České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

Hra pro více hráčů s využitím jejich lokace pro operační systém Android

Multiplayer location-based game for Android OS

Bakalářská práce

Studijní program: Softwarové technologie a management

Studijní obor: Web a multimédia

Vedoucí práce: Ing. Martin Klíma, Ph.D.

Matyáš Racek

Praha 2018

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze dne 9.1.2018 .....

Poděkování

Rád bych poděkoval své přítelkyni a rodině za podporu při tvorbě práce. Dále děkuji vedoucímu práce, panu ing. Martinu Klímovi, Ph.D., za prvotní námět a cenné rady při návrhu a tvorbě práce.

Abstrakt

Tato práce se zabývá vývojem hry pro operační systém Android. Hra je určena pro více hráčů a využívá jejich polohu. Součástí práce je analýza problematiky a vybraných existujících titulů. Po zhodnocení technických nástrojů a požadavků následuje návrh samotné hry včetně prototypu uživatelského rozhraní a návrhu komponent celého systému. Průběh implementace a výsledný produkt jsou pak popsány v další kapitole. Poslední část práce se věnuje veškerému testování projektu a uzavírá ji popis a vyhodnocení testu s uživateli.

Klíčová slova: Android, hra, vývoj hry, aplikace, lokace, poloha

Abstract

This thesis describes development of multiplayer location-based game for Android OS. Document contains analysis of selected games and describes key problems and concepts for development of such games. After analysis of technical tools follows game design description with user interface protope and system design. Game implementation is then discussed in the next chapter with result presented in the end. Final part of the document summarizes testing of the game, including test with users.

Key words: Location-based game, Android, game development, app, location

Obsah

1 Úvod

V posledních několika letech dosáhly velké popularity location-based hry. Jedná se o tituly, které nějakým způsobem využívají polohu hráče ve skutečném světě. Takové hry nejenže polohu využívají, ale přímo ji potřebují k fungování herních mechanik. Velmi často se odehrávají na mapě skutečného světa, která je nějakým způsobem obohacena o virtuální herní prvky. Z podstaty se hrají na přenosných zařízeních, především mobilních telefonech a tabletech. Poloha se zjišťuje hlavně systémem GPS, nebo podle otisku Wi-Fi sítí, či mobilních GSM sítí. Systémy se navíc dají kombinovat pro zvýšení přesnosti. Hry mohou být navržené pro více hráčů, od obyčejného sbírání bodů nebo virtuálních předmětů a porovnávání se s ostatními, až po boj s ostatními hráči na virtuálním hřišti.

Cílem této bakalářské práce je vytvořit location-based hru pro platformu Android. V analytické části práce se věnuji typickým problémům, které se objevují při návrhu, vývoji a nasazení mobilních her obecně, poté se zaměřím speciálně na location-based hry. Následuje analýza konkrétních titulů, ke které jsem vybral dva známé, úspěšné tituly a jeden méně známý, který je ale konceptem příbuzný mému návrhu. V návrhové části dokumentu popisuji detailně princip hry a herní mechaniky. Následuje návrh uživatelského rozhraní aplikace a kapitolu uzavírá návrh architektury aplikace a komunikace mezi serverem a aplikací. V další části popisuji detaily implementace a důležité problémy, které jsem při implementaci řešil. V poslední kapitole popisuji veškeré testování systému, jako jsou interní jednotkové testy ale i test samotné hry s uživateli.

Práce se soustředí hlavně na technickou stránku problematiky a řešení implementačních problémů. Především problémy s kolizním algoritmem a synchronizací serveru s klienty jsem při implementaci řešil nejvíce. Hru jsem se rozhodl vytvořit pro platformu Android, protože s ní mám více zkušeností a chtěl jsem je dále rozvíjet.

2 Analýza

V následujících podkapitolách analyzujeme problematiku mobilních her obecně, poté konkrétně problematiku location-based her. V další části analyzujeme existující tituly a zhodnotíme jejich klady a zápory, ze kterých potom vyjdeme při návrhu vlastní hry.

2.1 Prostředí mobilních her

Hry pro mobilní telefony jsou specifickým odvětvím herního průmyslu. Především kvůli rozdílným podmínkám oproti klasickým počítačovým hrám. Nyní probereme klíčové charakteristiky, ve kterých se mobilní hry liší od klasických her pro počítače a herní konzole. Zaměřujeme se především na problémy v návrhu her samotných.

- **Prostředí** – Hráči svůj telefon nosí pořád sebou a hrají ve volných chvílích. Proto musí být hra uzpůsobená ke hraní v jakémkoli prostředí.
- **Časová flexibilita** – Hráč často neví, kdy bude muset hru přerušit (např. při čekání na vlak apod.), proto je vhodné, aby herní iterace byly krátké nebo aby se daly bez problémů přerušit a hráč se k nim mohl později vrátit. Interakce s aplikací samotnou by měla být také uzpůsobená těmto podmínkám. Hra by se měla rychle spouštět, rychle vypínat, přerušení a vracení do hry by mělo opět probíhat rychle a bez problémů. Mnohdy hráči opustí hru jen na okamžik, pokud chtějí například poslat zprávu, nebo na telefon někdo volá.
- **Ovládání** – Drtivá většina mobilních telefonů má dotykový displej. Ovládání je třeba navrhovat jiným způsobem než na počítačích nebo herních konzolích. Hra by se navíc měla přizpůsobit ztíženým podmínkám zmíněným výše. Hráč může být při hře v pohybu, a proto by měla být interakce co nejjednodušší, s nízkými požadavky na přesnost. Ideálním způsobem je návrhový vzor One touch control, tzv. Ovládání jedním dotykem. Příkladem může být jakákoliv hra typu “dash” - nekonečná hra, kde se postava hráče neustále pohybuje, nezávisle na hráčově interakci. Hráč sleduje hru, v potřebných momentech se dotkne obrazovky a postavička například vyskočí nebo se vyhne překážce. V některých známých titulech bývá obohaceno o gesta (Např. Temple Run, Subway Surfers).

Dalším často používaným návrhovým vzorem je způsob ovládání, kdy má hráč provést jednu interakci, která způsobí hodně akce ve virtuálním světě. Poté hráč nějakou dobu čeká a sleduje dopady interakce v herním světě. Tato interakce bývá často vázána podmínkami na přesnost a v takovém případě je vhodné, aby nebyla limitována časem. Například v populární hře Angry Birds hráč vystřelí a poté poměrně dlouho čeká na výsledek své akce. Většinu času hry tedy interakce neprobíhá a hráč sleduje dění ve hře.

Vzhledem k výše uvedenému lze soudit, že hráči už cíleně hledají hru, která má výše uvedené charakteristiky. U mobilních her je potřeba se zaměřit na první dojem mnohem více než u desktopových titulů. Zatímco titulům na desktop hráč dává větší šanci a prostor, zpravidla protože už do hry investoval svůj čas na instalaci nebo i peníze na koupi. Hry na telefon čas ani peníze

navíc z velké většiny nevyžadují. Hráč může hru stáhnout a spustit během několika minut. Během dalších minut ji může otestovat a pak se rozhodnout, zda ji ponechá nebo odinstaluje.

Hra by tedy měla v tomto procesu zapůsobit co nejlépe. Do tohoto dojmu se počítá cokoliv, co ovlivní prožitek hráče při interakci s aplikací. Pokud je například aplikace příliš velká a musí se tedy dlouho stahovat, nebo hráči spotřebovává limitovaná data, už to může hráče odradit. Ideálně má tedy hra mít velikost, aby se rychle stáhla, rychlé spuštění, rychlé pochopení hry. V prvních sekundách by měl hráč pochopit herní mechaniky a dobře si je vyzkoušet. Ideálně by tedy hra neměla zdržovat hráče žádným přihlašováním, registrací, volbou postavy nebo přezdívky a jiné, pokud to není nutné nebo není součástí mechanik hry.

Poznatky v této kapitole vycházejí zejména z přednášky Vladislava Speváka, vývojáře mobilních her, která proběhla na ČVUT FEL v rámci předmětu Počítačové hry [1].

2.2 Prostředí location-based her

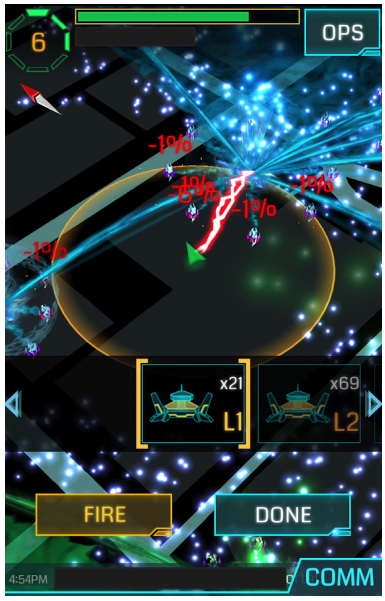
Zde probereme typické problémy, které vycházejí z principu location-based her. Opět se soustředíme hlavně na problémy spojené s návrhem hry. Poznatky vychází ze zkušeností a z výzkumného článku o problémech v location-based hrách [2].

- **Závislost na prostředí** - Podobně jako u obyčejných her uvedených výše je hra ovlivněna ztíženými podmínkami prostředí. U location-based her je situace složitější, protože ty jsou na prostředí přímo závislé. Cílem je hru přizpůsobit všem prostředím, ve kterých se hráč může nacházet. Typickým problémem je například situace, kdy hráč potřebuje dojít k lokaci na mapě, která je fyzicky nepřístupná. Dále může jít o prostředí, ve kterém se hráč pohybuje. Hráči v osamocených oblastech nemohou hrát tituly, které jsou uzpůsobené hraní ve městech.
- **Nároky na hráče** - Hra klade na hráče vyšší nároky ve srovnání s obyčejnými hrami. Především se jedná o fyzické nároky, jako je rychlost či jen schopnost pohybu. Pokud hra vyžaduje pohyb, může být náročné hrát ji delší dobu. Naopak jsou zde nároky na to se nepohybovat. Například v níže uvedeném Ingress, pokud chce hráč uspět při dobývání portálu, potřebuje dojít k místu, kde se portál nachází a pokoušet se ho dobývat právě na tomto místě.
- **Závislost na znalosti polohy** - Hry jsou často navrženy tak, že lokační data přímo vyžadují. Poloha hráče nemusí být pokaždé dostupná, jako je tomu například v budovách a hru tedy nejde hrát úplně všude. Zajímavou výjimku je Geocaching, který však není úplně hrou na telefon, ale spíše obecnou hrou, kde lze telefon využít. Hráči zde potřebují jen vědět souřadnice místa. To lze pak najít na mapě před hraním samotné hry.
- **Výdrž baterie** - Hry využívají GPS, internetové připojení a obsahují často náročnou grafiku. Všechny tyto funkce konzumují hodně prostředků a rychle vybíjí baterii. Systém GPS konkrétně je tím nejnáročnějším, protože za běhu nemůže telefon přejít do režimu spánku, čímž telefony běžně šetří velké množství energie [3]. Pokud hra navíc vyžaduje polohu často, technologie velmi rychle vybíjí baterii. Na novějších Android telefonech už je tato technologie úsporněji implementována.
- **GPS spoofing** - Technika kdy hráč falšuje svou lokaci a ve hře tak podvádí. Tímto hráč ovlivňuje negativně hru i ostatním hráčům.

2.3 Analýza existujících her

Pro analýzu jsme vybrali několik her se souvisejícím konceptem a zaměříme se na výše popsaných postupů v praxi a výsledky.

2.4 Ingress



Obrázek 1: Ingress

Ingress [4] je hra pro více hráčů, kteří jsou rozděleni do dvou nepřátelských frakcí. Hra zobrazuje mapu skutečného světa a přidává do ní virtuální prvky (Obrázek 1). Hráči jsou rozděleni do dvou týmů, chodí po světě a sbírají energii, kterou pak používají k hackování a obsazování portálů. Tři obsazené portály mohou hráči propojit a vytvořit mezi nimi trojúhelník obsazeného území. Hráči pak vzájemně na portály útočí, vylepšují je apod. Portály se mohou často vyskytovat na místech s významnými pamětihodnostmi a hráče tak vodí po zajímavých lokacích.

Hra hodně staví na sociální interakci. Hráči hrají spolu v týmu a musí spolupracovat. Jednou za čas se objeví anomálie - speciální událost na mapě, kde se hráči mají setkat fyzicky a ve virtuálním světě bojovat o dočasný portál [5]. Podobné principy hráče udržují ve hře díky kontaktu s ostatními hráči. To má ale i negativní stránku. Objevují se případy, kdy hráči v boji o portál nebo o jeho udržení pronásledují své protihráče nebo je dokonce i fyzicky ohrožují [6]. Na komunikačních portálech jako je Reddit se objevuje velké množství diskuzí, které o tomto jevu mluví.

Klady:

- Chytlavý herní systém
- Přívětivá vizuální stránka
- Po delším seznámení příjemná hratelnost
- Sociální stránka – Propojení s komunitou

Zápory:

- Hra je poměrně komplikovaná a zorientovat se v ní ze začátku není jednoduché. Navíc je už dlouho rozehraná a novým hráčům tak chvíli trvá, než se dostanou k nějakým úspěchům.
- Trvalé připojení k internetu a technologii GPS, což spolu s vypracovanou grafikou rychle vybíjí baterii.
- Pro úspěch ve hře musí hráč věnovat hodně úsilí. Hra není vhodná pro příležitostné hraní
- Sociální stránka – Stalking, hrozby násilím apod.

2.5 Pokémon GO

Obrázek 2: Pokémon GO

Pokémon GO [7] je pravděpodobně zatím nejúspěšnější location-based hra. Cílem hry je chytit co nejvíc Pokémonů. Hráči na virtuální mapě hledají a chytají příšerky z populárního seriálu (Obrázek 2). Pokud se hráč rozhodne nějakého chytit, učiní tak házením virtuálního Pokéballu na příšerku v rozšířené realitě. Hra obsahuje speciální místa, Pokéstopy a tělocvičny, která jsou opět často poblíž významných pamětihodností, nebo poblíž provozoven obchodních partnerů.

Oproti hře Ingress je hra v podstatě pouze pro jednoho hráče a sociální aspekty slouží především k motivaci hráčů mezi sebou. Jsou zde sice týmy a určité souboje v tělocvičnách, ale hra není postavená na boji proti ostatním hráčům. Hráči nemusí škodit ostatním hráčům, aby tím sami získali. Hru tak neprovází výše zmíněné problémy s nepřípustným chováním, tak jak to někteří hráči popisují u hry Ingress [6].

Hra je graficky velmi náročná. Trojrozměrná mapa, chytání pokémonů v rozšířené realitě a další grafické efekty dohromady zatěžují telefon do velké míry. Spolu s extenzivním využitím geolokačního systému a připojení k internetu hra rychle vybíjí baterii.

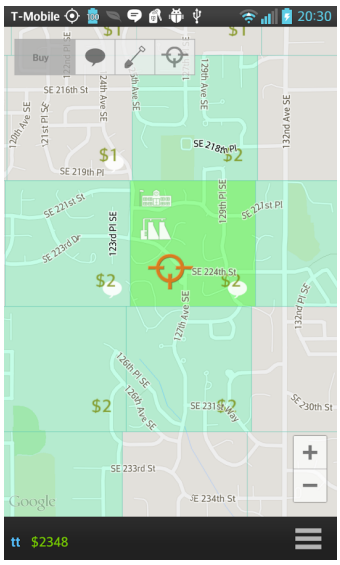
Klady:

- Neklade na hráče pohybové nároky, hráč si sám volí tempo hry
- Sběratelský systém je dobrou motivací pro hráče
- Sociální interakce udržují hráče ve hře

Zápory:

- Velká zátěž pro baterii a data - opět složitá grafika a trvalé připojení k internetu a GPS systému.
- Chyby a pády - postupně se lepší s novými aktualizacemi
- První spuštění hry je časově velmi náročné
- Hra se bez připojení GPS nedá vůbec spustit

2.6 GPS Tycoon



Obrázek 3: GPS Tycoon

GPS Tycoon [8] není sice tak populární jako předchozí zmíněné, ale svou jednoduchostí je blízká našemu konceptu. Hra nemá vysoké hodnocení, ale pro analýzu jsem ji vybral, protože dobře ilustruje některé techniky. Mapa světa je rozdělena na čtvercové sektory, veliké asi 500x500 metrů. Sektory mohou hráči nakupovat, ovšem koupit mohou pouze ten, ve kterém se fyzicky nacházejí. Sektor pak vynáší peníze hráči, který jej vlastní. Do sektoru je možné navíc ještě investovat a zvýšit tak jeho výnos. Hráči si navíc mohou sektory přebírat a proto lze peníze vložit i do ochrany sektoru.

Hra je nezávislá na prostředí. Na sektory je rozdělen celý svět a nakupovat lze tedy všude. Na hráče klade minimální pohybové nároky a tempo hry si opět volí hráč sám. Dá se spustit i bez připojení a uživatel tak má možnost se podívat do aplikace, i když hru nemůže hrát. Vady hry jsou především ve vyladění herních mechanik. Hráči musí koupit dost oblastí, než se dostanou k dalším mechanikám, jako jsou investice do oblastí a hra je tak ze začátku hodně monotónní.

Klady:

- Hra je jednoduchá a intuitivní
- Malé nároky na hráče
- Hra se dá spustit i bez GPS
- Méně náročná na baterii - za cenu strohého grafického zpracování

Zápory:

- Nestabilita, chyby
- Nevyladěné herní mechaniky
- Horší grafické zpracování

2.7 Analýza technických prostředků

Následuje popis a zhodnocení technických prostředků, které jsme uvažovali pro použití při vývoji hry. Nejprve se jedná o frameworky pro celou aplikaci, poté o knihovny pro práci s mapami, a nakonec serverová řešení.

2.7.1 Frontend

V následující části probereme několik možností knihoven pro implementaci samotné aplikace na klientské straně. Vybral jsem jednoduché knihovny zaměřené především na dvourozměrnou grafiku, protože naše hra má být dvourozměrná a nepotřebuje v principu moc grafických funkcí.

2.7.1.1 Android SDK

Hru jsem se rozhodl vytvořit pro platformu Android. Android architektura určuje základní strukturu aplikací a ty tak musí využívat některé komponenty SDK (např. Activity objekty), a to i když jsou aplikace napsané například nativně v C++ nebo v jiné knihovně. Z toho důvodu musí být všechny knihovny nakonec stejně integrovány s Android SDK.

Jednou z možností, jak hru implementovat je nepoužívat žádné specializované knihovny a napsat ji pouze s pomocí základního SDK, které je poměrně bohaté na funkcionality, a to i takové jako jsou animace a jiné složitější mechanismy. SDK je ale spíš orientované směrem k uživatelskému rozhraní než ke tvorbě her a jiné složitější grafiky. Trojrozměrnou grafiku nepodporuje žádným způsobem. I přesto se dá se základním SDK vytvořit hodně, pokud nejde o graficky zaměřené hry, jako jsou různé křížovky, karty, hlavolamy, sudoku a podobné tituly, které více využijí elementů uživatelského rozhraní než grafických efektů.

Výhody:

- Autor práce prostředí zná, není nutné tolik studovat dokumentaci
- Jednodušší integrace knihoven pro mapy, které jsou určené přímo pro Android prostředí
- Open-source, Výhodné licenční podmínky

Nevýhody

- Složitější grafické efekty není jednoduché implementovat

- Z podstaty není multiplatformní

2.7.1.2 LibGDX

LibGDX [9] je framework pro Android/iOS/HTML5/Blackberry. Používaný jazyk je Java, se kterým má autor nejvíce zkušeností. Oblíbený nástroj pro vývoj mobilních her. Knihovna je open-source a multiplatformní. Obsahuje veškeré potřebné funkce. Integrace knihoven pro API není tak jednoduchá jako v čistém Android prostředí. Oproti Android SDK je knihovna přímo zaměřena na vývoj her. Výše zmíněná hra Ingress využívá tuto knihovnu. I přesto, že je knihovna multiplatformní, uživatelé popisují i různé problémy s jednotlivými platformami. Našel jsem i různé způsoby, jak do knihovny integrovat mapy, ovšem žádná oficiální cesta není a zdrojů je málo a spíše uživatelských.

Výhody:

- Open-source, výhodné licenční podmínky
- Používá jazyk Java
- Multiplatformní knihovna

Nevýhody

- Nutnost naučit se pracovat s knihovnou
- Možné problémy při integraci mapovacích knihoven

2.7.1.3 Cocos2D-x

Cocos2D [10] je oblíbená nativní open-source knihovna pro Android/iOS, která je zaměřená na dvourozměrnou grafiku. Kód se píše v C++, Javascriptu či Lua. Velmi známá a hodně používaná knihovna. Díky nativnímu zaměření umožňuje hodně se zaměřit na výkon a stabilitu aplikace. Níže navržená služba Gamesparks poskytuje SDK pro integraci s Cocos2D-x. Knihovnu používají například známé tituly 2048, Geometry Dash nebo BadLand.

Výhody

- Open-source, dobré licenční podmínky
- Gamesparks SDK
- Multiplatformní knihovna
- Nativní rychlá knihovna

Nevýhody

- Programování na nižší úrovni abstrakce, více práce (v případě C++)
- Nutnost naučit se pracovat s knihovnou
- Integrace s různými Maps API je komplikovanější než v Javě a není k ní mnoho zdrojů

2.7.1.4 Zhodnocení

Pro vývoj aplikace jsem se rozhodl pro čisté Android prostředí bez knihoven. Především proto, že naše aplikace potřebuje pouze velmi málo z funkcí, které frameworky poskytují, a tak je vhodné

zvolit jednodušší nástroj. Pro obě další knihovny je nejasná možnost integrace API pro mapy a nechtěl riskovat ztrátu času kvůli problémům s integrací. Práci s oběma dalšími knihovnami bych se navíc musel naučit a po zvážení jsem dospěl k závěru, že přidaná hodnota tohoto úsilí by byla příliš malá. Ve hře budeme pouze zobrazovat mapu a na ní bude kreslit geometrické obrazce s jednoduchými animacemi. Částečným důvodem pro rozhodnutí byla také možnost do map kreslit přímo (vysvětleno v kapitole o implementaci), kterou poskytují API pro mapy.

2.7.2 API pro práci s mapami

V následující kapitole předvedu a zhodnotím různé API pro využití map v aplikacích. Co se týče map samotných, existují dva nejvíce používané zdroje. Těmi jsou komerční Google Mapy a nekomerční OpenStreetMaps. V následujícím seznamu srovnáme jednotlivé možnosti, které jsem při vybírání řešení uvažoval. Jedná se jak o zdroje map, tak o samotné knihovny pro Android, které je využívají.

2.7.2.1 Google maps API

Google Maps API [11] je jedním z nejpoužívanějších API pro práci s mapami. Největší výhodou těchto map je hotová infrastruktura pro stahování, caching a veškeré podstatné práce s mapami. Mapy se v poslední verzi dají i stylovat, podobně jako je tomu v obdobném Javascript API na webu. Další výhodou je možnost využití dalších souvisejících služeb Google, jako je například Places API. Google poskytuje knihovnu přímo pro Android a lze ho tak jednoduše integrovat do aplikace. Protože API je komerční, Google uplatňuje limity na jeho používání a za některé typy použití, zejména v komerčních aplikacích, vyžaduje zakoupit placený program.

Výhody:

- Velmi používané ověřené řešení, profesionální API
- Dobrá dokumentace i hodně zdrojů z komunity
- Stylování mapy
- Možnost kreslení na mapu
- Neomezené limity pro nekomerční aplikace
- Použití Google serverů, není nutné mít vlastní server pro mapy

Nevýhody:

- Komerční knihovna, není open-source
- Vyžaduje zakoupení licence pro některé funkce (například Asset Tracking), nebo podnikové a komerční aplikace

2.7.2.2 OpenStreetMap

Nekomerční možností jsou OpenStreetMaps [12], jejichž data využívají například i české Mapy.cz, které patří pod Seznam.cz. Zatímco Google mapy poskytují veškeré nástroje pro práci s mapami, OpenStreetMaps poskytují pouze mapy jako takové, ale neposkytují servery pro masové stahování [13] ani lokální SDK. Mapy lze stáhnout a ukládat v klientské aplikaci nebo ukládat na vlastním serveru. Další možností je využít třetí strany jako zdroj map, tedy někoho, kdo používá

OpenStreetMap data, ale vlastní infrastrukturu použitelnou pro masové stahování. Jako lokální knihovnu lze použít níže zmíněné MapsForge nebo OSMDroid.

Výhody:

- Open-source
- Neomezené použití i pro komerční účely

Nevýhody:

- Nutnost vlastního serveru, který bude hostit mapy nebo ukládat mapy lokálně
- Samo o sobě nemá SDK pro Android

2.7.2.3 MapsForge

MapsForge [14] je knihovna pro Android, která na rozdíl od ostatních zmíněných kreslí mapy lokálně vektorově (ostatní knihovny stahují již vykreslené obrázkové soubory). Knihovna používá vlastní formát souborů pro mapy a díky vlastnímu vykreslování lze jednoduše upravit vzhled mapy pomocí XML souborů. Knihovna je postavena pro mapy z OpenStreetMap, ale do používaného formátu lze přeložit i vlastní mapy.

Výhody:

- Open-source, volná licence (LGPL v3)
- Možné použití i pro komerční účely
- Vektorové kreslení

Nevýhody:

- Nutnost vlastního serveru, který bude hostit mapy nebo ukládat mapy lokálně
- Méně zdrojů informací oproti Google mapám

2.7.2.4 OsmDroid

Pro OpenStreetMaps existuje náhrada za třídu MapView z Google Maps Android API a tou je OSMDroid [15]. Osmdroid je knihovna, která částečně nahrazuje Google Maps API, ale využívá OpenStreetMap mapy. OSMDroid nenahrazuje kompletně Android API, je potřeba najít zdroj map, mít mapy offline, nebo získat mapy z třetího zdroje, které jsou ale také často komerční. Lze dokonce použít i Google Maps servery ale Google takové použití zakazuje v licenčních podmínkách.

Výhody:

- Open-source, volná licence

Nevýhody:

- Opět nutnost použít nějaký cizí zdroj map
- Open-source projekt, není tak spolehlivým řešením oproti Google mapám

2.7.2.5 Zhodnocení

Zvolil jsem nakonec Google Maps Android API. Hlavním důvodem byla jednoduchost používání bez nutnosti stavět další součásti systému zvlášť nebo je propojovat. Nepodařilo se mi najít poskytovatele map s výhodnějšími podmínkami, než má Google, který je zdarma pro nekomerční aplikace. Ukládat mapy na serveru jsem také zavrhl kvůli náročnosti řešení a nutnosti úložného prostoru. To by se dalo zčásti obejít tak, že bych mapy nahrával z OpenStreetMaps serveru Mapnik na svůj herní server pouze když by byly potřeba a poté je posílal uživateli, který o ně žádal. Mapnik totiž zakazuje používání serveru jako zdroje map pro koncové zákazníky, aby nebyl server přetížen a povoluje použití pouze pro prvotní nahrání map nebo vývoj aplikací. Takové řešení by bylo zbytečnou prací navíc, a proto jsem se nakonec rozhodl použít Google Maps API a v případě pozdější monetizace aplikace přistoupit ke komerčnímu plánu.

2.7.3 Backend

Hra, kterou vytváříme je typu MMO (Massive multiplayer online). Předpokládáme hraní v jednom velkém herním světě. Naše hra probíhá sice v reálném čase, není ovšem závislá na rychlé odezvě tolik jako více akční hry. Po serverové části tedy nevyžadujeme tak rychlou odezvu jako například akční FPS hry pro běžné počítače.

Budování vlastního řešení jsem hned z počátku zavrhl. Analyzoval jsme tedy několik služeb typu Backend-as-a-Service, především zaměřených na využití pro hry. Tyto služby nabízejí kompletní infrastukturu jako cloudovou službu. Vývojáři se tak mohou soustředit na tvorbu her či aplikací samotných a nemusí vytvářet a udržovat infrastrukturu, která s vlastní hrou nebo aplikací přímo nesouvisí.

Vzhledem k tomu, že v dnešní době potřebuje nějakou formu těchto služeb velké množství aplikací, existuje také hodně poskytovatelů. V analýze jsme se zaměřili na služby, které jsou vytvořené přímo pro hry.

2.7.3.1 GameSparks

Gamesparks [16] je služba typu Backend-as-a-Service (BaaS), která je jedním z největších hráčů na trhu služeb, zaměřených pro hry. Vývojář přistupuje k platformě z webového rozhraní, kde lze spravovat hru a přímo psát její logiku. Platforma má připravenou celou strukturu. Používá NoSQL databázi MongoDB. Platforma má modifikovatelný systém pro leaderboardy, virtuální měny, match-making (hledání vhodných hráčů pro společnou hru), týmy hráčů, propojení se sociálními sítěmi a více. Aplikační kód v cloudu se píše v JavaScriptu a organizuje do modulů a obslužných skriptů, které platforma nazývá Eventy. Webové rozhraní také nabízí různé analytické nástroje.

Platforma poskytuje SDK pro velké množství platforem, včetně Android a iOS SDK, JavaScript SDK, dále podporu pro Node.js, Unreal a Unity enginy a jiné. Nabízí i SDK pro Cocos2D-x knihovnu, kterou výše uvádíme. Integrace s LibGDX je možná s pomocí Android SDK.

Služba nabízí výhodné podmínky pro studenty. Společnost jsem kontaktoval a studentský program mi byl schválen.

Výhody:

- Kvalita služeb

- Uživatelská podpora
- Výhodný program pro studenty
- Trvalé připojení umožňuje obousměrnou komunikaci

Nevýhody:

- Dokumentace není úplná pro některé části API nebo je příliš stručná, na některých místech dokonce chybná

2.7.3.2 Playfab

Playfab [17] je opět služba vytvořená přímo pro hry. Mladší než Gamesparks, ovšem velmi podobně uzpůsobená. Opět nabízí SDK pro Android, iOS, JavaScript, Java, Cocos2D-x a mnoho dalších. Oproti Gamesparks platforma méně diktuje strukturu projektu. Uživatel může v podstatě všechen kód napsat do jednoho souboru. Platforma má v sekci pro skripty vyčerpávající seznam příkladů základních funkcionalit s dobře okomentovaným kódem, takže se vývojář velmi snadno zorientuje v novém prostředí a může snadno začít psát vlastní logiku. Stejně jako u GameSparks se kód píše přímo ve webovém rozhraní, ale je možné platfromu propojit s účtem na GitHubu a vyvíjet aplikaci lokálně, zatímco se PlayFab automaticky aktualizuje na nové verze.

Využití platformy je zdarma s určitým omezením prostředků. Uživatel si může připlatit hlavně za lepší uživatelskou podporu nebo pokud vyžaduje více prostředků. O limitech aplikace moc informací nepodává, ale pravděpodobně jsou velmi srovnatelné s Gamesparks.

Výhody:

- Kvalita služeb
- Dokumentace, tutoriály
- Dobrá pověst
- Zdarma pro běžné použití

Nevýhody:

- Absence některých funkcí oproti Gamesparks
- Nepoužívá stálé připojení

2.7.4 Zhodnocení

Obě platformy jsou si velmi podobné a co se týče funkcí srovnatelné. Rozhodl jsem se nakonec použít platformu Gamesparks, protože nabízí o něco více možností a pro naši hru má lepší způsob komunikace pomocí WebSocket protokolu. Díky tomu může podporovat obousměrnou komunikaci, a klient tak může dostávat informace o probíhající hře, aniž by přímo posílal požadavek.

2.8 Požadavky

Výše jsme popsali základní koncept hry. Nyní na základě analýzy a konceptu zformulujeme konkrétní požadavky na výsledek a vypracujeme detailní návrh. Požadavky jsou rozděleny podle standartního způsobu na funkční a nefunkční. Funkční požadavky popisují konkrétní funkce systému a nefunkční požadavky popisují kvalitativní omezení a nároky kladené na systém.

2.8.1 Funkční požadavky

- Vytvoření účtu – Aplikace umožní uživateli vytvoření účtu
- Autentizace – Aplikace umožní uživateli se autentizovat
- Autentizace pomocí sociálních sítí – Systém umožní uživateli se autentizovat pomocí sociálních sítí.
- Tutoriál – Hráč musí poznat veškeré důležité herní mechaniky při prvním průchodu
- Možnost hrát – Aplikace umožní autentifikovanému uživateli hrát hru.
- Leaderboard - Uživatel musí mít možnost vidět srovnání skóre s ostatními hráči
- Kontakt hráčů - Hráč musí mít možnost kontaktovat ostatní hráče.
- Sdílení v sociálních sítích – Uživatel má mít možnost sdílet svůj postup nebo samotnou aplikaci v sociálních sítích.
- Propojení se sociální sítí - Hráč může svůj herní účet propojit s účty v sociálních sítích
- Nalezení přátel - Hráč může ve hře najít své přátele ze sociálních sítí.

2.8.2 Nefunkční požadavky

- Velké množství uživatelů – Hra musí zvládnout velké množství připojených uživatelů
- Datová nenáročnost – Hra musí po síti přinášet malé množství dat. Limit stanovíme na maximálně 20 MB za hodinu běžného hraní.
- Výkonová nenáročnost – Hra musí být optimalizována pro šetření baterie přenosného zařízení

3 Návrh hry

V této části se zabýváme návrhem samotné hry. Jsou zde popsané všechny herní mechaniky i vzhled hry jako takové a scénáře používání. V této části také zmíníme otázky, které je potřeba během vývoje ještě zodpovědět při testování.

Princip hry je inspirován hrou Agar.io a podobnými tituly jako je Slither.io nebo Limax.io, které se v posledních několika letech staly populárními. Jedná se o jednoduché hry pro více hráčů, které se hrají v prohlížeči. Ve hře Agar.io ovládá hráč svojí jednoduchou bublinu a snaží se sníst bubliny ostatních hráčů, a tím zvětšovat svoji bublinu. Mým cílem bylo vytvořit hru s podobně jednoduchým konceptem, ovšem s využitím skutečné polohy uživatelů. Hra je zaměřena především pro mladší hráče.

3.1 Základní princip hry

Hráč chodí po mapě a obsazuje území. Může obsadit pouze území kolem své skutečné polohy. Obsazené území přináší hráči skóre. Hráč může obsadit území, které obsadil už jiný hráč a tím získat další bonusové skóre. Cílem hry je získávat co nejvyšší skóre a být mezi hráči nejlepší.

3.2 Mechanika

Hra je založená na jednoduché herní smyčce. Toto je základní smyčka bez jakýchkoliv mechanik navíc.

1. Obsazení oblasti
    a) Hráč obsadí oblast kolem své polohy.
    b) Za obsazení získá skóre. Skóre je přímo úměrné obsahu oblasti.
    c) Pokud hráč obsadí část oblasti jiného hráče, dostane kromě skóre za obsazenou oblast ještě bonus, který je úměrný obsahu ukradené oblasti. Ošizený hráč toto bonusové skóre naopak ztratí
2. Změna polohy a opět obsazení oblasti.
3. Po určitém čase oblast zmizí

3.3 Detailní návrh

V následujících podkapitolách rozpracujeme detailní návrh hry na základě konceptu, požadavků a analýzy. Představíme problémy, jejich možná řešení a doporučíme další postup.

3.3.1 Herní svět

Jak z principu vyplývá, hra se hraje proti ostatním živým hráčům. Zvolil jsme nakonec cestu MMO - massive multiplayer online, kdy po celém světě probíhá jedna velká hra a všichni hráči hrají proti

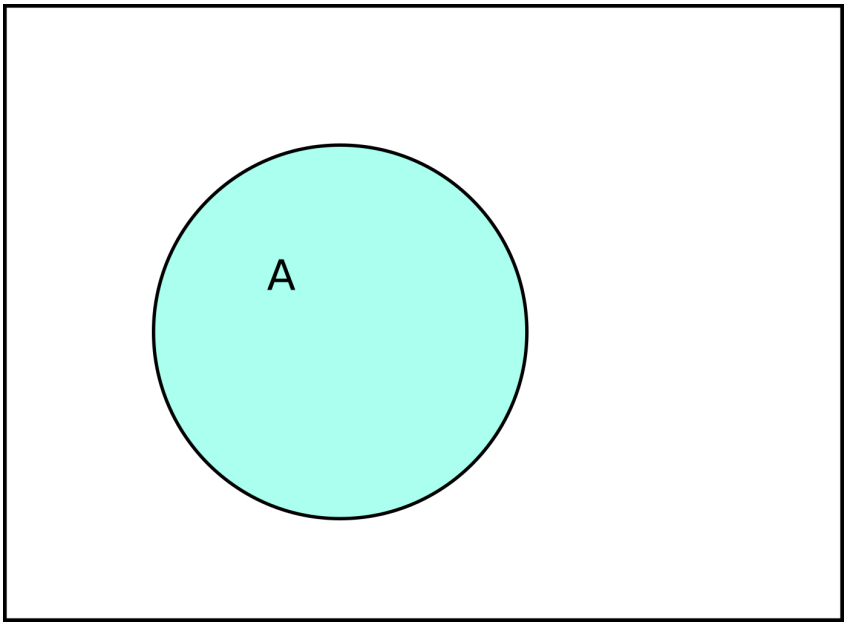
všem. Uvažoval jsem i možnost lokálních her, možnost založit vlastní instanci hry nebo týmovou hru.

Nakonec jsem tyto možnosti zavrhl, protože nemůžeme předpokládat, že hra bude tak masivně osídlená, aby si její hráči navzájem překáželi a její štěpení na menší instance by zbytečně separovalo hráče od sebe. Hraje se především na blízkých místech a najít větší množství hráčů kolem sebe není jednoduché. Separovat skupinu hráčů do zvláštní instance hry by tak mohlo osamostatnit nového uživatele v oblasti, ve které jinak protihráči jsou.

3.3.2 Mechanika obsazování

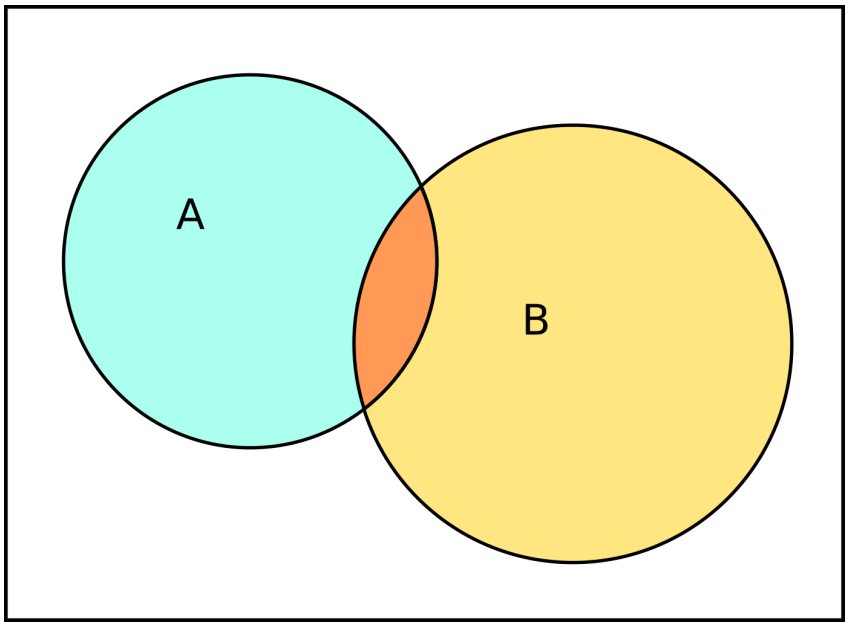
Jak je uvedeno v konceptu výše, hráč obsazuje oblasti jako kruhová okolí kolem své skutečné polohy. Mechaniku hry popisuje následující ukázkový scénář s obrázky pro jednodušší představu.

1. Mějme hráče A. V prvním kroku obsadí oblast na Obrázku 4 vyznačenou modře. Získá tím skóre, které je přímo úměrné obsahu této oblasti.



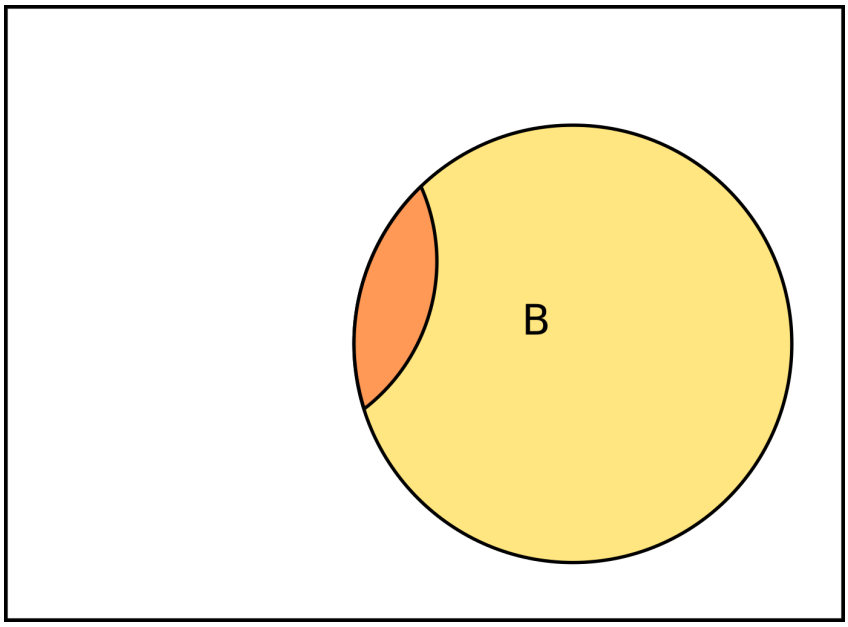
Obrázek 4: Oblast hráče A

2. Ve druhém kroku (Obrázek 5) vidíme hráče B, který obsadí oblast na obrázku vyznačenou béžově tak, že obsadí i kus oblasti, kterou původně obsadil hráč A. B získá skóre, které je opět přímo úměrné obsahu oblasti a k tomu získá navíc bonusové skóre, které je přímo úměrné obsahu průniku. Hráč A naopak ztratí stejně velkou část skóre.



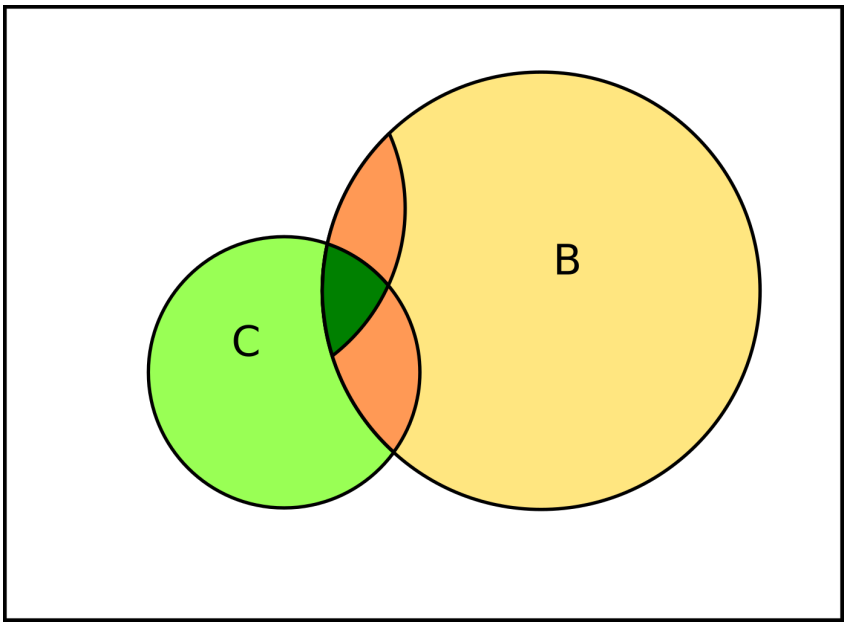
Obrázek 5: Oblast hráčů A a B

3. V dalším kroku (Obrázek 6) teritorium hráče A zmizí a na místě už je jen obsazená oblast hráče B. Skóre hráče B odpovídá součtu obsahu kruhu, který obsadil a obsahu průniku s již zmizelým kruhem hráče A.



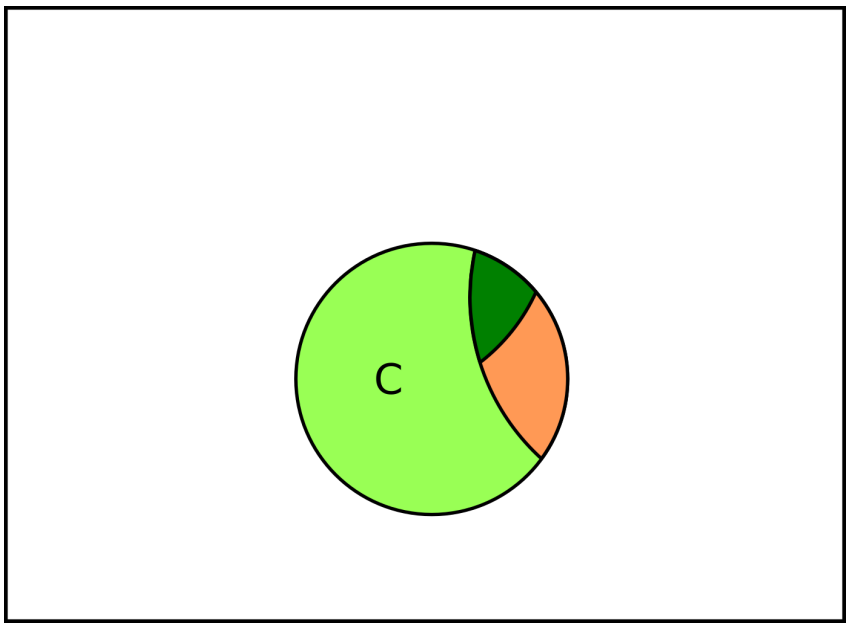
Obrázek 6: Oblast hráče B s průnikem navíc

4. Nyní přichází hráč C (Obrázek 7) a obsadí oblast ohraničenou převážně zeleným kruhem. Skóre hráče C je pak součtem obsahu kruhu, který obsadil a obsahu průniku se všemi zbývajícími oblastmi. Při ukrajování cizích území tedy hodnota jednotlivých útržků narůstá. Tmavě zelená oblast na obrázku 4 je například složena ze tří vrstev a má tak vyšší hodnotu na plochu než všechny ostatní.



Obrázek 7: Oblast hráčů B a C

5. Oblast hráče B nyní mizí a ve hře zůstává pouze oblast hráče C (Obrázek 8).



Obrázek 8: Konečná oblast hráče C

Takto může hra pokračovat dále. Tímto jsme ukázali základní herní mechaniku.

3.3.3 Leaderboad

Ve hře bude zobrazeno pořadí hráčů. Hráč bude mít na výběr z několika tabulek.

1. Lokální - Pořadí hráčů v okolí podle aktuálního skóre. Tato možnost také dobře simuluje lokální hry, které jsme postupně v původním návrhu zavrhli.

2. Přátelé - Pořadí známých hráčů. Může se jednat o přátele ze sociálních sítí nebo přátele přímo ze hry.
3. Globální - Pořadí hráčů na celém světě podle aktuálního skóre. Hra se odehrává po celém světě, pořadí je tedy na místě, ale protože v některých místech mohou být hráči zvýhodněni, nechceme tomuto pořadí přidávat takovou důležitost.
4. Globální Highscore - Ve hře budeme zaznamenávat i hráčovo nejvyšší dosažené skóre. Tento způsob je hodně důležitý, protože zatímco obyčejné skóre je pomíjivé, což chceme kvůli novým hráčům a krátké herní době, rank se dá zlepšovat dlouhodobě a poskytuje hráči dlouhodobou motivaci a důvod se do hry vracet.

3.3.4 Tutoriál při prvním průchodu

Podle požadavků je potřeba hráči ukázat všechny podstatné herní mechaniky hned při prvním spuštěním. Protože naše hra je typu MMO a hraje se proti ostatním živým hráčům, není možné zaručit přítomnost protihráčů při prvním spuštění hry. Kvůli tomuto jevu zavedeme do hry umělého protihráče, který bude sloužit k osvojení herních mechanik pro nové hráče.

Pokaždé když se do hry přihlásí hráč, který v okolí nebude mít žádného protivníka, server automaticky vygeneruje umělého hráče, který bude podle určitého klíče chodit a hrát poblíž po určitou omezenou dobu.

3.3.5 Sociální aspekt

Důležitým součástí hry je sociální stránka. Z analýzy existujících her si můžeme povšimnout, že kontakt s lidmi prospívá úspěchu hry. Proto jsme také do požadavků zahrnuli možnosti propojení se sociálními sítěmi a možnost mezi hráči komunikovat. Ve hře vytvoříme možnosti komunikace hráčů a propojení se sociálními sítěmi. Hráč bude moct chatovat s kterýmkoli jiným hráčem.

3.4 Návrh uživatelského rozhraní

V následující části představíme návrh uživatelského rozhraní. Toto rozhraní je vytvořené čistě pro účely vývoje a nemá představovat výslednou podobu produktu. Hlavním cílem návrhu je stanovit rozvržení elementů uživatelského rozhraní a vztahů mezi částmi aplikace. Výsledek tohoto návrhu je objektem pro testování a případné přepracování.

Návrh je plnou verzí aplikace, ze které v této práci implementuji jen nejdůležitější prvky, aby byla hra dobře hratelná. Vyhnu se především implementaci sociálních funkcí, které jsou důležité pro aplikaci při vydání a propagaci, ale pro hraní samotné nejsou kritické.

3.4.1 Hlavní menu

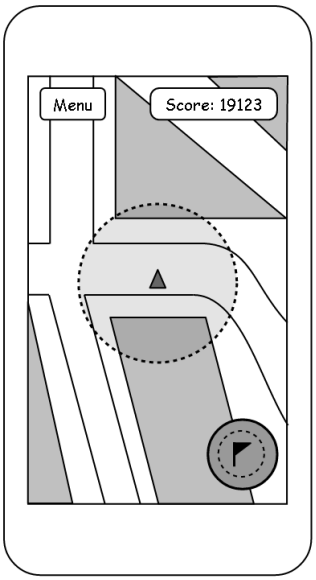
Obrazovka hlavního menu, která se objeví hned po startu hry (Obrázek 18). Tlačítko pro vstup do hry je nejdůležitější, proto je také největší. Ostatní tlačítka zavedou hráče k vedlejším funkcím jako je zobrazení tabulek s pořadími.

Obrázek 9: Hlavní menu aplikace

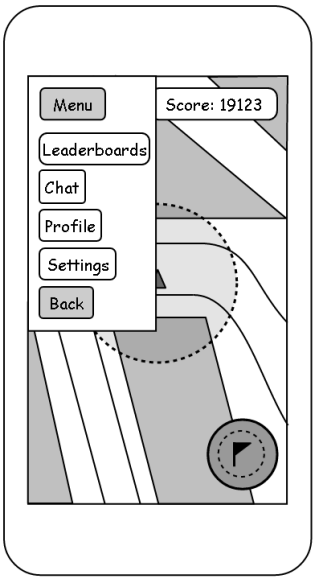
3.4.2 Herní obrazovka

Návrh obrazovky, pokud je hráč ve hře (Obrázek 10). Na pozadí je mapa s polohou, kde se hráč nachází. Hlavními prvky je plovoucí tlačítko vlevo dole, kterým hráč obsadí vyznačenou oblast a posuvník na hranici oblasti, kterým může hráč měnit její velikost. Cena oblasti se zobrazuje v bublině vedle. Stav hráčova účtu je vidět nahoře uprostřed obrazovky.

Rozmístění prvků jsem volil podle předpokládané četnosti jejich používání a podle Design Guidelines Android platformy, která nyní používá Material Design [18]. Z toho důvodu je například tlačítko pro obsazování vpravo dole jako takzvané Floating Action Button. Na dalším obrázku (Obrázek 11) je stejná obrazovka s otevřeným menu, které obsahem kopíruje hlavní menu aplikace, ale je menší.

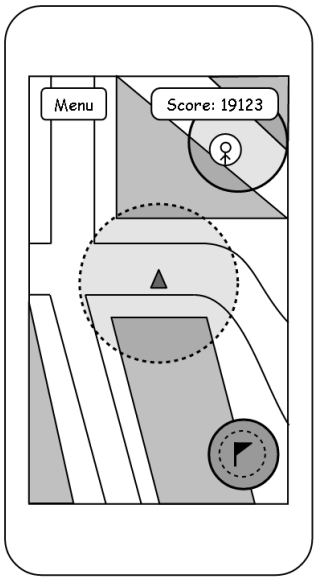


Obrázek 10: Herní obrazovka

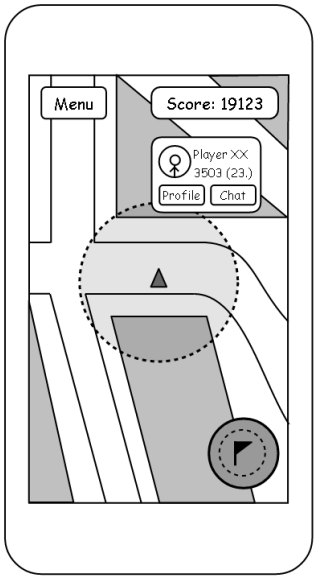


Obrázek 11: Herní obrazovka s otevřeným menu

Na obrázku 12 je vidět cizí oblast, která je označená ikonkou protihráče. Na ikonku se dá zmáčknout a otevře se okénko (Obrázek 13) s možností prohlížet profil nebo otevřít chat protihráče.



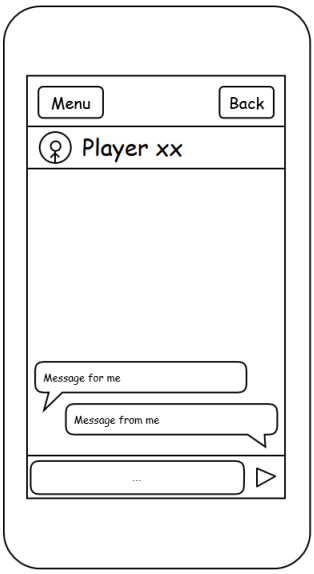
Obrázek 12: Herní obrazovka s protihráčem



Obrázek 13: Herní obrazovka s popiskem protihráče

3.4.3 Obrazovka chatu

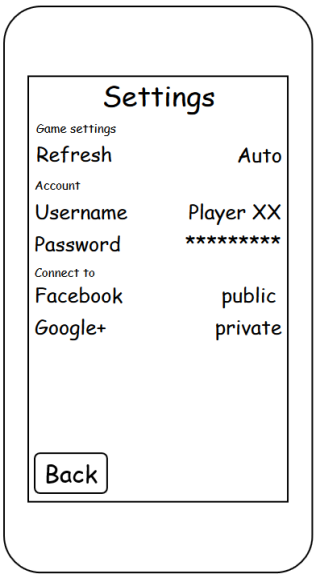
Obrazovka pro psaní zpráv mezi uživateli (Obrázek 14). Dle standartních konvencí pouze okénko s vláknem zpráv a možností napsat novou zprávu do okénka dole.



Obrázek 14: Chat

3.4.4 Obrazovka nastavení

Rozhraní pro nastavení aplikace (Obrázek 15). Opět dle standartních konvencí platformy Android pouze výčtové menu. V poslední části je možnost propojit účet hry s účty v sociálních sítích, spolu s možností profily zveřejnit ve hře, nebo ponechat neveřejné.

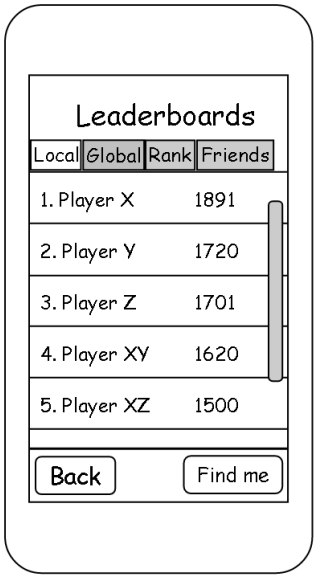


Obrázek 15: Nastavení

3.4.5 Leaderboard

Obrazovka s pořadím hráčů ve hře (Obrázek 16). Hráč může vybírat mezi lokálním a globálním pořadím, pořadím podle ranku hráčů a poté pořadí svých přátel. Zde se jedná o přátele ve hře i ze

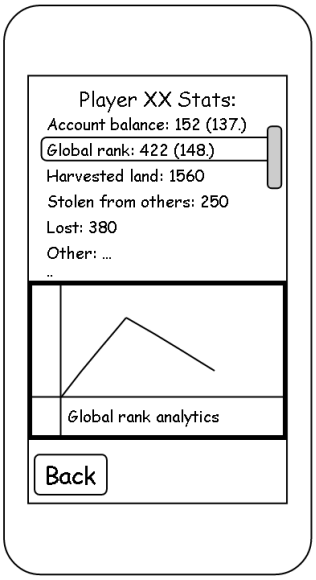
sociálních sítí, pokud uživatel propojil hru s nějakou z nich. V dolní části obrazovky je taky tlačítko s možností najít uživatele ve zvoleném listu.



Obrázek 16: Leaderboard

3.4.6 Statistiky hráče

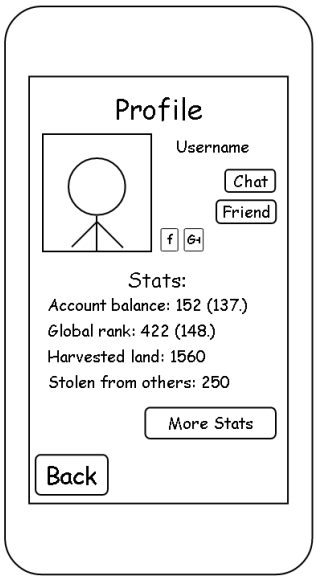
Obrazovka s hráčovými herními údaji s možností zobrazit graf zvolené hodnoty v závislosti na čase (Obrázek 17).



Obrázek 17: Statistiky hráče

3.4.7 Profil hráče

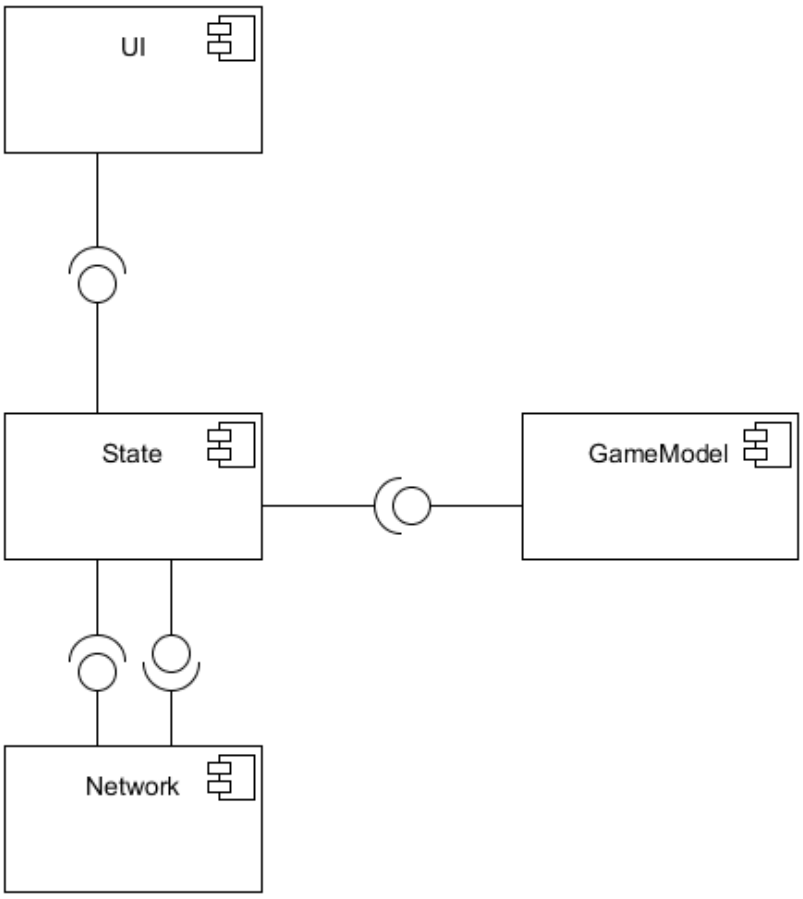
Obrazovka s profilovými údaji hráče (Obrázek 18). Na obrázku je profil protihráče, kde lze přejít do chatu nebo hráče přidat do přátel listu přátel. V dolní části obrazovky je taky výběr z hráčových herních statistik s možností zobrazit více, která uživatele přesměruje na plnohodnotnou stránku se statistikami.



Obrázek 18: Profil hráče

3.5 Návrh architektury aplikace

Při návrhu jsem se snažil co nejvíce odstínit Android prostředí od kódu vlastní hry. Zjednodušený model je vymodelován na obrázku níže (Obrázek 19). Komponenta UI reprezentuje uživatelské rozhraní v Android prostředí. State je komponenta která udržuje stav aplikace, komunikuje a udržuje lokální model hry, kterým je komponenta GameModel a deleguje požadavky na server komponentě Network, která se stará o internetové připojení a komunikuje se serverovou částí aplikace. Network poté zpět komunikuje s komponentou State a předává výsledky komunikace.



Obrázek 19: Model architektury aplikace

3.6 Návrh architektury na straně serveru

V následujících kapitolách popíšeme strukturu systému na straně serveru. Používáme platformu Gamesparks a proto je architektura do určité míry předepsána. Při návrhu jsem uvažoval více možností, a nakonec zvolil takovou, která nejefektivněji využívá poskytnuté nástroje. Popisuji zde databázi a práci s leaderboardy. Další komponenty na serveru souvisí s komunikací, a proto je popisuji v další kapitole o komunikaci.

3.6.1 Databáze

Strukturu na straně serveru jsem navrhoval s ohledem na používanou databízi. Gamesparks používá databázi Mongo. Mongo je dokumentová NoSQL databáze, která udržuje data v kolekcích JSON dokumentů, přesněji používá BSON, což je binární verze JSON. Databáze nepoužívá žádné schéma, a lze tedy schéma měnit za běhu, nebo do kolekcí přidávat v podstatě jakákoli data. Mongo neumí dobře pracovat se vztahy, nepodporuje referenční integritu, není dobrá pro komplikované JOIN dotazy. Je tedy výhodnější všechna související data skladovat v jednom JSON dokumentu i za cenu replikace dat v databázi.

Celá hra je vlastně kolekce právě obsazených teritorií. Objekt teritoria je složen z pozitivních a negativních kruhů. Teritorium je pak definováno jako plocha průniku pozitivních kruhů a od něhož odečteme negativní kruhy (jedná se o množinový rozdíl). Dále dokument obsahuje identifikátor uživatele, čas vzniku a životnost.

Správa uživatelů je součástí samotné platformy. Pro potřeby hry ji mohu použít bez modifikací ji zde nepopíšu.

3.6.2 Leaderboardy

Pro hru v platformě vytvořím několik leaderboardů. V platformě se vytváří pomocí formuláře a data pro ně se sbírají pomocí označených událostí. Data se pak ukládají v částečných součtech (v platformě Running Totals) a zobrazují v leaderboardech. Leaderboardy se také dají dělit podle atributů a tímto způsobem dosáhnout lokalizace podle měst, zemí či měsíců, ale i podle jakéhokoliv jiného atributu. S leaderboardy se dá pracovat i z kódu v platformě.

3.7 Návrh komunikace

Služba Gamesparks, kterou jsem se rozhodl využít, používá ke komunikaci WebSocket protokol. Tento protokol, na rozdíl od často používaného REST rozhraní, udržuje stálé TCP spojení mezi Klientem a Serverem. Uživatel se připojí a poté posílá různé požadavky. Velké množství požadavků už má platforma přímo zabudovaných a není nutné je vytvářet. Jedná se především o různé autentifikace ale také dotazy na Leaderboardy a další funkcionality systému. Požadavky posílá klient. Platforma navíc specifikuje takzvané Messages. To jsou naopak zprávy, které posílá server klientovi. Klient na ně dále neodpovídá.

Na začátku komunikace klient začne spojení a pokud je úspěšný, vytvoří se socket. Komunikace začíná autentifikací Gamesparks pomocí API klíče, tak aby se mohlo k API přistoupit pouze z autorizované aplikace. Poté může klient poslat uživatelský autentifikační požadavek, nebo požadavek na vytvoření účtu. Nyní lze přes socket posílat požadavky na server nebo přijímat zprávy ze serveru.

Samotná herní komunikace je pak navržena níže popsaným způsobem.

1. Klient pošle Refresh požadavek
2. Server pošle odpověď, ve kterém pošle všechny právě obsazená území
3. Server zároveň hráče přidá do skupiny posluchačů, tedy hráčů, kteří jsou připojeni a chtějí dostávat informace o změnách ve hře.

Spojení zůstává a nyní záleží na událostech ve hře. Dále tedy popíšeme průběh při obsazení.

1. Hráč se v aplikaci pokusí zabrat území.
2. Aplikace pošle Claim požadavek na server.
3. Server vyřeší herní logiku
4. Server pošle hráči zpět odpověď.
5. Server pošle zprávu o obsazení všem posluchačům.

Při návrhu hry jsem se snažil vymyslet co nejvíce deterministickou reprezentaci hry tak, aby hra mohla po síti přenášet minimální množství dat a klient mohl zbytek dopočítat sám. Z toho důvodu některé události neposílá server přímo. Je tím například smazání teritoria. Při rozesílání zprávy o obsazení pošle server také životnost teritoria, aby ho aplikace ve správné chvíli smazala v lokální kopii hry. Při smazání teritoria se ale mění i skóre a umístění hráčů na Leaderboardu. Aby to aplikace mohla spočítat, musela by mít data o všech hráčích ve hře, což zjevně nelze uskutečnit. Proto navíc server při smazání teritoria posílá zprávu Score Update, kde hráč získá své aktuální skóre a rank.

Pokud se hráč ze hry odpojuje, pošle požadavek Unsubscribe, který ho na straně serveru odhlásí z listu posluchačů.

Pokud se spojení přeruší, klient po opětovném navázání spojení opět pošle Refresh požadavek.

4 Implementace

V následujících řádcích popíšu implementaci aplikace a serverové části hry, tak jak jsem ji nakonec vytvořil. Z aplikace jsem implementoval jádro hry beze všech sociálních funkcí navíc. Během vývoje se také změnilo uživatelské rozhraní, které jsem více přiblížil stylu Material.

K popisu jsem vybral především problémy, které zabraly při implementaci nejvíce času a úsilí. Především návrh a implementace kolizních algoritmů zabraly z celkového času práce největší část.

4.1 Aplikace

Snažil jsem se co nejvíce odstínit Android prostředí od aplikační logiky, zvláště kvůli jednoduššímu testování a nezávislosti kódu. Testování Android komponent je složitější než testování jednoduchých Java tříd, protože je potřeba simulovat Android prostředí. Většina kódu aplikace je proto v jednoduchých Java třídách s obyčejnými JUnit testy. Architekturu jsem s menšími odchylkami dodržel tak jak je popsána výše v návrhu.

4.1.1 State

State je jedináček, který drží stav aplikace a je jádrem celé aplikační logiky. Komponenty uživatelského rozhraní tak k němu mají vždy přístup a není potřeba zavádět další instancovací mechanismy (v Android komunitě je to běžný způsob, jak udržovat globální stav aplikace). Zároveň State poskytuje rozhraní, na kterém lze poslouchat změny stavu aplikace. Komponenty uživatelského rozhraní, které mají omezenou životnost se tak mohou zaregistrovat a reagovat na změny stavu. Po skončení životnosti se komponenty opět odhlásí.

4.1.2 Network

State dále komunikuje s komponentou Network, která je reprezentována třídou NetworkController. Tato třída obaluje Gamesparks SDK, aby mohlo být v případě potřeby jednoduše zaměněno za jinou službu. Třída je opět jedináček i když to není nutné, ale protože Gamesparks SDK poskytuje také jedináčka, rozhodl jsem se nechat SDK ve správě vždy jen jedné instance mé třídy, aby nedocházelo k neočekávanému chování v případě více instancí, komunikujících s Gamesparks SDK. NetworkController poskytuje metody se zpětnými voláními.

4.1.3 GameModel

State udržuje také lokální model hry, kterým je v podstatě jen kolekce právě obsazených teritorií. Teritorium je reprezentováno jako kolekce pozitivních a negativních kruhů, a je definováno jako průnik pozitivních kruhů, od kterých jsou odečteny negativní kruhy. Model navíc udržuje i řádkovou reprezentaci teritorií, která se používá hlavně pro výpočet kolizí ale také pro vykreslení teritorií do mapy. Protože server posílá data pouze o jednotlivých akcích hráčů, musí si hra dopočítat kolize a aktuální stav hry sama.

4.1.4 Integrace map

Google Maps API je do aplikace zabudováno jako Fragment v hlavní MapActivity, která představuje hlavní obrazovku aplikace. Na mapu se dá kreslit více způsoby. Google Maps Android API nabízí možnosti překrytí mapy pomocí Overlay. Overlay jsou dva typy. Ground Overlay lze přichytit na mapu na určité souřadnice. API navíc pracuje s objektem jako by byl přichycený na skutečné zeměkouli a podle toho jej promítne a zakřiví. Pokud například přichytíme na mapu obrázek, na mapě bude zkresleně zobrazen. Dalším typem je Tile Overlay. Mapy v jsou distribuovány ve čtvercových blocích, nazvaných Tile. Každý blok se dá rozdělit na další čtyři bloky při přiblížení mapy. Tile Overlay dovoluje nakreslit svoje vlastní překryvné bloky přes bloky mapy.

Pokud se chceme vyhnout práci s mapu co nejvíce, můžeme také vytvořit svůj vlastní prvek UI (v Android prostředí nazvaný View) a jeho polohu měnit podle projekce mapy. Tím můžeme k mapě přichytit cokoliv, ovšem při každé změně mapy, když například uživatel mapu posune nebo přiblíží, musíme změnit polohu podle aktuální projekce mapy. Projekci reprezentuje Projection objekt z Google Maps Android API, který nabízí metody pro konverzi souřadnic ze systému mapy do souřadnic na obrazovce a obráceně. Vytvoření tohoto objektu je výpočetně náročné a v dokumentaci API se doporučuje metodu volat maximálně jednou za frame.

API nabízí i možnost nakreslit na mapu kruh, čáru nebo lomenou čáru. Tyto metody jsem nepoužil hlavně proto že promítají tvary na zeměkouli. Kruh nakreslený na mapě je brán jako úseč koule (API reprezentuje zeměkouli jako kouli). Nakreslená kružnice je tedy hrana této úseče promítnutá podle projekce. Ve hře ovšem obsazujeme kruhy až v promítnuté oblasti, a proto je potřeba nakreslit kruh jako kruh přímo na mapě. Zkreslení je sice velmi malé a v podstatě neznatelné, ale v principu by nebyl kruh zobrazen správně. Navíc aby mohlo API projekci spočítat, převede kruh na mnohoúhelník, a to je v našem případě zbytečné.

Ve hře používám kombinaci obou přístupů. Pro zobrazení existujících obsazených oblastí používám Tile Overlay a pro zobrazení kruhu kolem uživatele používám poslední zmíněnou metodu s projekcí.

4.2 Problém reprezentace teritorií na zeměkouli

Z charakteru hry vyplývá, že budeme muset jasněji definovat tvar teritorií a jejich reprezentaci vzhledem k zeměkouli. Pokud má hráč obsadit kruhovou oblast kolem své polohy, jedná se ve skutečnosti o kružnici na kouli (pokud aproximujeme zeměkouli koulí, v případě přesnějšího elipsoidu by byla definice útvaru komplikovanější). V této kapitole vyjmenuji uvažované varianty reprezentace a jejich výhody a nevýhody.

4.2.1 Pravá reprezentace

Pravou reprezentací se rozumí zmíněná reprezentace teritoria jako kruhu na kouli (použití sférické geometrie). V euklidovské trojrozměrné reprezentaci lze hranici teritoria vyjádřit jako jedinou hranu úseče koule. Teritorium je pak kulová část této úseče.

Výhody:

- Hráči na různých místech světa nemají žádnou výhodu ani nevýhodu, tato reprezentace je férová. Není potřeba zavádět žádné další mechanismy pro kompenzaci nevyrovnaných šancí mezi hráči.

Nevýhody

- Komplikované algoritmy kolize a výpočtu obsahu – Kolize teritorií vede na počítání průniku koulí s použitím mnoha výpočtů trigonometrických funkcí, které jsou výpočetně náročné.
- Nedeterministické řešení – Nutnost použít datový typ s plovoucí desetinnou čárkou
- Složitější vykreslování – Zobrazení v projekci mapy není kruh, je tedy nutné promítnout kruh do projekce mapy a zobrazit jiný tvar.

4.2.2 Projektovaná reprezentace

Další možností je definovat hru až v mapě, která vznikne projekcí z kulovitého povrchu země. Protože naprostá většina API využívá projekci Web Mercator, nabízí se navrhnout hru v této projekci. Tím se lze zbavit komplikovaných výpočtů ve sférické geometrii, ale protože projekce zkresluje povrch planety, kruh na mapě není kruhem na zeměkouli. Čím blíže pólům, tím zkreslenější tvar reprezentuje kružnice v projekci. Kruh u pólu na mapě zabírá na zeměkouli plochu s vejcovitým tvarem.

Výhody:

- Planární geometrie umožnuje jednodušší počítání kolizí i obsahu
- Jednodušší zobrazování, kruh lze nakreslit přímo stejně tak jako ostatní tvary vzniklé různými množinovými operacemi s kruhy.

Nevýhody:

- Zanedbává zkreslení mapy – Nutnost kompenzovat různé podmínky pro hráče na různých místech na světě.

4.2.3 Zhodnocení

Rozhodl jsem se nakonec použít druhou možnost a reprezentovat teritoria jako kruhy už v projektované mapě. Velkou výhodou nakonec byla reprezentace s použitím pouze celočíselných souřadnic. Díky tomu jsem mohl hru implementovat deterministicky a ušetřit velké množství přenášených dat.

4.3 Algoritmy kolize a obsahu

V následující části popisuji algoritmy, které jsou důležité z hlediska herní logiky.

- **Výpočet obsahu** - Hodnota obsazovaných území a potažmo hráčův zisk je přímo úměrný obsahu zabrané plochy. Musíme tedy umět spočítat obsah jakékoliv plochy, která může ve hře vzniknout

- **Detekce kolizí** - Při obsazování oblasti je potřeba zkontrolovat, zda se nová oblast nekříží s nějakou již existující.
- **Množinové operace, řešení kolizí** - Pokud je detekována kolize, potřebujeme spočítat průnik kolidujících oblastí a také rozdíl původní oblasti a průniku.

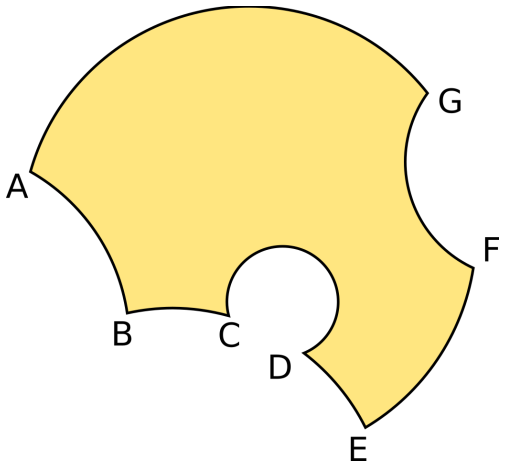
Během implementace jsem uvažoval dva různé přístupy k řešení těchto problémů. První je čistě analytický a druhý naopak bližší pixel-based přístupu, ale s optimalizovanější reprezentací teritorií jako řádků. V dalších podkapitolách postupně projdeme uvažované postupy.

4.3.1 Analytický výpočet obsahu

Vypočítat obsah obecného geometrického útvaru není lehký úkol. Následující algoritmus je inspirován podobným algoritmem pro počítání obsahu průniku libovolného množství kruhů, který jsem našel [19]. V našem případě můžeme využít několika faktorů, které vycházejí z principu hry. Všechny plochy ve hře vznikají množinovými operacemi s kruhy a takto vzniklými plochami. Platí tedy:

1. Obvod útvaru tvoří křivka, tvořená spojitými kruhovými oblouky a body nespojitosti, kde se tyto oblouky napojují.
2. Hrany útvarů se nekříží

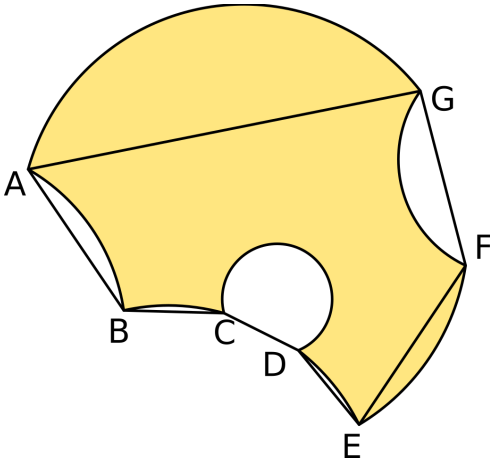
Příklad takové plochy je na obrázku 20.



Obrázek 20: Příklad možného útvaru

Obsah lze spočítat následujícím způsobem.

3. Z bodů, kde je hraniční křivka nespojitá (A,B,C,D,E,F,G na obrázcích 20 a 21) vytvoříme mnohoúhelník (Obrázek 21).



Obrázek 21: Útvar kombinovaný s polygonem

4. Nyní je na obrázku č.17 dobře vidět, že každé dva sousední body útvaru i mnohoúhelníku tvoří kruhovou úseč. Například hrana AG mnohoúhelníku a hrana AG původního obrazce tvoří dohromady úseč, která je v původním obrazci obsažena. Naopak hrany procházející body AB tvoří úseč, která není součástí původního obrazce.
5. Obsah celého obrazce spočítáme jako obsahu mnohoúhelníku, ke kterému budeme přičítat nebo od něho odečítat obsahy kruhových úsečí.

   Pro výpočet obsahu mnohoúhelníku lze použít následující vzorec [20].

$$S=\frac{1}{2}\left(\begin{pmatrix}x_1 & x_2\\ y_1 & y_2\end{pmatrix}+\begin{pmatrix}x_2 & x_3\\ y_2 & y_3\end{pmatrix}+\ldots+\begin{pmatrix}x_n & x_1\\ y_n & y_1\end{pmatrix}\right)$$

   kde x_i a y_i jsou souřadnice bodu v mnohoúhelníku.

   Vzorec pro výpočet obsahu kruhové úseče je například tento [21]:

$$S=\frac{1}{2}r^2(\alpha-sin\alpha)$$

kde *r* je poloměr kruhu a α je úhel který svírají krajní body úseče vzhledem ke středu kruhu.

Pokud chceme spočítat obsah útvaru s dírami, aplikujeme stejný postup i na díru a obsah poté odečteme od obsahu vnějšího útvaru.

Vzorec pro výpočet obsahu mnohoúhelníku vyžaduje, aby byly body seřazené. Proti směru hodinových ručiček dává kladné výsledky, po směru naopak záporné. Tohoto faktu můžeme využít při počítání obsahu obrazce s dírami. Pokud budeme vnější hranice seřazené proti směru hodinových ručiček a vnitřní naopak po směru, stačí nám pouze sečíst jejich obsahy, protože obsah vnitřních obrazců bude záporný.

Pro tento způsob výpočtu je výhodné reprezentovat teritorium jako kolekci cyklických grafů, kde uzly jsou průsečíky kružnic a hrany jsou kruhové oblouky mezi těmito body.

4.3.2 Analytický výpočet kolize

Tento algoritmus je navržen pro použití s výše uvedeným algoritmem pro výpočet obsahu a předpokládá stejnou grafovou reprezentaci teritorií.

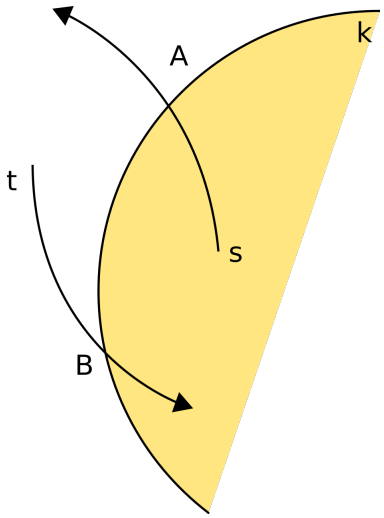
Pokud oblasti kolidují, podle návrhu hry potřebujeme spočítat jejich průnik a rozdíl. Protože to pro nás algoritmus bude výhodné, budeme předpokládat, že obrazec je v paměti uložen jako kolekce cyklických orientovaných grafů. Jeden graf představuje vždy jednu hraniční křivku. Vnější hranice jsou orientovány proti směru hodinových ručiček a vnitřní naopak po směru. V této reprezentaci na konci algoritmu opět vrátíme výsledek.

Protože testujeme novou oblast, která je z principu hry vždy pouze samotná kružnice, můžeme algoritmus zjednodušit. Algoritmus by však fungoval velmi podobně i pro obecnější útvary a lze ho modifikovat. Dále je nutné říci, že v algoritmu ignorujeme body doteku kružnic a za průsečíky považujeme pouze body na kružnicích, které se překrývají.

1. Nejprve najdeme pro každý segment každého grafu průsečíky s novou kružnicí.
2. Nalezené průsečíky seřadíme podle směru hodinových ručiček vzhledem k nové kružnici.

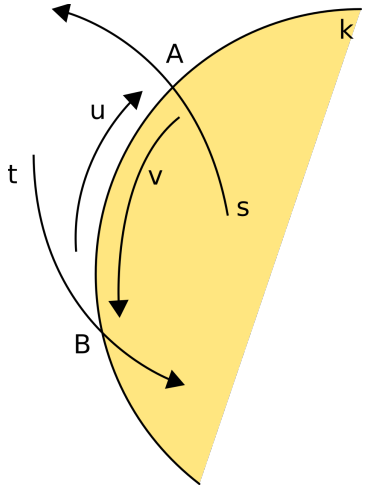
Následující postup ukážeme na obrázku pro jednodušší pochopení. Testujeme pro novou kružnici ‘k’ a našli jsme protínané hrany grafu ‘s’ a ‘t’ a průsečíky s hranami ‘A’ a ‘B’.

3. Pro každý průsečík otestujeme, zda je hrana, kterou průsečík protíná, orientovaná směrem dovnitř nové kružnice. Na obrázku 22 toto platí pro bod ‘B’, který leží na segmentu ‘t’.



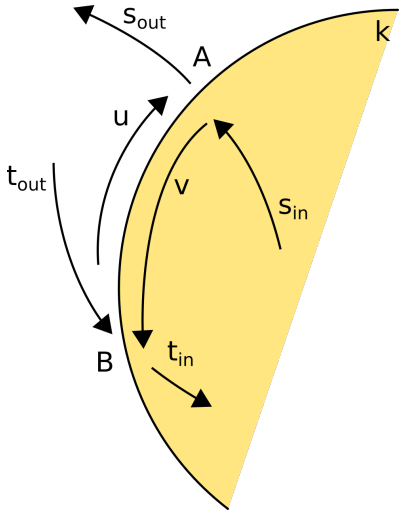
Obrázek 22: Základní kruh s protínajícími segmenty

4. Pokud takový bod máme, vytvoříme nové hrany grafu z předchozího bodu, tedy toho, který je na kružnici další v pořadí po směru hodinových ručiček. V našem příkladu na obrázku 23 je to bod ‘A’ vzhledem ke kružnici k. Vytvoříme jednu hranu orientovanou po směru a jednu proti směru hodinových ručiček. Na obrázku jsou to hrany ‘u’ (po směru hodinových ručiček) a ‘v’ (proti směru hodinových ručiček).



Obrázek 23: Vytvořené nové segmenty

5. Nyní rozdělíme původní segmenty protnuté kružnicí v nalezených bodech a propojíme je s nově vytvořenými segmenty podle jejich orientace (graf musí zůstat stejně orientovaný). Na obrázku 24 tedy rozdělíme segment ‘s’ na segmenty ‘s_{in}’ a ‘s_{out}’, a obdobně segment ‘t’ na segmenty ‘t_{in}’ a ‘t_{out}’. Poté vzniklé segmenty propojíme podle odpovídajících směrů. Vzniknou tedy dvě sekvence - (s_{in}; v; t_{in}) a (t_{out}; u; s_{out}).



Obrázek 24: Rozdělené původní segmenty a obnovení datové struktury

6. Tento postup opakujeme pro každý průsečík, jak bylo výše uvedeno.

V algoritmu využíváme invariantů datové struktury, které jsme si definovali na začátku. Hrany grafu se nemohou křížit, struktura je cyklická a směr šipky definuje hrany oblasti. Když tedy testujeme segmenty, nepotřebujeme ani vědět, jestli je tento segment součástí vnější nebo vnitřní hrany. Víme pouze, že další segment v řadě musí jít nutně opačným směrem.

4.3.3 Řádkový algoritmus kolize

Druhou možností pro počítání kolize a obsahu je způsob založený na tzv. Pixel-based kolizi. Vymyslel jsem rychlejší alternativu tohoto řešení.

Algoritmus funguje zkráceně takto. Převedeme analytickou reprezentaci obsazených teritorií na řádkovou reprezentaci. Každý řádek reprezentuje vybarvené pixely na určité souřadnici na ose y. Řádek je pole rostoucích celočíselných X souřadnic, které reprezentují vždy začátek a konec vyplněné série pixelů, tedy například [1,5,8,9] reprezentuje řádek o dvou linkách. Teritorium je poté reprezentováno jako pole takových řádků. Pro řádky samotné jsem vymyslel rychlý kolizní algoritmus, který spočítá jejich průnik a zbytkové části pouze jedním průchodem obou řádků za použití základních operací porovnání, sčítání a odčítání.

Objekty složené z řádků lze pak kolidovat použitím zmíněného algoritmu na odpovídající dvojice řádků na stejné y souřadnici. V našem problému nezáleží na tom, zda zaměníme y za x, protože předpokládáme podobné rozložení ve všech směrech. Pro některé geometrické tvary by mohlo být výhodnější využít jen jednu z reprezentací, například jeden vysoký sloupec by v námi zvolené reprezentaci vypadal jako dlouhé pole s velkým množstvím stejných řádků, kdežto ve sloupcové reprezentaci by byl pouze pole o jednom sloupci, jehož výšku lze vyjádřit dvěma čísly. V našem případě ale tvary jsou vždy kruhy a různé útvary složené z kruhových segmentů bez jakékoli preference v určitém směru, ve výsledku tedy na zvolené reprezentaci nezáleží.

4.4 Zhodnocení algoritmů

Původně jsem aplikaci navrhl s použitím analytických algoritmů a pokusil se implementovat hlavní algoritmus pro řešení kolizí, protože byl jedním z nejpodstatnějších prvků celé hry. Během práce se ale ukázal jako velmi implementačně náročný z více důvodů. I když to na první pohled není zřejmé, existuje velké množství speciálních případů, kvůli kterým jsem musel algoritmus víckrát přepisovat, upravovat a znovu debugovat. Největším problémem se pak ukázalo být počítání s desetinnými čísly, které je na počítači z povahy nepřesné. V určitém kroku algoritmu je potřeba spočítané body seřadit a na výsledku seřazení závisí výsledek algoritmu. Kvůli nepřesnostem ve výpočtu a špatným souřadnicím mohly být ale body seřazeny nesprávně a výsledná datová struktura tak nebyla konzistentní. Mimo to program někdy našel průsečíky kružnic i tam kde být neměly. Abych se vypořádal s tímto problémem, musel bych úplně předělat celý postup s nejistým výsledkem. Proto jsem se rozhodl použít druhý zmíněný postup, který má nakonec více výhod.

Hlavní výhody řádkového přístupu:

- Algoritmus používá pouze celočíselnou aritmetiku a je tedy deterministický. Díky tomu můžeme po síti přenášet menší redundantních dat, která si klient může dopočítat. Analytický algoritmus oproti tomu používá aritmetiku s plovoucí desetinnou čárkou, která nezaručuje stejné výsledky výpočtů v různých případech.
- Algoritmus může být velmi rychlý oproti počítání kolize analyticky, která vyžaduje hodně náročných operací, jako je počítání odmocnin či goniometrických funkcí.
- Spolehlivost a testovatelnost. Algoritmus je jednodušší napsat a otestovat, protože speciálních případů je málo.

- Databáze Mongo není dobře uzpůsobená pro grafové a cyklické struktury. Řádková reprezentace je mnohem jednodušší způsob reprezentace.
- Obsah lze spočítat jednoduše součtem délek všech linek, tento obsah je ale závislý na hustotě mřížky souřadnic, kterou je nutné stanovit.

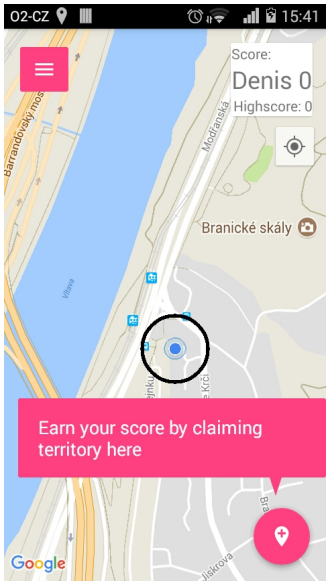
Nevýhody:

- Rychlost algoritmu je v ose y závislá na velikosti geometrických útvarů. Jinými slovy dvakrát vyšší útvar trvá dvakrát déle vytvořit a stejně tak bude déle trvat kolize s podobným objektem. Oproti tomu analytický algoritmus se zpomaluje s množstvím objektů. Při více objektech na menší ploše je řádkový algoritmus rychlejší. Jinými slovy řádkový algoritmus je rychlejší na menších plochách, analytický zase při menším počtu logických objektů.
- Řádková reprezentace zabírá více místa v paměti než analytická reprezentace, pokud jí nebudeme chtít počítat víckrát ale ukládat jí v paměti.

I tento druhý algoritmus jsem implementoval dvakrát, nejdříve v jednodušší formě a poté jsem přišel s mnohem rychlejší, optimalizovanou verzí. Kolizní algoritmus byl tak nejkomplikovanějším problémem celé práce a jeho řešení zabralo nejvíce času.

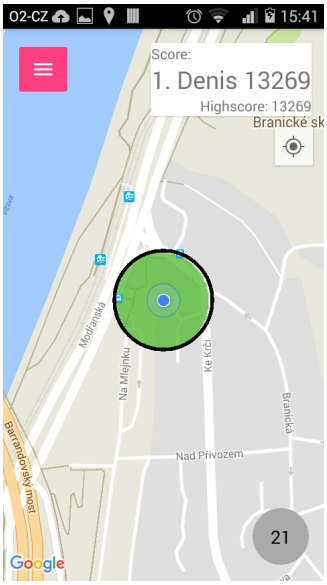
4.5 Výsledek

Výsledná aplikace se od návrhu liší v mnoha detailech. Změny vychází ze snahy přiblížit se více předepsanému Material stylu. Rozhodl jsem se zrušit zvláštní obrazovku pro menu a ponechat menu jen přístupné z hlavní herní obrazovky (25). Namísto popsaného tlačítka jsem použil klasickou ikonku „Hamburger menu“, která se v prostředí Androidu běžně používá. V pravém horním rohu se kromě skóre ukazuje i nejvyšší dosažené skóre.



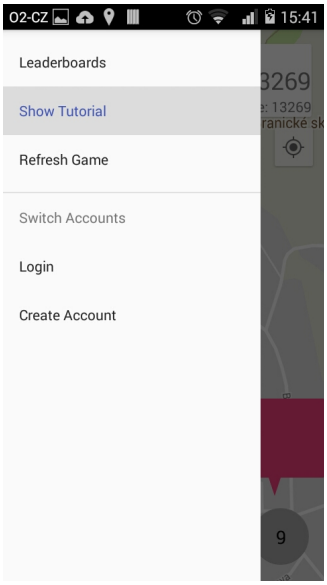
Obrázek 25: Obrazovka hry

Na obrázku 26 vidíme obsazené území a hráčovo skóre je rázem vyšší, stejně jako hodnota nejvyššího skóre. Vlevo dole vidíme implementované odpočítávání do dalšího možného obsazení.



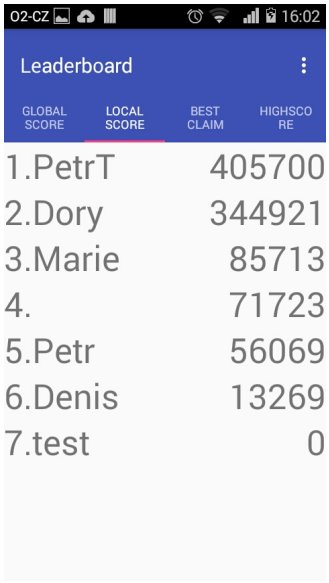
Obrázek 26: Obsazené území

Na dalším obrázku (27) vidíme otevřené menu. Menu neobsahuje některé neimplementované funkce, které byly uvedeny v návrhu. Nahradil jsem je tedy funkcemi, které jsou podstatné v této minimální verzi hry.



Obrázek 27: Otevřené menu

Na obrázku 28 je zachycen leaderboard. Jednotlivé leaderboardy se mírně liší od návrhu. V aplikaci jsou nyní přítomné ty, které byly nejpodstatnější pro testování, ale na platformě Gamesparks lze poměrně jednoduše vytvořit množství dalších. Zobrazen je leaderboard z hráčova okolí.



Obrázek 28: Leaderboard

5 Testování

Hra byla testována více způsoby. Aplikaci samotnou jsem testoval především jednotkovými testy, a to hlavně u součástí, které byly kritické pro logiku hry. Serverovou část hry jsem testoval hlavně pomocí nástrojů na platformě Gamesparks. Celou hru jsem pak testoval se skupinou uživatelů.

5.1 Testy aplikace

V následující kapitole stručně představím testy vnitřní struktury aplikace. Nejdůležitějšími testy byly testy kolizního algoritmu, který je kritický pro funkčnost hry. Protože algoritmus je potřeba počítat na serveru i v klientské aplikace, zvolil jsem pro vývoj následující postup. Algoritmus jsem implementoval nejprve v Javě v klientské aplikaci, současně s množstvím jednotkových testů, kterými jsem se snažil pokrýt co nejvíce hraničních případů a když jsem si byl dostatečně jist spolehlivostí algoritmu, přepsal jsem jej do Javascriptu, se používá ve skriptech na platformě Gamesparks. Tím jsem se vyhnul testování a řešení chyb na straně Gamesparks platformy, která zatím není pro jednotkové testování dobře uzpůsobena, a musel jsem tak vyřešit pouze problémy při přechodu mezi jazyky.

Testy analytického algoritmu kolize jsem psal pro velké množství případů, protože při psaní se postupně ukazovalo více a více hraničních případů, kde algoritmus selhával. Nejdříve jsem psal testy pouze pro různé případy dvou kruhů, ale poté co jsem začal testovat tři kruhy jsem brzy algoritmus opustil, protože množství speciálních případů a chyb bylo pořád velké a nemohl jsem si být jistý, že i když všechny mé testy projdou, neexistuje nějaký další speciální případ, který by testem neprošel. Během implementace jsem vytvořil také jednoduchý náhodný test, který testoval náhodné kolize různého počtu kruhů a počítal pouze za jakých podmínek algoritmus spadne, nebo zanechá datovou strukturu v nekonzistentním stavu. Než jsem algoritmus opustil, podařilo se mi snížit takovou úspěšnost algoritmu pouze na hodnotu kolem 0,5 % chybových kolizí.

Testy řádkového algoritmu kolize jsem psal nejdříve pro řádky samotné a poté pro skupiny řádků. Testovací třída je vždy napsána podobně. Základem je jedna metoda, která je šablonou pro test. Provede test a porovná očekávaný výsledek se skutečným. Ostatní metody pak pouze volají pouze tuto šablonu s různými vstupy a očekávanými výsledky. I přes toto zjednodušení byl v mnohých případech testovací kód rozsáhlejší než kód funkční. Velkou výhodou řádkového algoritmu oproti analytickému je právě jednodušší testovatelnost. Množství jednotlivých typů případů je mnohem menší, a tak jsem poměrně rychle dospěl ke stoprocentní spolehlivosti algoritmu.

5.2 Testy serverové části aplikace

Pro testování serverové části aplikace jsem využil především testovacího prostředí platformy Gamesparks. Platforma umožňuje posílat testovací požadavky, sledovat odpovědi a debugovat skripty, které jsou k jednotlivým požadavkům přidělené. Tímto se dá ale testovat pouze ručně případ od případu, tedy pouze základní funkcionality, ale platforma žádným způsobem neposkytuje jednotkové testování přímo.

U důležitých funkcionalit, které bylo potřeba implementovat jak na serverové, tak na klientské straně, jsem prvně implementoval klientskou část v Javě, důkladně ji otestoval pomocí jednotkových testů a poté ji přepsal do Javascriptu. Tím jsem se vyhnul nutnosti testování principu algoritmu v Javascriptu na straně serveru a pouze otestoval chyby vzniklé rozdíly mezi jazyky nebo chybami v přepisu.

5.3 Prvotní test hry

Než jsem hru testoval s uživateli, testoval jsem ji během vývoje víckrát sám, či pouze s jedním nebo dvěma dalšími uživateli. Díky těmto testům jsem objevil několik problémů v principu hry, které jsem poté musel vyřešit v implementaci.

1. Nutnost časového limitu.

   Hráč, který se pohybuje rychleji v dopravním prostředku má oproti ostatním výhodu a může území obsazovat rychleji. Z toho důvodu jsem do hry zavedl časový limit na obsazení dalšího území.

2. Možnost podvodu

   Pokud se dva nebo více hráčů setká na jednom místě a toto místo střídavě obsazují, přičítáním průniků tak vznikne situace, kdy se navzájem střídají ve vedení a jejich skóre pořád roste, aniž by ve hře vyvíjeli nějakou aktivitu. Proto jsem do hry přestal přičítat sekundární průniky teritorií. Pokud tedy nastane kolize, hráč dostane bonus za ukrojení cizího teritoria, toto bonusové teritorium ale na mapě nezůstane a mapa tak zůstává jednovrstvá. Tím zachováme bonus za obsazení cizího území, ale limitujeme ho tak, že při výše zmíněné situaci hráčům skóre neroste ale zastaví se na určité hranici.

5.4 Test hry s uživateli

Pro test hry jsem zorganizoval test s uživateli. Uživatelé byli vybrání na základě jednoduchého dotazníku (screener), který je obsažen v přílohách. Cílem testu bylo především ověření funkčnosti herních mechanik.

Participanty testu jsem vybíral pomocí screeneru, ve kterém jsem se zaměřil především na mladší uživatele, kteří hrají hry na telefonu a mají zkušenosti i s hraním location-based her. Dále jsem se snažil vybrat do skupiny i uživatele, kteří mají zkušenosti s webovými hrami jako je Agar.io, nebo podobné, kterými je tato hra inspirovaná.

5.4.1 Průběh testu

Test proběhl jako hodinová hra, kdy byli uživatelé rozmístěni po krajině zhruba v okruhu jednoho kilometru a poté zkusili hru hrát. Po hře proběhla diskuze s uživateli, kde jsme diskutovali pozitiva a negativa hry a možné změny. Testování bylo zaměřeno hlavně na test hry samotné.

Průběh hry jsem pozoroval jako jeden z hráčů na dalším zařízení. Osobně jsem taky s hráči chodil a pozoroval herní chování.

5.4.2 Seznam nálezů

Nálezy v oblasti hry samotné:

1. Uživatelé se navzájem sledují. Nejvíce bodů za jedno obsazení může hráč získat, pokud přebere oblast jinému hráči. Vyplatí se tedy sledovat jiného člověka a pouze mu přebírat oblasti které cestou obsadil.
2. Časový limit ve hře všichni hráči doporučili zkrátit z třiceti sekund na menší hodnotu.
3. Více hráčů také navrhlo zrušit mazání teritorií a tím odstranit časovou lokalitu hry ve prospěch dlouhodobého postupu ve hře.

Testování odhalilo mimo jiné několik technických chyb:

1. Za určitých nejasných okolností se po obsazení na mapě neobjeví poslední obsazená oblast
2. Hra všem uživatelům několikrát spadla, některým víckrát než ostatním. U některých stačilo aplikaci zapnout znovu, u jiných padala poté při každém startu. Řešením pak bylo zapnout hru s vypnutým internetem a poté internet znovu zapnout.
3. Chyba v počítání skóre. Skóre se hráčům nesprávně odečítalo při mazání oblastí, výsledkem pak byla nesprávně vyšší hodnota. Hráči, kteří už ztratili všechny obsazené oblasti pak měli pozitivní skóre, i když by správně měli mít nulové.
4. Překreslování teritorií je při aktivním hraní časté a vykreslování na mapu vždy chvíli trvá. Hra tak někdy nepříjemně bliká.

Hra všem uživatelům spotřebovala kolem 5 MB dat. Jedná se o odhad operačního systému a číslo tedy není přesné. Vzhledem k hodinovému hraní to není velké číslo.

5.4.3 Shrnutí testu s uživateli

Test s uživateli proběhl bez větších komplikací. Menším problémem byly instalace na různá zařízení. Především nová zařízení Android jsou důsledně chráněná proti instalacím z neznámých zdrojů a každá značka realizuje ochranu trochu jiným způsobem.

Uživatelé, a to především mladší, na které byla hra zaměřena, hodnotili hru pozitivně a měli přínosné připomínky a návrhy na změny. Testování považuji za úspěšné.

6 Závěr

Cílem práce bylo navrhnout a implementovat location-based hru na základě analýzy problematiky a existujících titulů. V analytické části jsem popsal některé charakteristiky her na přenosných zařízeních a poté přímo location-based her. V další části jsem analyzoval potřebné technické nástroje a zformuloval požadavky na hru. Poté jsem popsal návrh hry samotné, který byl ovlivněn poznatky z analýzy. Návrh obsahuje popis herních mechanik a funkcionalit aplikace, návrh uživatelského rozhraní a návrh architektury systému.

Implementoval jsem hlavně jádro hry s herními mechanikami. Soustředil jsem se především na technické aspekty hry, a ne tolik na aplikaci samotnou a uživatelské rozhraní. Implementace hry a komunikace byla velmi náročná. Nejnáročnější částí práce bylo napsání kolizního algoritmu.

Velmi náročným problémem byla taky synchronizace herních dat mezi serverem a hráči. I když byl systém naimplementován poměrně jednoduše, vyskytlo při hraní hry hodně problémů s konzistencí herních dat na serveru. Pravděpodobně se jednalo o chyby vzniklé vícevláknovým zpracováním požadavků a zápisů do leaderboardu. Na platformě lze ale takové těžko chyby dohledat. Velkým problémem byla obtížná testovatelnost situací s více hráči a reprodukovatelnost některých chybových situací.

Problémem byla taky neúplná dokumentace a některé chybějící informace na platformě Gamesparks. Často jsem musel hledat některé funkcionality přímo ve zdrojovém kódu SDK. Během vývoje jsem také našel v SDK chybu, kvůli které aplikace přenášela po sítí tisíckrát větší množství dat. Naštěstí ji ale na mé upozornění tým Gamesparks rychle opravil.

Hru jsem otestoval se skutečnými uživateli. Test proběhl úspěšně a odhalil různé technické chyby ve hře. Díky testování jsem také objevil různé mezery v principu hry samotné a během diskuze jsme přišli s dalšími nápady na vyladění herních mechanik. Uživatelé měli přínosná doporučení pro změny ve hře, ale s hrou byli spokojeni.

Hru nelze považovat za hotovou a pro úplné zprovoznění je potřeba ještě velké množství práce. To zahrnuje především implementaci a testování různých verzí herních mechanik s uživateli, optimalizace pro velké množství hráčů a implementace sociálních funkcí uvedených v návrhu. I když jsem původně plánoval udělat větší část výsledné hry, komplikace při implementaci mě donutily svůj plán přehodnotit a zjednodušit. I přesto jsem s výsledkem práce spokojen a považuji cíl za splněný.

7 Použité zdroje

[1] SPEVÁK, Vladislav. Mobile games. In: *A7B39PHA - Počítačové hry* [online]. Přednáška. ČVUT FEL. Dostupné z: https://cent.felk.cvut.cz/predmety/39PHA/data/prednasky/12-mobile-games.pdf

[2] JACOB, João Tiago Pinheiro Neto a António Fernando COELHO. Issues in the Development of Location-Based Games. *International Journal of Computer Games Technology* [online]. 2011, **2011**. Dostupné z: http://dx.doi.org/10.1155/2011/495437

[3] LOVE, Robert. Why does GPS use so much more battery than any other antenna or sensor in a smartphone? - Quora, odpověď. In: *Quora* [online]. [vid. 2018-01-07]. Dostupné z: https://www.quora.com/Battery-Life/Why-does-GPS-use-so-much-more-battery-than-any-other-antenna-or-sensor-in-a-smartphone

[4] INC, Niantic. *Ingress* [online]. B.m.: Niantic, Inc., 2017 [vid. 2018-01-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.nianticproject.ingress

[5] SUELLENTROP, Chris. Ingress, a Mobile Game From Google. *The New York Times* [online]. 2014 [vid. 2018-01-07]. ISSN 0362-4331. Dostupné z: https://www.nytimes.-com/2014/07/15/arts/video-games/ingress-a-mobile-game-from-google.html

[6] HERNANDEZ, Alex. How Safe Are You Playing Ingress? *Techaeris* [online]. 24. červen 2015 [vid. 2018-01-07]. Dostupné z: https://techaeris.com/2015/06/24/editorial-how-safe-are-you-playing-ingress/

[7] INC, Niantic. *Pokémon GO* [online]. B.m.: Niantic, Inc., 2017 [vid. 2018-01-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.niantic-labs.pokemongo

[8] SUAREZ, Rasiel. *GPS Tycoon Game* [online]. B.m.: Rasiel Suarez, 2015 [vid. 2018-01-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.gpstycoon

[9] *libgdx* [online]. [vid. 2018-01-08]. Dostupné z: https://libgdx.badlogicgames.com/

[10] *Cocos2d-x - World's #1 Open-Source Game Development Platform* [online]. [vid. 2018-01-08]. Dostupné z: http://www.cocos2d-x.org/

[11] Google Maps APIs. *Google Developers* [online]. [vid. 2018-01-08]. Dostupné z: https://developers.google.com/maps/

[12] OpenStreetMap. *OpenStreetMap* [online]. [vid. 2018-01-08]. Dostupné z: https://www.openstreetmap.org/

[13] *Tile Usage Policy* [online]. [vid. 2018-01-08]. Dostupné z: https://operations.osmfoundation.org/policies/tiles/

[14] MAPSFORGE. *mapsforge: Vector map library and writer - running on Android and Desktop* [online]. Java. 2018 [vid. 2018-01-08]. Dostupné z: https://github.com/mapsforge/mapsforge

[15] *osmdroid: OpenStreetMap-Tools for Android* [online]. Java. B.m.: osmdroid, 2018 [vid. 2018-01-08]. Dostupné z: https://github.com/osmdroid/osmdroid

[16] *GameSparks - The Ultimate Game Backend Development Platform* [online]. [vid. 2018-01-08]. Dostupné z: https://www.gamesparks.com/

[17] The Game Industry's Most Powerful Backend Platform. *PlayFab* [online]. [vid. 2018-01-08]. Dostupné z: https://www.playfab.com/

[18] Material Design. *Material Design* [online]. [vid. 2018-01-08]. Dostupné z: https://material.io/

[19] FREDERICKSON, Ben. *Calculating the intersection area of 3+ circles* [online]. [vid. 2018-01-07]. Dostupné z: http://www.benfrederickson.com/calculating-the-intersection-of-3-or-more-circles/

[20] WEISSTEIN, Eric W. Polygon Area. *Wolfram MathWorld* [online]. [vid. 2018-01-07]. Dostupné z: http://mathworld.wolfram.com/PolygonArea.html

[21] WEISSTEIN, Eric W. Circular Segment. *Wolfram MathWorld* [online]. [vid. 2018-01-07]. Dostupné z: http://mathworld.wolfram.com/CircularSegment.html

Příloha A

Zip Archiv - Zdrojové kódy, aplikace a materiály

Obsah zip archivu:

- /Screenshots – screenshoty z aplikace, obsažené i v této práci
- /Source – zdrojové kódy aplikace a skripty z Gamesparks platformy
- /app.apk – instalovatelný aplikace

Příloha B

Instalační příručka

K instalaci je potřeba Android zařízení s Android 4.0.3 a vyšší. Přiložený *.apk soubor je možné po připojení zařízení k počítači uložit do paměti telefonu. Potom stačí soubor spustit přímo v telefonu a aplikace se nainstaluje.

Různé verze Androidu a výrobci uplatňují různé ochranné prostředky proti škodlivým aplikacím. Z toho důvodu může být instalace zakázána a v nastavení aplikací pak bude potřeba zaškrtnout volbu „povolit instalaci aplikací z neznámých zdrojů“ (nebo podobně znějící nastavení).

Aplikace vyžaduje oprávnění k přístupu k internetu a informacích o poloze, která je na novějších zařízeních Android potřeba explicitně povolit. Aplikace si o přidělení oprávnění sama zažádá, ale v případě problémů můžete oprávnění přidělit ručně v nastavení aplikací.

Aplikace funguje správně pouze s internetovým připojením a zapnutým přístupem k poloze zařízení.

Příloha C

Screener

Otázky použité při výběru participantů testu.

Hrajete hry na mobilním telefonu?

Hrajete jednoduché hry v prohlížeči, jako je například Agar.io, Slither.io, Diep.io, Limax.io nebo podobné?

Hrajete nějaké hry, které používají Vaší lokaci, jako jsou například Pokémon GO nebo Ingress?

Kontakt:

Pokud nám chcete ještě něco sdělit, napište to prosím sem: