VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HERNÍ MECHANIKY ZALOŽENÉ NA KAPALINÁCH

GAME MECHANICS BASED ON LIQUIDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADÉLA POMAJBÍKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

144144

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Studentka: **Pomajbíková Adéla**
Program: Informační technologie
Specializace: Informační technologie
Název: **Herní mechaniky založené na kapalinách**
Kategorie: Počítačová grafika
Akademický rok: 2022/23

Zadání:

1. Nastudujte herní engine Unity a techniky tvorby her s využitím kapalin. Prozkoumejte existující hry využívající simulaci kapalin.
2. Navrhněte herní mechaniky využívající kapaliny v logických hádankách. Navrhněte herní editor.
3. Implementujte navržené herní mechaniky a hru a využijte tyto mechaniky v několika úrovních.
4. Zhodnoťte a vytvořte demonstrační video. Hru zveřejněte.

Literatura:

- Gregory, Jason. *Game engine architecture*. crc Press, 2018. ISBN 1351974289, 9781351974288
- Bishop, Lars, et al. "Designing a PC game engine." IEEE Computer Graphics and Applications 18.1 (1998): 46-53.
- Adams, Ernest, and Joris Dormans. Game mechanics: advanced game design. New Riders, 2012. ISBN 0321820274, 9780321820273

Při obhajobě semestrální části projektu je požadováno:
Body 1 a 2 a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Milet Tomáš, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Cílem bakalářské práce je vytvoření herního dema pomocí mechanik založených na kapalinách v herním enginu Unity. Práce zkoumá trh stávajících řešení, návrh herních mechanik využívající kapaliny v logických hádankách, popis implementace těchto mechanik a editoru pro tvorbu nových úrovní. Výsledkem je herní demo obsahující několik úrovní a editor pro tvorbu vlastních úrovní.

Abstract

The aim of the Bachelor's thesis is to create a game demo using mechanics based on liquids in the Unity game engine. The thesis investigates the market of existing solutions, the design of game mechanics using liquids in logic puzzles, description of the implementation of these mechanics and an editor for creating new levels. The result is a game demo containing several levels and an editor for creating custom levels.

Klíčová slova

hra, herní demo, Unity, hra založená na kapalinách, C#, plošinovka, plošinová hra, logická hra, editor, vytváření úrovní

Keywords

game, game demo, Unity, game mechanics based on liquids, C#, platform, platform game, puzzle game, editor, level creation

Citace

POMAJBÍKOVÁ, Adéla. *Herní mechaniky založené na kapalinách.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

Herní mechaniky založené na kapalinách

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Tomáše Mileta, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Adéla Pomajbíková
10. května 2023

Poděkování

Chtěla bych poděkovat vedoucímu práce doktoru Tomáši Miletovi za vedení práce a trpělivost, kterou vůči mně měl. Také bych ráda poděkovala svému partnerovi, který mi byl oporou v těžkých chvílích.

Obsah

Kapitola 1

Úvod

V současné době je herní průmysl velkou součástí světové ekonomiky. Nové generace vyrůstají v tomto rozmachu a pro většinu z nich se hraní her stalo denní rutinou. Neustálý příchod nových technologií má za následek náročnější požadavky hráčů, a proto dochází k neustálému vývoji v této oblasti. Hraní videoher již neznamená pouze způsob, jak si zkrátit čas a zabavit se. Umožňuje vzdělávat se zábavnou formu, procvičit své logické myšlení nebo si například vydělat peníze.

Každý den přibývají desítky nových her: počítačových, mobilních či na herní konzole. Proces vydání hry však není zcela jednoduchý. Herní studia potřebují investory, kteří jim poskytnou finanční podporu, tým programátorů, grafiků a dalších, kteří se na vytvoření hry podílí. Nové hry potom většinou vznikají z již osvědčených žánrů a mechanik, u kterých je větší šance, že se u hráčů uchytí. Menší studio či vývojář, který nemá k dispozici tyto prostředky, se proto snaží hráče zajmout inovativní hrou.

Logické hry existují více než čtyřicet let. Nejznámější z nich je hra Tetris, která i v dnešní době existuje v mnoha podobách. Další revoluční hrou se stal Portál, ve kterém se hráč musí dostat na druhou stranu místnosti za pomoci portálů, které vytvořil. Hra přinesla možnost přemýšlet nad tím, kde a kdy portály umístit, aby hráč získal potřebnou rychlost a směr pro zdolání překážek. Nedlouho na to vyšla hra Braid, která umožňuje manipulaci s tokem času. V dnešní době se tyto herní mechaniky využívají k vytvoření originálních her, které hráče ohromí.

Mám mnoho sourozenců, takže jsem celé dětství strávila hraním her venku. Nikdy jsem nehrála videohry, dokud jsem na střední škole nepoznala svého partnera. Ten se živí jako grafik a ukázal mi zákulisí tohoto odvětví. Našim společným cílem se stalo zaměstnání u velkého herního studia. Když jsem si prohlížela zadání bakalářských prací, bylo hned rozhodnuto, že je na čase vytvořit vlastní hru. Hru, kterou bych si chtěla sama zahrát, bude mě bavit, trochu procvičí mé logické myšlení a bude se něčím lišit od ostatních.

Tato práce se věnuje popisu návrhu a implementace hry v Unity. To obsahuje několik předem vytvořených úrovní, kde se hráč musí s omezeným množstvím oleje dostat na jeho konec. V cestě mu stojí logické hádanky, které musí vyřešit, a přitom myslet na to, aby nevyplýtval příliš mnoho oleje, bez kterého je obtížné úroveň splnit. Hra nabízí možnost vytvoření vlastní úrovně pomocí rastrového obrázku.

V kapitole 2 je rozebrán žánr logické plošinové hry a představení existujících her s podobnou tématikou, které byly inspirací pro vznik hry. Kapitola 3 popisuje herní vývoj, herní engin, prostředí Unity a komponenty využité při tvorbě herních mechanik. Kapitola 4 se zabývá návrhem těchto mechanik a hry a kapitola 5 jejich následnou implementací v Unity.

Kapitola 2

Představení hry a existujících řešení

Tato kapitola nejprve popisuje vytvořenou hru a herní žánry, do kterých lze hru zařadit. Následně představuje existující hry patřící do těchto herních žánrů a hry využívající herní mechaniky založené na kapalinách.

2.1 Představení hry

Tématem hry je temné prostředí jeskyně. Hráč ovládá postavu horníka, který uvízl v jeskyni a hledá cestu ven (na obrázku 2.1). Řeší při tom logické hádanky, sbírá vzácné kameny a vyhýbá se nebezpečím, jako je láva a nebezpeční draci, kteří po něm střílí ohnivé koule.

K řešení hádanek využívá kapaliny, které mají různé schopnosti. Jedna kapalina snižuje tření, což hráči umožní rychleji se pohybovat nebo pohnout s těžkou kostkou. Jiná zase umožňuje vytvořit louži, která funguje jako trampolína. Těchto kapalin má ale omezené množství. Hráč proto musí přemýšlet, kdy a jakou kapalinu použít. Cílem hry je dostat se na konec úrovně. Při cestě lze sbírat vzácné kameny. Počet posbíraných vzácných kamenů je zaznamenán na konci dokončené úrovně, kde má hráč přehled, jestli se mu povedlo sesbírat je všechny.

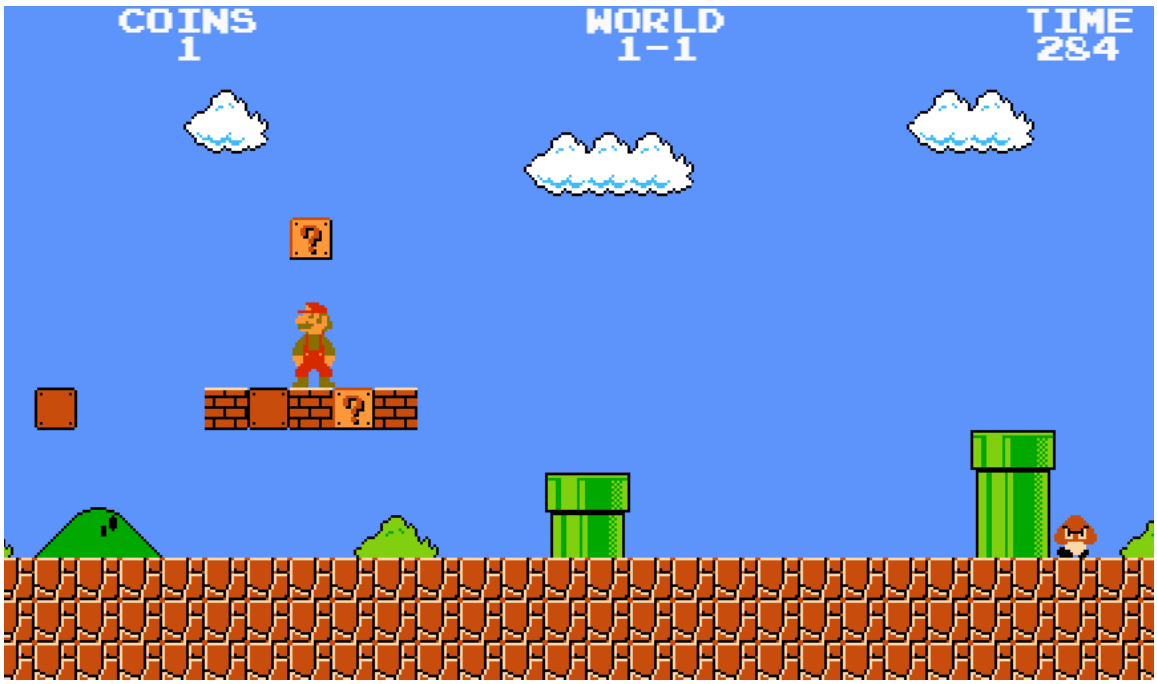
Obrázek 2.1: Ukázka ze hry. Pro představení je na snímcích zvýšená intenzita světla. Ve výsledné hře hráč vidí pouze na krátkou vzdálenost okolo sebe. Vlevo na obrázku hráč střílí žlutou kapalinu. Vpravo na obrázku se hráč vyhýbá ohnivým střelám nepřítele.

2.2 Herní žánry

Herní žánry kategorizují videohry do skupin na základě různých specifik. Může se jednat o perspektivu, kterou hráči nabízí (první/třetí osoba), platforma (mobilní, počítačové), nebo například na základě pocitu, který člověku navodí (akční, oddechová). Hra pak může spadat do několika žánrů zároveň.

Plošinovky

Hlavní roli v tomto žánru hrají plošiny, po kterých může postava běhat a skákat. Hráč pohybuje s postavou v prostředí pomocí pohybů, jako je běh, skok nebo třeba houpání po laně. Cestou tak překonává překážky, propasti, nepřátele nebo například sbírá mince, aby dosáhl cíle, získal vysoké skóre nebo prostě přežil. Mezi nejznámější patří Super Mario (na obrázku 2.2), který se řadí do arkádových plošinovek.



Obrázek 2.2: Obrázek ze hry Super Mario. Hráč skáče po plošině a zespod naráží do žlutého bloku s otazníkem. Nárazem se z plošiny můžou uvolnit různá vylepšení nebo mince. Ve chvíli, kdy hráč nasbírá sto mincí, doplní se mu jeden život.

Logické hry

Jedná se o hry, u kterých člověk rozvíjí myšlení a strategické a abstraktní uvažování. Smyslem je často logickým způsobem vyluštit zadané hádanky, reagovat na tahy protihráče nebo řešit problémy. Cílem těchto her nemusí být nutně dostat se na konec úrovně. Může se jednat o složení hlavolamu, nahrání nejvyššího skóre, či poražení nepřítele. Mezi nejznámější patří Sudoku a šachy, z videoher pak Tetris a Portal.

2.3 Hry využívající kapaliny

Tato sekce představuje hry využívající kapaliny, jež se staly inspirací k vytvoření dema.

Portal 2

Portal 2 je hra vydaná společností Valve Corporation v roce 2011. Jedná se o FPS[1] logickou hru, ve které hráč hraje za postavu Chell, která je testovacím subjektem v Aperture Science. Postupně prochází testovacími místnostmi a pomocí portálové zbraně vytváří dvojici portálů, skrze které se může pohybovat po místnosti a přenášet jimi předměty. Využívá přitom fyzikální zákon zachování hybnosti. Pokud hráč skočí do jednoho portálu z vyvýšeného místa, nabere vysokou rychlost a vyletí z druhého portálu s touto rychlostí, což mu umožní doskočit na vzdálená místa. Často je zapotřebí projít jedním portálem a ještě před dopadem pod sebou vytvořit nový, kterým postava projde a dostane se tak ještě dál[2].

V průběhy hry se Chell dostane do dolu, ve kterém objevuje barevné gely. Odpuzující gel umožňuje hráči se od něj odrazit. Pokud je tímto gelem potřený předmět, způsobí to jeho odrážení po místnosti. Pohonný gel způsobuje, že se po něm lidé a předměty pohybují vysokou rychlostí (na obrázku 2.3). Pomocí konverzního gelu lze vytvořit portály na površích, kde před tím nemohly být vytvořeny. Poslední je čistící gel, který má schopnost smýt jiné gely z povrchů a předmětů. Vlastnosti gelů byly inspirací k vytvoření herních mechanik ve výsledné hře.

Obrázek 2.3: Obrázek ze hry Portal 2. Ve spodní části se nachází portálová zbraň, pomocí které hráč vytváří dvojici portálů, skrze které může projít, nebo jimi nechat projít předměty a gely. Na obrázku je vytvořený portál na plošině pod padajícím gelem. Gel ze shora padá do portálu a vychází druhým s touto rychlostí. Hráč tak může přenést gel pomocí portálů na místa, kde je může využít tak, aby se dostal na konec úrovně.

[1] First Person Shooter – hra se zbraněmi z pohledu první osoby.
[2] https://cs.wikipedia.org/wiki/Portal_2

Mario Kart a Crash Team Racing

Jedná se o motokárové závodní hry, ve kterých hráč závodí za jednu z postav daného universa. Během závodu se vyhýbá různým překážkám a sbírá vylepšení, která mu umožňují získat větší rychlost nebo zasáhnout soupeře a zpomalit je tak. Vítězí ten, kdo jako první protne cílovou čáru.

Při závodu se ve hře Mario Kart může na dráze objevit olejová nebo inkoustová skvrna. Olej způsobí, že se motokára po jeho přejetí roztočí, nebo na krátkou dobu zpomalí. Inkoustová skvrna se po přejetí rozstříkne na obrazovku a na chvíli částečně omezí hráčovu viditelnost (na obrázku 2.4a).

Na závodních tratích Crash Team Racing se nacházejí kaluže a bahno. Závodník se jim snaží vyhýbat, protože tyto tekutiny způsobují zpomalení motokáry (na obrázku 2.4b).

(a) Mario Kart 8 Deluxe (b) Crash Team Racing Nitro Fueled

Obrázek 2.4: Snímek obrazovky ze hry Mario Kart 8 Deluxe na obrázku 2.4a. Hráčovu obrazovku pokrývá inkoustová skvrna, která omezuje jeho viditelnost na trati po několik vteřin. Snímek obrazovky ze hry Crash Team Racing Nitro Fueled, závodní dráha Tiny Arena na obrázku 2.4b. Tato dráha je na mnoha místech pokryta kalužemi bahna. Pokud do nich hráč vjede, dojde k jeho zpomalení a je pro něj proto těžké z kaluže vyjet.

Super Mario Sunshine[3]

Hráči ovládají Maria, který se snaží vrátit na ostrov světlo, jež bylo ukradeno a očistit ostrov od toxického slizu. V tom mu pomáhá robotický batoh jménem F.L.U.D.D. (Flash Liquidizing Ultra Dousing Device) obsahující dvě výchozí trysky, mezi kterými lze rychle přepínat. První stříká proud vody, který smyje nečistoty nebo smete nepřítele. Druhá umožňuje Mariovi vznášet se na krátkou dobu ve vzduchu a zároveň stříkat vodu na věci pod sebou. V průběhu hry Mario odemkne další dvě trysky, pro vystřelení Maria vysoko do vzduchu a k pohybu vysokou rychlostí. Každá tryska využívá vodu ze svých zásob, které lze doplňovat pomocí vodních zdrojů, jako jsou řeky a fontány (na obrázku 2.5). Trysky byly inspirací k vytvoření zbraně, která střílí kapaliny.

Splatoon 3[4]

Splatoon 3 (na obrázku 2.6), stejně jako předchůdci série Splatoon, je střílečkou z pohledu třetí osoby. Hráč ovládá zbraně, které používají barevný inkoust a obarvují tak herní pole. Množství inkoustu je omezené a postupně se doplňuje pohybem hráče po rozstříknutém

[3] https://en.wikipedia.org/wiki/Super_Mario_Sunshine
[4] https://en.wikipedia.org/wiki/Splatoon_3

Obrázek 2.5: Ukázka ze hry Super Mario Sunshine. Mario využívá vodní trysky, které má v robotickém batohu. Tyto trysky mu umožňují vznášet se delší dobu ve vzduchu, a tak překonat delší vzdálenost při letu a doskočit na odlehlou plošinu. V pravém dolním rohu se nachází zásobník vody. Při používání trysek voda ze zásobníku ubývá. Tuto zásobu lze doplnit pomocí vodních zdrojů, které se nacházejí ve hře.

inkoustu své barvy. Existuje zde humanoidní a „plovací“ forma. V plovací formě se postava změní v chobotnici, což jí umožňuje šplhat po stěnách pokrytých inkoustem nebo rychleji se pohybovat než v lidské formě. Také inkoust se v této formě doplňuje rychleji.

Zbraně většinou připomínají předměty z domácnosti a každá se od sebe schopnostmi liší. Například zbraň podobná válečku dokáže pokrýt velkou plochu, ale funguje pouze na blízko. Jiné připomínají odstřelovací pušky, mají velký dostřel, ale jsou méně účinné při natírání země. Tyto schopnosti byly inspirací k vytvoření různě chovajících se kapalin při výstřelu.

Obrázek 2.6: Ukázka kampaně pro jednoho hráče ve hře Splatoon 3. Hráč překonává úroveň, která je pokrytá slizem a přebarvuje ji na barvou svého inkoustu. Inkoust neslouží pouze k barvení mapy, ale taky jako zbraň proti nepřátelům.

Kapitola 3

Hra, herní vývoj, herní engine

Existuje mnoho definic, kterými lze popsat slovo „hra“. Na začátku této kapitoly je nastíněna jedna z těchto definic a následuje popis o tom, jak obecně probíhá herní vývoj. Předposlední podkapitola vysvětluje pojem *Herní engine* a jak vznikl. Závěr kapitoly popisuje Unity engine a stěžejní komponenty, které byly použity při vývoji hry.

3.1 Co je to hra

Obecný pojem „hra“ zahrnuje stolní hry, jako šachy nebo *Člověče, nezlob se!*, karetní hry jako *Prší!* nebo poker, počítačové hry a mnoho dalších. V kontextu virtuální zábavy slovo „hra“ obvykle vyvolá představu o prostoru s hlavní postavou, vozidlem, které hráč ovládá. Podle Gregoryho [4] je ve většině videoher určitá podmnožina reálného (či imaginárního) světa modelována matematicky tak, aby s ní bylo možné manipulovat pomocí počítače. Tento model je přiblížením a zjednodušením reality (i když jde o realitu imaginární), protože je nepraktické zahrnout do něj každý detail až na úroveň jednotlivých atomů.

Raph Koster ve své knize „*A Theory Of Fun For Game Design*“ popisuje, že hry se mohou zdát abstrahované od reality, protože jsou ikonickým zobrazením zákonitostí ve světě. Mají více společného s tím, jak si náš mozek věci vizualizuje, než s tím, jak se ve skutečnosti utváří realita, protože naše vnímání reality je stejně v podstatě abstrakce. Hry jsou podle něj hádanky k řešení, stejně tak jako vše ostatní, s čím se v reálném životě setkáváme. U řízení auta, hraní na hudební nástroj nebo násobení se učíme základní zákonitosti, plně je chápeme a ukládáme si je, abychom je mohli podle potřeby opakovat. Jediný rozdíl mezi hrami a realitou je v tom, že u her jsou sázky nižší. Dle jeho definice je tedy hra interaktivní zážitek, který hráči poskytuje stále náročnější sled vzorců, které se postupně naučí a nakonec je zvládne [6].

3.2 Herní vývoj

Herní vývoj je proces, který popisuje jednotlivé fáze vývoje videohry od konceptu až po její dokončení. Ačkoli se postup tohoto procesu může lišit podle projektů a studií, jeho jádro zůstává poměrně podobné, ať už se vyvíjí AAA[1] nebo mobilní hra. Tento postup pomáhá řídit časový plán vývoje a rozpočet, čímž napomáhá snížit neefektivitu a další slabá místa.

[1] AAA je neformální klasifikace používaná pro kategorizaci videoher vyráběných a distribuovaných středně velkým nebo velkým vydavatelem, které mají obvykle vyšší rozpočty na vývoj a marketing než ostatní úrovně her. Dostupné z https://en.wikipedia.org/wiki/AAA_(video_game_industry)

Postup nemusí být vždy lineární, protože se hra neustále vyvíjí a věci, které zněly teoreticky skvěle, nemusí ve skutečnosti dobře fungovat [7]. Herní vývoj se nejčastěji dělí na tři fáze: předprodukční, produkční a postprodukční (na obrázku 3.1).

Obrázek 3.1: Tři základní fáze herního vývoje. Fáze předprodukce zahrnuje prvotní myšlenku a plánování, například o čem hra bude, jaká je konkurence a jak bude hra vypadat. Důležitou roli zde hraje návrh herního designu, herních mechanismů, výběr herního žánru či cílové platformy. Obsah této fáze se liší na základě vydavatele. Pokud se jedná o studio, patří do této fáze plán o tom, jak dlouho bude vývoj trvat, jaký je odhadovaný rozpočet nebo jaký personál a zdroje bude vývoj vyžadovat. Z informací nashromážděných ve fázi předprodukce studio vytvoří základ pro GDD (Game Design Document). GDD je dokument, který se používá k pochopení vize daného projektu. Obsahuje například koncept hry, příběh a postavy, design úrovní či základní mechanismy. Do fáze předprodukce patří i prototypování. Jedná se o hrubé testy, které ověřují funkčnost, uživatelský zážitek, hratelnost a mechaniky. Testuje se, zda herní nápad bude fungovat a má smysl se jím zabývat. Nejdelší a nejdůležitější fází je produkce. Vytváří se postavy, prostředí, předměty, zvuky, úrovně, světy, pravidla a mnoho dalšího. V této fázi vzniká první hratelná verze, ukázka pro představení hry a Alfa verze. Vydání Alfa verze znamená, že byly přidány všechny hlavní prvky a hra je plně hratelná od začátku do konce. Následuje Beta verze, která má integrovány všechny prostředky i obsah a vývoj se tím zaměřuje spíše na optimalizaci. Výsledkem této fáze je finální hra připravená k vydání. Poslední fází je postprodukce. Po vydání hry se někteří členové týmu přesunou k dalšímu projektu. Zbývající členové se věnují údržbě hry nebo tvorbě bonusového obsahu [10].

3.3 Herní engine

Herní engine je software, který umožňuje vyvíjet digitální hry pro různé platformy. Poskytuje grafické rozhraní, fyzikální enginy, programovací a další nástroje potřebné pro vývoj her. Vznik tohoto termínu je datován na konec devadesátých let a je spojován s FPS hrami, jako je například Doom. Jeho architektura od sebe rozumně oddělovala jádro softwarových komponent (například vykreslování 3D grafiky, systém detekce kolizí) od uměleckých assetů[2]. Oddělovala tedy od jádra hry uměleckou stránku, která tvoří herní prožitek hráče. Přínos tohoto rozdělení se ukázal v momentě, kdy vývojáři začali licencovat hry a přetvářet je v nové produkty. K tomu stačilo vytvořit nové umělecké prostředky (zbraně, vozidla), herní prvky, prostředí a herní pravidla s minimálními změnami jádra. To mělo za následek zrod nové komunity, samostatných hráčů a malých nezávislých studií, která vytváří nové hry s využitím stávajících her a sad nástrojů poskytnutých původními vývojáři. Začaly se

[2]Zkratka pro vše, co je součástí videohry – postavy, objekty, zvukové efekty, mapy, prostředí atd.

vytvářet hry s ohledem na opětovnou modifikaci a použití, viz [4]. Tisková zpráva od id Software, tvůrce hry DOOM, slibovala nový druh otevřené hry a skutečně. Technologie herního enginu id se stala motorem nového odvětví počítačových her, jak uvádí [8].

V současnosti si mohou vývojáři licencovat herní engine a znovu tak použít jeho klíčové komponenty za účelem vzniku nových her, díky čemuž je vývoj rychlejší a levnější. Existuje mnoho herních enginů, open-source[3] i komerčních. Mezi nejpopulárnější patří Unity, Unreal Engine, Godot a další.

3.4 Unity

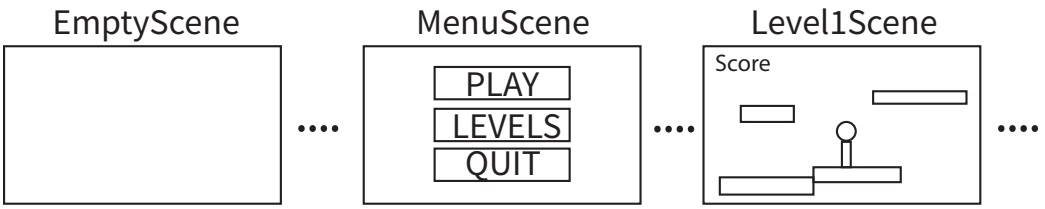
Unity je herní engine (kapitola 3.3), který byl poprvé vydán v roce 2005 pouze pro macOS. Od té doby byl postupně rozšířen a lze ho nyní použít k tvorbě 3D a 2D her pro více než 25 různých platforem. Unity však nalezne uplatnění i mimo herní průmysl. Používá se i v jiných oblastech a odvětvích, než jsou videohry, včetně virtuální/rozšířené reality, simulací, ve strojírenství, při vizualizaci výzkumných dat a u automobilového designu a marketingu [5].

Unity umožňuje vývojářům vytvářet kvalitní hry s minimálními znalostmi programování. Těm, kteří dávají přednost vizuálnímu programování před úpravou kódu, je k dispozici vizuální skriptování. To v Unity umožňuje tvůrcům vyvíjet herní mechaniky pomocí vizuálního systému založeného na grafech, namísto psaní řádků tradičního kódu.

Další předností je rozsáhlá vývojářská komunita. Existuje mnoho fór a různých tutoriálů, na kterých vývojáři sdílí znalosti a zkušenosti a podporují se navzájem. Tato komunita čítá přes milion vývojářů, což se ukazuje i na množství dostupných assetů, které si lze zdarma stáhnout či zakoupit.

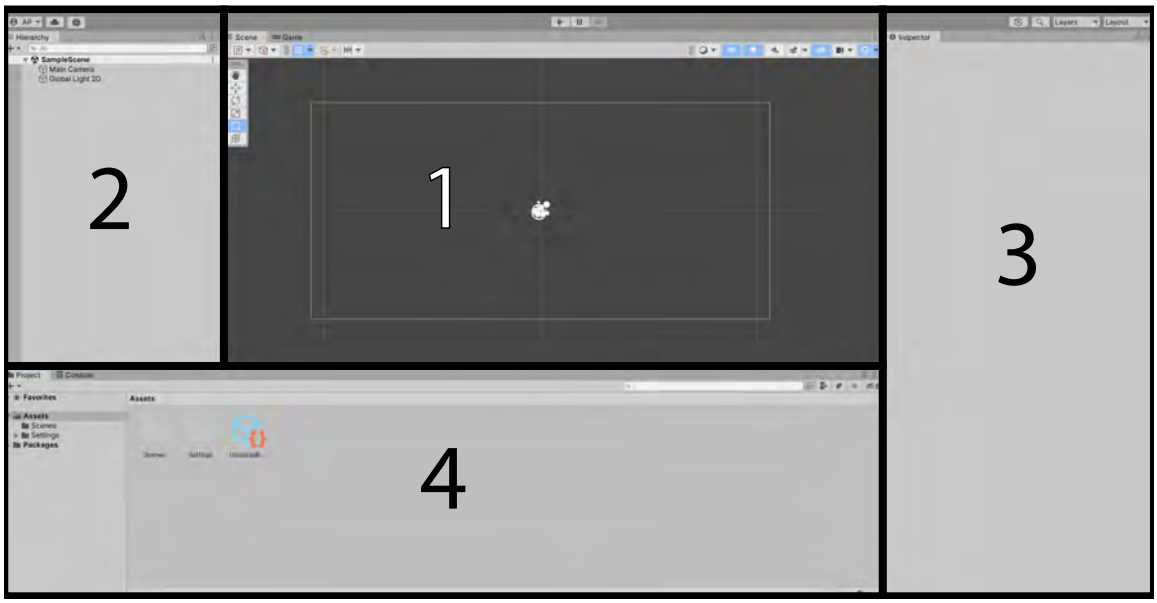
3.4.1 Prostředí Unity

Po vytvoření nového projektu a jeho otevření, Unity otevře ukázkovou scénu, která obsahuje pouze kameru (na obrázku 3.3). Scénu lze chápat jako kontejner, ve kterém Unity pracuje s obsahem. Je tvořena herními objekty, jako je hráč, světlo, tlačítko apod. Scény mohou obsahovat všechny aspekty hry, jako jsou jednotlivé herní úrovně, menu, celá aplikace nebo pouze její část. V projektu lze vytvořit libovolný počet scén (obrázek 3.2).



Obrázek 3.2: Obrázek znázorňuje několik různých scén vytvořených v rámci jednoho projektu v Unity.

[3] Open-source je něco (nejčastěji program, část kódu), co mohou lidé upravovat a sdílet, protože jeho návrh je veřejně přístupný.

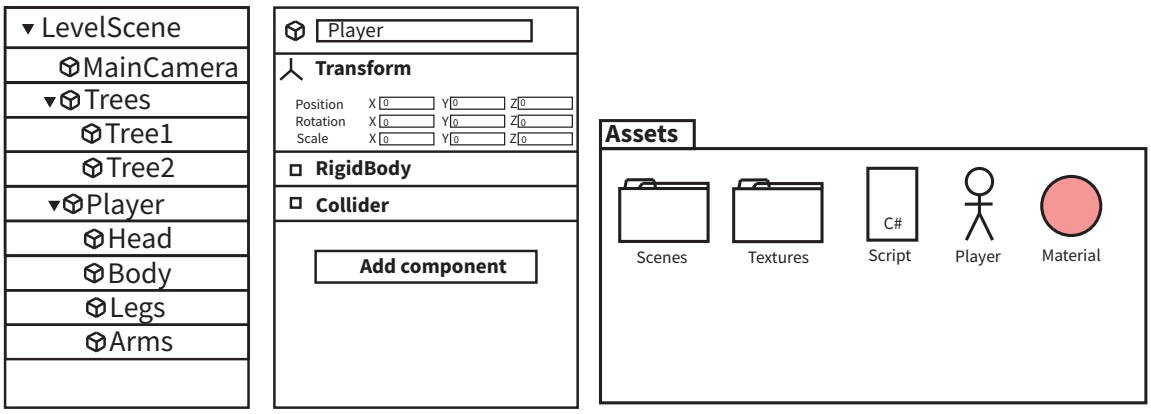


Obrázek 3.3: Prázdná scéna obsahuje kameru, pojmenovanou Main Camera. Ta vykresluje vše, co zachytí v oblasti, která se nazývá viewport (číslo 1). Vše, co se do této oblasti dostane, se stane viditelným pro hráče. V horní části projektu lze přepínat mezi zobrazením scény (tlačítko Scene) a oblastí, kterou vidí hráč (tlačítko Game). V levé horní části projektu se nachází okno Hierarchy (číslo 2), které zobrazuje strukturu scény. Zde jsou uvedeny všechny objekty obsažené v aktuálně otevřené scéně (na obrázku 3.4a). V pravé části se nachází okno Inspector (číslo 3), které slouží k zobrazení a úpravě vlastností téměř všech položek v editoru Unity. Nejčastěji slouží k zobrazení komponent (viz 3.4.2) přidaných k vybranému hernímu objektu (na obrázku 3.4b).Ve spodní části se nachází okno Project (číslo 4). Okno Project zobrazuje všechny soubory související s aktuálním projektem. To zahrnuje textury, skripty, zvuky, obrázky, předvytvořené herní objekty (prefabrikáty) nebo například vytvořené scény (na obrázku 3.4c).

3.4.2 Herní objekty

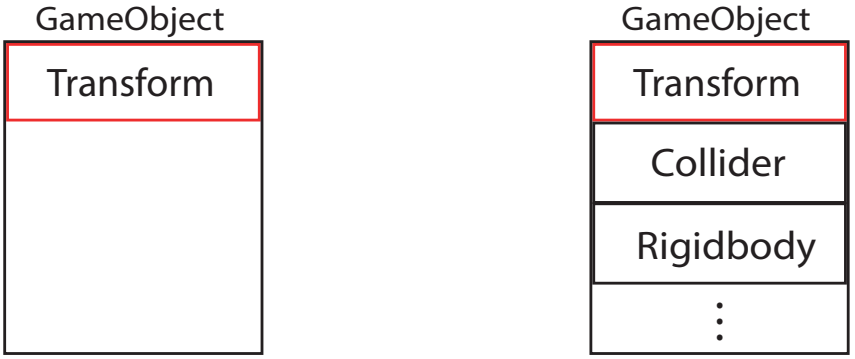
Herní objekt (GameObject) v Unity představuje cokoli, co může existovat ve scéně. Může to být hráč, nepřítel, zbraň, pozadí, nebo prázdný objekt, který je nosičem zvuku. Herní objekt funguje jako kontejner pro funkční komponenty, ale sám o sobě nemá žádnou funkcionalitu (na obrázku 3.5a). Komponenty implementují skutečnou funkci: určují, jak objekt vypadá, co dělá a jak reaguje na další objekty ve scéně (na obrázku 3.5b).

Skript je jedna z komponent, kterou lze připojit k hernímu objektu. Při vytvoření nového skriptu se automaticky vygeneruje vzorový skript, který je ve výchozím nastavení odvozen od třídy MonoBehaviour. Třída MonoBehaviour poskytuje vývojářům her rámec pro interakci s enginem Unity. Připojit skript ve scéně k herní objektu lze pouze tehdy, pokud třída v něm vytvořená dědí ze třídy Monobehaviour. Tato třída poskytuje přístup k rozsáhlé kolekci zpráv o událostech a vestavěným metodám, které umožňují spouštět kód na základě toho, co se v projektu právě děje. Patří sem například funkce `Start()` a `Update()`, které jsou při výchozím nastavení vždy vygenerovány v novém skriptu [3]. Diagram vztahů mezi těmito třídami je na obrázku 3.6.



(a) Hierarchy (b) Inspector (c) Assets

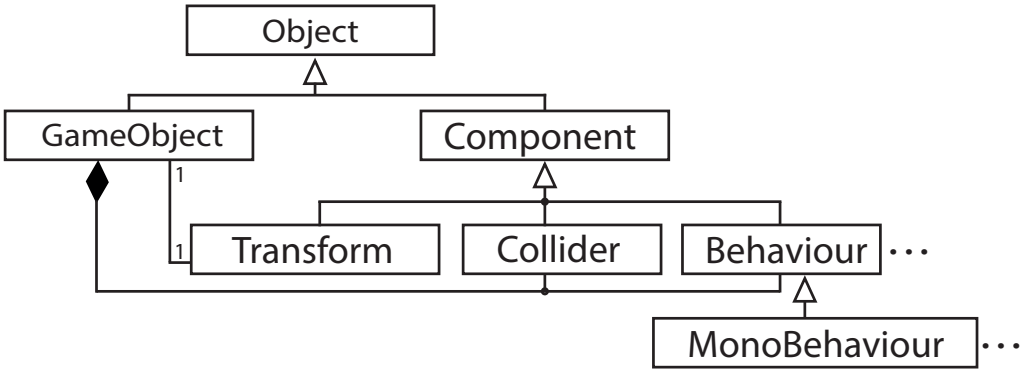
Obrázek 3.4: Na obrázku 3.4a je znázorněna hierarchie herních objektů vyskytujících se ve scéně. Objekty lze seskupovat (Head je potomkem objektu Player). Při pohybu, změně rotace nebo měřítka rodičovského objektu dojde ke změně i u všech podřízených objektů. Obrázek 3.4b ukazuje komponenty připojené k vybranému hernímu objektu a jejich vlastnosti. Nejdůležitější je zde komponenta Transform, kterou obsahuje každý objekt. Pomocí tlačítka „Add Component“ lze vybranému objektu přidat další komponenty, jako je kolizní prvek, textura nebo například skript. Obrázek 3.4c zobrazuje všechny soubory a adresáře, které obsahuje aktuálně otevřený projekt.



(a) Prázný herní objekt obsahující pouze komponentu Transform.

(b) Herní objekt obsahující komponenty Transform, Collider a Rigidbody.

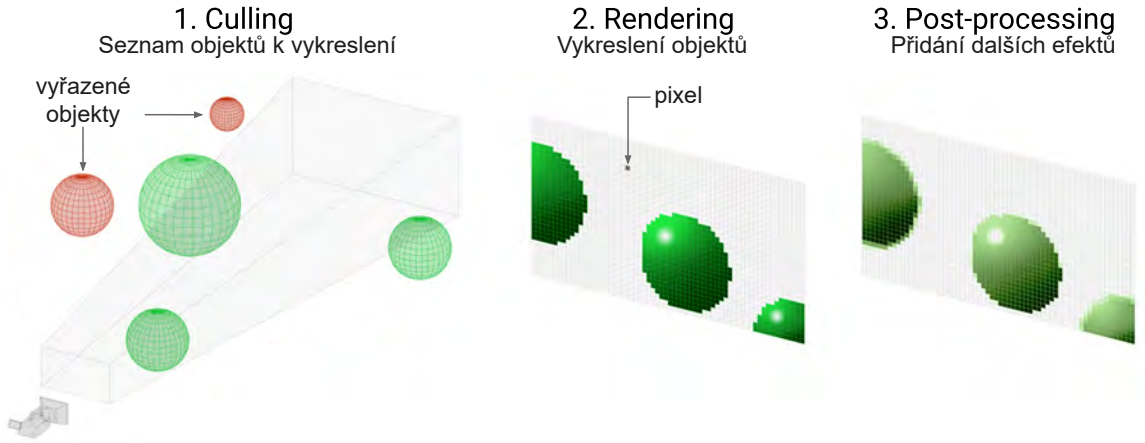
Obrázek 3.5: Znázornění herních objektů, jako kontejnerů na komponenty. Každý vytvořený herní objekt, i prázdný (obrázek 3.5a), má vždy komponentu Transform. Ta určuje jeho pozici, rotaci a měřítko vzhledem k hernímu světu nebo k jeho nadřazenému objektu (hierarchie objektů na obrázku 3.4a). Obrázek 3.5b ukazuje herní objekt s přidanými komponentami.



Obrázek 3.6: Schéma vztahů mezi nejčastěji používanými třídami v Unity. Object je základní třída pro všechny objekty, na které se Unity může v editoru odkazovat. Z ní vychází třída GameObject, která umožňuje pracovat s herními objekty v kódu. Tato třída poskytuje kolekci metod, včetně vyhledávání, vytváření spojení a odesílání zpráv mezi herními objekty, stejně tak jako nastavování hodnot a přidávání/odebírání, komponent danému objektu. Každý GameObject má vždy komponentu Transform. Všechny ostatní komponenty, jako je Collider nebo Behaviour, které dědí ze třídy Component, mohou být k objektu připojeny. Od třídy MonoBehaviour se ve výchozím nastavení odvozuje každý skript. Pokud chceme skript připojit k hernímu objektu jako komponentu, musí dědit právě z této třídy.

3.4.3 Universal Render Pipeline

Vykreslovací proces (render pipeline) provádí řadu operací, které přebírají obsah virtuální scény a zobrazují jej na obrazovku. Tyto operace se provádějí ve třech hlavních fázích, které se opakují mnohokrát za sekundu na základě snímkové frekvence (na obrázku 3.7).



Obrázek 3.7: Tři hlavní fáze vykreslovacího procesu. První fáze se nazývá Culling. Ten vypíše objekty, které je třeba vykreslit. Přednostně se vypisují ty, které jsou viditelné kamerou (frustum culling) a které nejsou zakryté jinými objekty (occlusion culling). Druhou fází je Rendering – vykreslení těchto objektů se správným osvětlením a některými jejich vlastnostmi do pixelových bufferů (vyrovnávací paměti). Poslední fází je Post-processing, tedy operace, které lze provádět s těmito buffery. Obrázek převzatý z internetu[4].

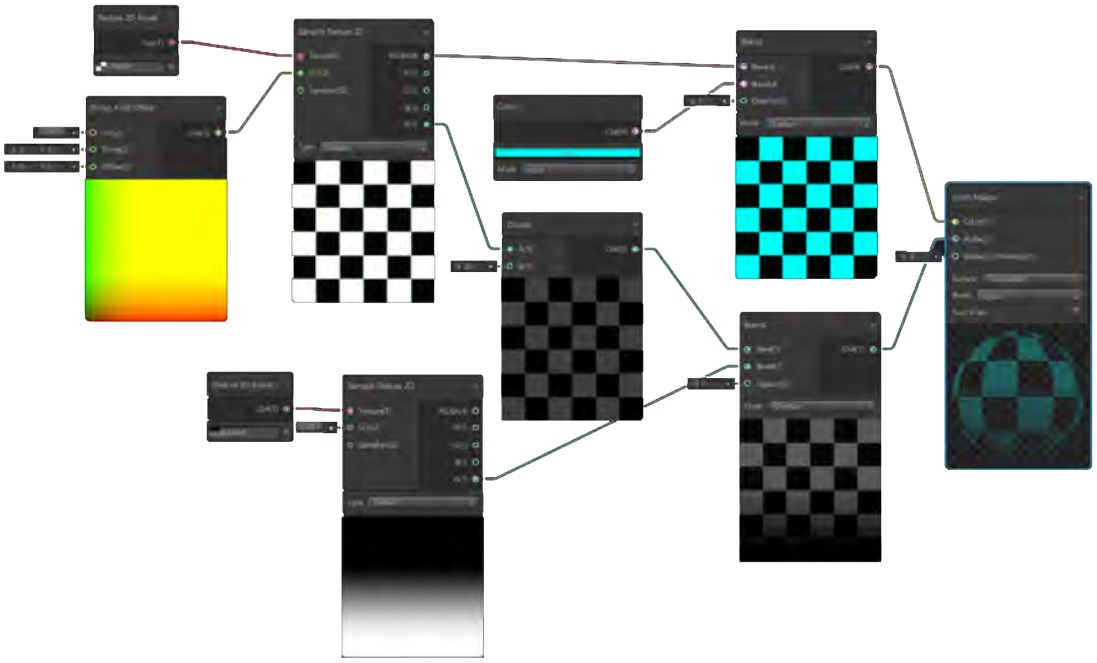
Unity poskytuje tři předpřipravené vykreslovací modely s různými schopnostmi a charakteristikami (vestavěný, univerzální URP a vykreslování ve vysokém rozlišení HDRP[5]). Další možností je vytvořit si vlastní render pipeline pomocí rozhraní SRP (Scriptable Render Pipeline)[6].

Universal Render Pipeline (URP) je proces vykreslování založený na skriptovatelném vykreslovacím modelu v Unity. Je vhodné jej použít pro většinu her, které nevyžadují mnoho pokročilých grafických funkcí, jako jsou mobilní hry nebo jednoduché počítačové hry, zatímco High Definition Render Pipeline (HDRP) poskytuje nástroje potřebné k vytváření aplikací na vysoké grafické úrovni. Před vytvořením projektu je nutné zvolit, který render pipeline projekt použije, protože funkce HDRP nejsou kompatibilní s URP, viz [2].

URP umožňuje nasadit projekt na většinu platforem, které Unity podporuje. Poskytuje podporu VFX grafu a Shader Grafu (na obrázku 3.8). Podporuje širokou škálu řešení přímého a nepřímého osvětlení a výraznou optimalizaci v komplexním vykreslování osvětlení za běhu, zejména na hardwaru s nižšími výkonnostními možnostmi [11].

Součástí URP je systém 2D osvětlení, který umožňuje snadným způsobem vytvořit ve scéně pohlcující atmosféru. Obsahuje několik základních světelných komponent. Globální světlo rovnoměrně osvětluje celou scénu. Hlavní roli zde hraje intenzita a barva. Pokud se intenzita nastaví na hodnotu nula, ve scéně nebude nic vidět při absenci jiného druhu osvětlení. Bodové světlo vyzařuje světlo z určitého bodu. Obsahuje vnitřní a vnější kruž-

[4] https://docs.unity3d.com/Manual/BestPracticeLightingPipelines.html
[5] https://docs.unity3d.com/Manual/high-definition-render-pipeline.html
[6] https://docs.unity3d.com/Manual/render-pipelines.html
[7] https://learn.unity.com/tutorial/august-17-shader-graph



Obrázek 3.8: Shader Graf v Unity, který umožňuje vizuálně vytvářet shadery v reálném čase pomocí nástroje založeného na uzlech. Místo psaní kódu se vytváří a propojují uzly v rámci grafu. Obrázek převzatý z internetu[7].

nici, které lze nastavit poloměr. Toto světlo je ideální pro osvětlení malých zdrojů, jako je žárovka nebo lampa. Dalším druhem je freeform světlo. To se liší od bodového tím, že ho lze vytvarovat do jakéhokoliv tvaru. Posledním je Sprite osvětlení, které umožňuje nastavit konkrétní asset jako tvar samotného světla.

3.4.4 Částicový systém

Částicový systém v Unity se používá k vytvoření efektů, jako je kouř, kapalina, plameny, magická kouzla a mnoho dalších. Umožňuje vytvořit efekty, které je obtížné zobrazit pomocí obrázků, protože se často jedná o efekty, které nemají definovaný tvar. Systém emituje částice, což jsou malé reprezentace obrázků, které tvoří celek. Pomocí těchto částic lze simulovat efekty, které hru obzvláštní, vytvoří pro hráče požadovanou atmosféru a více ho ponoří do děje [9].

Unity nabízí dvě řešení částicových systému. Jedná se vestavěný a Graf vizuálních efektů (VFX). Vestavěný částicový systém je kompatibilní se všemi třemi předpřipravenými vykreslovacími modely. Dokáže emitovat tisíce částic a jejich vlastnosti lze nastavovat v jednotlivých modulech uvnitř Inspektoru. Částice mohou interagovat se základním fyzikálním systémem a pomocí skriptu lze ovládat jejich chování. VFX není kompatibilní s vestavěným vykreslovacím modelem, pouze s URP a HDRP. Dokáže emitovat miliony částic, které mohou interagovat s konkrétními prvky, které jsou definovány v grafu vizuálních efektů. Pomocí grafu lze částicím nastavit vlastnosti a prostřednictvím skriptu k nim přistupovat. To nabízí širokou škálu vlastností, které lze částicím přiřadit.

3.4.5 Fyzikální systém

Stejně tak jako ve skutečném světě platí fyzikální zákony a různá omezení, spousta her využívá alespoň malou podmnožinu této fyziky. Fyzikální systém v Unity zajišťuje, že objektům lze nastavit rychlost, tělesnou hmotnost, tření, gravitaci a další různé realistické síly. Objekty mohou interagovat s ostatními objekty v okolním světě. Díky tomu získávají realističtější pohyb a reálné vlastnosti, které hráčům pomáhají při řešení úkolů, s nimiž se v průběhu hry setkávají.

Pořadí provádění funkcí události

Ve většině článků se uvádí, že změna pohybu hráče by měla být provedena výhradně ve funkci `FixedUpdate()`, ale v mnoha dostupných tutoriálech programátoři používají pouze funkci `Update()`. Obrázek 3.9 ukazuje pořadí provádění událostí fyzikálního enginu a popisuje rozdíly mezi těmito funkcemi.

Rigidbody

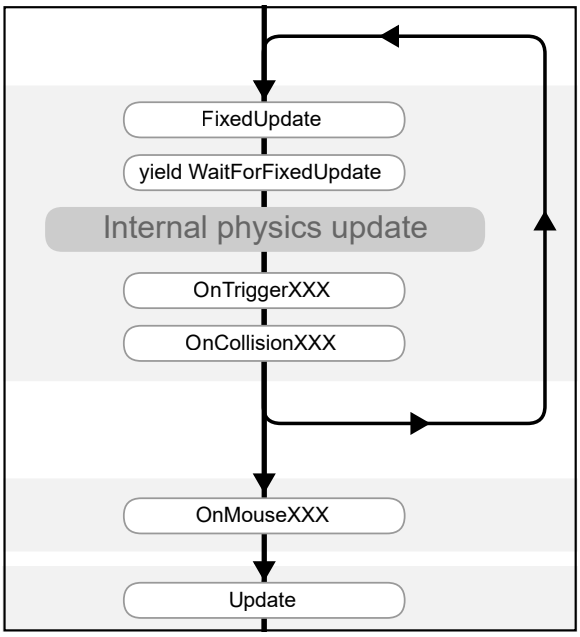
Komponenta Rigidbody umožňuje simulaci realistického chování hernímu objektu, ke kterému je připojena, skrze fyzikální engine Unity. Rigidbody dodá objektu atributy, které napodobují vlastnosti reálného života, jako je odpor vzduchu a hmotnost. I bez přidání jakéhokoli kódu je objekt tažen gravitací směrem dolů a reaguje na kolize s přicházejícími objekty, pokud je přítomna komponenta Collider. Rigidbody má také skriptovací API, které umožňuje aplikovat na objekt síly a ovládat jej fyzikálně realistickým způsobem.

Collider

Kolizní prvek (Collider) definuje tvar objektu pro účely fyzických kolizí. Tvoří neviditelnou bariéru, do které mohou jiné kolizní prvky narážet. S jeho pomocí lze získat informace o tom, že se herní objekt dostal do kontaktu s jiným. Pokud se střetnou dva objekty, které obsahují komponenty Collider a alespoň jedna z nich nekinetickou[9] komponentu Rigidbody, dojde k vyvolání událostí `OnCollision` a `OnTrigger`, které lze spravovat pomocí skriptu.

[8] https://docs.unity3d.com/Manual/ExecutionOrder.html

[9] Pokud je těleso kinetické, nemůže fyzikální systém použít síly k jeho pohybu nebo otáčení.



Obrázek 3.9: Fyzikální cyklus pracuje v iteracích dlouhých jako pevný časový krok (fixed time step). Ten je ve výchozím nastavení nastaven na hodnotu dvacet milisekund. Funkce `FixedUpdate()` se zpracovává těsně před aktualizací fyzikálního enginu (na obrázku 3.9). Proces začíná určením, zda uplynul dostatečný čas pro zahájení další iterace. Pokud ano, iterace vyvolá všechna zpětná volání `FixedUpdate()` definovaná ve všech aktivních MonoBehaviour na scéně. Jakmile jsou tyto úlohy hotovy, může fyzikální engine začít zpracovávat aktuální časový krok a vyvolat všechna potřebná zpětná volání spouštěčů a kolizních prvků. Pokud ne, aktuální iterace se přeskočí a všechny dříve uvedené úlohy se neuskuteční. V tomto okamžiku se vstupy, herní logika nebo vykreslování provede jako obvykle. Všechny fyzikální výpočty a aktualizace probíhají tedy bezprostředně po `FixedUpdate()`. Funkce je volána na spolehlivém časovači, který je nezávislý na snímkové frekvenci. Funkce `Update()` se volá jednou za snímek. Je to hlavní pracovní funkce pro aktualizace snímků. Umístění výpočtů pohybu Rigidbody do funkce `Update()` může způsobit neočekávané chování fyziky, protože může být provedeno více změn téhož objektu dříve, než je fyzikální engine stihne všechny zachytit a zpracovat [1]. Funkci `Update()` nebo `LateUpdate()` je vhodné použít například pro sledování hráče kamerou. Tím se zajistí, že pohyb postavy se provede předtím, než kamera začne sledovat její polohu. Celý diagram je dostupný na[8].

Kapitola 4

Návrh

Kapitola popisuje postup návrhu hry, rozdělený na návrh herní části a editoru. Implementaci se věnuje následující kapitola 5, tato kapitola je zaměřena pouze na popis návrhu.

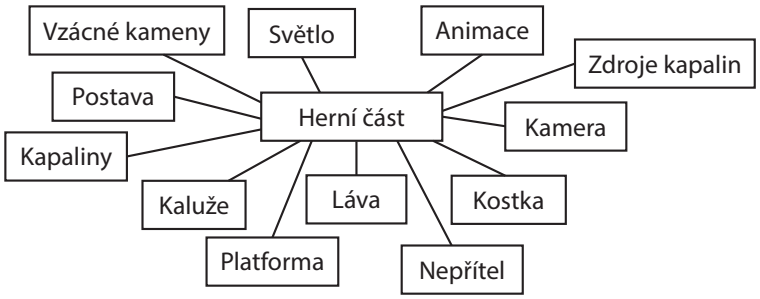
Prvním krokem při vývoji hry bylo navrhnout, o čem hra bude, jaký bude cíl, co bude obsahovat a v jakém herním enginu bude vytvořena. Prvním úspěšným nápadem se stala hra, ve které hráč využívá olej, aby snížil tření a dokázal pohnout s těžkými předměty. Z tohoto nápadu vzešlo zadání této práce.

Pro tuto hru byl zvolen žánr logických plošinových her (viz kapitola 2), který umožňuje hráči vytvořit nespočet jedinečných úrovní. Jediné omezení je hráčova představivost a schopnost vymyslet logickou hádanku. Rozhodla jsem se pro herní engine Unity (viz kapitola 3.4), z důvodů nízké vstupní bariéry pro začínající vývojáře.

Všechny assety vyskytující se ve hře jsou autorská díla vytvořená v rastrovém grafickém editoru Photoshop[1].

4.1 Herní část

Tato sekce se zabývá popisem návrhu herní části, stěžejních herních objektů a mechanik, které hra obsahuje. Výčet těchto objektů je znázorněn na obrázku 4.1.

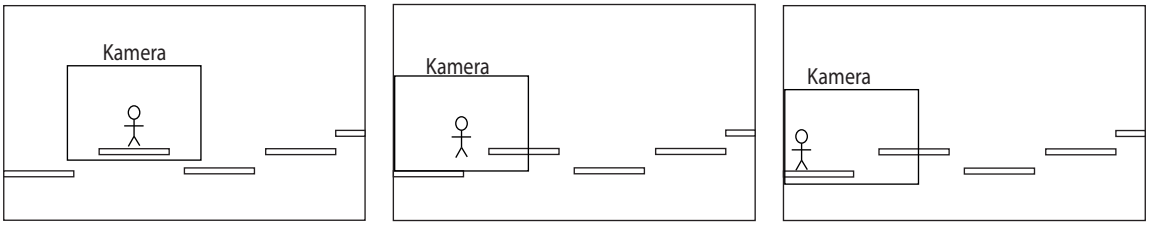


Obrázek 4.1: Obrázek znázorňuje komponenty hry, které spolu interagují. Jejich návrh je popsán níže v této sekci.

[1] https://www.adobe.com/cz/products/photoshop.html

4.1.1 Kamera

2D plošinové hry využívají statickou nebo dynamickou kameru. Statická kamera se při pohybu hráče nepohybuje a pouze přepíná jednotlivé „scény“ hry, když postava opustí zorné pole kamery. Zvolila jsem dynamický pohyb kamery, kdy kamera se se zpožděním pohybuje podle postavy. Pokud postava spadne z plošiny, nebo se dostane na okraj úrovně, kamera se zastaví spolu s okrajem a nedovolí hráči nahlédnout za okraj (na obrázku 4.2).



Obrázek 4.2: Ukázka pohybu kamery s postavou. Kamera „pronásleduje“ postavu, pohybuje se společně s ní, dokud se postava neocitne na okraji herní plochy. Kamera se v tomto případě zastaví o okraj a nepohybuje se. Jakmile se hráč vzdálí od okraje, kamera ho začne zase následovat.

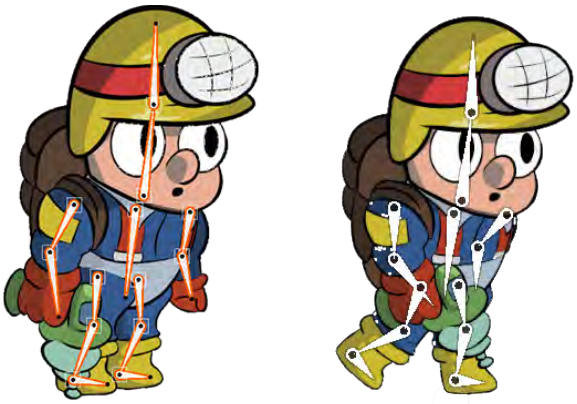
4.1.2 Postava

Postavu (na obrázku 4.3) lez ovládat pomocí kláves „A, D“ nebo „šipka vpravo, šipka vlevo“ v horizontálním směru a pomocí klávesy „mezerník“ ve vertikálním směru. Poloha kurzoru slouží k zaměření směru střelby z pistole a levé tlačítko myši k zahájení střelby.

Obrázek 4.3: Horník, hlavní postava hry. V ruce drží pistoli vystřelující několik druhů kapalin.

4.1.3 Animace

Postava je doplněna o animace pohybu. Ty jsou řízeny stavy, ve kterých se hráč nachází. Při pohybu se přehrává animace chůze (na obrázku 4.4), která se změní v okamžiku výskoku na výskok. Pokud se postava nepohybuje, spustí se animace nečinnosti.



Obrázek 4.4: Ukázka tvorby animace chůze pomocí kostí, které jsou připevněny k postavě. Vlevo na obrázku je zobrazena kostra v klidovém režimu. Vpravo jsou kosti posunuty tak, aby postava vypadala, že se pohybuje.

O jednoduché animace jsou doplněny také vzácné kameny, láva a nepřátelé. Kameny se ve smyčce neustále mírně roztahují do stran, aby upoutaly hráčovu pozornost. Nepřátelé se pohybují na místě a v lávových kamenech se pohybují pramínky lávy.

4.1.4 Platforma

Platforma ve hře představuje povrch, po kterém se může hráč pohybovat a jako jediný povrch může být pokryta kapalinami z pistole. Je tvořena z jednotlivých bloků (na obrázku 4.5).

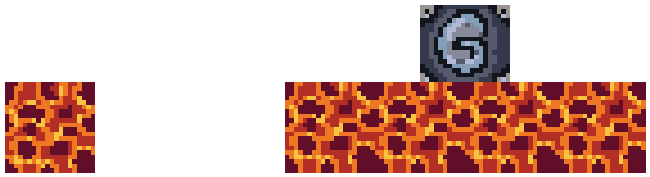
Obrázek 4.5: Platforma seskládaná z jednotlivých bloků. Pokud postava spadne z platformy, je vrácena za začátek úrovně.

4.1.5 Láva

Lávu ve hře představují polo zatvrdlé lávové kameny, kterých se postava nesmí dotknout (na obrázku 4.6). Pokud se jich dotkne, postava je přemístěna na začátek úrovně.

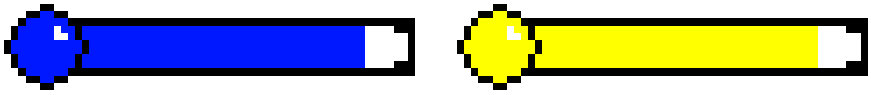
4.1.6 Kapaliny

Hráč pomocí klávesy „E“ přepíná mezi kapalinami, které zbraň využívá. Každá kapalina má své omezené množství zásob (na obrázku 4.7), které při střílení ubývá. Ve výchozím nastavení zbraň vystřeluje žlutou kapalinu, která reaguje s platformami a lávou. Při zásahu s horní stranou platformy rychle ztratí rychlost a zanechá po sobě mokrou skvrnu. Ze všech kapalin je vystřelována nejmenší silou a má největší viskozitu, proto její proud dostříkne



Obrázek 4.6: Láva seskládaná z jednotlivých bloků. Hráč může využít kostku, která se do lávy zaboří, a po ní bezpečně přejít. Žádná z kapalin nemá na lávu vliv, nejde ji uhasit nebo na ní vytvořit kaluži.

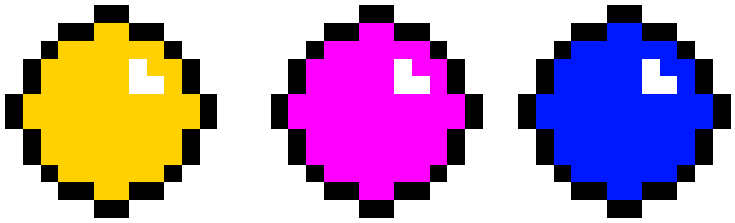
na nejkratší vzdálenost. Fialová kapalina funguje podobně jako žlutá s tím rozdílem, že při dotyku s horní stranou platformy se k ní ihned přichytí. Modrá kapalina se při výstřelu mění v mrazivé střely, proto při držení levého tlačítka myši zbraň vystřelí vždy pouze jednu střelu. Tyto střely lze využít ke zničení nepřátel nebo k posunutí kostky, proto je jejich zásoba nejnižší.



Obrázek 4.7: Ukazatele zásob kapaliny.

4.1.7 Zdroje kapalin

Zásoby kapalin lze doplnit pomocí zdrojů (na obrázku 4.8). Každý zdroj doplní část chybějící kapaliny příslušné barvy pouze do maximální kapacity zásobníku.



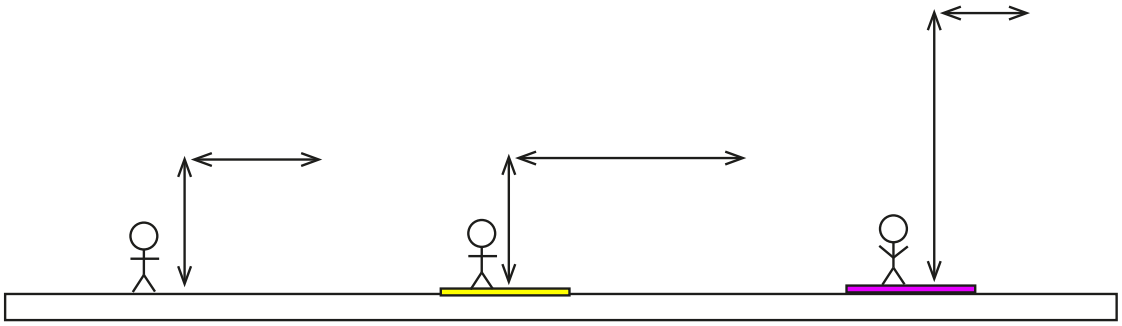
Obrázek 4.8: Zdroje kapalin ve hře. Každá barva může doplnit pouze konkrétní kapalinu.

4.1.8 Kaluže

Žlutá a fialová kapalina po sobě zanechává na horní straně platforem mokré skvrny (kaluže), které ovlivňují pohyb objektů (na obrázku 4.9).

4.1.9 Vzácné kameny

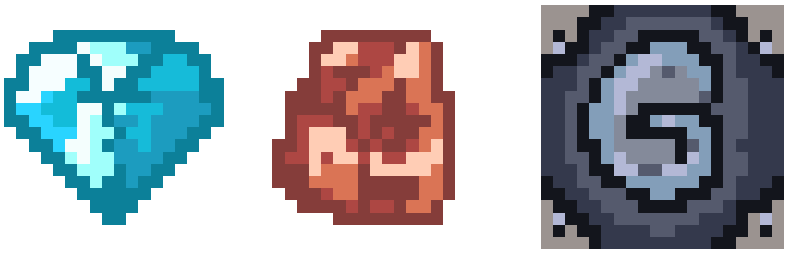
Hráč může při průchodu jeskyní sbírat diamanty a vzácnou rudu (na obrázku 4.10), které se ve hře na místě mírně pohybují. Při splnění úrovně se ve vítězném menu zobrazí, kolik kamenů z kolika možných posbíral.



Obrázek 4.9: Obrázek ukazuje maximální výšku výskoku a délku doskoku na vybraném povrchu. První obrázek zleva zobrazuje obyčejný skok na platformě. Žlutá kaluž snižuje tření mezi povrchem a dalšími objekty. Postava pohybem v kaluži může zvýšit svou rychlost a tím doskočit na vzdálená místa. Fialová kaluž funguje jako trampolína. Když se na ní postava rozeskáče, postupně doskočí výš. Manévrovat v letu je ale obtížné, proto při využití skoku na fialové louži se lze omezeně hýbat v horizontálním směru. Postava tedy vyskočí výš, ale ne dál. Pokud je jednou na platformě mokrá vrstva dané barvy, nelze ji již odstranit. Hráč proto musí dopředu zvažovat, v jakém místě kapalinu využít.

4.1.10 Kostka

Kostku (na obrázku 4.10) lze použít k bezpečnému průchodu po lávových kamenech. Je ale velmi těžká a mezi ní a platformou působí velká třecí síla. Aby se dalo s kostkou pohnout, musí se pod ni dostat žlutá kapalina, která sníží tření.



Obrázek 4.10: Ukázka diamantu, vzácné rudy a kostky.

4.1.11 Nepřítel

Nepřítel ve hře střílí ohnivé koule vždy do směru, ve kterém se nachází hráč. Pokud ho zasáhne, postava je přemístěna na začátek úrovně. Nepřítele lze zničit pomocí ledové střely.

Obrázek 4.11: Ukázka nepřítele, který střílí ohnivé koule.

4.1.12 Světlo

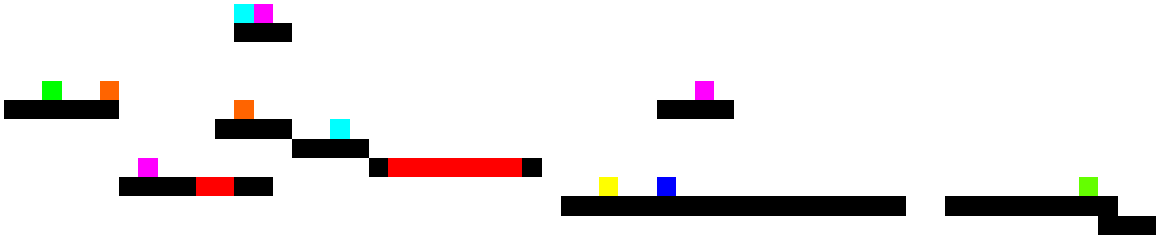
Ve hře se nenachází globální zdroj světla. V jeskyni je tma, až na lávové a vzácné kameny, které fungují jako malý zdroj světla. Hráč má vlastní zdroj světla v přilbě, díky němuž vidí na malou vzdálenost okolo sebe.

4.2 Editor

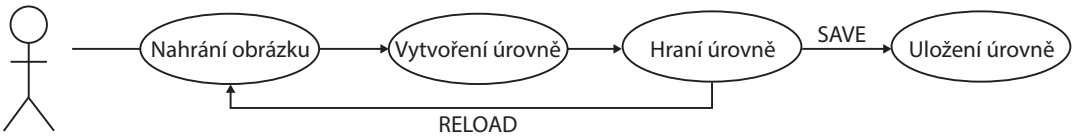
Editor umožňuje hráčům přidávat do hry nové vlastní úrovně pomocí rastrového obrázku (obrázek 4.12). Každý objekt, který lze využít při vytváření nové úrovně, je reprezentován hodnotou barvy RGBA (viz tabulka 4.1). Postup přidání nové úrovně je popsán na obrázku 4.13.

	RGBA hodnota		**Herní objekt**
	(0, 255, 0, 1)	→	Hráč
	(255, 30, 30, 1)	→	Nepřítel
	(0, 0, 0, 1)	→	Platforma
	(255, 0, 0, 1)	→	Lávový kámen
	(60, 30, 0, 1)	→	Kostka
	(100, 255, 0, 1)	→	Cíl
	(0, 255, 255, 1)	→	Diamant
	(255, 100, 0, 1)	→	Vzácná ruda
	(255, 255, 0, 1)	→	Zdroj žluté kapaliny
	(255, 0, 255, 1)	→	Zdroj fialové kapaliny
	(0, 0, 255, 1)	→	Zdroj modré kapaliny

Tabulka 4.1: Tabulka znázorňující hodnotu barvy RGBA a přidělený herní objekt.



Obrázek 4.12: Ukázka rastrového obrázku, ze kterého je editor schopný vytvořit úroveň. Každá barva pixelu znamená jiný herní objekt.



Obrázek 4.13: Obrázek zobrazuje smyčku vývoje nové úrovně. Hráč v menu nahraje rastrový obrázek, podle kterého editor vytvoří úroveň a spustí ji v editačním režimu. Tento režim slouží k vyzkoušení nově vytvořené úrovně. Hráč může vyzkoušet, zda úroveň vypadá, tak jak zamýšlel, zda lze dosáhnout cíle, posbírat všechny vzácné kameny apod. Ovládání funguje stejně, jako v běžném herním režimu. Pokud hráč není spokojen s úrovní, nemusí nahrávat nový obrázek v hlavním menu. Stačí, když v obrázku provede potřebné změny, které uloží a pomocí tlačítka „RELOAD“ načte obrázek znovu. Pokud je s úrovní spokojen, může kliknout na tlačítko „SAVE“ a úroveň přidat mezi ostatní již vytvořené.

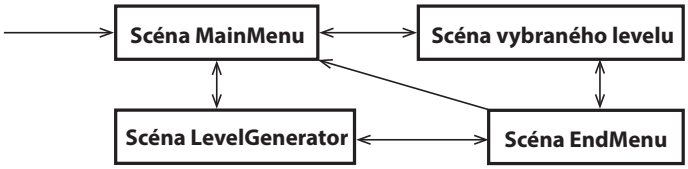
Kapitola 5

Implementace

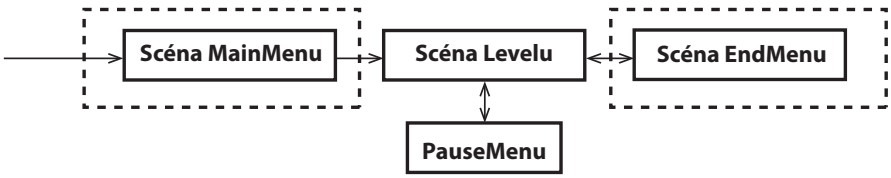
Tato kapitola popisuje implementaci hry. K tomu byl zvolen herní engine Unity (viz kapitola 3.4), verze 2021.3.24f1 s URP (viz 3.4.3), a programovací jazyk C#. Kapitola je rozdělena do tří částí. Nejprve je popsána struktura hry (všechny scény a menu), po ní následuje popis implementace herní části a herního editoru.

5.1 Struktura hry

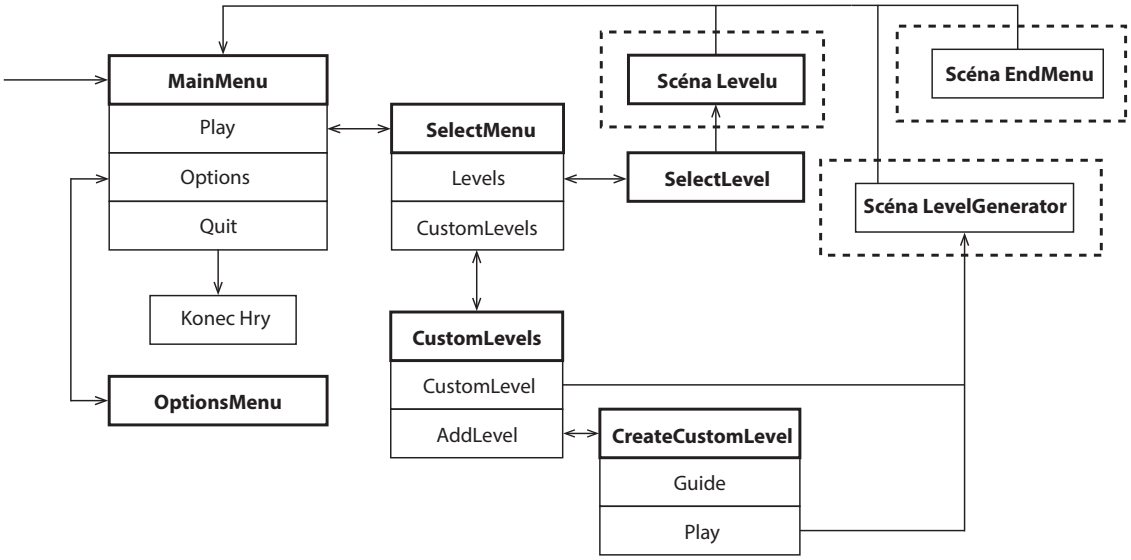
Hra obsahuje tři hlavní scény ovládající hru a tři scény s předvytvořenými úrovněmi. Přepínání mezi těmito scénami je znázorněno na obrázku 5.1. Všechny scény obsahují herní objekt s komponentou Canvas. Jedná se o oblast, ve které se nacházejí všechny prvky uživatelského rozhraní. K tomuto objektu je přidána také komponenta Canvas Scaler, která slouží k ovládání celkového měřítka a hustoty pixelů na plátně na základě aktuálního a referenční rozlišení obrazovky. Referenční rozlišení je nastaveno na 1920×1080. Pokud je velikost obrazovky různá od referenčního, dojde k úpravě velikostí u všech prvků na plátně.



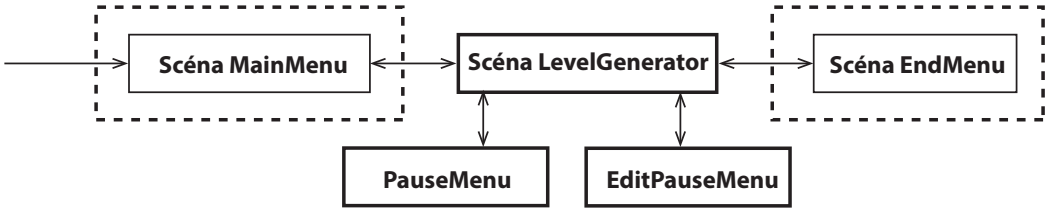
Obrázek 5.1: Diagram propojení jednotlivých scén v Unity. Šipky znázorňují přepínání mezi nimi. Scéna MainMenu je výchozí a načte se vždy po spuštění hry. Popis přepínání uvnitř scény je znázorněno na obrázku 5.3. Z této scény jsou dostupné scény předvytvořených úrovní (obrázek 5.2) a scéna LevelGenerator (obrázek 5.4). Scéna EndMenu je načtena po dokončení úrovně a lze z ní přejít zpět na scénu úrovně nebo MainMenu (obrázek 5.5).



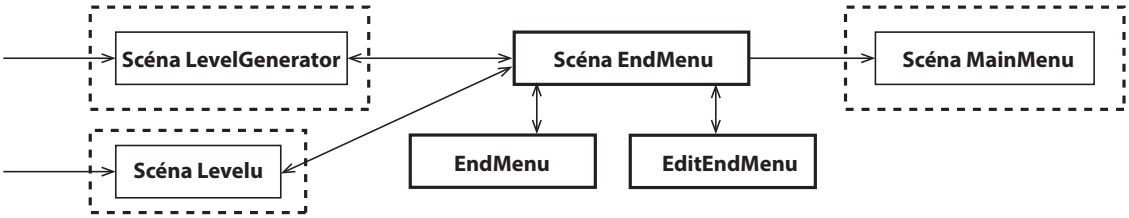
Obrázek 5.2: Diagram přechodů ve scénách předvytvořených úrovní. V obrázku jsou přechody do dalších scén znázorněny čárkovanými obdélníky.



Obrázek 5.3: Diagram přechodů mezi menu ve scéně MainMenu. Po spuštění hry je zobrazeno hlavní menu (na obrázku 5.7).



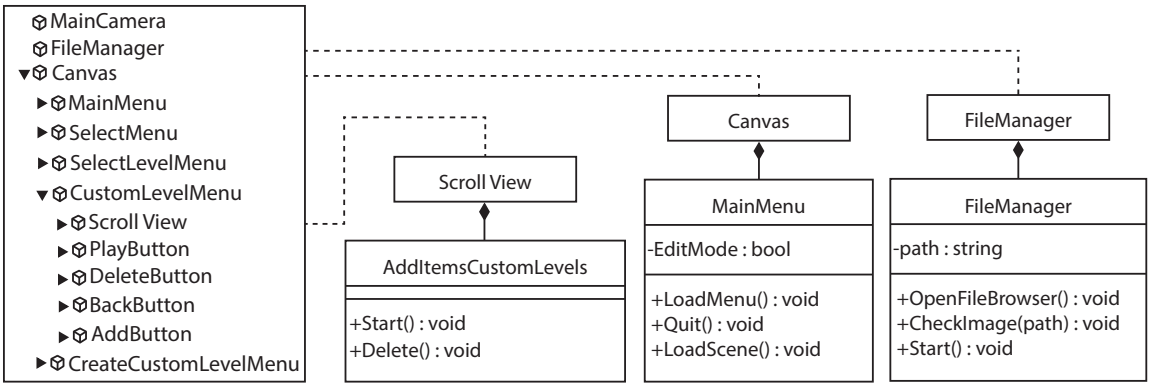
Obrázek 5.4: Diagram přechodů ve scéně LevelGenerator. Scéna je přístupná ze scény MainMenu. Na základě hodnoty proměnné `Edit` je zpřístupněno PauseMenu nebo EditPauseMenu.



Obrázek 5.5: Diagram přechodů ve scéně EndMenu. Scéna je přístupná ze scén předvytvořené úrovně a LevelGenerator. Na základě hodnoty proměnné `Edit` je zpřístupněno EndMenu nebo EditEndMenu.

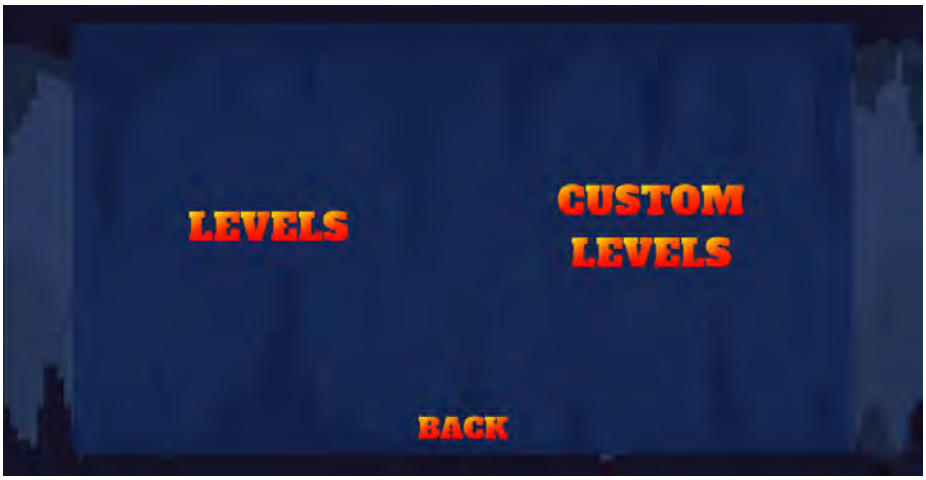
5.1.1 Scéna MainMenu

Tato scéna slouží pouze k zobrazení uživatelského rozhraní. Diagram přechodů je zobrazen na obrázku 5.3. Po spuštění hry je aktivováno hlavní menu, které obsahuje tlačítka „PLAY“, „OPTIONS“ a „QUIT“ (na obrázku 5.7). Mimo komponentu Canvas obsahuje prázdný objekt FileManager, ke kterému je přidána třída `FileManager` (na obrázku 5.6). Tato třída obsahuje deklaraci proměnné `path`, která uchovává cestu k úrovni, se kterou se právě pracuje.

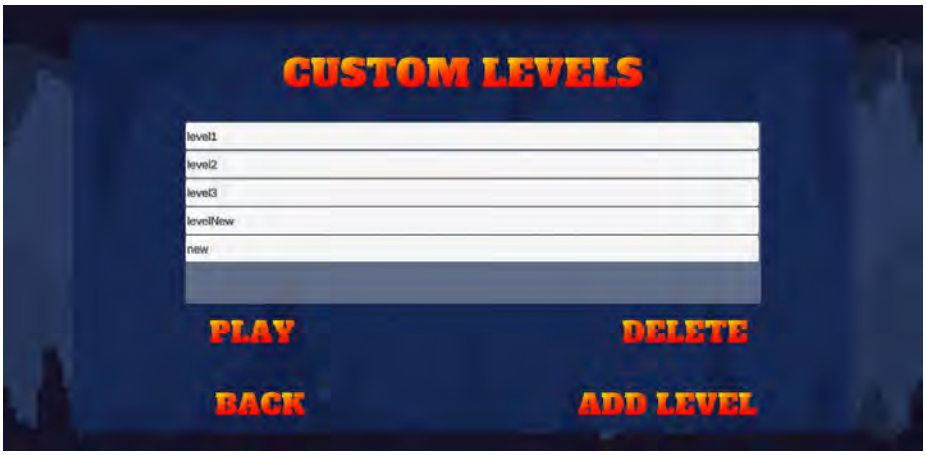


Obrázek 5.6: Na obrázku se nachází částečné zobrazení hierarchie herních objektů scény MainMenu. Zobrazeny jsou pouze objekty, jejichž funkce je popsána v této kapitole. K těmto objektům jsou připojeny třídy. Například k hernímu objektu Canvas je připojena komponenta MainMenu.

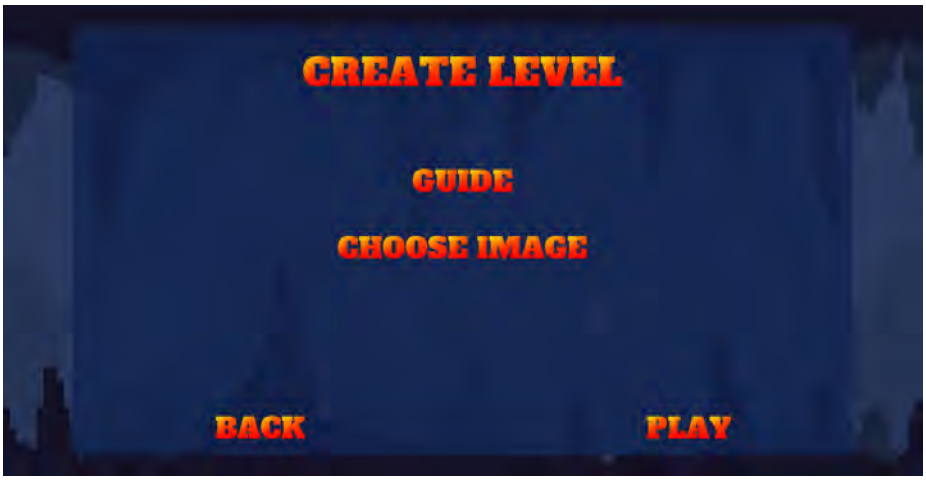
Obrázek 5.7: Snímek obrazovky hlavního menu ve scéně MainMenu. Po stisknutí tlačítka „PLAY“ je deaktivováno hlavní menu a aktivováno SelectMenu (na obrázku 5.8). Tlačítko „OPTIONS“ aktivuje menu pro změnu nastavení hlasitosti. Po kliknutí na tlačítko „QUIT“ se zavolá funkce `Quit()`, která ukončí aplikaci.



Obrázek 5.8: SelectMenu umožňuje hráči vybrat si mezi ukázkovými a vlastními úrovněmi. Stisknutím tlačítka „LEVELS“ se deaktivuje objekt SelectMenu a aktivuje objekt SelectLevel. Ten obsahuje tlačítka s názvy předvytvořených úrovní, po jejichž stisknutí se načte scéna s danou úrovní. Po stisknutí tlačítka „CUSTOM LEVELS“ se deaktivuje objekt SelectMenu a aktivuje objekt CustomLevels (na obrázku 5.9). Tlačítko „BACK“ slouží k aktivaci předchozího menu a deaktivaci stávajícího.



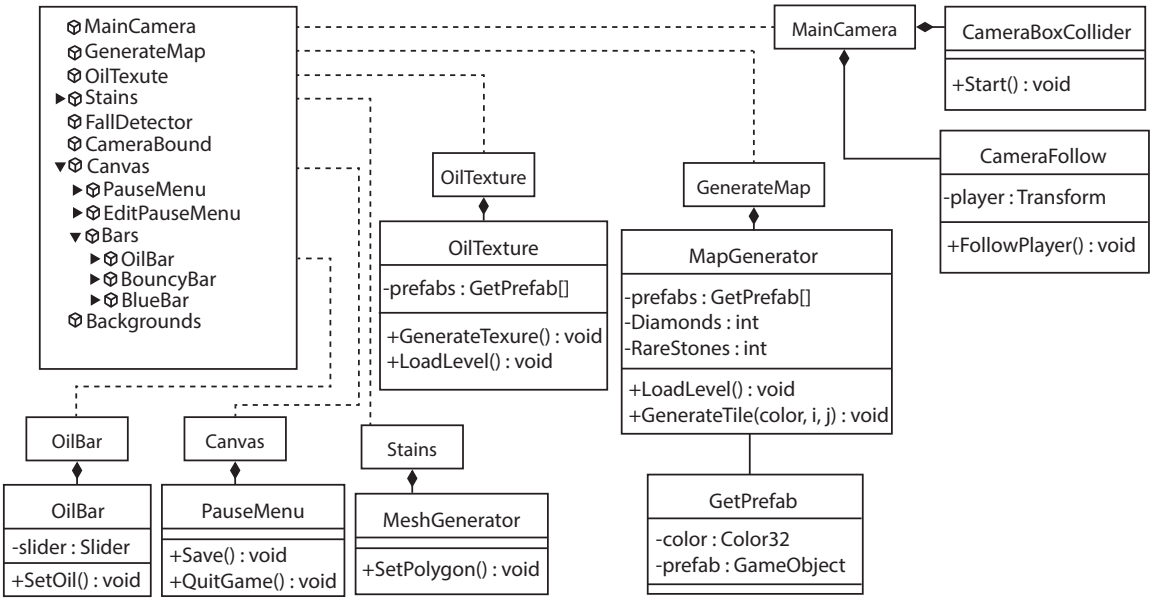
Obrázek 5.9: Menu CustomLevel slouží ke správě uživatelem vytvořených úrovní. Ve středu menu se nachází Scroll View, které zobrazuje úrovně vytvořené hráčem. K tomuto objektu je přidána třída `AddItemsCustomLevels`, která při aktivaci menu zkontroluje, zda existuje složka s úrovněmi, případně ji vytvoří. Pokud existuje, tak v ní nalezne všechny obrázky s příponou .png. Následně vždy vytvoří nové tlačítko se jménem daného souboru a přidá ho do obsahu ScrollView. Každé toto tlačítko má nastavený event `OnClick()`, který při stisknutí tlačítka nastaví do proměnné `FileManager.path` cestu k obrázku s názvem daného tlačítka a zpřístupní tlačítka „PLAY“ a „DELETE“. Pokud hráč stiskne tlačítko „PLAY“, je nastavena proměnná `MainMenu.EditMode` na false a načtena scéna LevelGenerator. Při stisknutí tlačítka „DELETE“ je zavolána funkce `AddItemsCustomLevels.Delete()`, která zničí tlačítko s vybranou úrovní a zároveň, pokud na cestě specifikované proměnnou `FileManager.path` existuje soubor, smaže jej. Stisknutím tlačítka „ADD LEVEL“ se deaktivuje objekt CustomLevel a aktivuje se objekt CreateCustomLevel (na obrázku 5.10).



Obrázek 5.10: CreateCustomLevel menu slouží k přidání nové úrovně. Při stisknutí tlačítka „GUIDE“ se otevře nápověda k vytvoření obrázku. Při stisknutí tlačítka „CHOOSE IMAGE“ se zavolá funkce `FileManager.OpenFileBrowser()`, která otevře prohlížeč souborů, ve kterém hráč zvolí obrázek, ze kterého se má úroveň vytvořit. Obrázek je následně zkontrolován a pokud vyhovuje požadavkům editoru, zpřístupní se tlačítko „PLAY“. Po jeho stisknutí je nastavená proměnná `MainMenu.EditMode` na true a načtena scéna LevelGenerator.

5.1.2 Scéna LevelGenerator

Ve scéně LevelGenerator se odehrává celá herní část i zkušební herní část editoru. Obrázek 5.11 zobrazuje hierarchii stěžejních herních objektů, které jsou použity ve scéně.



Obrázek 5.11: Na obrázku se nachází zobrazení hierarchie herních objektů scény LevelGenerator. K těmto objektům jsou připojeny třídy, jejichž funkce jsou popsány níže v této kapitole.

- **GenerateMap**
  K hernímu objektu je přidána třída `MapGenerator`, která vytvoří danou úroveň na základě obrázku, který je specifikován cestou uloženou v proměnné `FileManager.path` (viz 5.3).

- **OilTexture**
  Tento objekt slouží ke správě textury, která zobrazuje kaluže (viz 5.2.6) pomocí třídy `OilTexture`.

- **Stains**
  Objekt Stains funguje jako kontejner pro objekty představující jednotlivé kaluže.

- **FallDetector**
  Objekt FallDetector obsahuje komponentu BoxCollider2D, která má nastavenou volbu isTrigger na hodnotu true. Tato volba vyvolá event `OnTrigger` pokaždé, když se s tímto kolizním prvkem srazí jiný (viz sekce 3.4.5). FallDetector se nachází pod úrovní mapy a pohybuje se v ose x podle polohy postavy. Pokud postava spadne do mezery mezi plošinami a opustí tak herní pole, narazí na tento objekt, který vyvolá event `OnTriggerEnter2D`, ve kterém je postava přesunuta na počáteční pozici.

- **Canvas**
  Slouží k zobrazení ukazatele zásob daných kapalin (na obrázku 5.12) a k zobrazení PauseMenu (na obrázku 5.13).

Obrázek 5.12: Ukazatel zásob dané kapaliny je implementován pomocí komponenty Slider. Ta má pro každou kapalinu nastavenou vlastní maximální hodnotu a přidanou třídu, která tuto hodnotu aktualizuje, viz. obrázek 5.11 a objekt OilBar.

(a) PauseMenu dostupné při hraní úrovně. Umožňuje pozastavit hru a opět ji spustit stisknutím klávesy „ESC“ nebo pomocí tlačítka „RESUME“. Tlačítko „RESTART“ načte znovu danou úroveň. Po stisknutí tlačítka „MENU“ je načtena scéna MainMenu 5.1.1. Tlačítko „QUIT“ po stisknutí zavolá funkci `QuitGame`, která ukončí hru.

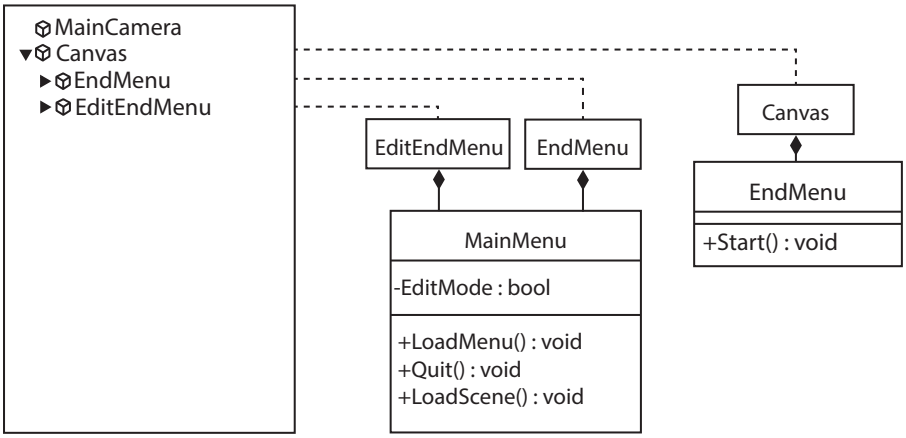
(b) Snímek obrazovky EditPauseMenu. Spouští se stejně jako PauseMenu. Tlačítko „RELOAD“ načte znovu danou scénu. Při stisknutí tlačítka „SAVE“ se zavolá funkce `PauseMenu.Save()`, která otevře prohlížeč souborů a získá od hráče název nové úrovně. Následně je obrázek, ze kterého byla úroveň vytvořena, překopírován do složky Levels pod daným názvem a je načtena scéna MainMenu.

Obrázek 5.13: PauseMenu je dostupné ve všech scénách s úrovněmi. V momentě, kdy hráč stiskne tlačítko „ESC“, dojde k pozastavení hry a je aktivován objekt PauseMenu na základě proměnné `MainMenu.EditMode`. Pokud je nastavena na hodnotu true, je aktivováno EditPauseMenu. Také dojde k deaktivaci objektu s ukazatel zásob kapaliny. Po stisknutí klávesy „ESC“ nebo tlačítka „RESUME“ je tento objekt deaktivován a objekt s ukazatelem zásob kapaliny znovu aktivován.

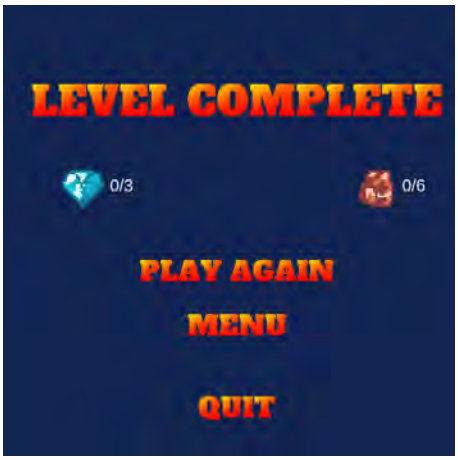
- **CameraBound**
  K objektu CameraBound je přidána komponenta BoxCollider2D. Tento objekt slouží k nastavení hranic úrovně. Pokud se postava nachází v blízkosti okraje úrovně, kamera je zastavena o tyto hranice (viz 5.2.1).

5.1.3 Scéna EndMenu

Scéna EndMenu je načtena po dosažení cíle a slouží k zobrazení EndMenu menu (na obrázku 5.15). Hierarchie herních objektů nacházejících se ve scéně je znázorněna na obrázku 5.14.



Obrázek 5.14: Na obrázku se nachází zobrazení hierarchie herních objektů scény EndMenu a tříd k nim připojených.



(a) EndMenu zobrazuje počet získaných drahých kamenů a tlačítka pro ukončení hry, návratu do hlavního menu a znovuspouštění hry.

(b) EndMenu v Edit módu umožňuje znovu načíst úroveň, uložit ji a navrátit se do hlavního menu.

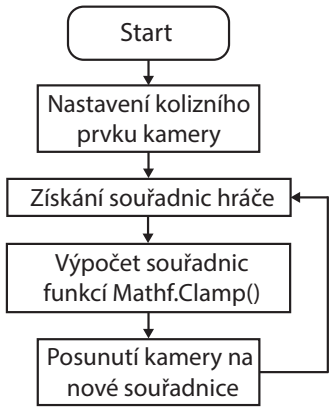
Obrázek 5.15: Scéna EndMenu zobrazí menu na základě proměnné `MainMenu.EditMode`. Pokud hráč zkouší novou úroveň (`EditMode = true`), načte se menu 5.15b, jinak 5.15a.

5.2 Herní část

Tato sekce se věnuje popisu implementace herní části, ve které hráč ovládá postavu a překonává úrovně. Skládá se z popisu tvorby animací, pohybu a implementace herních mechanik popsaných v kapitole 4.

5.2.1 Kamera

K objektu MainCamera jsou přidány třídy `CameraFollow` a `CameraBoxCollider`. Implementace pohybu kamery je znázorněna na obrázku 5.16.

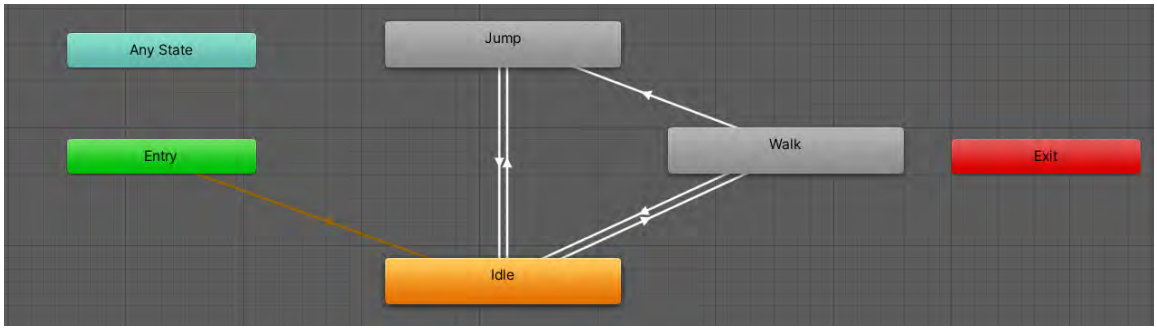


Obrázek 5.16: Třída `CameraBoxCollider` nejprve nastaví velikost kolizního prvku přidaného ke kameře, na základě velikosti okna obrazovky. Třída `CameraFollow` řídí pohyb kamery. Nejprve jsou získány nové souřadnice funkcí `Mathf.Clamp()`. Této funkci jsou zadány postupně obě souřadnice postavy a rozsah, ve kterém se může pohybovat kamera. Pokud je zadaná souřadnice menší než rozsah, je vrácena minimální hodnota rozsahu a obdobně, když je souřadnice větší než rozsah. Tímto způsobem se kamera nepohne za okraj úrovně.

5.2.2 Animace

Herní objekty v Unity engine mohou používat animační klipy prostřednictvím komponenty Animator. Animace jsou uspořádány v podobě stavového automatu – Animator Controller (na obrázku 5.17). Stavy v tomto automatu představují jednotlivé animační klipy. Každému přechodu mezi stavy lze nastavit proměnnou, která ovlivní, kdy se má začít přehrávat daná animace.

Ve hře jsou animace vytvořeny dvěma způsoby. Prvním je „Sprite Sheet“ animace (na obrázku 5.18), což je kolekce jednotlivých obrázků (nazývané Sprite) uspořádaných do mřížky. Z těchto obrázků je následně sestaven animační klip, který přehrává jednotlivé obrázky postupně za sebou. Druhým způsobem je animace kostry. Pomocí nástroje PSD Importer lze importovat části postavy přímo z vícevrstvé předlohy. Importér vygeneruje prefabrikát postavy na základě vrstev zdrojového souboru. Tomuto prefabrikátu lze v modulu Sprite Editor přidat kostru. Animační klipy následně vznikají animací polohy této kostry. Tímto způsobem je vytvořena postava a animace horníka.



Obrázek 5.17: Animator Controller přidaný k hráčově postavě.

Obrázek 5.18: Ukázka obrázků, ze kterých je vytvořena Sprite Sheet animace nepřítele.

5.2.3 Pohyb postavy

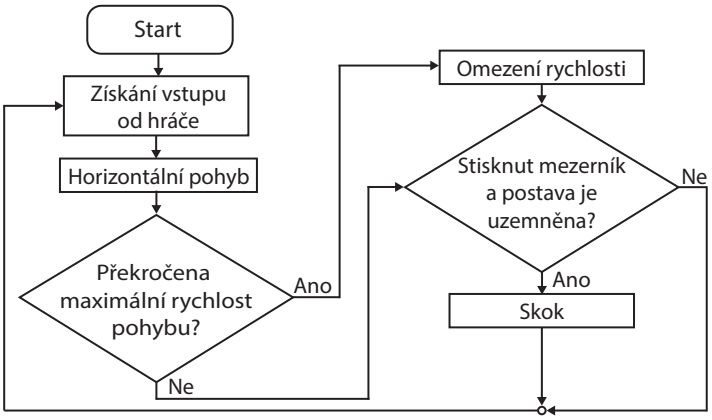
Pohyb postavy je řízen třídami `PlayerMovement` a `GunAim` (posána v sekci 5.2.6), které jsou přiloženy k prefabrikátu Player. Na obrázku 5.19 je popsán proces, při kterém je postava posunuta.

5.2.4 Kostka

Každá kostka obsahuje komponentu BoxCollider2D, ke kterému je přidán Physics Material 2D. Fyzikální materiál slouží k úpravě efektů tření a odrážení, ke kterým dochází mezi objekty při jejich kolizi. U 2D materiálu nelze měnit tyto fyzikální vlastnosti pouze jednomu objektu (jako je tomu u 3D). K materiálu se dá přistoupit pouze skrze funkci `Collider.sharedMaterial()`, která změní vlastnosti u všech kolizních prvků, které tento materiál používají. Při vzniku kostky je proto instancován[1] Physics Material 2D a přiřazen ke komponentě BoxCollider2D.

Aby nebylo možné s kostkou pohnout, je velikost tření u fyzikálního materiálu nastavena na maximální hodnotu. U komponenty Rigidbody2D je nastavena hmotnost tělesa tak, aby nestačila síla vyvinutá hráčem k jejímu posunutí.

[1] Instancování – vytváření kopií daného objektu.



Obrázek 5.19: Jednoduchý diagram popisující pohyb postavy. Třída `PlayerMovement` ve smyčce provádí tyto operace: získá informaci o stisknutých klávesách hráčem, provede pohyb v daném směru a provede kontrolu rychlosti pohybu. Komponenta Rigidbody2D, připojená k postavě, umožňuje tělesu přidat rychlost v daném směru pomocí funkce `AddForce`. Stisknutí tlačítka „mezerník“ způsobí výskok postavy, pokud postava stojí na platformě nebo kostce. To je detekováno funkcí `Physics2D.BoxCast()`, která vytvoří neviditelný kolizní box ve vybrané vrstvě a vrátí informaci o kontaktu s jinými kolizními prvky.

5.2.5 Láva

K hernímu objektu láva jsou přidány dva kolizní prvky. První z nich má stejné rozměry, jako láva a nastavenou volbu isTrigger na hodnotu true. Při srážce s kolizním prvkem postavy je hráč přemístěn na startující pozici v úrovni. Druhý Collider nezabírá celou výšku herního objektu a slouží jako záchytný bod, o který je zastavena kostka.

5.2.6 Kapaliny

Hlavní herní mechaniky ve hře představují kapaliny. Ty jsou implementovány částicovými systémy (viz 3.4.4). Každá kapalina je tvořena vlastním částicovým systémem, který je specificky nastaven. Všechny tyto systémy jsou umístěny v hierarchii prefabrikátu hráče a jsou potomky herního objektu, který obsahuje kost paže. Pohyb této paže ovládá třída `GunAim`, která je přidána k prefabrikátu postavy. V každé funkci `Update()` se spočítá nová hodnota rotace paže, na základě pozice myši. Zároveň je postava vždy otočena čelem ke směru, ve kterém se nachází kurzor.

První verze vytvářela kapalinu instancováním herního objektu, který vypadal jako částice kapaliny. Po vytvoření desítek těchto objektů docházelo k zasekávání hry, proto byl pro implementaci kapaliny použit částicový systém.

Nastavení částicového systému

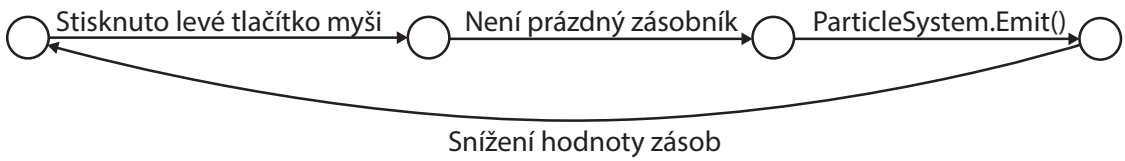
V Unity editoru je u částicových systému zásadní nastavení následujících modulů:

- Main modul – obsahuje globální vlastnosti, které ovlivňují celý systém. Částicím jsou zde nastaveny vlastnosti jako počáteční velikost a rychlost nebo modifikátor gravitace. Důležité je nastavení vlastnosti *Simulation Space*, které řídí, v jakém prostoru jsou částice animovány. To je nastaveno na *Wordl*. Částice se pohybují ve světovém prostoru a nejsou po emitování ovlivněny rodičovským objektem.

- Shape modul – definuje objem, z něhož mohou být částice emitovány a směr počáteční rychlosti. Vlastnost *Shape* nastavena na *Cone* definuje kužel jako tvar tohoto objemu, jehož poloměr se blíží k nule. Částice jsou tak emitovány za sebou v jedné přímce.
- Collision modul – řídí kolizi částic s herními objekty. Vlastnosti *Dampen* a *Bounce* ovlivňují rychlost částic po srážce. Každý částicový systém koliduje s objekty ve vybraných vrstvách, které jsou nastaveny vlastností *Collides With*. Nejdůležitější je zaškrtnutí možnosti *Send Collision Messages*, která umožňuje detekovat kolize ze skriptů pomocí funkce `OnParticleCollision()`.

Střílení kapaliny

Ke každému částicovému systému je připojena vlastní třída (`ParticleLauncher` žlutá, `BluePLauncher` modrá a `BouncyLauncher` fialová kapalina). Postup procesu střelby kapaliny je popsán na obrázku 5.20.



Obrázek 5.20: Zjednodušený diagram popisující výstřel kapaliny. Při stisknu levého tlačítka myši je emitována částice vybrané kapaliny (pokud není prázdný zásobník) pomocí funkce `ParticleSystem.Emit(počet)`. Pokud hráč střílí modrou kapalinu, při stisku je emitována vždy pouze jedna částice. V ostatních případech jsou částice emitovány, dokud je tlačítko stisknuto a vytváří tak souvislý proud kapaliny. Zároveň je zavolána funkce, která pro vybranou kapalinu sníží hodnotu ukazatele zásob (nastavení hodnoty `Slider.value`).

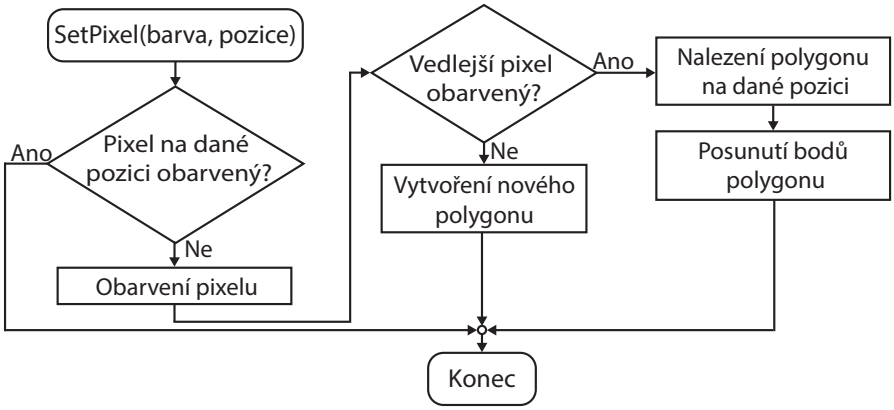
Při kolizi částice s kolizním prvkem objektu, který se nachází v jedné z vybraných vrstev, je zavolána funkce `OnParticleCollision()`. Třída `BluePLauncher` v této funkci získá odkaz na komponentu Rigibody2D kostky, se kterou se částice střetla a přidá jí rychlost funkcí `AddForce()`. Ve třídách `ParticleLauncher` a `BouncyLauncher` je při kolizi zavolána funkce `Physics2D.Raycast()`. Tato funkce v místě kolize částice vykreslí neviditelný paprsek směrem dolů. Pokud tento paprsek protne platformu, je zavolána funkce `OilTexture.SetPixel()` (viz sekce 5.2.6).

Vykreslování kaluží

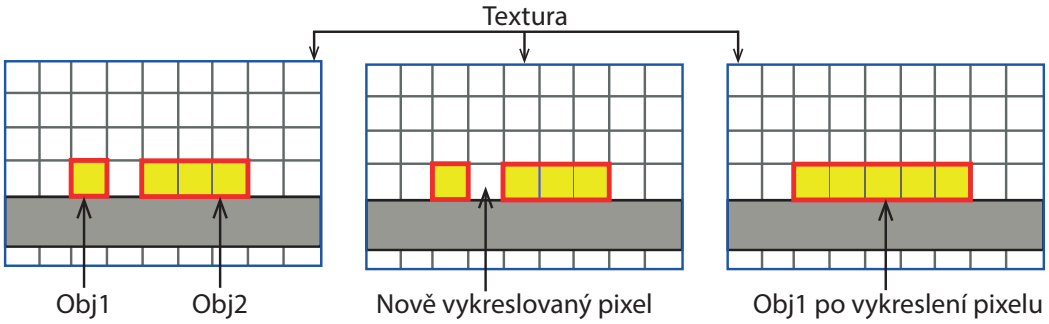
K zaznamenání informace o tom, kudy ve hře protekla kapalina, slouží třída `OilTexture` přidaná k hernímu objektu OilTexture (viz 5.1.2). Funkce `GenerateTexture()` na začátku hry vytvoří novou texturu o velikosti dané úrovně a umístí ji přes celou úroveň. Vykreslování pixelů do této textury je znázorněno na obrázku 5.21.

Při vykreslování pixelů vedle sebe, nevznikají ve scéně nové herní objekty kaluží. Funkce `MeshGenerator.SetPolygon()` získá odkaz na objekt kaluže a posune rohové body polygonu, které definují tvar kolizního prvku (na obrázku 5.22).

[2]Komponenta renderer slouží k zobrazení objektu na obrazovce.



Obrázek 5.21: Funkce `SetPixel()` vykreslí barevný pixel v textuře na místě, kde se srazila částice s platformou. Současně na daném místě ve scéně vznikne nový herní objekt s komponentou PolygonCollider2D. K jejich správě slouží třída `MeshGenerator`. Tyto herní objekty ve scéně představují kaluže. Nemají žádnou komponentu Renderer[2], tudíž nejsou vidět. Nacházejí se ve scéně na místech za vykreslenými pixely textury.



Obrázek 5.22: Obrázek popisuje práci s herními objekty, které reprezentují kaluže (červené obdélníky Obj1 a Obj2). Mřížka na obrázku představuje texturu, do které se vykreslují barevné pixely (žluté čtverečky). Obrázek vlevo ukazuje situaci, kdy existují vedle sebe dva objekty (dvě řady pixelů) oddělené pouhým pixelem. Při vykreslení pixelu mezi tyto objekty, na obrázku uprostřed, jeden z herních objektů je zničen (Obj2) a druhý je upraven tak, aby velikost polygonu (PolygonColliderů) odpovídala řadě vykreslených pixelů (obrázek vpravo).

Pohyb v kalužích

Při srážce kolizních prvků hráče a žluté kaluže, je ve funkci `OnTriggerEnter2D()` zvýšena maximální rychlost, kterou se může postava pohybovat. První verze pohybu postavy spočívala ve změně rychlosti komponenty Rigidbody. Postava pak doskočila na vzdálená místa, když stála v nejmenší možné kaluži. Rychlost je proto zvyšována postupným přidáváním sil. Hráč se musí pohybovat v kaluži, aby získal rychlost potřebnou k dlouhému skoku.

Při srážce kolizních prvků kostky a žluté kaluže, je ve funkci `OnTriggerEnter2D()` snížena hodnota tření fyzikálního materiálu a hmotnost tělesa. Pokud kostka opustí kaluži, hmotnosti i tření jsou nastaveny původní hodnoty.

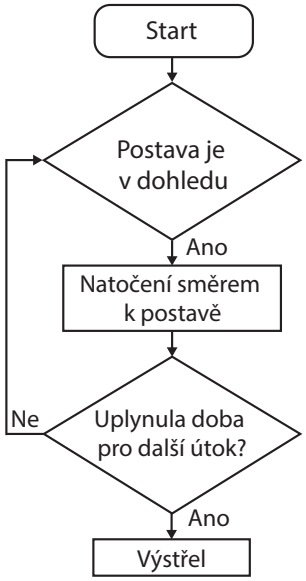
Srážka kolizních prvků postavy (či kostky) a fialové kaluže vyvolá event `OnCollision`. Ve funkci `OnCollisionEnter2D()` je komponentě Rigidbody2D přidána impulzní síla ve směru nahoru. Tato síla se zvyšuje při každém dotyku kaluže, dokud nedosáhne maximální hodnoty nebo dokud se objekt nedotkne platformy.

5.2.7 Zdroje kapalin

Každý zdroj obsahuje komponentu CircleCollider2D s nastavenou volbou IsTrigger na hodnotu true. Při kolizi s postavou je herní objekt deaktivován funkcí `SetActive(false)` a vybrané kapalině je zvýšena hodnota zásob.

5.2.8 Nepřítel

K ovládání nepřítele, v podobě draka, slouží třída `Dragon`. Jakmile se hráč přiblíží na dostatečnou vzdálenost k drakovi, začne se přehrávat animace střelby, při níž je instancována střela ve směru pohybu hráče. Pokud střela hráče zasáhne, je posunut na startující pozici v úrovní. Diagram na obrázku 5.23 popisuje postup střelby draka.



Obrázek 5.23: K drakovi je přidána komponenta BoxCollider2D, která slouží jako hranice viditelnosti. Drak střílí jednu ohnivou kouli každé dvě vteřiny. Pokud se uvnitř hranice nachází postava a pokud uplynul dostatečný čas od poslední střelby, drak po ní v horizontálním směru střílí ohnivé koule.

5.2.9 Osvětlení

Hra využívá funkcí URP, proto lze použít komponentu Light 2D (viz. 3.4.3). V herní části se nenachází globální zdroj světla. Hráč vidí díky bodovému světlu, které je potomkem prefabrikátu postavy. To se pohybuje spolu s postavou a umožňuje tak hráči vidět pouze tu část úrovně, ve které se nachází. Dalšími zdroji světla jsou freeform světla umístěná jako potomci lávových a vzácných kamenů.

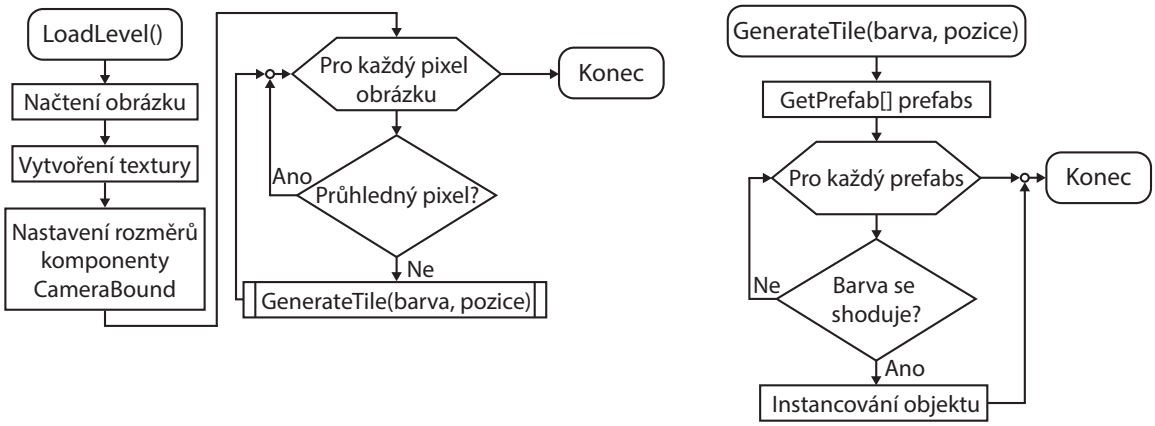
5.3 Editor

Editor ve hře slouží k přidávání nových úrovní. Přistupuje se k němu skrze hlavní menu (na obrázku 5.10). Třída `FileManager` získá obrázek s nakreslenou úrovní od hráče a provede na něm kontrolu. V obrázku musí existovat právě jeden pixel odpovídající prefabrikátu postavy a alespoň jeden odpovídající konci úrovně. Při správném rozlišení obrázku, existenci postavy a konce úrovně, je zpřístupněno tlačítko „PLAY“ a nastavena proměnná `path`, ve které je uložena cesta k obrázku. Po stisknutí tlačítka „PLAY“ je nastavena proměnná `MainMenu.EditMode` a načtena scéna LevelGenerator, která neslouží pouze k vytvoření nové úrovně, ale vytváří všechny úrovně, které hráč přidal.

Hráč má možnost vyzkoušet si úroveň v „Edit“ módu. Hra se ovládá a funguje stejně jako již vytvořené úrovně. Rozdíl je v možnostech, které nabízí PauseMenu (obrázek 5.13) a EndMenu (obrázek 5.15). Pokud hráč provede změnu ve zdrojovém obrázku, úroveň vytvoří znovu tlačítkem „RELOAD“ v PaseMenu. Úroveň uloží stisknutím tlačítka „SAVE“, které se nachází v obou menu.

5.3.1 Generování úrovně

Ke generování úrovně slouží třída `MapGenerator` a `GetPrefab`. Třída `GetPrefab` navazuje danou barvu na herní objekt. Třída `MapGenerator` následně vytváří pole prvků třídy `GetPrefab`. Generování úrovně je popsáno na obrázku 5.24.



Obrázek 5.24: Zjednodušený vývojový diagram generování úrovně z rastrového obrázku. Ve funkci `LoadLevel()` je otevřen obrázek uložený v cestě dané proměnnou `FileManager.path` a přečten jeho obsah. Na základě velikosti obrázku je vytvořena textura třídy `OilTexture` a jsou nastaveny rozměry komponenty BoxCollider2D objektu CameraBound (objekt popsán v kapitole 5.1.2). Postupně je zpracován každý pixel obrázku následovně. Pokud pixel není průhledný, je jeho barva a pozice v obrázku předána funkci `GenerateTile()`. Tato funkce projde pole prvků třídy `GetPrefab` a porovná proměnnou `GetPrefab.color` s barvou pixelu. Při shodě je instancován prefabrikát herního objektu přiřazený k proměnné `GetPrefab.prefab` dané barvy. Pixely s jinou hodnotou barvy, než předdefinované barvy prefabrikátů jsou ignorovány. Nová instance herního objektu (vyjma postavy hráče) je v hierarchii umístěna jako potomek objektu GenerateMap. Při instancování prefabrikátu postavy je třídě `CameraFollow` předán odkaz na pozici hráče, kterou kamera následuje. Každá instance prefabrikátů diamantu a vzácného kamene je zaznamenána a jejich počet je uložen do statických proměnných. K těmto proměnným přistupuje třída `ScoreBoard` ve scéně EndMenu.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout herní mechaniky využívající kapaliny, implementovat je a vytvořit herní demo s několika úrovněmi, které je využívají. Jak bylo určeno v zadání práce, byl prostudován herní engine Unity a prozkoumány existující hry využívající simulaci kapalin. Následně byly navrženy herní mechaniky v logických hádankách a herní editor. Po návrhu byly tyto mechaniky implementovány a použity k tvorbě několika úrovní.

Vývoj dema začal vytvářením kapaliny. Nejprve jsem vyzkoušela instancování herního objektu kapaliny, SPH simulaci s Unity Job systémem nebo použití renderovací textury. Tyto verze nefungovaly podle mých představ, proto jsem začala pracovat s částicovým systémem. Hledání a zkoušení možností simulace kapaliny zabralo podstatnou část času. Po správném nastavení částicového systému byly přidány animace a pohyb postavy. Následovala fáze implementace zaznamenávání průchodu kapaliny scénou a tvorba editoru. V této fázi byl testován editor při každém spuštění hry, jelikož neexistovala scéna s předvytvořenými úrovněmi. Dále vznikaly herní mechaniky a scény s uživatelským rozhraním. K animacím se přidaly zvuky a vytvořily se scény s předvytvořenými úrovněmi.

Výsledkem této práce je herní demo, využívající herní mechaniky založené na kapalinách. Obsahuje několik předvytvořených úrovní a editor, který umožňuje hráči přidat vlastní úrovně. Bylo vytvořeno demonstrační video, které zobrazuje implementované herní mechaniky ve hře. Hra byla zveřejněna na stránkách itch.io[1].

Při výběru zadaní práce jsem neměla žádné zkušenosti s tvorbou her, herním enginem Unity ani s programovacím jazykem C#. Postup vývoje hry byl proto ze začátku velmi pomalý. Najít řešení problémů, teď už jednoduchých, trvalo někdy několik dní. Přes všechny tyto nepříznivé okamžiky a nespočet probdělých nocí, jsem rozhodnuta se hernímu vývoji nadále věnovat.

V práci bych chtěla pokračovat dál, přidáním dalších herních mechanik. Jednou z nich jsou louče, které osvěcují úroveň po omezenou dobu. K jejich zapálení by musel hráč posunout kostku, která zatíží tlačítko. Dále bych upravila stávající animace a rozšířila je, například o animaci pádu a přidala další vizuální efekty. Věřím, že když se budu vývoji i nadále věnovat, realizuji postupně všechny své nápady a z tohoto dema vzejde zajímavá originální hra.

[1] https://itch.io/

Literatura

[1] Aversa, D. a Dickinson, C. *Unity Game Optimization: Enhance and Extend the Performance of All Aspects of Your Unity Games*. 3. vyd. Packt Publishing Ltd., 2019. 158–162 s. ISBN 978-1-83855-651-8.

[2] Borromeo, N. A. *Hands-On Unity 2021 Game Development*. 2. vyd. Packt Publishing Ltd., 2021. ISBN 978-1-80107-148-2.

[3] Chen, J. a Price, E. *Game Development with Unity for .NET Developers: The Ultimate Guide to Creating Games with Unity and Microsoft Game Stack*. 1. vyd. Packt Publishing Ltd., 2022. ISBN 978-1-80107-807-8.

[4] Gregory, J. *Game Engine Architecture*. 1. vyd. A K Peters/CRC Press, 2009. ISBN 978-1-4398-6526-2.

[5] Grey, S. *Mind-Melding Unity and Blender for 3D Game Development*. 1. vyd. Packt Publishing Ltd., 2021. ISBN 978-1-80107-155-0.

[6] Koster, R. *Theory Of Fun For Game Design*. 2. vyd. O'Reilly Media, 2013. ISBN 978-1-449-36321-5.

[7] Lemarchand, R. *A Playful Production Process: For Game Designers (and Everyone)*. 1. vyd. MIT Press, 2021. ISBN 9780262045513.

[8] Lowood, H. Game Engines and Game History. *History of Games International Conference Proceedings*. 1. vyd. KINEPHANOS. Únor 2014, č. 1, s. 181–182. ISSN 1916-985X. Dostupné z: https://www.kinephanos.ca/2014/game-engines-and-game-history/.

[9] Raappana, J. *Great Particles and how to make them*. 2018. Bakalářská práce. Oulu University of Applied Sciences. Dostupné z: https://www.theseus.fi/bitstream/handle/10024/156135/Raappana_Joona.pdf?sequence=1&isAllowed=y.

[10] Stefyn, N. *How video games are made: the game development process* [online]. CG Spectrum, 2022 [cit. 2023-04-20]. Dostupné z: https://www.cgspectrum.com/blog/game-development-process.

[11] Unity Technologies. *UNIVERSAL RENDER PIPELINE (URP)* [online]. 2021 [cit. 2023-04-20]. Dostupné z: https://unity.com/srp/universal-render-pipeline.