České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství

Vývoj her pro mobilní telefony

Development of Mobile Phone Games

BAKALÁŘSKÁ PRÁCE

Vypracoval: Pavel Urx
Vedoucí práce: Ing. Michal Moc
Rok: 2022

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství Akademický rok 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Pavel Urx**
Studijní program:	**Aplikace přírodních věd**
Obor:	**Aplikace softwarového inženýrství**
Název práce česky:	**Vývoj her pro mobilní telefony**
Název práce anglicky:	**Development of Mobile Phone Games**

Pokyny pro vypracování:

1. Seznamte se s fázemi vývoje hry.
2. Získané poznatky popište.
3. Navrhněte jednoduchou hru pro mobilní telefon, ve které získané poznatky využijete.
4. Popište možnosti vydání hry.
5. Popište možnosti marketingu a propagace hry.
6. Promyslete a popište možnosti komercializace hry.

Doporučená literatura:

[1] BUTTFIELD-ADDISON, P., MANNING J. a NUGENT T. *Unity Game Development Cookbook.* 1. WILEY UK: O'Reilly Media, 2019. ISBN 978-1491999158.

[2] *Unity: Documentation* [online]. [cit. 2021-10-15]. Dostupné z: `https://docs.unity3d.com/Manual/index.html`

[3] *Dart: Documentation* [online]. 2021 [cit. 2021-10-15]. Dostupné z: `https://dart.dev/guides`

[4] *Brackeys: BECOME A DEVELOPER* [online]. [cit. 2021-10-15]. Dostupné z: `https://brackeys.com/`

Jméno a pracoviště vedoucího práce:

Ing. Michal Moc
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta:

–

Michal Moc
.....
vedoucí práce

Datum zadání bakalářské práce: 15. 10. 2021

Termín odevzdání bakalářské práce: 7. 7. 2022

Doba platnosti zadání je dva roky od data zadání.

.....
garant oboru

.....
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE – Fakulta jaderná a fyzikálně inženýrská – 1 –

.....
děkan

V Praze dne 15. 10. 2021

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Děčíně dne ....

Pavel Urx

Poděkování

Děkuji Ing. Michalovi Mocovi za vedení mé bakalářské práce a pomoc s jejím vypracováním.

Pavel Urx

Název práce:
Vývoj her pro mobilní telefony

Autor: Pavel Urx

Studijní program: Aplikace přírodních věd
Obor: Aplikace softwarového inženýrství
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Michal Moc
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant: –

Abstrakt:
Tato práce se věnuje teorii a praxi herního vývoje. Jejím cílem je v první části seřadit a popsat jednotlivé fáze samotného vývoje i s ohledem na marketingovou a komercializační stránku. Práce provází vývojem od počátečních činností, věnuje pozornost i prostředkům a metodám, které se při vývoji užívají pro jeho zjednodušení. Popis fází byl vytvořen za pomoci informací ze zahraničních webů zabývajících se problematikou herního vývoje. Ve druhé části práce využívá získané znalosti pro vývoj jednoduché mobilní hry. Pro tvorbu byl využit modulární herní engine Flame jako součást frameworku Flutter nad jazykem Dart.

Klíčová slova: vývoj pro mobilní telefony, vývoj her, teorie vývoje her

Title:
Development of Mobile Phone Games

Author: Pavel Urx

Abstract: This thesis is focusing on theory and practical use of this information. The objective of the first part is to order and describe individual phases of game development, not forgetting to mention the marketing and commercialization options. The text is leading from beginnings of game development and mentioning software and methods available to ease the process of development. Many foreign resources were used to collect information provided by the work. In the second part, the thesis shows usage of written information in practice to create a simple mobile game. To make the game modular, game engine Flame as part of Flutter framework on top of the Dart language.
Key words: Mobile Phone Game development, Game development, Game development theory

Obsah

Úvod

Práce se věnuje teorii herního vývoje. Cílem práce je zmapovat možnosti, které jsou pro dnešní herní vývojáře k dispozici a popsat jednotlivé fáze, kterými by ve většině případů měla každý hra projít. Práce prezentuje pohled na herní vývoj spíše z pohledu jednotlivce, jako nezávislého vývojáře. Následně práce ukazuje využití znalosti z předchozí části k využití vybraných možností v praxi.

První pomyslná část práce řadí postupně fáze vývoje. Jako první práce rozebírá fázi pre-produkce, kde je vysvětleno, co vše je potřeba promyslet před začátkem vývoje. V této fázi probíhá rešerše technických možností a jejich volba. Jsou zde zmíněny i možnosti monetizace. Dále práce postupuje k tématu konceptu, zde je naznačeno, jakým způsobem připravit koncept hry, tvoření herních mechanik a jednoduché návrhy herních prvků. Dále se věnuje rozvrhu projektu a zejména důležitému design dokumentu. Po fázi konceptu přichází na řadu fáze vývoje, ve které jsou popsány jednotlivé části, ze kterých jsou herní objekty složeny a dále popisuje metody vývoje, které usnadňují vývoj. Dále se práce věnuje tématu testování hry a následující optimalizace a opravy chyb. V poslední řadě jsou popsány fáze dokončování hry jako je vydání beta verze hry a otázka marketingu, které jsou následovány poznámkami k vydání hry a následné post-produkci jako přidání dodatečného obsahu a případně další podpora formou aktualizací.

Ve druhé části práce je pak ukázáno využití konkrétních technologií pro tvorbu jednoduché hry pro mobilní telefon. Jmenovitě je ukázána práce s frameworkem Flutter pro tvorbu uživatelského prostředí aplikací a jazyku Dart pro popis logiky programu. Pro usnadnění vývoje je využit jednoduchý modulární 2D herní engine flame. Následně jsou ve hře implementovány jednoduché prvky monetizace. Veškeré kódy jsou následně zdokumentovány a vysvětleny. Výsledná hra je následně vydána pro mobilní telefony s operačním systémem Android, Flutter však umožňuje i jednoduchou migraci na zařízení používající IOS.

Kapitola 1

Preprodukce

1.1 Nápad

Pro absolutní začátek vývoje je nejprve potřeba přijít s nápadem na hru, který je ideálně rozšířitelný, tak aby hra mohla naplnit dostatečný obsah. Je tedy dobré, aby se nápad dal propojit s dalšími myšlenkami. V této fázi se zatím jedná pouze o hrubý náčrt, na který v dalších iteracích vývoje budeme přidávat další myšlenky. Je více než pravděpodobné, že se nápad postupně s dalšími fázemi vyvine trochu jiným směrem, než bylo na začátku zamýšleno. To je způsobeno tím, že s postupem času a hlubším ponořením do vývoje se mohou vyskytnout potíže s určitým řešením nápadu nebo také nápad, který celou myšlenku hry posune zase trochu jiným směrem. V této fázi je to naprosto přirozené. Je dobré mít na paměti, že zajímavý nápad může v budoucnu usnadnit práci s marketingem našeho produktu, protože dobrá a zajímavá myšlenka je sama schopna přitáhnout skupinu potenciálních uživatelů.

1.2 Žánr

V momentě, kdy je nápad na hru již dostatečně promyšlen, přichází na řadu výběr vhodného žánru pro hru samotnou. Často se k nápadu na hru sám vyskytne ideální žánr, nicméně je stejně důležité se zamyslet, jestli propojení našeho nápadu a žánru dává smysl, jestli bude rozumně proveditelný a co je nejdůležitější, jestli bude výsledný produkt zábavný. Pokud je žánr zvolen vhodně, nepochybně to usnadní další fáze vývoje, které nás ještě čekají.

1.3 Výběr cílové skupiny

Při výběru cílové skupiny je důležité mít na paměti, jaký nápad a žánr byl pro hru zvolen. Zejména by měli být zodpovězeny otázky typu „hraje naše cílová skupina takovýto typ her?“ nebo „Jaký styl a typ žánru je vhodný pro tuto skupinu?“. Dobrá

volba cílové skupiny zajistí určitou základnu uživatelů, kteří by potenciálně mohli přijít do hry. Je to tedy i důležitý počáteční marketingový tah.

1.4 Volba cílové platformy

S volbou platformy úzce souvisí zvolená cílové skupina. Je totiž důležité, aby na danou platformu cílová skupina měla snadný přístup a byla na platformě dostatečně zastoupena. Správný výběr nám do budoucna snadno zajistí přístup uživatelů, což je pro úspěch našeho titulu kritické.

1.5 Dostupné platformy

1.5.1 Steam

Zpracováno na základě [19]

Pro publikování hry na platformě Steam je nutné mít založený účet, na kterém v minulosti proběhla transakce v hodntě alespoň 5$. Následně je potřeba se skrze účet připojit ke službě Steamworks Distribution Program. Na steam je mimo her také možnost nahrát i jiný software.

Při publikaci hry je potřeba zaplatit jednorázový 100$ poplatek, který bude navrácen zpět v případě, že daná aplikace vydělá částku nad 1000$. Dále je potřeba počítat s tím, že Steam požaduje 30 % z celého výdělku, který hra vygenerovala.

Proces vydání hry začíná při vytváření stránky v obchodě. Dále je potřeba vyplnit klíčová slova, podle kterých bude pro uživatele hru snazší v obchodě nalézt a vyplnit údaje o minimálních a doporučených požadavcích na hardware počítače. Společnost Valve také požaduje alespoň přibližné datum vydání hry. Pro hru je také potřeba napsat krátký a dlouhý popis. Krátký, který se zobrazuje při listování nabídkou her a dlouhý pro samostatnou stránku našeho produktu.

V moment, kdy jsou všechny potřebné informace ke hře poskytnuty, může být hra publikována v obchodě. Předtím je ale ještě potřeba určit cenu, za kterou je hra nabízena a nahrát samotné soubory hry. Poté hra prochází kontrolní fází a je zveřejněna v obchodě.

1.5.2 Epic Games Store

zpracováno na základě [18]

Publikace hry na Epic Games Store je v současné době v uzavřené betě, kdokoliv se ale může přihlásit a tým zodpovědný za fungování obchodu posoudí, jestli vývojáři umožní přístup. Publikace je po přijetí do služby zdarma a Epic Games si bere pouze

12 % z ceny prodeje hry na této platformě. Velkou výhodou může být pro vývojáře v Unreal Engine 4, že pro publikaci na Epic Games Storu je osvobozen od poplatků.

1.5.3 Itch.io

zpracováno na základě [6]

Publikace na platormě je zdarma dostupná všem po registraci. Po vytvoření Landing page pro publikovanou hru je vše okamžitě připravenou pro veřejnou publikaci. Standardní část, kterou si poskytoval z prodeje hry bere je mezi 10 % až 30 %. Nastavit lze ale i 0 %.

1.5.4 Obchod Play

zpracováno na základě [20]

Pro publikaci aplikace nebo hry na Obchodě Play je nutné založit Google účet, ze kterého je nutné zaplatit jednorázový poplatek 25$ za přístup do Google Developer Console. Poté je možnost přidat libovolný počet aplikací do obchodu. Google požaduje 30 % z ceny za prodej aplikace a dalších 30 % ze všech příjmů tvořených In-game transakcemi.

1.5.5 App Store

zpracováno na základě [6]

Vydání hry na App Store je zpoplatněno 99$ za rok. Pro vstup do obchodu je nutné se přihlásit do Apple Developer Program. První rok od publikace si Apple nárokuje 30 % Příjmů aplikace. Po roce se poplatek sníží na 15 %, pokud má vývojář stále zaplacen roční poplatek.

1.5.6 další

Je velké množství dalších platforem, na které lze aplikace a hry publikovat a zde jsou uvedeny pouze nejrozšířenější možnosti. Dále lze publikovat například v Amazon Appstore, Humble Bundle, GOG a mnoho dalších.

1.6 Monetizace

zpracováno na základě [5]

Poslední kapitolou ve fázi preprodukce je otázka monetizace hry. K dispozici jsou různé způsoby, jak hru monetizovat. Je důležité zvolit vhodný způsob. Hry v dnešní době často užívají různých kombinací dohromady.

1.6.1 Prodej hry

Nejběžnější způsob monetizace hry je zpřístupnění hry hráči za poplatek předem zvolené částky. Prodej může být zprostředkování platformou, na kterou byla hra zveřejněna existují ale i společnosti, které se tomuto způsobu vyhýbají a hru raději prodávají sami například skrze vlastní webové stránky a vyhýbají se tak poplatkům, které si služba za zprostředkování transakcí nárokuje. Tento způsob je hojně využíván především ve světě her pro počítače a herní konzole. Jednorázová částka je často navýšena o prodejem volitelných stahovatelných součástí, které nejsou součástí jako jsou DLC a podobné.

1.6.2 Reklamy v aplikaci

Velmi rozšířeným způsobem, zejména ve světě her pro mobilní telefony, jsou reklamy v aplikaci. Možnost přidat do hry reklamy je velmi dostupná. K dispozici je velké množství poskytovatelů, kteří prostřednictvím kódu v aplikaci zobrazují reklamy od svých zákazníků. Aplikaci je pak vyplacena provize a zprostředkovatel si z této provize sám určitou částku také vezme. Reklamy jsou hojně využívány v kombinací s možností zaplatit určitý obnos peněz, aby se reklamy přestaly na daném zařízení nebo účtu v aplikaci zobrazovat. Je důležité zvolit správného poskytovatele reklam, protože se vyplacená částka jako odměna může výrazně lišit.

Nejčastějšími způsoby, jakým jsou v aplikaci zobrazeny reklamy jsou formou zobrazení malého banneru v okrajové části okna, tento způsob je ze všech druhů asi nejméně invazivní a uživatele příliš neruší. Dalším způsobem, jak zobrazovat reklamu v aplikaci je vsunutá stránka. Stránka s reklamou se v tomto případě zobrazí přes celou plochu displeje a pro pokračování v užívání aplikace je nutné jí zavřít tlačítkem. Tato reklama se obvykle zobrazuje při nějaké údálosti, která přeruší hru např. při postupu do dalšího levelu. Dalším hojně využívaným způsobem je typ rewarded ad, který zobrazí reklamu, často ve formě delší video ukázky. Po shlédnutí celé reklamy je pak hráči přidělená nějaká předem slíbená odměna. Mezi poměrně nové způsoby zobrazení reklamy patří hratelná reklama. Tato reklama je podobná interstitiální, pouze se liší tím, že není přehrávaný obsah, ale uživatel má možnost vyzkoušet si část aplikace, nejčastěji hry. Dále je možnost využít tzv. offerwall, kde je nabídka úkolů např. stáhnout nějakou aplikaci, které když splníme získáme nějakou předem definovanou odměnu. Je důležité dobře promyslet, jakou reklamu v aplikaci využít, protože se může snadno stát, že příliš invazivní reklamy snadno odradí uživatele od pokračování v užívání aplikace či hry.

1.6.3 Prodej obsahu v aplikaci

Dalším možným způsobem, jak monetizovat aplikaci je prodejem obsahu, který je ve hře. Prodávat lze téměř jakýkoliv obsah, co je ve hře zakomponovaný například od bonusových životů, přes kosmetické úpravy hry až po speciální herní předměty nebo prémiová měna. U většiny zprostředkovatelů stažení hry je ale nutné odvézt příslušnou část zisků, kterou si na in-game poplatky nárokuje.

1.6.4 Předplatné

Hra může být monetizována také na bázi předplatného, to funguje dvěma způsoby, jeden způsob je, že uživatel má přístup do hry jen v době, kdy má zaplacené časově omezené předplatné, nebo je hra volně přístupná a uživatel na dobrovolné bázi platí tzv. battlepass. Battlepass pak uživateli poskytuje přístup k prémiovému obsahu, který hra nabízí, nebo mu například přiděluje více odměn za herní aktivitu. V současné době je metoda monetizace prostřednictvím battlepassu hojně využívána a s dobrými výsledky.

1.6.5 Monetizace klíčových prvků ve hře

Častým způsobem generování zisku hry je monetizace klíčových prvků, bez kterých jinak nelze hru hrát, například energie pro pokračování ve hře, kterou lze doplnit pouze koupením doplňující energie nebo čekáním, než se energie zase doplní.

1.7 Výběr prostředků pro tvorbu

1.7.1 Herní Engine

zpracováno na základě [7]

Herní engine je prostředí, které obsahují sadu nástrojů, urychlující vývoj her. Enginy obvykle obsahují vlastní implementaci fyzikálního modelu, nástroje pro renderování objektů a shadery, nástroje pro usnadnění skriptování, umělou inteligenci nebo detekci kolizí. Každý engine navíc podporuje různá cílová zařízení a tak je důležité pamatovat, pro jakou platformu hra cílí a podle toho vybrat engine.

1.7.2 Unity

Unity je velice univerzální herní engine, podporující různé platformy zařízení a dají se v něm tvořit nejen hry ale například i animace a další. Engine samotný funguje velice jednoduše a je tak doporučován pro začínající vývojáře. Je zejména vhodný pro menší projekty ale i velké projekty zvládne poměrně dobře obstarat. U velkých projektů ale nepodává tak dobré výsledky jako jiné enginy. Engine má k dispozici Asset store, kde jsou k dispozici zdarma i zpoplatněné různé prostředky od jiných tvůrců, které urychlují a usnadňují vývoj hry. V poslední době navíc Unity zavedlo podporu vizuálních skriptování, čímž umožňujetvořit hry i lidem, kteří neumí vytvořit programovou složku hry.

1.7.3 Unreal Engine

zpracováno na základě [16]

Unreal Engine je velice univerzální, podporuje množství různých plaftorem. Není určen pouze pro vývoj her ale tvorbu filmů, architektury nebo simulací. Engine v porovnání s Unity není tak jednoduchý a je doporučován spíše pokročilým vývojářům. Je vhodný spíše pro větší projekty. K dispozici je pro vývojáře bez programovacích znalostí také vizuální skriptovací jazyk Blueprint.

1.7.4 Amazon Lumberyard

zpracováno na základě [17]

Open-source herní engine pod vývojem Amazonu. Je kompletně zdarma a přívětivý k začínajícímu uživateli. Je vhodný zejména pro vývoj 3D her. Výhodou enginu je snadná integrace se streamovací platformou Twitch. Nevýhodou může být horší optimalizace vedoucí k chybám. Engine také podporuje vizuální skriptovací jazyk. Základy jsou odvozené z funkcí CryENGINU.

1.7.5 Godot

jednoduchý zdarma open-source herní engine obsahující základní sadu nástrojů pro vývoj her. Je vhodný zejména pro začínající vývojáře a pro malé projekty. Na internetu existuje rozšířená knihovna assetů, které mohou urychlit a usnadnit vývoj.

1.7.6 CryENGINE

zpracováno na základě [15]

Univerzální zdarma engine, který je známý především díky velkému důrazu na vizuální stránku hry. Má k dispozici bohatou knihovnu assetů. Vhodný je zejména pro větší projekty.

1.7.7 další

Další herní Enginy jsou například Game Maker Studio, Cocos 2d-x, RPGmaker a velké množství ostatních.

1.8 Modelování a grafika

zpracováno na základě [4]

Pro tvoření modelů, textur a grafického rozhraní je k dispizici velké množství různých nástrojů. Mezi nejpoužívanější placený software pro modelování patří Maya, ZBrush, Cinema4D a další. Velmi kvalitní zdarma alternativou je pak Blender. Při tvorbě textur a grafického rozhraní je pak velmi často užíván Photoshop. Zdarma alternativou Photoshopu jsou pak open-source Gimp nebo Krita.

1.9 Animace

zpracováno na základě [8]

Pro animace je často užíván software Adobe Animate (pouze 2D animace), 3ds Max, Motionbuilder, Cinema4D a další. Mezi velmi kvalitní zdarma alternativy patří Blender. Animovat ale lze například i v Unity. K dispozici je pro vývojáře i velká knihovna s animacemi Mixamo, která je k dispozici zdarma.

1.10 Sound desing

zpracováno na základě [14]

Při designování zvuků do hry je možné použít například Wwise(zdarma za určených podmínek), FMOD(zdarma za určených podmínek), Nuendo a další. Pro tvorbu hudebního podkresu je možné využít zdarma Bosca Ceoil a pro střih zvuku Audacity.

Kapitola 2

Koncept

2.1 Detailní rozmyšlení hry

Nyní je na řadě dát hře konkrétní směr. Na konci této fáze by mělo být jasné, co vše ve hře bude a jakým způsobem to bude ztvárněné. Produktem této fáze by měl být přehledný design dokument. Dále je dobré mít časový rozvrh, ať je jasné, kolik času lze jaké fázi věnovat.

2.2 Design dokument

zpracováno na základě [1]

Design dokument je nejdůležitější částí projektu a měla by mu být věnována velká pozornost. Mělo by zde být definováno, co ve hře bude implementováno a jakým způsobem to bude fungovat. Dokument pak vede další fáze zamýšleným směrem a zabraňuje, aby projekt sešel z vytyčené cesty. V dokumentu by měla být poznamenána jakákoliv součást, ze které se výsledná hra bude skládat např. herní mechaniky, příběh, design úrovní nebo map, design zvuku, monetizační design. Veškerý obsah dokumentu by měl být popsán do nejmenších detailů, aby bylo jasné, jak přesně to má být implementováno.

2.3 Návrh příběhu

Pokud hra bude příběhová, měl by být napsán ještě před začátkem ostatních vývojových procesů, protože od něj by se mělo odvíjet vše ostatní, čím bude hra tvořena. Hra nutně příběh obsahovat nemusí, často je ale vhodné, aby alespoň minimální příběhové zasazení hru doprovázelo.

2.4 Návrh mechanik

První důležitou součástí konceptu jsou herní mechaniky. Mechaniky jsou hlavní část, která utváří průběh vytvářené hry, proto by jim měla být věnována dostatečná pozornost. Mechaniky definují, podle jakých pravidel bude hra hrána, co se stane když např. hráč zemře, dokončí level.

2.5 Koncept art

Pří fázi konceptu jsou řešeny základní otázky toho, jak hra bude vypadat, tak, aby vizuálně zapadala do kontextu. Například, pokud je postava hrdiny vykreslována je nutné, aby i samotný hrdina splňoval tyto vlastnosti vizuálně. V této fázi není potřeba zachycovat součásti herních objektů dokonale, ale načrtnout, jakým směrem bude hra vizuálně směřovat, jak se bude daná herní mechanika projevovat ve hře atd. V této fázi jsou tvořeny základy všech součástí hry, které jsou následně upravovány a vylepšovány v dalších fázích.

2.6 Vizuální design

zpracováno na základě [12]

Ve fázi tvoření hry po vizuální stránce je důležité věnovat pozornost různým aspektům, které by design měl splňovat. V první řadě je třeba dbát na komunikaci s hráčem. Hráč z vizuálních prvků může vyčíst velké množství informací, které mu usnadní průběh hrou. Hra naznačuje, kudy se má hráč vydat, kolik má životů, kdo je nepřítel a podobně. Dalším efektem vizuálů hry je uspokojení hráče. Hra by měla působit uspokojivě, když hráč uspěje, pokud například hráč úspěšně dokončí level, hra ho „odmění“ například „vítězným ceremoniálem“ atd. Podobně by hra měla komunikovat s hráčem i v případě neúspěchu. V třetí řadě je důležité, aby všechny vizuály hry byli laděny do stejného grafického stylu a nálady hry.

Dalším bodem, na který je potřeba při designování hry myslet je vizuální hierarchie. Je to způsob, jakým směrovat pozornost hráče směrem, jakým chceme. Důležité části mapy, nepřítele apod. lze zvýraznit například větší velikostí než ostatní herní objekty, světelným kontrastem nebo třeba i mírou detailu, který daná část hry má.

2.7 Level-Design

Level design je velice důležitou součástí dojmu, které v hráči hra evokuje. Dobrý level-design bere v potaz dobrou a snadnou orientaci hráče v prostoru, k čemu se pro usnadnění používají tzv. Landmarky(Landmark je bod zájmu, nějaká výrazná část mapy jako je socha, výjimečná budova a další.). Dalším prvkem, na který je kladen důraz v designování levelu jsou nenápadné vedoucí prvky. Vedoucí prvky

mají nasměrovat hráče správným směrem, často jsou dělané tak, aby nebylo na první pohled snadno rozpoznatelné, nicméně mozek tyto prvky vnímá a podvědomě se je snaží následovat. V případě kompetitivních her pro více hráčů je také důležité myslet na balancování mapy. Každá mapa by se měla snažit být co nejférovější pro každou stranu, která se proti sobě utkává. V tomto ohledu je nejjednodušší využít zrcadlení, které zajistí stejné podmínky pro každou stranu na mapě. V neposlední řadě je také potřeba dbát na vzhled mapy. To je ve výsledku to, co dělá hlavní hráčův dojem z celkové hry.

2.8 Sound-design

Součástí každé hry je i zvuková stránka. Kromě hudebního podkresu v pozadí hry se zvuk používá i k předání informací hráči např. zvuk stisknutí tlačítka, zvuk, když hráči dochází náboje apod. Zvuk je tedy prostředkem, jak hráči předávat informace, které pasivně přijme, aniž by o nich často věděl.

2.9 Plán projektu

Pro přehledný vývoj hry by měl být sestavený plán projektu, ve kterém jsou časově rozvrženy jednotlivé části vývoje. Plán projektu je zejména důležitý pro větší a časově náročnější projekty.

Kapitola 3

Vývoj

Ve fázi, kdy je koncept hry rozpracován do požadovaných detailů, může začít samotný vývoj hry. Dobrým zvykem při vývoji hry je používání tzv. Placeholderů a iterativní vývoj.

3.1 Placeholdery

Placeholder je hrubě připravený herní objekt např. postava hráče, která slouží k dočasné reprezentaci výsledného obsahu hry. Často se během vývoje stává, že se části v průběhu vývoje často přepracovávají, vylepšují a upravují tak, aby výsledná hra byla dobře uhlazená. V počátcích ale není nutné, aby vše fungovalo co nejlépe, stačí když je ve hře reprezentováno vše přibližně. Dělání modelů, skriptů atd. na vysoké úrovni je časově náročné, a proto se z počátku tímto způsobem šetří čas i rozpočet.

3.2 Iterativní vývoj

S konceptem placeholderů úzce souvisí iterativní vývoj hry. Všechny části hry se nejprve pospojují v základní hrubou ale fungující kostru, která se v dalších iteracích vývoje vylepšuje a přidává se další obsahu.

3.3 Modelování a animování

Cílem modelování je příprava všech vizuálů herních objektů, které se ve hře budou nacházet. Samotný model se skládá z množiny bodů(vertices), které jsou různě propojované stěnami tak, aby vytvořili trojúhelníkovou síť, která je pak vykreslována jako samotný 3D objekt. Základní metody využívané v modelování fungují na bázi posouvání jednotlivých bodů do určených tvarů nebo pokročilejší tvarovací nástroje, které práci s modelem urychlují. Existují i metody, kdy je reálný 3D objekt

pomocí speciálních technologií nasnímán a přetransformován do světa výpočetních technologií.

Hotové modely jsou následně připravované k animování nejčastěji tak, že je v modelu vytyčena pomyslná kostra(rigging), jejíž manipulací se pak přenáší na síť modelu simulace pohybu.

Když je model i s kostrou hotový je vše připraveno pro animování. To je nejčastěji praktikováno pomocí definování jednotlivých póz modelu v různých časových intervalech a software pro animaci následně dopočítá přechody mezi pózami, čímž vzniká simulace pohybu. Další možností animace je nasnímání pohybu reálného herce, který na sobě má množství senzorů, které jeho pohyb převedou tak, že je stačí pouze aplikovat na připravený charakter a není nutné nic animovat ručně.

3.4 Skriptování

Při skriptování se definují přesná pravidla, podle kterých se hra bude chovat. Vše, co se v průběhu herního času mění, je nutné definovat skriptem. Herní engine má připravenou sadu předpřipravených nástrojů, které se často ve hrách opakovaně užívají a urychlují práci při skriptování.

Hlavním nástrojem užívaným ve hrách je tzv. nekonečný cyklus. Nekonečný cyklus běží ve hře bez přestávky, pokud hra není pozastavena. V nekonečném cyklu jsou umístěny prvky, které jsou vázané na fyziku herního enginu, ovládání hráče a další prvky, které je nutné udržovat s každým snímkem aktualizované. Obvykle se užívají dva druhy nekonečného cyklu. Fixovaný cyklus má pevně daný počet iterací, které by měl, pokud je hra dobře optimalizovaná, projít během vykreslení jednoho snímku hry. Na fixovaný cyklus je obvykle navázána fyzika herního enginu, aby se projevovala co nejkonzistentněji. Tím druhým je cyklus, který se opakuje, co nejrychleji jak je to možné, je tedy vhodný zejména pro věci, které potřebují být nejrychleji ve hře aktualizovány.

Další skripty ovládající ostatní herní mechaniky jsou pak navázány obvykle na nekonečný cyklus podle potřeb hry.

Běžnou součástí herních enginů se stávají vizuální skriptovací jazyky. Vizuální skriptovací jazyk obvykle vypadá jako pomyslný diagram propojený tak, aby definoval fungování hry, aniž by byla potřeba cokoliv programovat klasickým způsobem. Vedle značné jednoduchosti ale vyplývá nevýhoda v omezenosti toho, čeho lze skriptem dosáhnout. Další značnou nevýhodou by se mohla stát i horší optimalizace skriptu samotného.

3.5 Zvuky

zpracováno na základě [9]

Zvuk má ve hře 3 hlavní cíle, které se hráčem komunikuje. Prvním cílem je podávat hráči informace o dění ve hře. Pokud například hráči dojdou náboje nebo se mu obnoví schopnost hra to hráči může nějakým zvukovým efektem oznámit. Dalším cílem zvukové stránky hry je dávat hráči zpětnou vazbu k dění ve hře. Hra přehraje „oslavnou“ melodii, pokud hráč dosáhne úspěchu a podobně nebo ho varuje, že činnost, kterou provedl nebyla správně. Zvuk je tedy velice užitečný při komunikaci hry s hráčem, kterou si často hráč ani neuvědomí. Třetím úkolem zvukových efektů a hudby je dodávat hře atmosféru dle potřeby. Zvuky dokážou hře dodat atmosféru a emoce tak, jak by se vizuálním efektem pouze obtížně navozovala.

3.6 Uživatelské prostředí

Uživatelské prostředí je obvykle první vizuální součást hry, se kterou je hráč v kontaktu. Je proto důležité, aby rozhraní plnilo své úkoly dobře a poskytovalo hráči nejlepší požitek z užívání.

Prostředí by mělo dobře zapadat do designu hry po a po vizuální stránce vypadat tak, aby co možná nejlépe odpovídalo zbytku designového řešení hry.

Pro dobrou uživatelskou přívětivost je důležité, aby bylo co nejlépe rozvržené tak, aby uživatel nemusel bloudit prostředím, a hledal například, jaké tlačítko ho přesune k požadovanému obsahu hry atd.

Rozlišujeme mezi prostředími, které jsou mimo herní svět a sloužící například k navigaci po menu a podobně a mezi tím, kdy jsou prvky rozhraní zabudované přímo v herním světě a jsou jeho součástí. V obou případech je důležité, aby rozhraní nepůsobilo rušivě na hráče, poskytovalo požadované informace přehledně a zejména důležité informace, které mají být hráči předány, aby byly dobře a rychle pro hráče k nalezení.

Kapitola 4

Testování a optimalizace

4.1 Testování

zpracováno na základě [2]

Když je dosáhne hra stavu, ve kterém je implementováno vše nebo alespoň většina, co bylo definováno ve fázi konceptu, může přejít do fáze testování. V této fázi je úkolem objevovat a opravovat veškeré chyby, které se ve hře mohou vyskytnout. Jednotlivé části hry jsou procházeny opakovaně se snahou hru přinutit chovat se nestandardním způsobem.

Prvním bodem, který je testován jsou problémy s herními mechanikami nebo stabilitou a funkčnosti herních objektů. Řeší se všechny problémy, které mohou vzniknout chybou ve vývoji včetně stability skriptů apod.

Dále je nutné otestovat kompatibilitu herních zařízení. Kvůli rozmanitost hardwarových zařízení, zejména těch mobilních, kde je trh rozdělen na mnoho výrobců a modelů s různými specifikacemi, je nutné otestovat, jestli se hra chová správně na každém podporovaném zařízení.

Důležitou roli v přívětivosti hry pro koncového uživatele je výkon. Hra musí být otestována, jestli použité skripty mají co nejoptimálnější chod, co se týče výpočetní náročnosti a hra nemá problém se ztrátami počtu snímků za vteřinu. Testuje se nakládání hry s hardwarovými prostředky, konektivitou k síti a další funkce vázané na hardware herního zařízení.

Nutné je splnit všechny právní požadavky zemí, ve kterých bude hra distribuována, ale také požadavky a podmínky platformy, na které bude hra distribuována. Testuje se i bezpečnost aplikace zejména pro hry, které využívají internetové připojení a uživatelské účty. V neposlední řadě je také otestovat chování synchronizace hry v případě her pro více hráčů. Například při nestabilním internetovém připojení.

4.2 První vydání

Jakmile hra projde úspěšně testovacími fázemi, může být vydána první herní verze. Alpha stádium hry často ještě může obsahovat chyby, které jsou nadále opravovány, nicméně hra by v základu měla být funkční a reprezentovat přibližnou finální verzi. Vydání je často prezentováno investorům, často také bývá zveřejňována pro hráče ve formě předběžného přístupu a komunitního testování.

Kapitola 5

Před vydáním

5.1 Marketing

zpracováno na základě [3]

Úspěch hry velice úzce souvisí s kvalitou marketingu, který hru doprovázel během vývoje i po vydání hry. Zpočátku slouží pro upoutání pozornosti potenciálních hráčů, následně pomáhá udržet hru v povědomí hráčů a přivádí nové hráče. Kvalitně provedená marketingová kampaň může být rozhodující pro úspěch hry jako celku, proto je nutné této kapitole vývoje věnovat velkou pozornost a mít dobrou strategii.

Na začátku marketingové kampaně by měl být známý cíl, kterého by měla kampaň dosáhnout, aby se mohla považovat za úspěšnou. Cílem může být například celkový počet stažení tvořené hry.

Dalším krokem, který výrazně ovlivní výsledky kampaně je zaměření se na cílovou skupiny. Cílová skupina snadno definuje, jaké prostředky pro kampaň budou nejvhodnější zvolit, protože budou mít největší dosah k vybrané skupině uživatelů.

Na základě analýzy předchozích kroků je na řadě vytvořit kampaň vhodnou pro prostředky, které využijeme. Mezi možnostmi jsou například: kampaň na sociálních sítích, reklamy v aplikacích a na webových stránkách, emailová kampaň nebo veřejné propagování hry v rámci různých médií jako jsou například podcasty o hrách, časopisy a mnoho dalšího.

Kvalitní obsah se lépe prodává, proto výrazně usnadní marketing, pokud je hra něčím unikátní a kvalitně zpracovaná. Produkt má větší šanci zaujmout uživatele, pokud pokud má dobře zpracovaná média, kterých si potenciální uživatel všimne jako první, tedy například ikonka obchodě nebo zajímavý vzhled webových stránek.

Kampaň je vhodné v průběhu měřit, jaký má vliv na počet hráčů a podobně. Někdy zdánlivě správně zvolená kampaň nemusí fungovat příliš dobře a můžeme dosáhnout lepších Výsledků jiným způsobem a nevydávat příliš prostředků na neefektivní kampaň. Dobrým zvykem v kampani je tedy její postupné testování a upravování tak, aby bylo dosaženo, co možná nejlepších výsledků.

Velice pozitivní vliv na oblíbenost hry může mít i naslouchání zpětné vazbě komunity. Změny, které komunita navrhuje mohou pomoci hře být zábavnější a více uživatelsky přívětivá ale také vrhají pozitivní světlo na vývojáře, který hru udržuje. Se spokojenou komunitou navíc roste počet uživatelů, kteří u hry déle zůstávají nebo se k ní zpět vrací a zvyšují tak její příjmy.

Kapitola 6

Vydání

Před vydáním hry je vhodné ještě naposledy provést kontrolu všech aspektů hry, aby se minimalizovala šance, že se po vydání pro veřejnost objeví zásadní chyby. Hra by měla být připravena s dostatečným předstihem před datem vydání, aby se všechny kontroly stihly provést včas. Během finální kontroly by se na hru nemělo sahat po obsahové stránce, aby se eliminovalo riziko vytvoření nových chyb. Pokud hra prošla kontrolou úspěšně, je připravena pro vydání. Častým zvykem zejména větší herních studií je tzv. „day one patch“, aktualizace, vydaná krátce po vydání hry, která opravuje chyby, které se nestihly před vydáním opravit.

Kapitola 7

Postprodukce

Po vydání hry se očekává, že hra bude po určitou časovou dobu následně udržována aktualizacemi a opravujícími záplatami. Nejdůležitější činností po vydání hry je opravování nově nalezených chyb. Dobře udržované hry si déle udrží svou komunitu a s tím generují zisky po delší dobu. V rámci prodloužení zájmu komunity je také dobré, pokud je to možné, hru obohacovat novým obsahem.

7.0.1 Dokoupitelný obsah

Nový obsah do hry je dalším způsobem, jak hře zvýšit atraktivitu a prodloužit tak interval po který generuje nové zisky. Důležitou součástí v otázce přidávání nového obsahu jsou rozšiřující balíčky, které hráči dají nové možnosti pro hraní, nové modely nebo například herní módy.

Kapitola 8

Vývoj hry

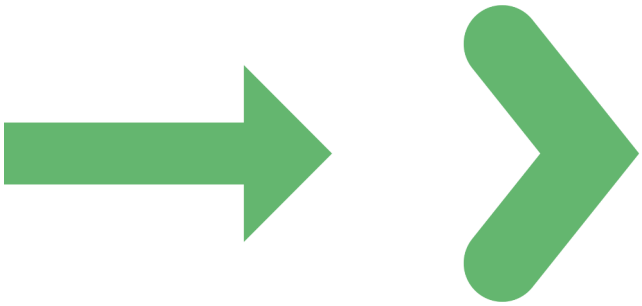
8.1 Příprava placeholderů

Při přípravě základů není nutné dbát při prvních iteracích na přehnanou dokonalost, modely budou předělávány a optimalizovány. Proto jsou využívány právě placeholdery, tedy objekty, které zatím pouze drží místo v hře a později budou nahrazeny za dokonalejší modely. Připravovaný model se bude pro přehled nazývat jako barvaSipky_smerSipky. Je potřeba se držet design dokumentu, aby se vývoj neodchýlil od plánu. V případě připravované hry je prvním placeholderem, který použijeme model šipky, které budou signalizovat hráči jaký tah má provést. Objekty dále není nutné nijak animovat, protože to hra nebude vyžadovat.

8.2 Skriptování

8.2.1 Třída ovládající běh hry

Třída GameBase, která obstarává propojení všech herních součástí ve funkční celek. Třída využívá nástroje pro pokročilé ovládání stavu widgetů Flutteru ChangeNotifier.



Obrázek 8.1: Ukázka vývoje vzhledu šipky

Konstruktor
Třídá má prázdný konstruktor.

```
import 'dart:async';
import 'package:bakalarkaflutter/game_assets/components/
    basic_block_component.dart';
import 'package:bakalarkaflutter/game_assets/game/game_event.dart';
import 'package:flame/game.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class GameBase extends FlameGame with ChangeNotifier {
  late int lives = 3;
  late BasicBlock currentBlock;
  late Vector2 screenSize;
  late int score = 0;
  double fromWhereToReact = 350;
  EventHandler eventHandler = EventHandler();
  String eventName = '';
  GameBase();
```

funkce z herního enginu

onLoad
Funkce, která proběhne při první iteraci nekonečného herního cyklu. Nastavuje vše potřebné pro začátek hry.

```
  @override
  Future<void> onLoad() async {
    spawnNext();
    eventHandler.startTimer();
    eventName = eventHandler.getEventName();
  }
```

onGameResize
Funkce která proběhne pokaždé, když se změní velikost zobrazaní hry. Přenastavuje proměnnou velikost plochy pro kreslení pro ostatní funkce.

```
  @override
  void onGameResize(Vector2 canvasSize) {
    screenSize = canvasSize;
    super.onGameResize(canvasSize);
  }
```

update

Funkce, která probíhá s každou iterací nekonečného herního cyklu. Používá se pro aktualizaci herních objektů a celkového stavu hry.

```
@override
  void update(double dt) {
    if (children.query<BasicBlock>().isNotEmpty &&
        children.query<BasicBlock>().last.position.y >=
   fromWhereToReact) {
      currentBlock = children.query<BasicBlock>().last;
```

```
    }
    updateEventName();
    super.update(dt);
  }
```

přidané funkce

SpawnNext
Funkce, která po doběhnutí intervalu časovače vyvolá další objekt šipky. Šipka je generována pomocí Třídy EventHandler, aby splňovala pravidla právě probíhajícího eventu.

```
  void spawnNext() {
    add(eventHandler.generateNext());
    spawnTimer();
  }
```

getNextBlock
Funkce, která se stará o přechod z úspěšně odehraného tahu na další objekt s instrukcí.

```
  void getNextBlock() {
    currentBlock = children.query<BasicBlock>().last;
  }
```

gestureHandler

zpracováno na základě [11]

Funkce, která identifikuje správnost hráčova tahu a podle toho reaguje dalším chováním. Funkce využívá nástroj notifyListeners, který aktualizuje hodnotu v příslušných widgetech flutteru.

```
void gestureHandler(String gesture) {
    if (currentBlock!.onGesture(gesture)) {
      score++;
      getNextBlock();
      HapticFeedback.lightImpact();
      getNextBlock();
    } else {
      lives--;
      if (lives < 1) {
        gameEnd();
      }
    }
    notifyListeners();
  }
```

updateTopScore
zpracováno na základě [13]

Funkce sloužící k aktualizaci maximálního skóre, pokud bylo překročeno předchozí maximum.

```
  void updateTopScore(int score) async {
    final prefs = await SharedPreferences.getInstance();
    final int? topScore = prefs.getInt('topScore');
    if (topScore! < score) {
      await prefs.remove('topScore');
      await prefs.setInt('topScore', score);
    }
  }
```

verticalDrag

zpracováno na základě [10]

Funkce sloužící k zachytávání hráčových tahů vertikálními směry po displeji.

```
void verticalDrag(DragEndDetails details) {
    String gesture = 'NONE';
    double speed = details.velocity.pixelsPerSecond.dy;
    if (speed > 0) {
      gesture = 'DOWN';
    } else if (speed < 0) {
      gesture = 'UP';
    }
    gestureHandler(gesture);
  }
```

horizontalDrag

zpracováno na základě [10]

Funkce sloužící k zachytávání hráčových tahů horizontálními směry po displeji.

```
  void horizontalDrag(DragEndDetails details) {
    String gesture = 'NONE';
    double speed = details.velocity.pixelsPerSecond.dx;
    if (speed > 0) {
      gesture = 'RIGHT';
    } else if (speed < 0) {
      gesture = 'LEFT';
    }
    gestureHandler(gesture);
  }
```

gameEnd
Funkce, která se stará o ukončení hry, pokud hráč neodehraje tah včas nebo správným směrem.

```
  void gameEnd() {
    pauseEngine();
    spawnTimer().cancel();
    removeAll(children);
  }
```

getScore
Funkce s návratovým typem celého čísla, která vrací současné hráčovo skóre.

```
  int getScore() {
    return score;
  }
```

updateEventName
Funkce, která se stará o poskytování informace hráči o probíhajícím eventu.

```
void updateEventName() {
    if (eventName != eventHandler.getEventName()) {
      eventName = eventHandler.getEventName();
      notifyListeners();
    }
  }
```

8.2.2 Skriptování objektu šipky

Objekt šipky ve hře bude ve skriptu zastupovat objekt třídy, který se bude obsahovat vše, co je od objektu ve hře požadováno.

Konstruktor
V konstruktoru se definují vlastnosti: Textura, která objekt zastoupí na vykresleném snímku hry. V případě herního enginu flame stačí v konstruktoru použít textový řetězec obsahující cestu k souboru, který má být použit. Rychlost bloku, definující, jak rychle se bude objekt během průběhu hry pohybovat. Pro definování rychlosti je zde využit typ double. Velikost bloku, upravující rozměry při vykreslování textury. Flame engine používá pro definici 2rozměrný vektor. Velikost je užívána v % velikosti displeje. Správné gesto, které bude blok požadovat pro úspěšné odehrání tahu. Pro jednoduchost je zvolen textový řetězec obsahující název instrukce.

```
import 'package:flame/components.dart';

class BasicBlock extends SpriteComponent with HasGameRef {
  late String spritePath; //path to image of the block
  late double blockSpeed = 1.0; //speed of block moving down
  late Vector2 blockSize = Vector2.all(100); //size of gameBlock
  late String correctGesture; //gesture required for correct play
  late Vector2 screenSize;

  BasicBlock(this.spritePath, this.blockSpeed, this.blockSize, this
    .correctGesture);
```

Funkce herního enginu
Herní engine, který je pro tvorbu hry definuje několik základních funkcí, které se spustí v určitých fázích životního cyklu herního objektu. Herní objekt je vytvořen pomocí dědičnosti, třídy spriteComponent definované v herním enginu.

onLoad
Funkce bez vstupního parametru, která se spustí právě jednou při vytváření instance objektu. V objektu šipky je nutné při instancování nastavit velikost pro vykreslení, která je převzatá z konstruktoru. Dále je využita funkce, která umístí herní objekt do souřadnicového systému, který engine používá. V tomto případě je pro všechny objekty typu šipky stejná počáteční souřadnice, proto mohou být souřadnice pro

první vykreslení objektu vždy stejné a není nutné je tedy definovat konstruktorem. V případě objektu šipky je to vždy polovina obrazovky na x ose a záporná velikost objektu, protože užívaný herní engine používá souřadnicový systém, který umisťuje souřadnice [0,0] do levého horního rohu displeje. Následně je nutné objektu nahrát texturu, ze zděděné třídy spriteComponent pro to využijeme proměnnou sprite typu sprite, do které pomocí cesty zadané v konstruktoru a funkce loadSprite.

```
1  @override //run while initializing
2  Future<void>? onLoad() async {
3    super.onLoad();
4    size = blockSize;
5    poseObject(size);
6    sprite = await gameRef.loadSprite(spritePath);
7  }
```

onGameResize
Funkce se vstupním parametrem velikost displeje, která proběhne vždy při inicializaci objektu a vždy při změně velikosti displeje. Velikost displeje je pro další potřeby objektu ukládána a dále spouští funkci definující změnu velikosti objektu.

```
1  @override //run on start and after resize
2  void onGameResize(Vector2 size) {
3    screenSize = size;
4    resizeObject(size);
5    super.onGameResize(size);
6  }
```

update
Funkce se vstupním parametrem čas od posledního snímku, která proběhne vždy při vykreslení nového snímku hry. V této funkci je užita funkce, která se stará o pohyb objektu směrem dolů po y ose displeje.

```
1  @override //run on every frame
2  void update(double dt) {
3    objectMover(blockSpeed, dt);
4    super.update(dt);
5  }
```

onRemove
Destruktor objektu, proběhne vždy při odstranění instance objektu. V užité verzi enginu flame je nutné postarat se o odstranění textury objektu.

```
1 @override
2  void onRemove() {
3    sprite!.image.dispose();
4    super.onRemove();
5  }
```

Funkce přidané

resizeObject
Funkce se vstupním parametrem, definuje velikost objektu na % velikosti displeje.

```
1 //set up gameBlock size to % of device screen
```

```
  void resizeObject(Vector2 size) {
    blockSize.x = size.x / 100 * blockSize.x;
    blockSize.y = size.x / 100 * blockSize.y;
  }
```

objectMover
Funkce se vstupními parametry čas od posledního snímku a rychlost, definuje, jak daleko se musí objekt před vykreslením posunout.

```
    //moves block
  void objectMover(double speed, double dt) {
    position.add(Vector2(0, speed * dt));
  }
```

poseObject
Funkce se vstupním parametrem velikosti displeje, která definuje kam se má objekt, při instancování ve hře, vykreslit.

```
  //sets block on correct sport
  void poseObject(Vector2 size) {
    position.x = (screenSize.x - blockSize.x) / 2;
    position.y = -blockSize.y;
  }
```

onGesture
Funkce se vstupním parametrem název gesta, která se stará o detekci správného gesta. Funkce vrací hodnotu true při správné detekci gesta.

```
    //reacts on gesture from display
  bool onGesture(String gesture) {
    if (gesture == correctGesture) {
      return true;
    }
    return false;
  }
```

8.2.3 Skriptování třídy správce eventů

Třída správce eventů se stará o chod a změny herních eventů.

Konstruktor
Třída obsahuje prázdný kosntruktor.

```
import 'dart:async';
import 'dart:math';
import 'package:bakalarkaflutter/game_assets/components/
   basic_block_component.dart';

import '../events/basic_event.dart';
import '../events/reversed_event.dart';

class EventHandler {
  BasicEvent currentEvent = BasicEvent('', 650);
  Timer eventTimer([int millis = 3000]) =>
```

```
      Timer(Duration(milliseconds: millis), getNewEvent);
  Timer noEventTimer([int millis = 3000]) =>
      Timer(Duration(milliseconds: millis), setBasicEvent);

  EventHandler();
```

getNewEvent
Funkce bez návratové hodnoty, která nastavuje nový herní event na kterýkoliv nezákladní event. A spouští časovač, který přepne po konci event zpět na základní event.

```
void getNewEvent() {
    noEventTimer();
    switch (Random().nextInt(1)) {
      case 0:
        {
          currentEvent = ReversedEvent('REVERSED', 650);
        }
        break;
      default:
        {
          currentEvent = BasicEvent('', 650);
        }
    }
  }
```

getEventName
Funkce s návratovým typem textového řetězce, která vrací název současně probíhajícího eventu.

```
  String getEventName() {
    return currentEvent.getName();
  }
```

generateNext
Funkce s návratovým typem basicBlock, která vrací nový herní objekt vygenerovaný podle pravidla současně probíhajícího eventu.

```
  BasicBlock generateNext() {
    return currentEvent.generateNext();
  }
```

setBasicEvent
Funkce bez návratové hodnoty, která nastavuje současný event na základní a spouští časovač spuštění náhodného nezákladního eventu.

```
  void setBasicEvent() {
    currentEvent = BasicEvent('', 650);
    eventTimer();
  }
```

startTimer
Funkce bez návratové hodnoty, která startuje na začátku hry běh střídání časovačů.

```
  void startTimer() {
    noEventTimer();
  }
```

8.2.4 Skriptování třídy hlavní event

Event, který běží ve hře v případě, kdy není spuštěný žádný jiný event. Event nijak neupravuje chování hry a nechává vše v základním stavu.

Konstruktor
V konstruktoru se definují vlastnosti: Název eventu typu String, slouží k identifikaci eventu pro další použití. Rychlost pohybu všech prvků, které event vygeneruje.

```
import 'dart:math';
import 'package:bakalarkaflutter/game_assets/components/
    basic_block_component.dart';
import 'package:flame/game.dart';

class BasicEvent {
  late String eventName;
  late double speed;
  BasicEvent(this.eventName, this.speed);
```

getName
Funkce s návratovou hodnotou typu string, která vrací název eventu.

```
  String getName() {
    return eventName;
  }
```

generateNext
Funkce s návratovou hodnotou typu BasicBlock, která vrací náhodně vygenerovaný objekt šipky.

```
 BasicBlock generateNext() {
    switch (Random().nextInt(7)) {
      case 0:
        {
          return (BasicBlock('up_green.png', speed, Vector2.all(50)
    , 'UP'));
        }
      case 1:
        {
          return (BasicBlock('down_green.png', speed, Vector2.all
    (50), 'DOWN'));
        }
      case 2:
        {
          return (BasicBlock('left_green.png', speed, Vector2.all
    (50), 'LEFT'));
        }
      case 3:
        {
          return (BasicBlock(
              'right_green.png', speed, Vector2.all(50), 'RIGHT'));
        }
      case 4:
        {
          return (BasicBlock('up_red.png', speed, Vector2.all(50),
    'DOWN'));
```

```
        }
      case 5:
        {
          return (BasicBlock('down_red.png', speed, Vector2.all(50)
    , 'UP'));
        }
      case 6:
        {
          return (BasicBlock('right_red.png', speed, Vector2.all
    (50), 'LEFT'));
        }
      default:
        {
          return (BasicBlock('left_red.png', speed, Vector2.all(50)
    , 'RIGHT'));
        }
    }
  }
```

8.2.5 Skriptování třídy obrácený event

Třída dědí z třídy základní event. Třída dědí vše ze základního eventu, upravuje pouze funkci generateNext.

generateNext
Funkce s návratovou hodnotou typu BasicBlock, která vrací náhodně vygenerovaný objekt šipky s obráceným správným tahem proti základnímu eventu.

```
@override
  BasicBlock generateNext() {
    double speed = 800;
    switch (Random().nextInt(7)) {
      case 0:
        {
          return (BasicBlock('up_green.png', speed, Vector2.all(50)
    , 'DOWN'));
        }
      case 1:
        {
          return (BasicBlock('down_green.png', speed, Vector2.all
    (50), 'UP'));
        }
      case 2:
        {
          return (BasicBlock(
              'left_green.png', speed, Vector2.all(50), 'RIGHT'));
        }
      case 3:
        {
          return (BasicBlock(
              'right_green.png', speed, Vector2.all(50), 'LEFT'));
        }
      case 4:
        {
```

```
25            return (BasicBlock('up_red.png', speed, Vector2.all(50),
    'UP'));
26          }
27        case 5:
28          {
29            return (BasicBlock('down_red.png', speed, Vector2.all(50)
    , 'DOWN'));
30          }
31        case 6:
32          {
33            return (BasicBlock('right_red.png', speed, Vector2.all
    (50), 'RIGHT'));
34          }
35        default:
36          {
37            return (BasicBlock('left_red.png', speed, Vector2.all(50)
    , 'LEFT'));
38          }
39      }
40    }
```

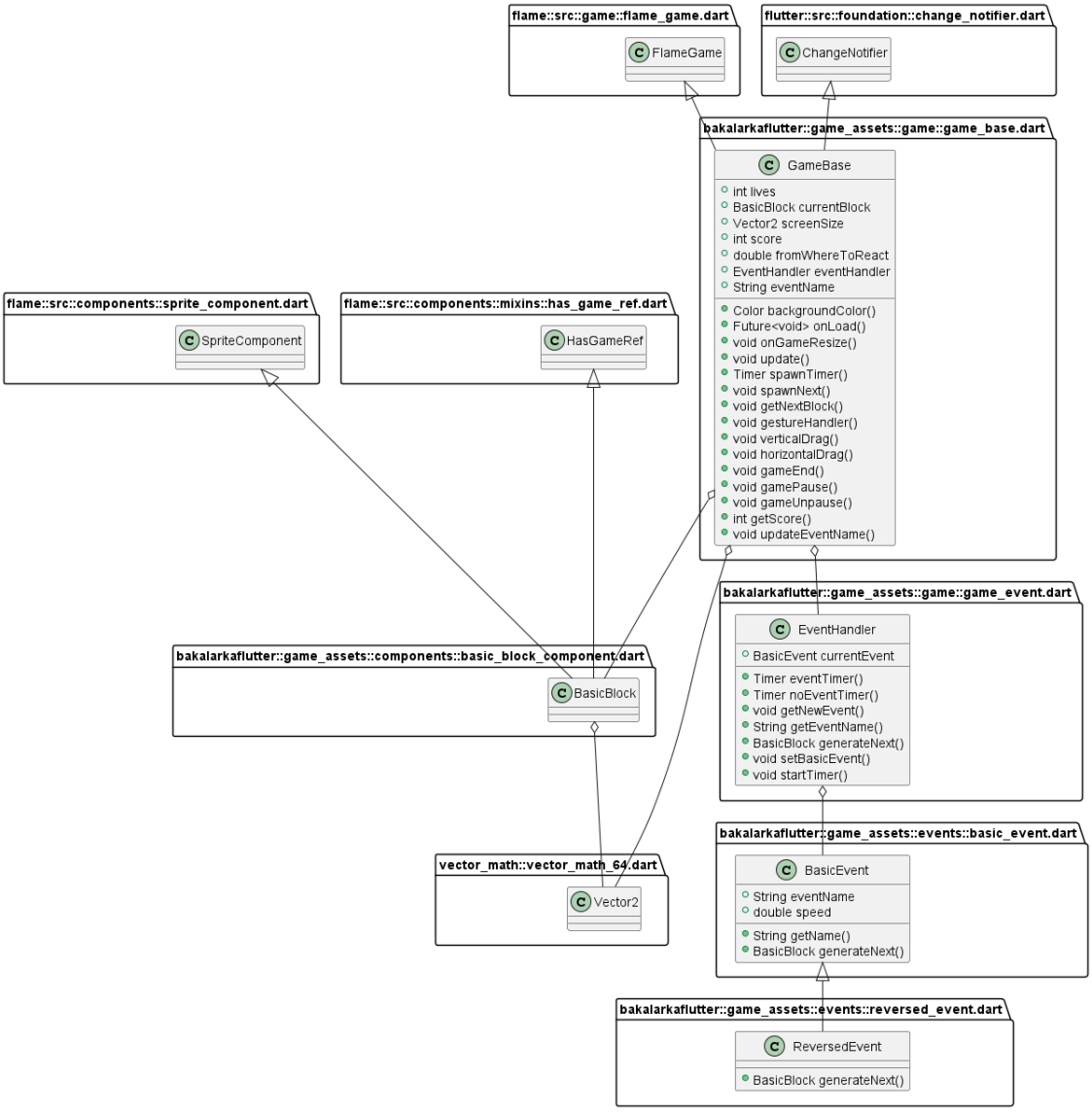
8.2.6 Třídy pro funkce tlačítek

Třída MenuRoutes
Třída obsahující skripty pro obsluhu tlačítek menu.

```
1  import 'package:bakalarkaflutter/UI/layouts/game_layout.dart';
2  import 'package:flutter/material.dart';
3  import 'package:provider/provider.dart';
4  import '../game_assets/game/game_base.dart';
5
6  class MenuRoutes {
7    void openGame(BuildContext context, GameBase base, GameScreen
     gameScreen) {
8      Navigator.push(
9          context,
10         MaterialPageRoute(
11             builder: (context) => ChangeNotifierProvider<GameBase>.
    value(
12                   value: base,
13                   child: gameScreen,
14                 )));
15   }
16 }
```

Třída GameRoutes
Třída obsahující skripty pro obsluhu tlačítek hry.

```
1  import '../game_assets/game/game_base.dart';
2
3  class GameRoutes {
4    void pausePlay(GameBase base) {
5      if (base.paused) {
6        base.resumeEngine();
7      } else {
```



Obrázek 8.2: Diagram propojení skriptů ovládající herní logiku

```
      base.pauseEngine();
    }
  }
}
```

8.3 Uživatelské prostředí

Aplikace obsahuje dvě různé rozložení stránky Menu, stránku se hrou a nastavení.

Stránka menu

Stránka menu v aplikaci obsahuje horní lištu s ukazatelem maximálního skóre, kterého hráč během hraní dosáhl. Dále obsahuje přesměrující tlačíko Play. Ve spodní části stránky menu je prostor, ve kterém je umístěn reklamní banner.

Menu je složeno ze základních flutter widgetů, které upravují vlastnosti rozložení stránky. Jako základ je použit upravený Flutter widget MenuScaffold, který upravuje základní Scaffold widget. MenuScaffold musí obsahovat horní lištu, základní Flutter widget AppBar, do kterého je na pravou stranu vložen text obsahující informaci o hráčově maximálním skóre. Widget scaffold dále vyžaduje body widget, který definuje vzhled stránky pod vrchní lištou. Jako body je zvolen Flutter widget Stack, který umožňuje detailní úpravu rozložení stránky. Stack widget vyžaduje 1 nebo více následujících widgetů. Jako nosný widget pro zobrazení jednotlivých tlačítek Menu je zvolen Flutter widget Column, který zobrazí tlačítka Menu v řadě pod sebe. Druhý widget vložený do následovných widgetů Stacku je vložen widget Align, který umožňuje definovat, kde se budou jeho následující widgety zobrazovat. Align je nastaven, aby se zobrazoval ve středu v dolní části displeje, a obsahuje widget pro zobrazení banneru z knihovny google mobile ads.

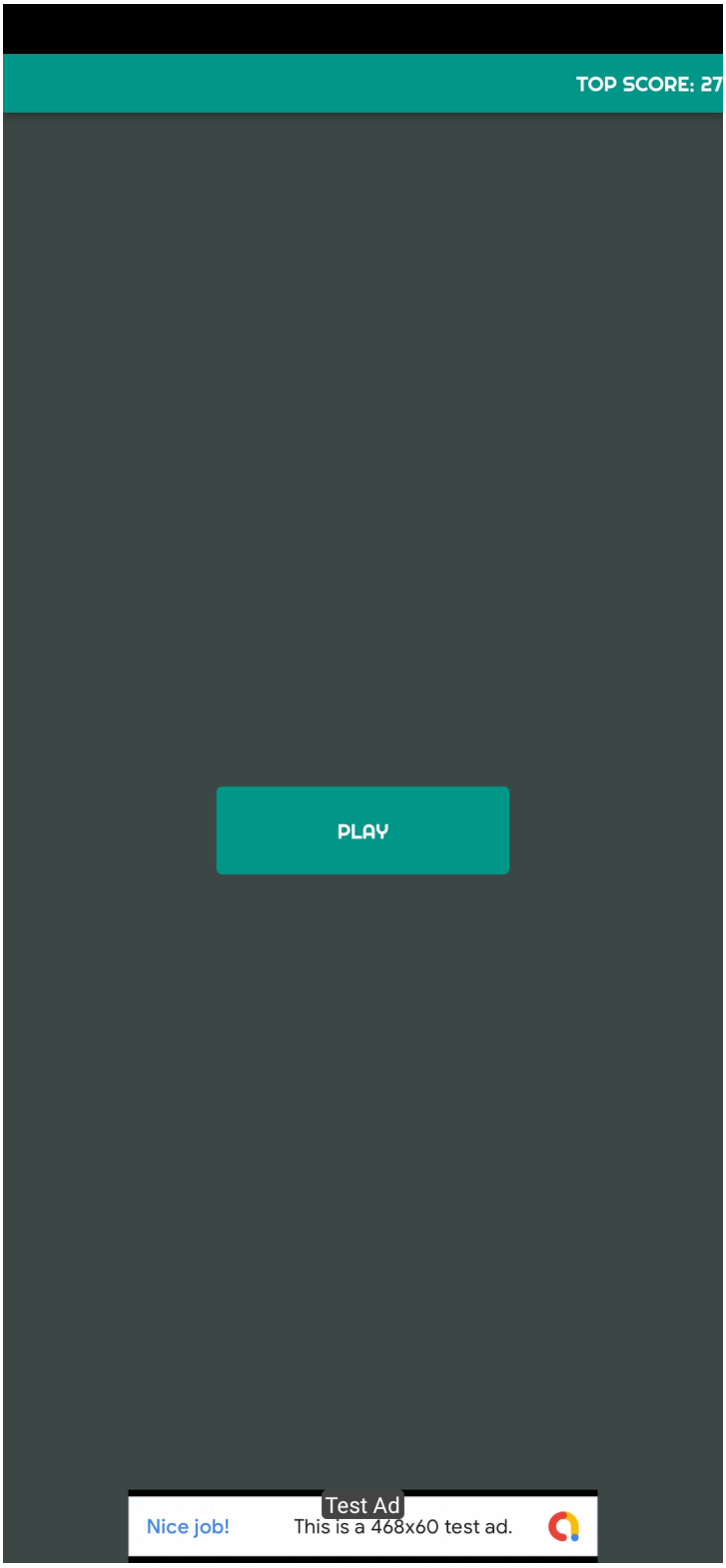
```
class Menu extends StatefulWidget {
  const Menu({Key? key}) : super(key: key);

  @override
  State<Menu> createState() => _MenuState();
}

class _MenuState extends State<Menu> {
  final BannerAd myBanner = BannerAd(
    adUnitId: '/*ID reklamy*/',
    size: AdSize.banner,
    request: const AdRequest(),
    listener: const BannerAdListener(),
  );

  @override
  void initState() {
    super.initState();
    myBanner.load();
  }

  @override
```



Obrázek 8.3: Ukázka vzhledu menu

```
23   Widget build(BuildContext context) {
24     MenuRoutes menuRoutes = MenuRoutes();
25     AppBarItem score = const AppBarItem(itemText:'TOP SCORE: 666');
26     AppBarContent content = AppBarContent(appBarItem: score);
27     return MenuScaffold(
28       appBar: AppBar(
29         actions: [content],
30       ),
31       body: Stack(
32         children: [
33           Center(
34             child: Column(
35               mainAxisAlignment: MainAxisAlignment.center,
36               children: [
37                 Column(mainAxisAlignment: MainAxisAlignment.center,
     children: [
38                   MenuButton(
39                       onPressed: () => menuRoutes.openGame(
40                           context, base, const GameScreen()),
41                       buttonText:'PLAY'),
42                 ]),
43                 Column(mainAxisAlignment: MainAxisAlignment.center,
     children: [
44                   MenuButton(
45                       onPressed: () =>
46                           GamesServices.isSignedIn == Future<bool>.
    value(true)
47                               ? GamesServices.showLeaderboards()
48                               : GamesServices.signIn(),
49                       buttonText: 'SIGN IN'),
50                 ]),
51               ],
52             ),
53           ),
54           Align(
55             alignment: Alignment.bottomCenter,
56             child: SizedBox(
57               width: 320.0,
58               height: 50,
59               child: AdWidget(ad: myBanner),
60             ),
61           )
62         ],
63       ),
64     );
65   }
66 }
```

Tlačítko menu

Jednoduché tlačítko s nápisem, využívající Upravený flutter widget MenuButton, který dědí vlastnosti flutter widgetu TextButton. Po kliknutí na tlačíto se provede daná událost.

Reklamní banner v Menu

Do středu spodní části Menu je vložena reklama, využívající widget z knihovny usnadňující implementaci GoogleAds.

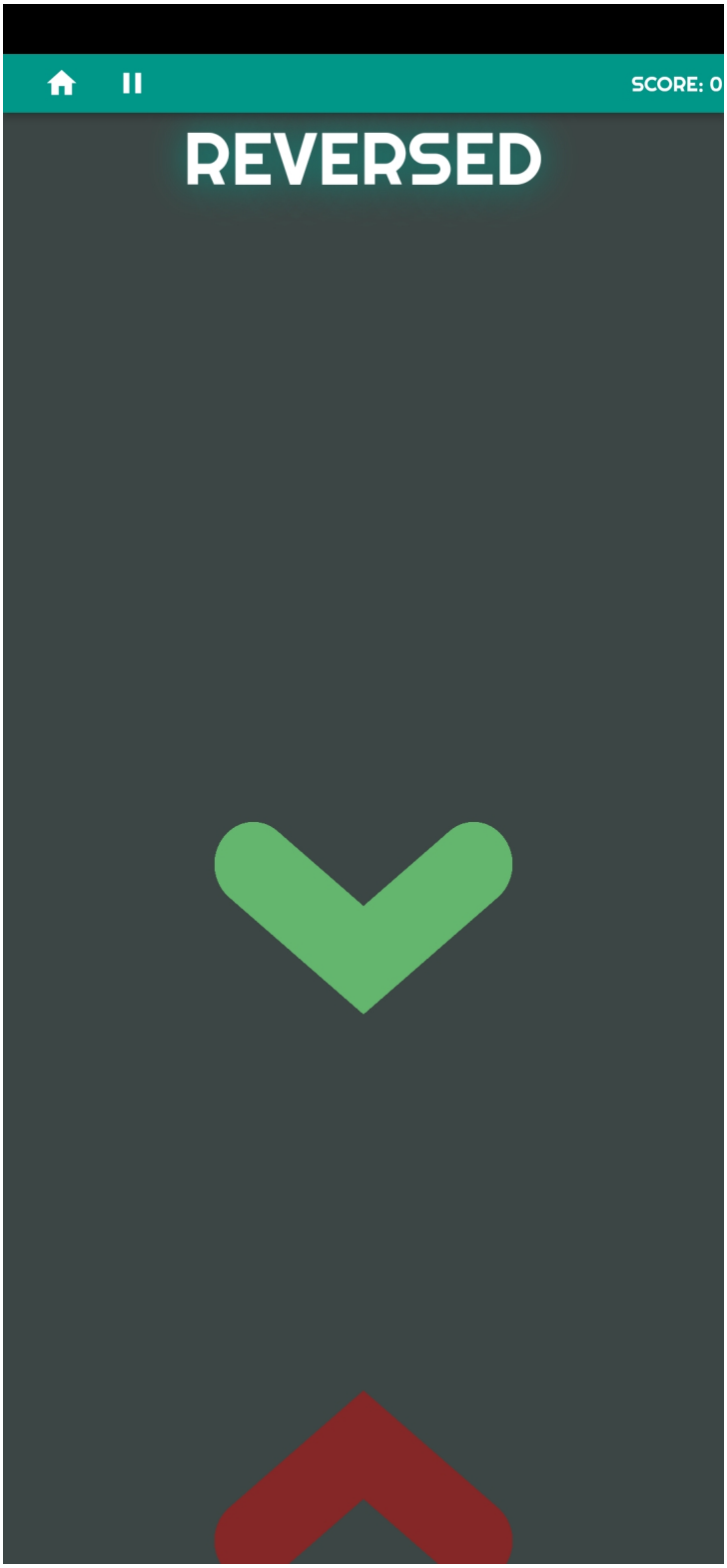
Stránka hry

Stránka hry je Scaffold widget obsahující Appbar, ve kterém je v levé části tlačítko pro návrat do stránky Menu a v pravé části Textový widget s ukazatelem současného skóre. V těle Scaffoldu je využitý widget Stack, který jako svého následovníka obsahuje základní Flutter widget GestureDetector, který identifikuje hráčův tah a posílá ho ke zpracování do části kódu zabývající se herní logikou, jako v pozadí widgetu je pak vložen widget z Flutter knihovny, usnadňující vývoj her, Flame který se stará o vizuální část hry. Ve Stacku je následně využito možnosti zobrazení přes ostatní widgety, kdy je nad GestureDetector vložen widget Textu, který zobrazuje informace o právě probíhajícím herním eventu.

```
GameBase base = GameBase();

class GameScreen extends StatefulWidget {
  const GameScreen({Key? key}) : super(key: key);
  @override
  State<GameScreen> createState() => _GameScreenState();
}

class _GameScreenState extends State<GameScreen> {
  @override
  Widget build(BuildContext context) {
    final provider = Provider.of<GameBase>(context);
    final GameRoutes gameRoutes = GameRoutes();
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: Row(
          children: [
            GameIconButton(
              onPressed: () {
                Navigator.pop(context);
              },
              icon: const Icon(Icons.home),
            ),
            GameIconButton(
                icon: base.paused
                    ? const Icon(Icons.play_arrow)
                    : const Icon(Icons.pause),
                onPressed: () => gameRoutes.pausePlay(base)),
          ],
        ),
        toolbarHeight: 40,
        actions: <Widget>[
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text('SCORE: ${provider.getScore()}'),
            ],
```



Obrázek 8.4: Ukázka vzhledu menu

```
          ),
        ],
      ),
      body: Stack(
        children: <Widget>[
          Center(
            child: GestureDetector(
              onHorizontalDragEnd: base.horizontalDrag,
              onVerticalDragEnd: base.verticalDrag,
              child: GameWidget(game: base),
            ),
          ),
          Align(
            alignment: Alignment.topCenter,
            child: Text(
              provider.eventName,
              style: AppTheme.eventTheme,
            ),
          ),
        ],
      ),
    );
  }
}
```

Upravované widgety

MenuButton

Upravený widget, který dědí vlastnosti ze základní třídy Flutteru StatelessWidget, neboli widget, který neupravuje svůj stav v průběhu životního cyklu aplikace. to znamená například, že po vykreslení widgetu na displej obrazovky již nezmění svůj vzhled. Widget je definován pomocí základního widgetu Container, který upravuje vzhled a rozměry svého následujícího widgetu, který je tvořen předdefinovaným widgetem TextButton. Pro vytvoření nové instance widgetu, vyžaduje widget funkci s návratovým typem GestureTapCallback, Textový řetězec, který bude vypsaný jako popis tlačítka. Ve widgetu je následně definován rozměr, který má tlačítko nabývat.

```
class MenuButton extends StatelessWidget {
  final GestureTapCallback onPressed;
  final String buttonText;
  const MenuButton(
      {Key? key, required this.onPressed, required this.buttonText
   })
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      margin: const EdgeInsets.fromLTRB(0, 0, 0, 10),
      child: TextButton(onPressed: onPressed, child: Text(
   buttonText)),
    );
  }
```

```
}
```

Úprava vzhledu aplikace

Třída téma aplikace

Třída upravující vzhled jednotlivých widgetů, které se v aplikaci objevují.

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class AppTheme {
  static ThemeData get lightTheme {
    return ThemeData(
        scaffoldBackgroundColor: const Color.fromARGB(255, 60, 70, 68),
        appBarTheme: const AppBarTheme(color: Colors.teal, toolbarHeight: 40),
        fontFamily: GoogleFonts.righteous().fontFamily,
        snackBarTheme: SnackBarThemeData(
            contentTextStyle:
                TextStyle(fontFamily: GoogleFonts.righteous().fontFamily)),
        textButtonTheme: TextButtonThemeData(
            style: ButtonStyle(
                fixedSize: MaterialStateProperty.all<Size>(const Size(200, 60)),
                foregroundColor: MaterialStateProperty.all<Color>(Colors.white),
                overlayColor: MaterialStateProperty.all<Color>(
                    const Color.fromARGB(255, 60, 70, 68)),
                backgroundColor:
                    MaterialStateProperty.all<Color>(Colors.teal)))
    );
  }

  static TextStyle get eventTheme {
    return TextStyle(
      fontFamily: GoogleFonts.righteous().fontFamily,
      fontSize: 50,
      color: Colors.white,
      shadows: const <Shadow>[Shadow(blurRadius: 50, color: Colors.teal)],
    );
  }
}
```

Závěr

Cílem práce bylo popsat jednotlivé fáze teorie herního vývoje. Dále rozebrat problematiku komercializace hry a marketingové možnosti.

V první části jsou seřazeny fáze tak, jak přibližně odpovídají realitě herního vývoje. Obsahují popis, co by se v dané fázi ve vývoji hry dít. K informacím o postupu jsou přidány i popisy metod, které se užívají, pro usnadnění a zlevnění celkového vývoje. V práci nechybí ani výběr možných softwarových nástrojů, které se pro vývoj běžně užívají a jejich popis. Dále jsou v práci detailně rozebrány možnosti komercializace hry a následně i způsoby marketingu pro získání větší uživatelské základny.

Ve druhé části je pak teorie využita v praxi k vývoji jednoduché hry pro mobilní telefony. K vývoji hry je využit framework Flutter s jednoduchým modulárním enginem Flame. Logiku a chod celé hry pak obstarává programovací jazyk Dart. Ve hře jsou implementovány prvky monetizace, formou reklamních bannerů, které se uživateli zobrazují v hlavním menu hry. K popisu funkcí hry jsou přiloženy i ukázky zdrojového kódu. V práci je popsán i zdrojový kód pro vizuální rozložení aplikace pomocí součástí frameworku Flutter. Z důvodu ranných vývojových fázi enginu Flame nebylo vše příliš dobře zdokumentováno, vývoj byl proto náročný. V současném stavu je hra plně funkční.

V budoucnu by se práce dala rozšířit o větší detaily a podrobnější poznámky k jednotlivým fázím. Dále aktualizovat informace z důvodu rychlého technologického posunu vpřed. Vyvíjená hra je připravena k rozšíření o další obsah například novými herními módy, více herními eventy, dalšími prvky upravující vzhled. Hru lze rozšířit o online tabulky s nejvyšším skóre, aby hráči mohli porovnávat své výsledky s ostatními hráči.

Literatura

[1] BABICH, Nick. Design Documentation: Why You Need It. *Adobe* [online]. San Jose: Nick Babich, April 22, 2020 [cit. 2022-02-02]. Dostupné z: https://xd.adobe.com/ideas/principles/web-design/design-documentation/

[2] HAMILTON, Thomas. Game Testing: Types & How to Test Mobile/Desktop Apps. *Guru99* [online]. Guru99, 2022, May 7, 2022 [cit. 2022-07-08]. Dostupné z: https://www.guru99.com/game-testing-mobile-desktop-apps.html

[3] HIETALAHTI, Juuso. The Basic Marketing Plan For Indie Games. *Gamasutra* [online]. London: UBM Tech [cit. 2022-05-07]. Dostupné z: https://www.gamasutra.com/view/feature/2695/the_basic_marketing_plan_for_indie_.php?print=1

[4] JARRATT, Steve. The best 3D modelling software in 2022. *Cretive Bloq* [online]. Bath: Creative Bloq, 2012, March 2, 2022 [cit. 2022-03-16]. Dostupné z: https://www.creativebloq.com/features/best-3d-modelling-software

[5] KNEZOVIC, Andrea. Mobile Game Monetization Trends: Best Strategies to Monetize Your Game in 2022. *Medium* [online]. Udonis, 2012, January 29, 2020 [cit. 2022-03-09]. Dostupné z: https://medium.com/udonis/mobile-game-monetization-trends-best-strategies-to-monetize-your-game-in-2020-fe2de42d4463

[6] NINICHI. 11 Places to Publish & Release Your Indie Game. *Ninichi* [online]. London: Ninichi, September 12, 2017 [cit. 2022-03-09]. Dostupné z: https://ninichimusic.com/blog/2017/9/1/11-places-to-publish-release-your-indie-game

[7] TYLER, Dustin. What is a Game Engine?. *Game designing* [online]. Dustin Tyler, 2022, March 9, 2022 [cit. 2022-03-14]. Dostupné z: https://www.gamedesigning.org/career/video-game-engines/

[8] WILLIAMS, Eva. 10 Best Video Game Animation Software in 2022. *Fix the Photo* [online]. Fix the Photo, 2003, February 20, 2022 [cit. 2022-03-16]. Dostupné z: https://fixthephoto.com/best-video-game-animation-software.html

[9] So You Wanna Make Games??: Episode 8: Sound Design. In: *YouTube* [online]. Riot Games, 13.12.2018 [cit. 2022-03-17]. Dostupné z: https://www.youtube.

com/watch?v=KcorIwJscFA&list=PL42m9XiTqPHJdJuVXO6Vf5ta5D07peiVx&index=8

[10] GestureDetector class. *Flutter* [online]. Google [cit. 2022-07-09]. Dostupné z: https://api.flutter.dev/flutter/widgets/GestureDetector-class.html

[11] Simple app state management. *Flutter* [online]. Google [cit. 2022-07-09]. Dostupné z: https://docs.flutter.dev/development/data-and-backend/state-mgmt/simple

[12] So You Wanna Make Games??. In: *YouTube* [online]. Riot Games, 2018, 13. 12. 2018 [cit. 2022-03-09]. Dostupné z: https://www.youtube.com/watch?v=RqRoXLLwJ8g&list=PL42m9XiTqPHJdJuVXO6Vf5ta5D07peiVx&index=1

[13] Shared preferences plugin. *Pub.dev* [online]. Flutter.dev, May 11,2022 [cit. 2022-07-09]. Dostupné z: https://pub.dev/packages/shared_preferences

[14] Audio Software for Game Development: 5 Solutions to Consider in 2021. *High Fidelity* [online]. High Fidelity, 2021, June 2, 2021 [cit. 2022-07-08]. Dostupné z: https://www.highfidelity.com/blog/audio-software-for-game-development

[15] CryENGINE. *Incredibuild* [online]. Incredibuild [cit. 2022-03-14]. Dostupné z: https://www.incredibuild.com/integrations/cryengine

[16] The world-s most open and advanced real-time 3D creation tool. *Unreal Engine* [online]. Epic Games [cit. 2022-03-14]. Dostupné z: https://www.unrealengine.com/en-US/

[17] Amazon Lumberyard. *Amazon* [online]. Bellevue: Amazon [cit. 2022-03-14]. Dostupné z: https://aws.amazon.com/lumberyard/

[18] Publish. *Epic Games* [online]. Cary: Epic Games, 1991 [cit. 2022-01-31]. Dostupné z: https://www.epicgames.com/store/en-US/publish

[19] Self-Publish On Steam: The Ultimate Guide. *Xsolla* [online]. Los Angeles: Xsolla, 2006 [cit. 2022-01-31]. Dostupné z: https://xsolla.com/blog/self-publish-on-steam-the-ultimate-guide

[20] Google Play Publishing: Publishing Your Game to the Google Play Store. *GameSalad* [online]. Austin: GameSalad, 2007 [cit. 2022-03-09]. Dostupné z: https://help.gamesalad.com/gamesalad-cookbook/publishing/google-play-publishing/

Příloha A

DVD

Příloha A

DVD obsahuje

- elektronická kopie této práce
- zdrojový kód vyvíjené mobilní hry
- apk soubor vyvíjené hry
- uživatelská příručka ke hře