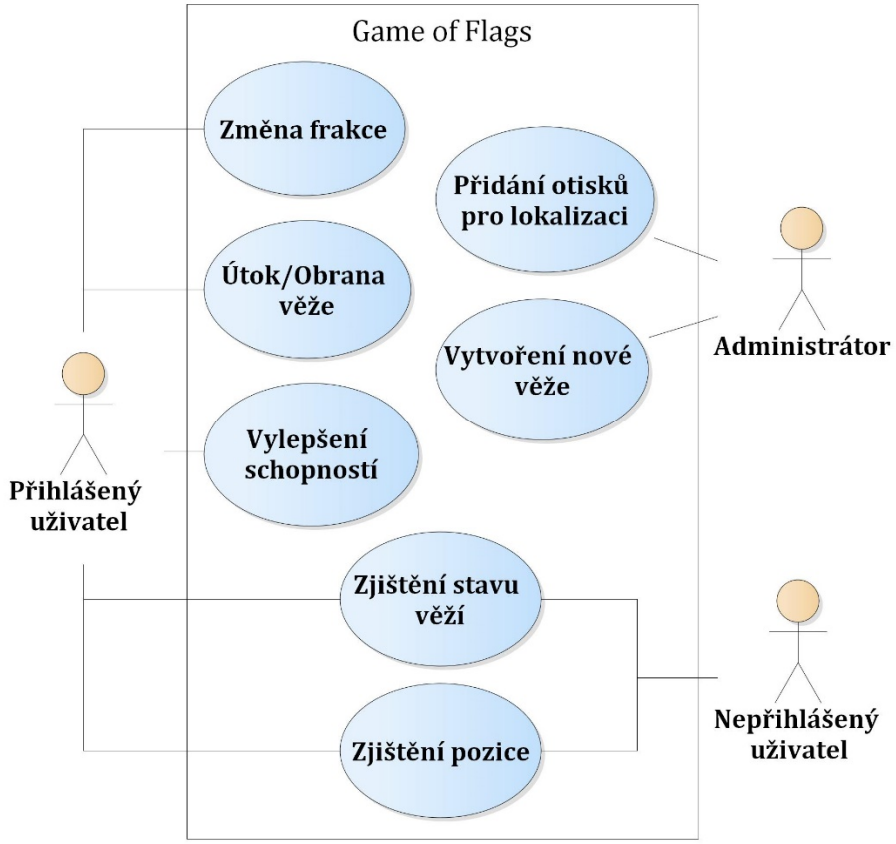
4.1.3 Shrnutí aplikace

Práce se zabývala gamifikací sběru fingerprintů bezdrátových sítí. Autor vytvořil aplikaci Game of Flags pro operační systém Android, ve které mezi sebou válčily 2 frakce o nadvládu nad vlajkami. Aplikace se skládala z mobilní a serverové části. Server sloužil pouze pro komunikaci s MySQL databází. Mobilní část se skládala ze 4 aktivit, hlavní aktivitou byla mapa s informacemi o frakcích a vlajkách. Mapa byla zobrazena jako obrázek, na kterém byly vytvořeny body označující vlajky. Pro ověření pozice při zabírání vlajek bylo využito QR kódů a zobrazena animace překreslujících se obrázků. Během zabírání byl vytvořen otisk bezdrátových sítí. Výsledný otisk včetně informací o zařízení byl následně poslán na server.

4.2 Požadovaná funkcionalita

Z analýzy předešlé aplikace bylo zjištěno, že první akcí každého nového uživatele bylo přihlášení do aplikace. Následovalo zobrazení mapy, kde uživatel netušil, kde se nachází a svoji polohu musel sám odhadnout. Při zabírání vlajky byla zobrazena jednoduchá animace s odpočtem zbývajícího času. Během zabírání byl vytvořen otisk okolních sítí a následně poslán na server. V nastavení bylo možné zapnout a vypnout notifikace, změnit frakci nebo se odhlásit.

Novým požadavkem na aplikaci je možnost použití aplikace i bez přihlášení. Uživatel po startu aplikace ihned uvidí herní mapu a bude mít možnost s ní interagovat. Vlajky budou nahrazeny věžemi z důvodu větších herních možností. Věže získají výdrž a úrovně podle oblíbenosti. Uživatel bude moci zjistit vlastní polohu přímo zobrazením na mapě. Přihlášený uživatel bude moci věže bránit nebo na ně útočit. Za každou provedenou akci získá zkušenosti, které mu zvýší úroveň a bude mít možnost vylepšit své schopnosti. Administrátor bude mít možnost přidávat otisky pro určení pozice, kontrolovat otisky v databázi a vytvářet věže.



Obr. 3 - Diagram použití aplikace, zdroj: vlastní

4.3 Analýza serverové aplikace

Požadavky na server se oproti minulé aplikaci příliš neliší. Hlavní částí serveru bude databáze, ve které budou uloženy informace o hráčích a věžích. Zároveň bude možné ukládat vytvořené otisky a získat ověřené otisky pro určení polohy uživatele. O ověření otisku se bude starat administrátor, který bude mít možnost vytvářet nové věže.

4.4 Analýza mobilní aplikace

Analýzu mobilní aplikace lze rozdělit do několika sekcí podle požadované funkcionality. Jedná se o analýzu přihlašování, frakcí, dostupných informací o hráči, analýzu mapy včetně způsobu zachování kompatibility s otisky z jiných aplikací, jak určit polohu uživatele a jak ověřit, že se nachází na určeném místě.

4.4.1 Přihlašování

V předešlé aplikaci byla pro přihlašování využita služba Gitkit. Nejnovější verze Google Identity Toolkitu byla vydána jako Google Cloud's Identity Platform a Firebase Authentication. Oba produkty nabízí možnost různých druhů přihlašování a službu pro obnovení zapomenutého hesla. (Google, 2019a)

Od nové aplikace se očekává podpora přihlašování pomocí sociální sítě Facebook a účtu Google, které vlastní každé zařízení s operačním systémem Android.

4.4.2 Frakce

Uživatelé aplikace budou patřit do jedné ze dvou frakcí. Aplikace by měla umožnit novému uživateli vybrat si požadovanou frakci a zároveň umožnit změnu frakce. Mělo by být nastaveno časové omezení změny frakce, aby nebylo možné příliš narušovat rovnováhu frakcí. Také by měl být zobrazen aktuální počet vlastněných věží danou frakcí.

4.4.3 Informace o hráči

Novou funkcí aplikace bude získávání zkušeností za opravu věže nebo útok na věž. Po dosažení určitého počtu zkušeností hráč postoupí na další úroveň a budou mu vylepšeny schopnosti. Zároveň získá bod na vylepšení vybrané schopnosti.

Uživatel bude mít možnost zobrazení informací o počtu zkušeností, dosažené úrovni, aktuální sílu útoku a hodnotu opravy věže.

4.4.4 Věže

Jak již bylo zmíněno, vlajky budou nahrazeny věžemi. Každá věž bude mít určitou výdrž. Pokud věži nezbude žádná výdrž, padne věž do vlastnictví druhé frakce. Věže budou mít vlastní systém úrovní, které jim zvýší počet bodů výdrže.

4.4.5 Mapy a interakce

Hlavní částí aplikace bude herní mapa. Z analýzy předešlé aplikace bylo zjištěno, že velikost plánků je pro každé patro rozdílná. Aby byla zachována kompatibilita s jinými aplikacemi využívajícími vytvořené otisky, je nutné zachovat rozměry plánků nebo výsledné souřadnice otisků přepočítávat.

Cílem aplikace je podobné použití mapy jako v Pokemon Go, kde je možné přiblížení nebo oddálení a rotace mapy s tím, že hráč je vždy uprostřed mapy a není možné se přesunout do jiné části. Nová aplikace by měla mít možnost volné kamery s možností zaměření hráče. Podobně jako ve zmíněné aplikaci bude pozice uživatele vypočítávána, nebude možné ji měnit.

Kromě zobrazení hráče bude možné na mapě vidět věže a ostatní hráče. Po kliknutí na věž budou zobrazeny informace o výdrži, úrovni, jaké frakci věž patří a jaký hráč ji naposledy obsadil. Zároveň bude možné zaútočit na věž nebo ji opravit, bude-li to možné. Informace o ostatních hráčích zobrazeny nebudou z důvodu možného přehlcení mapy interaktivními objekty.

4.4.6 Lokalizace uživatele

Jak již bylo v úvodu práce zmíněno, pomocí této aplikace bude možné určit přibližnou polohu uživatele v budově. Jelikož není možné využít GPS, budou pro lokalizaci využity signály z Wi-Fi vysílačů. Wi-Fi vysílače ale nejsou jedinou možností, jak určit polohu zařízení. Jelikož všechny telefony umí přijímat Bluetooth signály, lze využít i speciální zařízení iBeacon, nazývané také Bluetooth Low Energy (BLE). Jak je z názvu patrné, tato zařízení zvládnou vysílat signály až několik let za

použití malé baterie. Nízká spotřeba ale neznamená menší výkon, nové vysílače mají dokonce vyšší dosah než klasické Bluetooth. (Zhou, 2017)

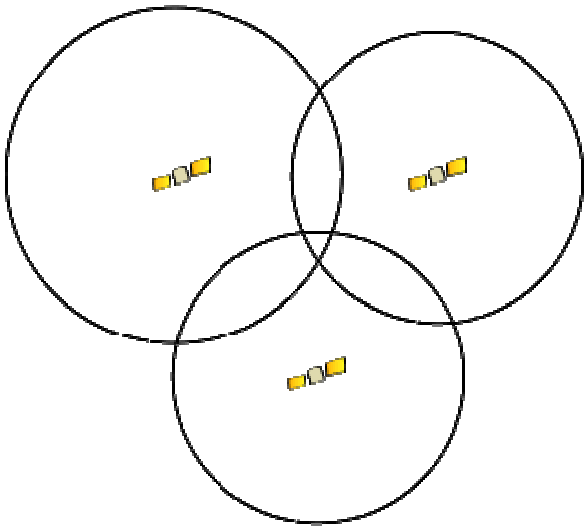
Pro určení polohy pomocí signálů z Wi-Fi a BLE vysílačů lze využít několik algoritmů. Prvním je využití multilaterace, kdy je potřeba znát polohu vysílačů a podle intenzity signálů se určí poloha. (Koo, 2011) Dalším algoritmem je algoritmus nejbližších sousedů (Li, 2016), kdy je nutné předem vytvořit databázi otisků sítí a následně najít nejpodobnější řešení nebo Particle Filter.

Polohu v uzavřeném objektu lze určit i pomocí jiných metod, které ale běžné chytré zařízení neumí zpracovat. Pro příklad lze uvést Time of Arrival (Time of Flight) využívaný v GPS, který vypočítává pozici podle času letu signálu od vysílače k přijímači, ale běžné Wi-Fi senzory nejsou schopny tuto hodnotu určit. Existují také akustické systémy nebo infrared systémy, které využívají paprsek světla, ale přesnost je pouze na místnost. (Dardari, 2015)

4.4.6.1 Multilaterace

Tento algoritmus je využívaný především pro GPS. Vysílač (v tomto případě družice) vysílá signál směrem k zemi s časovým otiskem a parametry dráhy družice. Přijímač poté porovná získaný čas a vypočte polohu družice. Tím získá kružnici, na které se jeho poloha nachází. Pro určení pozice je nutné získat pozici alespoň 3 družic, kde jejich průnik znamená polohu přijímače. Tato metoda se nazývá trilaterace a je znázorněna na obrázku 4 - trilaterace. V ideálním případě se získá 1 bod, kde se přijímač nachází, v reálné situaci je nutné ještě určit střed získané plochy. Multilaterace znamená použití více než 3 vysílačů, což vede k upřesnění pozice. (Gao, 2015)

Podobně jako u GPS, lze tuto metodu použít i u Wi-Fi a BLE vysílačů. Rozdíl je ten, že poloha vysílačů je známá a místo času se využívá síla signálu. Problém nastává u rušení signálu. Pokud by se tato metoda použila v otevřeném prostoru, byla by vypočtená poloha velice přesná. Bohužel v uzavřeném prostoru, kde se nachází mnoho překážek mezi vysílačem a přijímačem, dochází k poklesu signálu a vysílač, od kterého je získáván jen velmi slabý signál a očekává se vzdálenost desítek metrů, může být schován za zdí.



Obr. 4 - trilaterace, zdroj: commons.wikimedia.org

4.4.6.2 Algoritmus nejbližších sousedů

Druhým algoritmem je nearest neighbors (nejbližší sousedé), někdy také nazýván metodou otisků. Tato metoda nepotřebuje znát pozici vysílačů, ale je nutné nejprve vytvořit otisky okolních Wi-Fi a Bluetooth Low Energy signálů s definováním přesné pozice, kde byly otisky pořízeny. Po vytvoření dostatečně velkého množství otisků lze v daném prostředí vytvořit nový otisk, který se porovná s existujícími otisky a podle nejlepší shody je určena poloha zařízení. V ideálním případě se získá stejný otisk, jaký je pro danou pozici již uložený a výsledná pozice je přesně určena.

Poloha se ale nemusí nutně určit jen podle nejlepší shody. Existují různé varianty algoritmu, kdy se poloha určí podle více podobných otisků. První variantou je určení k-podobných sousedů, ze kterých se porovná pozice a výsledná pozice přijímače je ve středu sousedů. Druhou variantou je vážený průměr, kdy je polohám sousedů vypočtena i váha podle „vzdálenosti“ otisků. Poté je poloha přepočítána podle vah a je získána pozice. (Caso, 2015)

Problém u tohoto algoritmu nastává ve chvíli, kdy je do budovy nainstalován nový vysílač. Oblast, kam až signál z vysílače dosahuje, je nutné celou znovu projít a vytvořit nové otisky okolních sítí, jinak se snižuje kvalita vypočtené pozice.

4.4.7 Ověření pozice uživatele při snímání sítí

V předešlé aplikaci bylo pro ověření pozice uživatele využito QR kódů rozmístěných po budově. Při každém spuštění aktivity byly staženy všechny kódy, aby nebylo možné načíst cizí kód. Po načtení kódu se objevila animace překreslování vlajek obou frakcí s odpočtem oznamujícím zbývající čas do zabrání vlajky. Během animace bylo zapnuto snímání okolních sítí a výsledný otisk uložen.

V dnešní době vzniká stále více aplikací s rozšířenou realitou. Příkladem je již zmiňovaná hra Pokemon Go, kde je pokémon zobrazen v reálném světě. Pro vytvoření aplikace s rozšířenou realitou se v dnešní době používá především ARCore a Vuforia. Obě knihovny lze použít jak při vývoji v čisté aplikace pro Android, tak v Unity.

ARCore od společnosti Google umožňuje přidávat do reálného světa objekty a pracovat s nimi. Pro svou funkci potřebuje pouze fotoaparát a Android verze 7.0 nebo vyšší. (Google, 2019b)

Vuforia od společnosti PTC také umožňuje přidávat objekty do reálného světa. Objekty lze také zobrazovat na značkách, které jsou vytvořeny z nahraných obrázků v internetovém manažeru a uloženy do databáze, kterou lze poté stáhnout. Vuforia používá pro novější verze Androidu již zmíněný ARCore. Minimální verze Androidu pro použití Vuforie je 4.4. (Vuforia, 2019)

Cílem této aplikace bude nahrazení animace zabírání vlajky rozšířenou realitou. Během úspěšného zobrazení objektu bude zapnuto snímání signálů z okolních Wi-Fi a Bluetooth vysílačů. Výsledný otisk bude následně poslán na server. Také bude nutné ověřit skutečnou pozici zařízení například pomocí fotografie z rozšířené reality, aby bylo možné vytvořený otisk považovat za správný.

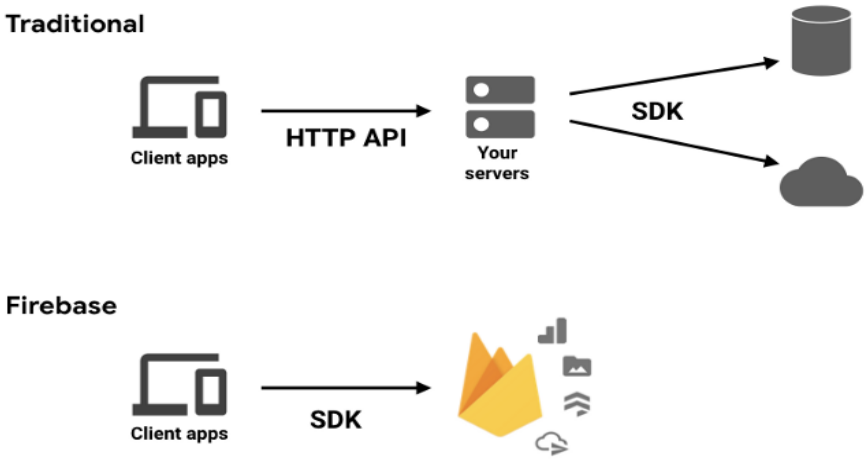
5 Návrh a implementace

V analýze bylo řečeno, že se nová aplikace bude skládat z části mobilní a z části serverové. Jelikož bude mapa vytvořena ve 3D a Unity je primárně určené na tvorbu 3D aplikací, bude pro vývoj aplikace zvolen tento herní engine. Jak již bylo v rešerši zmíněno, Unity má podporu pouze základních senzorů. Pro vytvoření otisku bezdrátových sítí a výpočet polohy zařízení bude tedy vytvořen Android plugin. Aby byla zajištěna podpora i starších zařízení, bude pro rozšířenou realitu využita Vuforie. Výsledná aplikace bude tedy určena pro operační systém Android verze 4.4 a vyšší z důvodu správné funkce rozšířené reality. Pro přihlášení bude využita služba Firebase Authentication, protože Firebase nabízí více služeb.

5.1 Firebase

„Firebase je platforma pro vývoj mobilních aplikací od společnosti Google, která vám pomáhá při tvoření, zlepšování a růstu aplikace.“ (Stevenson, 2018)

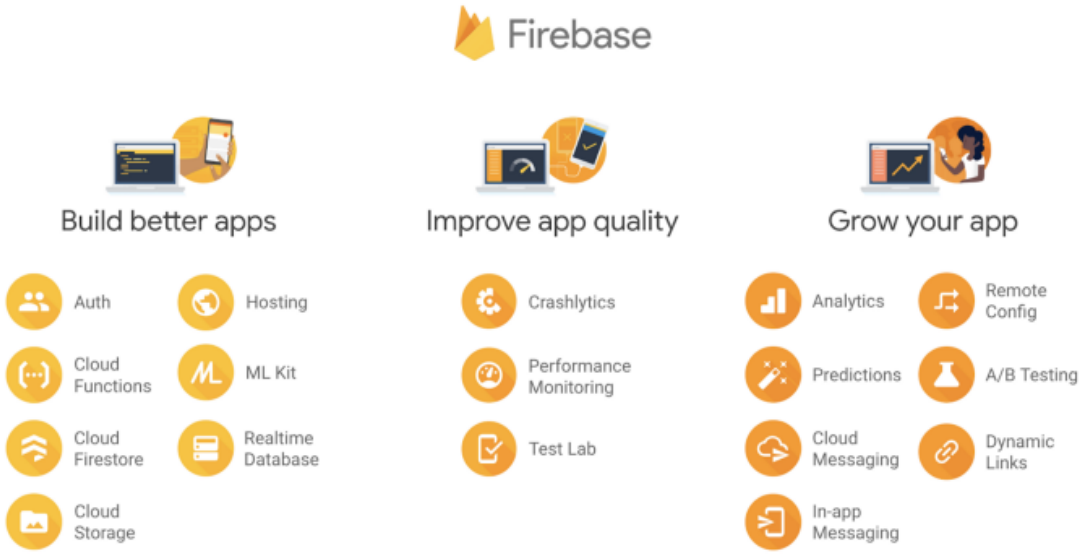
Firebase vznikla v roce 2011 jako NoSQL databáze. NoSQL databáze nevyužívají pro ukládání dat tabulková schémata. Formát je uzpůsoben projektu, ale obvykle se jedná o databázi typu klíč-hodnota. Tento typ využívá i Firebase pro svou Realtime databázi a data jsou ukládána do jediného JSON souboru. V roce 2014 byla společnost Firebase odkoupena společností Google a pojem Firebase se začal používat jako platforma obsahující více služeb. Nyní poskytuje Firebase 17 služeb a lze využít jako náhradu za server. Tento přístup výstižně popisuje obrázek č. 5.



Obr. 5 - způsob použití Firebase, zdroj: medium.com

5.1.1 Poskytované služby

Jak již bylo řečeno, Firebase poskytuje 17 služeb usnadňujících vývoj aplikací. Mnoho služeb je známých jako samostatné produkty od společnosti Google. Rozdělení služeb zobrazuje obrázek č. 6.



Obr. 6 - Seznam služeb Firebase, zdroj: medium.com

5.1.1.1 Přihlašování

Jak již bylo řečeno, Firebase Authentication navazuje na službu Gitkit. Poskytuje jednoduché přihlašování například pomocí jména a hesla, sítí Google, Facebook, Twitter nebo GitHub. Služba poskytuje komplexní zabezpečení, protože je vytvořena týmem vývojářů, kteří vyvinuli přihlašování Google Sign-in nebo manažera hesel v prohlížeči Chrome. (Firebase, 2019a)

5.1.1.2 Úložiště

Firebase poskytuje 3 služby pro ukládání dat. První službou je Cloud Storage, který slouží pro ukládání souborů. Cloud Firestore a Realtime Database jsou NoSQL databáze. Firestore je nová databáze určená pro mobilní vývoj. Nabízí rychlejší vyhledávání a lepší škálovatelnost oproti Realtime databázi. Firestore je také doporučen pro nové projekty. Obě databáze umožňují naslouchat novým změnám a posílat automaticky změny na připojená zařízení.

5.1.1.2.1 Realtime databáze

Jak již bylo zmíněno, tato databáze má data uložena v jediném velikém JSON souboru. Nevýhodou tohoto způsobu je ukládaní složitých objektů, které se poté hůře zpravují. Databáze má velmi nízkou odezvu a je uložena vždy v jediném datacentru. Cena je určena podle množství uložených a přenesených dat. Firebase nabízí prvních 10 GB přenesených dat v měsíci zdarma. (Firebase, 2019b)

5.1.1.2.2 Cloud Firestore

Tato databáze ukládá data jako kolekce dokumentů. Dokumenty se mohou skládat z dalších kolekcí a ty z dalších dokumentů. Struktura dokumentů je velmi podobná formátu JSON. Výhodou oproti Realtime databázi je jednoduchá organizace dokumentů při růstu databáze. Data jsou ukládána do více datacenter. Cena je určena podle množství uložených dat a vykonaných operací. Oproti Realtime databázi jsou operace počítány na dny. (Firebase, 2019b)

5.1.1.3 Shrnutí

Firebase je platforma usnadňující vývoj převážně mobilní aplikací. Nabízí mnoho služeb, které je možné v aplikacích kombinovat. Velká výhoda spočívá v absenci serveru, kdy je veškerá komunikace řízena přímo mezi zařízeními a platformou. Další výhodou je škálovatelnost služeb, kdy se služby automaticky přizpůsobují potřebám projektu až po určené limity uživatelem. Firebase lze využít pro tvorbu projektů dokonce zcela zdarma, pro malé projekty jsou limity dostačující.

5.2 Serverová část a Android plugin

Z analýzy vyplynulo, že server má obsahovat databázi pro uložení dat z mobilní aplikace a zajišťovat komunikaci mezi mobilní aplikací a databází. Právě pro tyto účely je určen Firebase, proto bude použit jako serverová část projektu. Bude využito služeb přihlašování, Cloud Storage, Realtime databáze a Cloud Firestore. Jelikož Firestore není možné použít s Unity, budou všechny služby použity v Android pluginu.

5.2.1 Přidání Firebase do Android pluginu

Pro práci s Firebase je nutné nejprve založit projekt ve Firebase konzoli dostupné v internetovém prohlížeči. Při vytváření projektu je podle názvu vytvořen unikátní identifikátor, který je použit v různých službách a po vytvoření již nelze změnit.

Po vytvoření projektu je nutné přidat Firebase do Androidí aplikace. Nejprve je nutné zadat aplikační identifikátor. Tento identifikátor slouží k rozlišení aplikací v zařízeních a v obchodě s aplikacemi. (Android, 2019a) Dále je možné zadat přezdívku, pod kterou bude aplikace viditelná v konzoli a otisk ladícího certifikátu SHA-1 zařízení, na kterém bude probíhat vývoj aplikace. Otisků je možné nastavit více, například pro podporu vývoje z více zařízení. SHA-1 otisk slouží také pro identifikaci zařízení při vydávání aplikací pro operační systém Android. Zde je vhodné jej vyplnit, protože bude potřebný pro zprovoznění přihlašování pomocí účtu Google. (Firebase, 2019c)

Po přiřazení aplikace do Firebase je možné získat konfigurační soubor pro zprovoznění Firebase v aplikaci. Soubor obsahuje informace o projektu, klientech a konfigurační verzi. V části o klientech jsou uloženy informace o aplikaci, zařízeních, na kterých probíhá vývoj, api klíč Googlu a další služby.

```
{
  "project_info": {
  "project_number": "513211408030",
  "firebase_url": "https://game-of-flags.firebaseio.com",
  "project_id": "game-of-flags",
  "storage_bucket": "game-of-flags.appspot.com"
 },
 "client": [...],
 "configuration_version": "1"
}
```

Ukázka kódu 3 - Konfigurační soubor google-services.json, zdroj: vlastní

Následně je nutné přidat Google Service plugin a závislosti Firebase služeb do Gradle souborů. Gradle je open-source nástroj pro automatizaci sestavení aplikací. (Gradle, 2019) Doporučenou závislostí je Firebase-core, která přidá analytika do aplikace. Po sestavení a spuštění aplikace na zařízení nebo emulátoru s přístupem k internetu je ověřeno úspěšné přidání Firebase do Androidí aplikace.

Během každého sestavování aplikace je konfigurační soubor převeden z formátu JSON do souboru XML. Následně je možné dané hodnoty použít v aplikaci. Jelikož se konfigurační soubor často nemění a jednotlivé hodnoty lze změnit ručně, je vhodné vygenerovaný XML soubor přesunout do projektu. Poté již není nutné mít v projektu Google Service plugin a JSON soubor a lze je z projektu odebrat. (Google, 2019c)

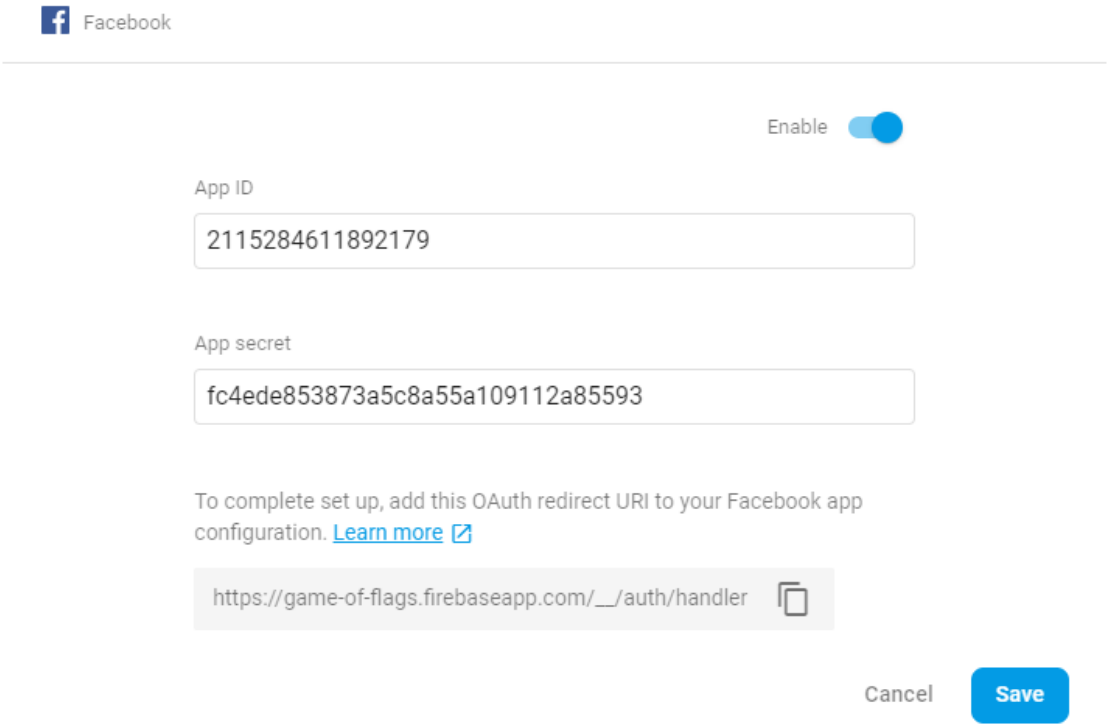
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="default_web_client_id" translatable="false">513211408030-
0088nahiv3epvd43iejatuhkq2lc5c9b.apps.googleusercontent.com</string>
  <string name="firebase_database_url" translatable="false">
https://game-of-flags.firebaseio.com</string>
  <string name="gcm_defaultSenderId" translatable="false">
513211408030</string>
  <string name="google_api_key" translatable="false">
AIzaSyBgnXdQ2Yy6EchdiR6T9QRmxDB-YcQRK50</string>
  <string name="google_app_id" translatable="false">
1:513211408030:android:cb1836e37d574d70</string>
  <string name="google_crash_reporting_api_key" translatable="false">
AIzaSyBgnXdQ2Yy6EchdiR6T9QRmxDB-YcQRK50</string>
  <string name="google_storage_bucket" translatable="false">
game-of-flags.appspot.com</string>
  <string name="project_id" translatable="false">game-of-flags</string>
</resources>
```

Ukázka kódu 4 - Vygenerovaný konfigurační soubor, zdroj: vlastní

5.2.2 Přihlašování

Firebase nabízí 2 způsoby přidání přihlašování do aplikace. První možností je vytvořit pro každý typ přihlašování vlastní kód. Druhou možností je využití FirebaseUI knihovny, která nabízí přihlašování pomocí nejpoužívanějších providerů, včetně Google a Facebooku. Dále nabízí možnost propojení více providerů pro 1 uživatelský účet nebo využití funkce Smart Lock, která slouží k bezpečnému zapamatování přihlašovacích údajů. (Firebase, 2019d) Vzhledem ke zmíněným výhodám bude pro přihlášení použit FirebaseUI.

Pro zprovoznění přihlašování pomocí Google účtu je nutné pouze přidat v konzoli již zmíněný SHA-1 otisk zařízení, na kterém bude probíhat vývoj. Přihlašování pomocí Facebooku vyžaduje vytvoření aplikace ve vývojové sekci Facebooku. Po vytvoření aplikace je vygenerován aplikační identifikátor a heslo aplikace. Tyto hodnoty je následně nutné vyplnit v sekci přihlašovacích metod v konzoli Firebase. Pro dokončení propojení je ještě nutné přidat adresu pro přesměrování přihlášení do vytvořené aplikace na Facebooku.



Obr. 7- Firebase, přidání Facebooku, zdroj: vlastní

Vytvoření přihlašování pomocí Facebook účtu vyžaduje ještě přidání aplikačního identifikátoru Facebook aplikace do zdrojové části pluginu, například do vygenerovaného konfiguračního souboru. (Firebase, 2019d)

```
<string name="facebook_application_id" translatable="false">
2115284611892179</string>
<string name="facebook_login_protocol_scheme" translatable="false">
fb2115284611892179</string>
```

Ukázka kódu 5 - Přidání identifikátoru Facebook aplikace, zdroj: vlastní

Následné vytvoření přihlašování pomocí FirebaseUI je jednoduché. Nejprve je nutné vytvořit seznam providerů, v této aplikace pouze Google a Facebook. Následně je vytvořena aktivita s přihlašováním pomocí vybraných providerů. Celá metoda z ukázky kódu 6 je volána při stisknutí tlačítka přihlásit. Vytvoření aktivita vrací informace o přihlašování. Pokud je přihlášení úspěšné, je zaregistrován naslouchač dat o uživateli, který získává data z Realtime databáze. Po odhlášení uživatele je naslouchač odebrán a informace o uživateli zapomenuty.

```
public void createSignInIntent() {
    // Choose authentication providers
    List<AuthUI.IdpConfig> providers = Arrays.asList(
            new AuthUI.IdpConfig.GoogleBuilder().build(),
            new AuthUI.IdpConfig.FacebookBuilder().build());

    // Create and launch sign-in intent
    activity.startActivityForResult(
            AuthUI.getInstance()
                    .createSignInIntentBuilder()
                    .setAvailableProviders(providers)
                    .build(),
            RC_SIGN_IN);
}
```

Ukázka kódu 6 - Metoda pro zapnutí přihlašovací aktivity, zdroj: vlastní

5.2.3 Realtime databáze

V projektu jsou využity všechny 3 zmíněné typy úložišť, které nabízí Firebase. Prvním typem je Realtime databáze, která ukládá informace do velkého JSON souboru. V této databázi jsou uloženy informace o aktivních hráčích, otiscích pro určení polohy uživatele a věžích.

Pro zobrazení aktivních hráčů na mapě je důležité mít uloženou pozici hráče. Je nutné znát nejen souřadnice, ale i patro, na jakém se hráč nachází. Další důležitou částí informace o frakci hráče, která slouží pro rozlišení hráčů na mapě. Poslední důležitou informací je čas poslední změny informací o hráči. Pokud není hráč delší dobu aktivní, přestává být zobrazen na mapě.

```
{
  "onlinePlayers" : {
    "ikivAiby9MS3YFsrZececgTKy833" : {
      "experience" : 320,
      "fraction" : 1,
      "lastChangedAt" : 1564173916386,
      "position" : {
        "floor" : "J1NP",
        "x" : 1000,
        "y" : 1000
      },
      "username" : "google"
    }, {...}
}
```

Ukázka kódu 7 - Podoba aktivních hráčů v Realtime databázi, zdroj: vlastní

Pro určení polohy zařízení se používají otisky uložené v zařízení. Aby bylo možné otisky přidávat bez nutnosti vydání nové verze aplikace, jsou nové otisky uloženy v Realtime databázi. Zde jsou uloženy 3 hlavní části. První částí je pozice, kde byl otisk vytvořen. Druhou částí je seznam BLE otisků a třetí částí je seznam Wi-Fi otisků. V obou případech je důležitá především síla signálu a adresa.

```
{
"scans" : {
  "1-LhMOV5jHQ0Co9F9C_ro" : {
   "bleScans" : [ {
    "address" : "00:E7:12:1B:3C:03",
    "rssi" : -71,
    "time" : 56
   }, {...} ],
   "level" : "J1NP",
   "timestamp" : 1,
   "wifiScans" : [ {
    "channel" : 7,
    "frequency" : 2442,
    "mac" : "e4:be:ed:31:8f:73",
    "rssi" : -35,
    "ssid" : "my",
    "technology" : "g/n",
    "time" : 30
   }, {...} ],
   "x" : 1000,
   "y" : 1000
  }, {...}
}
```


Ukázka kódu 8 - Podoba otisků pro lokalizaci v Realtime databázi, zdroj: vlastní

Poslední částí jsou informace o věžích. Kromě informací o poloze věže jsou zde uloženy i informace o oblíbenosti věže, výdrž věže, frakce, které věž patří, kdy a kým byla věž naposledy zabrána, hodnota pro ověření pozice rozšířenou realitou a název věže.

```
"towers" : {
  "tower14" : {
    "arTarget" : "J14",
    "capturedAt" : 0,
    "capturedBy" : "Nikdo",
    "floor" : "J2NP",
    "fraction" : 1,
    "healthPoints" : 20,
    "name" : "Věž u J14",
    "popularity" : 0,
    "x" : 2000,
    "y" : 1400
  }, {...}
}
```

Ukázka kódu 9 - Podoba věží v Realtime databázi, zdroj: vlastní

Pro získávání informací z Realtime databáze se používají naslouchače. Existuje více typů naslouchačů. Prvním typem je naslouchač, který čeká na informace a po získání je jeho činnost ukončena. Druhým typem je naslouchač, který čeká na změny v určité části databáze a při každé změně obdrží nové informace. Důležité je, že obdrží i změny v potomcích objektů, takže výběr naslouchané části je důležitý. (Firebase, 2019e)

Nahrávání informací probíhá vždy na konkrétní místo v databázi. Pokud v daném místě existuje potomek uložený pod stejným jménem, je potomek nahrazen. Měnit lze i hodnoty konkrétních potomků. V ukázce kódu 10 je vidět změna aktivních hráčů. Databáze vyhledá konkrétní místo a změní potomka. Pokud potomek (nebo více) neexistuje, je vytvořen.

```
private DatabaseReference dbRealtime;
dbRealtime = FirebaseDatabase.getInstance().getReference();
dbRealtime.child(ONLINE_PLAYERS).child(userUid).setValue(onlinePlayer);
```

Ukázka kódu 10 - Java, nahrání dat o aktivním hráči, zdroj: vlastní

Pro zamezení přístupu k informacím nepovoleným uživatelům umožňuje Firebase nastavit pravidla v konzoli. Jelikož je možné využívat aplikaci bez přihlášení, je čtení informací povoleno všem uživatelům. Zapisovat informace mohou pouze přihlášení uživatelé, jak je ukázáno na obrázku 8.

```
{
  "rules": {
    ".read": "true",
    ".write": "auth.uid != null"
  }
}
```

Obr. 8 - Realtime databáze - pravidla přístupu, zdroj: vlastní

5.2.4 Cloud Firestore

Druhým typem použitého úložiště je Cloud Firestore. Jak již bylo řečeno, tato databáze se skládá z kolekcí dokumentů. V této databázi jsou uloženy informace o každém hráči a nové otisky, které vytvořili hráči.

```
bonusFreePoints: 0
damagePoints: 3
experience: 560
fraction: 2
fractionChanged: 8
lastAttackTimestamp: 1564259669003
lastChangeFractionAt: 1564259689726
lastRepairTimestamp: 1564259723405
nameChanged: true
nickname: "google"
repairPoints: 2
```

Obr. 9 – Firestore – informace o hráči, zdroj: vlastní

Pro každého hráče jsou uloženy informace o jméně, informace, zdali bylo jméno změněno, která slouží pro případnou změnu jména, o frakci, včetně času poslední změny a počtu změn, časy posledního útoku a obrany věží, zkušenosti hráče a rozdělení bodů do vlastností.

Struktura nových otisků je podobná otiskům pro lokalizaci. Obsahuje navíc zjistitelné informace o zařízení, na kterém byl otisk pořízen, včetně informací ze senzorů – akcelerometru, gyroskopu a magnetometru. Dále obsahuje informace o vypočtené poloze zařízením, aby bylo případně možné zjistit odchylku od skutečné pozice. Nakonec je u každého otisku uložena adresa, na které se nachází snímek obrazovky z doby dokončení tvorby otisku sloužící k ověření pozice zařízení.

Pro přidání informací do databáze existují 2 způsoby. Prvním způsobem je použití metody set, která vytvoří nebo přepíše určitý dokument předaným objektem. Druhým způsobem je metoda add, která vytvoří v kolekci nový dokument s náhodným jménem a uloží v něm objekt. Pokud je potřeba pouze upravit určitý atribut objektu, lze použít metodu update. (Firebase, 2019f)

```
private FirebaseFirestore dbFirestore;
dbFirestore = FirebaseFirestore.getInstance();
DocumentReference docRef =
dbFirestore.collection(PLAYERS).document(userUid);
docRef.update("damagePoints", player.getDamagePoints() + 1);
```

Ukázka kódu 11 - Java, Firestore - přidání bodu útoku, zdroj: vlastní

Získávání informací v Cloud Firestore probíhá pomocí metody get, po které je volán určitý naslouchač. Naslouchače se liší podle úspěšnosti získání informací. I zde je možná naslouchat změnám vybraného dokumentu přiřazením naslouchače přímo na určitý dokument.

Jelikož jsou v této databázi uloženy pouze informace o hráčích a nové získané otisky, jsou oprávnění číst i psát povolena pouze přihlášeným uživatelům, jak zobrazuje obrázek 10.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth.uid != null;
    }
  }
}
```

Obr. 10 - Cloud Firestore - pravidla přístupu, zdroj: vlastní

5.2.5 Cloud Storage

Posledním typem použitého úložiště od Firebase je Cloud Storage. Toto úložiště je určeno pro ukládání obsahu vytvořeného uživateli, například obrázky nebo videa. Soubory jsou uloženy v Google Cloud Storage, který nabízí vysokou dostupnost a škálovatelnost. Toto úložiště využívá například Spotify pro ukládání hudby. (Google, 2019d)

Aby bylo možné toto úložiště použít, je nutné ho nejprve přidat k projektu ve Firebase konzoli. Pro přidání je nutné vybrat, v jakém oblasti se bude úložiště nacházet. Je vhodné vybrat místo, které je označeno jako více oblastní, protože jsou data replikována do více regionů. Vybraná oblast je následně použita i pro Cloud Firestore. (Google, 2019d)

V této práci je Cloud Storage využito pro ukládání snímků obrazovky po dokončení tvorby otisku. Nejprve je nahrán snímek do úložiště a až poté je nahrán otisk do Firestore s adresou odkazující na vytvořený snímek. Pro ukládání snímků je vytvořena nová složka v Cloud Storage s názvem „fingerprintScreenShot“.

```
private StorageReference storage;
storage = FirebaseStorage.getInstance().getReference();
StorageReference screenShotRef = storage.child("fingerprintScreenShot/" +
System.currentTimeMillis() + "_" + floor + "_" + x + "_" + y + ".jpg");
UploadTask uploadTask = screenShotRef.putBytes(bytes);
```

Ukázka kódu 12 - Java, Storage - přidání snímku obrazovky, zdroj: vlastní

Aby bylo možné soubor nahrát, je nutné nejprve vytvořit odkaz na konkrétní místo ve složce. Soubory lze poté nahrát pomocí 3 metod, které se liší od typu

zpracování souboru. Lze nahrát přímo konkrétní soubor uložený v zařízení pomocí metody putFile, nebo ve zpracovávané formě pomocí putStream a putBytes. (Firebase, 2019g)

Získat soubor lze pomocí 2 metod. První je getBytes, kde je možné zadat maximální velikost získaných bajtů naráz, například při zpracování objemného souboru. Druhou metodou je getFile, která uloží celý soubor do zařízení. (Firebase, 2019h)

I Cloud Storage umožňuje nastavit pravidla přístupu k souborům. Jelikož otisky vytvářejí pouze přihlášení uživatelé, jsou pravidla ponechána v základním nastavení, ve kterém mohou číst a zapisovat pouze přihlášení uživatelé. Podoba psaní pravidel je stejná jako pro Cloud Firestore.

5.2.6 Rozdělení informací mezi Realtime databázi a Cloud Firestore

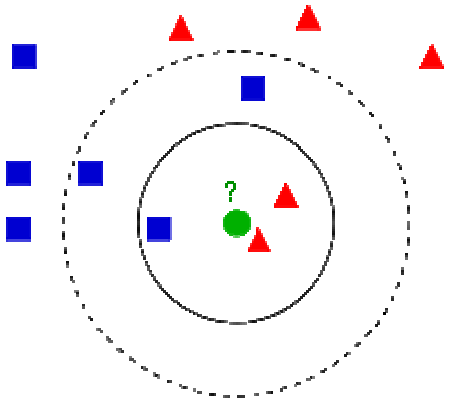
Ačkoliv je použití obou databází velmi podobné a obě nabízejí možnost naslouchání změn s následným informováním připojených zařízení a pro nové projekty je doporučována Cloud Firestore, jsou informace rozděleny. Důvodem je rozdílný způsob placení obou databází. Jak již bylo zmíněno, Realtime databáze je placena od počtu přenesených dat, Cloud Firestore podle počtu vykonaných operací. Jelikož jsou o aktivních hráčích a věžích informovány všechny připojené zařízení každých několik vteřin a Firestore počítá informování každého zařízení o změně zvlášť, došlo by při připojení více zařízení k vykonání velkého počtu operací v krátké době. Naopak získání nebo změna informací o hráči a nahrání nového otisku je záležitostí konkrétního zařízení, takže použití Cloud Firestore je vhodné.

5.2.7 Lokalizace uživatele

Jak již bylo několikrát řečeno, výsledná aplikace umožní určit polohu zařízení v budově. Vzhledem k již získaným otiskům je pro určení polohy použit algoritmus nejbližších sousedů.

Tento algoritmus je určen například pro rozpoznávání vzorů ve strojovém učení a jejich klasifikaci. Podle parametru *k* se porovná počet nejbližších sousedů a výsledný objekt je zařazen do skupiny (třídy) s nejvíce výskyty v dané lokaci. Na

obrázku 11 je získán nový zelený bod. Pro k=3 bude přiřazen mezi červené trojúhelníky, pro k=5 mezi modré čtverce.



Obr. 11 - algoritmus k-nejbližších sousedů, zdroj: commons.wikimedia.org

Algoritmus je možné použít podobným způsobem i pro určení polohy pomocí vytvořeného otisku vysílačů. Nejprve je nutné pro každý otisk zjistit průměrnou sílu signálu každého vysílače. Poté je potřeba porovnat získané průměrné hodnoty síly signálu vysílačů s uloženými hodnotami ostatních otisků. Nejprve je nutné doplnit do porovnávaných seznamů hodnoty nulového signálu pro všechny vysílače, které se nachází pouze v opačném seznamu. Tím se získají dva seznamy se stejnými názvy vysílačů, ale různými průměrnými hodnotami signálů. Následně se vypočítá vzdálenost pomocí rovnice pro výpočet Euklidovské vzdálenosti.

$$d = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Q a p jsou v tomto případě seznamy a d výsledná vzdálenost. Výsledný seznam vzdáleností je srovnán od nejmenší hodnoty a je možné podle určitého počtu nejbližších sousedů určit polohu zařízení.

Jak již bylo zmíněno, určit polohu pomocí vypočtené vzdálenosti od sousedů lze více způsoby. Nejjednodušší variantou je zvolit polohu nejbližšího souseda za polohu zařízení. Další možností je vzít určitý počet sousedů a výslednou polohu určit zprůměrováním jejich poloh. Třetí variantou je použití váženého průměru. Variant výpočtu váhy je ale více. První možností je použití převrácené hodnoty pozice souseda v seznamu, kdy nejbližší soused má váhu 1, další 1/2, poté 1/3 až po 1/k. Tato varianta je jednoduchá, ale nezabývá se rozdíly mezi vzdálenostmi jednotlivých

sousedů od zařízení. Druhou možností je použití převrácené hodnoty vzdálenosti souseda. Čím více je soused vzdálen, tím méně ovlivní výslednou polohu. Tento způsob je reálnější a je použit pro implementaci řešení. (Caso, 2015)

5.2.7.1 Tvorba otisků

Tvorba otisku pro zjištění polohy se neliší od tvorby otisku, který je posílán do Cloud Firestore. Tvorbou otisků se zabývala i předešlá práce, která sloužila k vytvoření databáze otisků.

Nejprve je nutné vytvořit naslouchače informací z Wi-Fi a BLE senzorů. Následné naslouchání probíhá vždy po určitou dobu. Aby byly otisky sjednocené s otisky z ostatních aplikací, probíhá naslouchání v intervalech po 10 vteřinách. Výsledný otisk obsahuje seznamy okolních Wi-Fi a BLE vysílačů a síly signálu.

Pro naslouchání je nutné, aby byly zapnuté přijímače obou technologií. Jelikož cílem této práce je vytvoření hry a snímání probíhá po celou dobu užívání aplikace, jsou přijímače při startu aplikace automaticky zapnuty a při ukončení vypnuty.

5.2.7.2 Lokální databáze

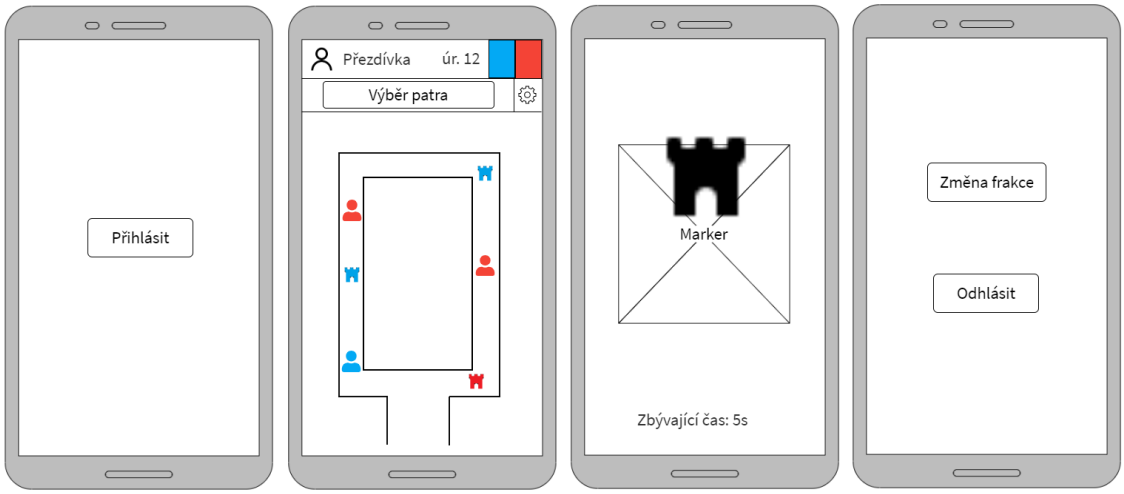
Jak již bylo zmíněno, pro určení polohy zařízení je nutné mít seznam otisků s polohou pořízení. Aby nebylo nutné otisky pokaždé stahovat z Realtime databáze, jsou veškeré otisky uloženy v lokální databázi zařízení.

Pro ukládání otisků je využita technologie Room, která je abstraktní vrstvou nad SQLite databází. Room obsahuje 3 hlavní komponenty. První komponentou je databáze. Tato komponenta se stará o vytvoření instance databáze. Druhou komponentou je entita, která představuje tabulku v databázi a třetí komponentou je Data Access Objects, která obsahuje metody pro vykonávání akcí v databázi. (Android, 2019b)

Pro prvotní naplnění databáze jsou v aplikaci uloženy 4 JSON soubory obsahující vybrané otisky pro každé patro budovy J. Po spuštění aplikace je ověřeno, zdali se v Realtime databázi nenachází nové otisky, které by mohly být staženy. Pokud se nachází, jsou uloženy do databáze a ihned použity pro lokalizaci.

5.3 Mobilní část – Unity

Jak již bylo v analýze a návrhu zmíněno, pro grafickou část aplikace je použit herní engine Unity. Aplikace je rozdělena na 4 části podle způsobu použití. První částí je přihlašování, které již bylo popsáno v části Android pluginu. Druhou částí, která je zobrazena nejčastěji, je herní mapa s objekty. Třetí částí je použití rozšířené reality pro operace s věžemi a poslední částí je nastavení aplikace. Přibližná podoba částí je zobrazena na obrázku 12.



Obr. 12 - Wireframe aplikace, zdroj: vlastní

5.3.1 Možnosti všech uživatelů

Při každém spuštění aplikace je zobrazeno načítání. Aby bylo možné zobrazit věže a ostatní hráče, je nutné, aby bylo zařízení připojeno k internetu. Pokud se nepodaří získat informace z databází, není možné aplikaci využívat. Po úspěšném načtení je zobrazena interaktivní mapa, na které jsou zobrazeni aktivní hráči, uživatel a věže.

Horní část obrazovky slouží pro zobrazení informací o frakcích, hráči, zobrazení jiných pater a vstup do nastavení. Pro frakce je důležitý pouze aktuální počet vlastněných věží, proto je počet uveden nad barvou příslušné frakce. Podoba načítání a základní zobrazení aplikace je zobrazeno na obrázku 13.

Obr. 13 - Načítání a zobrazení nepřihlášeného uživatele, zdroj: vlastní

5.3.2 Zobrazení hráčů a informací o hráči

Informaci o hráči lze zobrazit pomocí kliknutí na avatara hráče nebo na jméno hráče. Nepřihlášený uživatel vidí upravené hodnoty, aby získal představu o možném zlepšení. Zobrazené informace jsou rozděleny do 2 částí. První část zobrazuje dosaženou úroveň, počet zkušeností a množství potřebných zkušeností na další úroveň a volné body k rozdělení mezi schopnosti. Schopnosti jsou zobrazeny v druhé části. Každý hráč má určitou sílu útoku a hodnotu opravy, které může za volné body vylepšovat. Za každou dosaženou úroveň získá hráč 1 bod. Po vylepšení schopnosti již není možné získat bod zpět, takže se hráči musí pečlivě rozhodnout, jakou schopnost si vylepší.

Každý nový hráč začíná na úrovni 1. Síla útoku i hodnota opravy jsou vypočítávány podle počtu přidělených bodů. Základní hodnotou útoku jsou 4 body, každý přidaný bod zvedá sílu o 2. Rozptyl útoku je 20 %, vypočtená síla je tedy

v rozsahu 80-120 %. Základní hodnota obrany je 2 body. Každý bod zvýší hodnotu také o 2 body, ale rozptyl je nastaven na 30 %. Výsledná hodnota opravy je tedy z rozsahu 70-130 %.

Výpočet úrovní probíhá podle počtu získaných zkušeností. Hodnota konstanty je zvolena tak, aby každá další úroveň vyžadovala o 40 zkušeností více.

```
const double levelConst = 0.22360679775;
public static int CalculateLevel(int experience) {
    return (int)Math.Floor(levelConst * Math.Sqrt(experience)) + 1;
}
public static int CalculateExperienceForNextLevel(int experience)  {
    return (int) Math.Round(Math.Pow(CalculateLevel(experience) /
levelConst, 2));
}
```

Ukázka kódu 13 - C#, Výpočet úrovní a potřebných zkušeností na další úroveň, zdroj: vlastní

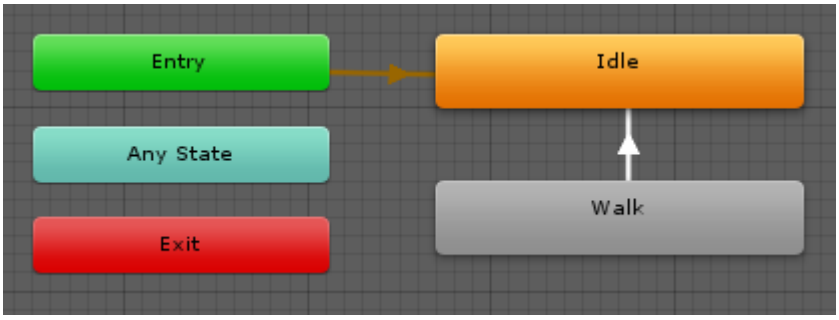
Obr. 14 - Informace o hráči, zdroj: vlastní

5.3.2.1 Zobrazení na mapě

Pro zobrazení hráče na mapě byl zvolen model robota, který je zdarma dostupný v obchodu Unity. Model obsahuje animaci pro chůzi, je tedy vhodný pro použití v aplikaci. Je důležité připomenout, že uživatel nemá možnost s modelem hýbat, veškerý pohyb je uskutečněn výpočtem pozice v Android pluginu.

Animace pohybu je nastavena v animátoru. Animátor obsahuje 3 základní stavy. Prvním je vstupní stav, do kterého se dostane každý objekt při vytvoření. Druhý stav je speciální. Zastupuje všechny stavy v animaci a slouží k označení, že se do následujícího stavu lze dostat z jakéhokoliv jiného stavu. (Unity, 2019e) Poslední stavem je Exit, který ukončí probíhající animace a zavolá vstupní stav.

Zvolený model robota obsahuje 2 animace. První animace je základní, ve které robot čeká a nehýbe se. Druhou animací je chůze. Po dokončení animace chůze je zavolána čekající animace, jak ukazuje obrázek 15.



Obr. 15 – Unity, použití animátoru, zdroj: vlastní

Pro použití vytvořeného animátoru je nutné tuto komponentu přiřadit objektu, který ji bude používat, v tomto případě objekt hráče. Po získání komponenty v kódu je možné spustit animaci chůze, jak je zobrazeno v ukázce 14.

```
private Animator animator;
animator = playerObject.GetComponentInChildren<Animator>();
animator.Play("Walk");
```

Ukázka kódu 14 - C#, Použití Animátoru v kódu, zdroj: vlastní

Aby bylo možné rozeznat nepřihlášeného uživatele aplikace nebo aktivního hráče, je na mapě zobrazen zelenou barvou. Ostatní aktivní hráči mají barvy své frakce. Zobrazení hráčů na mapě je znázorněno na obrázku 13.

5.3.3 Informace o věži

Každá věž obsahuje vlastní informace, které je možné zobrazit. Po kliknutí na vybranou věž je zobrazeno okno s danými informacemi. Jedná se o název věže, výdrž věže včetně maximální možné hodnoty, úroveň věže, frakce, které daná věž patří, kdo a kdy danou věž zabral a část obsahující akce, případně informaci, kdy bude možné akci vykonat.

Obr. 16 - Informace o věži, zdroj: vlastní

Výpočet úrovně věže probíhá pomocí stejné metody, kterou využívají i hráči. Pro zvýšení oblíbenosti věže, která se používá pro výpočet úrovně, je nutné, aby nad ní byla úspěšně vykonána akce. Za každý provedený útok a opravu získává věž 5 bodů oblíbenosti. Hráči získávají za každý úspěšný útok a opravu věže 10 bodů zkušeností, za úspěšné získání věže pro svoji frakci dokonce dvojnásobek. Každá nová úroveň zvýší maximální počet bodů výdrže o 20.

Každé okno s informacemi může obsahovat tlačítko pro vykonání akce. Existují 3 akce, které je možné vykonat. Kromě již zmíněného útoku a opravy věže, které spustí část s rozšířenou realitou, existuje zde tlačítko pro přihlášení.

Pokud věž patří frakci hráče a je plně opravena, zobrazí se informace o maximálním počtu bodů výdrže a není možné provést žádnou akci. Pokud věž není zcela opravena, ale hráč opravil nějakou věž v posledních 10 minutách, zobrazí se mu informace, kdy je možné věž opravit. Pro útok je důležité, aby hráč v posledních 10 minutách neprovedl úspěšný útok a zároveň aby na věž nebyl proveden útok v posledních 10 minutách. Tyto ochrany slouží především k zabránění domluvené výměny věží za účelem získání zkušeností mezi hráči různých frakcí.

5.3.4 Přihlašování a nastavení

Jak již bylo zmíněno, pro přihlášení je využito FirebaseUI, které vytvoří Androidí aktivitu s vybranými druhy přihlášení. Na tuto aktivitu se kromě z informací o věži dá dostat také z nastavení. Nepřihlášený uživatel vidí v nastavení pouze tlačítko pro přihlášení.

Po přihlášení je ověřeno, zdali se hráč přihlásil do aplikace poprvé. Pokud ano, je mu zobrazeno okno s polem pro zadání přezdívky a také si musí zvolit frakci. Pokud je splněn požadavek na délku přezdívky a je vybrána frakce, je registrace nového hráče dokončena.

Každý hráč má v nastavení 2 možnosti. První možností je změna frakce a druhou možností je odhlášení ze hry. Pokud se hráč rozhodne změnit frakci, je mu zobrazeno okno s informacemi o poslední změně frakce a kdy je možné frakci změnit. Po úspěšné změně frakce hráč nepřichází o získané zkušenosti nebo schopnosti, pouze mu je znemožněna opětovná změna frakce po následující týden. Podoba nastavení a okna pro změnu frakce je zobrazena na obrázku 17.

Po odhlášení ze hry jsou informace o hráči z mobilní aplikace odebrány a je znovu zobrazena ukázková podoba hráče.

Obr. 17 - Nastavení, zdroj: vlastní

5.3.5 Možnosti interakcí s mapou

Jelikož se jedná o aplikaci na mobilní zařízení, je nutné zajistit pohyb virtuální kamery v 3D scéně pomocí dotyků prsty. Kamera je zařízení, pomocí kterého uživatel vidí herní svět. Pohyb kamery je rozdělen na 2 části podle počtu současných dotyků. První částí je pohyb jedním prstem, druhou částí je pohyb pomocí dvou prstů. Více současných dotyků není v této aplikaci podporováno.

Aby bylo možné zjistit velikost posunu nezávisle na vzdálenosti kamery od mapy, je nutné přepočítat souřadnice dotyku obrazovky na souřadnice bodu pod dotykem. Zároveň je nutné počítat s tím, že se pod dotykem může nacházet objekt, který výslednou pozici ovlivní. Zjištění prostorových souřadnic bodu je znázorněno v ukázce kódu 15. Unity obsahuje metodu, pomocí které je možné přeměnit dotyk na paprsek vržený určeným směrem od kamery. Následně je možné zjistit veškeré objekty, kterými paprsek prochází. Jelikož aplikace obsahují mnoho objektů, lze

nastavit maximální vzdálenost, do jaké bude paprsek vržen. K ještě většímu upřesnění objektů, které má metoda vracet, je možné definovat vrstvu, do které je objekt zařazen. Vrstvy jsou běžně používány pro určení, jaké objekty mají být zobrazeny. Unity umožňuje použít 32 vrstev. Prvních 8 vrstev není možné měnit, ostatní je možné pojmenovat dle potřeby. (Unity, 2019f) Pro určení bodu je zvolena vrstva 10, do které je zařazen terén mapy, který je rovný.

```
private Vector3 GetWorldPoint(Vector2 screenPoint)
{
    bool hit = Physics.Raycast(Camera.main.ScreenPointToRay(screenPoint),
out RaycastHit hitInfo, 1000, 1 << 10);
    return hitInfo.point;
}
```

Ukázka kódu 15 - C#, získání bodu pod dotykem, zdroj: vlastní

5.3.5.1 Interakce pomocí dotyku

První variantou pohybu je změna pozice kamery pomocí posunu prstu po obrazovce zařízení. Při každém vykreslení obrazu je ověřeno, zdali se obrazovky neustále dotýká pouze 1 prst a pokud ano, jestli je nějaký rozdíl v pozici dotyku. Následně je kamera posunuta o vypočítaný rozdíl na nové souřadnice. Aby nebylo možné přesunout kameru příliš daleko od herních objektů, je nastaven limit vzdálenosti od středu mapy.

Důležitou částí před pohybem je také ověření, zdali se prst nenachází nad informačními okny nebo nad horním panelem s informacemi. V takovém případě by docházelo k nechtěnému pohybu například při klikání na tlačítka.

Jelikož zobrazení informací o věži probíhá také pomocí dotyku jedním prstem, je důležité rozlišit, jakou akci chce uživatel vykonat, aby nedocházelo k nechtěnému zobrazení okna s informacemi o věži, ačkoliv uživatel chce pouze posunout kameru. Z tohoto důvod je při zjištění, zdali se pod dotykem nenachází věž, ověřeno, jestli je velikost pohybu dostatečné velká. Samotné vybírání věží pro zobrazení informací je vytvořeno podobně jako v ukázce 15, rozdílem je pouze zvolená vrstva objektů, která je nastavena na vrstvu s věžemi.

5.3.5.2 Interakce pomocí dvou dotyků

Druhou částí pohybu kamery je pohyb pomocí dvou současných dotyků. Existují 3 možnosti pohybu. První možností je změna úhlu pohledu kamery. Druhou možností je rotace kamery kolem určitého bodu a poslední možností je změna přiblížení pohledu kamery.

Aby nedocházelo ke střídání možností během pohybu prstů, je na začátku zvolena možnost podle prvotního pohybu. Pokud se oba prsty posouvají současně nahoru nebo dolů, je zvolena první akce změny úhlu kamery. Pokud je dostatečně měněn úhel mezi prsty, například pohybem kopírujícím tvar kružnice, je zvolena možnost rotace. Pokud není zvolena nějaká z předešlých možností a prsty se dostatečně od sebe vzdalují nebo přibližují, jedná se o třetí možnost.

Pro určení velikosti změny úhlu pohledu kamery je vybrán dotyk s menším rozdílem pozice po zvolené ose. Následně je kamera natočena ve zvoleném úhlu. Aby nedocházelo k nechtěnému otočení kamery, jsou nastaveny limity určující minimální a maximální úhel. Minimální úhel je nastaven na 45 stupňů, maximální na 89,5 stupně. Rozdíl oproti kolmému úhlu v pohledu viditelný není, ale kolmý úhel způsobuje špatné zobrazení detailů na terénu.

Rotace kamery je rozdělena podle způsobu pohybu prstů. Pokud je zaznamenán pohyb obou prstů, je bod, kolem kterého je kamera otáčena, určen ve středu vzdálenosti prstů. Pokud uživatel mění pozici pouze jednoho prstu, je za bod, kolem kterého je rotace prováděna, zvolena pozice druhého prstu.

Při poslední možnosti nedochází ke změně pozice nebo rotace kamery. Přiblížení nebo oddálení je dosaženo pomocí změny úhlu zorného pole kamery. Čím je úhel menší, tím působí kamera blíže objektům. Aby nedocházelo k extrémnímu přiblížení nebo oddálení, je úhel udržován mezi 40 a 100 stupni.

5.3.5.3 Funkce tlačítka následování

Tlačítko následování, které je viditelné na obrázcích s informacemi nebo mapou jako černý terč, slouží k přesunu pohledu kamery na model uživatele. Tlačítko má 2 funkce, které se liší podle počtu stisknutí.

První funkcí je přesun kamery za model uživatele se zachováním úhlu kamery. Pokud se uživatel nachází na jiném patře, než které je aktuálně zobrazeno,

je patro změněno. Přesun kamery probíhá plynule pomocí metody, která postupně mění pozici na cílovou ve zvoleném čase. Po přesunu kamery je model uživatele uprostřed obrazovky.

Druhá funkce je aktivována, pokud uživatel stiskne tlačítko vícekrát. Kamera je následně postupně přesunuta téměř nad model hráče, úhel pohledu je nastaven na 89,5 stupně, rotace je nastavena na 0 stupňů a zorné pole je vráceno na základní hodnotu 90 stupňů.

Tlačítko neslouží pouze k zaměření pozice hráče nebo změně nastavení kamery. Pokud uživatel nevyvolá nějakou akci pro kameru, následuje kamera model uživatele při každé změně pozice modelu vyvolané změnou pozice uživatele.

5.3.6 Podoba a tvorba mapy a terénu

Při vývoji aplikace byl pro mapu vybrán jednoduchý objekt s texturou daného patra. Pro pohyb kamery byl vytvořen jednoduchý plochý objekt šedivé barvy nastavený do příslušné vrstvy. Ačkoliv toto řešení velmi připomínalo podobu mapy v předešlé aplikaci převedenou do 3D prostředí, nebylo dané řešení vhodné pro tuto práci. Proto byl objekt s texturou obrázku mapy patra nahrazen vytvořeným modelem patra. Objekt sloužící pro určení pozice bodu v paprsku byl nahrazen terénem.

Jak již bylo řečeno v analýze, velikost plánků pater budovy je odlišná. Aby byla zachována kompatibilita zobrazení s ostatními aplikacemi, jsou rozměry modelů určeny podle velikosti na pláncích.

V části rešerše bylo zmíněno, že Unity obsahuje nástroj na tvorbu terénu. Tento nástroj je použit na tvorbu terénů pro tuto aplikaci. Aby nebyl terén prázdný, je doplněn o detaily v podobě trávy. Jelikož je každý model patra budovy jiný a zobrazené detaily terénu byly zobrazeny pod zdmi vytvořených modelů, je pro každé patro vytvořen vlastní terén. V projektu tedy existují 4 modely pater a 4 terény.

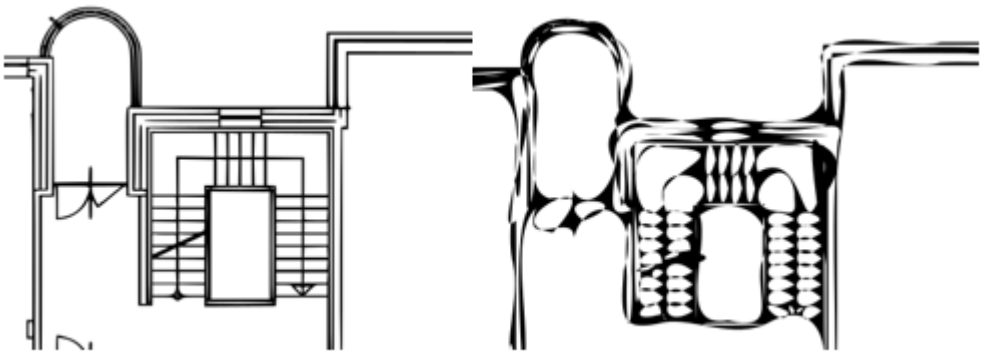
5.3.6.1 Tvorba modelů mapy

Pro tvorbu objektu byl použit již zmíněný nástroj Blender. Blender umí vytvářet objekty z vektorového formátu SVG, bylo tedy nutné zajistit převedení

veškerých čar v pláncích PNG souborů do vektorové podoby. Popsanou schopnost má program Inkscape. Inkscape je open-source vektorový grafický editor určený především pro operační systém Linux, ale je možné jej použít i v dalších desktopových systémech.

Inkscape nabízí více způsobů vektorizace obrázku. Vektorizace se dělí na 2 skupiny podle výsledného počtu křivek. Tyto skupiny se dále dělí podle způsobu detekce křivek. První skupinou tvoří způsoby, které vytvoří jedinou křivku. Jedná se o odříznutí jasu, detekci hran a kvantizaci barev. Druhou skupinu tvoří způsoby, které vytvoří více výsledných křivek podle stupňů jasu nebo barev. Vektorizace také umožní odebrat pozadí z obrázku.

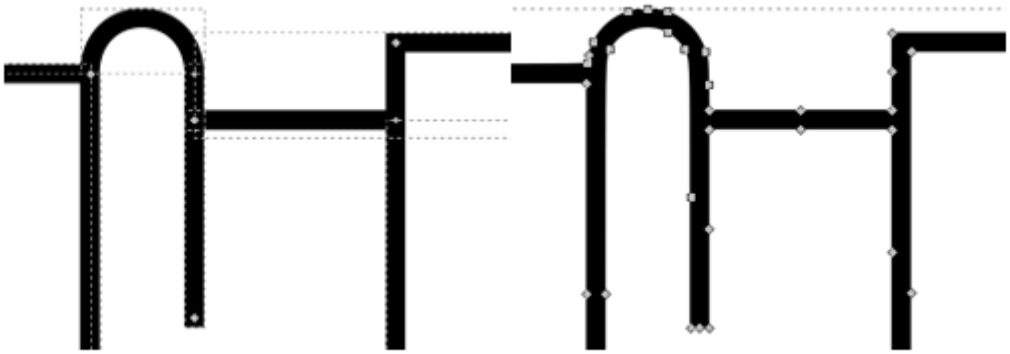
Pro vektorizaci byly použity 4 obrázky plánků pater. Každý obrázek obsahoval názvy místností modrou barvou a stěny budovy černou barvou. Po odebrání textů a následné vektorizaci bylo zjištěno, že výsledný objekt obsahuje příliš mnoho vrcholů a samotná tvorba zdí v Blenderu počet vrcholů ještě znásobila. Inkscape obsahuje možnost zjednodušení čar, výsledek bohužel není použitelný, jak je znázorněno na obrázku 18. Z obrázku je také patrné, že vytvořených čar pro konkrétní zeď je příliš mnoho a tento způsob tedy není možné použít. Vektorové mapy pro všechny patra byly následně nakresleny ručně.



Obr. 18 - Inkscape, Zjednodušení čar, zdroj: vlastní

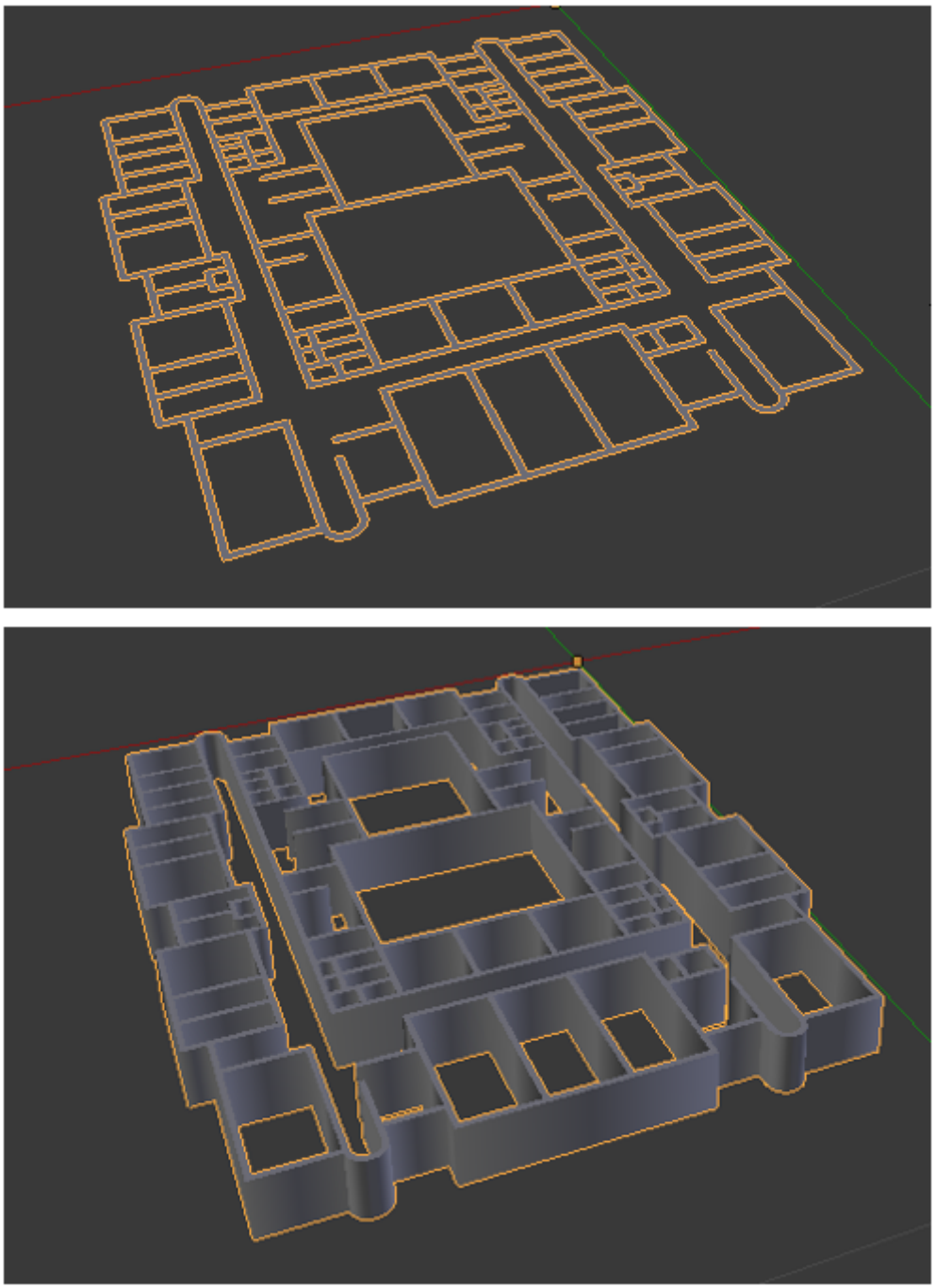
Samotná tvorba objektu pro rozsáhlejší obrázky je náročná, naštěstí místnosti v budově mají převážně rovné stěny, které na sebe navazují, takže je možné tvořit delší křivky. Po vytvoření všech křivek je nutné ověřit, zdali rozměry

vytvořeného objektu odpovídají rozměrům obrázku a případně rozměry upravit, protože Inkscape při otevření obrázku rozměry zmenší. Tuto operaci lze provést i před samotnou tvorbou objektu. Pokud dojde k úpravě rozměrů až po vytvoření objektu, je šířka vytvořených křivek také přepočítána, jinak je šířka nastavena na 1 pixel. Taková šířka je nedostatečná pro zobrazení velkých objektů, je proto nutné ji zvětšit. Po importu objektu do Blenderu bylo zjištěno, že Blender neumí zachovat nastavenou šířku křivek v SVG formátu, proto je nutné vytvořit objekt jiným způsobem. Šířka křivek je nastavena na 20 pixelů a objekt je exportován do PNG obrázku. Následně je v Inkscape pomocí nástroje detekce hran vytvořen výsledný objekt. Jak je patrné na obrázku 19, objekt vytvořený tímto způsobem obsahuje několikanásobně více vrcholů než verze, která je vytvořena ručně, ale množství v řádech jednotek tisíců není pro žádné chytré zařízení problém. Takto vytvořený objekt neobsahuje záznam o šířce křivek a je tedy správně importován v Blenderu.



Obr. 19 - Inkscape, Vytvořený objekt (vlevo) a detekovaný objekt (vpravo), zdroj: vlastní

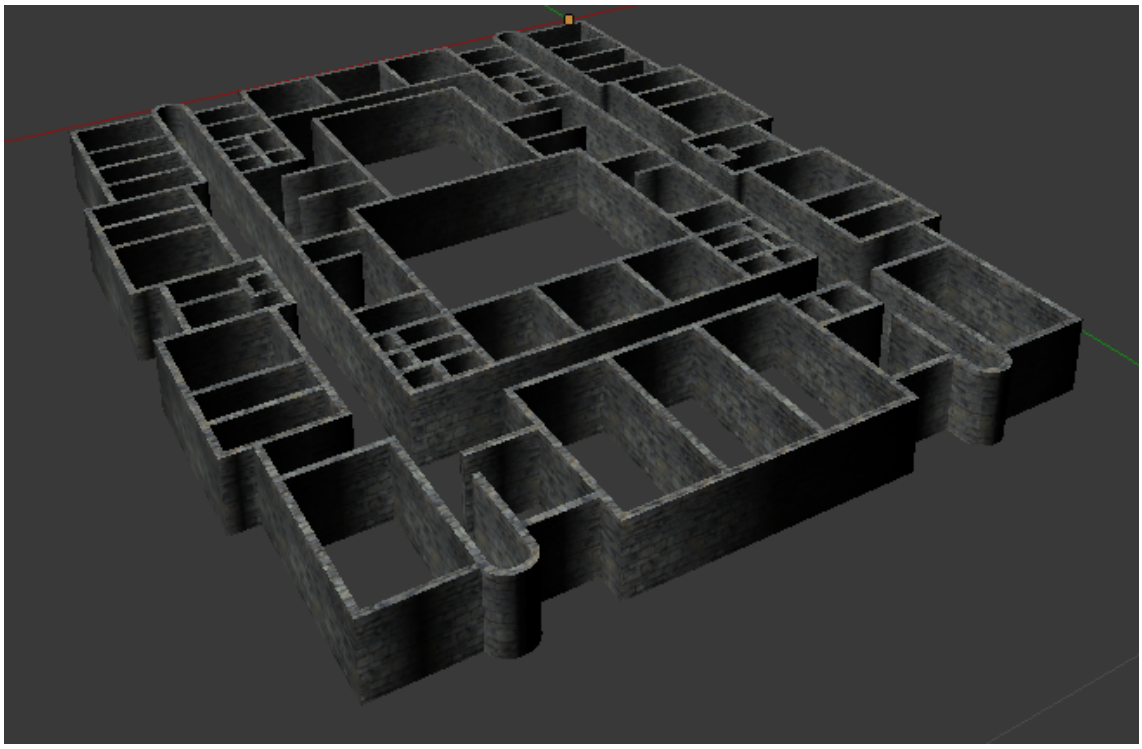
Po importu objektu mapy je následně nutné vytvořit z plochého tvaru objektu obsahujícího půdorys stěn opravdové stěny. Pro tento účel obsahuje Blender funkci Vytlačení. Tato funkce vytvoří kopie vybraných vrcholů na nové pozici, ale zachová spojení s původními vrcholy. Vrcholy jsou následně propojeny do hran a hrany vytvoří plochy. (Blender, 2019) Ukázka funkce je patrná na obrázku 20.



Obr. 20 -Blender, Použití funkce vytlačení, zdroj: vlastní

Pro použití modelu je ještě nutné přidat texturu připomínající zeď. Pro přidání textury na objekt s velkým množstvím ploch je nutné mít bezešvou texturu. Taková textura na sebe ze všech stran navazuje. Nejprve je nutné převést křivky na síť (mesh). Poté je možné vytvořit UV mapu modelu. Je nutné použít variantu projekce na krychli, jinak budou mít některé plochy texturu jiným směrem.

Následně je nutné změnit velikost UV mapy na požadovanou velikost, aby byla použita celá textura. Podoba výsledného modelu je zobrazena na obrázku 21.



Obr. 21 - Blender, Podoba výsledného objektu, zdroj: vlastní

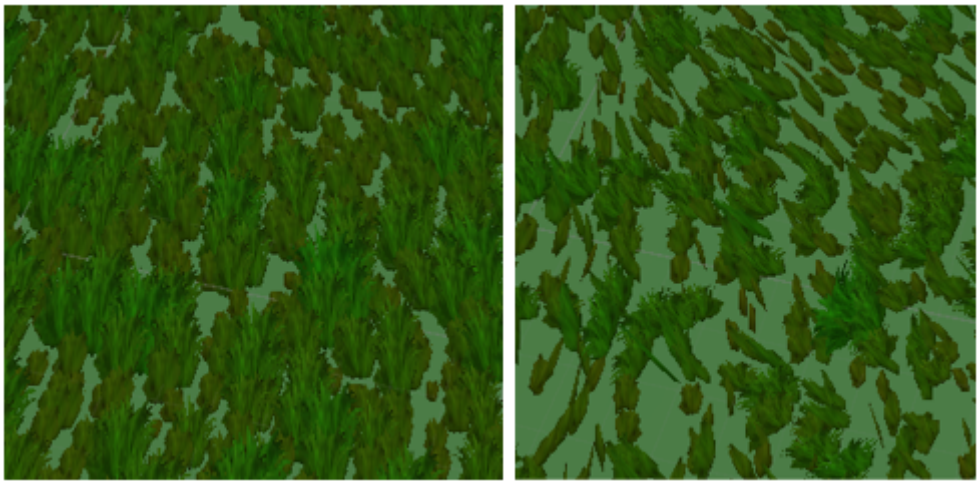
Blender používá pro ukládání vlastních modelů soubory s příponou blend. Aby bylo možné tyto soubory použít v Unity, je nutné mít nainstalovaný Blender na stejném zařízení. Po přidání souboru do projektu Unity automaticky spustí Blender a vytvoří použitelný model. Již během tvorby modelu v Blenderu bylo zjištěno, že Blender změní velikost objektu. Aby bylo možné použít lokalizaci uživatele s výsledným modelem stejným způsobem jako při použití obrázku mapy, je nutné upravit velikost modelu na velikost obrázky mapy a umístit ho nad daný obrázek. Následně je obrázek vymazán. Vzhledem k rozdílným velikostem obrázků pater je popsaný proces tvorby modelu včetně umístění modelu v projektu nutné udělat pro každé patro zvlášť.

5.3.6.2 Tvorba terénů

Jak již bylo zmíněno, Unity má pro tvorbu terénu vestavěný editor. Při vytváření nového terénu je vytvořen čtvercový plochý terén, který nemá texturu, ale má již nastavené parametry pro vítr a vykreslování objektů. Vítr se vztahuje na

detaily, které představují trávu. Lze nastavit sílu větru, rychlost a ohýbání trávy. Nastavení vykreslování objektů znamená, do jaké vzdálenosti budou objekty viditelné. Tato hodnota byla oproti základní hodnotě zvýšena, aby byla tráva viditelná i při změně úhlu pohledu.

Pro použití v tomto projektu není vhodné měnit členitost terénu, tato funkce editoru tedy není využita. Pro podklad terénu je zvolena zelená barva. Následně je nutné přidat detaily terénu v podobě textury trávy. Editor neobsahuje žádné textury trávy, pro zobrazení trávy byla tedy získána dostupná textura na internetu. Každá vytvořená textura má vlastní nastavení, které je možné i dodatečně měnit. Jedná se o minimální a maximální šířku a výšku trávy, různorodost trávy – prolínání oblastí suché a zdravé trávy, barvu zdravé a suché trávy a takzvaný „billboarding“. Tato funkce automaticky otáčí texturu proti kameře, takže detail trávy je viditelný stejným způsobem z různých úhlů. Kvůli této funkci je také nastaven maximální úhel kamery pod 90 stupňů, protože docházelo ke špatnému zobrazení textury. Následně je tráva rozmístěna po celém terénu.



Obr. 22 - Unity, Textura trávy s funkcí billboard (vlevo) a bez ní (vpravo), zdroj: vlastní

Aby nebyla tráva zobrazena pod zdmi vytvořeného objektu v minulé podkapitole, případně aby ani její části do zdi nezasahovaly, je tráva v okolí objektu vymazána. Je důležité vzít v úvahu rotaci kamery a působící vítr na trávu. Aby tato oblast nebyla prázdná, je vytvořena textura s menšími rozměry, která je následně použita i mezi zdmi objektu.

5.3.7 Zobrazení rozšířené reality a snímání signálů

Jak již bylo v úvodu této kapitoly zmíněno, pro zobrazení rozšířené reality je využit engine Vuforia. Aby bylo možné ověřit přibližnou pozici hráče, je pro zobrazení virtuálních objektů využito označení místností v celé budově.

Pro použití Vuforie v Unity je potřeba tento engine do Unity nainstalovat. Následně je nutné přidat Vuforii do projektu. Po přidání je vytvořen konfigurační soubor Vuforie, kde je nutné zadat klíč vytvořený ve vývojářské konzoli Vuforie a vložit databázi se značkami.

Vuforie nabízí 2 druhy klíčů. První klíč je vývojářský, zdarma dostupný, ale s různými omezeními, například na počet objektů v cloudové databázi. Druhý klíč je určen pro vydání aplikace. Vzhledem k použití pouze databáze se značkami, je pro projekt použita vývojářská licence.

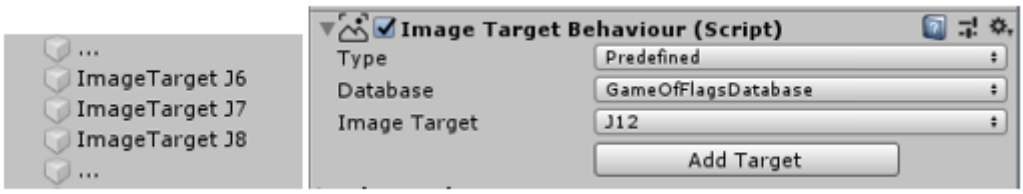
Pro použití značek je vytvořena databáze obsahující obrázky s označením jednotlivých místností. Po nahrání obrázku Vuforie označí body použitelné pro rozpoznání obrázku. Pokud je detekováno příliš málo bodů, není možné obrázek použít. Detekce je znázorněna na obrázku 23.

Obr. 23 – Vuforia, Detekce bodů na značce, zdroj: vlastní

Po nahrání všech obrázků je nutné databázi stáhnout a přidat do Unity. Stažený soubor stačí vložit do adresáře projektu, Unity jej rozpozná a přidá do projektu.

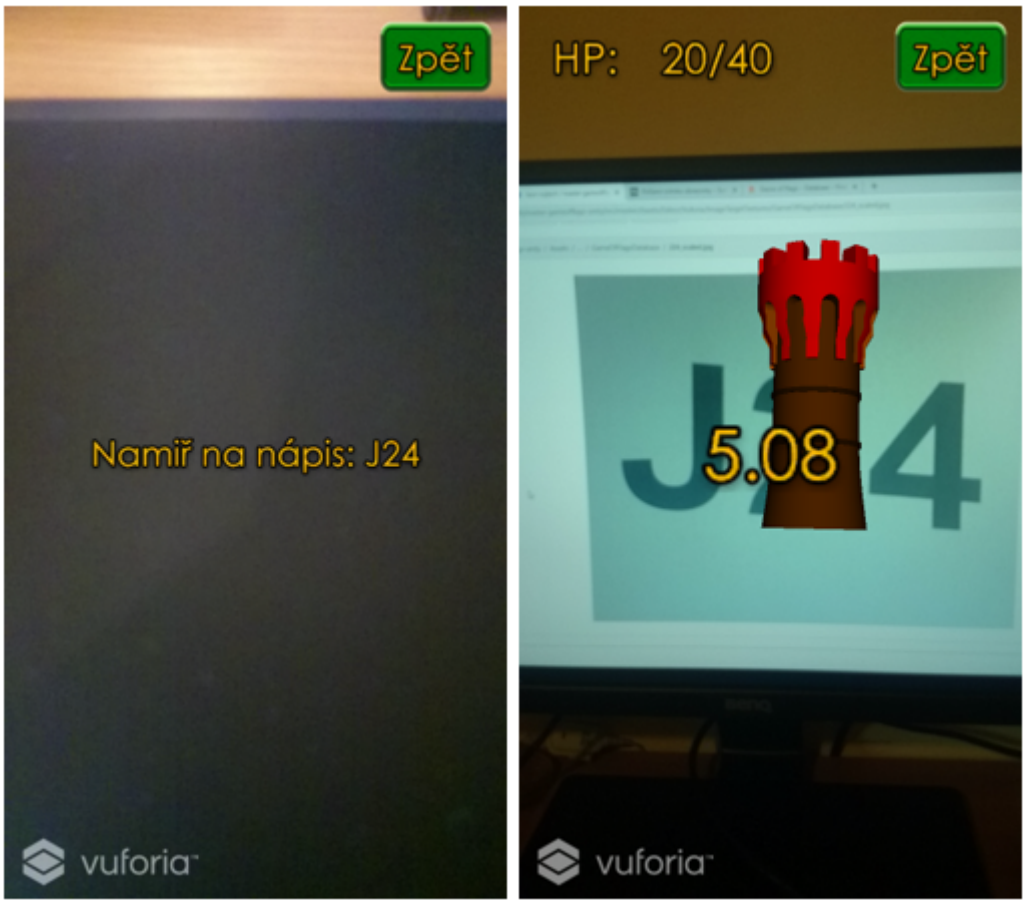
Vuforia používá vlastní objekt kamery, který je nutné použít pro zobrazení objektů. Následně je možné přidat vytvořené značky pro detekci místností. Aby bylo

možné rozlišit, jaká značka byla detekována, je nutné pro každou značku vytvořit nový objekt, jak je znázorněno na obrázku 24.



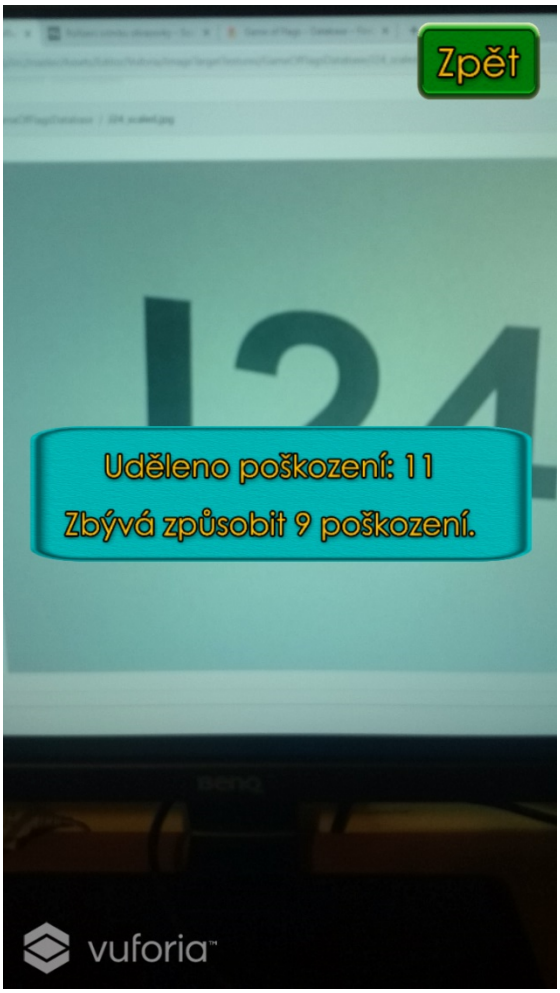
Obr. 24 - Unity, vytvoření objektů podle značek, zdroj: vlastní

Po kliknutí na útočení nebo obranu věže je zapnuta kamera zařízení a zobrazen nápis informující hráče, na jaké označení místnosti má kameru namířit. Po úspěšné detekci je zobrazena vybraná věž, zbývající výdrž a zbývající čas do vykonání vybrané akce. Pokud během odpočtu zařízení ztratí danou značku, je odpočet vrácen do základní podoby a hráč je informován o ztrátě značky.



Obr. 25 – Aplikace, Zobrazení nápisu a objektu, zdroj: vlastní

Po úspěšném doběhnutí odpočtu jsou změněny informace o věži a hráči v databázi. Následně je zobrazeno informační okno s výsledkem dané akce. První zpráva obsahuje hodnotu vykonané akce, druhá zpráva hráče informuje o aktuálním stavu věže. Věž může být hráčem zabrána, poškozena, částečně opravena nebo zcela opravena. Obrázky 25 a 26 zobrazují popsané chování v simulovaném prostředí. V reálném prostředí bude hráč mířit na skutečné tabulky u dveří učeben.



Obr. 26 - Zařízení, Informační okno s výsledky z akce nad věží, zdroj: vlastní

Během zobrazení odpočtu jsou snímány okolní Wi-Fi a BLE signály. Snímání probíhá po celou dobu zobrazení odpočtu. Po dokončení snímání je vytvořen snímek obrazovky, který je poslán do Cloud Storage. Následně je získána adresa uloženého snímku a vytvořen otisk, který je poslán do Cloud Firestore. Tyto operace probíhají nezávisle na dalších akcích hráče a hráč o nich neví.

5.3.8 Oprávnění

Od verze Androidu 6.0 (api 23) má uživatel možnost odebírat a přidávat oprávnění aplikacím dodatečně. Aby aplikace měla potřebná oprávnění pro správnou funkci, je nutné při každém spuštění aplikace ověřit povolená oprávnění. (Android, 2019c) Při instalaci aplikace na starší verzi systému je nutné souhlasit s požadovanými oprávněními, jinak není aplikace nainstalována.

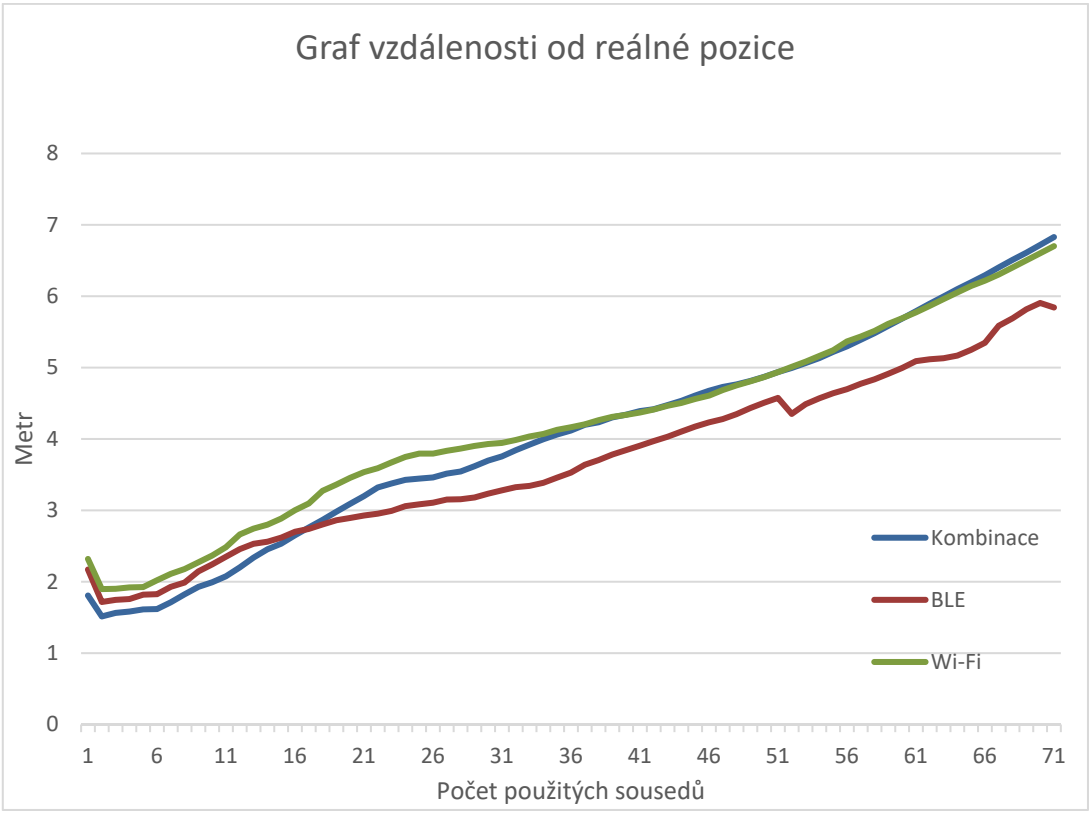
Aplikace potřebuje pro správné fungování několik oprávnění. Jedná se o přístup a změnu k Wi-Fi a Bluetooth senzorům, zjištění informací o telefonu a lokalizačním senzoru, které jsou následně poslány ve vytvořeném otisku, aby bylo možné případně rozlišit použitá zařízení v databázi otisků. Posledním oprávněním je použití kamery zařízení. Toto oprávnění kontroluje Vuforia automaticky a pokud hráč dané oprávnění neudělí, není schopen vykonávat akce nad věžemi.

6 Testování a zhodnocení výsledků

Aplikace byla testována především v uměle vytvořených podmínkách autorem této práce, případně v prostorách budovy J. Testování probíhalo na dvou zařízeních. Prvním zařízením bylo Sony Xperia Z1 Compact, které má operační systém Android verze 5.1.1. Druhým zařízením bylo Xiaomi Mi A2, které má operační systém Android verze 9.0. Plynulost aplikace byla především kontrolována na prvním zmíněném zařízení, které je více než 5 let staré.

Během vytváření lokalizační části aplikace bylo nutné ověřit, jaké nastavení algoritmu bude dosahovat nejlepších výsledků. Aby bylo možné určit přesnost používaného algoritmu, byla vytvořena jednoduchá aplikace, která porovnala všechny otisky mezi sebou a pro každý otisk za použití různého počtu sousedů určila rozdíl mezi vypočtenou a skutečnou polohou. Zároveň bylo ověřeno, jaký parametr *k* vyjadřující počet sousedů pro výpočet polohy je vhodné použít.

Testovány byl varianty za použití pouze Wi-Fi otisků, pouze BLE otisků nebo kombinace obou variant. Výsledné hodnoty byly uloženy do JSON soborů dle složek. Poté byly hodnoty zpracovány v programu Microsoft Excel verze z roku 2019.



Obr. 27 – Excel, Porovnání algoritmů pro lokalizaci, zdroj: vlastní

V obrázku 27 je patrné, že algoritmus v průměru dosahuje nejlepší odchylky při použití pouze 2 nejbližších sousedů. Nejlépe vychází kombinovaný algoritmus. Zajímavostí je, že při použití 17 a více nejbližších sousedů dosahuje nejlepších výsledků čistý algoritmus složky BLE. Pro parametr *k* byla tedy použita hodnota 2. Dosažené průměrné hodnoty v metrech jsou zobrazeny v tabulce 1.

BLE	Wi-Fi	Kombinace
1,72 m	1,90 m	1,51 m

Tabulka 1 – Excel, průměrná odchylka algoritmů v metrech, zdroj: vlastní

Při vývoji aplikace nastalo několik problémů, které bylo nutné vyřešit. Aby bylo možné použít Android plugin se závislostmi v Unity, je nutné použít speciální Gradle soubor, který je vygenerován v Unity. Tento soubor obsahuje předdefinované proměnné, za které Unity při sestavování aplikace doplní potřebné informace. (Unity, 2019g) Problém nastal při sestavování přihlašovací části. Implementace FirebaseUI závislosti obsahuje základní hodnoty pro vygenerovaný konfigurační soubor zobrazený v ukázce kódu 4. Při každém sestavení aplikace byly hodnoty z konfiguračního souboru nahrazeny základními hodnotami a nebylo možné použít přihlašování v aplikaci. Byly vzneseny dotazy na oficiálních stránkách Unity a portálu stackoverflow, kde vývojáři pomáhají ostatním vývojářům s problémy při vývoji aplikací. Ačkoliv nebyla získána odpověď na dotazy, řešením se ukázalo špatná pozice předdefinované proměnné. Po přesunutí proměnné doplňující závislosti projektu za závislosti použité v pluginu byl problém vyřešen.

Dalším problémem v plugin části bylo pomalé získávání prvních informací ze serveru. V horších případech aplikace čekala i několik minut, než se podařilo navázat spojení. Rychlost získání dat byla výrazně zvýšena po vypnutí perzistence získaných informací z databáze v zařízení.

V Unity nastal problém při animaci objektů z Blenderu. Animace chodícího hráče měla chybně orientované ruce. Problém byl odstraněn použitím jiné verze Blenderu, kterou Unity použilo pro přidání objektů z Blenderu do Unity. Také bylo zjištěno, že tento problém se projevuje pouze u některých verzí Blenderu, takže je nutné při změně verze Blenderu ověřit správné přidávání objektů do Unity.

Zdrojové kódy Android pluginu jsou dostupné na adrese https://bitbucket.org/leon-vojtech/gameofflags-plugin. Zdrojové kódy Unity části

jsou dostupné na adrese https://bitbucket.org/leon-vojtech/gameofflags-unity. V sekci stažení se nachází APK soubor poslední verze aplikace, který je možné stáhnout do zařízení s operačním systémem Android.

7 Závěry a doporučení

Tato práce se zabývala implementací grafického frameworku pro geolokační mobilní hru. Nejprve byla provedena rešerše dvou nejznámějších možností pro tvorbu her pro operační systém Android. Následně byla provedena analýza předešlé aplikace a nové aplikace. Poté byly popsány návrh a implementace aplikace, a nakonec byly zmíněny problémy při vývoji aplikace. Pro tvorbu grafické části byl zvolen engine Unity. Výsledná aplikace je určena pro operační systém Android verze 4.4 a vyšší. Zadání práce se podařilo splnit a výsledná aplikace je plně funkční.

Po zapnutí aplikace je uživateli zobrazena interaktivní herní mapa s věžemi a ostatními hráči, je určena poloha zařízení a jsou zobrazeny ukázkové hodnoty vlastností hráče. Hráči se dělí do dvou frakcí. Po přihlášení je hráč schopen vykonávat akce nad věžemi. Během akcí jsou tvořeny otisky okolních sítí, které jsou následně poslány do databáze. Po dokončení akce získá hráč určité zkušenosti, které po dosažení určitého počtu zvýší úroveň hráče a umožní vylepšit schopnosti.

Získané otisky jsou pouze uloženy v databázi. Možným rozšířením aplikace je kontrola získaných otisků podle uložených snímků v Cloud Storage. Pokud by nějaký hráč vykonal akci podvodem, byl by varován, případně by byl jeho účet zablokován. Ověřené otisky by následně bylo možné použít pro určení polohy zařízení.

Dalším možným rozšířením je přidání více budov Univerzity Hradce Králové. Aplikace nyní obsahuje 4 patra budovy J. Pro přidání další budovy by bylo nutné vytvořit databázi s otisky pro určení polohy zařízení, získat plány pater, podle kterých by byly následně vymodelovány objekty a vytvořit nové terény.

8 Seznam použité literatury

AGRAWAL, Sunnet, 2018. Native Android in Unity. [online]. Medium 22.06.2018 [cit. 12.07.2019]. Dostupné z: https://medium.com/@agrawalsuneet/native-android-in-unity-8ebfb42edfe8

ANDROID, Developers, 2019a. Set the application ID. [online]. Android [cit. 25.07.2019] Dostupné z: https://developer.android.com/studio/build/application-id

ANDROID, Developers, 2019b. Save data in a local database using [online]. Android [cit. 28.07.2019] Dostupné z: https://developer.android.com/training/data-storage/room/

ANDROID, Developers, 2019c. Android 6.0 Changes [online]. Android [cit. 05.08.2019] Dostupné z: https://developer.android.com/about/versions/marshmallow/android-6.0-changes

BLENDER, 2019. Extrude — Blender Manual [online]. Blender [cit. 02.08.2019]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/duplicating/extrude.html

BRODKIN, Jon, 2013. How Unity3D Became a Game-Development Beast. [online]. Dice 03.06.2013 [cit. 10.07.2019]. Dostupné z: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

CASO, Giuseppe, Luca DE NARDIS a Maria-Gabriella DI BENEDETTO, 2015. A Mixed Approach to Similarity Metric Selection in Affinity Propagation-Based WiFi Fingerprinting Indoor Positioning. Sensors [online]. 15(11), 27692-27720 [cit. 20.07.2019]. DOI: 10.3390/s151127692. ISSN 1424-8220. Dostupné z: http://www.mdpi.com/1424-8220/15/11/27692

DARDARI, Davide, Pau CLOSAS a Petar M. DJURIC, 2015. Indoor Tracking: Theory, Methods, and Technologies. IEEE Transactions on Vehicular Technology [online]. 64(4), 1263-1278 [cit. 20.07.2019]. DOI: 10.1109/TVT.2015.2403868. ISSN 0018-9545. Dostupné z: http://ieeexplore.ieee.org/document/7042271/

DOWNIE, Clive, 2016. Evolution of our products and pricing [online]. Unity Technologies 16.06.2016 [cit. 10.07.2019]. Dostupné z: https://blogs.unity3d.com/2016/06/16/evolution-of-our-products-and-pricing/

EWALD, Janis Daniel Pascal, ORIZ, Eduardo, 2019. The Asset Store, new and improved, starting [online]. Unity Technologies 30.04.2019 [cit. 11.07.2019]. Dostupné z: https://blogs.unity3d.com/2019/04/30/the-asset-store-new-and-improved-starting-today/

FIREBASE, 2019a. Firebase authentication [online]. Firebase [cit. 21.07.2019] Dostupné z: https://firebase.google.com/products/auth/

FIREBASE, 2019b. Choose a database: Cloud Firestore or Realtime Database [online]. Firebase [cit. 21.07.2019] Dostupné z: https://firebase.google.com/docs/firestore/rtdb-vs-firestore

FIREBASE, 2019c. Add Firebase to your Android project [online]. Firebase [cit 25.07.2019] Dostupné z: https://firebase.google.com/docs/android/setup

FIREBASE, 2019d. Easily add sign-in to your Android app with FirebaseUI. [online]. Firebase [cit. 26.07.2019] Dostupné z: https://firebase.google.com/docs/auth/android/firebaseui

FIREBASE, 2019e. Read and Write Data on Android [online]. Firebase [cit. 27.07.2019] Dostupné z: https://firebase.google.com/docs/database/android/read-and-write

FIREBASE, 2019f. Add data to Cloud Firestore [online]. Firebase [cit. 27.07.2019] Dostupné z: https://firebase.google.com/docs/firestore/manage-data/add-data

FIREBASE, 2019g. Upload Files on Android [online]. Firebase [cit. 28.07.2019] Dostupné z: https://firebase.google.com/docs/storage/android/upload-files

FIREBASE, 2019h. Download Files on Android [online]. Firebase [cit. 28.07.2019] Dostupné z: https://firebase.google.com/docs/storage/android/download-files

GAO, W., S.W. KIM, H. BOSSE, et al., 2015. Measurement technologies for precision positioning. CIRP Annals [online]. 64(2), 773-796 [cit. 20.07.2019]. DOI: 10.1016/j.cirp.2015.05.009. ISSN 00078506. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0007850615001481

GOOGLE, 2019a. Google Identity Toolkit [online]. Google [cit. 15.07.2019]. Dostupné z: https://developers.google.com/identity/toolkit/

GOOGLE, 2019b. ARCore – Quickstart for Android [online]. Google [cit. 20.07.2019]. Dostupné z: https://developers.google.com/ar/develop/java/quickstart

GOOGLE, 2019c. The Google Services Gradle Plugin. [online]. Google [cit. 25.07.2019] Dostupné z: https://developers.google.com/android/guides/google-services-plugin

GOOGLE, 2019d. Cloud Computing Services | Google Cloud [online]. Google [cit. 28.07.2019] Dostupné z: https://cloud.google.com/storage/

GRADLE, 2019. What is Gradle? [online]. Gradle [cit. 25.07.2019]. Dostupné z: https://docs.gradle.org/current/userguide/what_is_gradle.html

JOE, 2019. Exporting a project to Gradle and building/deploying on Android Studio – Unity. [online]. Unity Technologies 28.06.2019 [cit. 13.07.2019]. Dostupné z: https://support.unity3d.com/hc/en-us/articles/115005695763-Exporting-a-project-to-Gradle-and-building-deploying-on-Android-Studio

JOHN, 2019. Unity Game Development [online]. GameToolKits [cit. 10.07.2019]. Dostupné z: http://gametoolkits.com/unity-game-development-history.html

KOO, Jahyoung a Hojung CHA, 2011. Localizing WiFi Access Points Using Signal Strength. IEEE Communications Letters [online]. 15(2), 187-189 [cit. 20.07.2019]. DOI: 10.1109/LCOMM.2011.121410.101379. ISSN 1089-7798. Dostupné z: http://ieeexplore.ieee.org/document/5672458/

LI, Dong, Baoxian ZHANG a Cheng LI, 2016. A Feature-Scaling-Based k -Nearest Neighbor Algorithm for Indoor Positioning Systems. IEEE Internet of Things Journal [online]. 3(4), 590-597 [cit. 20.07.2019]. DOI: 10.1109/JIOT.2015.2495229. ISSN 2327-4662. Dostupné z: http://ieeexplore.ieee.org/document/7307751/

SEANFELIPE, 2019. Importing Blender models in LibGDX [online]. GitHub 04.02.2019 [cit. 07.07.2019]. Dostupné z: https://github.com/libgdx/libgdx/wiki/Importing-Blender-models-in-LibGDX

SKUPNIK, Radosław, 2018. Project Setup Gradle [online]. GitHub 23.09.2018 [cit. 10.07.2019]. Dostupné z: https://github.com/libgdx/libgdx/wiki/Project-Setup-Gradle

STEVENSON, Doug, 2018. What is Firebase? The complete story, abridged. - Firebase Developers [online]. Medium 25.09.2018 [cit. 21.07.2019] Dostupné z:

https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

UNITY, 2019a. Programming in Unity [online]. Unity Technologies [cit. 10.07.2019]. Dostupné z: https://unity3d.com/programming-in-unity

UNITY, 2019b. Scripting API: Physics.Raycast. [online]. Unity Technologies [cit. 11.07.2019]. Dostupné z: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

UNITY, 2019c. Creating and editing Terrains. [online]. Unity Technologies [cit. 12.07.2019]. Dostupné z: https://docs.unity3d.com/Manual/terrain-UsingTerrains.html

UNITY, 2019d. Building apps for Android. [online]. Unity Technologies [cit. 13.07.2019]. Dostupné z: https://docs.unity3d.com/Manual/android-BuildProcess.html

UNITY, 2019e. Unity – Manual: Animation States. [online]. Unity Technologies [cit. 30.07.2019]. Dostupné z: https://docs.unity3d.com/Manual/class-State.html

UNITY, 2019f. Unity – Manual: Layers. [online]. Unity Technologies [cit. 31.07.2019]. Dostupné z: https://docs.unity3d.com/Manual/Layers.html

UNITY, 2019g. Gradle for Android. [online]. Unity Technologies [cit. 07.08.2019]. Dostupné z: https://docs.unity3d.com/Manual/android-gradle-overview.html

VOJTĚCH, Leon, 2016. Gamifikace sběru fingerprintů bezdrátových sítí [online]. Hradec Králové [cit. 10.07.2019]. Dostupné z: https://theses.cz/id/zvucw2/

VUFORIA, 2019. Vuforia Engine Supported Versions. [online]. Vuforia [cit. 20.07.2019]. Dostupné z: https://library.vuforia.com/articles/Solution/Vuforia-Supported-Versions

WOITASCHEK, Paul, 2017. Tile maps [online]. GitHub 17.12.2017 [cit. 06.07.2019]. Dostupné z: https://github.com/libgdx/libgdx/wiki/Tile-maps

ZECHNER, Mario, 2013. Goals and Features [online]. LibGDX [cit. 05.07.2019]. Dostupné z: https://libgdx.badlogicgames.com/features.html

ZECHNER, Mario, 2014. libGDX 1.0 released [online]. LibGDX 20.04.2014 [cit. 05.07.2019]. Dostupné z: https://www.badlogicgames.com/wordpress/?p=3412

ZHOU, Cheng, Jiazheng YUAN, Hongzhe LIU a Jing QIU, 2017. Bluetooth Indoor Positioning Based on RSSI and Kalman Filter. Wireless Personal Communications

[online]. 96(3), 4115-4130 [cit. 20.07.2019]. DOI: 10.1007/s11277-017-4371-4. ISSN 0929-6212. Dostupné z: http://link.springer.com/10.1007/s11277-017-4371-4

9 Přílohy

Přílohou této práce je CD obsahující:

- Zdrojové kódy Android pluginu
- Zdrojové kódy Unity aplikace
- JSON soubor pro naplnění Realtime databáze – věže
- Text práce
- APK soubor pro nainstalování aplikace na zařízení

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Vojtěch Leon	Jiskrova 672, Úpice	I1600894

TÉMA ČESKY:

Implementace vybrané grafické knihovny pro geolokační mobilní hru

TÉMA ANGLICKY:

Implementation of Particular Graphics Library into a Geolocation Mobile Game

VEDOUCÍ PRÁCE:

Ing. Pavel Kříž, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Provést rešerši existujících grafických knihoven a frameworků pro Android. Zaměřit se na takové, které by umožnily vývoj hry ve stylu Pokemon Go nebo Ingress, kde je hlavním prvkem herní mapa s polohou hráče a dalšími herními předměty. S pomocí vybraného řešení (případě více) re-implementovat a rozšířit původní hru Game of Flags určenou pro sběr rádiových fingerprintů.

Osnova:
1. Úvod
2. Cíl
3. Přehled frameworků/knihoven
4. Analýza
5. Návrh a implementace
6. Testování a výsledky
7. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

https://docs.unity3d.com/Manual/index.html
https://libgdx.badlogicgames.com/documentation/
https://firebase.google.com/docs/
https://developer.android.com/docs

Podpis studenta: ..... **Datum:** 10.1.19

Podpis vedoucího práce: ..... **Datum:** 20.3.19