Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Herní frameworky na platformě Android

Plzeň 2021 Martin Lácha

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin LÁCHA**
Osobní číslo:	**A17B0418P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Herní frameworky na platformě Android**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte vybrané herní frameworky na platformě Android.
2. Proveďte srovnání těchto frameworků z různých hledisek použití (na jaké hry je lze použít, komplexnost a snadnost použití aj.).
3. Na jednom z vybraných frameworků implementujte hru dle vlastního výběru pro platformu Android, která využije výhod daného frameworku.
4. Otestujte použitelnost realizované hry, navrhněte její možná rozšíření a zhodnoťte přínosy použití frameworku.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Ladislav Pešička**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Poděkování

Tímto bych rád poděkoval Ing. Ladislavu Pešičkovi, za odborné vedení práce, cenné rady a čas, který mi poskytl při zpracování této bakalářské práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2021

Martin Lácha

Abstract

This work aims to compare the properties of selected game frameworks and use one of them to implement a game for mobile phones with the Android operating system. The resulting application will take advantage of the selected framework. The theoretical part will focus on the analysis of selected frameworks and comparison of their properties from different perspectives (e.g. types of games, complexity, etc.). In the practical part, a game application will be designed (game description, game loop, scenes, controls). The programming documentation will focus on the description of implemented classes, the most important methods, user rights, project structure, etc. At the end, the application will be tested according to the described scenarios and other possible extensions will be proposed.

Abstrakt

Cílem této práce je porovnat vlastnosti vybraných herních frameworků a pomocí jednoho z nich implementovat hru pro mobilní telefony s operačním systémem Android. Výsledná aplikace bude využívat výhod vybraného frameworku. Teoretická část se bude zaměřovat na analýzu vybraných frameworků a porovnání jejich vlastností z různých hledisek (např. typy her, komplexnost atd.). V praktické části bude navrhnuta herní aplikace (popis hry, herní smyčka, scény, ovládání). Programátorská dokumentace bude zaměřena na popis implementovaných tříd, nejdůležitějších metod, uživatelských oprávnění, struktura projektu aj. Na závěr bude aplikace otestována podle popsaných scénářů a budou navržena další možná rozšíření.

Obsah

1 Úvod

V dnešní době využívá mobilní telefon denně téměř každý, ať již za účelem sdílení fotek s přáteli, vyhledávání informací na internetu nebo hraní her. Není zapotřebí velkého herního studia a velmi zkušených programátorů, ale i běžný programátor si dokáže s podporou dostupného softwaru vytvořit kvalitní herní aplikaci.

Cílem této bakalářské práce je porovnat vybrané herní frameworky z různých hledisek použití (např. typ her, nástroje, užitečné funkce atd.) a vytvoření aplikace pro mobilní zařízení pomocí jednoho z nich. V práci bude proveda analýza jednotlivých frameworků a porovnání jejich vlastností. Po analýze budou shrnuty výhody a přednosti jednotlivých herních frameworků. Na jednom z nich bude implementována aplikace na mobilní zařízení s operačním systémem Android, která bude využívat výhod daného frameworku.

V další části bude poté uveden návrh hry a popis jednotlivých částí implementace - např. popis hry, scény, ovládání, herní smyčka apod. Nebude chybět ani otestování hry podle připravených scénářů na mobilních zařízeních s odlišnými parametry.

Poslední část práce bude popisovat a hodnotit výsledky testování a dále bude obsahovat další možná rozšíření pro implementovanou mobilní hru.

2 Herní Frameworky/Enginy

Framework neboli aplikační rámec je skupina knihoven, které dávají ucelený přístup k řešení nějaké problematiky. Je to základ, na kterém lze vyvíjet aplikaci pro specifickou platformu. Obsahuje například předdefinované třídy a funkce, jež může vývojář použít při vývoji daného softwaru. Tyto třídy a funkce slouží jako podpora při vývoji aplikací. Framework také může obsahovat i podpůrné programy a knihovny API.

Framework se využívá např. pro zpracování vstupu, správu hardwarových zařízení a interakci se systémovým hardwarem. Zjednodušuje vývojový proces, takže vývojáři nemusí programovat vše pokaždé, když začnou vytvářet novou aplikaci [12].

Obecně se hry dělí na dvě skupiny a to 2D a 3D, viz obr. 2.1. Dvoudimenzionální (zkráceně 2D hry), používají plochou grafiku pro zobrazení a nemají trojrozměrnou geometrii. Na obrazovce jsou vykreslovány ploché obrázky a kamera nemá perspektivu. Příkladem takové hry je **Hollow Knight**, viz obr. 2.1(a). Trojrozměrné (zkráceně 3D hry), využívají trojrozměrnou geometrii, tedy textury jsou vykresleny jako okolní prostředí, postavy nebo jiné objekty, které poté tvoří svět hry. Tyto 3D hry většinou využívají perspektivy, tedy objekty se zvětšují při přiblížení kamery. Příkladem pro 3D hru je *Osiris: New Dawn*, viz obr. 2.1(b) [13].

(a) Hollow Knight

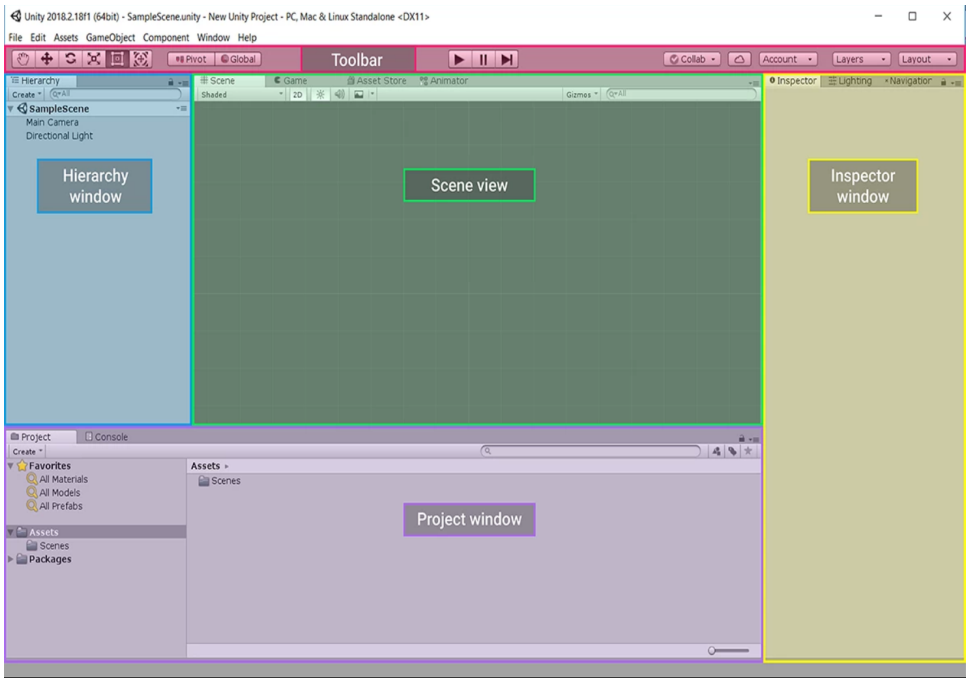
(b) Osiris: New Dawn

Obrázek 2.1: Ukázka 2D a 3D hry (zdroj: [13])

2.1 Unity

Unity [4] je multiplatformní herní engine určený pro vývoj a podporu při tvorbě 2D a 3D videoher, simulací, virtuální reality (Oculus VR, PlayStation VR, HTC vive), konzole (Xbox, PlayStation, Nintendo Switch) mobilních zařízení (Android, iOS). Unity podporuje celkem přes 25 různých platforem a technologií. Jedná se o komerční systém vyvinutý společností Unity Technologies. Poskytuje trojrozměrnou manipulaci a také simulaci prostřednictvím funkcí definovaných pomocí programovacích jazyků. Jedná se o nástroj, který je schopný vytvářet velmi kvalitní hry a design. Tento engine není určený jen pro vývoj her, ale může se použít pro tvorbu interaktivních 3D modelů, vytváření animací nebo jako nástroj pro vytváření návrhů pro architekty a strojní inženýry. Unity obsahuje tři důležité komponenty.

- Herní engine - umožňuje vytvářet a testovat hry na různých platformách.
- Aplikace - zde se design nebo uživatelské rozhraní spojuje s grafickým náhledem a funkcí ovládání.
- Editor kódu - IDE poskytuje textový editor pro psaní kódu, viz obr. 2.2.



Obrázek 2.2: Ukázka Unity editoru (zdroj: [6])

Unity je vhodná platforma pro vývoj herních aplikací a především pro začínající vývojáře, kteří chtějí začít s vývojem herních aplikací. Oficiální stránky poskytují mnoho videí, kde vysvětlují různé principy, které vývojáře seznámí s vývojovým procesem. Vybrané známé hry na mobilní zařízení vytvořené v Unity:

1. **Angry Birds 2**

   Angry Birds 2 je logická hra. Pomocí praku střílíte ptáky na prasata, s cílem je zničit, viz obr. 2.3.

Obrázek 2.3: Ukázka ze hry Angry birds (zdroj: [5])

2. **Heartstone: Heroes of Warcraft**

   Karetní hra, jejíž cílem je porazit soupeře pomocí kouzel a bytostí z Vámi vytvořeného balíčku karet, viz obr. 2.4.

Obrázek 2.4: Ukázka ze hry Heartstone: Heroes of Warcraft (zdroj: [17])

3. **Call of Duty: Mobile**

   Mobilní akční střílečka pro více hráčů, viz obr. 2.5. Nabízí více herních módů (všichni proti všem, dva týmy bojující proti sobě a další).

Obrázek 2.5: Ukázka ze hry Call of Duty: Mobile (zdroj: [22])

Unity nabízí více různých verzí (plánů) z pohledu zákazníka nebo firmy, která bude Unity využívat.

- **Individuální plán**
  - **Student** verze je určena pro studenty. Pro získání této verze je potřeba splňovat několik podmínek. Student musí být zapsaný v akreditové vzdělávací instituci v zákonném věku (USA - 13 let, EU - 16 let), aby mohl souhlasit se shromažďováním a zpracováním svých osobních údajů. Pro ověření se musí připojit ke *Github Student Developer Pack*. Obsahuje poslední verzi Unity, real−time cloud diagnostiku (nástroje pro identifikaci a shromažďování problémů, se kterými se uživatel setká v aplikaci).
  - **Personal** verze je také zdarma. Obsahuje poslední verzi Unity a zdroje pro začátek vývoje a učení Unity. Příjmy z vytvořených her musí být menší než $100 000 (přibližně 2,15 mil. Kč) za posledních 12 měsíců.

- **Týmové plány**
  - **Plus** obsahuje poslední verzi Unity, přizpůsobení úvodní obrazovky (např. deaktivovat logo nebo přidat další loga apod.). LiveOps analýzu pro úpravu, optimalizaci a aktualizaci aplikace. Realtime cloud diagnostiku pro identifikaci a shromažďování problémů, se kterými se uživatelé setkají v aplikaci. Jedná se o placenou verzi. Příjmy z vytvořených her musí být menší než $200 000 (přibližně 4,3 mil. Kč) za posledních 12 měsíců.
  - **Pro** obsahuje vše, co obsahoval plán **Plus**. Dále obsahuje *Unity Teams Advanced* (Cloud úložiště, Build sharing, historie verzování a další), prioritní přístup k zákaznickému servisu a Success Advisors (specialisté, kteří pomohou s výběrem např. správného nástroje). Dále si můžete doplatit technickou podporu a jiné. Jedná se o placenou verzi. Příjmy z vytvořených her musí být větší než $200 000 (přibližně 4,3 mil. Kč) za posleních 12 měsíců.
  - **Enterprise** obsahuje vše, co se nachází v plánu **Plus**, a navíc například technickou podporu, učební plán na míru, živé relace *Enterprise Learn*. Jedná se o placenou verzi. Příjmy z vytvořených her musí být větší než $200 000 (přibližně 4,3 mil. Kč) za posledních 12 měsíců a minimálně 10 *seats* (osob využívající software Unity).

Přehled cen jednotlivých plánů [21] jsou k nalezení viz tab. 4.3. Detailní srovnání jednotlivých plánů jsou popsána na stránce:
`https://store.unity.com/compare-plans`

2.2 LibGDX

LibGDX [15] je relativně nízkoúrovňový, open source, multiplatformní herní framework. Cílem tohoto projektu je podpořit vývojáře ve vytváření her nebo aplikací, a nasazení na jednu nebo více podporovaných platforem. Podporuje jak vývoj 2D, tak 3D her.

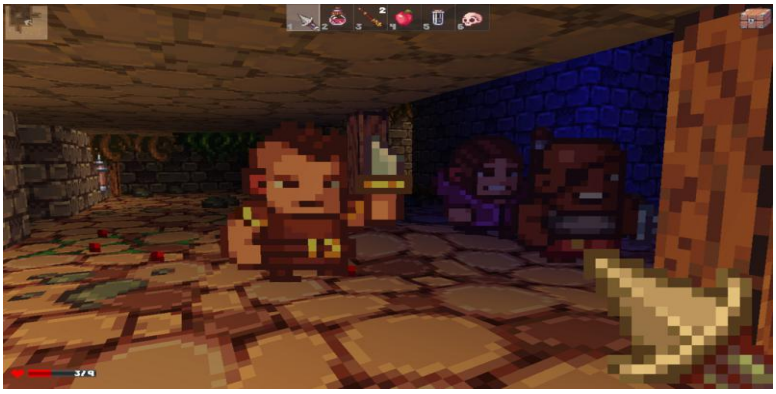
LibGDX je napsán v jazyce Java, C a C++. Aplikace a hry lze psát v jazycích: Java, Scala, Kotlin, Clojure, Groovy. Je dostupný na oficiálních stránkách zdarma a je licencován open source softwarovou licencí Apache 2, která umožňuje každému tento projekt používat [19].

Framework využívá intezivně programovací jazyk C pro kritické úlohy z hlediska výkonu a umožnil použít funkce napříč jednotlivými platformami. Obsahuje funkční API, které je kombinací všech podporovaných platforem.

Jedním z důležitých prvků je schopnost spouštět a ladit zdrojový kód na ploše jako nativní aplikaci. To umožní vývojářům používat funkce JVM, jako jsou Code Hot Swapping, který umožnuje změnit kód spuštěného programu, aniž byste museli přerušit běh tohoto programu. To výrazně zkrátí čas na hledání a opravu nepříjemných chyb. Vybrané známé hry na mobilní zařízení vytvořené v LibGDX:

1. **Delver**

   Akční roguelike hra viz obr. 2.6. Ocitnete se v náhodně generovaném dungeonu, cílem je získat *Yithidian Orb* a vrátit se na místo, kde jste začali.



Obrázek 2.6: Ukázka ze hry Delver(zdroj: [11])

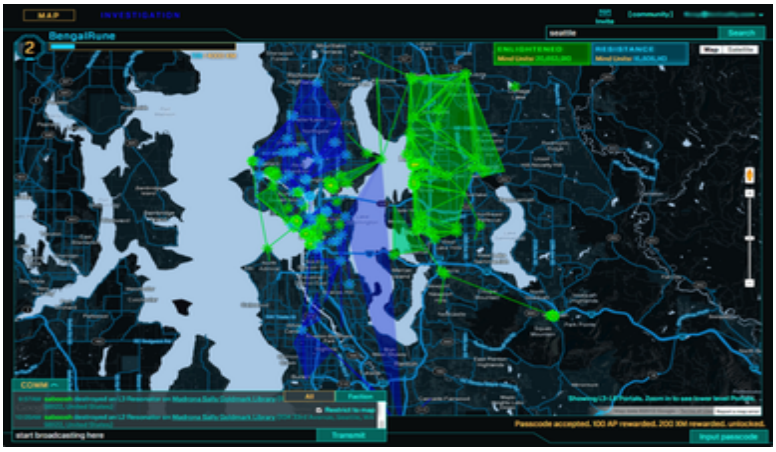
2. **Nubs' Adventure**

   Skákací akční hra, ve které prozkoumáváte svět, viz obr. 2.7.

Obrázek 2.7: Ukázka ze hry Nubs' Adventure (zdroj: [20])

3. **Ingress**

   Jedná se o online hru pro více hráčů, využívající rozšířenou realitu, viz obr. 2.8. Ta umožňuje pohyb v rámci reálného světa, kde nalézáte virtuální předměty. Cílem hry je obsadit portály dříve než nepřátelský tým.



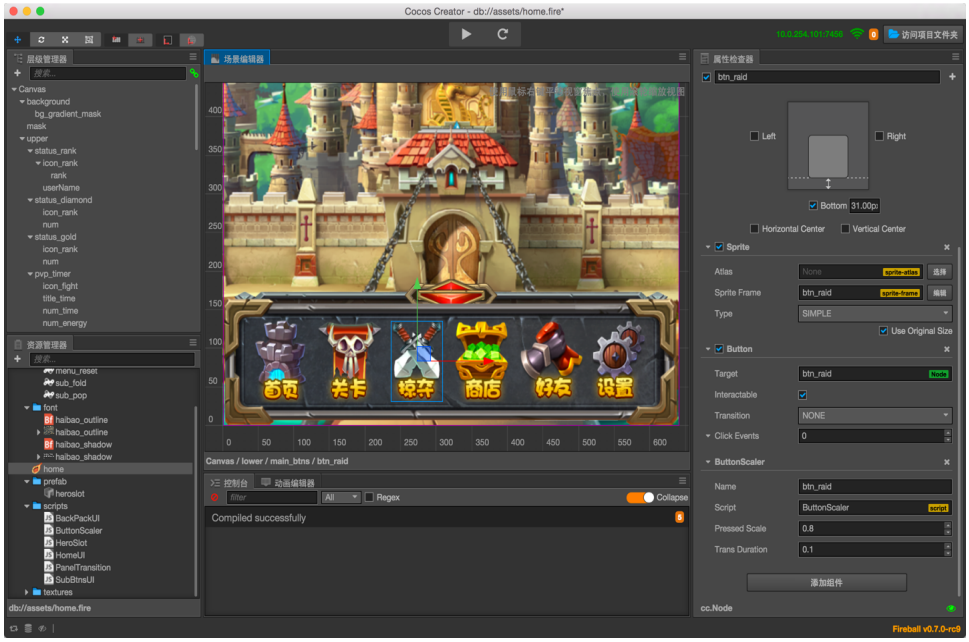
Obrázek 2.8: Ukázka ze hry Ingress (zdroj: [31])

2.3 Cocos2d-x

Cocos2d-x je multiplatformní, volně dostupný, open source softwarový framework pod licencí MIT, používaný pro tvorbu her, aplikací nebo interaktivních programů (např. animované pozadí). Existuje více verzí tohoto frameworku. Zde jsou nejznámější z nich a programovací jazyky, které lze v nich použít:

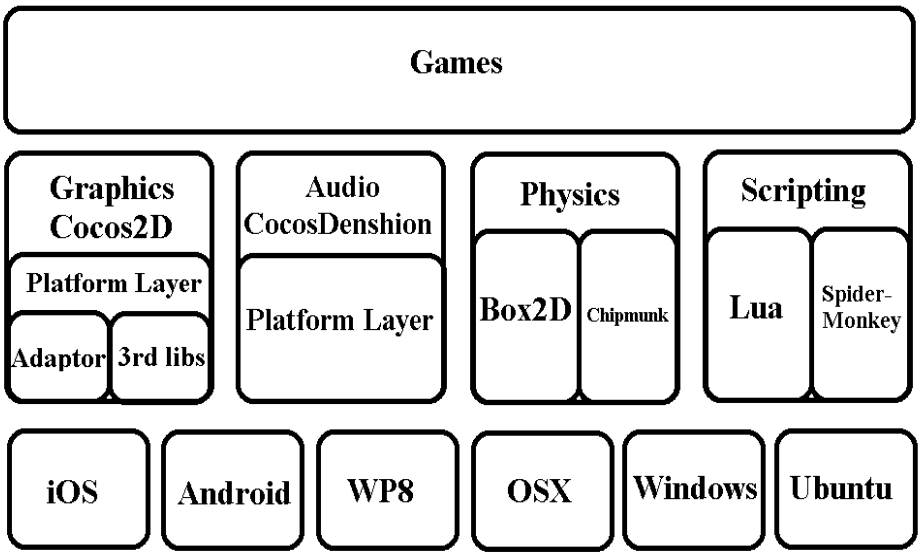
- Cocos2d-x (C++, Lua, JavaScript)
- Cocos2d-JS (JavaScript)
- Cocos2d-XNA (C#)
- Cocos2d-Swift (Objective-C, Swift)
- Cocos2d (Python)

Poskytuje mnoho užitečných funkcí, jako je například vykreslování grafiky, grafické uživatelské rozhraní, zvuk, fyziku, vstupy od uživatelů, správu paměti nebo například síťování. Jádro tohoto frameworku je napsáno v jazyce C++. Framework obsahuje herní engine a konzolové okno pro zadávání příkazů. Užitečným nástrojem je Cocos Creator, viz obr. 2.9. Vytváří se v něm aplikace pro Android, iOS, Windows a Mac s funkcemi zaměřenými na nativní mobilní platformy [27].



Obrázek 2.9: Cocos Creator (zdroj: [25])

Architektura Cocos2d-x se dělí do tří hlavních vrstev, přičemž každá z nich obsahuje další podvrstvy, viz obr. 2.10. Z těchto třech hlavních vstev se skládá aplikace/hra [1].



Obrázek 2.10: Cocos Architecture (zdroj: [1])

Vybrané známé hry na mobilní zařízení vytvořené v Cocos2d-x:

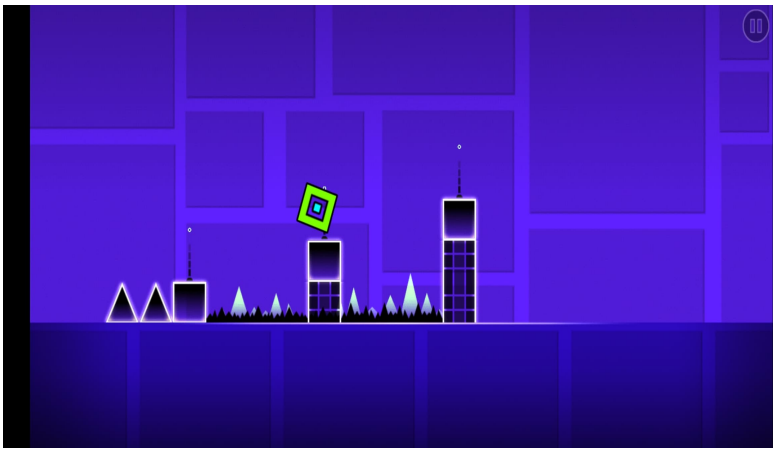
1. **Hill Climp Racing**

   Závodní 2D hra, ve které je cílem ujet co nejdelší vzdálenost, sbírat body a benzín, který během cesty ubývá, viz obr. 2.11.

Obrázek 2.11: Ukázka ze hry Hill Climb Racing (zdroj: [16])

2. **Geometry Dash Lite**

Skákací hra, kde se snažíte vyhnout nebezpečným objektům pro získání co nejvyššího skóre, viz obr. 2.12.



Obrázek 2.12: Ukázka ze hry Geometry Dash Lite

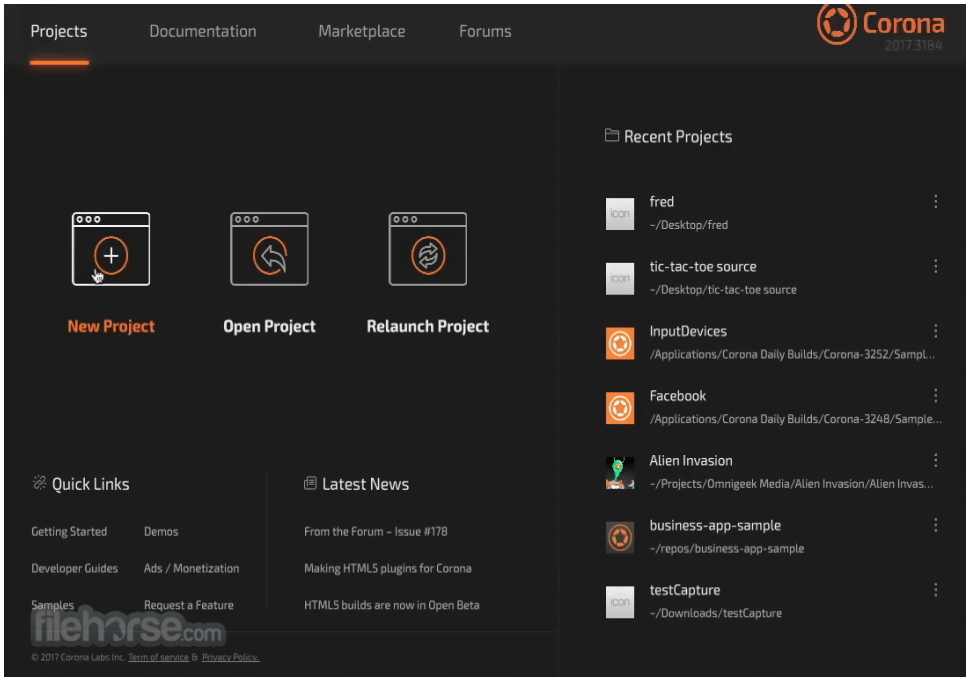
3. **Castle Clash: Age of Legends**

Strategická hra, kde stavíte vlastní vesnici, cvičíte vojáky a bojujete proti jiným hráčům, viz obr. 2.13.

Obrázek 2.13: Ukázka ze hry Castle Clash: Age of Legends (zdroj: [25])

2.4 Solar2D/Corona SDK

Solar2D (původně Corona SDK) je bezplatný, multiplatformní framework pro vytváření her a aplikací na mobilní zařízení a desktopové systémy. Pro tvorbu her se používá skriptovací jazyk *Lua*. Framework dále využívá více než 1000 vestavěných API rozhraní, široký výběr pluginů a rozšíření *Corona Native* pro programování v jazyce *C/C++/Obj-C/Java*. Obsahuje takzvané *Live Builds*, které zobrazují projekt na více zařízeních současně. Corona podporuje vývoj aplikací na všechny hlavní platformy (iPhone, iPad, Android, Amazon Fire, Mac, Windows), ale i například Apple TV, Fire TV a Android TV. Není tedy nutné přepisovat kód při změně platformy. API rozraní poskytuje přístup pro animace, zvuk, hudbu, Box2D fyzice, dále pokročilým grafickým filtrům, ovládání systému, síťování, správě textur, nativním prvkům a dalším. Dále se nabízí využít pro vytváření aplikací interaktivní Corona Simulator (viz obr. 2.14), který dokáže okamžitě reagovat na změny v aplikaci a poskytuje náhled v reálném čase - jak bude vypadat a jak se bude chovat na skutečných zařízeních [10].



Obrázek 2.14: Ukázka z Corona SDK (zdroj: [9])

Příklady her vytvořených pomocí Solar2D:

1. **Designer City**

   Hra zaměřená na budování a rozvoj města, viz obr. 2.15. Zaměřuje se na prvek tvorby a libovolný návrh města.

Obrázek 2.15: Ukázka ze hry Designer City (zdroj: [26])

2. **Grue the monster**

   Jde o tahovou RPG roguelike hru s prvky strategických hádanek, viz obr. 2.16. Hrajete za monstrum žijící v temné kobce a lovíte skřety, lidi, trpaslíky nebo elfy.

Obrázek 2.16: Ukázka ze hry Grue the monster (zdroj: [8])

3. **Merchant**

   Hra kombinující typické prvky RPG, viz obr. 2.17. Cílem hry je stát se co největším obchodníkem. Najímáte hrdiny, kterým vylepšujete vybavení a posíláte je na výpravy. Dále můžete najímat pracovníky a získávat peníze.

Obrázek 2.17: Ukázka ze hry Merchant (zdroj: [18])

3 Android

Android operační systém je mobilní operační systém. Jeho vývoj vede společnost **Google**. Používá se především pro zařízení s dotykovou obrazovkou, jako jsou mobilní telefony a tablety. Celý systém je ovládaný pohybem prstů po dotykové obrazovce (přejetí prstem, klepnutí), popřípadě pomocnými tlačítky umístěnými na mobilu. Tento operační systém se ale využívá i například v televizorech, automobilech nebo náramkových hodinkách. Zdrojový kód operačního systému *Android* je ve formátu **open-source**, aby pomohl prosazovat standardy mezi mobilními zařízeními [2].

V dnešní době je to nejpoužívanější operační systém pro mobilní zařízení. Android nabízí jednotný přístup k vývoji aplikací. Důvody, proč vyvíjet pro operační systém Android:

1. Open source
2. Široká vývojářská komunita (mnoho lidí Vám může pomoct při řešení problému)
3. Inter-App integrace (dva různé softwary se mohou propojit)
4. Kvalitní vývojové prostředí (obsahuje mnoho užitečných funkcí pro vývoj aplikací)
5. Nízké náklady na vývoj (pro vývoj není potřeba žádného drahého softwaru)
6. Jednoduchý vstup na trh (např. na Google Play Store stačí zaplatit $25 za vývojářskou licenci a dále nahrát aplikaci)

3.1 Vlastnosti

- Uživatelské rozhraní s mnoha funkcemi a moduly (dialogy, notifikace, menu)
- Připojení (GSM/EDGE, 5G, CDMA, Bluetooth, Wi-Fi, LTE, NFC, WiMAX)
- Multi-touch (možnost snímat více dotyků najednou)
- Úložný prostor (pro účely ukládání dat se používá odlehčená relační databáze **SQLite**)

- Multi-tasking (možnost běhu více tasků najednou a možnost přepínání mezi nimi)
- Nastavitelné widgety (jednoduchá aplikace)
- Vícejazyčný text (podporuje right-to-left i left-to-right text)
- Firebase Cloud Messaging (služba pro zasílání zpráv a oznámení pro Android, iOS a webové aplikace)

3.2 Úroveň API rozhraní

Při vývoji platformy Android se vydávají nové verze. Těmto jednotlivým verzím je vždy přiřazena jedinečná celočíselná hodnota, tzv. úroveň rozhraní API, viz tab. 3.1.

Název	**Úroveň API rozhraní**	**Verze**
BASE	1	1.0
BASE_1_1	2	1.1
Cupcake	3	1.5
Donut	4	1.6
Eclair	5 - 7	2.0 - 2.1
Froyo	8	2.2.x
Gingerbread	9 - 10	2.3 - 2.3.7
Honeycomb	11 - 13	3.0 - 3.2.x
Ice Cream Sandwich	14 - 15	4.0 - 4.0.4
Jelly Bean	16 - 18	4.1.x - 4.3.x
KitKat	19 - 20	4.4 - 4.4.4
Lollipop	21 - 22	5.0 - 5.1
Marshmallow	23	6.0
Nougat	24 - 25	7.0 - 7.1
Oreo	26 - 27	8.0 - 8.1
Pie	28	9
Android 10	29	10
Android 11	30	11

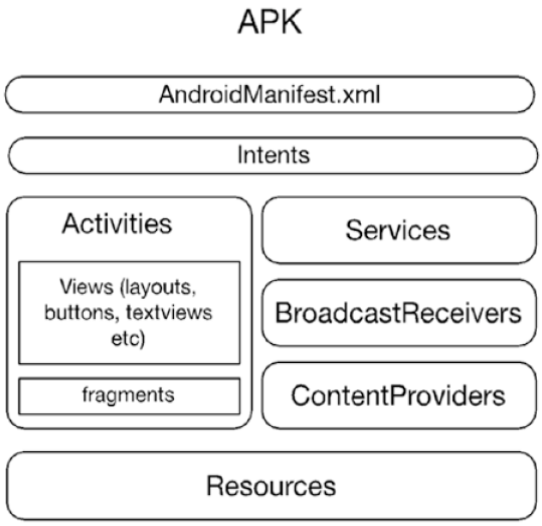
Tabulka 3.1: Android API

Aplikace jsou vyvíjeny za použití programovacího jazyka Java nebo Kotlin pomocí sady SDK (Software Development Kit). Po vytvoření je aplikace zabalena a dále se může distribuovat v aplikačních obchodech (Google Play,

SlideME, Opera Mobile Store, Amazon Appstore a jiné). Distribuovat aplikaci lze i jiným způsobem, než ji nahrát na aplikační obchod, například se může jednat o webovou stránku s odkazem na stažení přes nějaké úložiště [3].

3.3 Android projekt

Odlišnost aplikace pro Android od klasické desktopové aplikace je v tom, jak jsou aplikace strukturované, tedy Android aplikace se liší stukturálně od desktopových. Desktopová aplikace obsahuje všechny rutiny a podprogramy, které jsou využívány pro správné fungování dané aplikace. Dále může používat různé knihovny. Aplikace je tvořena volně spojenými komponenty, které mezi sebou komunikují pomocí předávání zpráv, viz obr. 3.1. Mezi takové základní kompomenty patří např. *Activities, Services, ContentProviders, BroadcastReceivers*, a jsou to klíčové stavební kameny aplikace. Jedná se o vysokoúrovňové abstrakce, které se používají např. pro zobrazování obrazovky uživateli, spuštění úlohy na pozadí a jiné. Komponenty jsou předkódované třídy se specifickým chováním. Těmto komponentám můžeme upravit nebo jim přidat další chování, které bude pro danou aplikaci jedinečné. V systému Android jsou aplikace vytvářeny pomocí komponent.



Obrázek 3.1: Struktura Android aplikace (zdroj: [30])

- **Activities** - prvek pro vizuální část aplikace
- **Services** - komponenta aplikace, která provádí operace na pozadí

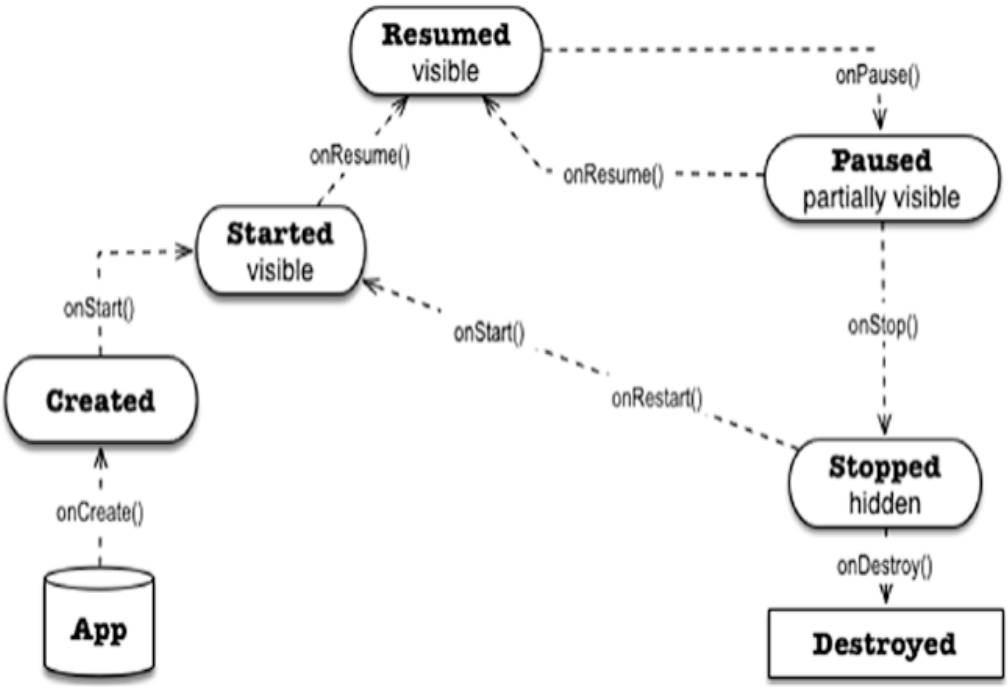
- **BroadcastReceivers** - umožňují aplikaci přijímat zprávy ze systému nebo jiných aplikací
- **ContentProviders** - pomocí této komponenty jsme schopni vytvářet aplikace, které mohou sdílet data s jinými aplikacemi, dále spravuje přístup k úložišti dat

3.3.1 Manifest

Je jedním z nejdůležitějších souborů. Jedná se o vstupní bod do aplikace. Definuje název aplikace, první zobrazenou aktivitu, využité komponenty, práva (přístup k internetu, souborům, kontaktům), externí knihovny, požadavky na hardware atd.

3.3.2 Aktivita

Tuto komponentu si můžete představit jako jednu obrazovku aplikace, která slouží pro interakci s uživatelem. Každá aktivita má při běhu aplikace vlastní životní cyklus, viz obr. 3.2.



Obrázek 3.2: Životní cyklus aktivity (zdroj: [30])

- *onCreate()* - metoda volána při spuštění aktivity (používá se pro inicializaci třídy)
- *onStart()* - metoda volána po prvním spuštění aplikace nebo opětovném aktivování aktivity
- *onResume()* - metoda pro přesunutí aktivity na popředí
- *onPause()* - metoda pro přechod aktivity na pozadí
- *onStop()* - metoda volána, pokud se má daná aktivita zastavit
- *onRestart()* - metoda pro opětovné spuštění aktivity
- *onDestroy()* - metoda pro zrušení aktivity

3.3.3 Intent

V objektově orientovaném programování je zvykem, že aktivace chování objektů spočívá ve vytvoření nového objektu a následném volání jeho metod. Jedná se o přímý a jednoduchý způsob vzájemné komunikace mezi objekty. Android využívá tzv. *Intent* pro vzájemnou komunikaci jednotlivých komponent, např. pro přepnutí z jedné aktivity na jinou [30].

4 Porovnání frameworků

Frameworků pro tvorbu her existuje velké množství. Pro porovnání jsem vybral čtyři ze známějších. Jednotlivé herní frameworky budou porovnávány z více různých pohledů, např. z podporované platformy, způsobu instalace a vytvoření projektu, funkce pro podporu vývoje mobilních aplikací, ceny, programovacích jazyků atd. Na konci kapitoly bude popsáno shrnutí porovnávání.

4.1 Podporované platformy

Jelikož každý z porovnávaných frameworků je multiplatformní a výsledná aplikace je spustitelná na více možných platforem, nemusí podporovat úplně stejné platformy jako ostatní, a pokud je podporují, nemusí se jednat o stejnou verzi.

- **Unity**
  - Windows, Linux, Mac OS X
  - iOS, Android, WebGL, Android TV, tvOS, ARCore
  - PlayStation 4, PlayStation 5, Xbox One, Xbox Series X, Series S
  - Nintendo Switch, Wii U, Oculus Rift, HoloLens, MagicLeap
- **LibGDX**
  - Windows, Linux, Mac OS X
  - Android (4.4 a vyšší), Blackberry, iOS, Java Applet
  - Javascript/WebGL (Chrome, Safari, Opera, Firefox, IE)
- **Cocos2d-x**
  - Windows, Linux, Mac OS X
  - Windows Phone 8, Android, iOS, Tizen
- **Solar2D/Corona SDK**
  - Windows, Linux, Mac OS
  - iOS, Android, tvOS, Android TV

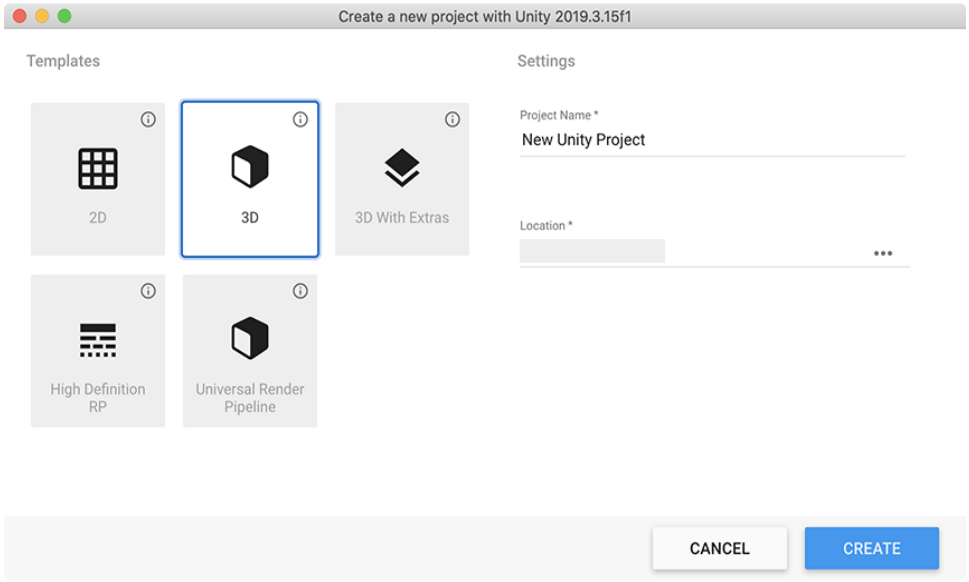
4.2 Instalace, vytvoření a nastavení projektu

Zde je popis instalace jednotlivých frameworků a vytváření nového projektu, popřípadě nastavení některých parametrů pro projekty (platfotmy pro výslednou aplikaci, název projektu, místo uložení projektu atd.).

4.2.1 Unity

Nejprve je potřeba stáhnout si soubor *Unity Hub* z oficiálních stránek, pomocí kterého získáme *Unity Editor*, a než dojde ke stažení editoru, je nutné se přihlásit nebo zaregistrovat a poté aktivovat licenci. Je možnost nastavit umístění složky editoru a jazyk. Při instalaci se zvolí podporované platformy [23].

Pro vytvoření nového projektu si stačí vybrat typ nového projektu, název projektu a umístění na disku, kde bude projekt uložen, viz obr. 4.1.



Obrázek 4.1: Vytvoření nového projektu (zdroj: [23])

- **2D** šablona pro 2D aplikace. Obsahuje osvětlení, textury, Scene view, Sprite packer, Ortografickou kameru.
- **3D** šablona pro 3D aplikace.
- **3D with extras** šablona pro 3D aplikace, která využívá vestavěné vykreslovací a *post-processing* funkce Unity. Dále obsahuje několik předvoleb pro rychlý start vývoje.

- **High Definition RP** šablona pro projekty, které používají platformy s *Shader Model 5.0*. Zahrnuje moderní vykreslování, obsahující pokročilé typy materiálů a osvětlení.
- **Universal Render Pipeline** šablona pro konfiguraci projektu, kde je nejdůležitějším faktorem především vysoký výkon, široká podpora platforem a dobré přizpůsobení grafiky. Obsahuje URP pro tvorbu optimalizované grafiky, a to i pro 2D grafiku. Dále používá Shader Graph pro tvorbu stínů.

4.2.2 LibGDX

Pro použití frameworku *LibGDX* není potřeba instalace žádného programu či aplikace. Stačí si pouze stáhnout příslušný JAR soubor z oficiálních stránek, který vygeneruje nový projekt, viz obrázek 4.2.

Obrázek 4.2: Generátor projektu LibGDX

Pro vytvoření nového projektu/aplikace je pouze potřeba vyplnit následující položky:

- **Name** – název aplikace
- **Package** – název balíčku aplikace
- **Game class** – název třídy
- **Destination** – umistění složky pro vytvořený projekt
- **Android SDK** – cesta k umístění Android SDK
- **Sub Projects** – cílové platformy aplikace
- **Extensions** – další knihovny a frameworky pro vývoj aplikace

Popis jednotlivých knihoven a frameworků, které nabízí **LibGDX**.

- **Bullet** – knihovna pro detekci kolizí nebo kontaktu
- **Freetype** – škálovatelné fonty
- **Tools** – sada nástrojů (particle editor pro 2D a 3D, bitmapový font a balíčky textur)
- **Controllers** – knihovna pro řízení ovladačů např. pro Xbox 360
- **Box2d** – knihovna s 2D fyzikou
- **Box2dlights** – framework pro 2D osvětlení, využívá **Box2D** pro *Raycasting* a *OpenGL ES 2.0* pro vykreslování
- **Ashley** – malý framework entit
- **Ai** – framework pro umělou inteligenci

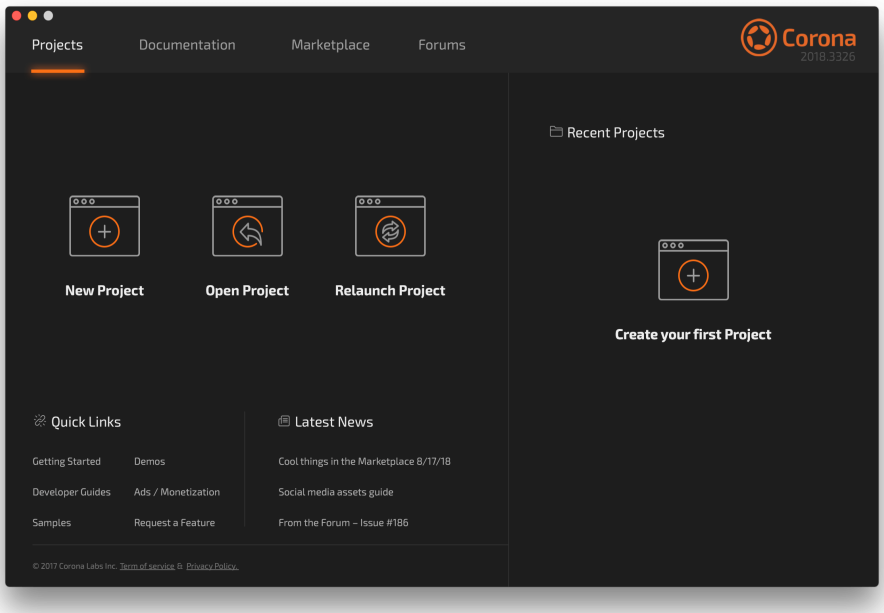
4.2.3 Solar2D/Corona SDK

Jedná se o podobný postup jako u **Unity**, tedy si stačí stáhnout instalační soubor z oficiálních stránek a zadat umístění na disku.

Je možné vytvořit si nový projekt nebo otevřít již existující, viz obr. 4.3. Při vytváření nového projektu se zadává název a umístění na disku. Dále je možnost vybrat si šablonu:

- **Blank** tato možnost vytvoří složku projektu a prázdný soubor *main.lua*

- **Tab Bar Application** víceokenní aplikace využívající panel karet
- **Physics Based Game** je aplikace s použitím fyziky a *Composer* pro správu knihoven
- **eBook** vícestránkové rozhraní používající *Composer* pro správu knihoven a zdrojů



Obrázek 4.3: Ukázka tvorby nového projektu v Corona (zdroj: [7])

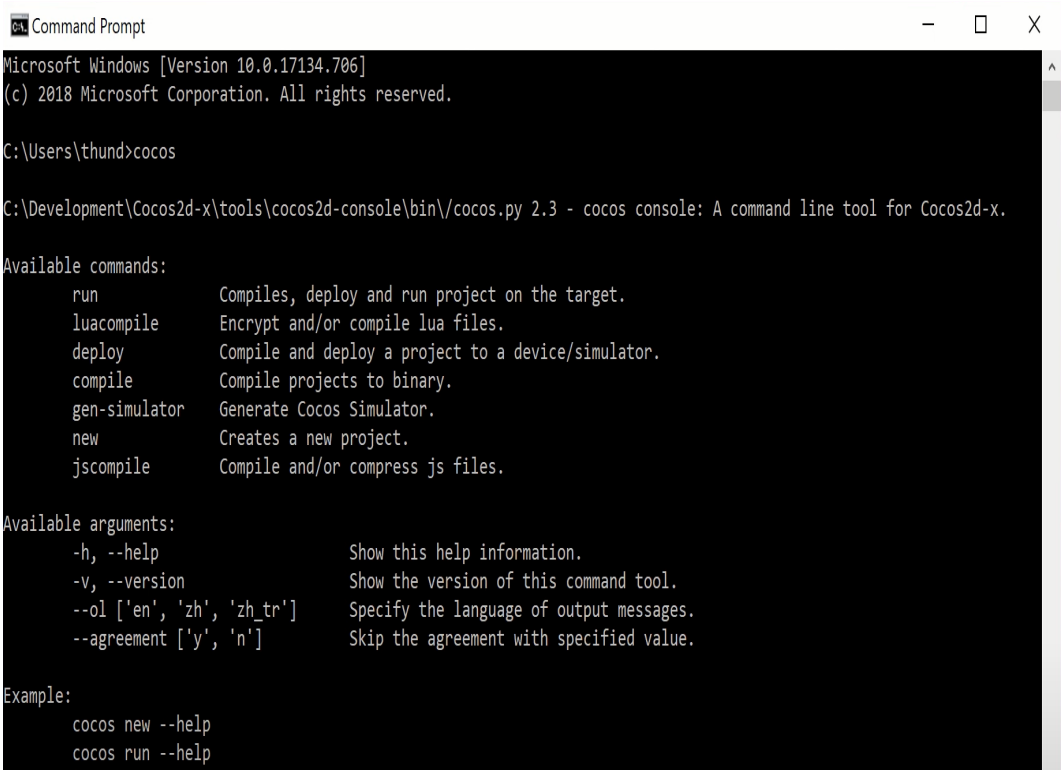
Také je možnost nastavit velikost obrazovky (telefon, tablet, nebo své vlastní, to znamená šířku a výšku) a výchozí orientaci (svislá nebo vodorovná).

V dolním části se nachází několik odkazů na stránky pro začátečníky, dema, vývojářské příručky, ukázky aplikací a další.

4.2.4 Cocos2d-x

Pro vývoj v tomto frameworku je potřeba Python 2.7.x, CMake (pro automatizaci překladu zdrojových souborů). Z oficiálních stránek Cocos2d-x stačí stáhnout *.zip* nebo naklonovat z **GitHub**. Není nutnost ručně nastavovat systémové proměnné, ale z příkazové řádky spustit pomocí Python souboru *setup.py*, který nastaví všechny systémové proměnné. Pro práci s *Cocos2d-x* se používá příkazová řádka, viz obr. 4.4. Příkazy začínají slovem *cocos*. Na výběr je několik příkazů, pomocí kterých pracujete s projekty.

- *run* - zkompiluje, nasadí a spustí projekt pro daný *target*
- *luacompile* - zkompiluje *lua* soubory
- *deploy* - zkompiluje a nasadí projekt pro dané zařízení nebo simulátor
- *compile* - zkompiluje projekt do binárních souborů
- *gen-simulator* - vygeneruje *Cocos Simulator*
- *new* - vytvoří nový projekt
- *jscompile* - zkompiluje *javascript* soubory



Obrázek 4.4: Ukázka tvorby nového projektu v Cocos2d-x

Pro vytvoření nového projektu se použije příkaz:
cocos new <name> -p <package> -l <language> -d <location>.

- *name* - název nového projektu
- *package* - název balíčku

- *language* - programovací jazyk
  - *cpp* - C++
  - *lua* - Lua
  - *js* - JavaScript
- *location* - umístění projektu na disku

Pokud by někomu nevyhovovalo vyvíjet aplikaci a přitom překládat, nasazovat a spouštět ji z příkazové řádky, je možnost využít vývojové prostředí např. *Visual Studio* [14].

4.3 Programovací jazyky

Pro jednotlivé frameworky se používají různé programovací jazyky. Přehled těchto programovacích jazyků, viz tab. 4.1.

Framework/Game engine	**Programovací jazyk**
Unity	C++, C#
LibGDX	Java, C++/C
Cocos2d-x	C++, JavaScript, Typescript
Solar2D/Corona SDK	Lua, C++/C, Java, Objective-C

Tabulka 4.1: Podporované programovací jazyky

4.4 Funkce, nástroje

Funkce a nástroje, lze při vývoji aplikací použít pro zrychlení vývoje nebo dosažení lepšího výsledku. Funkce a nástroje jsou rozdělené do částí podle použití.

4.4.1 Unity

Popis funkcí a nástrojů pro Unity. Více informací naleznete zde [24].

Grafika

- **LOD** (level of detail) - funkce pro ovládání komplexnosti *Mesh*, podle vzdálenosti od kamery. Pokud se kamera nachází blízko, je možné vykreslit síť podrobněji. Pokud je naopak kamera dál, vykreslí jednoduchou sít, jelikož všechny detaily stejně nebudou viditelné. Jedná se o velmi užitečnou funkci, díky které se nemusí detailně vykreslovat vzdálené objekty a nechávat tak více prostoru pro hratelnost.

- **Video playback and streaming** - tato funkce slouží pro streamování videa z Internetu. Pro mobilní platformy to znamená jednu nevýhodu. Video lze přehrát pouze na celou obrazovku a nemůže být použito, jako textura pro efekty, protože toto video prochází integrovaným systémem přehrávání videa.

- **Lightmapping with global and area lights** - podporuje tvorbu složitých statických stínů a světelných efektů. Pro tvorbu co nejvíce realistických scén se používá globální a prostorové osvětlení. Unity podporuje import vlastní světelné mapy z jiného programu.

- **Static batching** - funkce pro urychlení procesu vykreslování. Nevykresluje jednotlivé objekty, ale spojuje je do jedné sítě, a tím snižuje počet volání na vykreslení jednotlivých objektů. Urychluje běh hry.

Zvuk

- **Audio filter** - zvukový filtr umožňující přidávat efekty do zvukových klipů za běhu. Například je možné vytvořit pro postavu zvuk stop na štěrku. Pokud postava bude chodit, bude znít jako klasický zvuk pohybu, ale pokud vejde postava např. do tunelu, tato funkce upraví daný zvuk, aby to znělo jako kroky vycházející z tunelu (možnost přidat i časovou prodlevu). V projektu tedy není potřeba velkého množství zvuků.

Analytické nástroje

- **Rendering Statistics Window** - statistické okno, které poskytuje informace o hře (vykreslování, zvuk, FPS, zpracování procesorem, počet připojených hřáčů). Používá se pro optimalizaci výkonu.

- **Profiler** - analytický nástroj. Poskytuje informace o běžící hře (využití procesoru, paměť, vykreslování).

Hledání cest

- **NavMeshes** - funkce pro hledání cest a pro simulaci davu. Jedná se o integrovaný systém umožňující najít cestu z jednoho bodu do jiného. Může se použít v případě, kdy se postava musí dostat do nějakého cíle, a v cestě jsou objekty, kterým se má vyhnout.
- **NavMesh - dynamic obstacles and priority** - funkce zabývající se hledáním cest. Můžete přiřadit postavám priority, podle kterých budou vybírat cestu k cíli.

Užitečné funkce

- **Native plugins support** - podpora vlastních knihovem ve formě DLL (Dynamic Link Library).
- **Simple device connection** - jednoduché propojení mobilních zařízení s Unity pro testování mobilních aplikací.
- **Unity remote** - velmi užitečná aplikace pro mobilní zařízení podporující platformu Android. Umožňuje propojení mobilního zařízení s Unity editorem pro testování aplikace.

4.4.2 LibGDX

Popis funkcí a nástrojů pro LibGDX. Více informací naleznete zde [15].

Grafika

- **Spine** - nástroj pro tvorbu 2D animací.
- **HyperLap2D** - vizuální editor pro tvorbu komplexních 2D scén.
- **Particle Editor** - výkonný nástroj s jednoduchým uživatelským rozhraním pro tvorbu efektů.
- **Tiled** - jednoduchý bezplatný nástroj pro tvorbu samostatných úrovní, úpravou map různých forem, umístění obrázků, anotaci úrovně podle informacích použitých ve hře.
- **Texture Packer** - součást *LibGDX* pro ukládání velké textury do více menších obrázků. Používá se pro zrychlení ukládání textur.
- **User Interface** - LibGDX podporuje mnoho programů pro tvorbu uživatelského rozhraní, nebo možnost stažení již existujících vzhledů.

- **Texture Packer GUI** - jednoduchý nástroj pro zabalení a správu textur.

Testování

- **gdx-dbgagent** - Java agent pro ladění hry (debugging).
- **HotSwapping** - možnost provádět změny za běhu programu bez přerušení běhu aplikace.

Služby

- *gdx-facebook* - podpora pro *Facebook Graph API*.
- *gdx-fireapp* - multiplatformní API pro Firebase.
- *gdx-firebase* - Android/Desktop API pro Firebase.
- *gdx-gameanalytics* - analytický REST API klient pro hry.
- *gdx-gamesvcs* - jednoduchá integrace herních služeb (Google Play, Apple Game Center).
- *gdx-pay* - služba nabízející nákup extra obsahu ve hře za skutečné peníze.
- *steamworks4j* - wrapper pro Java aplikaci pro přístup do Steamworks API (sada nástrojů pro přípravu a distribuci her pro *Steam*).

Užitečné funkce

- **KTX** - rozšíření pro podporu programovacího jazyka Kotlin.
- **Input Handling** - podpora vstupů z myši, klávesnice, dotyková obrazovka (detekce gest - klepnutí, posouvání, přiblížení, dvojitý poklepání), ovladače, joysticky, compass, gyroskop, akcelerometr.
- **Physics** - fyzika her, je obsažena v několika knihovnách. *Box2D* je populární knihovna pro 2D hry. *Bullet* je knihovna pro detekci 3D kolizí. *Jbump* je knihovna pro jednoduchou implementaci pro skákací 2D hry.
- **Artificial Intelligence** - umělá inteligence pro řízení chování, hledání cest, strom chování, konečné automaty, chování skupin.

4.4.3 Cocos2d-x

Popis funkcí a nástrojů pro Cocos2d-x. Více informací naleznete zde [27].

Grafika

- **Cocos Creator** - stejně jako u herního enginu Unity i Cocos má vlastní nástroj pro tvorbu aplikací. Pomocí tohoto editoru můžete jednoduše tvořit jednotlivé scény hry a programovat logiku hry, nebo konfigurovat chování kamery pro 2D i 3D.
- **Texture Auto Trim** - pokud budete v aplikaci používat obrázek, jehož obsah je pouze ve středu a okolo je prázdné místo, tato funkce ořízne obrázek, aby neobsahoval zbytečné prázdné části okolo obrázku.
- **Preload** - možnost načtení nové scény v pozadí aplikace v době, kdy ještě v popředí běží jiná scéna.
- **Director** - koncept pro řízení nahrazování scén a přechod z jedné scény do další.
- **Scene Graph** - datová struktura pro uspořádání grafických scén. Scény jsou uspořádány do stromové struktury.
- **3D Physics Manager** - systém pro simulaci fyziky, detekci kolizí a výpočet síly objektu při pohybu.
- **Agora RTC** - funkce pro přehrávání videa, zvuku, živého vysílání.

Analýza a monitorování

- **Analytics Kit** - nabízí velkou škálu přednastavených analytických modelů pro získání hlubšího pohledu na uživatele, produkty a obsah. Informace z těchto pohledů se mohou použít pro optimalizaci aplikace.
- **Crash Service** - služba pro analýzu pádů aplikace a monitorování (výhodou je, že je možná integrace do aplikace bez nutnosti něco programovat), pád aplikace se kategorizuje pro upřednostnění nejdůležitějšího z nich. O každém pádu aplikace poskytuje informace. Poskytuje informace i o aplikaci, operačním systému a zařízení. Pád aplikace může být detekován v reálném čase a po povolení zasílání upozornění může zasílat aplikace email.
- **APM** - služba poskytující monitorování aplikace, dále je možné si prohlížet a analyzovat data o výkonu aplikace.

Užitečné funkce

- **Cache Manager** - modul, který spravuje mezipaměti pro soubory, stažené z jiných než webových platforem.
- **Remote Configuration** - dovoluje vzdálenou konfiguraci pro správu parametrů online (správa oprávnění, verzí, podmínek, parametrů). Může tak měnit chování a vzhled aplikace online, aniž by byla vyžadována aktualizace aplikace.
- **App Messaging** - pro posílání relevantních zpráv uživatelům, např. nabídku předplatného určitého produktu, tipy pro úspěšné dokončení úrovně, doporučení restaurace.
- **App Linking** - umožňuje vytvářet odkazy napříč platformami. Po kliknutí na daný link budete přesměrováni na daný obsah aplikace.
- **Auth Service** - služba pro rychlé sestavení a zabezpečení. Poskytuje spolehlivý autentizační systém ověřování uživatelů.
- **SDKHub** - sada frameworků pro snadný přístup k nativním SDK.
- **Location Kit** - sada kombinující funkce GPS, Wi-fi a lokalizační funkcí pro globální určení polohy.

4.4.4 Solar2D/Corona SDK

Popis funkcí a nástrojů pro Solar2D. Více informací naleznete zde [28].

Grafika

- **Animation** - animace se vytvoří pomocí obrázkového listu. Tento list je rozdělený na rámce. Vykreslení jednotlivých rámců tvoří animaci.
- **Adaptive Scaling** - popisuje, jak pomocí adaptivního škálování přizpůsobit virtuální pixely, aby se chovaly jako nativní iOS/Android virtuální pixely.
- **Composer Library** - oficiální knihovna pro tvorbu a správu scén (obrazovek). Poskytuje vývojářům snadný způsob, jak vytvářet nové scény a přechod mezi jednotlivými scénami.
- **Custom Fonts** - možnost přidat a použít vlastní vložený styl písma pro stylizaci aplikace napříč platformami.

Testování

- **Simulator** - testovací prostředí pro aplikaci. V simulátoru je možné vidět, jak aplikace vypadá a jak bude fungovat při nasazením do skutečného zařízení. Umožňuje zobrazení změny kódu v reálném čase.
- **Console window** - konzolové okno zobrazující diagnostické zprávy o tom, co se děje v programu. Automaticky se zobrazí při spuštění simulátoru.
- **Live Build** - slouží k efektivnímu vývoji aplikace. Stačí pouze jednou sestavit a nasadit aplikaci. Poté budou jakékoliv změny kódu okamžitě viditelné i při používání nativních doplňků (např. poskytovatele reklam, síťové hry). Je možné vidět chování aplikace na více různých platformách. Tato funkce je velmi užitečná při testování.

Užitečné funkce

- **ATS** (App Transport Security) - poskytuje novou ochranu pro Apple zařízení před nezabezpečeným *HTTP* a starším *HTTPS* připojením.
- **Facebook** - integrace aplikace a Facebooku pomocí *Facebook Developer Portal* pro implementaci běžných úkolů a procesů (přihlášení, odhlášení, requesty).
- **API references** - v projektu je možné použít mnoho existujících API, které jsou dostupné na oficiálních stránkách. Je možnost si vybrat v seznamu obsahující více než 1000 API (např pro Facebook, zvuk, kolize, gyroskop, myš, operační systém a mnoho dalších).
- **Plugins** - *Solar2D* podporuje third-party služby. Slouží k přidáním reklam a zpeněžnění obsahu (AdMob), analýzu (Google Analytics), zabezpečení (OpenSSL), nákup v aplikaci (Google IAP) a další pomocné programy (Memory Bitmap, Notifications, iCloud).
- **System/OS Libraries** - knihovny pro shromažďování systémových informací a funkcí řídicího systému - získávání informací o zařízení, povolení multi-touch (vícedotykové ovládání), ovládání doby nečinnosti, akcelerometru, GPS a dalších informací souvisejících s operačním systémem.

4.5 Minimální systémové požadavky

V následujících seznamech jsou uvedeny minimální požadavky na operační systémy pro práci s jednotlivými frameworky.

4.5.1 Unity

- **Windows** - Windows 7 (64-bit), Windows 8, Windows 10
- **Mac** - mac OS 10.12+
- **Linux** - Ubuntu 16.04, Ubuntu 18.04, CentOS 7

4.5.2 LibGDX

Na internetu nebyly nalezeny žádné informace, jaké jsou minimální požadavky pro operační systém. Pravděpodobně postačí mít JDK 8+ a Android SDK. Popřípadě RoboVM OSS Intellij plugin pro vytváření aplikací pro iOS.

4.5.3 Cocos2d-x

- **Windows** - Windows 7 (64-bit), Windows 8, Windows 10
- **Mac** - mac OS 10.9+

4.5.4 Solar2D

- **Windows** - 7, 8, 10 (32-bit nebo 64-bit)
- **Mac** - mac OS 10.11+

4.6 Typ her

Hry se rozdělují (z pohledu počtu dimenzí) na 2D a 3D hry. Frameworky podporující tyto typy her, viz tab. 4.2. Z pohledu žánru (skákací, akční, závodní atd.) je možné pomocí každého frameworku, vytvořit hru založenou na tomto žánru.

Framework/Herní engine	2D	3D
Unity	ano	ano
LibGDX	ano	ano
Cocos2d-x	ano	ano
Solar2D/Corona SDK	ano	ne

Tabulka 4.2: Typy her

4.7 Cena

Většina herních frameworků je zdarma, ale mohou mít některé funkce omezené. Tyto funkce jsou dostupné po zaplacení měsíční/roční licence. Přehled cen pro jednotlivé frameworky, viz tab. 4.3. Ceny v tabulce se vztahují k dubnu 2021.

Framework/Game engine	Cena/měsíc	Cena/rok
Unity Student/Personal	Zdarma	Zdarma
Unity Plus	$40	$399
Unity Pro	$150	$1800
Unity Enterprise	$200	$2400
LibGDX	Zdarma	Zdarma
Cocos2d-x	Zdarma	Zdarma
Solar2D/Corona SDK	Zdarma	Zdarma

Tabulka 4.3: Ceny za licence měsíčně/ročně

4.8 Komunita

Komunita uživatelů je užitečná, pokud nastane jakýkoliv problém týkající se daného frameworku. Postačí napsat otázku na některou z platforem z následujícího seznamu.

- **Unity** - Discord, oficiální stránky (forum, blog, live help atd.), Reddit
- **LibGDX** - Discord, Reddit
- **Cocos2d-x** - oficiální stránky (Blog, forum), Reddit
- **Solar2D/Corona SDK** - Discord, oficiální stránky (forum)

4.9 Porovnání vlastností frameworků

Přehled některých vlastností jednotlivých frameworků, viz tab. 4.4.
x - nepodporuje, o - podporuje

	Unity	**LibGDX**	**Cocos2d-x**	**Solar2D**
Open-source	x	o	o	o
Virtuální realita	o	o	o	x
GPS	o	o	o	o
Bluetooth	o	o	o	o
Akcelerometr	o	o	o	o
Gyroskop	o	o	o	o
Správa paměti	o	o	o	x
Ortografická kamera	o	o	o	x
Animace	o	o	o	o
Podpora ovladače	o	o	o	o
GUI/Editor	o	x	o	x
Analytické nástroje	o	x	o	o

Tabulka 4.4: Přehled vlastností

4.10 Další funkce a nástroje pro vývoj

V následujícím seznamu jsou další užitečné funkce frameworků, které se také dají použít při vývoji aplikace. Některé z těchto funkcí není možné použít pro mobilní aplikace např. požaduje velké nároky na procesor atd.

4.10.1 Unity

Více informací o jednotlivých nástrojích a funkcích naleznete zde [24].

- **HDR** (High dynamic range) - funkce pro mapování tónů, se kterými se vytváří realističtější světelné efekty. HDR dovoluje systému používat hodnoty přesahující rozsah pro vykreslování (standardně hodnoty od 0 do 1) a mapování tónů. Nevýhoda je delší čas potřebný pro zpracování procesorem.

- **Render-to-texture effects** - umožňuje použít výstup z kamery jako texturu. Tuto texturu pak použít na *Mesh* a tím působit jako bezpečnostní kamera. Může být náročné na procesor.

- **Fullscreen post-processing effects** - funkce pro přidání efektů na scénu jako rozostření při pohybu pro efekt rychle pohybujícího se tělesa nebo vírový efekt pro pokřivení obrazu. Jedním z nejlepších efektů je použití neonové záře. Velmi náročná na procesor a prakticky nepoužitelná pro mobilní zařízení.

- **Occlusion culling** - optimalizační funkce. Standardní kamerový sytém vykresluje vše, co se nachází v zorném poli kamery a prostoru. Tato funkce nám dovolí nastavit objemy v prostoru, kam může vstoupit kamera. Tyto objemy se používají k výpočtu toho, co kamera ve skutečnosti vidí. Pokud je pohled kamery nastaven na zeď, je zbytečné, aby se vykresloval i prostor za ní, který není vidět.

- **Deferred rendering** - funkce pro tvorbu vysoce podrobného osvětlení a stínů. Jedná se o víceprůchodový proces pro výpočet herního světla a detailů stínů. Je to nákladný proces, pro plné využití je potřeba dobrá grafická karta. Proto tato funkce nelze použít pro mobilní zařízení.

- **Dark skin** - kosmetická funkce. Pro tvorbu hladkého vzhledu tmavé pleti.

- **Asset bundles** - poskytují funkci, kdy můžete přidávat nový obsah do projektu a tento nový obsah streamovat uživatelům bez potřeby aktualizace (nové postavy, úrovně).

- **Asset Store** - nejedná se o funkci, ale o knihovnu s balíčky, které mohou být implementovány do projektu, a tím ušetřit čas. Tyto balíčky tvoří členové komunity. Může se jednat o animace, textury, objekty, modely, ale i celé projekty či rozšíření editoru.

- **Real-time points and soft shadow** - funkce pro překonání tzv. *Blob shadow* a místo nich používá realistické stíny. Bohužel tato funkce nefunguje pro většinu mobilních zařízení. Tato funkce se může hodit, pokud chcete realistické stíny a přímé světlo.

- **Mecanim** - nový animační systém s mnoha funkcemi, např. IK (Inverse Kinematics). Ta pomáha definovat cílový bod animace a systém se pokusí přijít na to, jak se do tohoto bodu dostat. Dále umožňuje synchronizovat vrstvy, které budou udržovat více animačních stavů najednou (pohyb člověka podle stavu zdraví nebo velikost gravitace ovlivňující objekt podle hmotnosti).

- **Light probes** - jedná se o sondy, které zjistí, jak má být daný předmět osvětlen. Pokud se objekt pohybuje okolo této sondy, bude mu od této sondy předána informace, jak má vytvořit stín. Na objekt samozřejmě dopadá světlo ze scény, ale jsou nastavené limity, kolik světel může svítit na jeden objekt v jeden okamžik. Sondy provádějí předem složité výpočty, což umožní lepší stínování za běhu, ale může to mít dopad na výpočetní výkon.

4.10.2 LibGDX

Více informací o jednotlivých nástrojích a funkcích naleznete zde [15].

- **KryoNet** - knihovna poskytující jednoduché API pro efektivní síťovou komunikaci mezi klientem a serverem pomocí TCP a UDP, používající Kryo knihovnu k přenosu po síti.
- **Gdx.net** - pro jednoduché síťování(TCP sockety, HTTP requesty).
- **Assets** - sada bezplatných vysoce kvalitních assetů pro vytvoření hry. Většinou se jedná o online repozitáře s velkým obsahem assetů pro zvukové efekty a úryvky, 2D a 3D art nebo různou grafikou.
  - *Kenney Assets*
  - *OpenGameArt.org*
  - *Game-Icons.net*
  - *bfxr.net*
  - *freesound.org*
- **VR support** - podpora pro virtuální realitu.
- **noise4j** - jednoduchý generátor map založený na různých programech pro generování procedurálního obsahu (mapa je generována algoritmicky podle vstupních dat).
- **gdx-jnigen** - knihovna, která dovoluje psát C/C++ kód přímo se zdrojovým kódem Java.
- **Controllers** - funkce pro podporu ovladačů a joysticků.
- **Skin Composer** - nástroj pro tvorbu uživatelského rozhraní. Obsahuje živé náhledy a možnosti úpravy např. barva, velikost atd.

- **Netty** - asynchronní síťový aplikační rámec založený na událostech. Používá se pro rychlý vývoj výkonných protokolů pro servery i klienty.
- **Talos** - nástroj pro tvorbu vizuálních efektů.
- **fbx-conv** - konzolový nástroj pro konvertování souborů FBX/Collada/Obj do známých souborů pro *LibGDX*.
- **Flame** - nástroj pro tvorbu 3D efektů. Plně využívají pohybu ve 3D prostoru. Tento nástroj je součástí *LibGDX*.
- **Vizual Effects** - stejně jako pro uživatelské rozhraní, existuje velké množství programů a nástrojů pro tvorbu vizuálních efektů a animací. Některé jsou již součástí frameworku (Box2DLights).

4.10.3 Cocos2d-x

Více informací o jednotlivých nástrojích a funkcích naleznete zde [27].

- **Asset Manager** - během vývoje hry je obecně potřebné k obohacení obsahu hry použít velké množství obrázků, zvukové a další zdroje. Tento modul pomáhá vývojářům se správou využití zdrojů.
- **Assets import** - možnost jednoduchého importování Assetu (přetáhnutí souboru do Cocos Creator) např. zvuk, fonty, formáty souborů, komprese a další.
- **Node Tree** - kombinací uzlů a komponentů je možné vytvořit ve scéně všechny druhy obrázků, textových a interaktivních prvků. Tento strom definuje uspořádání jejich pořadí vykreslení a hierarchie.
- **Modularize script** - Cocos Creator umožňuje rozdělit kód do více různých souborů (Python, C#, C/C++, HTML), a navzájem je mezi sebou volat. Modularizace umožňuje odkazovat na další skript (volání jeho metody, přístup k parametrům, použít nebo zdědit komponenty).
- **Skybox** - wrapper scény, které ukazují svět jak vypadá mimo geometrii. Dá se například využít pro simulaci nekonečné oblohy, hor a jiných jevů.
- **fbx-conv** - nástroj pro převod FBX souborů do souborů, které podporuje *Cocos2d-x*.
- **Ray Cast** - funkce pro získání prvního průniku paprsku a objektu, na který paprsek směřuje.

- **Node pool** - tvorba a mázání uzlů za běhu je velmi nákladná (doporučuje se to dělat při změně scény). Node pool je kolekce opakovatelně používatelných nodů pro zvýšení výkonu. Uzel definuje vstup, výstup, nebo operaci v Shader Graph, může mít libovolný počet vstupních i výstupních portů. Shader Graph vytvoříte spojením těchto portů pomocí hran.
- **Plugin script** - slouží k importování third-party knihoven.
- **Networking** - podpora standardního síťového rozhraní pro krátká spojení (XMLHttpRequest) a pro dlouhá spojení (WebSocket).
- **Cloud DB** - databáze poskytující API pro správu dat nebo datové modely. Kromě zajištění spolehlivosti, konzistence a zabezpečení dat umožňuje synchronizaci dat mezi zařízením a cloudem.
- **Cloud Storage** - škálovatelné a bezúdržbové úložiště, které umožňuje bezpečně ukládat velké objemy dat (obrázky, videa, zvuk). Přenos je zajištěn HTTPS protokolem a ukládáním v cloudu pomocí zabezpečených šifrovacích protokolů.
- **Cloud Functions** - jsou jednoúčelové funkce. V mnoha případech se logika aplikace přenáší na server, aby nedocházelo k manipulaci na straně klienta. Cloud Functions jsou plně izolované od klienta, tedy funkce jsou soukromé a zabezpečené.

4.10.4 Solar2D

Více informací o jednotlivých nástrojích a funkcích naleznete zde [28].

- **Development environment** - celé vývojové prostředí se skládá ze dvou částí (konzolovém okně a simulátoru).
- **Hardware requirements** - doporučené hardwarové požadavky nejsou tak vysoké pro tvorbu aplikací (Procesor 1 GHz, RAM 1 GB).
- **Sandbox** - z bezpečnostních důvodů běží aplikace v *Sandboxu*. Má tedy omezený přístup k souborům, paměti, síťovým prostředkům atd. Soubory, obrázky, data a další jsou uloženy na místě, kam nemá přístup jiná aplikace.
- **Live Server** - aplikace pro sychronizaci změny projektu a lokální sítě. Simulátor spouští a automaticky přidává Live Build po vytvoření do Live Serveru.

- **Snapshots** - dovoluje pořídit snímek skupiny objektů jako jeden obrázek. Při úpravě jednotlivých objektů se bude aktualizovat i pořízený snímek.
- **Controllers** - podpora pro používání gamepadu, joystiku a jiných ovladačů.
- **MFi Controllers** - rozšíření Controllers pro Apple ovladače, které se mohou spárovat s iOS, tvOS nebo macOS zařízeními
- **LFS** - výkonná knihovna, umožňující provádět činnosti souborového systému (získávání atributů souborů, tvorba a odstraňování adresářů a iterace nad soubory v adresáři nebo časová razítka). Poskytuje i pokročilejší funkce, jako je sdílení souborů.
- **CoronaCards** - způsob, jak přidat interaktivní obsah do jiných platforem nebo frameworků (Unity, Xamarin, Appcelerator, Apache Cordova).

4.11 Shrnutí

Pokud bychom shrnuli vlastnosti jednotlivých frameworků, zjistíme, že každý z nich je velmi užitečný v nějakém směru. Každý podporuje velké množství platforem, instalace, a založení nového projektu není zvlášť složité. Každý obsahuje užitečné funkce a nástroje, které jsou od ostatních frameworků unikátní a některé jsou podobné. Například Unity podporuje mnoho funkcí pro vytvoření co nejvíce reálného vykreslení herní scény, světla a stínů, LibGDX obsahuje mnoho nástrojů pro tvorbu scén a uživatelského rozhraní, Cocos2d-x má editor, pomocí kterého lze jednoduše tvořit scény aplikace a Solar2D má užitečné funkce pro testování aplikace.

Při výběru frameworku hraje velkou roli programovací jazyk, ve kterém chce programátor vyvíjet aplikace např. C# nemusí podporovat každý herní framework. Komunita u všech frameworků je celkem rozsáhlá a vždy se někdo najde, kdo dokáže poradit s řešením problému. Nelze jednoznačně určit, který z frameworků je lepší než ostatní.

5 Tvorba vlastní aplikace

Jedním z bodů této bakalářské práce je vytvořit hru s využitím jednoho z analyzovaných herních frameworků. Hra by měla využít výhod daného frameworku. V první řadě bude odůvodněn výběr frameworku a vývojového prostředí. Dále bude popsána hra, která bude implementována (scény, životní cyklus, ovládání atd.).

5.1 Analýza

Před samotným vývojem aplikace je třeba se podívat na několik bodů, kterými se budeme zabývat, než začneme psát zdrojový kód.

5.1.1 Framework

V rámci analýzy jsem si zkoušel vytvořit malou hru v *Unity* a v LibGDX. Jako framework bude při vývoji použitý LibGDX. Pro tento framework jsem se rozhodl na základě mých zkušeností s jazykem Java. Jelikož ve frameworku *LibGDX* lze vytvořit jakýkoliv typ hry, rozhodl jsem se vytvořit 2D hru, protože tomuto typu her dávám přednost a tento typ her je jednodušší na vytvoření.

5.1.2 Popis hry

Pro vytvoření hry jsem se inspiroval hrou *Soul Knight*, viz. obr. 5.1. Hra bude mít pixelovou grafiku, jako mají staré retro hry. Tento typ grafiky jsem vybral, protože pixelová grafika je moje oblíbená. Nepůjde však o procházení jednotlivých místností, ale o nejdelší přežití a získání nejvyššího skóre v jedné místnosti, kde se budou náhodně objevovat nepřátelé. Při každém zabití nepřítele dojde ke zvýšení skóre. Po ztrátě všech životů bude zobrazeno dosažené skóre a počty zničených nepřátel.

5.1.3 Vývojové prostředí

Pro vývoj hry bude použito vývojové prostředí InteliJ IDEA Ultimate od společnosti JetBrains. Toto vývojové prostředí jsem si zvolil na základě dlouholetého používání jak pro školní projekty, tak pro pracovní projekty. Další výhodou tohoto vývojového prostředí je zabudovaný emulátor pro Android.

Obrázek 5.1: Ukázka hry Soul Knight (zdroj: [29])

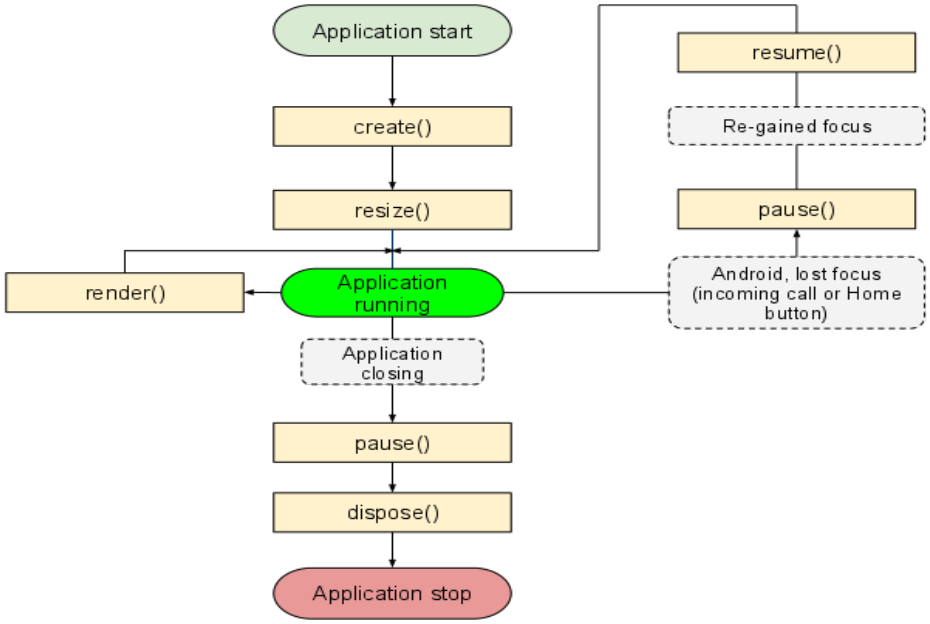
5.2 Návrh

V této části si rozebereme návrh celé hry (herní smyčku, scény, logiku hry atd.).

5.2.1 Životní cyklus hry/aplikace

Každá aplikace, vytvořená pomocí frameworku LibGDX, má životní cyklus, viz obr. 5.2.

- *create()* - Metoda volaná při vytvoření aplikace.
- *resize()* - Tato metoda se volá při každé změně velikosti herního okna (také je volaná hned po metodě. *create()*).
- *render()* - Metoda je volaná herní smyčkou aplikace při každém vykreslení snímku obrazovky.
- *pause()* - Obvykle se volá při stisku domovského tlačítka nebo při příchozím hovoru, nebo se používá pro uložení současného stavu hry.
- *resume()* - Metoda volaná pro obnovu stavu aplikace.
- *dispose()* - Metoda volaná při ukončení aplikace.



Obrázek 5.2: Životní cyklus aplikace (zdroj: [32])

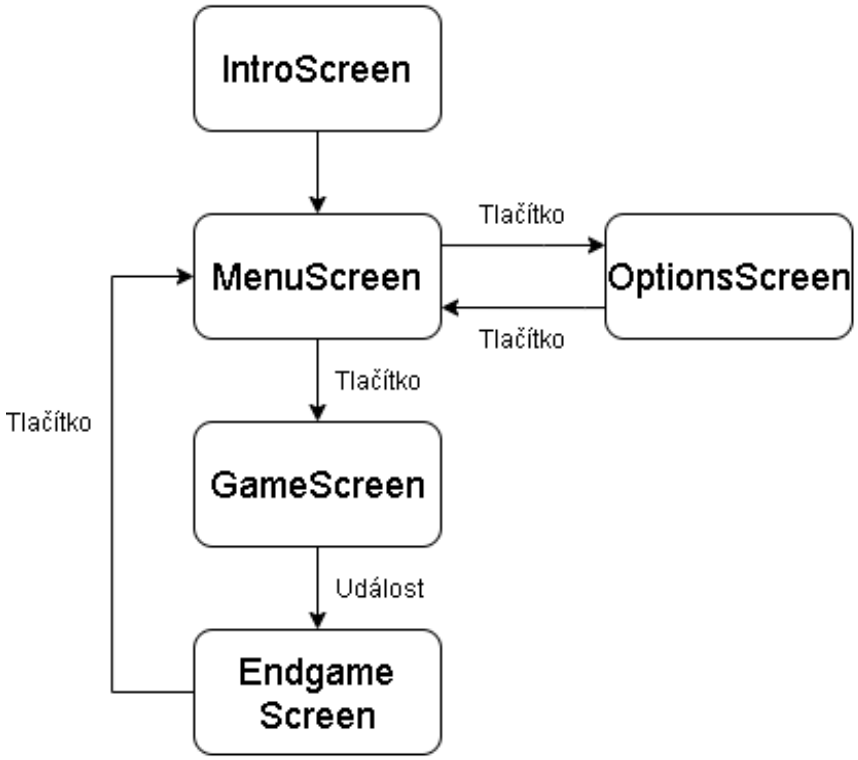
5.2.2 Scény

Každá scéna aplikace bude představovat jednu část hry. Pomocí tlačítek nebo událostí bude možné mezi jednotlivými scénami přecházet, viz obr. 5.3.

- IntroScreen
- MenuScreen
- OptionScreen
- GameScreen
- EndGameScreen

Při spuštění aplikace se jako první zobrazí scéna *IntroScreen*. V této scéně bude malá úvodní animace. Po ukončení animace se přepne obrazovka na scénu *MenuScreen*, reprezentující menu hry. Z této scény aplikaci ukončíte tlačítkem *Exit*, pomocí tlačítka *Options* lze přejít do nastavení hry, nebo pomocí tlačítka *Play* zobrazit scénu *GameScreen*. Ve scéně *OptionsScreen* se nastavují některé vlastnosti hry (hudba v pozadí hry, zvukové efekty postav, obtížnost) a zpátky do hlavního menu se lze vrátit pomocí tlačítka *Back*. Po stisku tlačítka *Play* v hlavním menu bude zobrazena *GameScreen*, která bude reprezentovat samotnou hru. Po ztrátě všech životů se obrazovka přepne do

Endgame Screen, kde se zobrazí dosažené skóre. Tlačítkem *Menu* se scéna přepne zpět do hlavního menu.

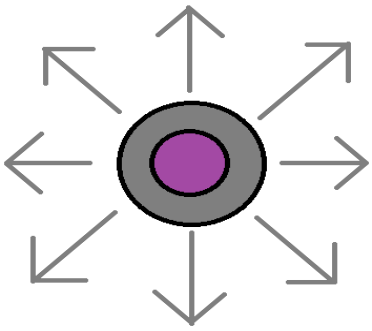


Obrázek 5.3: Diagram obrazovek hry

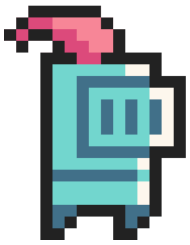
5.2.3 Hrdina

Hrdina je postava hráče. Pohyb postavy bude možný do osmi různých směrů a bude záviset na pozici touchpadu nebo náklonu zařízení, viz obr. 5.4.

Hrdina bude útočit pomocí meče. Útok provede po stisktutí tlačítka **Attack** a směr útoku bude záviset na posledním směru pohybu hrdiny. Pokud dojde ke kolizi meče a nepřítele, bude nepříteli odečten počet životů podle velikosti poškození hrdiny. Dokud nebude dokončena celá animace jednoho útoku, tak nebude možné uskutečnit další útok. Při zásahu hrdiny bude na malou chvíli imunní proti zásahu jakéhokoliv nepřítele. Textura postavy viz obr. 5.5.



Obrázek 5.4: Směry pohybu hrdiny



Obrázek 5.5: Textura hrdiny

5.2.4 Nepřátelé

Ve hře existuje více druhů nepřátel. Jednotliví nepřátelé se od sebe liší nejen vzhledem, ale i vlastnostmi postavy. Nekteří mají útok na blízko a jiní mají útok na dálku.

Imp

Je malý a velmi rychlý démon, který může způsobit malé poškození, zároveň bude mít malé množství životů. Může poškodit hrdinu, pokud se přiblíží na určitou vzdálenost. Textura postavy viz obr. 5.6.

Obrázek 5.6: Textura malého démona

Chort

Chort má podobné vlastnoti jako Imp, ale s tím rozdílem, že má větší počet zdraví a útoku, ale má pomalejší rychlost pohybu. Může poškodit hrdinu, pokud se přiblíží na určitou vzdálenost. Textura postavy viz obr. 5.7.

Obrázek 5.7: Textura většího démona

Necromancer

Mág s velkým počtem životů a magickým poškozením. Oproti Chortovi a Impovi se nebude pohybovat a ani nezpůsobí poškození hrdinovi, pokud se přiblíží na určitou vzdálenost. Jednou za daný čas se provede animace hůlky, čímž vykouzlí fireball, který se objeví nad současnou pozicí hrdiny a bude padat dolů. Pokud bude hrdina v blízkosti dopadu fireballu, způsobí mu tak poškození. Textura postavy viz obr. 5.8.

Obrázek 5.8: Textura mága

5.2.5 Přehled vlastností postav

Tabulka vlastností jednotlivých postav ve hře, viz tab. 5.1. Jednotlivé hodnoty představují jednotky dané vlastnosti, tzn.: pokud bude mít hrdina 6 jednotek zdraví a Imp mu způsobí útokem zblízka 1 jednotku poškození,

hrdinovi zbyde 5 jednotek zdraví. V tomto případě to tedy znamená, že Imp musí šestkrát zasáhnout útokem zblízka, aby zabil hrdinu. Pohyb značí počet pixelů za jeden snímek, tedy pokud hra má 30 snímků za sekundu a hrdina má rychlost pohybu 2, rychlost pohybu hrdiny na obrazovce bude 60 pixelů za sekundu. XP představuje skóre, které se připočte k současnému stavu skóre po zničení postavy.

Typ postavy	**Zdraví**	**Útok zblízka**	**Útok na dálku**	**Pohyb**	**XP**
Hrdina	6	2	0	2	-
Chort	3	2	0	1	3
Imp	1	1	0	2	1
Necromancer	5	0	3	0	4

Tabulka 5.1: Vlastnoti postav

5.2.6 Aréna

Herní plocha pro pohyb postav. Pro tvorbu arény bude použit některý z nástrojů, který *LibGDX* podporuje, jako je např. Hyperlap2D, Tiled. Tyto nástoje dovolují nahrát vytvořené části map, které jsou následně vloženy do čtvercového pole, a tím se vytváří herní svět. Pro tvorbu aplikace bude použit Tiled z důvodu jednoduchosti použití. Tiled je editor 2D úrovní pro tvorbu map. Pro tvorbu nové mapy stačí načíst sadu částí mapy, viz obr. 5.9, a vkládat je do čtvercového pole.

Obrázek 5.9: Dlaždicová sada

Ukázky jednotlivých scén hry viz kapitola A.5.

6 Programátorská dokumentace

Programátorská dokumentace popisuje projekt z pohledu programátora. V této kapitole jsou popsány použité třídy a jejich metody, struktura projektu, uživatelská oprávnění atd.

6.1 Struktura projektu

Struktura projektu je předem vygenerována, viz kapitola 4.2.2. Tato struktura je rozdělena do několika složek.

- *.gradle* - Složka se skripty, jak sestavit aplikaci
- *.idea* - Soubory pro nastavení vývojového prostředí projektu.
- *android* - Složka se soubory pro vývoj aplikace pro mobilní zařízení s operačním systémem Android (Manifest, assets atd.).
- *core* - Složka se zdrojovými kódy, které budou přeloženy pro všechny zvolené platformy.
- *desktop* - Složka s nastavením a spuštěním pro desktopové aplikace.
- *gradle* - Nástroj pro sestavení aplikace.

Zbylé soubory jsou konfigurační soubory pro gradle.

6.2 Třídy

Zde je popis jednotlivých tříd použitých v projektu. Ke každé třídě je krátký popis dané třídy a seznam nejdůležitějších metod s popisem.

6.2.1 Hra

Třídy popisující nejdůležitější části samotné hry, bude uživatel ovládat postavu v aréně a ničit nepřátele.

ArenaOfDeath

Třída reprezentující hlavní třídu celé aplikace. Jedná se o vstupní bod aplikace.

- *create* - Metoda volána při prvním spuštěním aplikace.
- *dispose* - Metoda volána při ukončení aplikace.
- *createAllScenes* - Metoda vytvářející všechny scény aplikace.
- *changeScreen* - Metoda volána při přepnutí jednotlivých scén.

Arena

Třída představující arénu hry, kde se bude pohybovat hrdina a nepřátelé. Jelikož vytvořená mapa má menší rozlišení než všechny mobilní zařízení, je použita ortgrafická kamera pro její vykreslení.

- *draw* - Metoda pro vykreslení mapy.

6.2.2 Postavy

Popis jednotlivých tříd postav ve hře:

Character

Rodičovská třída všech postav ve hře. Obsahuje základní vlastnosti postav (současné souřadnice, počet zdraví, velikost poškození, rychlost pohybu, aktuální činnost atd.). Podle současného směru pohledu je postava natočena na levou, nebo pravou stranu.

- *draw* - Metoda pro vykrelení postavy.
- *initDeathScene* - Inicializace animace úmrtí postavy.
- *distanceFromActor* - Metoda pro výpočet vzdálenosti od jiné postavy.

Chort

Třída reprezentující nepřítele (většího démona).

- *draw* - Metoda pro vykreslení, kontroluje, zda počet životů neklesl nebo se nerovná hodnotě nula, a mění souřadnice postavy podle aktuální pozice hrdiny.
- *initIdleAnimation* - Inicializace animace pro stání postavy.
- *initRunAnimation* - Inicializace animace pro běh postavy.
- *getDefaultSprite* - Metoda vracející výchozí obrázek postavy.

Hero

Třída hrdiny představuje postavu, kterou ovládá uživatel.

- *draw* - Metoda vykreslující postavu a aktualizaci parametrů zbraně a grafický ukazatel životů.
- *changePositions* - Metoda nastavující souřadnice postavy a směr pohledu, podle pozice touchpadu nebo směru náklonu zařízení.
- *calculateNewPositions* - Metoda pro výpočet nových souřadnic a kontrolu kolize s okraji arény.
- *initIdleAnimation* - Inicializace animace pro stání postavy.
- *initRunAnimation* - Inicializace animace pro běh postavy.
- *getDefaultSprite* - Metoda vracející výchozí obrázek postavy.

Imp

Třída představující malého démona.

- *draw* - Metoda pro vykreslení, kontroluje, zda počet životů neklesl nebo se nerovná hodnotě nula a mění souřadnice postavy podle aktuální pozice hrdiny.
- *initIdleAnimation* - Inicializace animace pro stání postavy.
- *initRunAnimation* - Inicializace animace pro běh postavy.
- *getDefaultSprite* - Metoda vracející výchozí obrázek postavy.

Necromancer

Třída pro nepřátelského mága. Nepohybuje se jako ostatní postavy. Také nemá útok zblízka jako ostatní postavy, ale jednou za určitý interval proběhne animace hůlky. Po ukončení animace hůlky se vykouzlí *Fireball*.

- *draw* - Metoda pro vykreslení a kontrolu počtu životů a hůlky při útoku.
- *initIdleAnimation* - Inicializace animace pro stání postavy.
- *initRunAnimation* - Inicializace animace pro běh postavy.
- *getDefaultSprite* - Metoda vracející výchozí obrázek postavy.

Spawner

Třída vytvářející nové nepřátele. Zjistí náhodnou volnou pozici pro vytvoření nového nepřítele po uplynutí časového intervalu. Počet vytvořených nepřátel závisí na obtížnosti hry.

- *getRandomSpawnPoint* - Metoda pro zjištění volných souřadnic.
- *spawnChort* - Metoda pro vytvoření většího démona.
- *spawnImp* - Metoda pro vytvoření malého démona.
- *spawnNecromancer* - Metoda pro vytvoření mága.
- *removeDeathCreatures* - Metoda pro odstranění nepřátel s počtem životů menších nebo rovno nula, zvýšením celkového skóre a zvýšením počtu zničených nepřátel pro statistiky na konci hry.
- *restartKillsRecords* - Metoda pro vynulování hodnot zničených nepřátel.

6.2.3 Scény

Popis komponent jednotlivých scén ve hře.

EndGameScreen

Třída pro scénu na konci hry po ztrátě všech životů hrdiny. V této scéně se zobrazí statistiky počtu zničených nepřátel a současné dosažené skóre a nejvyšší dosažené skóre.

- *render* - Metoda pro vykreslení scény.
- *setTable* - Metoda pro nastavení komponent ve scéně (labelů, tlačítek, obrázků).
- *setFontSizes* - Nastaví velikosti fontů.
- *setListeners* - Nastaví události po stisknutí tlačítka.
- *setLabels* - Nastaví fonty pro text ve scéně.
- *removeRecords* - Odstraní statistiky o počtech zabitých nepřátel ze souboru *Shared Preferences* z poslední hry.

GameScreen

Třída pro zobrazení herní scény s arénou, hrdinou, nepřáteli, ovládacími prvky a ukazateli zdraví. Jednou za časový interval se vytvoří noví nepřátelé (záleží na obtížnosti) na náhodné pozici.

- *render* - Metoda pro vykreslení scény, kontroluje zda hrdinovi neklesl na hodnotu nula. Po ztrátě všech životů hrdiny se scéna přepne do **EndGameScreen**. Dále kontroluje dotyk touchpadu nebo náklon zařízení a stisk tlačítka **Attack** pro útok.
- *resize* - Metoda pro úpravu pohledu kamery při změně velikosti rozlišení.
- *pause* - Metoda volána při přesunutí aplikace na pozadí. Ukládá současná data do souboru *Shared Preferences*.
- *setStage* - Metoda pro nastavení komponent scény.
- *updateScoreLabel* - Metoda pro nastavení textu ukazatele současného skóre.
- *setScoreSharedPreferences* - Metoda pro uložení statistik a dosaženého skóre do souboru **Shared Preferences**.
- *restartGame* - Metoda volána při spuštění nové hry. Vymaže všechny existující nepřátele, firebally a krev ze scény.
- *spawnNewCreature* - Metoda pro vytvoření nepřítele, pokud uplynul určitý časový interval. Nového nepřítele přidá do seznamu všech existujících nepřátel.
- *setLabels* - Nastaví fonty pro text ve scéně.

IntroScreen

Třída pro scénu, která se zobrazí jako první po spuštění aplikace. V této scéně proběhne krátká animace a následně se scéna přepne do menu.

- *render* - Metoda pro vykreslení.

MenuScreen

Třída pro scénu přestavující menu. Tlačítka ve spodní části obrazovky slouží pro přepnutí scény. Pomocí tlačítka **Play** se scéna přepne do scény s arénou. Tlačítko **Options** přepne scénu do nastavení a tlačítko **Exit** ukončí aplikaci.

- *render* - Metoda pro vykreslení scény.
- *setTable* - Metoda pro nastavení komponent ve scéně.
- *setLabelFonts* - Nastaví velikosti fontů pro text.
- *setListeners* - Nastaví události po stisknutí tlačítka.
- *setLabels* - Nastaví fonty pro text ve scéně.

OptionScreen

Třída pro scénu s nastavením hry. Kromě možného nastavení hry je možné vidět i nejvyšší dosažené skóre pro jednotlivé obtížnosti.

- *render* - Metoda pro vykreslení scény.
- *setTable* - Metoda pro nastavení komponent ve scéně.
- *setFontSizes* - Nastaví velikosti fontů.
- *setListeners* - Nastaví události po stisku tlačítka.
- *setLabels* - Nastaví fonty pro text ve scéně.

6.2.4 Efekty, hudba, ukazatel životů

Audio

Třída pro hudbu hry. Nastavuje typ hudby při přechodech mezi některými scénami. Obsahuje informace, zda jsou zvukové efekty a hudba ve hře povoleny. Podle volby uživatele vypíná a zapíná hudbu.

- *disableMusic* - Metoda volána při zákazu hudby a zvukových efektů ve scéně *OptionScreen*.
- *enableMusic* - Metoda volána při povolení hudby a zvukových efektů ve scéně *OptionScreen*.
- *setGameAudio* - Metoda volána při přechodu mezi jednotlivými scénami, podle které nastaví hudbu (hudba se nezmění při přechodu z nastavení do menu).

HeroDamaged

Třída pro zvukový efekt, pokud bude hrdina zasažen nepřítelem.

- *runSound* - Metoda pro spuštění zvukového efektu.

SwordSound

Třída pro zvukový efekt meče při útoku hrdiny po stisknutí tlačítka **Attack**.

- *runSound* - Metoda pro spuštění zvukového efektu.

Blood

Třída pro animaci krve při zasažení hrdiny.

- *draw* - Metoda pro vykreslení animace.
- *initBloodAnimation* - Metoda pro inicializaci animace krve při zasažení hrdiny.

Life

Třída reprezentující zdraví hrdiny. Zdraví je graficky znázorněno v levém horním rohu obrazovky v aréně. Toto zdraví se aktualizuje při zasažení hrdiny nepřítelem.

- *draw* - Metoda pro vykreslení grafického ukazatele životů podle současného počtu zdraví hrdiny.

6.2.5 Zbraně

Weapon

Třída reprezentující předka pro zbraně.

- *update* - Metoda pro aktualizaci souřadnic a směru útoku.
- *draw* - Metoda vykreslující zbraň.

Staff

Třída pro zbraň nepřítele **Necromancera**. Textura může být zrcadlově přetočena podle aktuálního pohledu postavy.

- *draw* - Metoda pro vykreslení hůlky.
- *initStaffTexture* - Inicializace textury hůlky.

Fireball

Třída reprezentující magický útok postavy **Necromancer**. Fireball se vytvoří nad souřadnicemi hrdiny po dokončení animace hůlky. Fireball se přibližuje k souřadnicím, kde stál hrdina při vytváření Fireballu. Pokud Fireball dopadne na zem a zároveň bude v jeho blízkosti hrdina, způsobí mu to poškození.

- *draw* - Metoda pro vykreslení Fireballu, který se při každém vykreslení posune blíže k místě dopadu.
- *distanceFromHero* - Metoda pro výpočet vzdálenosti od hrdiny.
- *initFireballAnimation* - Incializace animace Fireballu.

Sword

Třída pro zbraň hrdiny. Textura může být zrcadlově přetočena podle aktuálního pohledu hrdiny. Animace bude spuštena po stisknutí tlačítka **Attack**. Pokud bylo tlačítko stisknuto a animace probíhá, není možné spustit animaci útoku znovu, dokud první animace neskončí.

- *update* - Metoda pro aktualizaci souřadnic a směru útoku.
- *draw* - Metoda pro vykreslení meče podle směru pohledu hrdiny.
- *checkHitEnemy* - Metoda pro kontrolu kolize meče a nepřátel.

- *isInCollision* - Metoda pro zjištění, zda je meč v kolizi s postavou.
- *distanceBetweenEnemyAndSword* - Metoda pro výpočet vzdálenosti meče a postavy.

6.2.6 Ostatní

enums

Balíček obsahující výčtové typy použité v aplikaci.

- *CharacterAction* - činnost postavy
- *Difficulty* - obtížnost hry
- *LookDirection* - směr pohledu postavy
- *MeleeDirection* - směr útoku zblízka
- *Screens* - obrazovky aplikace

Time

Třída reprezentující čas, který je spuštěn po stisknutí tlačítka **Play** v hlavním menu. Čas je v aplikaci použit pro zjištění, zda se má vytvořit nový nepřítel.

- *spawnNextCreature* - Metoda zjistí, zda uběhl časový interval pro vytvoření nového nepřítele.

Constants

Třída konstant použitých v aplikaci.

LabelFont

Třída pro nastavení fontu pro text.

- *getLabelStyle* - Vrací styl fontu pro text.

Point

Třída reprezentující bod na obrazovce.

6.3 Ovládání postavy

Ovládání postavy je možné pomocí touchpadu, nebo náklonem mobilního zařízení. Přepínání těchto ovládání lze změnit v nastavení. Hodnoty současné pozice touchpadu nebo náklon zařízení se získávají ve třídě **GameScreen** a směr pohybu postavy se získá ve třídě **Hero**.

6.3.1 Touchpad

Směr pohybu postavy spočívá ve výpočtu úhlu současné pozice touchpadu. tento úhel se pomocí matematických funkcí (sinus, cosinus) převede do kartézské soustavy souřadnic, ze kterého získáme směr pohybu postavy.

6.3.2 Náklon zařízení

Pro pohyb postavy pomocí náklonu mobilního zařízení se bude využívat akceleromer. Pro získání směru hrdiny musí být překročena určitá hodnota náklonu zařízení, která bude získána z akcelerometru a hrdina se bude pohybovat tímto směrem.

6.4 Obtížnost

Obtížnost ve hře definuje počet nepřátel, který se vytváří jednou za časový interval. Čím vyšší stupeň obtížnosti, tím více nepřátelských jednotek se bude vytvářet. Vlastnosti nepřátel zůstávají pořád stejné pro jednotlivé druhy postav. Změnu obtížnosti lze změnit pomocí tlačítka v nastavení hry. Zde je přehled obtížností hry a k nim počet nepřátel vytvořených za časový interval:

- *EASY* - 1 nepřítel
- *NORMAL* - 2 nepřátelé
- *HARD* - 3 nepřátelé

6.5 Uživatelská oprávnění

Při poškození postavy hrdiny mobilní zařízení na krátkou chvíli zavibruje jako upozornění uživatele, že hrdina byl zasažen. Pro tuto notifikaci je nutné přidat do souboru *AndroidManifest.xml* oprávění:

<uses-permission android:name="android.permission.VIBRATE"/>

6.6 Textury

Ve hře byly použity dvě sady textur. Jedna sada byla použita pro textury postav, arény, ukazatel, životů a zbraní. Druhá sada byla použita na vzhled tlačítek a labelů. Textury byly staženy zdarma z následujících webů.
`https://itch.io/game-assets/free/tag-pixel-art`
`https://github.com/czyzby/gdx-skins`

6.7 Ortografická kamera

Ortografická kamera se používá pro 2D projekci. Jelikož velikost arény je mnohem menší, než je rozlišení obrazovky telefoního zařízení, použijeme ortografickou kameru pro bližší zobrazení.

6.8 Hudba

Ve hře bude hrát hudba v pozadí hry a některé herní postavy budou mít zvukové efekty. Pokud se budete nacházet v menu, hudba bude vždy stejná. Pokud spustíte novou hru pomocí tlačítka *Play*, spustí se jedna ze tří melodií (výběr je náhodný). Hrdina bude mít své zvukové efekty při útoku a při jeho poškození. Hudba a zvukové efekty byly zdarma staženy z následujících odkazů.
`https://freemusicarchive.org/music/sawsquarenoise`
`https://freesound.org/people/SexyNakedBunny/sounds/274717/`
`https://www.soundfishing.eu/sound/sword`

6.9 Shared Preferences

Jelikož v aplikaci není potřeba ukládat velké množství dat, budou některé hodnoty uloženy do *Shared Preferences* souboru jménem *app_preferences*. Do souboru budou v průběhu aplikace ukládány hodnoty:

- *high_score_easy* - Nejvyšší dosažené skóre pro lehkou obtížnost.
- *high_score_normal* - Nejvyšší dosažené skóre pro normální obtížnost.
- *high_score_hard* - Nejvyšší dosažené skóre pro težkou obtížnost.
- *current_score* - Skóre dosažené v aktuální hře.

- *chort_kills* - Počet zničených větších démonů.
- *imp_kills* - Počet zničených malých démonů.
- *necromancer_kills* - Počet zničených mágů.

Všechny hodnoty, kromě nejvyššího skóre všech obtížností, jsou jen dočasné. Při opětovném spuštění scény s Arenou budou vymazány, aby na konci hry mohla být použita aktuální data.

6.10 Přínosy frameworku

Použití herního frameworku při vývoji herní aplikace, bylo velmi prospěšné. Pokud by se při vývoji nepoužil, musel bych sám implementovat herní smyčku, vykreslení pro *Sprite*, získání parametrů z mobilu (rozlišení, hodnoty akcelerometru), hudbu atd. Při použití frameworku jsou tyto problémy už vyřešené a urychlují tak proces vývoje. V této práci byly použity jen některé z výhod daného frameworku, ale framework jich obsahuje mnohem více.

7 Testování

Testování bude provedeno několika uživateli na různých mobilních zařízeních s operačním systémem Android, viz tab. 7.1 . Každé mobilní zařízení bude mít odlišné parametry. Na konci kapitoly bude shrnutí celého testování na zařízeních.

Zařízení	**Rozlišení**	**Verze systému**	**CPU [GHz] (jádra)**	**RAM [GB]**
Huawei P30 Lite	2312 x 1080 (6,15")	10	2.2 (8)	4
Samsung Galaxy A51	1080 x 2400 (6,5")	11	2.3 (4)	6
Honor 9 Lite	2160 x 1080 (5,65")	9	2.36 (4)	3
Honor 20 Lite	1080 x 2246 (5,84")	9	2.2 (4)	4
Xiaomi Pocophone F1	2246 x 1080 (6,18")	10	2.8 (8)	6

Tabulka 7.1: Přehled mobilních zařízení pro testování

7.1 Popis

Aplikace bude testována na mobilních zařízeních s různým rozlišením obrazovek. Na každém zařízení se bude testovat podle následujících scénářů. Testování se bude snažit odhalit nestandardní chování aplikace.

7.1.1 Zobrazení a přechod scén

Na zařízeních se testovalo správné zobrazení komponent (tlačítka, labely, arény, postavy atd.) jednotlivých scén, zda se nevykreslují mimo viditelnost obrazovky a text tlačítek nepřesahuje velikost tlačítka, ve kterém se nachází. Dále se otestuje, zda se správně přechází do dalších scén, viz obr. 5.3. Vykreslování jednotlivých scén bude přizpůsobeno změně orientace obrazovky mobilního zařízení.

7.1.2 Hra

Testování samotné hry v aréně se skládá z několika částí.

Zobrazení zdraví a animací

V této části se otestuje správné vykreslování animací jednotlivých postav při stání, pohybu a umírání. Testovat se bude správné vykreslení zbraní a při zasažení hrdiny animace zranění. Dále se bude testovat správné zobrazení počtu životů na začátku každé nové hry a při každém zasažení hrdiny.

Kolize

Kolize meče a nepřítele bude mít za následek snížení životů nepřítele. Při kolizi nepřítele s hrdinou na krátkou vzdálenost dojde ke snížení životů hrdiny a dojde ke krátké vibraci mobilního zařízení.

Skóre

Testovat se bude celkové skóre hry po ztrátě všech životů hrdiny. Skóre bude záležet na počtu zničených nepřátelských jednotek. Za každou zničenou jednotku se k současnému skóre připočte určitý počet jednotek skóre, viz tab. 5.1. Dále se bude testovat zobrazení správného počtu jednotlivých druhů nepřátel. Nejvyšší skóre z každé obtížnosti se bude ukládat do souboru *Shared Preferences*. Skóre z každé obtížnosti se bude zobrazovat ve scéně s nastavením hry. Otestuje se, zda budou hodnoty uchovány i po ukončení aplikace a vypnutí či restartu mobilního zařízení.

Nastavení hry

V nastavení lze nastavit hudba ve hře. Pokud bude hudba vypnutá, nebude v žádné z jednotlivých scén aplikace spuštěna hudba či zvukové efekty. Dále lze změnit ovládání hrdiny pomocí touchpadu a náklonem zařízení. Pokud bude nastaveno ovládání pomocí náklonu telefonu, ve hře nebude zobrazen touchpad v levé dolní části obrazovky. A při ovládání hrdiny pomocí touchpadu nebude hrdina reagovat na náklon zařízení. Posledním otestováním je možnost nastavení obtížnosti hry. Změna obtížnosti hry se projeví v počtu vytvoření nepřátelských postav za časový interval, viz kapitola 6.4.

7.1.3 Nestandardní situace

Za nestandardní situace se považují situace nezávislé na aplikaci, např. příchozí hovor nebo notifikace jiné aplikace (SMS, Instagram, Facebook,

Livesport a další). Notifikace nebude mít vliv na průběh aplikace. Pokud uživatel přijme příchozí hovor bude hra pozastavena a po ukončení hovoru a návratu do aplikace bude hra pokračovat. Stejně bude hra pokračovat po stisku domovského tlačítka a následného návratu do aplikace. Pokud uživatel zobrazí navigační panel v dolní části mobilního zařízení, bude tento navigační panel po chvíli opět skryt.

7.2 Vyhodnocení výsledků testů

Při testování aplikace byla nalezena a následně opravena chyba pro pohyb postavy na okraji arény. Postava se zasekávala, když se pohybovala při okraji. Další problém byl zobrazování některých labelů a tlačítek, která se zobrazovala mimo obrazovku.

Jiné další problémy, týkající se aplikace, nebyly nalezely v průběhu testování. Přepínaní jednotlivých scén probíhalo bez problémů. Scény a jejich komponenty byly správně vykreslovány. Jednotlivé animace postav, zbraní a efektů probíhaly dle očekávání. Z pohledu výkonu nebyl nalezen žádný problém na testovacích zařízeních.

7.3 Možnosti rozšíření

Hra je vytvořena tak, aby bylo jednoduché implementovat nové prvky hry. Zde je pár možných tipů pro rozšíření:

- Více různých úrovní (různé tvary arény, překážky atd.).
- Nové nepřátele s jinými typy vlastností a útoků.
- Přidání ukazatelů životů nepřátel.
- Výběr z více různých postav s různými vlastnostmi (rychlost pohybu, typ útoku, počet životů).
- Možnost získání jiné zbraně od nepřítele po jeho zničení (každá zbraň bude mít jiné vlastnosti, např. velikost poškození, rychlost útoku atd.).
- Po určitém množství zničených nepřátel se obnoví určitý počet životů.

8 Závěr

Cílem této bakalářské práce bylo analyzovat vybrané herní frameworky využitelné pro platformu Android a vytvořit hru na mobilní zařízení s operačním systémem Android, který bude využívat výhod daného herního frameworku.

Nejprve byla provedena analýza jednotlivých herních frameworků a porovnání jejich vlastností z různých hledisek (např. typ her, funkce, nástroje atd.). Další část byla věnována návrhu samotné hry pomocí herního frameworku LibGDX. Aplikace se skládá z několika scén, mezi kterými uživatel přepíná pomocí tlačítek, nebo pokud nastane určitá událost, změní se současná scéna. Grafika hry je podobná grafice starých retro her. Tento typ je jednoduchý na implementaci a jedná se o můj oblíbený typ grafiky.

V samotné hře uživatel ovládá postavu pomocí touchpadu nebo náklonem zařízení. Cílem je získání nejvyššího skóre, které se zvyšuje po zničení nepřítele. Nepřátel je více druhů s odlišnými vlastnostmi. V nastavení hry lze nastavit některé vlastnosti hry (ovládání postavy, obtížnost hry, zvuk a zvukové efekty).

V následující části byly popsány použité třídy, implementace nejdůležitějších metod, uživatelská oprávnění, textury, hudba a ukládání dat.

Poslední část práce se věnovala testování aplikace na mobilních zařízeních s odlišnými parametry. Hru testovalo více uživatelů podle popsaných scénářů. V práci jsou navržena další vhodná rozšíření stávající hry.

Aplikace může sloužit jako vzor pro tvorbu podobných 2D her pomocí herního frameworku LibGDX.

Bibliografie

[1] Ali Abdul-Karim a Linda Karlsson. "Game Development Project with Cocos2D-X". In: (2014). URL: https://www.theseus.fi/bitstream/handle/10024/74656/ali_thesis_final2.pdf?sequence=1.

[2] *Android Operating System*. 2021. URL: https://www.investopedia.com/terms/a/android-operating-system.asp.

[3] *Android Overview*. 2021. URL: https://www.tutorialspoint.com/android/android_overview.htm.

[4] Ismail Buyuksalih et al. "3D Modelling and Visualization Based on the Unity Game Engine–Advantages and Challenges". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2017), s. 161.

[5] Edgar Cervantes. *Angry Birds 2 review: how does it compare to the original?* 2015. URL: https://www.androidauthority.com/angry-birds-2-review-631029/.

[6] Jonathan Chen. *Unity Tutorial*. 2018. URL: https://chenjonathan.github.io/.

[7] Michael Chen. *Solar2D*. URL: https://michaelchen.tech/solar2d/helloworld/.

[8] CoolAI. *Grue the monster screenshot*. 2018. URL: https://coolai.itch.io/grue-roguelike.

[9] *Corona SDK*. 2020. URL: https://www.filehorse.com/download-corona-sdk/screenshots/.

[10] *Corona: The 2D Game Engine*. URL: https://coronalabs.com/.

[11] *Delver Promo Art*. 2017. URL: https://www.mobygames.com/game/delver/promo/imageType,1/promoImageId,300241/.

[12] *Framework*. Software Terms: Framework Definition. 2013. URL: https://techterms.com/definition/framework.

[13] *Full toolset for 2D and 3D video games*. URL: https://unity.com/how-to/difference-between-2D-and-3D-games.

[14] Reza Ghobadinic. *Installing Cocos2d-x (Windows)*. 2018. URL: https://rezghob.com/installing-cocos2d-x-windows/.

[15] *Introduction to LibGDX*. LibGDX definition. 2013. URL: https://libgdx.badlogicgames.com/documentation/.

[16] Tereza Kratochvílová. *Hill Climp Racing*. 2014. URL: https://wmmania.cz/clanek/hill-climb-racing/.

[17] Loran$_C$*leric. You may not like it but this is what peak hearthstone gameplay looks like*. 2018. URL: https://www.reddit.com/r/hearthstone/comments/7qhf03/you_may_not_like_it_but_this_is_what_peak/.

[18] *Merchant screenshot*. URL: https://retora-games.itch.io/merchant.

[19] Suryakumar Balakrishnan Nair a Andreas Oehlke. *Learning LibGDX Game Development*. Packt Publishing Ltd, 2015.

[20] *Nubs' Adventure – Exploratory Platformer*. 2013. URL: http://www.imake-games.com/nubs-adventure/.

[21] *Plans and pricing*. 2021. URL: https://store.unity.com/#plans-individual.

[22] Phillip Prado. *Call of Duty: Mobile: Everything you need to know*. 2019. URL: https://www.androidauthority.com/call-of-duty-mobile-966556/.

[23] *Project Templates*. 2021. URL: https://docs.unity3d.com/Manual/ProjectTemplates.html.

[24] *Project Templates*. 2021. URL: https://docs.unity3d.com/Manual/index.html.

[25] Jose Carlos Recuero. *Cocos Flagship Products: Cocos Creator, Cocos2d-x*. URL: https://cz.pinterest.com/pin/551057704396824869/.

[26] Sai Sai. *Designer City screenshot*. 2020. URL: https://apkpure.com/designer-city-empire-edition/com.spheregamestudios.designercity.empireedition.

[27] Walzer Jare SlackMoehrle Minggo. *Cocos2d-x Docs*. Cocos2d-x documentation. 2019. URL: https://docs.cocos.com/cocos2d-x/manual/en/.

[28] *Solar2D Docs*. Solar2D documentation. URL: https://docs.coronalabs.com/guide/index.html#developer-guides.

[29] *Soul Knight*. URL: https://play.google.com/store/apps/details?id=com.ChillyRoom.DungeonShooter&hl=cs&gl=US.

[30] J.F. DiMarzio Ted Hagos Mario Zechner a Robert Green. *Beginning Android Games Developer*. fourth. Apress, 2020.

[31] Wikipedia. *Ingress (video game)*. 2013. URL: https://en.wikipedia.org/wiki/Ingress_(video_game).

[32] Mario Zechner. *The Life-Cycle*. 2013. URL: https://github.com/libgdx/libgdx/wiki.

Seznam použitých zkratek

2D – dvoudimenzionální/dvourozměrný svět popsaný dvěma rozměry, např. šířkou a výškou.
3D – trojdimenzionální/trojrozměrný svět popsaný třemi směry, viz kartézká soustava souřadnic.
API (Application Programming Interface) – rozhraní pro programování.
Blob shadow – je jednoduchý koncept, kdy objekt vrhá stín pod sebe a tento stín připomínal jen rozmazaný ovál.
Dungeon – prostředí dané hry představuje podzemní katakomby.
Emulátor – je druh softwaru umožňující běh počítačovému programu na jiné platformě, než pro kterou byl původně vytvořen.
Fireball – ohnivé kouzlo, které při nárazu exploduje.
Grafický řetězec – sekvence procesů, jejich aplikací na data, která popisují scénu, získáme dvourozměrný obraz této scény.
JAR – Java Archive je formát, používaný pro seskupení více souborů.
Knihovna – soubor funkcí a tříd pro usnadnění tvorby aplikací.
Main – jedná se o funkci, která je při spuštění programu volána jako první. Jedná se o vstupní bod aplikace. Ve zdrojových souborech se musí vyskytovat právě jednou.
Mesh – síť uspořádaných trojúhelníků do prostoru tak, aby tvořily dojem pevného objektu. Tedy jedná se o tvar 3D objektu, který ho definuje.
Multiplayer – jedná se o termín převzatý z angličtiny, který označuje hru pro více hráčů.
Open source – počítačový software s otevřeným zdrojovým kódem.
Pathfinding – metoda hledání nejkratší cesty z jednoho bodu do bodu druhého.
Plugin – zásuvný modul pro aplikace rozšiřující její funkčnost.
Post-processing – proces aplikace filtrů nebo efektů do bufferu, předtím než se zobrazí na obrazovku.
Raycasting – proces, který pomáhá při identifikaci řešení problémů spojených s grafikou
Roguelike – žánr her, který se vyznačuje procházením náhodně generovanými dungeony, procedurálně generované úrovně atd.
RPG – žánr hry, kdy jednotliví hráči zaujímají role fiktivních postav.
Sandbox – bezpečnostní mechanismus pro omezený přístup ke zdrojům počítače.
SDK – software development kit je sada vývojových nástrojů, které umožňují tvorbu aplikací

Shader – počítačový program pro řízení jednotlivých částí programovatelného grafického řetězce grafické karty.
Shared Preferences - soubor obsahující hodnoty typu klíč-hodnota. Používá pro uložení hodnot aplikace.
Third-party – v počítačovém programování se jedná o softwarovou komponentu, která je volně distribuována nebo prodávána. Zvyšuje efektivitu a kvalitu vývoje aplikací.
Widget – je jednoduchá softwarová aplikace.

Seznam příloh

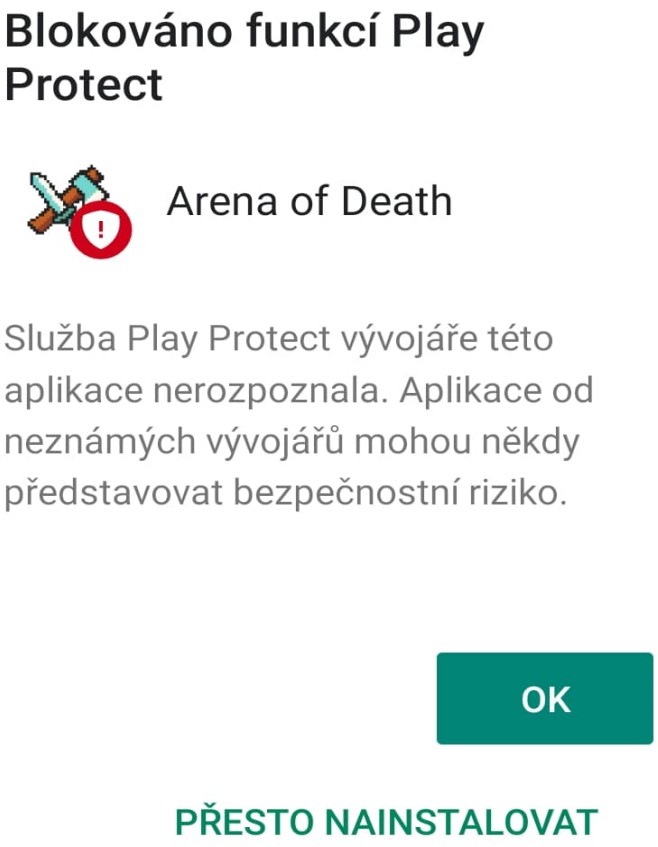
- Příloha A: Uživatelská dokumentace
- Příloha B: Obsah CD

A Uživatelská příručka

Tato uživatelská příručka poskytuje základní informace o aplikaci a o instalaci aplikace.

A.1 Instalace

Pro instalaci aplikace na mobilní zařízení musí obsahovat operační systém minimálně verzi 4.4 (KitKat). Při instalaci aplikace je nutné povolit v nastavení instalaci aplikací z neznámých zdrojů, nebo při instalaci dojde k blokování pomocí funkce **Play Protect** a zvolí se možnost **Přesto nainstalovat**, viz obr. A.1.



Obrázek A.1: Varování při instalaci aplikace

A.2 Ovládání hry

Celá aplikace se ovládá pomocí tlačítek a pohyb postavy pomocí touchpadu nebo náklonem telefoního zařízení (lze změnit v nastavení hry). Po animaci intra, viz obr. A.2, se zobrazí menu, viz obr. A.3. Z menu můžete přejít do nastavení pomocí tlačítka **Options**, tlačítkem **Play** přejít do hry a tlačítkem **Exit** ukončit aplikaci. Ve hře, viz A.5, se postava ovládá pomocí touchpadu, viz obr. A.5, nebo nakláněním mobilu, viz obr. A.8, a útok pomocí tlačítka **Attack**. Útok hrdiny závisí na posledním směru pohybu hrdiny (dolů, nahoru, doleva, doprava) a po stisknutím tlačítka pro útok hrdina zaútočí tímto směrem. Další ukázky hry viz obr. A.6 a obr. A.7. Na konci hry, viz obr. A.9, se lze dostat zpět do menu pomocí tlačítka **Back**.

A.3 Nastavení

V nastavení, viz obr. A.4, je vidět nejvyšší dosažené skóre pro jednotlivé obtížnosti hry. Dále je zde možné vypnout nebo zapnout hudbu ve hře. Lze nastavit i možnost, jak bude uživatel ovládat hrdinu ve hře (pomocí touchpadu nebo náklonem mobilního zařízení).

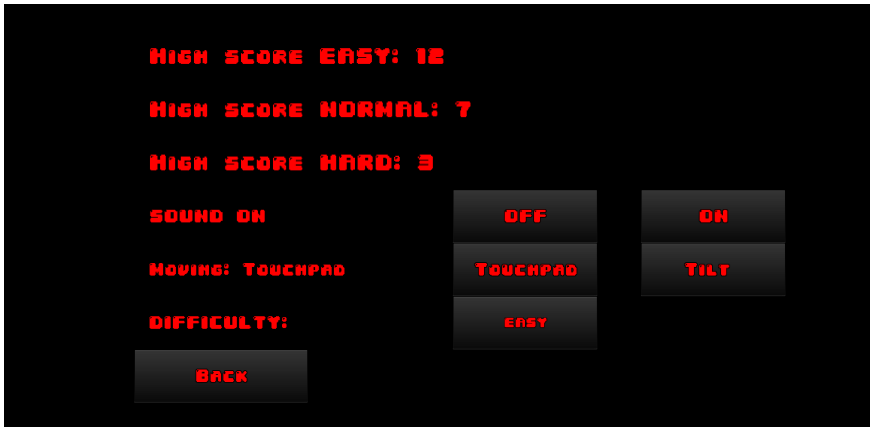
A.4 Skóre

Skóre se zvyšuje po každém zničení nepřítele. Po ztrátě všech životů hrdiny se zobrazí skóre dosažené v této hře a nejvyšší dosažené skóre. Skóre je ukládáno do souboru *Shared Preferences*, a pokud bude současné skóre vyšší než nejvyšší dosažené skóre v dané obtížnosti, bude v souboru nejvyšší skóre přepsáno. Dále v této scéně jsou zobrazeny statistiky počtu zničených jednotlivých druhů nepřátel.

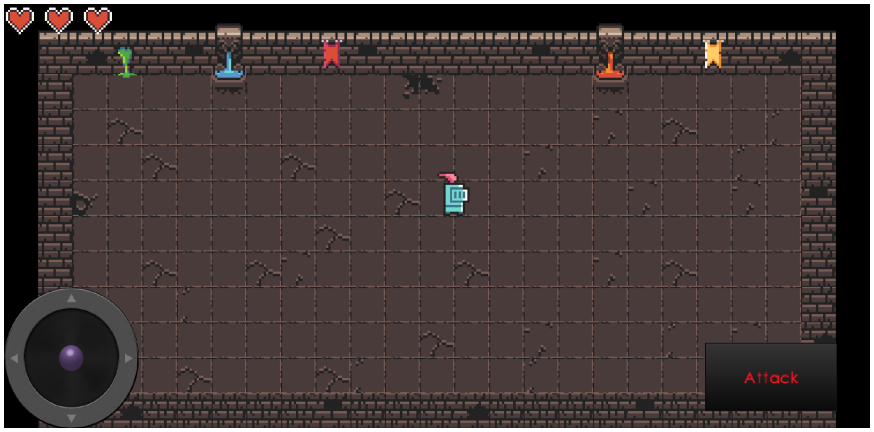
A.5 Obrázky ze hry

Obrázek A.2: Obrázek z intra

Obrázek A.3: Obrázek z menu

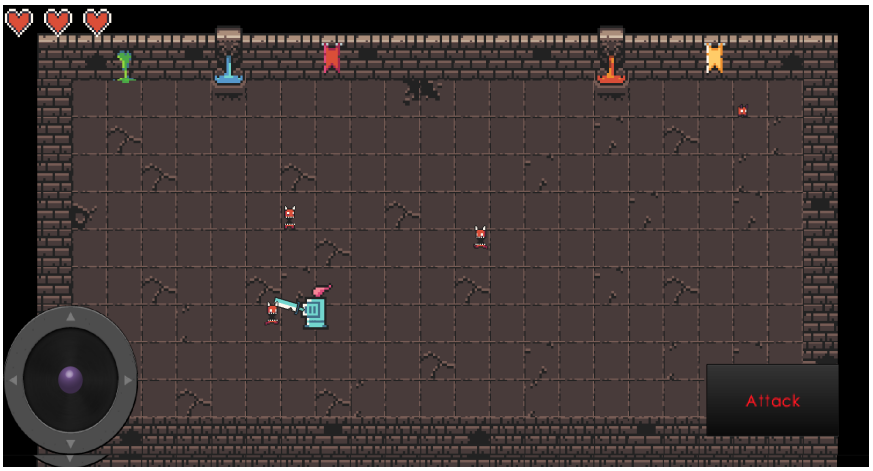


Obrázek A.4: Obrázek z nastavení



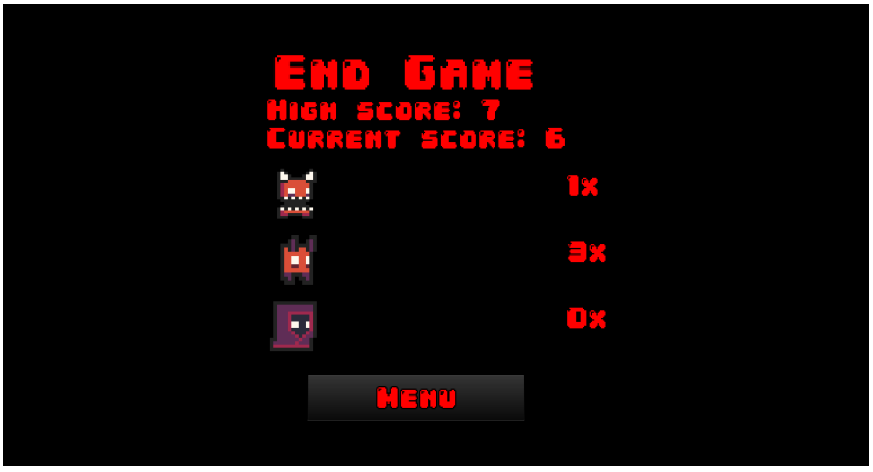
Obrázek A.5: Obrázek ze hry I.

Obrázek A.6: Obrázek ze hry II.



Obrázek A.7: Obrázek ze hry III.

Obrázek A.8: Obrázek ze hry IV.

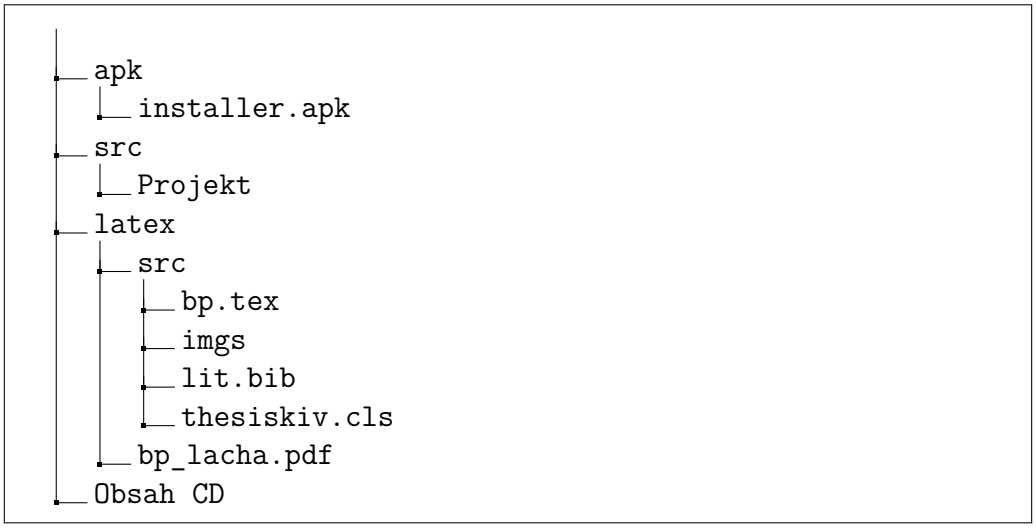


Obrázek A.9: Obrázek z konce hry.

B Obsah CD

Přiložené CD má strukturu - viz obr. B.1.

- apk - adresář s instalačním souborem aplikace
  - installer.apk - instalační soubor aplikace
- src
  - Projekt - adresář se zdrojovými soubory pro InteliJ IDEA Ultimate
- latex - adresář se zdrojovými soubory pro LaTeX
  - src
    - bp.tex - LaTeXsoubor
    - imgs - adresář s obrázky použitými v textu
    - lit.bib - literatura knih použitá v textu
    - thesiskiv.cls - konfigurační soubor
  - bp_lacha.pdf
- Obsah_CD.txt



Obrázek B.1: Struktura CD