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

NÁVRH A IMPLEMENTACE 2D HRY V UNITY

DESIGN AND IMPLEMENTATION OF A 2D GAME IN UNITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN DVOŘÁČEK**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. MARTIN ČADÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147454

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Dvořáček Martin**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Návrh a implementace 2D hry v Unity**
Kategorie:	Počítačová grafika
Akademický rok:	2022/23

Zadání:

1. Prozkoumejte historii a aktuální stav herního návrhu v žánru 2D her.
2. Seznamte se s herním enginem Unity s ohledem na tvorbu 2D her.
3. Navrhněte novou 2D hru na základě získaných poznatků.
4. Navrženou hru implementujte a v postupných iteracích experimentuje s různými návrhy a herními mechanikami.
5. Prezentujte výslednou hru formou plakátu a krátkého videa.

Literatura:

- Koster, Raph. Theory of fun for game design. O'Reilly Media, Inc., 2013.
- Schell, Jesse. The Art of Game Design: A book of lenses. CRC press, 2008.
- Unity Learn. Unity, https://learn.unity.com/.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3 zadání.

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

Vedoucí práce:	**Čadík Martin, doc. 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 této práce je vytvořit 2D rougelike platformovací hru se základním chováním nepřátel a skládáním úrovní z modulů. Hra má být celkem modifikovatelná a bude tvořena v herním enginu Unity.

Abstract

The goal of this thesis is to create 2D roguelike platforming game with basic enemy behavior and creating levels randomly from modules. Game is also supposed to be quite modifiable and will be made in game engine Unity.

Klíčová slova

2D, Hra, Roguelike, Roguelite, Plošinovka, Hack'n'slash, Dungeon

Keywords

2D, Game, Roguelike, Roguelite, Platformer, Hack'n'slash, Dungeon

Citace

DVOŘÁČEK, Martin. *Návrh a implementace 2D hry v Unity.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Martin Čadík, Ph.D.

Návrh a implementace 2D hry v Unity

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Martina Čadíka Ph.D.. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Dvořáček
8. května 2023

Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Martinovi Čadíkovi Ph.D. za jeho vhodné a trefné připomínky a celkové vedení práce.

Obsah

Kapitola 1

Úvod

Prvně by bylo příhodné definovat, co vlastně hra je. Hry mají mnoho definicí. Například: Sid Meierova je "Série významných rozhodnutí", ale jak Raph Koster poukazuje, prakticky žádná definice neobsahuje faktor zábavy. Podle jeho popisu jsou hry výborné pomůcky pro učení, jelikož se jedná o před připravené a zjednodušené úryvky, které mozek lehčeji vstřebá. Toto učení zaměstnává mozek a zabavuje nás. Ale při vytváření můžou vzniknout problémy kazící zábavu. Stylu: hra je příliš lehká, těžká nebo pomalá. A z toho vychází, že perfektní hra by měla hráče zabavit dokud neobjeví vše co hra nabízí. Mé chápání jeho definice by znělo následovně: Interaktivní, naučné a zjednodušené struktury s postupně zvyšující se obtížností [6].

Videohry jsou součástí naší historie již pěknou řádku let. Jejich historie se dá dohledat až do dob krátce po druhé světové válce nebo Pongu roku 1970 na konzolách prvních generací. Hry byly postupně vyvíjené z jednoduchých do výrazně složitějších, dělených do různých žánrů a pod žánrů. Mezi takové vyvinuté významné hry patří i Rogue, který dal vznik fascinujícímu žánru roguelike. I když většina mé generace, včetně mě, Rogue nikdy nehrála a ani o něm neslyšela, poznali jsme hry z žánru roguelike, který minimálně mne fascinoval na tolik, abych se pokusil vytvořit vlastní výtvor z volnější varianty tohoto žánru - roguelite. Samozřejmě fascinace žánrem nebyl jediný faktor mého rozhodnutí pracovat na téhle práci. Mezi další faktory patří moje celková afekce k videohrám a jejich vývoji, nad kterým často uvažuji když je hraji. Tudíž cílem této práce je vytvořit hru v žánru roguelite, kde bude kladen velký důraz na modularitu a znovuhratelnost.

Práce pojednává lehce o historii her, pro mne či pro žánr významných, a o mých volbách pro vývoj a i o nějakém tom popisu samotného vývoje. Tyto informace jsou rozděleny do 6 kapitol. Kde v 2.kapitole blíže seznamuji a definuji žánry, podle kterých bude hra formována. Též zde uvádím příklady některých historicky nebo pro mě významných her, i s pár historickými fakty.Ve 3.kapitole definuji, jaké funkce bude hra mít a hrubý popis toho, jak fungují a také zde definuji herní engine. Popisuji rozdíly mezi různými enginy, které jsou zdarma a píši, proč jsem si vybral Unity. Ve 4. kapitole popisuji praktickou implementaci. Poslední, 5. kapitola před závěrem popisuje, jak byla hra testována a upravována podle výsledků testování.

Kapitola 2

Seznámení s žánry

Herních žánrů a pod žánrů je mnoho. Tato kapitola je zaměřena na vysvětlení a definovaní žánrů využitých pro tento projekt.

2.1 Roguelike/Roguelite

Roguelike může být brán jako pod žánr hry na hrdiny, či anglicky role playing game (RPG), a dle populární Berlínské interpretace [18] je definován hodně specificky. Mezi významné faktory patří:

- Náhodně generovaný svět – Svět pokaždé vypadá jinak. Vždy když se hra zapne, vygeneruje se nový svět, který uživatel může objevovat.
- Permanentní smrt – Když hráč zemře, musí začít úplně od znova. Neočekává se, že hráč vyhraje s jedním charakterem. Hru je možné uložit, ale pokud ji hráč načte, soubor je smazán.
- Hra je tahová – Každý příkaz je jedna akce, nezáleží na tom, jak dlouho bude hráči trvat, než se pro danou akci rozhodne.
- Využití mřížky – Svět je reprezentovaný mřížkou, kde hráč zabírá jedno políčko. Pohyb není úplně volný, je podle mřížky.
- Nemodální – Všechny akce se nachází ve stejném módu. Každá akce by měla být vždy dostupná.
- Komplexnost – Hra by měla dovolovat několik řešení k jednomu problému.
- Spravovaní zdrojů – Zdroje jsou limitovány a hráč je musí využívat s uvážením.
- Hack'n'slash – Nedílnou součástí hry je zabíjení spousty monster, jedná se o styl hráč proti světu.
- Objevování – Úrovně musí být pečlivě prozkoumány.

Mezi méně významné faktory patří:

- Jeden Charakter – Svět je zaměřen na hráče a na pohled na svět skrz jeho postavu.
- Nepřátelé jsou podobní hráči – Pravidla, která se aplikují na hráče, se aplikují i na nepřátele.

- Taktická výzva – Hráč se musí postupně naučit taktiku, jak porážet výzvy. Díky tomu hráč pravděpodobně neporazí hru s prvním charakterem.
- ASCII zobrazení – Tradiční zobrazování je reprezentace mřížky ASCII znaky.
- Dungeons – Úrovně jsou složeny z chodeb a místností. Jelikož, když je řeč o hrách, tak v českém jazyce není pro anglické slovo dungeon, jen jeden překlad, tudíž se může jednat o jeskyni, kobku nebo dokonce i pevnost nepřítele tak může být označena.
- Informace o postavě – Čísla udávající hráče jsou záměrně ukázána.

Aby byla hra označena jako roguelike nemusí splňovat pedanticky všechny faktory. Tato definice může být považovaná za zastaralou a je například kritizována v článku "Screw the Berlin Interpretation!"od Darrena Grey [5]. Jelikož se mnoha lidem nelíbí části Berlínské interpretace, a tak vznikají podobné, upravené definice, například bez části jako ASCII vyobrazení. Některá pravidla jsou částečně sloučená nebo upravená, aby zahrnovala více situací. Mezi taková pravidla lze považovat permanentní následky nebo selhání. Toto pravidlo obsahuje permanentnost smrti hráčova charakteru, ale třeba i nenávratnost použití nějakého vybavení s omezeným počtem použití.

Roguelite či neo roguelike je téměř stejný jako roguelike, ale je zde poměrně malý rozdíl a to, že si charakter něco po smrti charakteru uchová. Zkušenosti, peníze, cokoliv, a to následně může využít na vylepšení či odemykaní nových věcí. Tudíž hráč vždy začíná s novou postavou, ale už může být vylepšená, nebo i jiná, úroveň může obsahovat nové předměty, nepřátele a podobně. Tento rozdíl se může zdát velmi malý, ale je velmi důležitý.

Samotná definice roguelike Berlínské interpretace je až z roku 2008, ale počátek roguelike začíná již v 70. až 80. letech 20. století. Roguelike je pojmenovaný podle jedné z prvních her s prvky žánru Rogue.

2.1.1 Příklady Roguelike/Roguelite her

Rogue

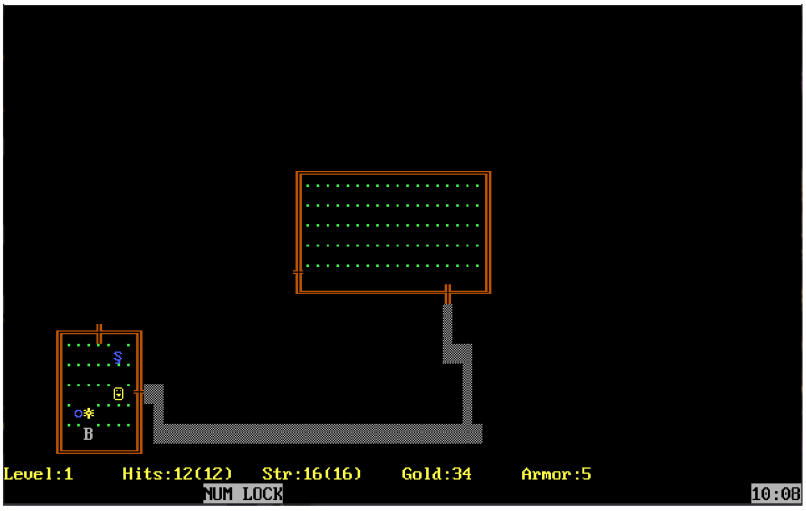
Tato dungeon-crawler hra vyšla roku 1980 a vytvořili ji programátoři Michael Toy a Glenn Wichman. Tím odstartovali celý nový žánr, dnes známý jako roguelike. Premisa hry byla velice jednoduchá, stejně jako její grafika. Vše bylo reprezentováno ASCII znaky na mřížce. Hráč byl označen jako @ a objevoval kobku, kde hledal Amulet Yendoru a po cestě nacházel monstra, předměty a zlaťáky.

Hra po každé smrti začala úplně od začátku s čerstvě vygenerovanou úrovní i efekty předmětů a kouzel. To znamenalo, že se nemohli lidé naučit jen trasu, ale museli se naučit, jak hru hrát a přizpůsobovat se situacím za běhu. Hra je dokonce open-source, což ji umožnilo dalším upravovat a přetvářet ji do jiných her.

Cílem hry je dostat se přes 26 úrovní a získat již zmíněný amulet Yendoru. Což je mu znepříjemněno náhodně rozloženými nepřáteli, a dokonce i únavou z objevování. Naopak je usnadněno přes předměty rozházené přes úrovně. [17].

Na obrázku 2.1 je vidět vzhled celé hry Rogue. Místnosti jsou vyobrazeny hnědými obdélníky a vstupy do daných místností jsou vyznačeny jako mezera v obdélníku označujícím místnost. Místnosti jsou propojeny šedými koridory. Místa v místnosti jsou zobrazeny, zelenými tečkami. Hráč je zobrazen žlutým "smajlíkem", modrá písmena a jiné znaky jsou

označení pro předměty k sebrání. Jiné žluté symboly jsou peníze a šedá písmena a symboly jsou nepřátelé. Úroveň se postupně odhaluje s postupem hráče. Pod úrovní lze vidět statistiky hráče, úroveň, životy, sílu, zlato a zbroj.



Obrázek 2.1: Výstřižek scény ze hry Rogue. Nejedná se o snímek z originální platformy, ale hru je možné zakoupit na platformě Steam.

The Dungeons of Moria

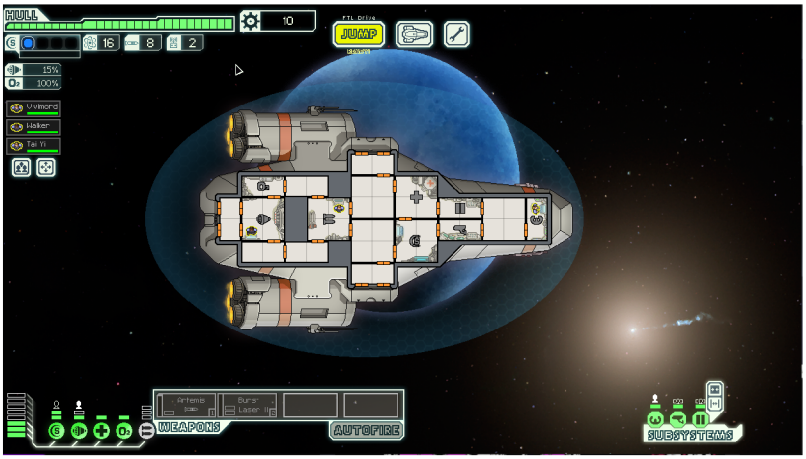
Hra z roku 1983 si zde zaslouží být také zmíněna, jelikož se jedná o jednu z prvních roguelike her, která navíc není vytvořená ze samotného Rogue. Mimo jiné je to první roguelike hra, jenž obsahuje úroveň s městem, kde se dá nakupovat vybavení, což by se dalo přirovnat k lobby tohoto projektu jistým způsobem. Celkem zajímavé je, že hra obsahuje výrazný prvek hry na hrdiny s vytvořením charakteru. Při vytváření charakteru si hráč vybírá rasu, pohlaví a další statistiky, pro představu by se dalo říci, že je to hodně podobné jako u celkem známého Dungeons & Dragons. Vzhledově je hra podobná Rogue, stále je vyobrazena v ASCII znacích.

FTL: Faster Than Light

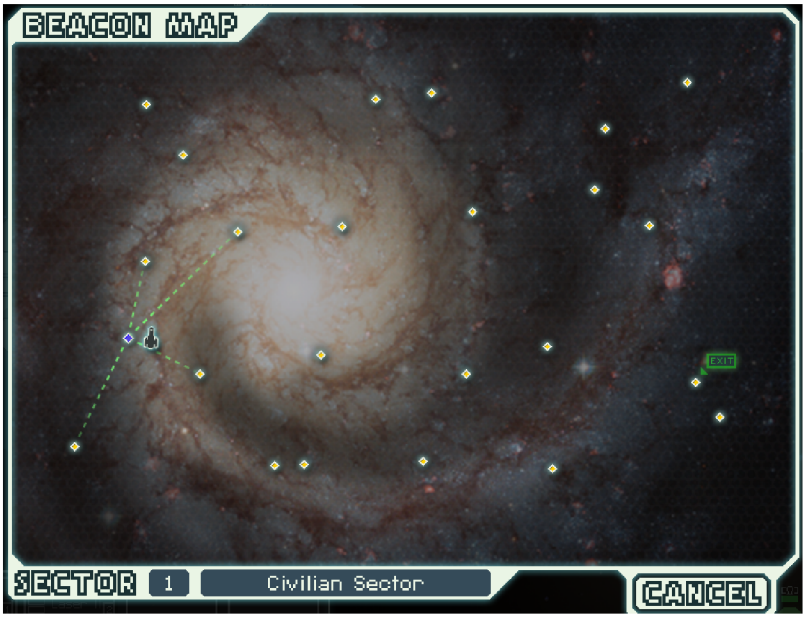
FTL je hra z roku 2012, kde je cílem utéct ve vesmírné lodi několik sektorů a poté porazit nepřátelskou loď, která má několik fází souboje. V této hře je poměrně zajímavé, že hráč se musí strategicky rozhodovat podle více faktorů jako jsou životy, peníze a palivo, takže si musí plánovat cestu přes sektory, interpretované uzly obsahující různé události, jako například obchodníky, nepřátele a náhodné události s možnostmi rozhodnutí.

Na prvním obrázku 2.2 na levé straně nahoře jsou vidět životy lodě a na pravé straně od životů jsou v tomto pořadí kredity, tlačítko pro posunutí se v úrovni, která se zobrazí je vidět na obrázku 2.3. Mapa obsahuje další možné zastávky v úrovni, takové zastávky jsou označeny zelenou pruhovanou čárou a další zastávky aktuálně nedostupné jsou označené oranžovými kosočtverci. Další tlačítko zobrazí statistiky lodě a poslední tlačítko v řadě otevře nastavení a menu hry. Pod životy je vidět aktuální stav štítů, paliva, speciální silnější munice a částí bezpilotních letounů. Procentuální výpis pod statusem štítů označuje šanci na vyhnutí a stav kyslíku na lodi. Poslední část v tomto rohu označuje posádku a pomocná

tlačítka pro její ovládání. Uprostřed je vidět loď a její interiér. Ovál kolem lodi je štít a uvnitř lodi jsou místnosti často s nějakým strojem, se kterým musí posádka pracovat. Označením člena posádky a pravým kliknutím do lodi, by se člen přesunul. Levý spodní roh obsahuje ovládání energie na lodi, každý zelený obdélníček v levé části označuje počet volné, rozmístitelné energie. Po pravé straně volné energie ji lze rozmístit do jednotlivých funkcí lodi, aby fungovaly. Další komponenta jsou zbraně celkem mající čtyři místa, zde jsou jen dvě zabrané. Každá zbraň vyžaduje energii ve zbraních pro fungování, když mají energii mohou se nachystat pro palbu. V pravém spodním rohu jsou vidět aktivní podsystémy.



Obrázek 2.2: Výstřižek scény ze hry Faster than Light. Jedná se o obrázek vzhledu herní plochy.



Obrázek 2.3: Výstřižek scény ze hry Faster than Light. Jedná se obrázek ukazující způsob pohybu po úrovni.

The Binding of Isaac

The Binding of Isaac, hra z roku 2011, přepracovaná v roce 2014 s rozšířeními, kde poslední je z roku 2017, která patří mezi jedny ze známějších a oblíbených her v žánru a pro tento projekt je dost inspirující, jelikož je celkem unikátní oproti předchozím titulům v tom, že čím vícekrát hráč hru dokončí, tím dál se může dostat. Když hráč dohraje hru poprvé, odemkne si první konec a pokud se mu podaří dosáhnout tento konec opakovaně, zjistí, že tentokrát hra nekončí zde, ale v dalších patrech. Mimo postupné odemykání konců, které vypráví příběh Izáka, se odemykají i nové postavy, výzvy a věci pro posílení charakteru. V základu je hra velice primitivní, pohyb a střelba do 8 směrů. Hra však využívá různých nepřátel, jenž jsou postupně těžší a těžší na poražení. Různorodost nepřátel způsobuje neustálou potřebu získávat posílení a lepší koordinaci pohybu, aby se hráč vyhnul smrti. Mapy jsou vždy náhodně generované z místností s pravidly, jako například: vždy jsou dvě tajné místnosti a jedna Boss místnost. V originálním vydání hry byla jen jedna velikost těchto místností, ale v novějším je jich více.

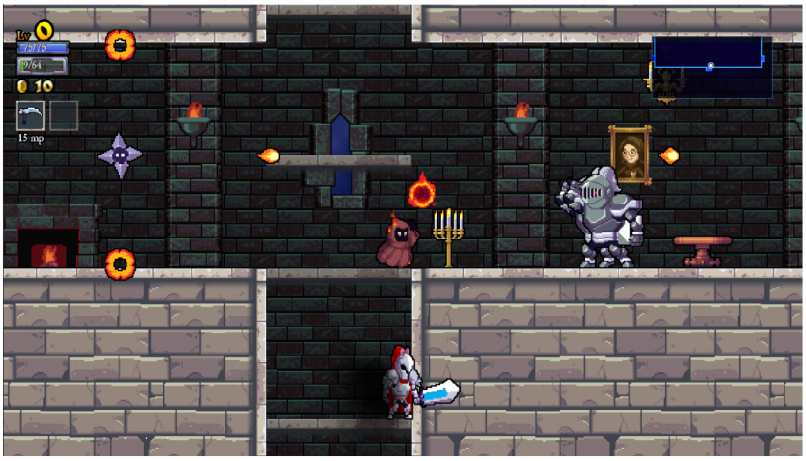
Na obrázku z Izáka 2.4, je možné si všimnout, že hra omezila herní prostor kompletním zabráním horní části, pro informace o postavě a mini mapou. V tomto prostoru je na levé straně zobrazena mini mapa. Tato mini mapa ukazuje místnosti, které hráč navštívil nebo byl vedle nich. Některé místnosti mají na sobě symbol a tím se označují speciálním obsahem. Vedle mini mapy je výpis peněz, klíčů a bomb. Další je zde symbol pro aktuální projektily a místo pro aktivní předmět. Poslední jsou zde životy zobrazené srdíčky, srdíčka se berou minimálně po polovině. Pod tímto pásem s informacemi je hrací plocha, místnost obsahuje překážky jako kameny. Je zde i hráč v levém horním rohu hrací plochy, další dva charaktery jsou jeden z typů nepřátel. Mezi místnostmi se pohybuje dveřmi, které jsou v obrázku zavřené, jelikož jsou nepřátelé naživu. Poslední viditelná část je na pravé straně dole a jedná se o výpis jména úrovně.

Obrázek 2.4: Výstřižek scény ze hry the Binding of Isaac.

Rogue Legacy

Původní vydání Rogue Legacy bylo 2013. Zde si hráč vybírá z několika stylů postav a snaží se projít generovaným hradem. Hrad se skládá z několika částí, které fungují jako úrovně, ale na rozdíl od Izáka se mezi nimi dá volně pohybovat. Jednotlivé sekce jsou vytvořeny z daného množství místností. Cílem je porazit v každé lokaci Bosse, aby se zpřístupnil poslední, který na hráče čeká za dveřmi na začátku každého průběhu hradem. Za zabíjení nepřátel je obvykle hráč odměněn penězi, které ale musí utratit před začátkem dalšího běhu, jinak je ztratí u smrtky čekající před vstupem do hradu.

Na obrázku 2.5. si je možné všimnout, že veškeré hráčovi informace i mini mapa jsou na herní ploše. Na levé straně jsou životy, mana, peníze a úroveň, pod penězi je magie. Na pravé straně nahoře se nachází mini mapa zobrazující tvar místnosti a její vstupy. Jsou zde vidět tři typy nepřátel a některé ozdobné předměty. Charakter nejspodněji v místnosti je hráčova postava.



Obrázek 2.5: Výstřižek scény ze hry Rogue Legacy.

2.2 Plošinové hry

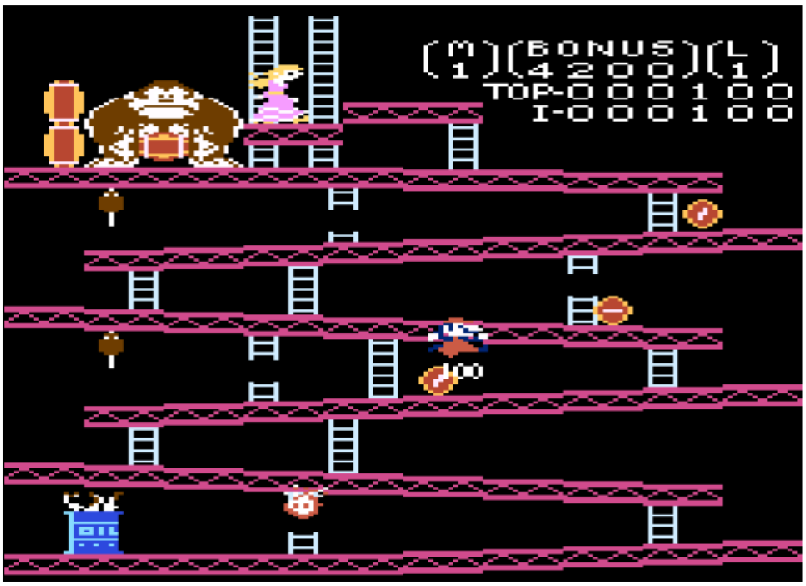
Plošinové hry jsou hry, které jsou založené na využití platforem pro dosažení jinak nedosažitelných cílů. Základem je koordinace pohybu hráče. I když nyní obvykle obsahují jistou formu skákání, ne vždy tomu tak bylo a být nemusí. Tento žánr je využívaný dost dlouho a zažil hodně vývoje a změn. Hry přešly od vyobrazení na jedné obrazovce, jako je tomu ve hře Donkey Kong, přes pohyb po jedné ose až po úplně volný přechod. Během let se též přesunuly z 2D her do 2,5D až do 3D. Jako jedny z prvních příkladů se dají uvést Congo Bongo z roku 1983 pro 2,5D. První 3D plošinová hra je Alpha waves z roku 1990 [13] a první s pravým 3D je Jumping Flash z roku 1995 [14].

Donkey Kong

Donkey Kong je původně 2D hra z roku 1981. Cílem je zachránit slečnu jménem Pauline od Donkey Konga. Hrdina této hry se jmenuje Mario. Ano jedná se stejnou postavičku jako ve hře Mario a vlastně z této video herní série pochází. Hra se dělí do úrovní a jejím cílem až na poslední úroveň je dostat se nahoru. V poslední úrovni musí hráč sesbírat objekty

držící platformy pohromadě a tím shodit a omráčit Donkey Konga, čímž hráč vyhrál hru. Jelikož se původně jednalo o arkádovou hru a ta měla skóre, které se zvyšovalo při ničení či přeskakování překážek, které když se srazí s hráčem tak, ztratí život a musí začít úroveň od začátku. V základu má hráč 3 životy. Hratelnost byla primitivní chůze do stran, skákání a lezení po žebříku.

Na obrázku 2.6 je vidět první úroveň, která je minimálně pro mě ikonická. Lze zde vidět na pravé horní straně aktuální skóre a nejvyšší dosažené skóre, dokonce i životy. Každá úroveň je jiná, v této úrovni je několik plošin, které hráč musí projít, aby se dostal nahoru na konec úrovně. Tyto plošiny jsou propojeny žebříky zkracující potřebnou vzdálenost. Aby to nebylo jednoduché, Kong posílá barely, které jedou po plošinách, ale mohou spadnout i po žebříku. Barely je třeba buď přeskočit nebo zničit palicí.



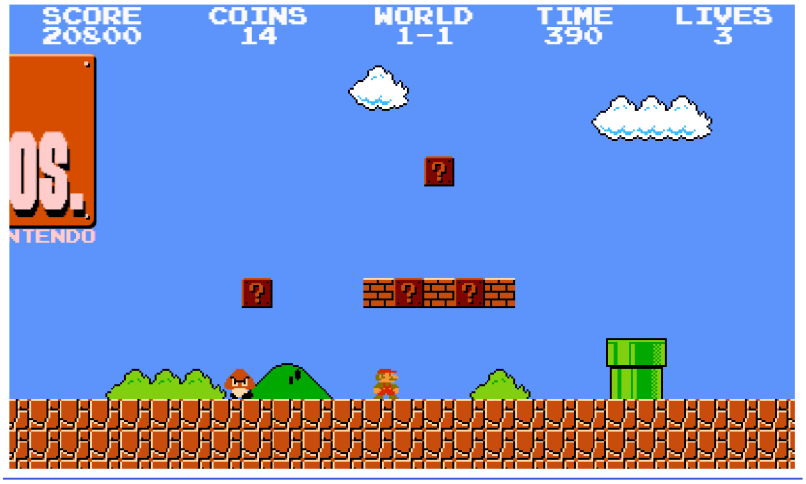
Obrázek 2.6: Výstřižek scény ze hry Donkey Kong z roku 1981 vytvořeného na Atari. Nejedná se o snímek z originální konzole, ale emulátoru na webu https://www.retrogames.cz.

Mario

Mario je původně 2D hra z roku 1985. Cílem je zachránit princeznu za instalatéra Maria, princezna by měla být vždy na konci úrovně. Úrovně jsou v základu rovné a terén se mění pomocí plošin a trubek. V cestě Mariovi překáží nepřátelé, kteří jsou složitější čím dál ve hře se hráč nachází. Hratelnost je v základu poměrně jednoduchá, ale stává se složitější, když se člověk snaží dosáhnout vyšších rychlostí a plynulosti pohybu. Hráč může běhat ze strany na stranu a při běhu zrychlí, dále má možnost skákat, čímž může zničit některé bloky a odemknout posílení nebo porazit nepřátele. Hra končí prohrou, když hráč ztratí všechny své životy a ty ztrácí naražením do nepřátel z boku a zespodu bez posílení nebo pádem mimo mapu.

Na obrázku 2.7 lze vidět začátek první úrovně, což alespoň pro mě je jeden z nejznámějších pohledů ze hry. Na vrcholu obrazovky lze vidět informace jako výpis skóre úplně na levé straně, další je zde počet sesbíraných mincí. Po dosažení sta mincí obdrží hráč jeden bonusový život, mince jsou obvykle schované v blocích nebo skrytých částech úrovní. Uprostřed

je vypsáno, jaký svět (první číslo) a jaká úroveň (druhé číslo) je aktuální, svět označuje především vzhled úrovně. Čtvrtá položka je čas, který je vypsaný v sekundách a jedná se o čas během kterého hráč musí dokončit úroveň. Nevyužitý čas se po skončení úrovně přepočítá na skóre. Poslední položka vypisující informace obsahuje počet životů, pokud číslo klesne na nulu, hra končí. Mimo pás s informacemi je vidět, že se úrovně skládají ze základní plošiny, která v sobě může mít mezery jenž umožňují pád z mapy. Cihlové bloky, které lze rozbít jen když má hráč aktivní posílení, dále z bloků, kde lze vidět otazníky. Z těchto bloků vypadávají peníze nebo posílení. Zelené trubky mohou obsahovat nepřítele nebo cestu do skryté části úrovně. Dále je zde vidět nepřítel vypadající jako houba. Tento nepřítel vždy jde jedním směrem, dokud nenarazí do překážky. Poslední je zde vidět samotný Mario bez posílení.



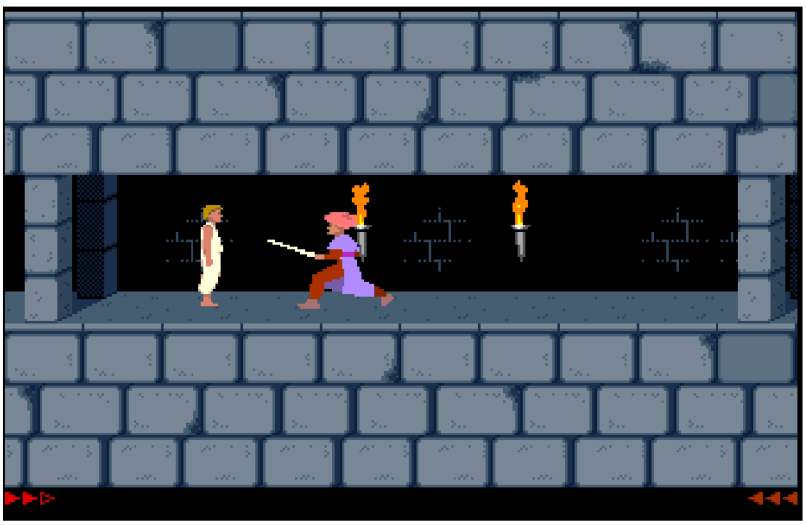
Obrázek 2.7: Výstřižek scény ze hry Super Mario Bros. Nejedná se o snímek z originální konzole, ale z webu https:supermarioplay.com.

Prince of Persia

Prince of Persia je původně z roku 1989. Hra se dělila na 12 úrovní. V této hře se na rozdíl od předchozích nespoléhá na skákání jako základ hratelnosti. Místo toho je hra spíše založená na rozuzlení úrovně a občasného poražení nepřátel. Také na rozdíl od předem zmíněných titulů má hra nějaký soubojový systém. Asi největší rozdíl je neomezenost životů před vynuceným resetování hry, místo životů využívala časového limitu jedné hodiny. Aby se hráč necítil úplně bez možností, je možné přeskočit jednu úroveň, ale také to sníží čas na patnáct minut nebo po třetí úrovni si postup ukládat.

Obrázek 2.8 je obrázek ze začátku hry, kde je vidět, jak může jedna část úrovně vypadat. Uživatel ovládá blonďatého panáčka, ten má tři životy. Když hráči dojdou životy, musí začít úroveň od začátku. Zmiňované životy jsou reprezentovány červenými šipkami v levém spodním rohu. Dále je zde vidět časový limit šedesáti minut. Limit není na úroveň, ale na hru. Obrázek 2.9 mimo hráčova charakteru i jednoho z nepřátel, tohoto nepřítele lze porazit v souboji, pokud hráč najde meč. Životy nepřítele jsou naproti hráčovým v pravém dolním rohu.

Obrázek 2.8: Výstřižek scény ze hry Prince of Persia. Nejedná se o snímek z originální platformy, ale emulátoru na webu https://www.retrogames.cz.



Obrázek 2.9: Výstřižek scény ze hry Prince of Persia. Nejedná se o snímek z originální platformy, ale emulátoru na webu https://www.retrogames.cz. Jedná se o část úrovně, kde hráč může bojovat.

Sonic the Hedgehog

Sonic je opět původně 2D hra z roku 1991. V této hře hrajete za Sonica, super rychlého ježka. Hra je rozdělena na světy, které jsou rozděleny do úrovní. Světy jsou zde zamořené výtvory Dr. Robotnika, který uvěznil zvířata. Pro dokončení hry není třeba je osvobodit, ale je to možné. Je myšleno, aby hra byla hrána v rychlejším tempu, dost často větší rychlost umožní lepší cesty. Cílem je dostat se na konec mapy v co nejkratším čase nebo v případě konce světa porazit Dr. Robotnika v jeho nějakém šíleném zařízení. Hratelnost je velmi jednoduchá, hráč se pohybuje doleva či doprava s tím, že jako v Mariovi se hráč zrychluje čím déle ten směr drží a skákání umožnuje porazit Robotnikovi stroje.

Na obrázku 2.10 lze vidět jeden z oblouků. V první úrovni, kterou aktuálně probíhá charakter hráče, Sonic. Na levé horní straně se nachází skóre a čas. V této hře se počítá čas, jak dlouho trvá hráči proběhnout úroveň, hráč se snaží projít úroveň co nejrychleji. Třetí položkou jsou prsteny, které nahrazují mince z jiných her. Dokud má hráč alespoň jeden prsten nezemře, ale vyhodí je. V levé dolní části obrazovky jsou vidět životy.

Obrázek 2.10: Výstřižek scény ze hry Sonic the Hedgehog. Nejedná se o snímek z originální konzole, ale ze Sonic Mania na platformě Steam.

Kapitola 3

Princip hry a návrh

Tvorba návrhu Dle Jesse Schella dobrý herní designér potřebuje určité kvality. Mezi tyto kvality patří animování, znalost lidí, vymýšlení stovek nápadů, kreativitu, hry, matematiku, historii a mnoho dalších. Ale i když dobrý návrhář potřebuje tolik kvalit, tak za nejdůležitější považuje naslouchání. Návrhář by měl naslouchat svému týmu, fanouškům, klientovi, sobě, ale i hře samotné [15]. V tomto případě jde říci, že základ návrhu pro tuto hru byl z mé znalosti her. Prvně se tento návrh a hra upravovaly podle toho jak jsem "naslouchal hře". Pozdější změny vznikly z naslouchání testerům.

3.1 Základy hry

Produktem této práce je vytvoření 2D roguelite hry. V této hře se nepočítá s tím, že člověk uspěje na poprvé, ale že bude muset mnohokrát začít od znovu s možným vylepšením statistik charakteru. Toto vylepšování bude probíhat mezi instancemi hry či běhy v samostatné scéně, zde pojmenované Lobby (více o lobby v 4.1.2). Takto bude mít hráč pocit nějakého pokroku i když neuspěje. Jednotlivé úrovně jsou modulově generovány pro zajištění co největší variace pro hru. Při studiu předchozích úspěšných titulů ve stejných žánrech jsem si uvědomil, že mechanicky jsou tyto hry obvykle poměrně jednoduché, ale těžší na dohrání.

I když se nejedná o 3D hru, je zapotřebí určit pohled, ze kterého bude hráč ovládat charakter a sledovat svět. Existují tři možné volby pro 2D, pohled z vrchu jako ve hře the Binding of Isaac nebo isometrické 2D aneb pohled snažící se vypadat jako 3D pohyb. Ale pro hru je určena třetí možnost, pohled z boku jako například ve hrách Rogue Legacy nebo Mario.

3.2 Hráč

Hráčův charakter bude jediná část ve hře, kterou ovládá. Tento charakter potřebuje základní charakteristiky a vybavení. Mezi základní charakteristiky patří životy, které lze buď vyobrazit přes srdíčka. Ta jsou obvykle využívaná, když životy jsou počet přežitelných ran, nebo se dají zobrazit posuvníkem či číslem, když bývají různější síly zásahů. Pro tento projekt je vhodnější vyobrazení číslem, posuvníkem nebo obojím. Stejně funguje i mana, obvykle zdroj charakteru umožňující využívat magii a jiné nadpřirozené schopnosti. Dále charakter potřebuje nějakou rychlost a místo pro vybavení, například zbroj, zbraň a pomůcky, jako například forma léčení, doplnění many, poškozování s omezeným počtem použití nebo využitím many. V tomto projektu bylo určeno, že charakter může mít tři části zbroje určující

obrané statistiky. Jednu zbraň určující způsob, jakým hráč poškozuje bez využívání prostředků, které dojdou, a nakonec může mít 3 pomůcky, které jsou vždy nějakým způsobem omezené, ale velmi silné

Samotný charakter musí mít základní funkce jako pohyb a útočení pro vypořádání se s nepřáteli. Jeho pohyb bude volný jen po vodorovné ose. Na vertikální ose bude omezen skákáním a to tak, že mapa bude obsahovat gravitaci, tak se nebude vznášet nekonečně. Jelikož v této hře nebudou žádná lana či žebříky, bude skákání jedním ze dvou způsobů, jak se dostat výš a gravitace, jak spadnout níž. Třetí možností pohybu bude úskok, v této hře to je naplánováno jako rychlý přesun možný v osmi směrech, během kterého hráč nemůže být zraněn.

Druhá a poslední hlavní funkcionalita charakteru je útočení. Základní formu útoku určují, zda se jedná o útok na dálku nebo blízko. Na blízko bude jen do základních čtyř směrů, nahoru, dolů, doleva a doprava, oproti tomu útok na dálku bude jako úskok v osmi směrech. Dále se útok rozdělí podle typu zbraně, nebo pomůcky.

3.3 Nepřátelé a překážky

Nepřátelé by měli alespoň v tomto projektu fungovat hodně podobně jako hráč, možná lehce primitivněji. Pohyb bude rozdělený do létajících (bez gravitace) a nelétajících (s funkční gravitací), ty nelétající bude potřeba omezit, aby nepadali z plošin. Pro nelétající bude útočení omezeno pro osu x.

Boss bude vždy po vyčištění dostatečné části úrovně dostupný ve své upravené aréně. A navíc bude mít více typů unikátních útoků.

3.4 Herní engine

Pojem herní engine pochází zhruba z poloviny 90. let 20. století. V této době začaly populární hry být vytvářeny s dobrým rozdělením hlavních funkcí. Například vykreslování je vytvořeno nezávisle na pravidlech hry. Hodnota těchto rozdělení byla brzy znatelná, když se začaly hry upravovat vytvářením nové grafiky, map nebo pravidel. Zpočátku bylo možné jen modifikovat původní hru a tím vytvářet obdobné hry. Postupem času to přešlo přes umožnění tvorby v jistém žánru, až po program umožňující tvorbu her [4].

Dnes by se dal engine popsat, jako pomocná sada nástrojů nebo rozhraní určené pro usnadnění a optimalizaci vývoje videoher. Engine obvykle pomáhá zpracovat uživatelské vstupy, vykreslovat ve 2D nebo i ve 3D, stará se o herní cyklus, obvykle už má zabudovanou fyziku, jako například osvětlení, stíny, gravitaci, kolize, dále umožňuje lehčí zpracování zvuku a připojení k síti. Může obsahovat i editor pro vytváření světa a propojovaní všech komponentů hry.

Veškerý vývoj je závislý na volbě herního enginu a ne každá volba je vhodná nebo dělaná pro všechny žánry. Některé žánry potřebují podporovat více objektů s horším vykreslováním, jiné zase potřebují realistickou grafiku [4],tudíž je zapotřebí zvolit vhodný a pro vývojáře vyhovující engine. Následující zmínky jsou volně dostupné, ale různě vhodné pro projekt tohoto typu. Ať již jednoduchostí používání, velikostí komunity či podporou 2D.

3.4.1 Unity

Unity je engine, který je zdarma, pokud osoba či firma nepřesáhne příjem sto tisíc dolarů v posledních dvanácti měsících. V případě, kdy přesáhne, musí se nakoupit licencované verze. Aktuální stabilní verze je Unity 2022.2.9 [16].

Unity je velice všestranný, mezi platformní engine s podporou tvorby 2D i 3D her a jeho skripty pracují s jazykem c#, je vhodný pro tvorbu na počítač (Windows, Linux, Mac), mobil (iOS, Andorid, Windows), web (WebGL), konzole (Xbox One, Playstation 4, Playstation 5), ve 2D i ve 3D klidně i ve virtuální realitě ani na žánru nijak víc v tomto směru nezáleží. Jako správný a dobrý engine obsahuje vše potřebné pro tvorbu her. Osvětlení, stínování, kolize, spouštěče a jejich zpracování probíhá pomocí události (event), dokonce i podporu a editor animací, například pro 2D buď prohazováním spritů nebo přes vytvoření kostry. Se všemi těmi možnostmi je poměrně snadné se s ním naučit a vytvořit poměrně kvalitní hru, už jen díky jeho editoru scén, kde stačí věci jen přetahovat.

Mezi hry vyvinuté na nějaké verzi Unity patří: Kerbal Space Program – oblíbená simulace vesmírného programu, Plague Inc. – Strategie kde se hráč snaží vyhladit lidstvo pomocí jím ovládané nemoci nebo takovou nemoc vyléčit, Oblíbená mobilní hra Subway Surfers – hra kde hráč neustále běží dopředu a zrychluje, cílem je nenarazit, dále Pokemon Go, Ori and the Blind Forest, Cuphead nebo i česká hra Beat Saber [11].

3.4.2 Unreal Engine

Unreal Engine je opět zdarma, dokud osoba či společnost nezíská z prodeje jeden milion dolarů, poté si společnost Epic Games účtuje 5% zisků. Celkem nově 5.dubna 2022 vyšel Unreal Engine 5.

Unreal je engine s podporou tvorby 2D i 3D her, ale jeho skripty pracují s jazykem C++, dále podporuje vizuální skriptování přes Blueprints. Tento engine umožňuje fotorealistickou grafiku, v čemž exceluje a nyní je jedním z nejlepších. V novém Unreal Enignu 5 je animování animací v kontextu zrychlené a zjednodušené, Lumen sloužící pro lepší dynamické osvětlení a odrazy, procedurální zvukový design, a dokonce se chlubí množstvím možných detailů bez ztráty výkonu [2].

Příklady známějších her vyvinutých v nějakém Unreal Enginu by byly: Fortnite - hra od společnosti Epic Games vyvíjející engine, Harry Potter a Kámen mudrců, Unreal tournament, celá série Bioshock, Borderlands, Duke Nukem Forever, Fable a Mass Effect [12].

3.4.3 Cryengine

Cryengine řeší zpeněžení podobně jako Unreal. Před získáním 5000 dolarů z projektu zdarma, poté si Crytek účtuje 5% zisků. Aktuální verze je Cryengine 5.6.

Cryengine je ideální pro tvorbu 3D titulů a dokonce je lépe hodnocený než Unreal Engine 4. Toto hodnocení si rozhodně zaslouží, má jednu z nejlépe vypadajících grafických možností a zároveň není pomalý. Obsahuje dobrou simulaci fyziky, podporuje složité efekty, dynamickou destrukci, reálně vypadající osvětlení, a stejně jako předchozí i přehledný editor [1]. Tyto vlastnosti z něj dělají ideální engine pro tvorbu střílečeк a her na hrdiny, ať již z první či třetí osoby.

Bohužel tento projekt je ve 2D na což není Cryengine zaměřený. Tím pádem, i když je to možné a zmínku si zaslouží, není pro tuto práci úplně ideální.

Cry engine je známý tituly: Ryse: Son of Rome, Far Cry, Crysis a také známým českým titulem od Warhorse studia Kingdom Come Deliverance [10].

3.4.4 Godot

Godot je 2D i 3D kompletně zdarma, open-source engine, který podporuje c#, c++ nebo speciálně vytvořený GDScript už jen díky výběru jazyka se jedná o poměrně dobrý engine pro začátečníky, čemuž nadále napomáhá dobrá komunita a například i menší velikost instalace. Hodně funkcí je stejných či podobných enginu jako unity. Mezi takové funkce se dá považovat stromová struktura uzlů usnadňující design hry, flexibilní systém scén a možnost vytvářet vlastní nástroje [3] a jako u Unity i možnost lehce kliknutím určit koncovou platformu.

Významné hry pro Godot jsou: Sonic Colors: Ultimate, Commander Keen in Keen Dreams, Deponia [9].

3.4.5 Monogame

Monogame je podobně jako Godot zdarma a open-source, ale narozdíl od Godotu se jedná o framework, sadu nástrojů do c# [7]. Monogame je spíše pro tvorbu 2D her. Teoreticky je sice možné vytvořit pomocí monogame 3D hru, ale musel by uživatel udělat úplně všechny funkce pro 3D zobrazení. Jelikož se jedná pouze o framework a minimálně oficiálně nemá žádný editor, veškerá práce probíhá v kódu a to od vytváření mřížky mapy po nejdetailnější funkčnosti.

Pomocí Monogame vzniklo několik známých 2D titulů jako jsou Celeste, Jump King, Stardew Valley [8].

Kapitola 4

Implementace

Celý Projekt je vyvinutý v herním enginu Unity na verzi 2020.3.19f1, všechny skripty jsou psány v jazyce c# vzhledem k tomu, že k tomu je Unity stavěné. Pro jednoduchost jsou zde využity jen čtyři scény, a to hlavní menu, lobby, herní scéna v projektu pojmenována level a editor.

Pro práci v Unity se budou vytvářet před připravené objekty (anglicky prefabs), které dále budou v této práci označovány jako prefab. Tyto objekty slouží jako vzor pro lehké vytváření kopií. Tato praxe zrychluje a zpříjemňuje tvorbu složitějších objektů. Může sloužit jako základ pro nové objekty, které z něj budou dědit jeho obsah a funkce.

V tomto projektu jsou některé funkce od Unity využívané na každém nebo téměř každém objektu tyto funkce jsou:

- Start - Funkce volaná při vytvoření objektu například, vytvoření instance nebo po načtení scény.
- Update - Funkce volaná na každém snímku hry, kdy je přiřazený objekt aktivní. Je výhodná pro snímání uživatelských vstupů, ale nevhodná pro složité výpočty a náročné instrukce.
- FixedUpdate - Funkce volaná na každém daném počtu snímků
- OnCollision(Enter/Stay/Exit)2D - Ovládá události objektu pokud má kolizní komponentu bez zapnuté spouště. Spuštění je podle zvoleného buď vstup, stálý dotyk nebo konec kolize.
- OnTrigger(Enter/Stay/Exit)2D - Ovládá události objektu pokud má kolizní komponentu se zapnutou spouští. Spuštění je podle zvoleného buď vstup, stálý dotyk nebo konec kolize. Event je definovaný v unity a funkce obsahuje kolizní komponentu druhého objektu

Všechny zvuky a hudební kousky jsou stahovány z "Unity Asset store".

4.1 Rozložení Scén

Scéna se vytvoří přes přidání nové položky scene v okně zobrazující projekt. Klasické úložiště pro tento objekt je složka Scenes v Assets. Po vytvoření scéna obsahuje jen objekt hlavní kamery. Takto nově vytvořenou scénu se poté upravuje, aby po načtení obsahovala potřebné

skripty bez nutného přidávání přes kód při každém načtení, například bude obsahovat uživatelské rozhraní, skript pro generování úrovní a hráče.

Načítaní scén se řeší přes UnityEngine.SceneManagement a příkaz SceneManager. LoadScene(int index);. Index se nastavuje v nastavení sestavení. Pro tento projekt jsou:

- 0 - Hlavní menu
- 1 - Lobby
- 2 - Level
- 3 - Editor

Objekty se dají přidávat buď přetažením prefabu nebo kliknutím tlačítka přidat v záložce hierarchie objektů. Každý objekt ve scéně lze pojmenovat a je výhodné pojmenovávat objekty podle jejich funkčnosti. Zmiňuji to tu zde proto, že některé objekty se nemusí jmenovat stejně pro funkcionalitu a jedná se jen o mé pojmenování například GameManager.

Každá scéna je vytvořena s alespoň jednou unikátní funkcí, aby se neprolínaly. Takto je to udělané pro usnadnění práce. Hlavní menu je úvodní obrazovka, zařizuje veškerá nastavení a výběr jednoho ze tří možných uložených souborů, mimo jiné i když existuje scéna editoru, tak se zde nachází editory. Scéna editoru je speciálně pro vytváření místností, jelikož se jedná o složitější vytváření jak u ostatních položek. Lobby slouží jako přestupní bod mezi menu a hrou, zde hráč využívá roguelite mechanik. Level je scéna kde se odehrává hra jako taková.

4.1.1 Scéna: Hlavní menu

Jak je již zmíněno, hlavní menu je úvodní scéna, kterou hráč vidí jako první věc po zapnutí hry a která funguje jako hlavní rozcestník pro hru. Scéna obsahuje hlavní kameru nastavenou na zobrazení objektu Canvas, EventSystem vložený od Unity, herní objekt GameManager, který má na sobě skript BaseView obsahující ovládání pro menu zobrazování okna uložených souborů a obsahuje také funkce použitelné ve více objektech, například instanciace objektů a jejich následné přiřazení rodičovského objektu a jména. Poslední je v této scéně MainCanvas typu Canvas, tento objekt obsahuje objekt typu Image(obrázek) fungující jako pozadí. Tento objekt je nastavený na to, aby se roztáhl podle velikosti rodiče aneb MainCanvas, dále jsou zde panely, první je nastavený pro tento projekt s kotvou na pravý roh kde si přeji mít menu jako takové, obsahuje samotné "menu", tlačítka umožňující interakci. Od spodku obrazovky jsou tlačítka:

- Quit/Odejít - Tlačítko s přiřazenou funkcí obsahující řádek kódu Application.Quit(); což je funkce v Unity enginu vypínající hru.
- Manual/Manuál - Tlačítko otevírající okno s krátkým návodem ke hře.
- Options/Nastavení - Tlačítko otevírající okno, kde je možné nastavit základní ovládání postavy.
- Play/Hrát - Tlačítko, které nechá vytvořit prefab s oknem, kde si hráč může vybrat, zda začít hrát na jednom ze tří uložených souborů nebo zda je vy restartovat aby začínal od znova.

Další panely jsou v projektu pojmenované Editor, sloužící pro výběr typu položky k vytvoření či úpravě. "Custom"pro vytvoření nové položky nebo výběru z již vytvořených. A jako poslední "NewItem", panel obsahující vyplnitelná pole dle typu položky, případně předvyplněna při načítání.

Obrázek 4.1: Výstřižek vzhledu hlavního menu

4.1.2 Scéna: Lobby

Lobby je scéna, kde přichází na řadu implementace roguelite, funguje jako mezistupeň mezi jednotlivými hrami. Lobby může být jen okno ve stylu normálního menu nebo jako zde využití lokace, kde se předá hráči kontrola nad avatarem a přidá mu nějakou interakci, kde si může hráč vyzkoušet pohyb a útok.

Jako v každé scéně je zde kamera, aby bylo cokoliv vidět, ale tato kamera má na sobě skript, který vyhledá objekt hráče a na každém snímku přes funkci Update si aktualizuje pozici, aby byl hráč vždy v jejím středu. Vycentrování probíhá přes komponentu transform obsahující informace o pozici ve scéně, tuto komponentu má každý herní objekt v Unity. Kamera, která následuje hráče je nejpohodlnější v těchto stylech hry, bez následování hráče by byla hra mnohem komplikovanější minimálně přes přidání koordinace kamery do vyhýbání se a útočení.

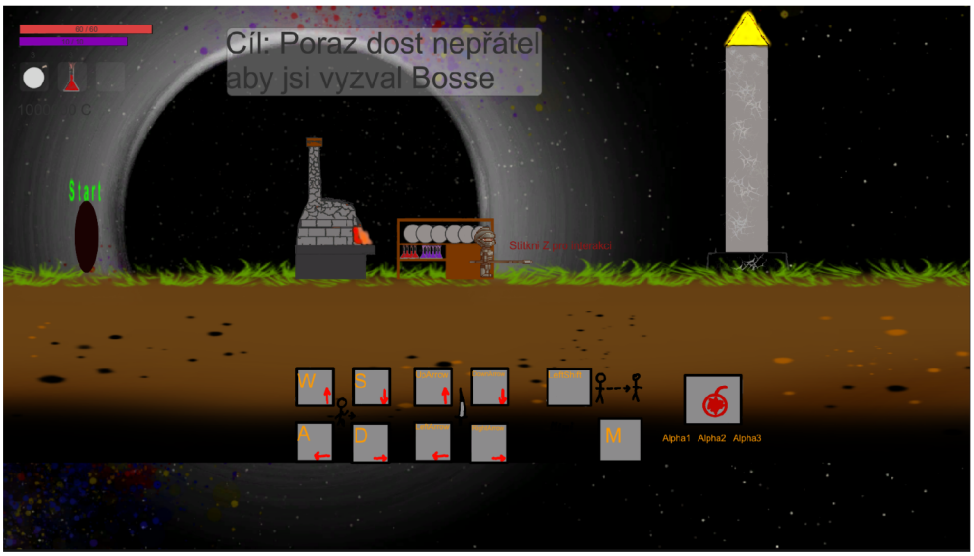
Po kameře je zde Canvas tentokrát obsahující všechny prvky uživatelského rozhraní pro hru i pro lobby 4.4. Tyto prvky, jako okno pro nákupy, vylepšení nebo obrázky s ovládání mají všechny referenci do scény ve skriptu UIManager pro ulehčený přístup hlavně k neaktivním prvkům například obchod s vylepšeními. Dále tento skript obsahuje kontrolu peněz, zda je hráč schopný si koupit pomůcky, podle těchto funkcí se zamknou tlačítka pro nákup.

I když okno pro vylepšování je kontrolované přes UIManager tak samotná tlačítka pro nákup vylepšování mají vlastní skript, jelikož v základu jsou aktivní jen první vylepšení, tak se při aktivaci nebo vytvoření přesunem do scény potřebují podívat jsou-li již zakoupené, kolik úrovní mají, pokud jsou zakoupené, tak aktivují následující tlačítka, tvořící tak strom vylepšení. Asi nejlehčí způsob vytvoření takového stromu je na skriptu mít list s odkazy na následující vylepšení. Vytvořit strom vylepšování, kde se postupně odemykají, je jedna z

možností. Další jsou například vytvoření pár tlačítek obsahující různá vylepšení nebo jedno co bude postupně vylepšovat po dané cestě. První možnost zpřístupnění všeho hned dává sice největší volnost, ale vede hráče k tomu, aby vylepšoval pouze ty věci, které chce nejvíce a jsou nejsilnější. Druhá možnost nedává žádnou svobodu, což taky není správně. Možnost zde využitá poskytuje poměrně vysokou svobodu vylepšování, ale zase ne úplnou a využití takového stromu alespoň u mě vytváří pocit nějakého postupu.

Další je objekt se jménem Lobby. Zde se nachází všechny objekty unikátní pro tuto scénu grafické zobrazení terénu a jeho kolize, zóna pro zapnutí obchodu a portál, který začíná hru. Portál je objekt obsahující jeho vzhled, kolizní komponentu se zapnutým spouštěčem ve tvaru co nejlépe sedí tvaru portálu, zde je to ovál a skript. Tento skript ovládá kolize přes OnTriggerEnter2D. Zde se kontroluje, zda kolize obsahuje jméno Player nebo jak aktuálně je pojmenovaný objekt hráče, mezi další možnosti kontroly jsou porovnání označení/tag nebo vrstvy. Při vstupu hráče se vypne objekt hráče (běžící funkce Update a FixedUpdate mohou hodit chybu kvůli dočasné ztrátě referencí) nastaví se objekt hráče a Canvas na "DontDestroyOnLoad", což zaručí, že se při změně scény objekty nezničí, tím pádem se nemusí znova vytvářet nebo vkládat data. Poté se dá přes SceneManager načíst herní scéna. Obchod obsahuje opět vzhled objektu, kolizní komponentu ideálně kruhovou nebo čtvercovou se zapnutým spouštěčem a skriptem pro snímaní a zpracování vstupu nebo zůstání v kolizi a výstupu do kolizní zóny plus snímání stisknutí vybraného tlačítka v tomto případě 'Z', když hráč vstoupí aktivuje se text oznamující, jak otevřít obchod, pokud neodejde a zmáčkne 'Z' aktivuje se obchodní okno.

V neposlední řadě je zde vložen objekt hráče blíže popsaný v 4.7, LevelManager je prázdný herní objekt kam se v lobby generují hráčovi projektily a EventSystem.



Obrázek 4.2: Výstřižek vzhledu Lobby při pohledu ze hry

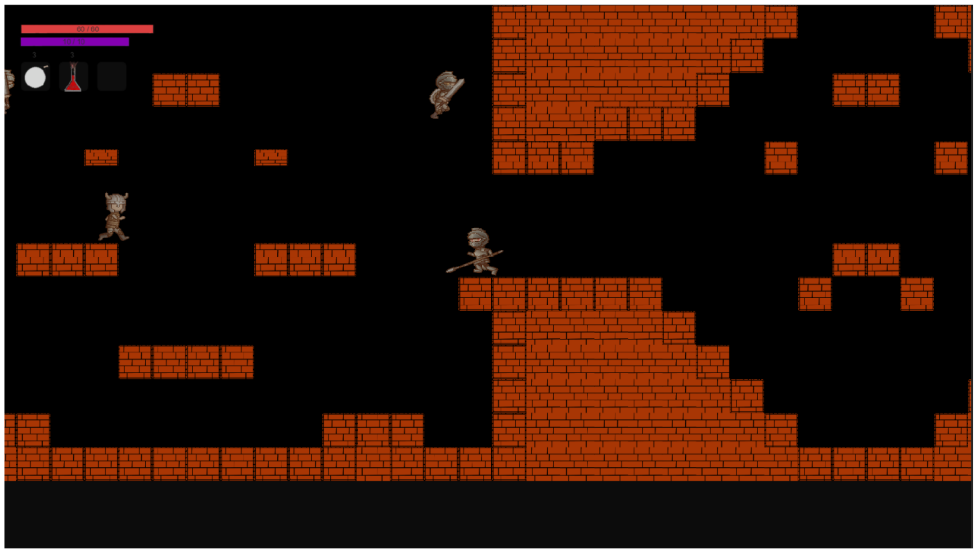
4.1.3 Scéna: Herní scéna

Herní scéna obsahuje samotnou hru, zde se generuje úroveň a nepřátelé, a i když by se dalo říct, že se jedná o nejdůležitější scénu, zároveň i v základu obsahuje nejméně herních objektů.

Jelikož v tomto projektu je interaktivní lobby, tak zde kamera funguje stejně jako v Lobby.

V této scéně je opět LevelManager, který zde obsahuje skript, generující úroveň a zároveň se jedná o rodičovským herním objektem pro všechny generované objekty mapy. Po vygenerování obsahuje prefab místnosti, objekt o dané velikosti s obrázkem na pozadí. Tyto místnosti jsou rodičovským objektem pro bloky a nepřátele skládající místnost. Více v sekci o generování 4.9.

Poslední herní objekt, co je zde v základu, obsahuje skript, který přečte z disku kolik existuje místností, zbraní, pomůcek, pastí, bossů a nepřátel. Skript při vytvoření objektu zkontroluje danou složku pro JSON objekty.



Obrázek 4.3: Ukázka z herní scény

4.1.4 Scéna: Editor

Scéna editoru zde slouží pro vytváření a úpravy místností. Kamera v této scéně je ovládaná uživatelem, jelikož jsou místnosti různých velikostí, tak je možné se přesouvat a přibližovat či oddalovat kameru.

Ve scéně se nachází editorské uživatelské rozhraní zobrazující informace o vybraném bloku, možnost je otočit, pole pro název místnosti, tlačítko pro odstartování a ukončení testu místnosti a ještě jedno tlačítko slouží pro odchod zpět do menu. Alespoň na jeden objekt je vhodné přiřadit skript s načtenými vzhledy, opakované načítání by mohlo být celkem pomalé. Nejdůležitější součást se do této scény přesouvá již z hlavního menu, jedná se o mřížku, na kterou se vytváří samotná místnost. Mřížka je zde čtená a ukládaná jako pole a každé číslo je poté rozděleno do os x a y v závislosti na jeho velikosti. Každá buňka této mřížky představuje místo pro jeden blok. Buňka je pojmenovaná podle čísla, které okupuje a obsahuje skript, jenž na přejetí myši přes buňku se aktivuje a očekává pravé nebo levé tlačítko myši pro položení bloku a smazání bloku.

4.2 Ukládání dat a lokalizace

Data potřebují být nějak uložena, v tomto projektu se to řeší přes ukládání do souborů s formátem JSON a do složky StreamingAssets. Ukládání do JSON souborů má své výhody i nevýhody. Mezi výhody bych započítal, že si každý může upravit svůj zážitek ze hry poměrně jednoduše. Další dobrá možnost je využít mix JSON souborů a ScriptableObjects, tyto objekty jsou výhodné pro zrychlení tvorby prefabů se stejnými daty.

Pro ukládání do JSON jsou zde vytvořeny skript FileManager a serializovaná třída. Data jsou v listu a jsou složeny z řetězcového klíče a řetězcové hodnoty. FileManager obsahuje funkce pro čtení z disku a zápis na disk. Převod v těchto funkcích probíhá přes příkazy JsonUtility.ToJson(objekt) pro zápis na disk a JsonUtility.FromJson(řetězec) pro čtení dat.

Načítání dat do modelů probíhá jednoduchým přepínačem pro každou GameData položku v listu se určuje, co má přepínač udělat podle řetězcového klíče z aktuálního GameData objektu.

4.2.1 Úložný objekt

Úložný objekt je objekt vytvořený po výběru uloženého souboru v hlavním menu. Při tomto výběru se tento objekt nastaví na "DontDestroyOnLoad"a takto zůstane, dokud se hráč nerozhodne vrátit do hlavního menu nebo vypnout hru. Tento objekt si udržuje všechny důležité informace v sobě uložené, mezi tyto informace patří statistické údaje (výhry, prohry) a počet a typ vylepšení.

Informace jsou uložené ve skriptu pojmenovaném SaveFileManager. Mimo informace jsou zde funkce:

- Start() - Po vytvoření objektu se vytvoří list aktuálních dat a nastaví se objekt na "DontDestroyOnLoad".
- CreateDefaultSaveFile() - Funkce s napevno napsaným listem herních dat pro zapsání nového či přepsání existujícího souboru. Tato funkce se používá při resetování nebo při vytvoření nového souboru v hlavním menu.
- SaveFile() - Do Listu dat se zde načítají data a ty se potom zapíší na disk.
- LoadSaveFile() - Načte soubor do listu dat a ten se načte do lokálních proměnných.

Funkce SaveFile() se volá automaticky v případě smrti hráče v úrovni, což znamená, že se nepodporuje ukládání během hry. Ukládání je možné pouze po dokončení hry nebo při nákupu nových vylepšení.

4.2.2 Lokalizace

Lokalizace slouží k usnadnění pochopení hráčem a pro dosažení větších skupin lidí. Její provedení je poměrně snadné. Jako všechny data i lokalizace je uložena ve formátu JSON a po přečtení převedena do slovníku. Na každé scéně se nachází objekt se skriptem, který najde všechny objekty s typem text. Poté je jméno objektu vyhledáno ve slovníku a k tomuto jménu je přiřazena hodnota připojená ke klíči. Některé objekty je potřeba aktualizovat na interakci s nimi, a ty jsou pak přepsány zvlášť v rámci interakcí, samotné přepsání je opět vyhledáním ve slovníku podle jména.

4.3 Animace

Animování je dlouhý a složitý proces, bohužel nejsem zrovna umělec, takže v tomto projektu je využita sada z Asset Store, oficiální web pro prodej, nákup a stahování položek projektů, například jako je hudba, různé animace nebo i funkce. I když je postavička zakoupená, neznamená to, že se člověk vyhne animování. Položka jménem Customizable Pixel Character od uživatele Cainos obsahuje animovaný lidsky vypadající pixelový charakter. Ale neobsahuje několik potřebných animací.

Pro přidání animace se musí přidat na objekt nebo prefab komponenta Animator a vytvořit nový Animator Controller v okně pro zobrazení projektu. Unity má i okno pro animator, tohle okno zobrazuje možné stavy charakteru, jejich vztahy a podmínky pro přechod mezi stavy. Poté se musí vytvořit nová animace, a to je možné udělat po výběru objektu s animátorem v oknu animace. Okno animace je pro nastavení rozložení animace a přidání objektů ovlivňujících animaci.

Sprite - označení používané v počítačové grafice jedná se 2D obrázek integrovaný do scény. Je možné ho využít jak ve 2D tak ve 3D.

4.3.1 Prohazováním obrazů

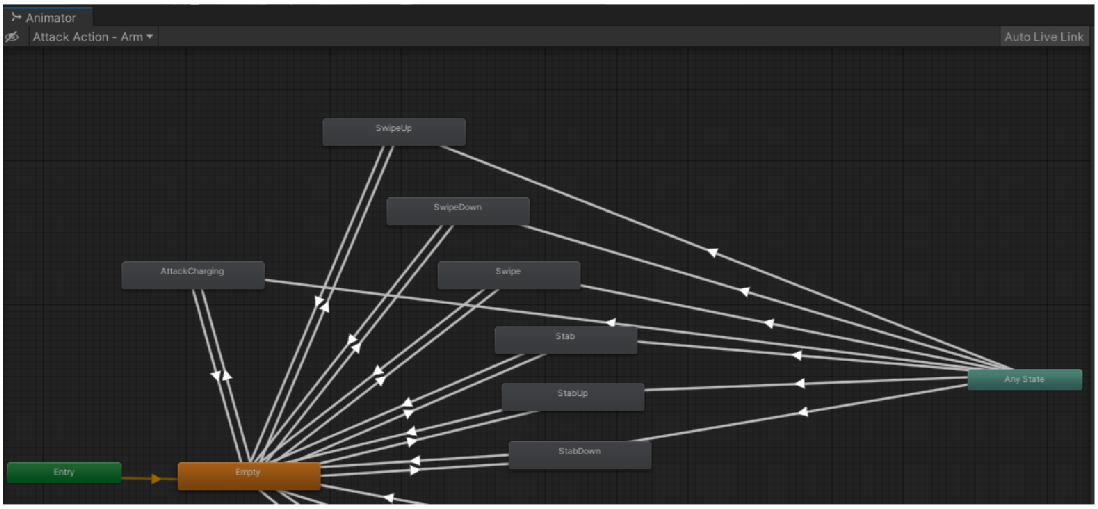
Pro tento styl animace je potřeba více spritů nebo jeden, který jich obsahuje více, jelikož Unity obsahuje sprite editor. Jedna z funkcí sprite editoru je rozpoznání jednotlivých spritů v souboru obsahujícím více spritů. Po rozdělení na jednotlivé sprity se přetáhnou do okna animace a nastaví se, jak rychle se mají sprity přepínat. Nově vytvořenou animaci poté se musí přiřadit do stavu animátoru.

Tento typ animace je lepší pro lehčí hry, kde charaktery nejsou modulové a nemusí se malovat tucet verzí. Například pro hru jako Bomberman nebo starý Super Mario Bros. Tento typ animace by se v tomto projektu dal použít například pro efekty. A opravdu je využit pro pasti.

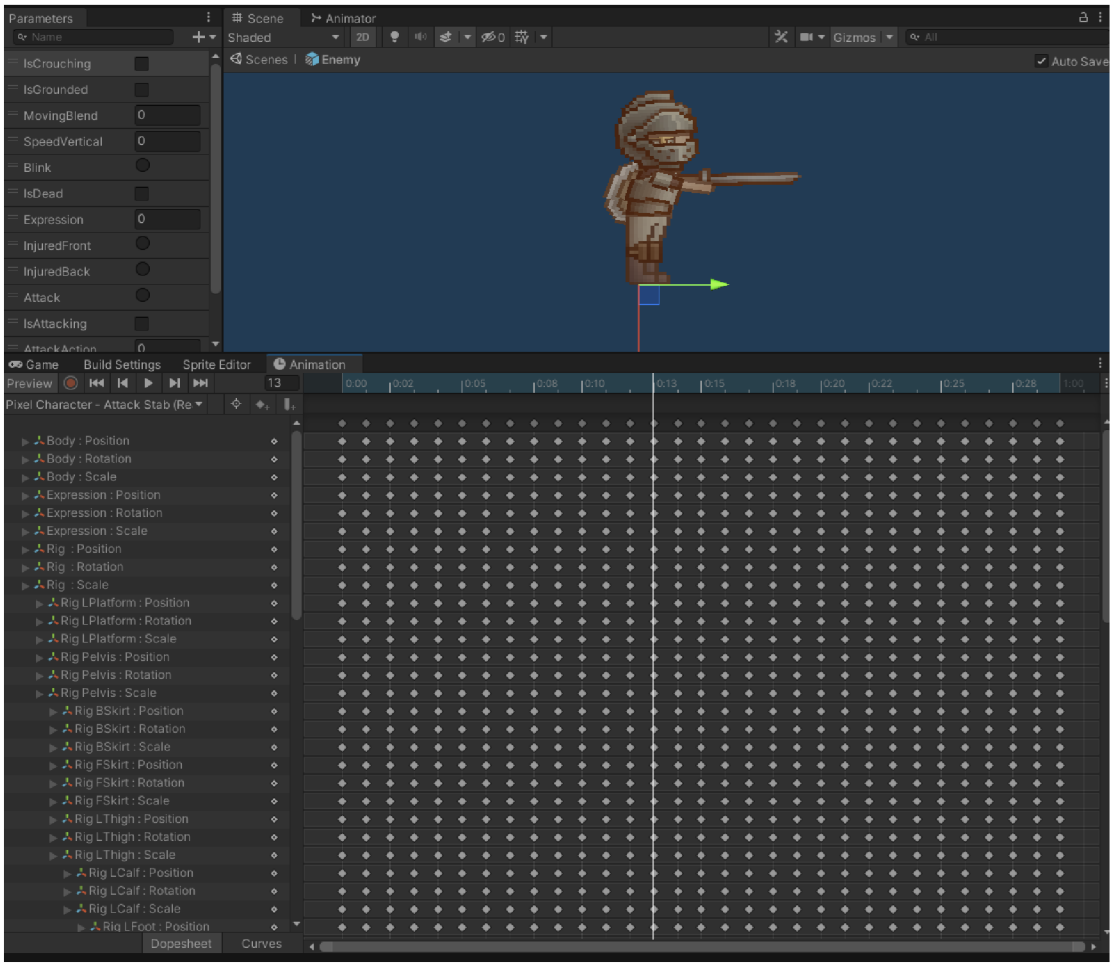
4.3.2 Nahráváním pohybu kostí

Tento styl vyžaduje sprite ideálně vytvořený na několika vrstvách. V sprite editoru se poté dají vytvořit "kosti"a k nim přiřadit jednotlivé sprity z oněch vrství. Existuje jedna základní kost, obvykle tělo a na ni jsou napojeny všechny ostatní. Kosti se pak přidají v okně Animation jako sada vlastností, pozice, rotace a měřítko. Jakmile jsou všechny kosti přidané, tak se zapne nahrávání animace a v časech se změní požadovaná vlastnost například rotace. Při spokojeností s animací se přiřadí do stavu animátoru.

Tento typ je lepší pro hry a objekty, které jsou modulárnější, například se mění zbraně, zbroje, či jiná část spritu. Tento typ je vhodnější pro charaktery v tomto projektu. Tento styl je využit pro hráče a nepřátele.



Obrázek 4.4: Výstřižek okna animátoru s ukázkou, jak můžou stavy vypadat



Obrázek 4.5: Výstřižek okna animace a aktuálního pohledu ve scéně, v horním okně je vidět aktuální vzhled v čase animace a ve spodním okně je jedna z nakoupených animací (převážně ukazujících složitost animování)

4.4 Prvky uživatelského rozhraní

Uživatelské rozhraní je velmi důležitou komponentou každé hry. Je zapotřebí, aby bylo jednoduché a přehledné, ale zároveň musí obsahovat všechny důležité informace.

Základem je objekt typu Canvas, jedná se o objekt obsahující všechny části uživatelského rozhraní. Pro každou část je výhodné si udělat prázdné objekty reprezentující jednotlivé části, což usnadňuje následující rozmisťování a udržování pohromadě. Většina segmentů se vyplatí udržovat u okraje, u jakého okraje záleží na typu hry a vzhledu rozhraní. Pro styl v tomto projektu jsem se rozhodl pro levý horní roh.

Pro veškeré reference na objekty v scény a operace s rozhraním je zde vytvořený a připojený k objektu Canvas, skript UIManager.

4.4.1 Životy a Mana

Životy i Mana jsou zobrazeny pomocí posuvníků. Posuvníky jsou nastaveny tak, aby umožňovaly pouze celá čísla, a jsou v základu nastaveny na 1. Tyto nastavení se nachází na hlavním objektu posuvníku. Objekt vytvořený při vytvoření posuvníku obsahuje tři další automaticky vygenerované objekty. První je objekt Background, pozadí, jedná se o jednoduchý objekt obsahující komponentu obrázku a nastavením této komponenty vlastním spritem vytvoříme vzhled pozadí, stálá část neměnící se s vyplněním, která obvykle bývá světlejší. Druhý objekt je Fill Area, objekt pro určení velikosti výšky a šířky výplně, obsahuje ještě jeden objekt. Tento objekt se jmenuje Fill, výplň, opět se jedná o jednoduchý objekt s komponentou obrázku, ale má nastaveno, aby se roztáhl po celém objektu Fill Area. Výplň obvykle bývá tmavší varianta pozadí, měl by tam být vidět rozdíl. Třetí objekt je podobný druhému jmenuje se Handle Slide Area, tento objekt je nechtěný proto minimálně objekt, co obsahuje je zapotřebí smazat. Poslední objekt je potřeba vložit, jedná se textový objekt, kde se vypisuje množství aktuálních životu/many z maximálního množství životů/many.

Oba dva posuvníky jsou ovládané ze skriptu hráče. Pro jejich ovládání je potřeba ve skriptu referenci na objekt, zmíněný objekt je zapotřebí najít nebo mít vložený. Pro uvedení příkladu řekněme že máme přes projekt vloženou referenci nebo nalezenou přes GameObject.Find("jméno");, tato reference se bude jmenovat HpBar. Poté HpBar.maxValue = Maximální hodnota;, HpBar.Value = aktuální hodnota;, jako poslední je potřeba přistoupit textové vyobrazení těchto hodnot, toho dosáhneme příkazem HpBar.GetComponentInChildren<Text>().text = Aktuální hodnota + "/ "+ Maximální hodnota;

4.4.2 Použitelné pomůcky a peníze

Vyobrazení pomůcek je vytvořeno ze sady dvou obrázků a jednoho textu. První obrázek je nastaven na UIMask, který je v základu volně dostupný v Unity a vypadá jako dobré pozadí. Druhý obrázek bude v základu prázdný a neviditelný. Neviditelnosti či úplné průhlednosti se dosáhne nastavením průhlednosti v nastavení barvy u komponenty Image. Při vygenerování hráče či při sebrání pomůcky se obrázek nastaví na viditelný a vloží se odpovídající sprite. Pro vložení správného obrázku a nastavení viditelnosti je opět zapotřebí reference, s jménem například Usable. Příkaz pro vložení obrázku je Usable.sprite = Resources.Load<Sprite>("Cesta ke spritu ve složce Resources bez přípony souboru"); a pro nastavení viditelnosti se použije Usable.color = Color.White;, kde Color.White je bílé světlo neovlivňující vzhled spritu. Při odstraňování pomůcky stačí nastavit viditelnost na nulu přes příkaz Usable.color = Color.Clear;, Color.Clear má všechny složky RGB nastavené na nulu

i s viditelností.

Peníze jsou vyobrazeny pod pomůckami pomocí objektu Text. Tento text se aktualizuje při sebrání mincí nebo při nákupu a nastavuje se podle uloženého souboru při vytvoření scény. Změna proběhne pomocí příkazem reference.text = "XXXX C"; kde XXXX představuje aktuální množství peněz a C je jen označení měny.

Obrázek 4.6: Výstřižek vzhledu uživatelských informací

4.4.3 Menu během hry

Jedná se o malé menu, které je snad absolutní nezbytností, jelikož umožňuje hráči vrátit se do hlavního menu nebo ukončit hru. U podobného menu se obvykle hra pozastaví, toho by se dalo dosáhnout přes pozastavením skriptů a komponenty RigidBody (komponenta pro simulaci fyziky). V tomto projektu k pozastavení nedojde, jelikož zde není žádný časovač a existují bezpečné body.

Menu je složeno z objektu panelu, tlačítek a textu. Panel je umístěný uprostřed, a buď se vytvoří z prefabu nebo jelikož je vždy jen jedno takové menu, je pravděpodobně lepší přímo vložené do scény, akorát neaktivované. Menu obsahuje celkem tři tlačítka:

- Continue/Pokračuj - Tlačítko opět schová menu. Na tlačítko je zde připojená funkce ClosePauseMenu() obsahující příkaz PauseMenu.SetActive(false);.
- Main Menu/Hlavní menu - Tlačítko přesměruje uživatele do scény hlavního menu a smaže objekty nastavené na "DontDestroyOnLoad". Nedojde k uložení hry. Na tlačítko je zde připojená funkce GoToMainMenu() obsahující příkaz Destroy, volaný pro úložný objekt, Canvas, a hráče.
- Quit/Odejít - Vypne hru a nedojde k uložení hry. Na tlačítko je zde připojená funkce Exit() obsahující příkaz Aplication.Quit();.

4.4.4 Mini mapa

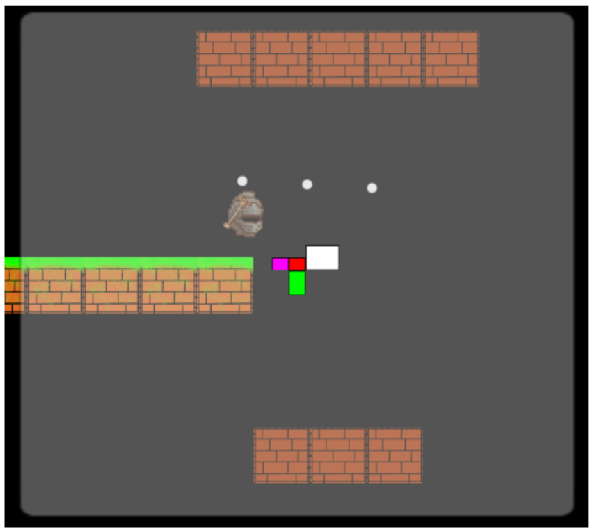
Mini mapa je další častou a důležitou součástí. Mini mapa zobrazuje rozložení úrovně, ať už vzdáleným pohledem na danou úroveň nebo pomocí vygenerování zmenšené a zjednodušené verze úrovně.

Umístění v projektu je pravý horní roh, v základu není mini mapa viditelná. Další validní umístění jsou pravý dolní roh a levý dolní roh, teoreticky i střed, ale střed není možný

při zobrazení mini mapy zároveň s menu a v levém horním rohu už jsou jiné prvky. Cílem je, aby se mini mapa nepřekrývala po zapnutí s dalšími prvky.

První možnost, vzdálené zobrazení úrovně by fungovala vytvořením druhé kamery, objektu Raw Image do objektu Canvas a Render Texture v okně projektu. Nová kamera by byla nastavená, aby zobrazovala nový Render Texture, stejný Render Texture by byl nastavený i v Raw Image. Aby byla tato mini mapa lépe viditelná potřebovala by pozadí, pro tento účel postačí objekt Panel. Výhoda této verze mini mapy je následování hráče pomocí skriptu použitém na hlavní kameře. Tento pohyb by mohl být výhodný při vygenerování extrémně dlouhé nebo široké mapy a možností mini mapu zmenšit.

Druhá možnost je mít opět objekt Panel na objektu Canvas a při vytváření mapy zároveň generovat i prefaby reprezentující dané místnosti, prefab je objekt Image s nastavenou velikostí, více v 4.9. U boss části úrovně se negeneruje nová mapa, jelikož je zbytečná. Výhodou tohoto stylu je snadnější zobrazení navštívených, aktuálních a doposud nenavštívených místnosti. Pro rozpoznání místnosti se přistupuje k proměnné color v komponentu Image, příkaz je reference.color = Color.Cyan; obsažený při vstupu nebo odchodu z místnosti. Na barvách nezáleží. Jde jen o to, aby byli rozeznatelné. V tomto projektu se používá bílá (Color.White) pro nenavštívené, tyrkysová (Color.Cyan) pro již navštívené místnosti, zelená (Color.Green) pro aktuální místnost, červenou (Color.Red) pro začáteční místnost a fialová pro místnost s vylepšeními. Začáteční místnost barvu nikdy nemění. Pohyb mini mapy by zde byl složitější a probíhal posouváním o velikost objektu, aby byla aktuální místnost vždy uprostřed.



Obrázek 4.7: Výstřižek ukazující vybraný styl mini mapy

4.4.5 Obchod

Obchod je část, kde se ve hře mění žánr roguelike na roguelite. Jedná se o objekt obsahující funkce nákupů vylepšení i nákupu pomůcek, ale ne kouzel. Objekt se zobrazí zmačknutím tlačítka Z, když hráč stojí na správném místě. Tlačítko Z je napevno nastavené v kódu a snímání probíhá příkazem Input.GetKeyDown(KeyCode.Z) v podmínce if, dále je v této podmínce, že se jedná o hráče. Kontrola že se jedná o hráče proběhne při vstupu do kolize

se spouštěčem nastavením proměnné, když v kolizi odpovídá jméno objektu.

Obchod je reprezentovaný panelem s tlačítky pro nákup a texty pro zobrazení ceny. Tlačítka jsou aktivní jen když má hráč dostatek peněz. Obchod je rozdělený na dvě části, obchod s pomůckami a obchod s vylepšeními a pro jednoduchost pojmenujme tyto části obchod a kovář. Obchod je potřeba odemknout za cenu 100 mincí, odemknutí je přiložené na tlačítku z funkcí BuyShopWindow(), tato funkce odečte peníze z uloženého souboru, aktualizuje text zobrazující tuto hodnotu, zapne objekt obsahující tlačítka pro nákup pomůcek a nakonec zkontroluje zda má hráč stále dost peněz na nákup a tato kontrola probíhá skrze vzorce pro aktuální cenu v porovnání k aktuálnímu množství peněz. Vzorce pro aktuální cenu využívají pro nákup pomůcek základní cena plus zvýšení krát počet již nakoupených pro aktuální běh a vylepšení je základní hodnota plus zvýšení krát počet vylepšení. Když hráč po nákupu nemá dost peněz tak se tlačítka vypnou.

Nákup samotných pomůcek probíhá přes tři tlačítka volajících funkci BuyUtility(int utility), kde utility označuje nastavené očíslování ve výčtovém typu pomůcek, využije se zde vygenerování nové pomůcky blíže popsané v 4.7. Touto funkcí vygenerovaná pomůcka se přiřadí do volného místa, pokud není volné místo jsou tlačítka vypnuta. Při přiřazování do místa pro pomůcky se také načte sprite reprezentující danou pomůcku. Opět se pro načtení spritu použije Usable.sprite = Resource.Load<Sprite>("soubor");. a jako po každém nákupu se aktualizují peníze, zde i cena další pomůcky a zkontroluje, zda může hráč dále nakupovat. Cena pomůcek se zobrazuje vedle tlačítek pro nákup.

Strom pro vylepšení je schovaný na posuvném okně, okno v základu obsahuje tlačítka s prvními vylepšeními. Tlačítka mají omezení na maximální možnou úroveň. Dále obsahují hodnotu vylepšení za úroveň, aktuální úroveň, co se načítá z úložného souboru a odkazy na předchozí a následující vylepšení. Pokud je vylepšení zakoupené, aktivuje se tlačítko pro další vylepšení.

4.4.6 Okno pro odměny za poražení

Okno pro odměny za poražení bossů, či specifických úrovní. Panel se skládá ze čtyř částí, každá obsahující tlačítka pro volbu bonusu. Hráč si vždy může vybrat pouze jeden bonus v každé řadě. Tlačítka jsou vypnuta, pokud není v uloženém souboru uvedeno, že hráč porazil dané patro. Tlačítka se odkazují na funkci SelectNewRewardN(int RewNum), kde N reprezentuje část kde se tlačítka nachází a RewNum označuje číselnou hodnotu pro přepínač. Při výběru se změní vybraná hodnota uložená v úložném souboru, tato hodnota určuje, které tlačítko je vybrané při načtení scény a jaké bonusy dostane hráč. Při změně je zapotřebí přiřadit nové bonusy, ale také odebrat předchozí. První a třetí sada bonusů obsahuje výběr mezi za prvé o jeden úskok víc, ale kratší úskoky, za druhé rychlejší obnova úskoků, ale opět kratší úskoky a za třetí delší úskoky. Druhá a čtvrtá sada bonusů obsahuje výběr mezi dalším úskokem a celkem delšími úskoky nebo dalším skokem.

4.5 Zbraně

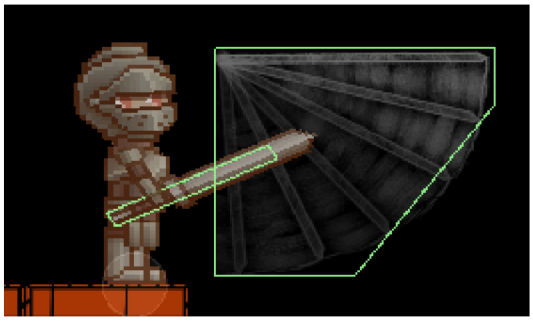
Zbraň je model reprezentující všechny informace potřebné k útočení. Model se skládá z listu modelů pojmenovaných Dmg, Dmg je model pro uložení množství a typu zranění. Mezi typy zranění patří fyzické, ohnivé, vodní, zemní, vzdušné, světlé, temné, léčení a typ

žádný, který je základ pro model Dmg a není udělaný, aby se dostal do hry. Typ léčení není pro zraňování nepřátel, ale pro léčení uživatele zbraně. Fyzické, ohnivé, vodní, zemní, vzdušné, světlé, a temné slouží jako různé typy zranění pro různé typy imunit. Další je počet projektilů a jejich úhel, dosah a vlastnost ovlivňující rozsah komponent kolize se zapnutým spouštěčem, rychlost projektilu u luku nebo vzdálenost jakou může projektil hůlky urazit. Třetí je modifikátor rychlosti útoků, modifikátor násobí dobu mezi útoky. Čtvrtý je typ zbraně, který ovlivňuje mnoho věcí, mezi takové věci číslo nepřátel, které lze trefit jedním útokem, tvar kolize a sprite zobrazený na charakteru. Pátou položkou je velikost projektilu, tato položka ovlivňuje jen zbraně typu luk nebo hůlka, jedná se o průměr kolize projektilu. Šestou položkou je rychlost projektilu, další modifikátor ovlivňující rychlost a vzdálenost jenž šíp urazí nebo rychlost projektilu hůlky ovlivňující jen za jakou dobu vzdálenost urazí. Dále je zde možné vložit, zda projektil bude mít gravitaci, zda projektily explodují nebo zda se rozbijí až kolizí s blokem.

4.5.1 Kolizní komponenty

Pro tento projekt jsou vytvořeny dva typy kolizí pro zbraně z blízka. Jednoduché obdélníky, kde dosah ovlivňuje délku kolizního komponentu a jeho posun. Délka je rovnou vložena, zatímco posun je vypočítán vzorcem, kde hodnota x posunu je desetina plus polovina krát dosah zbraně. Velikost i posun jsou reprezentovány v Unity vektorem a tento projekt zajímá v tomto případě jen vektor2 obsahující dvě hodnoty, x a y. Druhý typ kolize je polygon s pěti body 4.8. Pro tuto kolizi je třeba nastavit všechny body:

- První bod musí mít nastavené hodnoty pozičního vektoru na x i y rovno polovině
- Druhý bod musí mít nastavené hodnoty pozičního vektoru na x mínus polovinu a y rovno polovině
- Třetí bod musí mít nastavené hodnoty pozičního vektoru na x mínus polovinu a y rovno polovině
- Čtvrtý bod musí mít nastavené hodnoty pozičního vektoru na x nulu a y rovno mínus polovině
- Pátý bod musí mít nastavené hodnoty pozičního vektoru na x polovině a y rovno čtvrtině



Obrázek 4.8: Výstřižek ukazující možný vzhled polygonové kolize

4.5.2 Projektily

Projektil je uložen jako prefab. Prefab se vytvoří na aktuální pozici hráče s přidáním sedmi desetin na ose y. Do něj je nutné ihned po vytvoření vložit aktuální zbraň a směr útoku. Pro vložení těchto dat se u nově vytvořeného objektu získá přes objekt.GetComponent<ProjectileControl>(); tato funkce vrátí referenci na skript projektilu, pak můžeme přistoupit na jednotlivé elementy jako je zbraň a proměnné pro směr útoku jménem AttackX pro x hodnotu vektoru a AttackY pro y hodnotu vektoru. Při vytvoření a přiřazení proměnných se ve funkci Start najde hráčův skript přes funkci GameObject.FindGameObjectWithTag("Player") .GetComponent<PlayerView>();. Dále se podle zbraně nastaví například gravitace, nastavení gravitace proběhne přes vyhlednání komponenty RigidBody2D objektu a nastavením RigidBody2D.gravityScale = 1; pro zapnutou gravitaci a 0 pro vypnutou. Dále se liší nastavení podle toho, zda zbraň je luk nebo hůlka. Pro luk je posun po ose y provedený aplikací síly. Po přidání síly se nastaví opět u Kolize.offset na nový vektor s posunem na ose x o dvě desetiny, aby byla komponenta kolize na části objektu, kde se zobrazí hrot šípu. Jako poslední se pomocí komponenty Sprite Renderer nastaví vzhled projektilu na šíp a sprite šípu se načte pomocí již zmíněné funkce Resource.Load. Pro hůlku stačí nastavit barvu spritu podle typu prvního zranění na zbrani. Výběr barvy je napevno daný pro typ zranění. Výběr barvy je jednoduchý přepínač podle prvního typu zranění. Jelikož projektil musí být viditelný, tak se nastaví nová pozice v komponentě Transform, vlastně se jedná o stejnou pozici na 2D ploše, ale pozice z bude -5, aby byl projektil před objekty nepřátel a hráče.

Projektil využívá pro posun ve scéně funkci FixedUpdate. Zde se pro hůlku počítá uražená vzdálenost přes vzorec vzdálenost += (new Vector3(AttackX, AttackY, 0).normalized * Time.deltaTime * Weapon.ProjectileSpeed * 0.5f).magnitude;, stejný vzorec až na .magnitude se přičte i do komponenty Transform, jako poslední to zkontroluje jestli se nerovná dosahu na zbrani plus jedna. Pokud byla dosažená maximální vzdálenost projektil sám sebe zničí příkazem Destroy(this.gameObject);. Dále je ve funkci FixedUpdate otáčení podle aktuální rychlosti. Otáčení probíhá funkcí this.gameObject.transform.rotation = Quaternion.AngleAxis (Mathf.Atan2(ProjetileRigidBody.velocity.y, ProjetileRigidBody.velocity.x) * Mathf.Rad2Deg, Vector3.forward);.

Projektil potřebuje snímat kolize, na to se využije funkce OnTriggerEntry2D. Přes komponentu kolize jenž je parametrem funkce, lze přistoupit na herní objekt a na něm se nachází jméno kolizního herního objektu, kde lze přistoupit k funkci name.Contains("hledaný výraz"). Mezi hledané výrazy patří řetězce "Player", "Enemy"a "Block". Při kontrole, zda projektil narazil do hráče, je dále třeba zkontrolovat, že náraz není do kolize se spouštěčem nebo do dalšího objektu hráče (všechny objekty vytvořené pro hráče obsahují ve jméně "Player"). Kontrola může proběhnout přes označení/tag místo jména. Stejně to platí pro nepřítele.

Při nárazu do hráče se zavolá funkce ze skriptu hráče skript.TakeDmg(List<Dmg> zraněníVeZbrani). Poté se zkontroluje, zda zbraň není nastavená, aby penetrovala více objektů nebo aby projektil vybouchl, když je výbušná a není nastavená pro průraz více objektů se zavolá Destroy(this.gameObject);, což projektil zničí.

Když dojde k nárazu do nepřítele, přidá se objekt nepřítele do listu trefených objektů. A jako u hráče se zkontroluje, zda zbraň nemá nastavené, aby penetrovala více objektů nebo explodovala, když je výbušná a není nastavená pro průraz více objektů se zavolá Destroy(this.gameObject);, což projektil zničí.

Sražení s Blokem rovnou zničí objekt funkcí Destroy(this.gameObject);.

Když je objekt zničen, zavolá se event OnDestroy, zde se zavolá funkce Explode(), pokud je tak zbraň nastavená. Samotná funkce Explode(), vytvoří nový objekt "Explosion"blíže popsaný v 4.6.2.

4.6 Pomůcky

Pomůcky jsou objekty usnadňující průchod hrou. Mohou sloužit pro doplňování využitelných zdrojů nebo jako další varianty zranění. Všechny pomůcky jsou nějakým způsobem omezené, ať již počtem použití nebo manou. Pomůcky se získávají za prvé nákupem v obchodě nacházejícím se v lobby, nebo když vypadnou náhodou ze zabitého nepřítele nebo zničeného bloku. Pomůcky nemají být příliš silné, aby hráči stačilo používat jen je.

Model pomůcek obsahuje typ pomůcky, List zranění, jenž je stejný jako se nachází ve zbrani a udává veškeré doplňování i zranění, dosah opět jako ve zbrani, dosah je přítomen jen pro kouzla a specificky s typem projektilu. Počet použití označuje, kolikrát může hráč použít nekouzelnou pomůcku, než o ni přijde. V základu je počet použití nula pro kouzla a bomby (bomby musí mít počet použití definovaný v souboru) a tři pro elixíry, nakonec je v modelu typ kouzla v základu nabývající hodnoty projektil i když se nejedná o kouzlo.

4.6.1 Elixír života a many

Elixíry doplňují jeden ze dvou zdrojů, životy nebo manu. Elixíry jsou napevno definované kódem vždy 3 použití a hodnota obnovení je vypočtena ((základní hodnota plus vylepšení zranění) plus pět krát (číslo úrovně + vylepšení škálování)).

4.6.2 Bomba

Bomba funguje jako rychlé plošné zranění. Bomba má na sobě časovač, který je ve funkci FixedUpdate snižovaný pomocí Time.deltaTime(změna času mezi snímky). Když dojde čas, tak se objekt zničí a v OnDestroy se vytvoří objekt exploze, kterému se vloží zranění z modelu bomby.

Objekt představující explozi je nehybný objekt opět s časovačem. Objekt obsahuje komponentu kolize se spouštěčem a zraňuje jako zbraně. Po vypršení času zmizí.

4.6.3 Kouzla

Kouzla fungují jako další možnost útoku. Jsou zde tři typy: projektil fungující jako projektil hůlky; rapidní plamenomet magie, jenž má nižší poškození, ale zraňuje častěji; explozivní magie fungující jako bomba, ale místo omezení počtu použití využívá magii.

Plamenomet magie se projevuje spoušťovou kolizí a spritem na hráčově objektu SpellThrowerDummy. Hráč má herní objekt s odsazením, který se otáčí a posouvá podle vektoru zadaných útoků. Přesun a otočení probíhá přes funkci objekt.transform.SetPositionAndRotation (aktuální pozice hráče plus odsazení, Quaternion.Euler(x, y, z));, funkce potřebuje novou pozici a rotaci určenou v stupních na jednotlivých osách, pro tento projekt se otáčí jen kolem osy z. Samotná spoušťová kolize je časovaná a po vypršení se smaže.

Magie projektilů funguje jako projektily hůlky a explozivní jako exploze bomby.

4.7 Avatar hráče

V lobby a herní scéně hráč ovládá hru přes svůj objekt. Jak je již zmíněno vzhled a animace jsou zakoupené na Unity Asset storu. Objekt hráče je vložený do scény Lobby a do herní scény se dostane přes "DontDestroyOnLoad". Objekt se skládá z častí prefabu z Asset storu, objektu pro kolize útočení, objektu pro kouzlení a objektu snímajícího kolize ze spodku pro skok na nepřátelích.

Hlavní objekt Player obsahuje dva kolizní komponenty, kapslovou bez spouštěče pro pohyb po terénu a krajovou se spouštěčem pro snímaní přistání ze skoku. Dále obsahuje dva skripty, jeden ovlivňující vzhled a animace, druhý pro ovládání a nakonec má ještě komponentu RigidBody2D pro fyziku. AttackDummy, objekt pro kolize útočení má na sobě skript pro vytváření kolizí. Další objekt je AttackJump s komponentou spoušťové kolize ve tvaru obdélníku a skriptem snímajícím dané kolize s odpočtem mezi snímáním, a nakonec SpellThrowerDummy pro vytvoření kolize plamenometné magie popsané v 4.6.3 s přiloženým skriptem snímající zranění způsobené magií.

Základ funkce hráče je rozdělen do funkcí Start, Update, FixedUpdate a TriggerEntry2D.

Funkce Start vytvoří skript pro ovládání modelu, který jsem si pojmenoval PlayerViewModel (pvm), dále najde skript úložného objektu a kolize na hlavním objektu. Dále si vybere pomocí funkce pro náhodný výběr čísla z rozsahu nula až počet odemčených stylů charakterů pro výběr základních statistik hráče. Tato náhodná hodnota se převede na výčtoví typ PlayerClass a jeho řetězcová hodnota se vloží jako parametr do funkce pvm.SetBasePlayer(PlayerClass);. Funkce SetBasePlayer přečte soubor pro vybrané povolání a vyplní svůj model přečtenými daty. Dále se vyplní uživatelské rozhraní, počet životů, many a zobrazení pomůcek. Jelikož pro první kontrolu možnosti koupit pomůcky musí hráč být načtený, tak se zde zavolá i první kontrola. Nakonec se vloží odměny za poražení úrovní a správný obrázek zbraně.

Update je zde pro snímání vstupů uživatele a kontrolu podmínek. Vstup uživatele se snímá přes funkce Input.GetKeyDown(Kód tlačítka). Dále se zde kontroluje, zda hráč zemřel nebo vyhrál.

FixedUpdate je funkce, kde se probíhá většina funkčnosti. Většina hráčových funkcí má určitý čas trvání nebo dobu mezi aktivacemi, odpočet funguje pomocí proměnná -= Time.deltaTime;. Po odpočtech je kontrola, zda časovače dosáhly nuly a přenastaví se pravdivostní hodnoty. Mimo jiné je zde i kontrola zabitých nepřátel a bossů pro nastavení možnosti přesunu mezi úrovněmi. Dále je zde cyklus pro zraňování nepřátel z listu trefených nepřátel, tento cyklus se přeruší, pokud počet zásahů přesáhne počet možných zásahů pro zbraň. Nakonec se zde volají jednotlivé funkce, pohyb, útok a použití pomůcek.

4.7.1 Model

Základ modelu je využíván jak nepřítelem tak i hráčem, základ se jmenuje CharacterModel a obsahuje maximální počet životů a many, aktuální počet životů a many, modifikátory rychlosti útoku a pohybu, staticky určené objekty pro povolání (kousky zbroje) s obranou proti zranění, zbraň a pole tří pomůcek. Hráčův model poté obsahuje maximální množství

úskoků, aktuální množství úskoků, čas dobíjení úskoku a počet skoků. Všechny operace i samotný model hráče je součástí PlayerViewModel skriptu.

4.7.2 Pohyb

Funkce pro pohyb hned volá funkci pro úskok, v této funkci je celé ovládání úskoku. Pokud hráč má úskok a snaží se pohnout během přidržení nastaveného tlačítka, nastaví se mu ignorování kolizí s nepřáteli (kolize se vypínají mezi vrstvami), nastaví se nová rychlost v daném směru, do listu dobíjení se přidá čas a nastaví se čas trvání úskoku a pravdivostní hodnota, že hráč uskakuje, aby nemohl se během úskoku hýbat a útočit. Když vyprší čas úskoku nastaví se pravdivostní hodnota úskoku na nepravdu, zapnou se kolize a nastaví se čas během kterého nejde znovu uskočit. Pokud hráč neuskakuje, tak může buď skočit přidáním síly nebo se posunout upravením pozice objektu, aby byl posun plynulý násobí se změna Time.deltaTime.

4.7.3 Útok a pomůcky

Útok probíhá buď zbraněmi nebo využitím pomůcek.

Útok zbraní je dělený do sekvencí, podle sekvence se zvolí, jaký bude tvar a velikost kolize. Sekvence má určitý čas trvání, po kterém se resetuje. Útok zbraní zblízka jako takový probíhá podobně jako plamenometná magie, objekt se posouvá a rotuje podle směru útoku s tím rozdílem, že útok zbraní je jen do čtyř hlavních směrů, nahoru, dolu, doleva a doprava. Při vytvoření kolize se nastaví čas, po jakou dobu kolize bude existovat, při zničení kolize se začne odpočítávat čas mezi útoky. Čas mezi útoky je násoben s modifikátory rychlosti útoku, a v základu je jedna sekunda. Pokud hráč zaútočí dolů aktivuje se objekt AttackJump na hráči, který pokud zasáhne nepřítele odrazí hráče nahoru. Tento odraz není zrovna silný, takže pokud nejsou útoky rychlé, jedná se spíše o zpomalení pádu. Útok zbraní na dálku je jednodušší, neboť se vytvoří projektil a nastaví se mu zbraň a směr útoku. Na rozdíl od útoku zblízka je dálkový útok do osmi směrů. Jelikož každá zbraň a nepřátelé obsahují počet sériových a paralelních projektilů, plus úhel prvního projektilu a posun pro každý další, je zapotřebí získat si útočící úhel podle uživatelského vstupu a následně přepočtu úhlu prvního projektilu a každého následujícího. Každá zbraň má po zásahu nepřítele jiný efekt. Některé odrazí dále, některé ignorují zbroj a jiné způsobí nepřátelům zranění za sekundu nebo si zvyšují poškození množstvím zásahů.

Poslední výraznou funkcí hráče je možnost sebrat objekty vypadlé z bloků a nepřátel. Když hráč vstoupí do zóny sebrání, najde se mu obsah objektu a při zmáčknutí klávesy se poté sebere. Tyto objekty je vhodné vyobrazit v okně, aby hráč věděl, co vlastně má možnost sebrat.

4.7.4 Zranění

Hráč, když obdrží zranění, tak nejprve zkontroluje, zda v nedávné době již zranění neobdržel, a tudíž není chvilku nezranitelný. Pokud je zranitelný, tak se aplikuje každá položka listu zranění, tyto zranění jsou modifikovány zbrojí. Po aktualizaci životů v modelu se aktualizují životy na obrazovce a nastaví se čas nesmrtelnosti.

4.8 Nepřátelé a pasti

Nepřátelé jsou zranitelní oponenti hráče oproti pasti, kterou hráč nemůže zničit a obvykle ani zranit. Dle pravidel roguelike by měli mít hráč a nepřítel stejná pravidla, ale jelikož by nepřátelé byli příliš složití na poražení, musí mít jistá omezení a jisté bonusy. Mezi omezení patří nabíjení útoku, kdyby zaútočil hned, když je hráč v dosahu, hráč by se nemohl útoku vyhnout. Mezi bonusy patří pro některé možnost létat. Pro ulehčení nepřátele neskáčou, složitost naučení skákání za hráčem by skončila pády mezi místnostmi. Aby nepadali, každý nepřítel si vytváří vlastní paprsek (Raycast2D), pokud tento paprsek nezasáhne vrstvu bloků, nepřítel se zastaví.

V projektu jsou implementované tři typy nepřátel. První typ "normální"může mít všechny typy zbraní a normálně se pohybovat. Druhý typ "nabíhající"je určen k tomu, aby neustále běžel za hráčem a tím ho zranil. Třetí typ je "statický"tento typ může mít jen zbraně na dálku a nemůže se hýbat. Dále je tento typ rozdělen na dva typy míření, a to na hráče nebo na neměnnou pozici.

Nepřátelé jsou funkčně velice podobní, směr pohybu a útoku se určuje po aktivaci (vstup hráče do místnosti) podle aktuální pozice hráče. Výrazné funkční rozdíly jsou neschopnost používání pomůcek, určování směru dle hráče, útok projektilem v kružnici. Když trefí hráče, lehce ho odhodí a když je trefen, je sám odhozen, obě odhození jsou provedena aplikováním síly na RigidBody2D a směr je určen odečtením pozic. Rozdíly v modelu jsou drobné, základ CharacterModel je stejný, ale nepřítel navíc obsahuje, zda může létat, jak dlouho se napřahuje k útoku, svůj objekt, zda je aktivován, počet a úhel projektilů, a jeho počáteční souřadnice potřebné pro restartování nezabitých nepřátel po opuštění místnosti.

Když nepřítel zemře, zavolá se funkce, kde se podle šance vytvoří objekt k sebrání. Mezi tyto objekty patří peníze, pomůcky a nové zbraně. Funkce třikrát vybere náhodné plovoucí číslo mezi nulou a jedničkou. Pokud je číslo dostatečně velké, vytvoří se jeden objekt k sebrání.

4.8.1 Boss

Boss je v této hře nepřítel, který je sám na patře ve své aréně. Poražením bosse se umožní pokračovat do další úrovně s náhodně vybraným bonusem do statistik, nebo se ukončí hra výhrou a přesune hráče do lobby. Boss na rozdíl od normálních nepřátel je definovaný v kódu a jen jeho arénu lze předefinovat. Boss má více typů útoků, kde se náhodně střídají, model má stejný jako nepřítel.

Aktuální boss je na vrcholku místnosti a hráč se k němu musí proskákat, zatímco se vyhýbá projektilům a pastem, jakmile se k němu hráč dostane, tak začne používat slabší a pomalejší útok z blízka. Tímto způsobem je vytvořen, aby bylo složitější se k němu dostat a hráč musel využívat většinu svých mechanik, aby přežil cestu. Zároveň i když je slabší z blízka, stále to neznamená, že cesta je jediné, kdy se hráč musí vyhýbat útoku. Útoky, které provádí na dálku jsou projektily padající přes šířku místnosti nebo několik mířících na hráče a aktivace pastí v aréně. Na blízko švihne pod sebou zbraní, která hráče zraní a odhodí, pokud se jí nevyhne.

4.8.2 Pasti

Pasti jsou objekty, které mohou zranit hráče a hráč je nemůže zničit. V tomto projektu se aktivují nezávisle na hráči, ale podle času aktivace nebo externím příkazem (útok bosse).

Ve funkci Start si nalezne hráčův skript pro udílení zranění hráči, načte aktuálně vybranou past (Pro každou místnost se vybírá náhodně jiná), nastaví se časovače a velikost spoušťové kolize, a nakonec se past aktivuje, pokud nemá nastavený aktivační čas.

Ve funkci FixedUpdate se odpočítávají časy aktivace a aktivity. Pokud čas aktivace klesne na nebo pod nulu a past není aktivovaná, tak se aktivuje. Když čas aktivity uplyne, past se deaktivuje. Aktivace a deaktivace probíhá vypnutím spoušťové kolize pro zranění hráče (zranění hráče je jako v ostatních objektech).

Model Pasti obsahuje list zranění jako zbraně, pravdivostní hodnotu, zda je aktivovaná, základní aktivační čas, základní čas aktivity, šířku a výšku.

4.9 Generování mapy

Herní svět lze vytvořit třemi odlišnými způsoby, kde každý má své výhody i nevýhody.

Předem vytvořený svět Předem vytvořený svět je asi nejlehčí, nejstarší a doposud nejvyužívanější způsob. Takový svět trvá celkem dlouho na vytvoření a mění se jen přes předem nastavených spouštěčů nebo aktualizací hry. Ale samozřejmě jsou zde i výhody a největší z nich je, že svět je udělaný na míru pro daný žánr a hru. Proto je optimální u her silně zaměřených na příběh a obtížnost. Jako příklady lze uvést hry jako Ori and the Blind Forest, Hollow knight nebo méně spojitelné s tímto projektem assassin's creed.

Generovaný svět Generovaný svět může být celkem lehký, ale i extrémně složitý na vytvoření. Dost záleží na velikosti, komplexnosti světa a z čeho se skládá. V tomto projektu je sice jako nejmenší jednotka pro tvorbu blok, ale pro generování světa nebo úrovně se využívají místnosti složené z bloků. Takto je svět sice generovaný, ale stále celkem dělaný na mírů. Tento způsob zaručuje hratelnost za cenu většího množství kombinací. Svět vytvářený o hodně náhodněji je vhodný například pro hry o přežití, kde hráč může ovlivňovat tvar světa, jako například v Minecraftu a Terrarii, hry, které mají stálejší pohyb nebo jednodušší tvorbu pravidel, aniž by musely obětovat hratelnost kvůli množství kombinací.

4.9.1 Úroveň

Součástí definice roguelike je náhodná mapa, měnící zážitek po každé smrti. Úplně náhodná mapa je příliš složitá pro 2D hru z boku s modifikátory rychlosti a výšky skoku. Další nejlepší možnost jsou definované místnosti a jejich náhodné rozmístění. Design místností je takový, aby je hráč mohl proskočit do dalších místností. U bosse je aréna buď jako skákačka, během které se vyhýbá útokům a snaží se dostat k bossovi a porazit ho.

Při přesunu do herní scény začne generování umístěním začáteční místnosti (jediná zaručená místnost) do středu jako počátečního bodu. Počáteční místnost vždy vypadá stejně, jedná se o malou místnost se čtyřmi cestami. Náhodně se vybere, kudy generování půjde a jak velkou místnost s jakým posunem vygeneruje. Kořenová místnost pro další generování se mění náhodně se zvyšovanou šancí čím méně cest existuje. Při generování je potřeba kontrolovat, zda do vybrané pozice místnost vejde. Následující generování probíhá, dokud nedosáhne hodnota místností úrovně nuly. Po konci generování proběhne kontrola míst kolem každé místnosti a každých dveří, které nevedou do sousední místnosti, jsou za generovány bloky.

Zmíněné kontroly probíhají přes 3D paprsky (3D RayCast), které nesnímají 2D položky a musí být převedeny do 2D prostoru funkcemi: new Ray (objekt místnosti.transform

.position + new Vector3 (posun), Vector3.forward); pro vytvoření 3D paprsku (3D jelikož chceme paprsek do hloubky) a Physics2D.GetRayIntersection (ray); pro získání kolizí s 2D prostorem. Kontrolu můžeme provést jednoduše, jelikož místnosti mají pevně dané velikosti, k nimž máme přístup.

Samotné místnosti jsou definované v souboru číslem určujícím pozici a hodnotou určující, jaký blok či nepřítel se má na pozici objevit. Model místnosti obsahuje pole dveří, vytvářející obousměrný seznam místností, velikost místnosti, svůj objekt a zda je místnost vertikální.

Blok je základní částice místnosti o velikosti 256 pixelů, což v unity nastavíme jako velikost jednoho políčka. Existuje zde více typů bloků, s kolizemi, bez kolizí, poloviční, rozbitelné a zpomalovací. V souboru z místností jsou definovány i pozice možných nepřátel a pastí.

Kapitola 5

Testování

Tato kapitola je kratší souhrn formy vývoje a testování. Testování zde proběhlo ve dvou hlavních fázích, osobní vývojové testování a testovaní na uzavřené skupině, které by obvykle bylo nazvané alfa testováním.

První fáze byla postupné testování aktuálně vyvíjených funkcí, už během jejich vývoje. Lehčí funkce byly prvně vytvořeny a následně testovány. Složitější části, jako generování mapy, byly testovány a upravovány dle potřeby ještě během vývoje. Mnoho funkcí bylo změněno po osobním testování a následné nespokojenosti s jejich podobou. Mezi prvně vyvinuté funkce patřil pohyb a základní útok. Tyto funkce byly následně upravovány do komplexnějších podob. Do pohybových funkcí se přidal úskok pro umožnění větších a složitějších segmentů skákačky a později pro vyhýbání se nepřátelům. Po vyvinutí první uspokojivé funkčnosti hráče byli vytváření nepřátelé, jelikož mají být podobní hráči. Nepřátelé zde nejsou příliš chytří a soustředí se prakticky jen na pozici hráče, což bez jistých oslabení z nich udělalo neporazitelná monstra. Tudíž byla přidána omezení na rychlost jejich reakce na útok, aby hráč měl šanci se vyhnout.

Na generování úrovně bylo vynaloženo poměrně dost času, už jen v návrhové části a po vytvoření návrhu generování úrovní po místnostech začal vývoj. Tato funkce byla testována během vývoje a postupem času nebyla příliš měněna. Zde byli pouze postupně nacházeny a opravovány vzácné chyby a popřípadě přidávány typy stavebního materiálu místností, jako jsou například zničitelné a zpomalovací bloky nebo pasti.

Vylepšování podlehlo asi největším změnám. První varianta vylepšování základních statistik, jako jsou útok nebo životy, byla změněna na větší strom základních vylepšení, odemykaní zbraní a funkcí. Změna proběhla, jelikož mi přišla první verze vylepšování nezáživná, netaktická a bez celkově hloubky.

Jako poslední byl přidán editor, který nebyl v původním návrhu, ale celá hra byla vytvářená tak, aby byl uživatel schopný lehce upravit zážitek a tomu má napomoci editor. Mimo jiné editor usnadnil tvorbu herních dat i během zbytku vývoje.

Druhá fáze zahrnovala zpřístupnění anonymního testování na uzavřené skupině obsahující kolegy studenty z fakulty. Studenti byli požádáni o vyplnění krátkého formuláře a zahrání si aktuální sestavy. Bohužel i když hru spustilo možná více lidí, formulář vyplnili jen tři lidi a jeden mi napsal detailnější odpověď mim formulář. Všichni testeři spustili sestavu bez problému. Orientace v menu byla ohodnocena jako nadprůměrná a doporučení na vylepšení byli vzhled uživatelského rozhraní a nepoznání znaků podle označení unity. Intuitivnost ovládání byla lehce podprůměrná, proto byl přidán panel s ukázaným ovládáním v lobby.

Jasnost cíle hry byla absolutně nejasná, takže byl přidán panel objasňující, že hráč má porážet nepřátele, aby se mohli posunout dál. Z hratelnosti obtížnost byla označena jako nadprůměrná, což je lehce způsobeno daty upravenými pro testování, alespoň podle jednoho vypsanějšího reportu a oprava by byla jen upravit zbraně a nepřátele vytvořené pro vyzkoušení různých možností. Zážitek ze hry byl ohodnocen jako podprůměrný, převážně kvůli ovládání, zejména kvůli více pohybu. Během tvorby hráče pohyb byl několikrát předěláván a ke konci vývoje měl několik schovaných funkcí, na které měl hráč přijít. Ty však byly pravděpodobně obtížné na objevení a otravné pro nové hráče. Tudíž byl jako oprava pohyb opět zjednodušen. Jako poslední bylo vyznačit nejasné funkce bonusů, které jsou k sebrání ve speciální místnosti na každém patře.

Kapitola 6

Závěr

Jedním z mých cílů bylo naučit se pracovat s enginem Unity a nastudovat si alternativy. Cílem projektu však bylo získat informace o historii her, a nakonec vytvořit 2D hru. V této hře musí hráč porážet nepřátele, kteří jsou na začátku hry mnohem složitější než později, kdy si hráč nakoupí vylepšení.

Při tvorbě práce jsem si nejprve zjistil co nejvíce o používaných žánrech, aktuálně existujících hrách a o tvorbě v Unity. Historie a aktuální hry mi pomohly definovat, jak mohu žánry využít a co byly nejúspěšnější části těchto her. Vyzkoušení zmíněných her a učení se o práci v Unity mi umožnilo postupně dělat úpravy v návrhu hry, který byl z počátku poměrně hrubý.

Jelikož jsem měl alespoň hrubý návrh, během úprav již probíhala samotná implementace tohoto návrhu. Naštěstí se velice zřídka stalo, že bylo třeba měnit návrh již na vyvinutých částech. Finální verze implementace byla následně otestovaná a došlo k pár dalším změnám.

Nakonec bych zmínil, že práce mi přinesla spoustu nových znalostí k Unity, ale dokonce i jazyku C#. Mezi znalosti o Unity patří naučení se nových předtím mě neznámých funkčností samotného enginu, které usnadňují práci. Mimo jiné jsem si vyzkoušel, jak složité je dělat sprity a animace do her a to ještě bez přidání třetí dimenze do hry. Samotný vývoj byl velice zábavný a přínosný, ukázal mi práci na větším projektu a složitost tvorby nezávislých her. Přesto mě přesvědčil, že bych rád pokračoval v tvorbě her i nadále.

Dle zadání je vytvořena buď kratší hra nebo demo větší hry. Další práce by obsahovala rozšíření, která mě jako vždy napadají až když je pozdě na samotnou jejich implementaci, taková práce by obsahovala vytvoření dalšího obsahu, více typů nepřátel, bossů, typů terénu (bloků), stylu útoků a zakomponování příběhu.

Literatura

[1] Crytek. *Cry Engine* [online]. Crytek, 2022 [cit. 2022-25-4]. Dostupné z: https://www.cryengine.com/features.

[2] EpicGames. *Unreal Engine* [online]. Epic, 2022 [cit. 2022-25-4]. Dostupné z: https://www.unrealengine.com/en-US/unreal-engine-5.

[3] Godot. *Godot* [online]. Godot, 2022 [cit. 2022-25-4]. Dostupné z: https://godotengine.org/features.

[4] Gregory, J. *Game Engine Architecture*. 3. vyd. A K Peters/CRC Press, 2018. ISBN 1138035459.

[5] Grey, D. *Do háje s Berlínskou interpretací* [online]. Darren Grey, květen 2013 [cit. 2023-04-29]. Dostupné z: https://web.archive.org/web/20191104071743/http://www.gamesofgrey.com/blog/?p=403.

[6] Koster, R. *Theory of Fun for Game Design*. 2. vyd. O'Reilly Media, 2013. ISBN 1449363210.

[7] Monogame, V. *Monogame* [online]. Monogame, 2022 [cit. 2022-25-4]. Dostupné z: https://www.monogame.net/.

[8] Monogame, V. *MonoGame* [online]. Monogame vývojáři, 2022 [cit. 2022-25-04]. Dostupné z: https://www.monogame.net/showcase/.

[9] Neznámý. *Godot (game engine)* [online]. Wikipedia, duben 2022 [cit. 2022-25-04]. Dostupné z: https://en.wikipedia.org/wiki/Godot_(game_engine).

[10] Neznámý. *List of CryEngine games* [online]. Wikipedia, duben 2022 [cit. 2022-25-04]. Dostupné z: https://en.wikipedia.org/wiki/List_of_CryEngine_games.

[11] Neznámý. *List of Unity games* [online]. Wikipedia, duben 2022 [cit. 2022-25-04]. Dostupné z: https://en.wikipedia.org/wiki/List_of_Unity_games.

[12] Neznámý. *List of Unreal Engine games* [online]. Wikipedia, duben 2022 [cit. 2022-25-04]. Dostupné z: https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games.

[13] rekordů, G. kniha. *První 3D plošinová hra* [online]. Guinnessova kniha rekordů, duben 2022 [cit. 2023-30-04]. Dostupné z: https://www.guinnessworldrecords.com/world-records/89373-first-3d-platform-videogame.

[14] REKORDŮ, G. kniha. *První plošinová hra s pravým 3D* [online]. Guinnessova kniha rekordů, duben 2022 [cit. 2023-30-04]. Dostupné z: https://www.guinnessworldrecords.com/world-records/first-platformer-in-true-3d.

[15] SCHELL, J. *The Art of Game Design: A Book of Lenses*. 3. vyd. A K Peters/CRC Press, 2019. ISBN 1138632058.

[16] TECHNOLOGIES, U. *Unity real time development platfrom* [online]. Unity Technologies, 2022 [cit. 2022-25-4]. Dostupné z: https://unity.com/.

[17] VALVE. *Rogue* [online]. Valve, říjen 2020 [cit. 2022-25-04]. Dostupné z: https://store.steampowered.com/app/1443430/Rogue/.

[18] YEHIELI, I., DOPIERALSKI, R. a LAIT, J. *Berlínská interpretace* [online]. Roguebasin, květen 2008 [cit. 2022-01-28]. Dostupné z: http://www.roguebasin.com/index.php/Berlin_Interpretation.