FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Platformová 2D hra pro OS Android v Unity3D
Student:	Michal Šveigr
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat platformovou 2D hru pro operační systém Android. Vývoj aplikace proběhne v Unity3D.
Úkolem hráče bude ovládat postavičku tak, aby se úspěšně vyhnula všem překážkám a nástrahám v úrovních. Ovládání bude možné buď pomocí dotyků, nebo pomocí UI (tlačítek na displeyi).

Pokyny pro vypracování:
1. Seznamte se s enginem Unity3D.
2. Navrhněte hru a zasaďte ji do příběhu.
3. Diskutujte možnosti editoru úrovní.
4. Implementujte aplikaci.
5. Aplikaci podrobte uživatelským testům a diskutujte případné změny podle zpětné vazby.
6. Nahrajte aplikaci na Google Play v beta fázi.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Platformová 2D hra pro OS Android v Unity3D

Michal Šveigr

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

15. května 2018

Poděkování

Velice děkuji panu Ing. Miroslavu Balíkovi, Ph.D. za rady, korektury a veškerý čas, který mi věnoval při psaní mé bakalářské práce.

Prohlášení

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

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

V Praze dne 15. května 2018 ……

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2018 Michal Šveigr. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Šveigr, Michal. *Platformová 2D hra pro OS Android v Unity3D*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce seznamuje čtenáře s herním enginem Unity3D a popisuje vývoj 2D platformové hry pomocí tohoto enginu. V rámci přípravy na vývoj této hry byl vypracován herní návrhový dokument (game design document), který popisuje herní mechaniky a vlastnosti hry. Práce obsahuje popis podobných řešení a jejich srovnání s vizí autora, popisuje zvolený engine, analýzu a především implementaci a následné uživatelské testování . Vývoj a testování výsledné hry bylo úspěšné. Hra byla vydána ve fázi beta na obchod Google Play.

Klíčová slova Android, Hra, Unity

Abstract

The thesis is introducing gaming engine Unity3D and describes development of a 2D platform game made by the engine. Game design document was made while doing preparition for the development of this game which describes mechanics and features of the game. The thesis includes description of a similar games and its comparison with the vision of the author, describes the chosen engine, analysis and mainly the implementation and subsequent user testing.

Development and testing of the resulting game was succesful. The game was released as a beta version to the Google Play store.

Keywords Android, Game, Unity

Obsah

Seznam obrázků

Úvod

Dnešní doba je často označována za dobu mobilních telefonů a chytrých zařízení. Chytrá zařízení, včetně mobilních zařízení, zažívají v posledních letech velký boom. Když se rozhlédnete po ulici, těžko budete hledat člověka, který v ruce nedrží telefon či tablet, případně na zápěstí nemá chytré hodinky. Lidé pomocí těchto zařízení pracují, komunikují, plánují svůj čas, ale také si svůj čas krátí, když zrovna nemají co dělat. A to vše díky aplikacím, které jsou na tato zařízení v hojném počtu vyvíjeny.

Hry jsou v posledních letech také velkým tématem, jsou oblíbené napříč všemi věkovými skupinami a jejich popularita stále roste. Mezi nejoblíbenější a nejstarší žánry počítačových her patří platformové arkády, hovorově „platformovky“, ve kterých se hráč musí dostat z jednoho místa na druhé, přičemž mu cestu ztěžují různé překážky, pasti či protivníci. Tento žánr především prověřuje dovednosti hráče a jeho postřeh.

Hry jsou spojovány s počítači nebo různými herními konzolemi. Ovšem pokud spojíme mobilní zařízení a hry, dostáváme velice silné spojení, které má v dnešní době značné zastoupení v herním průmyslu hned vedle počítačových a konzolových her. A právě hrou na mobilní zařízení se tato práce zabývá.

Cílem práce je vytvořit platformovou hru na mobilní zařízení, která uživateli poskytne zábavu, odpočinek, ale třeba mu i pomůže přečkat dlouhou chvíli.

Kapitola 1

Cíle práce

Hlavním cílem práce je vytvoření 2D platformové hry a její následné nasazení na obchod Google Play ve fázi beta. Tento hlavní cíl je rozdělen do čtyř bodů. Prvním bodem je vytvořit herní návrhový dokument. V tomto dokumentu je hra podrobně popsána z pohledu designu. Druhý bod je navržení a implementace výsledné hry pro operační systém Android. Třetí bod představuje uživatelské testování a následné případné úpravy hry podle zpětné vazby. Posledním bodem práce je nasazení již částečně otestované hry na obchod Google Play pro otevřené beta testování.

Kapitola 2

Unity3D

Jako stavební kámen pro tuto práci byl zvolen herní engine Unity3D (též označován jako Unity). Jak už název enginu napovídá, Unity bylo vytvořeno primárně pro vývoj 3D her společností Unity Technologies. Později byl však engine vybaven značným množstvím nástrojů, které přímo podporují vývoj 2D her.

Mezi hlavní výhody enginu patří počet podporovaných platforem, dostupnost a počet jeho uživatelů. Pro rozšíření aplikace na různorodé platformy stačí minimální úpravy. Úpravy se týkají zejména oblasti vstupu.

Unity je dostupné ve třech verzích - Personal, Plus a Pro. Pouze verze Pro není omezena žádným ročním výdělkem. Pro Plus platí omezení 200 000$ za rok a pro Personal 100 000$ za rok [1]. Verze Plus a Pro jsou placené a disponují větším množstvím nástrojů, které ovšem nemají žádný vliv na výslednou aplikaci z hlediska kvality. Zpravidla tyto nástroje slouží pouze pro usnadnění spolupráce a zrychlení vývoje v týmu. V neplacené verzi lze tedy vydávat téměř stejně neomezeně jako ve verzích placených. Jediná nevýhoda neplacené verze je přítomnost animace Unity loga při načítání aplikace, kterou nelze odstranit. Tato nevýhoda je ovšem pro většinu uživatelů zanedbatelná a díky tomu Unity nabývá značné vývojářské základny. Tento velký počet vývojářů postupem času vytvořil nespočetné množství návodů a doplňků, které usnadňují práci.

Tato kapitola se zabývá základními částmi enginu, především částmi souvisejícími s 2D a vychází z manuálových stránek Unity enginu [2].

2.1 Základní prvky a pracovní plochy

Unity má několik oken a důležitých částí, na kterých je práce v něm postavena. Vývoj v tomto enginu se odehrává na dvou místech. Prvním z nich je Unity Editor, kterému se také říká editor scén. Druhým místem je editor skriptů, což

může být jakékoliv vývojové prostředí. Pro psaní skriptů si může programátor vybrat jazyk C# nebo JavaScript.

2.1.1 Pracovní okna

Samotný Unity editor je v základu rozdělen do několika oken (viz obr. 5.2), která mají vždy určité zaměření. Okna v editoru lze libovolně přemisťovat. Je zde i možnost vytvoření vlastního okna s úplně novou funkcionalitou.

Mezi nejčastěji používaná okna patří Projektové a Hierarchické okno, Náhled scény, Náhled hry, Inspektor a Profiler. Dále je zde Animační okno, Konzole, Zvukové okno a mnoho dalších.

Projektové okno Projektové okno slouží k přístupu k jednotlivým assetům projektu. Assetem se v Unity rozumí jakýkoliv soubor, který může být v projektu použit. Assety mohou být následně nataženy na scénu nebo použity na libovolný herní objekt. Je zde také možné spravovat adresářovou strukturu projektu nebo vytvářet nové assety.

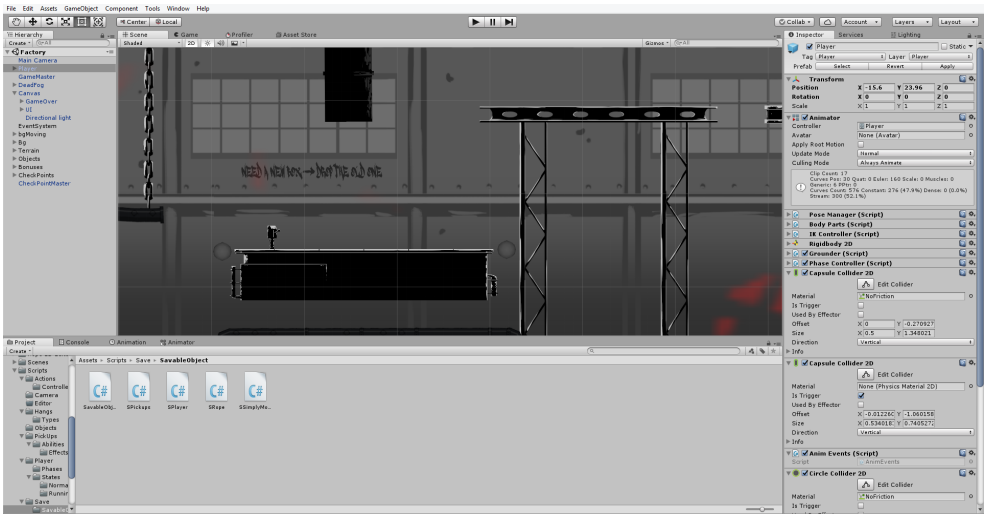
Hierarchické okno V tomto okně je seznam všech herních objektů aktuální scény. Ve výchozím nastavení se objekty řadí dle času vytvoření. Je zde možné vytvářet nové herní objekty nebo nastavovat herním objektům jejich potomky či rodiče.

Náhled scény Jedná se o interaktivní okno, které slouží k tvorbě celého herního světa. Lze zde manipulovat s objekty nacházejícími se na scéně. Například měnit jejich natočení, polohu i velikost.

Inspektor Inspektor slouží k zobrazení detailních informací vybraného herního objektu včetně jeho komponent. Komponenty zde můžeme modifikovat, případně přidávat nebo odebírat. Pokud má herní objekt vlastní skript s veřejnými proměnnými, je možné přes inspektor tyto proměnné přímo inicializovat.

Náhled hry V tomto okně je vyobrazen aktuální výsledek zvolené scény. Okno je vykresleno na základě kamer ve scéně. Je zde možnost nastavení konkrétního rozlišení a poměrů stran obrazovky, díky tomu je možné snadno provádět simulace jednotlivých displejů. Náhledové okno nabízí tzv. herní mód. V něm je možné si danou scénu přímo zahrát. Během tohoto módu je možné provádět změny ve scéně, ovšem po opuštění jsou všechny provedené změny navráceny.

Profiler Profiler je nástroj, který pomáhá odhalovat slabá místa projektu z pohledu optimalizace času. Tento nástroj poskytuje přesné informace o konkrétních výpočetních časech, které spotřebují jednotlivé části projektu při běhu hry.



Obrázek 2.1: Vzhled Unity editoru v základním nastavení

Profiler nabízí dva možné způsoby, jak tyto informace vidět. Prvním z nich je zobrazení grafů, kterých je celkově třináct. Mezi ty nejzajímavější patří graf využití CPU, využití GPU nebo graf ukazující počet aktuálních fyzikálních objektů na scéně společně s informacemi o jejich aktivitě. Druhým způsobem je hierarchické okno profileru. V tomto okně je vyobrazena hierarchie všech výpočtů a akcí, které jsou v daný moment hry prováděny. Nejlepší je pak používat kombinaci obou způsobů, kdy je pomocí grafu nalezen problematický moment hry a následně přes hierarchické okno konkrétní objekt a část kódu, která je za problém zodpovědná.

2.1.2 Scéna

Scénou se v Unity rozumí 3D prostředí, které slouží pro tvorbu herního světa. Při vytvoření scény je prostor téměř prázdný, obsahuje pouze jednu kameru. Následně je možné do scény přidávat herní objekty a utvářet tak herní svět. Mezi scénami je možné přecházet. Je zde tedy například možnost mít samostatnou scénu pro každou úroveň hry.

2.1.3 GameObject

GameObject, neboli herní objekt, je hlavní myšlenka Unity, na které je celý tento engine postaven. Vše co je ve hře vidět, od kamery, hráče, speciálních efektů, až po tlačítko v menu, je tvořeno minimálně jedním herním objektem.

Sám o sobě je GameObject ve hře neviditelný, obsahuje pouze komponentu **Transform**, která nese informace o jeho pozici, natočení a velikosti.

Tyto informace jsou vždy uložené ve vztahu k předkovi daného herního objektu, případně k počátku scény, pokud GameObject žádného předka nemá. **Transform** komponenta také obsahuje spoustu metod, které velice usnadňují práci vývojáře. Herní objekty, které jsou součástí uživatelského rozhraní, nemají přímo komponentu **Transform**, nýbrž její odvozenou instanci **RectTransform**.

2.1.4 Komponenty

Pokud od herního objektu chceme nějaké chování, ať už to je vizuální aktivita či kolize, je nutné mu přidělit patřičné komponenty, které toto chování zajistí. Tyto komponenty pak lze pomocí případných parametrů nastavovat a tím ještě lépe specifikovat chování herního objektu.

Aby komponenta mohla být přidělena hernímu objektu, musí dědit od třídy **MonoBehaviour**.

MonoBehaviour MonoBehaviour je jednou ze základních tříd Unity, od které musí všechny komponenty dědit. Tato třída poskytuje přístup k hernímu objektu, ke kterému je komponenta přiřazena, ale také nabízí různé životní metody, které jsou volány po celý průběh hry. Například metodu **Start** nebo **Update**. Start je volána při první aktivaci objektu tak, aby vždy proběhla před prvním voláním Update. Metoda Update je volána každý herní tik za předpokladu, že je komponenta aktivována.

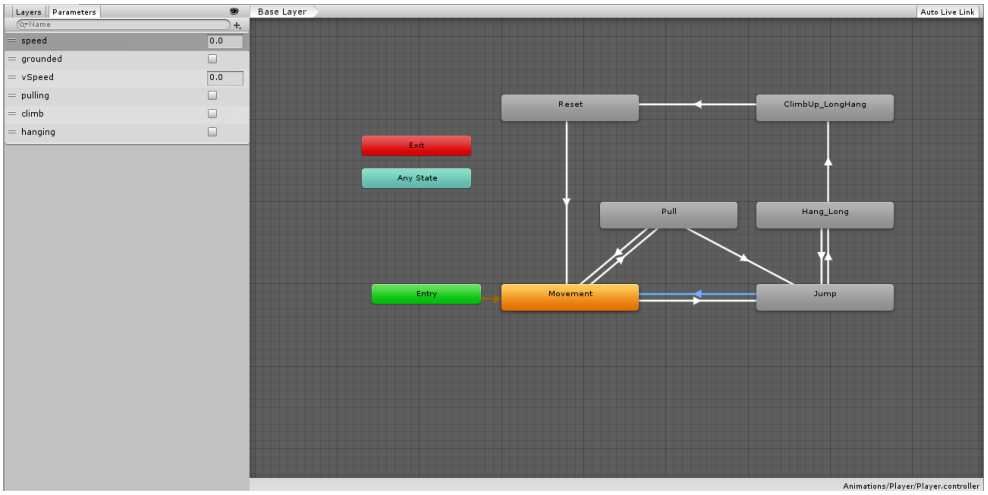
2.1.5 Mecanim

Unity disponuje animačním systémem jménem **Mecanim**, který nabízí široké spektrum možností tvorby a úpravy animací. Mecanim pracuje na principu animačních klipů. Jednotlivé animační klipy mohou být připraveny v různých externích programech, například Maya nebo Blender. Klipy lze také vytvořit přímo v Unity.

Hotové animační klipy jsou pak následně uspořádávány do **Animation Controlleru**. Animation Controller je stavový automat (viz obr. 5.2). Každý stav v tomto automatu je animační klip, případně seskupení několika takových klipů. Přechody v tomto automatu umožňují míchání klipů tak, aby bylo možné dosáhnout plynulých přechodů mezi nimi. Jednotlivé přechody se dají řídit pomocí proměnných. Rozhraní Animation Controlleru tyto proměnné zprostředkovává zbytku herního světa. Díky tomu je lze nastavovat mimo tento controller, typicky pomocí kódu.

2.1.6 Prefab

Když je potřeba mít více duplikátů jednoho objektu ve scéně zároveň, není potřeba je všechny postupně vytvářet a nastavovat. Unity nabízí možnost



Obrázek 2.2: Animation Controller hlavní postavy ze hry BadDream.

uložení herního objektu v jeho aktuálním stavu včetně nastavení jednotlivých komponent. Pro uložení objektu je nutné jeho přetažení ze scény do adresáře projektu. Takto uložený GameObject se nazývá **Prefab**. Prefab pak můžeme použít k vytvoření několika kopií jednoho objektu. Pokud je následně potřeba změnit nastavení některé z komponent, stačí toto nastavení změnit pouze u Prefabu. Všechny jeho instance použité ve scéně se automaticky změní také. Zároveň je Prefab přenosný mezi projekty za předpokladu, že společně s ním jsou přeneseny i všechny jeho nově vytvořené komponenty. Tedy ty, které nejsou standardně součástí Unity enginu.

2.2 Podpora 2D

Jak už bylo řečeno, Unity bylo dříve orientováno především na vývoj 3D. Z tohoto důvodu byl vývoj 2D her v tomto enginu problematický, ale přece jen možný. Ovšem vyžadoval spoustu externích doplňků a ústupků v rámci návrhu, hlavně kvůli optimalizaci.

To platilo do verze 4.3 [3]. Po vydání této verze se vývoj 2D her v Unity posunul na výrazně lepší úroveň. Byla přidána podpora Spritů, vlastní fyzikální engine pro 2D, možnost přepnutí náhledu scény do 2D módu a mnoho dalších věcí. Vývojáři Unity se stále snaží přispívat novými a lepšími nástroji, díky kterým by byl vývoj 2D prostředí ještě o něco snazší.

2.2.1 Fyzika

Při aktualizaci na verzi 4.3 byl do Unity přidán také další fyzikální engine. Od té doby existují v Unity dva oddělené fyzikální enginy. **Nvidia PhysX**

a **Box2D**. Nvidia PhysX se stará o 3D fyziku a Box2D o fyziku 2D. Fyzikální komponenty zaměřující se na 2D je možné poznat podle "2D" v názvu. Například komponenta **BoxCollider** je zaměřena na 3D prostředí, kdežto **BoxCollider2D** pouze na 2D.

V Unity je fyzika jednotlivých herních objektů zprostředkována skrz komponenty. Následující fyzikální komponenty patří mezi nejpoužívanější.

RigidBody2D **Transform** komponenta již byla zmíněna. Za normálních okolností je pomocí této komponenty ovládána pozice, natočení, či velikost herního objektu. Pokud je jedna z těchto hodnot změněna, Transform komponenta uvědomí ostatní komponenty, které na základě toho patřičně reagují, například změní pozici kolizního tělesa.

Fyzikální engine musí být ze své podstaty schopen hýbat s herními objekty. Proto je nutné, aby měl schopnost uvědomit jednotlivé Transform komponenty o změně pozice. K tomuto účelu byla vyvinuta komponenta **RigidBody2D**. RigidBody2D je hlavní fyzikální komponentou. Všechny herní objekty vybavené touto komponentou má pod kontrolou fyzikální engine.

Tyto objekty jsou rozděleny do tří skupin podle typu těla. Typ těla se nastavuje přímo v komponentě a může nabývat hodnot **Static**, **Kinematic**, **Dynamic**. Výběr typu těla ovlivňuje možnosti nastavení komponenty **RigidBody2D**.

Statický herní objekt se po celou dobu hry nehýbe. Takový objekt může vzniknout dvěma způsoby. Prvním je přidání pouze kolizní komponenty. Pokud na objektu není žádná komponenta RigidBody2D, spolu s kolizní komponentou se automaticky přidá neviditelné RigidBody2D označené jako **Static**. Druhý způsob je přidání RigidBody2D a následné nastavení typu těla na **Static**. Tento způsob vznikl až v novějších aktualizacích především z důvodu optimalizace. U statických objektů je možné nastavit pouze fyzikální materiál a aktivovat/deaktivovat simulaci. Simulací se v tomto případě myslí, zda má objekt kolidovat s jinými.

Kinematický objekt je takový objekt, který se smí v průběhu hry hýbat, může nabýt rychlosti, ale nedodržuje žádná fyzikální pravidla a nepůsobí na něj žádné vnější síly včetně té gravitační. Zjednodušeně řečeno, veškerý pohyb kinematického herního objektu je nutné ovládat pomocí skriptů. Aby se objekt stal kinematickým, je potřeba mu přidat RigidBody2D a nastavit typ jeho těla na **Kinematic**. Tím se zajistí kinematické chování herního objektu. Pokud navíc má objekt s něčím kolidovat, je nutné mu přidat nějakou kolizní komponentu.

Posledním typem je dynamické tělo. Herní objekt s komponentou RigidBody2D nastavenou na **Dynamic** podléhá fyzikálním zákonům hry. Působí na něj gravitační síla i ostatní vnější vlivy. U takového objektu je možné nastavit například jeho hmotnost nebo velikost gravitační síly, která na něj působí. Dynamické tělo je nejdražší, co se výkonu týče.

Collider Collider je kolizní komponenta, která vytyčí danému hernímu objektu tvar pro možné fyzikální kolize. Sám o sobě je neviditelný a typicky nemusí být stejného tvaru jako renderovaný objekt. Collidery lze pomyslně rozdělit do dvou skupin: primitivní a volného tvaru. Unity má rozdílné collidery pro 3D a 2D. Níže uvedené jsou zaměřeny čistě na 2D prostředí.

Mezi primitivní collidery patří:

- **BoxCollider2D**: Tento collider lze upravovat do libovolných čtverců či obdélníků. Jeho strany jsou vždy rovnoběžné s osami souřadnicového systému. Pokud collider chceme natočit, musíme otáčet s celým herním objektem.
- **CircleCollider2D**: Collider ve tvaru kruhu, jehož střed lze posouvat a poloměr měnit.
- **CapsuleCollider2D**: Má tvar kapsle. Tento tvar si lze zjednodušeně představit jako kombinaci výše zmíněných. Jedná se obdélník, případně čtverec, který má dvě protější strany zakulacené a vypouklé směrem ven. U toho collideru lze měnit velikost a orientace, která může být horizontální nebo vertikální. Orientace ovlivňuje jaké dvě strany jsou zakulacené. Ohledně dodatečného natočení platí to stejné, co u BoxCollideru.

Do skupiny volného tvaru patří:

- **EdgeCollider2D**: Jedná se o lomenou čáru, která může být jakkoliv dlouhá a nabývat libovolných tvarů. U tohoto collideru nelze docílit úplně uzavřenosti, tedy toho, že by výsledný tvar byl zcela uzavřený.
- **PolygonCollider2D**: Tento collider může být jakéhokoliv tvaru a je vždy zcela uzavřený.

Speciálním případem je **CompositeCollider2D**. Ten slouží ke spojení libovolného množství Box a Polygon colliderů.

Effector Effektor je komponenta měnící chování collideru přiřazenému ke stejnému hernímu objektu jako effector vůči okolnímu světu. Effectorů je několik typů, zde je pár příkladů:

- **PlatformEffector2D**: Tento effector zajistí takové chování collideru, jaké lze čekat od platformy ve 2D hrách. Lze nastavit například možnost prostupnosti platformy z jednoho směru tak, aby hlavní postava hry mohla platformu směrem zespodu proskočit a zároveň touto platformou nepropadla po dopadnutí na ni.
- **SurfaceEffector2D**: Surface effector aplikuje na všechny kolidující objekty určitou sílu, která lze specifikovat v nastavení této kompoenty. Pomocí toho lze dosáhnout například efektu jezdícího pásu.

- **PointEffector2D**: PointEffector podobně jako Surface effector aplikuje na kolidující předměty sílu. Ovšem u Point effectoru se síla odvíjí od vzdálenosti kolize a středu effectoru. Díky tomu lze snadno simulovat výbuch, gravitační pole planety a mnoho dalších věcí.

Joint Jednotlivé RigidBody2D objekty k sobě můžeme různě spojovat. Takové spojení v Unity zajišťuje komponenta **Joint**. Základní podmínkou pro spojení dvou objektů pomocí Jointu je přítomnost RigidBody2D komponenty na obou spojovaných herních objektech. Pokud RigidBody2D není na objektu přítomné, přidá se spolu s komponentou Joint automaticky. Joint lze také použít k udržení kinematického nebo dynamického objektu v určitém místě herního světa. V takovém případě k vytvoření spojení není vyžadován druhý herní objekt.

Jointům je možné nastavit i deformační sílu. To je síla, která musí na spojení působit, aby bylo prolomeno. Pokud je spojení prolomeno, objekty jsou od sebe odděleny a komponenta jointu je automaticky smazána.

Stejně jako u effectorů, existuje mnoho typů jointů. Zde jsou vyjmenované ty, které jsou použity při implementaci výsledné hry:

- **HingeJoint2D**: Hinge joint zajistí, že objekt, ke kterému náleží, je připojen k vybranému objektu nebo bodu ve světě. Okolo tohoto objektu nebo bodu se může otáčet v určitém rozsahu, který je specifikovatelný v nastavení tohoto jointu. S určitou nadsázkou lze říci, že HingeJoint slouží ke tvorbě jakýchsi kloubů.

- **DistanceJoint2D**: Tento typ jointu udržuje spojené objekty v určité vzdálenosti od sebe, jak už jeho název napovídá.

PhysicMaterial2D Ke Collider2D a RigidBody2D komponentám je ještě možné přiřadit PhysicMaterial2D. Nejedná se o komponentu, nýbrž materiál. Material v Unity definuje, jak se má povrch daného objektu renderovat. PhysicMateraal2D je speciální druh materiálu, který definuje, jak se má povrch objektu chovat z pohledu fyziky. V tomto materiálu lze nastavit dva pamametry. Prvním z nich je ***Friction***. Tento parametr nastavuje velikost tření povrchu daného objektu, tedy množství energie, které je ztraceno při pohybu po tomto objektu. Druhým parametrem je ***Bounciness***, který nastavuje jak velkou silou budou kolidující objekty od tohoto objektu odráženy, případně jak velkou silou bude tento objekt odražen od ostatních.

2.2.2 Zobrazení

Základní grafickým objektem pro 2D je **Sprite**. Sprity jsou automaticky vytvořené z obrázků, které jsou vložené do adresáře projektu, pokud je projekt nastaven v módu 2D. Při vytváření spritu lze nastavit počet pixelů rovných

jedné jednotce délky herního světa. Tím lze jednotlivé sprity zvětšovat nebo zmenšovat. Pomocí **Sprite Editoru** je možné nastavit střed spritu nebo jeden obrázek rozřezat na několik menších. To se hodí například pro zpracování Sprite sheetu.

Pokud jsou Sprity už vytvořeny, je potřeba je vykreslit. O to se stará komponenta **SpriteRenderer**. Tato komponenta má proměnnou, do které se přiřadí Sprite k vykreslení. Tuto proměnnou lze nastavovat i při běhu aplikace. Díky SpriteRendereru lze také Sprite přebarvit či mu nastavit jeho alfa kanál. Ze základu je jako materiál Spritu nastaven ***Sprite-Default***. S tímto materiálem nebude Sprite ovlivňován světelnými objekty. Pokud má Sprite reagovat na světelné objekty, je potřeba materiál ve SpriteRendereru změnit. Například na ***Sprite-Difuse***.

Particle System Komponenta **Particle System** slouží k animaci nebo reprezentaci herních objektů, které nejdou vyjádřit jednou texturou. Systém je tvořen velkým množstvím textur, které jsou společně nastaveny a tak tvoří jeden celek. Mezi takové celky patří například déšť, oheň, písečný prach, krev nebo různé efekty kouzel.

Každá částice je vytvořena ve zdroji. Jako zdroj se v částicovém systému označuje bod, případně těleso, ze kterého jsou částice vypouštěny. Základní vzhled částic lze měnit pomocí textur, nejčastěji pomocí Texture Sheetu. Komponenta částicového systému dále nabízí možnost upravovat barvu, velikost, rychlost, ale i mnohé další vlastnosti částic. Existuje zde i možnost měnit jednotlivé vlastnosti v závislosti na čase či jiném parametru.

Každý herní objekt může mít maximálně jednu komponentu Particle Effectu. Proto při vytváření složitějších objektů je potřeba tento objekt rozdělit na jednotlivé dílčí části. Například pro oheň je potřeba vytvořit tři části: plamen, jiskry a kouř.

2.3 Unity Asset Store

Jedním z mnoha důvodů, proč je Unity tak populární, je právě **Asset Store**. Jedná se o internetový obchod, ve kterém se dá sehnat snad vše, co pro tvorbu své hry potřebujete a tím ušetřit spoustu času. Můžete zde koupit různé kusy kódu, 2D grafiku všeho druhu, 3D modely postav včetně animací, kompletně celé UI nebo dokonce nová okna a rozšíření pro Unity jako takové. Tyto zakoupené assety může vlastník následně použít opakovaně ve svých hrách. Nachází se zde i nemalé množství assetů, které jsou zdarma.

Na Asset Store přidává produkty samotná firma Unity Technologies, ale také spoustu ostatních uživatelů Unity. Pro vydávání vlastní assetů na Asset Store je potřeba pouze bezplatná registrace.

V obchodě je možné nakupovat přes internetový prohlížeč, ale také přímo Asset Store okně v Unity. V případě že nákup proběhne přes internetový pro-

hlížeč, nový majitel je odkázán do svého Unity, kde pomocí pár kliků v Asset Store okně importuje nové assety přímo do otevřeného projektu.

2.4 Optimalizace

Unity poskytuje velké množství kvalitních nástrojů. Ty ovšem při špatném používání nebo náročných projektech nemusí stačit. Aby hra nakonec dosáhla dobrých výsledků a fungovala plynule, jsou často potřeba různé optimalizace. Unity poskytuje mnoho možností, jak optimalizace v jednotlivých odvětvích řešit.

2.4.1 Fyzikální optimalizace

Fyzikální výpočty mohou být značně výpočetně náročné, zvlášť pokud jsou prováděny pro velké množství objektů v poměrně krátkém časovém úseku. Unity tyto dva aspekty umožňuje z části korigovat a dosáhnout tak velkých úspor v podobě CPU času.

Fixed Timestep Tímto časem lze upravit doba mezi jednotlivými fyzikálními simulacemi. Čas je zadán v milisekundách a přesně udává jak dlouho bude fyzikální engine čekat, než znovu provede výpočty u všech simulovaných objektů. Hodnotu je nutné nastavovat s ohledem na povahu hry. Čím vyšší hodnota, tím více výkonu je ušetřeno. Ovšem simulace pak ztrácejí na přesnosti a to může být u některých her problém.

Tento čas také ovlivňuje dobu mezi jednotlivými voláními metody **FixedUpdate**, kterou je možné přepisovat v rámci **MonoBehaviour** scriptů.

Matice kolizí Každý GameObjekt ve scéně je součástí nějaké vrstvy, v základním nastavení ***Default*** vrstvy. Je možné vytvářet vrstvy nové a libovolně je u objektů měnit, podmínkou je, aby každý herní objekt měl právě jednu vrstvu. Matice kolizí pak umožňuje nastavit, které vrstvy mezi s sebou mají kolidovat.

Kapitola 3

Analýza podobných existujících her

Z obchodu Google Play byly vybrány hry dle následujících kritérií:

- **hodnocení uživatelů alespoň 4,**
- **počet instalací alespoň 100 000,**
- **2D prostředí,**
- **platformová hra,**
- **hráč ve hře nemá možnost přímého boje.**

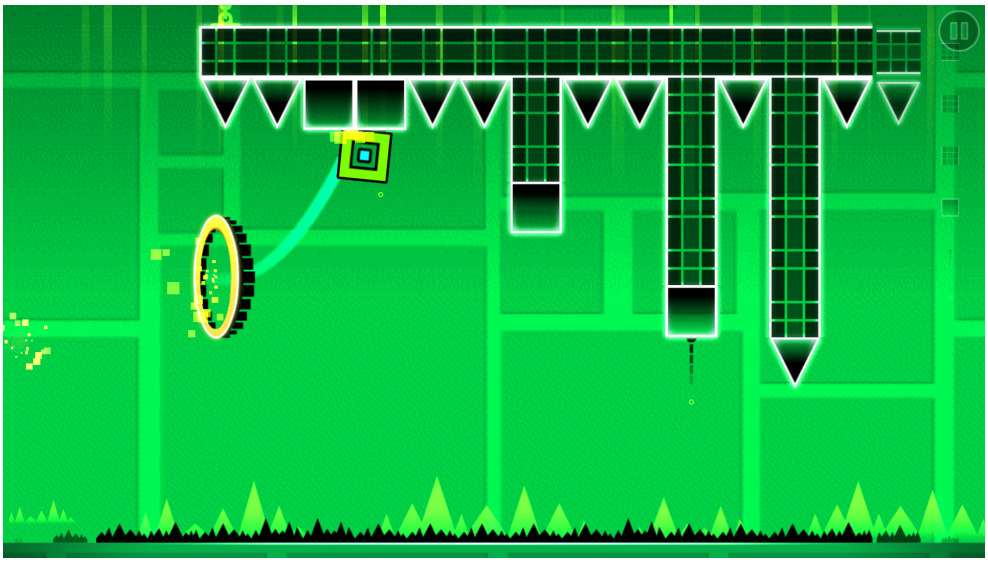
Výše zmíněná kritéria splňovalo velké množství her. Z nich byly zvoleny tři, které mají společné nebo podobné herní mechanismy s navrženou hrou. Tato kapitola se bude zabývat popisem těchto tří her a jejich následným srovnáním.

3.1 Geometry Dash [4]

Hodnocení: 4.8
Počet instalací: 1 000 000 - 5 000 000

Geometry Dash je arkádová hra od vývojáře RobTop Games. Podstatou této hry je dostat se na konec všech úrovní, kterých je už nyní velké množství a stále jsou vyvíjeny další. V této hře se hráč vžije do role kostičky a musí projet různé druhy úrovní. Zároveň s počtem úspěšně dokončených úrovní se postupně zvedá jejich obtížnost. V úrovních se vyskytují různorodé překážky, které se hráči snaží překazit průchod úrovní. Hráč zde může narazit na různé ostny, čepele, ale i obyčejný obdélník, který mu může snadno jeho cestu úrovní zarazit. Každý náraz znamená jistou smrt. Kostička má konstantní rychlost, kterou se pohybuje. Hráč má pouze jednu možnost, jak se překážkám vyhnout, a tou je skok. Pokud se hráč dotkne obrazovky zařízení, kostička vyskočí. To je jediný ovládací prvek hry.

Ve hře se mimo jiné nacházejí i modifikační portály, které různými způsoby modifikují hráčův pohyb. Hráč zde může nalézt portál měnící gravitaci (viz obr. 5.2), portál měnící vodorovný směr pohybu, portál, po jehož průchodu hráč ovládá létající loď či robota s větším doskokem a spoustu dalších portálů.



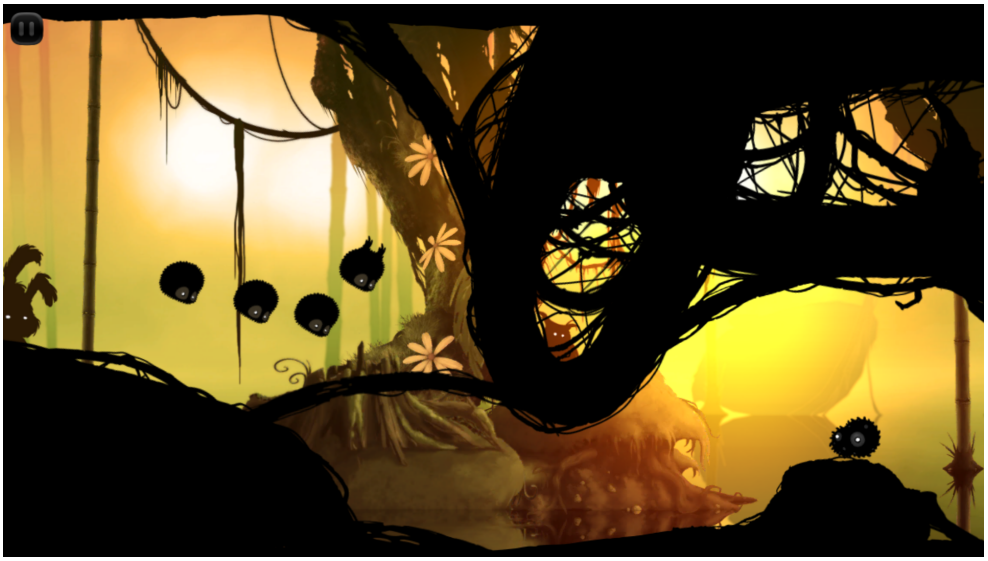
Obrázek 3.1: Portál otáčející gravitaci ze hry Geometry Dash

Hra klade důraz především na audiovizuální stránku. Graficky je hra zpracována dosti barevně, obsahuje velké množství efektů. Každá úroveň je tvořena tak, aby rytmicky seděla do svého soundtracku, který hraje v pozadí.

3.2 Badland [5]

Hodnocení: 4.5
Počet instalací: 10 000 000 - 50 000 000

Hra se jménem Badland vznikla ve studiu Frogmind. Ve hře se hráč převtělí do poletujícího ptáčka. Herním cílem je dostat postavičku z jednoho konce úrovně na ten druhý. Úrovně v Badland jsou rozdělené podle částí dne, což znamená, že některé se odehrávají ráno, jiné při západu slunce a další například v noci. Ve hře je několik takovýchto úrovní, jejich počet se pohybuje v řádu desítek. V úrovních se nachází všemožné věci, které se hráče snaží rozmáčknout, rozřezat nebo srazit. Hráči se může stát osudným i to, že nestihne držet tempo s kamerou, která jede automaticky kupředu. Tyto zmíněné věci, které zdaleka nesčítají všechny možnosti, zapříčiní hráčovu smrt. Hráč ovládá ptáčka pomocí dotyku obrazovky. Pokud hráč drží prst na obrazovce, ptáček stoupá, pokud se hráč obrazovky nedotýká, ptáček klesá. Jiný ovládací prvek zde není.



Obrázek 3.2: Naklonované hráčské postavičky a klonovací bonus nacházející se v pravém dolním rohu ze hry Badland

Ve hře se vyskytují různá zlepšení, která upraví chování hlavní postavičky. Například je zde rostlina, která z poletujícího ptáčka udělá ostnatou kuličku, která se ke všemu lepí. Dalším příkladem takového vylepšení je zrychlení /zpomalení nebo zvětšení/zmenšení postavičky. Ve hře se vyskytuje bonus, který je, dle mého úsudku, nejzajímavějším. Tento bonus zapříčiní naklonování postavičky (viz obr.5.2).

Mimo jiné hra nabízí mód pro jednoho hráče a také mód pro více hráčů. Hra pro více hráčů funguje jak lokálně, tak přes internet. Při lokální hře se obrazovka rozdělí na dvě herní plochy, přičemž každý hráč hraje na jedné z nich. Do jedné internetové hry se mohou připojit maximálně čtyři hráči.

Pro Badland je stěžejní především její audiovizuální zpracování. Popředí je vždy pouze černé, kdežto pozadí je barevné. Vše doprovází nejrůznější zvuky přírody, což vytváří příjemnou atmosféru hry.

3.3 Limbo [6]

Hodnocení: 4.7
Počet instalací: 500 000 - 1 000 000

Limbo je platformová adventura od vývojáře Playdead, která je považovaná za jednu z nejlepších svého druhu. Hra původně vyšla pro PC, PlayStation 3 a Xbox 360. Později byl rozšířen počet podporovaných platforem o iOS a Android. Hráč ovládá malého chlapce, který se probudí v Limbu, což je svět, ve kterém se hra odehrává. Limbo svým vzhledem připomíná temný les se spoustou nástrah a příšer snažících se chlapce usmrtit nebo mu alespoň znepříjemnit jeho cestu. Cílem malého mužíka je nalézt svojí sestru, která se nachází někde v Limbu. Pro hráče to znamená, že musí projít celou úroveň z levé strany na stranu pravou.

Obrázek 3.3: Scéna ze hry Limbo

Hlavní mechanikou této hry jsou hádanky, o které se celá hra opírá. Při své

cestě lesem hráč narazí na velké množství logických hádanek, které je nezbytné vyřešit pro další postup hrou. Hra má pouze jednu úroveň, ve které se nacházejí záchytné body. Tyto body obvykle bývají za složitějšími hádankami tak, aby když hráč umře, nemusel danou hádanku řešit znovu. Úroveň je rozčleněna na několik částí, které při hraní nejsou patrné. Ovšem je možné je následně nalézt v menu, tudíž není problém si vybrané části úrovně zopakovat.

Nenacházejí se zde žádná tlačítka, pomocí kterých by hráč mohl postavičku ovládat. Pohyb postavy a skok je řešen pomocí tahů po obrazovce. Jednotlivé akce, jako uchopení bedny nebo páky, se vykonávají na základě jednoduchého dotyku obrazovky.

Grafická stránka hry (viz obr.5.2) je omezena pouze na barvu bílou a černou. To utváří lehce hororovou atmosféru, která v hráči vyvolává pocity napětí, nejistoty a možná i strachu.

3.4 Shrnutí a srovnání

Všechny výše uvedené hry jsou 2D platformovky. První dvě, tedy Geometry Dash a Badland, jsou ve stylu auto-runner, což znamená, že jejich vodorovný pohyb je řízen hrou samotnou. Limbo je přesným opakem, umožňuje hráči na jednotlivých místech setrvat nebo se vracet, pokud to charakter terénu v daném místě dovolí. Po vyzkoušení jednotlivých her jsem dospěl k následujícím závěrům.

Badland a Geometry Dash se částečně shodují v nárocích na hráče. Kladou důraz především na fyzické dovednosti hráče, tedy schopnost hru kvalitně ovládat. Dle mého názoru tvůrci hry Badland uchopili tuto myšlenku o něco lépe. Obtížnost jednotlivých úrovní je adekvátní, není potřeba žádného delšího trénování k úspěšnému průchodu jednotlivých úrovní. To se o hře Geometry Dash nedá tak úplně říct. Hra sice disponuje tréninkovým módem, kdy je možné jednotlivé úrovně trénovat bez stálého vracení se na začátek úrovně po smrti, ale časová náročnost na průchod jedné úrovně je i tak poměrně velká. Po delším hraní jsem začínal být frustrovaný a hra ve mě vyvolávala především negativní pocity. Limbo také požaduje po hráči jistou fyzickou dovednost, ovšem větší nárok je kladen na hráčovu rafinovanost a logické myšlení.

Hra Limbo oproti svým dvěma konkurentům postrádá jakékoliv modifikace či bonusy, které by herní průběh měnily, což má za následek, že se hra po delší době může stát lehce monotónní.

Po grafické stránce se hry rozcházejí. Geometry Dash volí velmi barevné kombinace a spousty efektů. Limbo se omezilo pouze na černou a bílou barvu a práci se světlem. Badland je po grafické stránce někde mezi oběma hrami. Černé popředí, barevné pozadí, které je obohaceno o hříčky se světlem.

Mnou navržená hra by měla být v jistém ohledu kompromisem mezi všemi zmíněnými hrami. Hráč by měl prokázat všímavost, logické myšlení, ale i do-

vednost při ovládání hlavního hrdiny. Vše v rozumném množství, aby hra v hráči po delší době nevyvolávala negativní pocity.

Ve hře jsou navrženy herní modifikace v podobně speciálních předmětů, podobně jako u her Geometry Dash a Badland. Speciální předměty ale není vždy nutné sbírat/používat, mají zajistit především možnost různorodého průchodu úrovní. U zmíněných dvou her je většinou nutnost modifikace sebrat, pokud hráč chce úroveň dokončit.

Grafickým návrhem se nejvíce přibližuje hře Limbo, ovšem není omezen pouze na černou a bílou barvu. Nejvíce se pak vizuálně distancuje od hry Geometry Dash.

Kapitola 4

Analýza a návrh

Tato kapitola se zabývá analýzou a návrhem 2D platformové hry, jejíž základní princip je popsán v zadání práce. Hra byla pojmenována jménem **Bad Dream**.

4.1 Herní návrhový dokument

Herní návrhový dokument by měl být součástí analýzy a návrhu každé větší i menší hry.

Tento dokument má každého čtenáře naučit, jak hra funguje. K tomu je potřeba vysvětlit nejen mechaniky, ale také jak herní objekty (postavy, nepřátele, zbraně, prostředí atd.) interagují mezi sebou, o čem hra je a jak vypadá její grafická stránka[7].

4.1.1 Hlavní koncept

BadDream je platformovou adventurou z prostředí snů, částečně inspirovaná hrou Limbo. Jedná se o 2D skákačku s prvky auto-run hry, která klade důraz na zručnost a logickou zdatnost hráče.

4.1.2 Příběh

Úvodní slovo k příběhu:

„Ten zvláštní pocit…
Ten pocit, kdy chodíte v noci po pokoji a víte, že něco není úplně v pořádku. Rozhodnete se tento pocit ignorovat. Zhasnete světlo a tiše se přesunete do postele. Nad něčím přemýšlíte, ale myšlenky se pomalu rozplývají. Ovšem najednou se objeví obraz, obraz něčeho, co byste nechtěli vidět ani v nejhorších nočních můrách. Bum! Je ráno a jste vzhůru, nic se nestalo. Vstanete a pokračujete ve svém normálním denním rozvrhu. Ale zkuste se zamyslet. Co by se stalo, pokud byste se stali součástí toho hrozného snu?“

Hra umožní svému hráči prožít nejhorší noční můry hlavního bezejmenného hrdiny. Hrdina je mladý kluk, který jedné noci upadne do hlubokého temného spánku, ze kterého se není možné jednoduše probudit. Než procitne, musí čelit sérii svých nejhorších nočních můr.

4.1.3 První minuta

Nastínění průběhu prvního spuštění hry.

Po načtení úvodního loga Unity se spustí krátké úvodní video. V tomto videu je částečně vyobrazen příběh hry. Shrnutý scénář do pár bodů vypadá následovně:

- Hlavní hrdina se nachází ve světnici a kouká z okna do temného lesa.
- Hrdina jde k posteli a lehá si na ni.
- Sundavá boty, které pokládá vedle postele a následně si lehá na postel.
- Pomalu usíná. Obraz videa se začne pomalu zatemňovat.
- Obraz videa je černý.
- Zobrazení malé bílé tečky uprostřed obrazu, která se postupně zvětšuje.
- Konec videa.

Až video skončí, bude načtena úroveň s pohybovým tutoriálem, kde se hráč naučí ovládat hru. Po dokončení tutoriálu se hra přesune do hlavního menu.

4.1.4 Průchod hrou

Základní zobrazení průchodu hrou je možné vidět na obrázku 5.2. V hlavním menu má hráč možnost aplikaci vypnout nebo spustit novou hru. Při spuštění nové hry se otevře menu s výběrem úrovní. Po vybrání úrovně se načte herní scéna s vybranou úrovní. Pokud hráč umře, je možné danou úroveň hrát znovu nebo se vrátit do hlavního menu. Naopak pokud hru dokončí, načte se automaticky další úroveň. Hráč má možnost hru pozastavit. Když tak učiní, otevře se pause menu. Z tohoto menu je možné ve hře pokračovat nebo se vrátit do hlavního menu.

4.1.5 Dokončení úrovně

Pro dokončení úrovně je nutné nalézt postel, aby se hrdina mohl posunout do své další noční můry. Hráč při hledání nesmí umřít. Pokud umře, je potřeba úroveň projít znovu, případně pokračovat od posledního záchytného bodu.

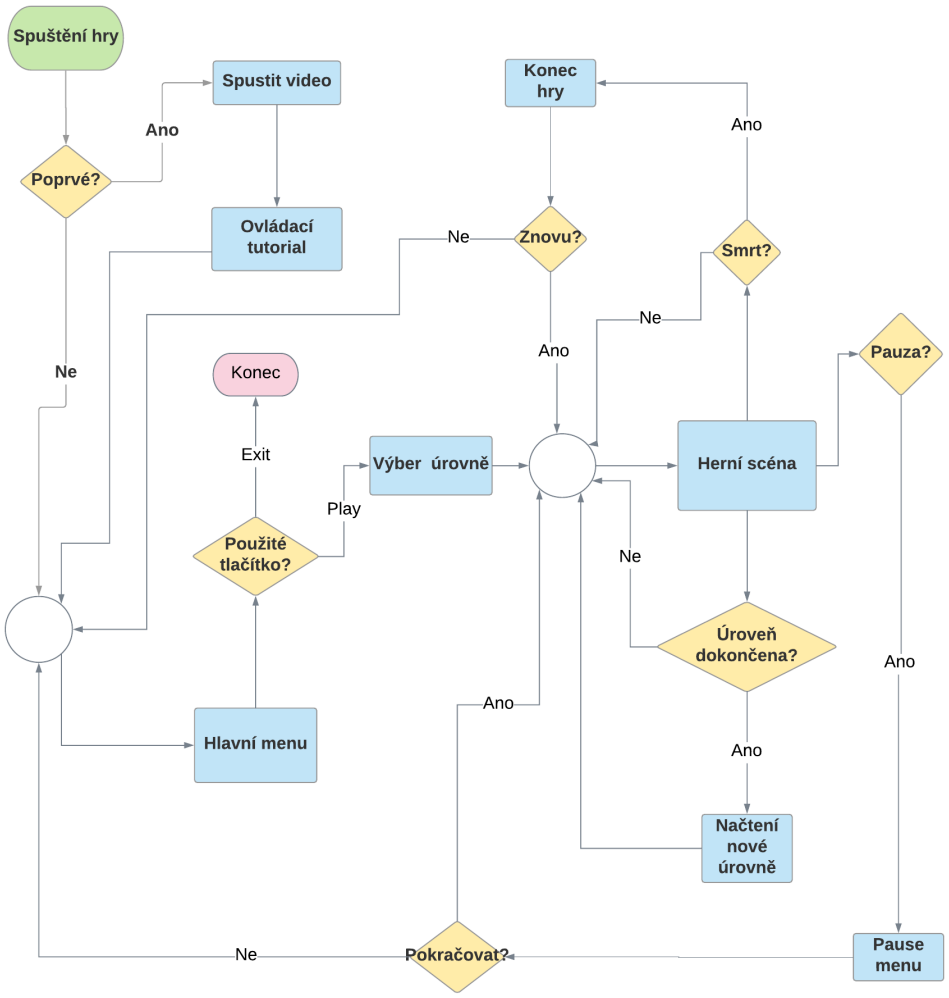
4.1.6 Počet hráčů

Jedná se o single-player hru, tudíž hru může v jeden okamžik hrát pouze jeden hráč. Hra nenabízí možnost více hráčů na jednom zařízení ani po internetu.

4.1.7 Fyzika

Fyzika je z velké části podobná té v reálném světě.

Pro hráče je 1.5x zvětšená gravitační síla. Tato úprava by měla pomoci k rychlejšímu spádu hry, především v auto-run fázích.



Obrázek 4.1: Vyobrazení základního průchodu hrou

4.1.8 Herní objekty

Objekty ve hře se dělí na tři základní skupiny:

- **Objekty vizuální** - Vizuální objekty žádným způsobem neovlivňují hlavního hrdinu a pro hráče neexistuje možnost s nimi interagovat. V úrovni jsou pouze proto, aby utvářely její vizuální stránku.
- **Objekty prostředí** - Hráč nemá možnost s těmito objekty hýbat nebo jakkoliv s nimi interagovat. Tyto objekty mohou být statické (stromy, skály, povrch…) nebo dynamické (pohyblivé platformy, výtahy…).
- **Interaktivní objekty** - S těmito objekty může hráč provádět jednotlivé akce. Může se jednat o různé páky, vypínače, vozíky, bedny atd.

  Interaktivní objekty jsou snadno rozšiřitelné, tedy přidat nový interaktivní objekt by mělo být možné bez většího zásahu do kódu.

4.1.9 Pohyb

Hráč se může pohybovat horizontálně i vertikálně. Vertikálního pohybu hráč nabývá pomocí skoku nebo pomocí speciálních předmětů nebo bonusů.

Hráč se může dostat do dvou fází:

- **Normální fáze (Normal phase)** - Hráč má nad horizontálním i vertikálním pohybem plnou kontrolu. Může se zdržovat na jednom místě jak dlouho uzná za vhodné a vracet se na již prošlá místa, pokud to povaha terénu dovoluje. Pokud se hráč nedotýká země, není možné měnit směr jeho pohybu.
- **Utíkající fáze (Running phase)** - Hráč má kontrolu pouze nad vertikálním pohybem, horizontálního pohybu hráč nabývá automaticky, lze ho ovlivnit pouze speciálními předměty a bonusy.

V obou fázích má hráč možnost dvojitého skoku (hráč může vyskočit jednou, ve vzduchu má poté možnost použít skok znovu). Vertikální síla druhého skoku bude o 25% menší než vertikální síla skoku prvního. Pokud oba skoky pomyslně rozdělíme na dvě fáze, tedy fázi stoupání a klesání, pak fáze stoupání by měla trvat delší časový úsek než fáze klesání. Tento prvek pomůže k lepší ovladatelnosti hlavního hrdiny ve vzduchu.

4.1.10 Bonusy

Po herním světě se nacházejí bonusy, které je možné sbírat. Pokud tento bonus hráč sebere, nabije se mu jeho schopnost daným bonusem. Schopnost může být v jeden okamžik nabitá pouze jedním bonusem. V případě, že hráč sebere bonus a schopnost je již nabita, starý bonus zanikne a přepíše se bonusem novým.

Gravitační káča Bonus otáčející gravitaci. Po sebrání tohoto bonusu má hráč schopnost otočit gravitaci. Modifikovaná gravitace ovšem působí pouze na hráče, nikoliv na okolní objekty.

Magnetické tenisky Sebrání tohoto bonusu propůjčí hráčovi schopnost okamžitého nabití vertikálního pohybu ve směru aktuální gravitační síly.

4.1.11 Ovládání

Hra má několik ovládacích prvků. Prvním z nich je joystick, kterým se ovládá horizontální pohyb hlavního hrdiny. Dalším ovládacím prvkem jsou dotyky. Typy dotyků jsou následující:

- **Tap** - dotyk obrazovky
- **Swipe up** - potáhnutí po obrazovce směrem nahoru
- **Swipe down** - potáhnutí po obrazovce směrem dolů
- **Swipe left** - potáhnutí po obrazovce směrem vlevo
- **Swipe right** - potáhnutí po obrazovce směrem vpravo

Jednotlivé dotyky se navíc mohou lišit na základě pozice na obrazovce v jednotlivých fázích.

Normální fáze: V normální fázi je obrazovka rozdělena na dvě poloviny: pravou a levou. V dolním levém rohu obrazovky se nachází joystick. Pravá strana slouží k ovládání pomocí dotyků. Dotyky jsou vnímány pouze na pravé straně obrazovky. Tento prvek má zamezit nechtěným dotykům okolo joysticku.

Utíkající fáze: Joystick v této fázi není dostupný. Dotyky jsou vnímány v rámci celé obrazovky.

V obou fázích je dostupné tlačítko schopností. Tímto tlačítkem se aktivuje aktuální bonus. Tlačítko se nachází v pravém dolním rohu.

4.1.12 Popis uživatelského rozhraní

Hlavní menu: Hlavní menu je zobrazené jako první po startu hry. Obsahuje pět tlačítek:

- **Play** - Pomocí tohoto tlačítka se přejde do menu s výběrem úrovní.
- **Exit** - Tímto tlačítkem se opouští hra.

- **Movement** - Tlačítko, kterým je možné spustit úroveň s pohybovým tutoriálem.
- **Credits** - Ukáže seznam lidí, kteří měli zásluhy na tvorbě této hry.
- **Bug report** - Otevírá dialog, pomocí kterého je možné odeslat vývojáři zprávu o chybě ve hře.

Výběr úrovní: Úrovně jsou zobrazené pomocí panelů. Každý panel obsahuje jméno úrovně, její ilustrační obrázek a informaci o tom, zda je úroveň již otevřena. Panely jsou uspořádány vedle sebe a je možné mezi nimi procházet pomocí tahů. Dotykem na konkrétní panel se spustí úroveň, která je na tomto panelu vyobrazena. V dolním pravém rohu se nachází tlačítko, které slouží k návratu do hlavního menu hry.

Hra Uživatelské rozhraní ve hře bylo již částečně popsáno v části „Ovládání“. Nachází se zde joystick, tlačítko pro aktivaci nabitého bonusu a tlačítko pro pozastavení hry (otevření pause menu).

Pause menu Toto menu se otevře po použití pauzovacího tlačítka ve hře. Nacházejí se zde tři tlačítka:

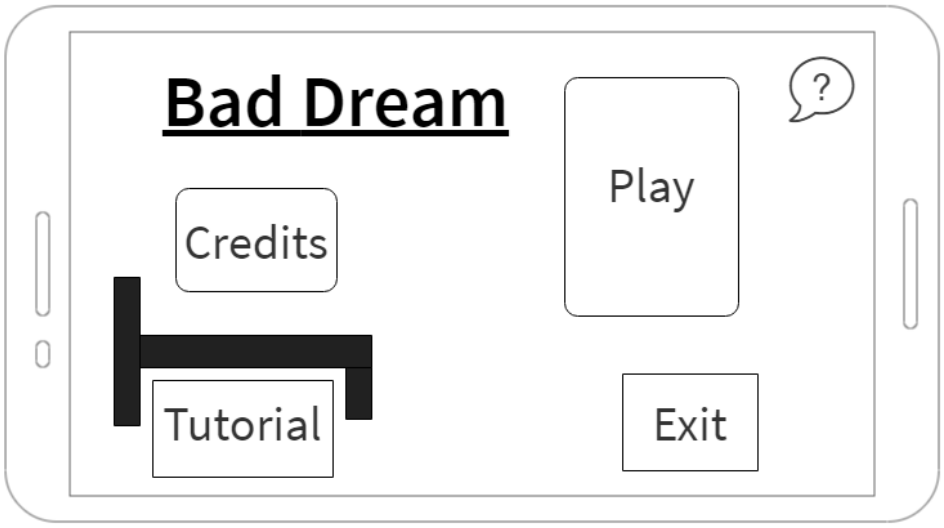
- **Menu** - Po zmáčknutí tlačítka se hra vrátí do hlavního menu.
- **Continue** - Tímto tlačítkem se hra znovu spustí a hráč může pokračovat ve hraní.
- **Restart** - Tlačítko sloužící k restartování aktuální úrovně.

Game Over Obrazovka otevírající se vždy po úmrtí hlavního hrdiny. Obsahuje tři tlačítka:

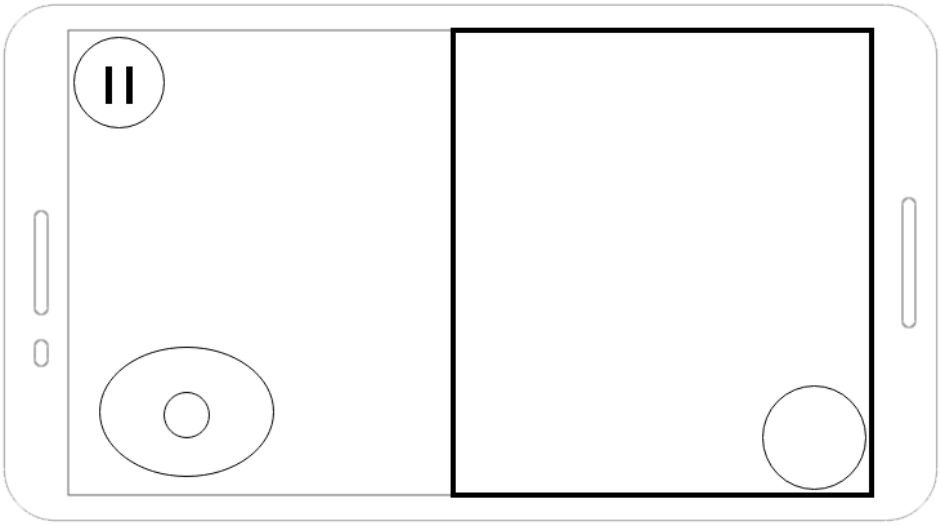
- **Menu** - Tlačítko sloužící pro návrat do hlavního menu hry.
- **Restart** - Úplné restartování úrovně, hráč tímto přichází o případné záchytné body.
- **Continue** - Po použití se hra vrátí k poslednímu záchytnému bodu. Pokud aktuálně hráč nemá aktivovaný žádný záchytný bod, tlačítko není možné použít.

Bug report Okno pro odesílání nalezených chyb vývojáři. Okno obsahuje kolonku pro vyplnění emailu odesílajícího, kolonku pro vyplnění popisu chyby a tlačítko pro odeslání.

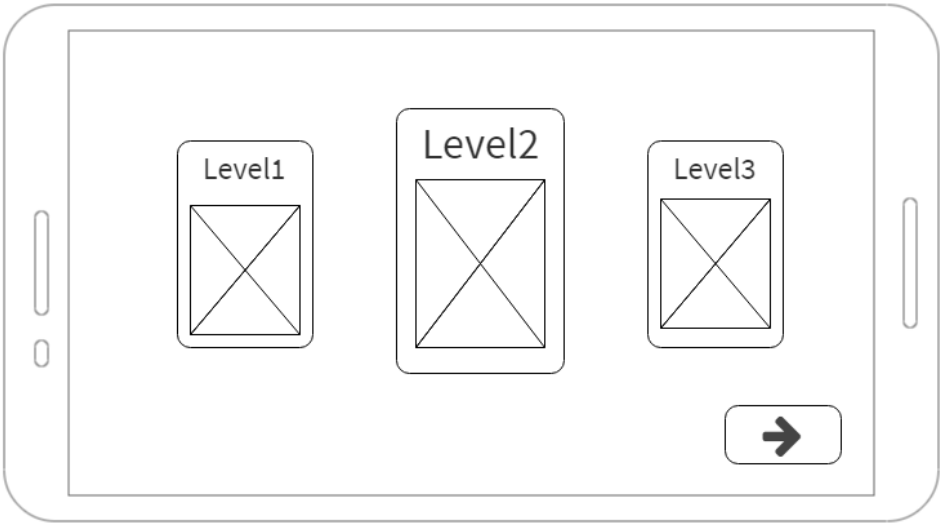
4.1.13 Grafický návrh uživatelského rozhraní



Obrázek 4.2: Grafický návrh hlavního menu



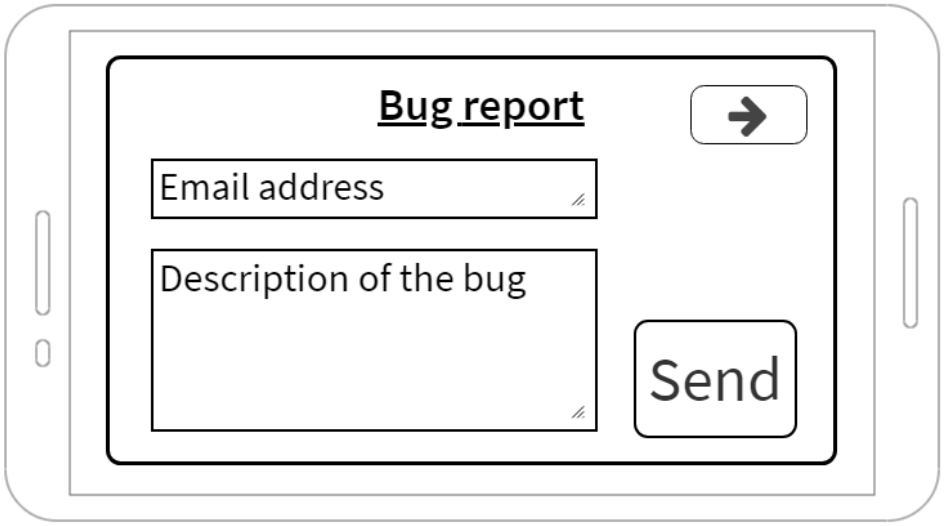
Obrázek 4.3: Grafický návrh uživatelského rozhraní v herní scéně



Obrázek 4.4: Grafický návrh výběru úrovní

Obrázek 4.5: Grafický návrh pause menu

Obrázek 4.6: Grafický návrh GameOver obrazovky



Obrázek 4.7: Grafický návrh bug reportu

4.2 Funkční a nefunkční požadavky

Funkční požadavky Funkčních požadavků je velké množství. Vycházejí z herního návrhového dokumentu. Některé z nich jsou zde pro příklad zmíněny.

- **Každou úroveň je možné restartovat.**
- **Z hlavního menu je možné kontaktovat vývojáře.**
- **Postava hlavního hrdiny se může hýbat.**
- **Při prvním spuštění hry se automaticky spustí tutoriál.**

Nefunkční požadavky

- **N1 FPS** - Frame rate musí být minimálně 25 FPS na průměrném zařízení.
- **N2 Platforma Android** - Hra musí fungovat na operačním systému Android minimální verze 4.0.
- **N3 Rozšiřitelnost** - Hra musí být snadno rozšiřitelná, především v oblasti bonusů a interaktivních objektů.
- **N4 Jednoduchost a intuitivnost ovládání** - Ovládání by mělo být takové, aby s ním hráč neměl problémy a rychle si ho osvojil.

4.3 Možnost editoru úrovní

Hlavním typem mapy pro 2D, kterou lze vytvořit pomocí editoru, je Tile mapa. Tile mapa je označení pro takovou mapu, která je tvořená z dlaždic. Unity v základní verzi nenabízelo možnost tvorby Tile map. Existuje ale mnoho doplňků, které tento problém řeší. Nejpopulárnějším z nich je doplněk Tiled2Unity [8], který zprostředkuje nahrání Tile mapy vytvořené v programu Tiled [9] do Unity. Tento postup je účinný při tvorbě jednoduchých map. Ovšem tvorba složitějších věcí, jako tvorba interaktivních dlaždic, přináší spoustu komplikací.

Tvůrci Unity si problému ohledně Tile map všimli a souběžně s tvorbou této práce vznikl nový nástroj pro tvorbu Tile map integrovaný přímo v Unity [10]. Práce s tímto nástrojem je snazší a rychlejší v mnoha ohledech. Nařezání textur nástroj provádí automaticky a poskytuje možnost přidávání kinematických i dynamických objektů. Tento nástroj by byl pro tvorbu map v navrhované hře ideální. Bohužel při vzniku prvního návrhu hry tento integrovaný nástroj ještě neexistoval.

Tvorba Tile map pomocí dříve dostupných nástrojů by byla problematická, protože nástroje neposkytovaly všechny potřebné funkcionality. Pomocí těchto nástrojů by se částečně usnadnila pouze tvorba terénu, ovšem k tomu by musely být všechny textury manuálně rozřezány do dlaždic. Jelikož téměř každá úroveň má jiný povrch, jednalo by se o rozřezání mnoha textur. Tím by se tvorba map spíše zkomplikovala. V úrovních je navíc mnoho kinematických a dynamických objektů, které by musely být stejně přidány manuálně.

Z výše uvedených důvodu byla zvolena cesta manuální tvorby jednotlivých úrovní přímo v Unity. Při tvorbě byl použit doplněk Ferr2D, pomocí kterého jsou vytvořeny všechny povrchy úrovní. Tento doplněk částečně zastoupil editor úrovní. Pomocí Ferr2D je možné tvořit pouze povrch úrovně, ovšem pro tvorbu povrchu není potřeba rozřezávat textury do dlaždic. Ostatní objekty jako páky, bedny, výtahy nebo různé dekorace jsou do úrovní přidány manuálně.

4.4 Systém záchytných bodů

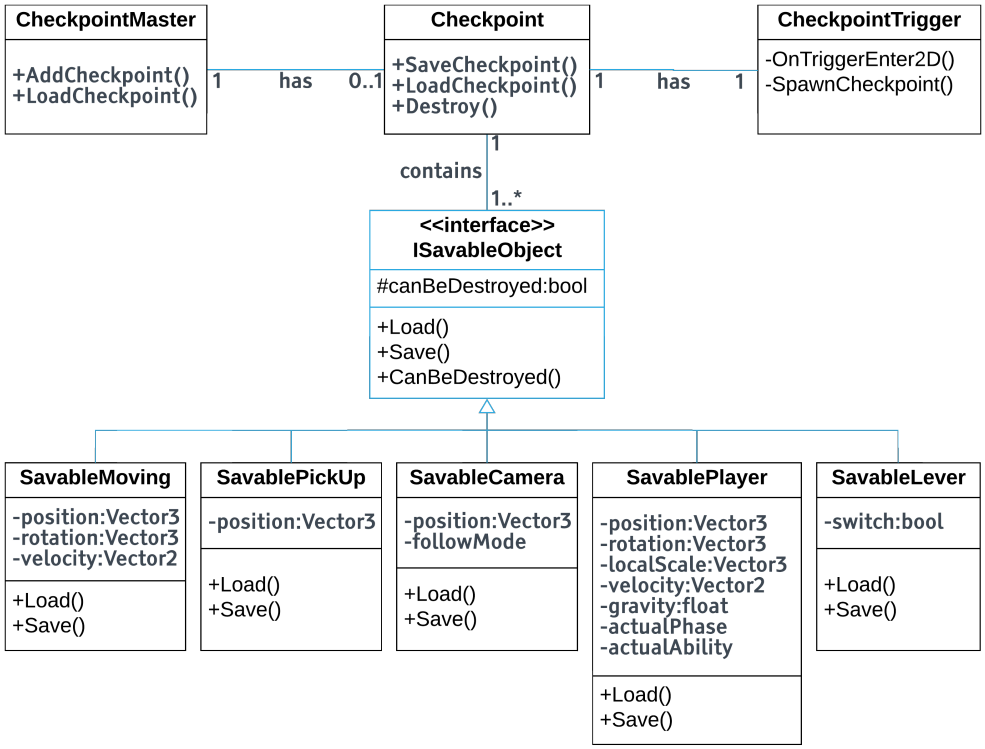
V platformových hrách se hráč obvykle potýká s řadou neúspěchů, především u her, které cílí na hráčovu zručnost. V navrhované hře hráč řeší logické hádanky. Pokud hráč hádanku vyřeší a následně umře, měl by se vrátit zpět na začátek úrovně a celý dosavadní průchod opakovat, včetně vyřešených hádanek. Tento přístup by mohl po čase v hráči vyvolávat negativní pocity, proto je ve hře navržen.

Systém má následující vlastnosti:

- Záchytné body lze přidávat na libovolné místo v úrovni.
- Počet záchytných bodů je neomezený.
- Záchytné body je možné přidávat v průběhu hry.
- V jeden moment může být aktivovaný maximálně jeden záchytný bod.
- Pokud hráč využije záchytný bod, vždy je použit ten, který byl aktivovaný jako poslední.
- Záchytné body nemusejí být nutně využity.
- Při restartování úrovně hráč přichází o své záchytné body.

Systém je řízen objektem **CheckpointMaster**. Tento objekt zprostředkovává přidávání a načítání jednotlivých záchytných bodů. Přidání bodů vyvolá aktivace triggeru. Není potřeba ukládat všechny objekty.

Ve hře se vyskytují objekty různé povahy, například dekorativní objekty se nijak nehýbou ani nereagují s hráčem, tudíž by bylo zbytečné držet jakékoliv informace o těchto objektech. Například pro obnovení bedny je nutné znát



Obrázek 4.8: Návrh záchytného systému

pouze její předešlou pozici, rychlost a předešlé natočení, s jakými se pohybovala v okamžiku uložení. Naopak pro obnovení hlavního hrdiny je potřeba více informací. Z tohoto důvodu je nezbytné odlišit jednotlivé objekty, které mají být ukládány.

Pro tento účel je navrženo několik tříd, mezi které se dají ukládané objekty rozdělit. Tyto třídy mají společné rozhraní **ISavableObject**(viz obr. 5.2). Každý objekt, který má být ukládán, musí dědit alespoň od jedné z těchto tříd. Pokud objektu žádná z již existujících tříd nevyhovuje, je nutné vytvořit třídu novou.

Kapitola 5

Implementace

5.1 Použité technologie

Přestože Unity v základní verzi poskytuje dostatečné funkcionality pro tvorbu navržené hry, vývoj hry by zabral mnoho času. Z tohoto důvodu byly využity následující doplňky, které byly staženy z Unity Asset Store.

5.1.1 Anima2D

Anima2D je volně dostupný nástroj pro 2D animace [11]. Za pomocí tohoto nástroje je možné vytvářet a animovat objekty přímo v Unity editoru. Anima2D přináší možnost skeletálních animací [12], včetně využití inverzní kinematiky a mnoho dalších užitečných animačních technik. Díky tomu byla výrazně zrychlena tvorba animací a zároveň bylo možné realizovat věrohodnou interakci s ostatními objekty za pomocí inverzní kinematiky [13].

5.1.2 Ferr2D

Ferr2D je placený doplněk přinášející snadnou tvorbu terénu. S tímto doplňkem není nutné ručně natahovat do scény mnoho textur pro terén, srovnávat je do mřížky a následně jim ručně přiřazovat kolizní komponenty. S Ferr2D stačí vybrat materiál a nakreslit obrys terénu, který má být vytvořen [14]. Doplněk poté sám vytvoří příslušné textury podle umístění a úhlu, ve kterém se daná textura má nacházet. Pokud je například v nakresleném obrysu ostrý zlom, je přidána textura rohu. Kolizní komponenty jsou také přidány automaticky. Součástí tohoto doplňku je několik textur terénů, které byly modifikovány a použity pro vývoj navrhované hry.

5.1.3 2D Rope Editor

2D Rope Editor je též placený doplněk sloužící k modelování řetězů a provazů [15]. Pro tvorbu dynamického provazu v Unity je potřeba provaz rozdělit do několika segmentů. Tyto segmenty je pak nutné seřadit a srovnat za sebe tak, aby vytvářely provaz. Následně je potřeba segmenty postupně spojit jointy a přidat případně kolizní komponenty. Tvorba jednoho provazu by tak zabrala spoustu času. Rope Editor je postaven na stejném principu jako Ferr2D. Stačí přiřadit textury segmentů a nakreslit čáru, kterou má provaz kopírovat. Editor pak vygeneruje po této čáře provaz z přiřazených segmentů a všechny potřebné komponenty automaticky nastaví a přiřadí.

5.2 Pohyb

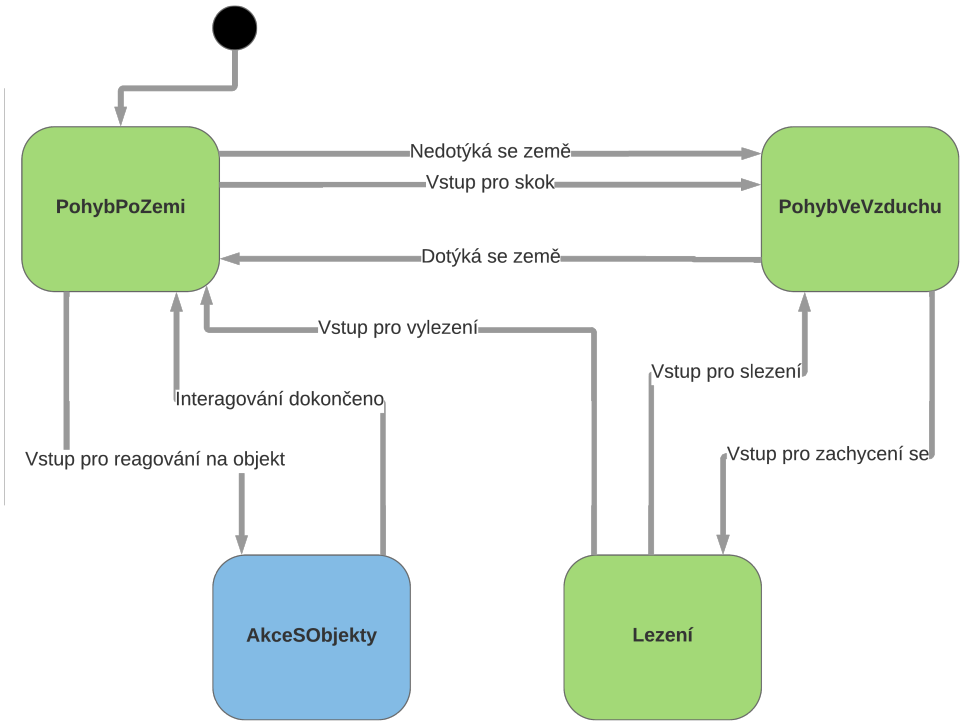
Hlavní hrdina se může dostat do dvou fází. Tyto fáze jsou dále rozděleny do jednotlivých stavů. Stavy i fáze jsou řešeny pomocí návrhového vzoru **State**.

Návrhový vzor State spočívá ve vytvoření stavového automatu, kde jednotlivé stavy jsou úkony daného objektu a přechody mohou být například hráčův vstup [16]. Výhodou tohoto návrhového vzoru je především přehlednost kódu a snadné hledání chyb. Díky tomu, že jsou jednotlivé úkony odděleny do samostatných stavů lze snadno deklarovat, které akce půjdou v daném stavu provádět. Zároveň je snadné při hledání chyb určit část kódu, která je případně nefunkční. Objekt je zásluhou tohoto vzoru snadno rozšiřitelný o další úkony, jelikož stačí pouze vytvořit další stav a přidat pro něj přechody z příslušných stavů.

V kontextu hlavního hrdiny Bad Dream jsou stavy normální fáze například pohyb po zemi, pohyb ve vzduchu, akce s objekty a lezení. Na obrázku 5.2 lze vidět příklad jednotlivých přechodů. Stavy mají společné rozhraní **ObjectState**. Každý stav musí implementovat následující metody:

- **Enter** - Metoda se volá při vstupu do stavu.
- **HandleInput** - V této metodě se definují veškeré reakce daného objektu na hráčův vstup v rámci stavu.
- **Update** - Metoda pro aktualizování herního vývoje daného objektu v rámci aktuálního stavu.
- **EarlyUpdate** - Stejná metoda jako metoda Update. Rozdíl mezi nimi je ten, že je volána před HandleInput i Update.

Momentální fáze si pamatuje aktuální stav a volá tyto metody v určitém pořadí každý herní tik. Při vstupu do stavu se volá metoda Enter. Následně se opakovaně volají metody HandleInput, EarlyUpdate a Update přesně v tomto pořadí.



Obrázek 5.1: Příklad stavů hlavního hrdiny v normální fázi

5.3 Schopnosti

Hráč je po sebrání bonusu obdařen určitou schopností. Schopnosti jsou realizovány pomocí návrhového vzoru **SubClass Sandbox**. Tento návrhový vzor je velice jednoduchý. Spočívá v tom, že existuje jedna hlavní třída, která definuje metody a případně pár užitečných operací, které mohou odvozené třídy využívat. Každá odvozená třída implementuje tyto metody [16].

Ve hře Bad Dream je hlavní třídou třída SuperAbility, která definuje metody Activate, Update a IsFinished. Od ní je odvozená například třída GravitySwap. V této třídě je přepsána metoda Activate. V metodě Activate se definuje chování, které je vyžadované při aktivování schopnosti. Při aktivování je schopnost přidána do seznamu aktivních schopností. Každý herní tik je vykonávaná metoda Update všech schopností, které jsou v seznamu aktivovaných. Aktualizování schopností každý tik je nutné kvůli plynulým přechodům a časovačům, například u schopnosti SpeedBoots se rychlost zvedá postupně až do doby, kdy je dosažena maximální rychlost. Poté, co je schopnost dokončena, je odebrána ze seznamu aktivních schopností a smazána. Informaci o dokončení schopnosti poskytuje metoda IsFinished.

5.4 Akce s objekty

Jedním ze základních požadavků na hru byla snadná rozšiřitelnost, především v oblasti interaktivních herních objektů. Za tímto záměrem je vybudován systém komunikace mezi interaktivními objekty a hlavní postavou.

Když postava narazí na interaktivní objekt, dostane pouze informaci, že se v okolí nachází nějaký herní objekt, se kterým je možné interagovat. Pokud se hráč rozhodne na akci reagovat, hlavní hrdina vyjádří danému objektu zájem o akci. Objekt následně hlavnímu hrdinovi poskytuje informace o dané akci a veškeré úkony jsou řešeny pomocí interaktivního objektu.

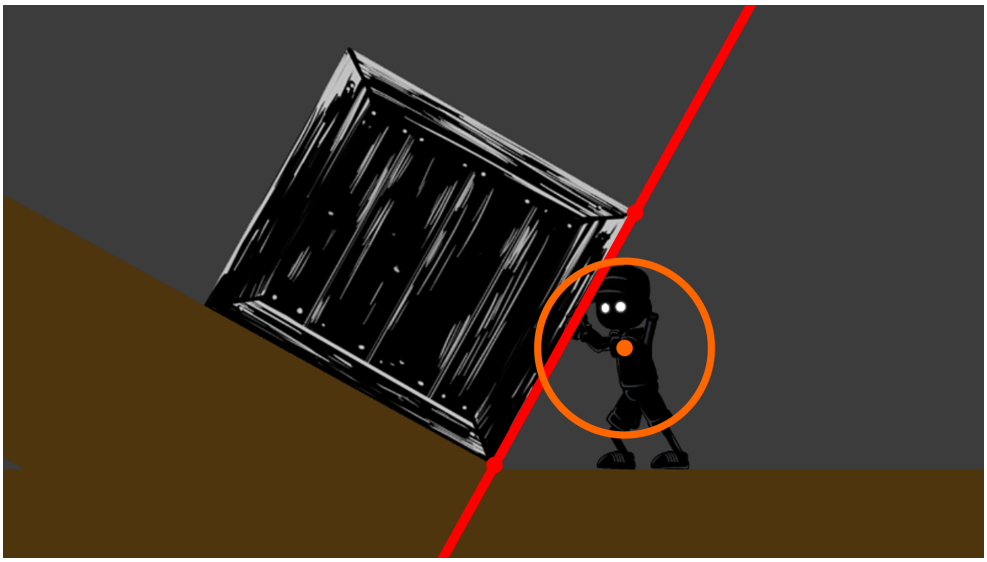
Implementačně je toto chování zajištěno v Action stavu hlavního hrdiny. Pokud je v okolí objekt, na který hráč reaguje, přejde do Action stavu. V tomto stavu si hráč drží odkaz na objekt, se kterým interaguje. Každý takový objekt implementuje rozhraní ActionObject, které definuje metody Enter, Update a HandleUpdate. Následně stavové metody Enter, Update a HandleUpdate volají stejnojmenné metody objektu, se kterým hráč interaguje.

Takové chování zajišťuje právě snadnou a rychlou rozšiřitelnost interaktivních objektů. Při přidání nového objektu do hry stačí takový objekt vytvořit a definovat přímo v něm hráčovo chování vzhledem k tomuto objektu. Není potřeba zasahovat jakýmkoliv způsobem do chování hráče.

5.5 Tlačení objektů

Po fyzikální stránce je tlačení objektů v Unity velice jednoduchá věc. Stačí přidat objektům patřičné komponenty, nastavit hmotnosti, dostatečnou sílu hráči a je hotovo. Ovšem vyskytl se problém, se kterým nebylo při návrhu počítáno. Tím problémem byla pozice rukou. Dokud byl objekt tlačen po rovné ploše, bylo vše v pořádku. Ale pokud byl povrch nerovný a objekt se natáčel, pozice rukou byla nesprávná a po grafické stránce tento jev nevypadal dobře. Do tohoto okamžiku byly animace řešeny pomocí sprite sheetů, které byly vytvářeny v externím programu. Kvůli tomuto problému bylo nutné přejít na skeletální animace s podporou inverzní kinematiky. Za tímto účelem byl využit doplněk Anima2D, jak už bylo řečeno.

Pozice rukou je počítána na základě průniku kružnice s přímkou. Jak je vidět na obrázku 5.2, rovnice přímky je vytvořena pomocí dvou krajních bodů objektu. Kružnice má střed v ramenou hlavního hrdiny a poloměr je rovný délce rukou. Na základě výpočtu se problém rozpadne na několik případů podle počtu a pozice průsečíků. Po vyhodnocení těchto případů jsou ruce nastaveny tak, aby dlaně byly na správném místě. K tomu jsou využity nástroje Anima2D poskytující inverzní kinematiku. Tento výpočet je opakován každý tik hry. Díky tomu se ruce hýbou s pohybem objektu a tlačení tak vypadá přirozeně.



Obrázek 5.2: Ilustrace průniku přímky a kružnice při tlačení bedny

Kapitola 6

Testování a nasazení

Hra má projít testováním a má být vydána na obchod Google Play ve fázi beta, jak je uvedeno v zadání.

6.1 Google Play

Aby bylo možné hru na obchod Google Play vydat, je nutné získat vývojářský účet. Vývojářský účet lze získat povýšením běžného google účtu. Toto povýšení je provedeno na základě odsouhlasení distribuční smlouvy pro vývojáře, zaplacení registračního poplatku ve výši 25$ a vyplnění osobních údajů. Díky vývojářského účtu je možné používat službu Google Play Console, která slouží k publikování a správě aplikací na obchodě Google Play [17].

Google Play Console poskytuje velké množství nástrojů, které usnadňují vydání a následnou správu aplikace. Mimo jiné se zde nacházejí i nástroje pro testování. Jsou zde tři možnosti testování.

První z nich je interní test. Interní test je nejnižším stupněm testování. Toto testování probíhá uzavřeně a je možné v něm mít maximálně 100 testerů. Velkou výhodou tohoto testování je rychlá distribuce testované verze mezi testery.

Druhou možností je alfa test, třetí pak beta test. Mezi alfa a beta testem nebyl velký rozdíl. Testování u obou druhů mohlo probíhat uzavřeně i otevřeně. Od roku 2018 je nově otevřené testování dostupné pouze u verze beta. U tohoto testování je možnost nastavení maximálního počtu testerů, ten však musí být minimálně 1000 [18].

Pokud je testování uzavřené, je potřeba ručně zadat emaily jednotlivých testerů. Google Play Console následně vygeneruje odkaz, který je potřeba všem testerům odeslat. Tester v něm je obeznámen s riziky testování a následně je požádán o vyslovení souhlasu s testováním hry. S vysloveným souhlasem tester obdrží i přístup k testované hře přes obchod Google Play.

6.2 Interní test

Jako první typ testování byl zvolen interní test s pěti testery. Tento typ testování byl vybrán na základě rychlé distribuce aktuální testované verze. První verze hry obsahovaly spoustu chyb, přičemž některé chyby byly kritické pro běh hry. Bylo nutné tyto chyby hned po objevení opravit a rychle doručit novou verzi testerům, aby bylo možné v testování pokračovat. Verze interního testu nesou označení 1.1.X.

6.2.1 Verze 1.1.1

Jednalo se o první verzi pro interní test. Tato verze obsahovala pouze hlavní menu bez animací, jednu úroveň a pause menu.

Chyby a připomínky:

- Joystick je moc malý.
- Při dotyku obrazovky v pause menu projde dotyk do samotné hry.
- Kláda na vyvýšené platformě nejde odtlačit.
- Některé louče občas nehoří.

6.2.2 Verze 1.1.2

Ve druhé verzi interního testu byl zvětšen joystick a byly opraveny chyby předešlé verze. Také byly přidány animace hlavního menu.

Chyby a připomínky:

- Hlavní menu není vidět celé (problém u některých rozlišení).
- Pokud je kláda v první úrovni odtlačena na pravou stanu, nelze pak vylézt ze skryté místnosti.
- Běh hráče je moc pomalý.
- Pokud hlavní hrdina proskočí loučí, hra zamrzne.

6.2.3 Verze 1.1.3

Do třetí verze interního testu byla přidána další úroveň a bylo přidáno menu pro výběr úrovní a menu, které se ukazuje po smrti hlavního hrdiny. Pohyb hlavního hrdiny byl zrychlen a chyby předešlé verze byly opraveny.

Chyby a připomínky:

- V menu výběru úrovní jsou jednotlivé panely daleko od sebe.
- Přechod kamery mezi jednotlivými fázemi není plynulý.
- Provazový most se při interakci s hráčem chová nepřirozeně.
- Kamera se v utíkající fázi po smrti hráče nezastaví.
- Po dokončení druhé úrovně se druhá úroveň zamkne.

6.2.4 Verze 1.1.4

Poslední verze interního testu, kde byly opraveny všechny chyby, které byly nahlášeny v předchozí verzi. Byl rozšířen počet podporovaných rozlišení, terén druhé úrovně byl kompletně předělán a byl přidán tutoriál pohybu.

Chyby a připomínky:

- V druhé úrovni chybí trigger, který by ukončil utíkající fázi.
- Smrtící mlha se v některých místech vykresluje před objekty.
- Při specifickém průchodu tutoriálem se může hlavní hrdina zaseknout v terénu.

6.3 Otevřený beta test

Když byla hra stabilní a obsahovala důležité funkcionality, byla posunuta do otevřeného beta testu. Od této chvíle je hra dostupná pro veřejnost ve fázi beta pod jménem **Bad Dream**. Verze beta testu nesou označení 1.2.X.

6.3.1 Verze 1.2.1

Jedná se o první verzi otevřeného beta testu. V této verzi byly opraveny všechny chyby z poslední verze interního testování. Také byla přidána třetí úroveň a bonusy, které lze sbírat.

Chyby a připomínky:

- Pohyb hlavního hrdiny je stále příliš pomalý.
- Pokud hráč chce vyskočit na místo a stojí přímo pod ním, není možné na toto místo vyskočit.
- Pokud zařízení není připojeno k Internetu, nevyskočí žádná hláška o neodeslání zpětné vazby.

- Některé bonusy po sebrání nemizí.
- Mezi hlavním hrdinou a pohyblivými platformami ve třetí úrovni není žádné tření, takže hlavní hrdina padá pokaždé, když se platforma rozjede.

6.3.2 Verze 1.2.2

V této verzi byla přidána možnost ovládat hlavního hrdinu ve vzduchu, animační efekty pro bonusy a checkpoint systém. Zároveň byly opraveny chyby z předchozí verze.

Chyby a připomínky:

- Ovládání není intuitivní.
- Při načtení záchytného bodu se neobnoví sebrané bonusy.

6.3.3 Verze 1.2.3

Jedná se o poslední a aktuální verzi otevřeného beta testu. V této verzi byl opraven problém s obnovou bonusů v rámci systému záchytných bodů. Také bylo provedeno přemapování dotykového ovládání pro snazší osvojení ovládání.

6.4 Shrnutí testování a zpětné vazby

V interním testu byly nalezeny především chyby ve funkcionalitách hry, jelikož se na testování podíleli hlavně lidé, kteří s vývojem a testováním her mají nějaké zkušenosti. S přechodem do otevřeného beta testu se hra dostala i mezi uživatele, kteří jsou pouze konzumenty her a zpravidla nemají žádné zkušenosti s testováním nebo vývojem her. Díky tomu se soustředili více na vizuální problémy a problémy spojené s ovládáním a pocitu ze hry. Toto testování ukázalo, že navržené ovládání je příliš složité. Tím pádem neodpovídalo požadavkům na hru a bylo nutné ho přepracovat. Ovládání bylo jednou předěláno a následně přemapováno. Tato změna má velmi kladnou zpětnou vazbu, ovšem stále se k ovládání objevují připomínky a návrhy na jeho předělání.

Závěr

Hlavním cílem této bakalářské práce bylo navrhnout, implementovat, otestovat a vydat 2D platformovou hru ve fázi beta na obchod Google Play. Hra měla být vyvíjena v herním enginu Unity3D. Součástí práce bylo také prostudovat tento herní engine, osvojit si práci s ním a prověřit možnost využití editoru pro jednotlivé úrovně.

K prostudování Unity enginu byly nejvíce využity jeho manuálové stránky a oficiální videa. Unity v základní verzi stačilo téměř na všechny funkcionality hry. Byly využity pouze tři externí doplňky, které byly použity pro tvorbu terénu, provazů a animací. Následně pak byly prostudovány hry s podobnými herními mechanikami. Na základě počáteční vize a srovnání těchto her byla navržena nová hra Bad Dream. Editor úrovní nebyl shledán jako přínosný a tím pádem nebyl použit při tvorbě úrovní. Vzhledem k dostupným nástrojům a povaze této hry by použití editoru úrovní nepřineslo žádnou větší úsporu práce, spíše naopak. Implementace se obešla bez větších potíží. Jediný problém nastal při implementaci tlačení objektů, kde bylo potřeba vyřešit umístění rukou vzhledem k tlačenému objektu. Souběžně s implementací byla hra nahrána na obchod Google Play, kde byly jednotlivé verze pomocí interního testu distribuovány mezi pět uživatelů, kteří hru testovali. Poté, co byla hra stabilní, byla posunuta do otevřeného beta testu. Hra je od té doby dostupná v beta verzi na obchodě Google play pod jménem Bad Dream.

Všechny cíle se podařilo splnit. Autor práce vnímá jako jediný neúspěch ovládání hry. Jedním z nefunkčních požadavků na hru bylo snadno pochopitelné ovládání. Takové ovládání se na základě zpětné vazby podařilo vyvinout pouze částečně a bude nutné ho do budoucna přepracovat. Aby hru bylo možné vydat v plné verzi, je nutné mimo jiné přidat více úrovní a dodělat jednotlivé grafické prvky hry.

Literatura

[1] Unity technologies: *Unity Store - Unity Personal*. [cit. 2018-04-21]. Dostupné z: https://store.unity.com/products/unity-personal

[2] Unity technologies: *Unity user manual*. [cit. 2018-04-21]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[3] Unity technologies: *What's new in Unity 4.3*. [cit. 2018-04-22]. Dostupné z: https://unity3d.com/unity/whats-new/unity-4.3

[4] RobTop Games: *Geometry Dash - Google Play[online]*. [cit. 2018-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.robtopx.geometryjump

[5] Frogmind: *Badland - Google Play[online]*. [cit. 2018-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.frogmind.badland

[6] Playdead: *Limbo - Google Play[online]*. [cit. 2018-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.playdead.limbo.full

[7] Gamux: *Effectively Organize Your Game's Development With a Game Design Document*. [cit. 2018-04-21]. Dostupné z: https://code.tutsplus.com/articles/effectively-organize-your-games-development-with-a-game-design-document--active-10140

[8] Sean Barton: *Tiled2Unity: Tiled Support for Unity*. [cit. 2018-04-22]. Dostupné z: http://www.seanba.com/tiled2unity

[9] Thorbjørn Lindeijer: *Tiled - MapEditor*. [cit. 2018-04-22]. Dostupné z: https://www.mapeditor.org

[10] Unity technologies: *What's new in Unity 2017.2*. [cit. 2018-04-22]. Dostupné z: https://unity3d.com/unity/whats-new/unity-2017.2.0

[11] Unity technologies: *Unity Asset Store - Unity Anima2D*. [cit. 2018-04-22]. Dostupné z: https://assetstore.unity.com/packages/essentials/unity-anima2d-79840

[12] MARIONETTE STUDIO: *Skeletal Based Animation*. [cit. 2018-05-2]. Dostupné z: https://marionettestudio.com/skeletal-animation/

[13] Roblox Corporation: *Inverse kinematics*. [cit. 2018-05-2]. Dostupné z: http://wiki.roblox.com/index.php?title=Inverse_kinematics

[14] Unity technologies: *Unity Asset Store - Ferr2D Terrain Tool*. [cit. 2018-04-22]. Dostupné z: https://assetstore.unity.com/packages/tools/level-design/ferr2d-terrain-tool-11653

[15] Unity technologies: *Unity Asset Store - 2D Rope Editor*. [cit. 2018-04-22]. Dostupné z: https://assetstore.unity.com/packages/tools/sprite-management/2d-rope-editor-62300

[16] Nystrom, R.: *Game Programming Patterns*. Genever Benning, první vydání, ISBN 978-0-9905829-0-8.

[17] Google: *Jak používat službu Play Console*. [cit. 2018-05-1]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/6112435?hl=cs

[18] Google: *Příprava testování alfa a beta verzí nebo interního testování*. [cit. 2018-05-1]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/3131213?hl=cs

Dodatek A

Seznam použitých zkratek

2D two-dimensional

3D three-dimensional

FPS frames per second

Dodatek B

Instalační příručka

K úspěšné instalaci hry je potřeba mít Android zařízení s verzí 4.0 nebo novější. Následně je nutné vykonat tyto kroky:

- Zajistit připojení k Internetu na Vašem zařízení.
- Otevřít obchod Google Play a v něm vyhledat hru se jménem **Bad Dream**. Případně je možné jít přímo přes tento odkaz: `https://play.google.com/store/apps/details?id=com.Usak.BadDream`.
- Tuto hru stáhnout a nainstalovat tlačítkem „Instalovat“.

Dodatek C

Ukázka výsledné hry

Obrázek C.1: Screenshot hlavního menu hry

Obrázek C.2: Screenshot části první úrovně

Obrázek C.3: Screenshot části druhé úrovně

Dodatek D

Obsah přiloženého USB disku

- `readme.txt` ..... stručný popis obsahu USB disku
- `baddream.apk` ..... instalační soubor hry
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF