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 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Dvořáček Martin**
Program: Informační technologie
Název: **Návrh a implementace 2D hry v Unity**
Design and Implementation of a 2D Game in Unity
Kategorie: Počítačová grafika

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/.

Pro udělení zápočtu za první semestr 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, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Cílem této práce je vytvořit 2d rougelike platformovací hru se základním chováním nepřátel a skládáním úrovní z modulů. Hra 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 will be made in game engine Unity.

Klíčová slova

2D, Roguelike, Rougelite, Platformovací, Hra, Hack'n'slash, Dungeons, sprite

Keywords

2D, Roguelike, Rougelite, Platforming, Game, Hack'n'slash, Dungeons, sprite

Citace

DVOŘÁČEK, Martin. *Návrh a implementace 2D hry v Unity.* Brno, 2021. 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 X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Dvořáček
10. května 2022

Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Martinovi Čadíkovi Ph.D..

Obsah

Kapitola 1

Úvod

Cílem této práce je vytvořit hru v žánru roguelite kde bude kladen velký důraz na modularitu a znovu hratelnost. O tohle téma jsem si zažádal, jelikož se velice zajímám o tvorbu a hraní her, dokonce občas když vidím či hraji hru přemýšlím jak něco takového vytvořit. Mimo jiné již delší dobu si přeji podobný produkt vytvořit.

V 2.kapitole blíže seznamuji a definuji žánry podle, kterých bude hra formována, 3.kapitole probírám historii video her,4.kapitole definuji herní engine, popisuji rozdíly mezi různými enginy co jsou zdarma a píši proč jsem si vybral Unity, 5.kapitole definuji, jaké funkce bude hra mít a hrubý popis jak fungují, 6. kapitole popisuji praktickou implementaci.

Kapitola 2

Seznámení s žánry

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

2.1 Plošinovky

Plošinovky jsou hry jsou založeny využití platforem pro dosažení jinak nedosažitelných cílů. Základem je koordinace pohybu hráče, aby byl hráč úspěšný musí koordinovat pohyb se skoky.

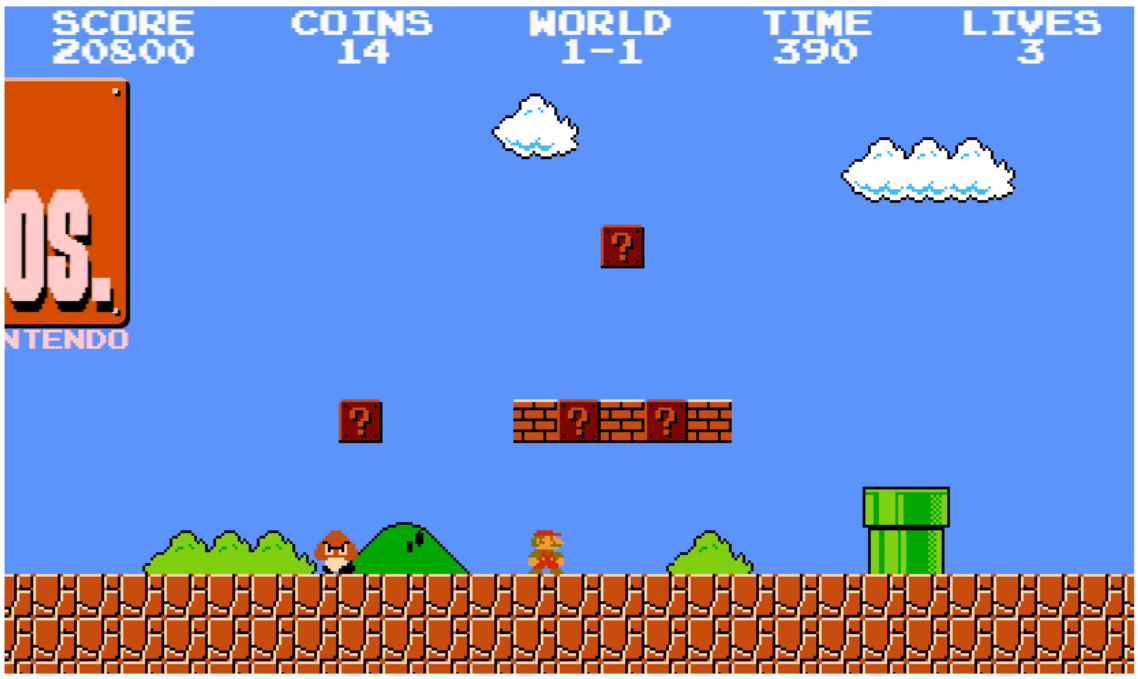
2.1.1 Příklady Plošinových her

Mario

Mario je původně 2D hra z roku 1985. Cílem je zachránit princeznu za instalatéra Maria, princezna se má nacházet vždy na konci úrovně. Úrovně jsou v základu na rovné a terén se mění pomocí plošin a trubek, v cestě Mariovi překáží nepřátelé, kteří jsou složitější čím dále ve hře se hráč nachází. Hratelnost je poměrně jednoduchá v základu, ale složitější, když se člověk snaží dosáhnout vyšších rychlostí a plynulosti. Hráč může běhat ze strany na stranu a když běží déle tak zrychlí, skákat a tím může zničit či odemknout posílení nebo porážet nepřátelé. Prohra nastane v případě, že hráč ztratí všechny své životy a ty ztrácí narážením do nepřátel z boku a zespodu bez posílení nebo pádem mimo obrazovku/mapu.

Na obrázku 2.1 lze vidět začátek první úrovně, což alespoň pro mě jeden z nejznámějších pohledů ze hry. Na vršku lze vidět informace jako výpis skóre úplně na levé straně, další je zde počet sesbíraných mincí. Mince mohou dosáhnout sta, poté 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, čas 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 a nulu hra končí. Mimo pás s informacemi je vidět, že se úrovně skládají ze základní plošiny, která v sobě může mít mezery. Cihlových bloků, které lze rozbít, když má hráč aktivní posílení, dále z bloků kde lze vidět otazníky z těchto vypadávají posílení nebo peníze. 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 bloku. Poslední je zde vidět samotný Mario bez posílení.

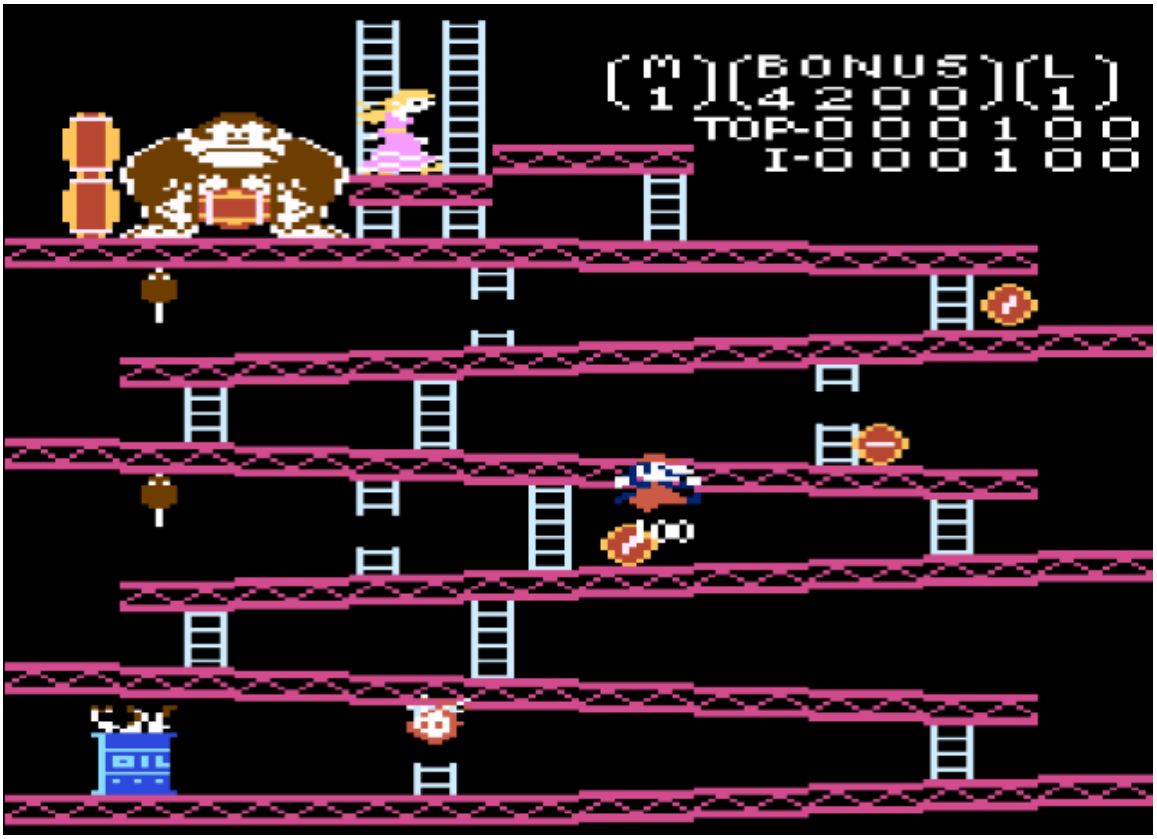


Obrázek 2.1: Výstřižek scény ze hry Super Mario Bros. Nejedná se o snímek z originální konzole, ale z webu https:supermarioplay.com.

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 arkádovou hru tak měla skóre, které se zvyšovalo při ničení či přeskakování překážek, jenž když se srazí z hráčem tak ztratí život a musí začít úroveň od začátku. V základu má hráč 3 životy. Hratelnost byla primitivní chůze do stran, skákání, lezení po žebříku.

Na obrázku 2.2 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, tyto barely jedou po plošinách, ale mohou spadnout i po žebříku. Barely je třeba buď přeskočit nebo zničit palicí.



Obrázek 2.2: 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.

Sonic the Hedgehog

Sonic je opět původně 2d hra z roku 1991. V této hře hrajete za Sonica super rychlého ježka. Rozdělení hry je 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 zamyšleno 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 šíleným zařízení. Hratelnost je velice jednoduchá pohyb doleva či doprava s tím že jako v máriovi hráč zrychluje čím déle ten směr drží a skákání umožnující porazit Robotnikovi stroje.

Na obrázku 2.3 lze vidět jeden z oblouků, v první úrovni, kterým aktuálně probíhá hráčuv charakter Sonic. Na levé horní straně se nachází skóre, čas, v této hře se počítá čas, jak dlouho trvá hráči proběhnout úroveň, hráč se snaží projít úroveň co nejrychleji. Třetí položkou jsou prsteny, prsteny nahrazují mince z jiných her, dokud má hráč alespoň jednu minci nezemře, ale vyhodí je. V pravo dole je vidět životy.

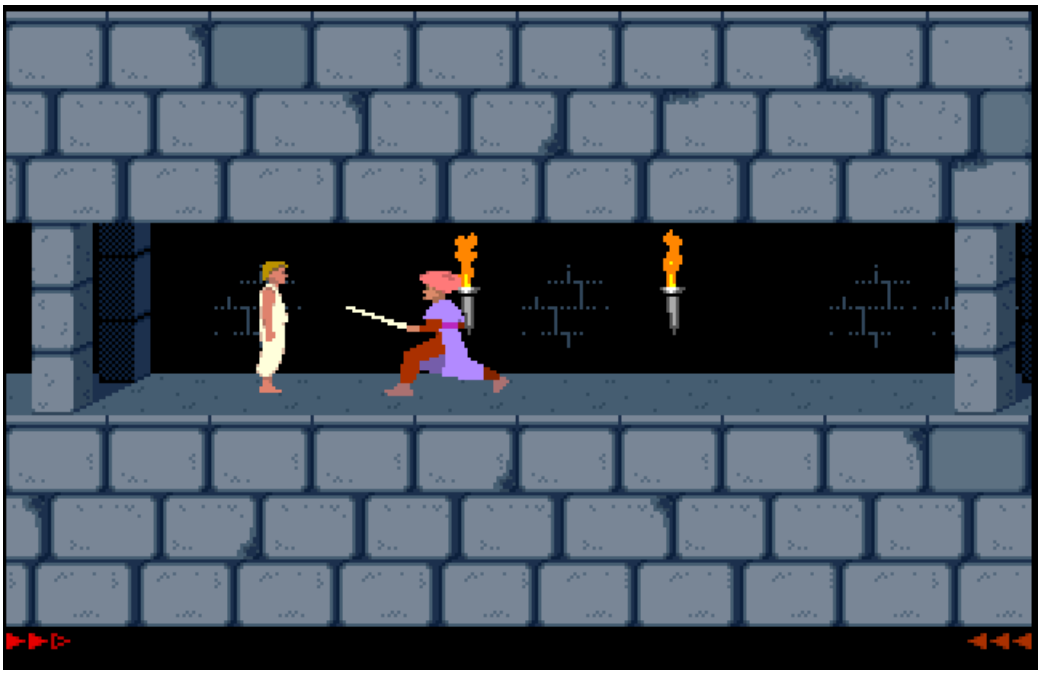
Obrázek 2.3: Výstřižek scény ze hry Sonic the Hedgehog. Nejedná se o snímek z originální konzole, ale z, Sonic Mania na platformě Steam.

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 občasné poražení nepřátel. Také na rozdíl od předem zmíněných titulů má hra nějaký soubojový systém. Asi největší rozdíl je ale neomezenost životů před vy 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 2.4 je obrázek ze začátku hry, kde je vidět, jak může jedena část úrovně vypadat. Uživatel ovládá blonďatého panáčka, ten má tři životy, než musí začít úroveň od znovu, lze je vidět v levém spodním rohu reprezentované červenými šipkami. Dále je zde vidět časový limit šedesátí minut. Limit není na úroveň ale na hru. Obrázek 2.5 mimo hráčova charakteru i jednoho z nepřátel, tohoto nepřítele lze porazit v souboji, pokud hráč najde meč, nepřítelovi životy jsou naproti hráčovým v pravém dolním rohu.

Obrázek 2.4: 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 začátek hry.



Obrázek 2.5: Výstřižek scény ze hry Prince of Persia. Nejedná se o snímek z originální platformy, ale emulátoru na webu https:www.retrogames.cz. Jedná se část o úrovně, kde hráč může bojovat.

2.2 Roguelike/Roguelite

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

- Náhodně generovaný svět – Svět pokaždé vypadá jinak, vždy když se zapne hra vygeneruje se nový svět, který uživatel může objevovat.
- Permanentní smrt – Když hráč zemře je musí začít úplně od znova. Neočekává se, že hráč vyhraje s jedním charakterem. Hru je možné uložit ale pokud ji hráč načte je soubor 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 limitované 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 hodně jako hráč – Pravidla, která se aplikují na hráče se aplikují i na nepřátele
- Taktická výzva – Hráč se musí postupně naučit taktiky jak porážet výzvy. Díky tomuto hráč pravděpodobně neporazí hru s prvním charakterem
- ASCII zobrazení – Tradiční zobrazování je representace mřížky ASCII znaky.
- Dungeons – Úrovně jsou složeny z chodeb a místností.
- 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.[7]

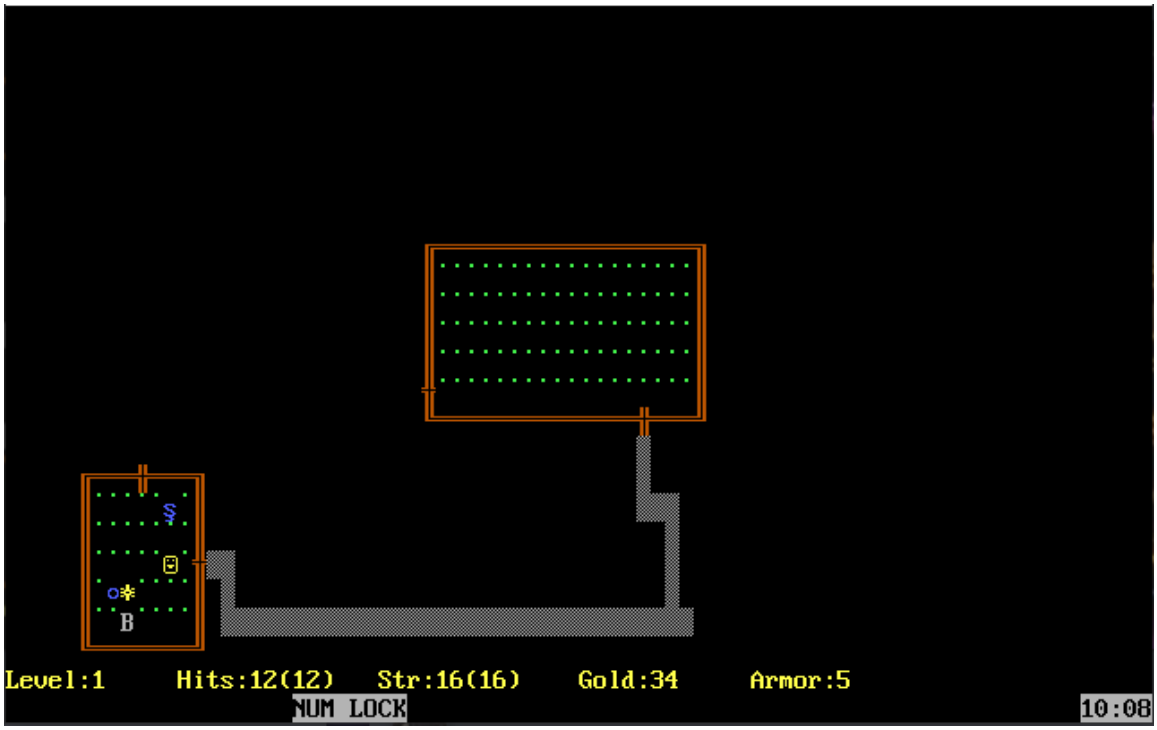
Roguelite či neo roguelike je téměř stejný jako roguelike ale je zde poměrně malý rozdíl a to, že si charakter něco po smrti charakteru uchová. Zkušenosti, peníze, cokoliv, a to následně může využít na vylepšení čí odemykaní nových věcí. Tudíž hráč vždy začíná s novou postavou, ale už může být vylepšená,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

2.2.1 Příklady Roguelike/Roguelite her

Rogue

Hra z roku 1980 podle které vznikla definice roguelike. Hra byla poměrně inovativní přes generované úrovně, které byli vždy jiné a také tím, že když hráč zemřel musel začít úplně od znova. Hra se nachází na mřížce reprezentované ASCII znaky a obsahuje jistou míru náhodnosti například při útočení. Cílem je dostat se přes 26 úrovní získat amulet a utéct. Mimo jiný vzhled úrovní, který znamenal, že se nedalo naučit na zpaměti rozložení, ale dokonce i jiné efekty kouzel.[14]

Na obrázku 2.6 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 označeny, zelenými tečkami. Hráč je označen 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.



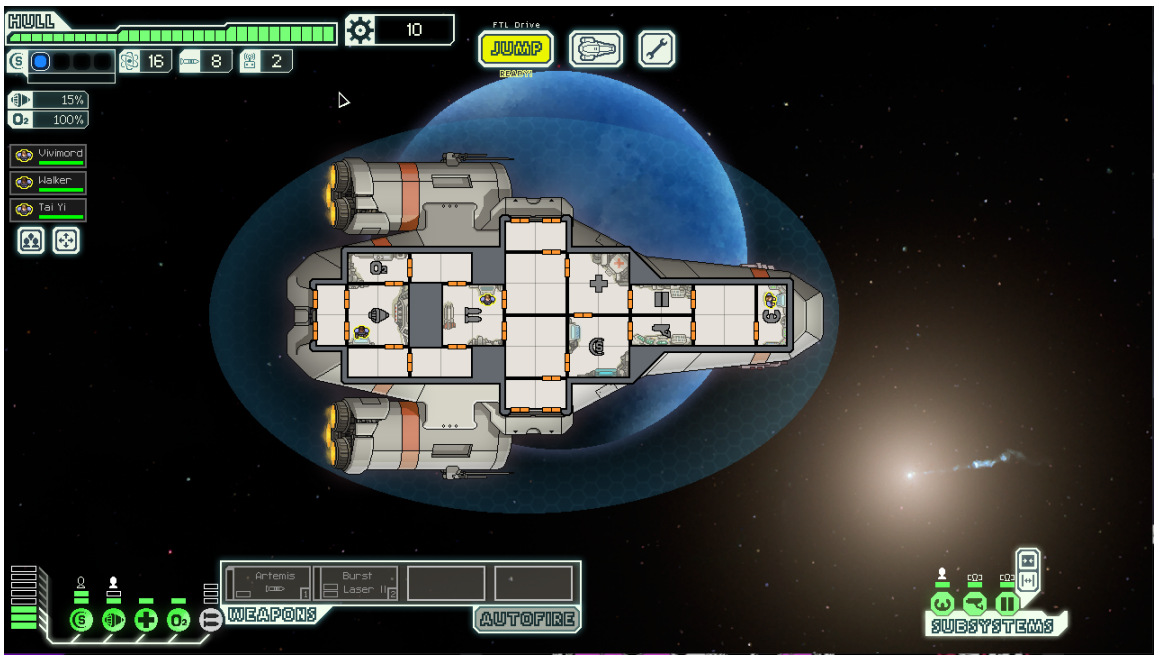
Obrázek 2.6: Výstřižek scény ze hry Rogue. Nejedná se o snímek z originální platformy, ale hru je možné zakoupit na platformě Steam.

FTL: Faster Than Light

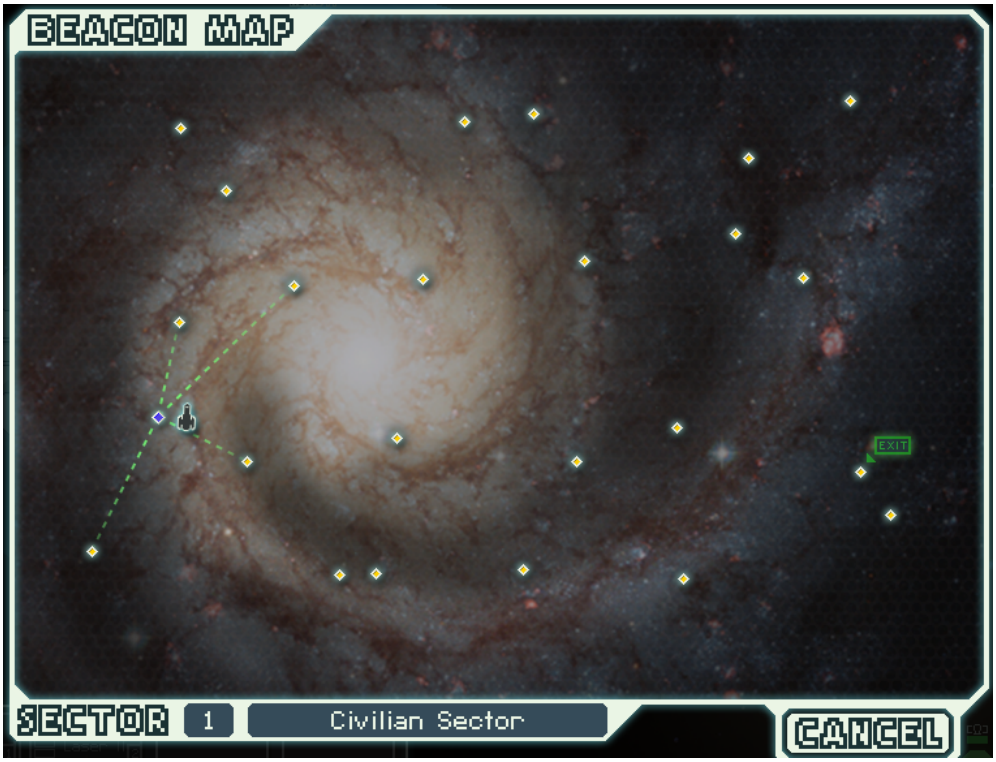
FTL je hra z roku 2012 kde je cílem utéct ve vesmírné lodi několik sektorů a poté porazit nepřátelskou loď která má několik fází souboje. V této hře je poměrně zajímavé, že hráč se musí strategicky rozhodovat podle více faktorů životů, peněz a paliva, takže si musí plánovat cestu přes sektory interpretované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

obchodníky, nepřátele a náhodné události s možnostmi rozhodnutí.

Na prvním obrázku 2.7 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 2.8, mapa obsahuje další možné zastávky v úrovni, takové zastávky jsou označeny zelenou pruhovanou čárou a další zastávky aktuálně nedostupné jsou označený 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 dronů. 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 s 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 nejlevější části označuje počet volné, rozmístitelné energie. Po pravé straně volné energie ji lze rozmístit do jednotlivých funkcí lodi, aby fungovali. Další komponenta jsou zbraně celkem 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 2.7: Výstřižek scény ze hry Faster than Light. Jedná se o obrázek vzhledu herní plochy.



Obrázek 2.8: 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 je poměrně 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 párkrát tak zjistí, že tentokrát hra nekončí zde ale v dalších patrech. Mimo postupné odemykání konců, které vypráví příběh Isaaca tak 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 pro obtížnost. 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 a v novějším je jich více.

U obrázku z Isaaca 2.9, 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, srdíčky 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

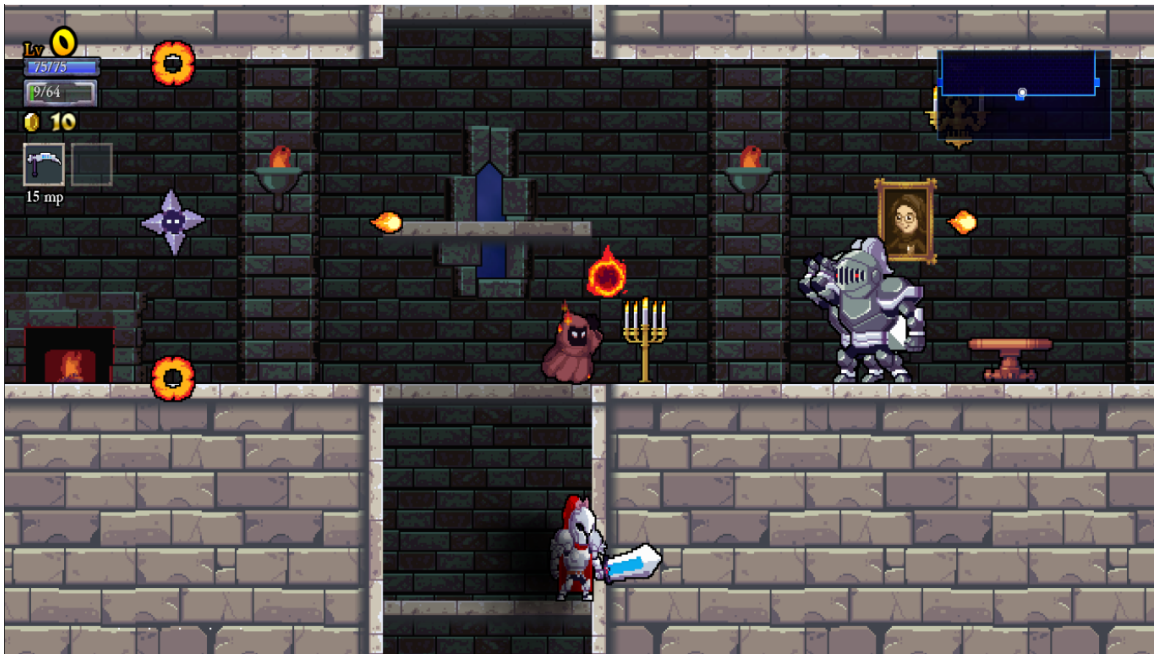
zavřené, jelikož jsou nepřátelé naživu. Poslední viditelná část je na pravé straně dole a jedná se o výpis jména úrovně.

Obrázek 2.9: 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 Isaaca se mezi nimi dí volně pohybovat. Jednotlivé sekce jsou vytvořeny z daného množství místností. Cílem je porazit V každé lokaci Bosse, aby se odemknul poslední, který na hráče čeká za dveřmi na začátku každého průběhu hradem. Za zabíjení nepřátel je obvykle hráč odměněn penězi. Peníze ale musí utratit před začátkem dalšího běhu jinak je ztratí u smrtky čekající před vstupem do hradu.

Na obrázku 2.10. na obrázku si je možné všimnout, že veškeré hráčovi informace i mini mapa je 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 2.10: Výstřižek scény ze hry Rogue Legacy.

Rogue Tower

Tato hra je poměrně nová z 28.Ledna 2022 a pro tento projekt, protože byla vytvořena v herním enginu unity. Jedná se tower defense(obrana cesty pomocí věží). Cílem je přežít množství vln, aniž by se dostali nepřátelé do hlavní věže. Za počet přežitých vln hráč dostane měnu, která slouží k nakupování neustále aktivních vylepšení anebo vylepšení aktivovatelných ve hře za splněné úrovně.

Na obrázku 2.11 má všechny elementy vycentrované. Na vrchní straně jsou životy, aktuální úroveň, skóre a peníze, poslední je zde mana a text vypisující ovládání. Dole se nachází tlačítka pro výběr věže, kterou chce hráč umístit, na tlačítku je i aktuální cena. Uprostřed je vidět hrací plocha, která se rozšiřuje s každou úrovní. Nepřátelé chodí jen po žluté cestě a na zelené ploše je možné stavět věže, některé jsou i viditelné.

Obrázek 2.11: Výstřižek scény ze hry Rogue Tower.

Kapitola 3

Stručná historie her

3.1 Historie plošinových her

3.1.1 Pohyb na jedné obrazovce

Mezi první hry se občas počítá Space Panic, ale jak bylo obvyklé v té době, tak se hra spíše soustředila na lezení spíše než skákání. Donkey Kong vydaný roku 1981 byl první hra umožňující přeskakovat překážky, tato hra pomohla upevnit Nintendo mezi důležitá jména video herního průmyslu. Následující rok vyšlo pokračování Donkey Konga jménem Donkey Kong Jr. a později až roku 1983 Mario Bros (první Mario hra), která dokonce umožňovala kooperativní hraní pro dva hráče. Tohle dalo základ vývoji kooperativních plošinových her jako Fairyland Story a Bubble Bobble. Na začátku roku 1982 se objevili první hry kde úroveň byla na více než jednu obrazovku, a ještě se neposouvala. Mezi tyto hry patřil Pitfall! od Davida Crana s 256 propojenými obrazovkami a stala se jednou z nejlépe prodávaných her na Atari 2600 a jednalo se o průlom pro žánr. Jako poslední hru bych v této podsekci zmínil Smurf: Rescue in Gargamel's Castle, který přidal nerovný terén do mixu a animované přechody mezi obrazovkami.

3.1.2 Pohyb s posouváním do boku

První hra s posunem byla plošinová střílečka Jump Bug, který vyšel jen 5 měsíců po Donkey Kongovi. Úrovně se zde posouvali horizontálně a v jednom případě vertikálně. Roku 1982 vyšla hra Moon Patrol kombinující střelbu a skákání přes překážky, dále také zavedla Paralaxní posun. Paralaxní posun je posun pozadí pro pocit hloubky ve 2D, pozadí nemusí být jeden obraz ale více pohybujících se různými rychlostmi. Měsíc po Moon Patrol vyšel Jungle Hunt, jenž také používal paralaxního posunu. Dále v tomto roce vyšla hra Track Attack! obsahující jednu úroveň s posunem obrazovky, v této úrovni hráč skákal po vagónech vlaku, postavička byla jednoduchá tyčková figurka, přesto inspirovala akrobacií hry jako Prince of Persia. B.C.'s Quest for Tires je hra vydaná roku 1983 a k již obvyklým funkcím jako posun do boku a skákání přes překážky se rozhodla ztížit skákání správným časováním skoků.

Roku 1984 vyšla hra jménem Pac-Land, hra měla obousměrný pohyb, horizontální posun obrazovky, chůzi i běh, skákání, odrazové můstky, posílení a sérií unikátních úrovní.

Roku 1985 vyšla hra, která se stala archetypem plošinových her, byla vydaná Nintendem pro Nintendo Entertainment System(NES). Samozřejmě je řeč o hře jménem Super Mario Bros, hra se prodávala se zařízeními a prodala přes čtyřicet miliónů kopií. Sega se snažila

tento úspěch napodobit s roku 1986 vydaným Alex Kidd in Miracle World. Tato hra měla jak horizontální, tak vertikální posun obrazovky, možnost praštit nepřátelé a překážky a obchody pro nákup posílení a vozidel. Další hra ze stejného roku byla Wonder Boy, tato hra se více inspirovala Pac-Landem než Mariem, hra měla skateboardovací segmenty, které dali pocit vyšší rychlosti. Poktačování Wonder Boy in Monster Land přidala elementy akční adventury a Role Playing Game(RPG - hra na hrdiny). Wonder Boy zase inspiroval další tituly jako Adventure Island, Dynastic Hero.

Jedna z prvních plošinových her s volným posunem na obou osách, následující hráčuv pohyb byla vektorová hra Major Havoc. Jednalo se o sadu miniher, kde pár bylo jednoduchých "plošinovek". Jedna z prvních rastrových plošinových her s podporou volného pohybu po obou osách je Legend of Kage z roku 1984. Roku 1985 vyšla od Enixu hra pojmenovaná Brain Breaker, jedná se o hru s otevřeným světem, míněno že hráč má volný pohyb s volností výběru cíle. Následující rok vydalo Nintendo úspěšnější hru s otevřeným světem Metroid, hra byla chválená svým balancováním volného prozkoumávání a vedením hráče po dané cestě.

Mega Man od Capcomu vydaný roku 1987 uvedl nový přístup k úrovním a to tím, že nebyli lineární, hráč si zde mohl vybrat v jakém pořadí splní jednotlivé úrovně. Tohle byl výrazný kontrast oproti oboum lineárním hrám jako Super Mario Bros a hře s otevřeným světem jako Metroid. Videoherní web GamesRadar připisuje možnosti výběru úrovní z Mega Mana, že se jedná o základ nelineárních misí, a vedleších úkolů ve hrách. Další hra od Capcomu v tom roce byla Bionic Commando, hra s pohybem po obou osách, která představila mechaniku uchopovacího háku (v angličtině Grappling hook). Tento hák se od té doby čast objevuje ve hrách například Tomb Raider.

Ke konci osmdesátých let se plošinové hry dostali na přenosné zařízení jako Game Boy mezi takové hry patřil například Super Mario Land, tento vývoj událostí pomohl žánru udržet si svoji popularitu.

V době, kdy vznikly systémy jako Genesis nebo TurboGrafx-16 byli plošinové hry nejpopulárnějším žánrem pro konzole a bylo celkem klasické mít nový exkluzivní titul obsahující maskota pro nové systémy. Tenhle zvyk potvrzuje například vydání Super Nintendo Entertainment Systému s novou hrou z Mariovi série, vysoce očekávaný Super Mario World, zatímco Sega vydala hru jménem Sonic the Hedgehog pro Genesis. Sonic se pochlubil novým designem, který umožnila nová generace hardwaru, mezi takové vymoženosti patřily velké úrovně, kde se obrazovka posouvala jak horizontálně tak vertikálně, zaoblené kopce, smyčky a fyzika umožňující dostávat se přes úrovně rychleji za správně načasované skoky a kotouly. Sonic se stal modelem pro nové maskoty.

Druhá generace her s posunem na počítači nastala zároveň s novými konzolami, hry jako Shadow of the Beast a Turrican ukázali že počítačové plošinové hry mohou konkurovat konzolovým hrám. Například Prince of Persia vydaný ke konci roku 1989 měl vysokou kvalitu animací.

3.1.3 Úpadek 2D

Hry byli vyvíjeny na šestnácti bitové konzole poměrně pozdě do generace, s úspěšnými tituly jako Vectorman, Donkey Kong Country 2:Diddy's Kong Quest a Super mario World 2: Yoshi's Island ale vyvinutí nového hardwaru hráče odrazovalo od 2D žánrů, i přesto Sega Saturn, PlayStation a Nintendo 64 obdrželi řadu úspěšných 2D her. Například Rayman byl velice úspěšný, Mega Man 8 a Mega man X4 uživili zájem o Mega Mana jako postavu, Castlevania: Symphony of the Night oživila její sérii a dala základ pro pozdější tituly. Hry

jako Oddworld a Heart of Darkness udrželi nový pod žánr stvořený z Prince of Persia naživu. Úpad 2D her byl ale zřetelný a vývojáři se snažili přizpůsobit a první taková varianta byla 2.5D.

3.1.4 2.5D a první 3D

Když se řekne 2.5D je myšlena 3D grafika a modely ale 2D hratelnost. Jedna z prvních takových plošinových her bylo Congo Bongo z roku 1983 od Segy. První plošinové hry simulující 3D perspektivu a pohyb kamery se objevili už brzké polovině osmdesátých let. Jeden z prvních příkladů je Antarctic Adventure od Konami, zde hráč ovládal tučňáka a obraz se posouval dopředu, pohled byl z třetí osoby a hráč se musel přeskakovat jámy a překážky. Roku 1986 vyšlo pokračování pro Antarctic Adventure, hra byla designovaná nyní velice významnou osobou pro průmysl jménem Hideo Kojima. Hra jako taková obsahovala více akčních a RPG prvků, dokonce měla více konců. Roku 1987 vyšla jedna z prvních stereoskopických 3D her a to 3-D World Runner.

První opravdový příklad 3D plošinové hry je Alpha Waves. Alpha waves bylo inovativní tím, že mělo velké množství 3D objektů(alespoň na dobu vydání), první rozdělení obrazu pro dva hráče naráz, a ořezávaní objektů podle hloubky ostrosti. Bug! nabízel mnohem konzervativnější přístup k opravdovému 3D, hráč se směl pohybovat jen po jedné ose(nemohl chodit na diagonále mezi osami) a charaktery byli před renderované sprity. První pokus převést populární 2D sérii do 3D byl Fade to Black, jenž byl pokračování pro Flashback. Bohužel hra nesplňovala kritéria a byla zařazená do akčních adventur.

Sony adoptovalo existující projekt od Naughty Dogs jménem Crash Bandicoot a Crash zůstal neoficiálním maskotem Sony po následujících několik let, než přešli na více platformní vydávání. Sega chtěla převést Sonica do 3D z titulem Sonic Xtreme ale kvůli problémům hra nikdy nezpatřila světlo světa.

3.1.5 Pravé 3D

Na začátku devadesátých let se začali hry měnit z pseudo 3D na pravé 3D. Rozdíly byli na omezení postav a kamery. S novým systémem tohle bylo umožněno ale předastivlo to i nové problémy, jako například vynucení upevněných kamer, aby náhodně neořezávali skrz úrovně.

První plošinová hra, která byla pravé 3D a umožnila volný průzkum prostředí byla Geograph Seal. Zde hráč ovládal mecha ve tvaru žáby, mohl až třikrát vyskočit, než znovu dopadl na plošinu. Jako pomoc pro takto vysoké skoky se kamera nasměrovala dolů. pro porážení nepřátel se zde dalo střílet nebo dopadat na nepřátele. Hra není příliš známa, jelikož nikdy nebyla vydaná mimo Japonsko. Následující rok vydali Jumping Flash!, hra byla velice podobna Geograph Seal ale byla vydaná i mimo Japonsko. Dokonce byla i dost úspěšná pro dvě pokračování. Podle Roba Faheye se jednalo o jednu z nejdůležitějších předků 3D plošinových her. Podle Guinnesovi knihy rekordů se jedná o první pravou 3D plošinovou hru. Další brzký příklad pravé 3D plošinové hry je Floating Runner pro PlayStation.

Roku 1996 Nintendo vydalo hru se jménem Super Mario 64. Hratelnost umožnila prozkoumat otevřené 3D prostředí s větší svobodou než kdykoliv předtím. Této svobodě napomohlo přidání analogového joysticku na svůj Nintendo 64 ovladač, což nebylo viděno od vydání konzole Vectrex se nyní stalo standardem na nových ovladačích. Přidání joysticku umožnilo ovládání s větší přesností. Super Mario 64 přivedl změnu, na rozdíl od 2D titulů kde se stačilo dostat do bodu B, zde obsahovali úrovně struktury misí, které odměňovali věcmi pro odemykání nových míst. A tento trend opět následovalo spousty her jako Banjo-

Kazooie, Spyro rhe Dragon a Donkey Kong 64. Sega vytvořila 3D Sonic hru jménem Sonic Adventure, kde podobně jako v Super Mario 64 použili podobnou strukturu Centra (anglicky Hub) ale jednalo se o hodně víc lineární hru. Samo sebou se zaměřením na takto volný pohyb se muselo zlepšit i ovládání kamery.

I přes všechny RPG tituly, střílečky a komplexní akční adventury se Tomb Raider jedna z mála 3D her uchovávající si 2D design úrovní stala jednou z nejprodávanějších her pro PlayStation.

Několik vývojářů, jenž našlo úspěch ve vývoji 3D her se pokusily zaměřit na dospělejší publikum i přes vzhled těchto her. Mezi takové hry patří Conker's Bad Fur Day, Gex: Deep Cover Gecko, Legacy of Kain: Soul Reaver a Messiah.

[12]

3.2 Historie roguelike/roguelite her

Samotná definice roguelike Berlínské interpretace je až roku 2008 ale historie začíná již v 70. až 80. letech 20. století. Roguelike pojmenovaný podle jedné z prvních her s prvky žánru Rogue. Tato dungeon-crawler hra vyšla roku 1980 a vytvořili ji programátoři Michael Toy a Glenn Wichman. Premisa hry byla velice jednoduchá stejně jako její grafika, vše bylo reprezentováno ASCII znaky. Hráč byl označen jako @ a objevoval dungeon kde hledal Amulet Yendoru a po cestě nacházel monstra, předměty a zlaťáky. Hra po každé smrti začala úplně od znovu s čerstvě vygenerovanou úrovní, takže se lidé nemohli naučit jen trasu, ale museli se naučit jak hrát hru a přizpůsobovat se situacím. Hra je dokonce open-source což umožnilo dalším upravovat a přetvářet do jiných her.

První "roguelike"hry byly Hack a Moria, které byly vytvořené od počátku bez využití zdrojového kódu Rogue. Hack měl vylepšenou umělou inteligenci, povolání charakteru, obchody, a schopnost jíst mrtvé nepřátele pro získaní dovedností. Moria, jenž byla inspirovaná Pánem prstenů skrz hlavního a finálního oponenta Balroga. Hra též změnila tvar, velikost a hloubku dungeonu, dokonce i přidala možnost těžit rudu mimo jiné. Hry na podobné téma vycházeli i dál například Omega a ADOM(Ancient Domains of Mystery), každý další spin-off přidal nějakou mechaniku nebo něco jiného.

Dále se roguelike hry inspirovali i společnosti které začali vytvářet něco čemu by se začalo říkat roguelite. Roguelite je prakticky pod-žánr roguelike a roguelike je zase pod-žánr her na hrdiny též anglicky označovaných jako RPG(role playing game). Mezi tyto hry patří například velice úspěšný titul od společnosti Blizzard Entertainment jménem diablo, kde byl žánr velice zjednodušen. přes zlepšení bojem v reálném čase, a ne tazích a přehledným rozhraním.

Dále jsou zde hry typu soulslike, které jsou pojmenované podle dark souls. Tyto hry nemají pernamentní smrt ale ztrátu zkušeností, dark souls se ale jako roguelike ani roguelite označit nedají.[3]

Mezi novější roguelike hry patří například Hades, Faster than light(FTL) a mezi novější roguelite hry, Rogue legacy, The Binding of Isaac a Warhammer: Vermintide 2.

Kapitola 4

Herní engine

Jedná se o pomocnou sadu 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 komponent hry.

4.1 Unity

Unity je engine, který pokud osoba či firma nepřesáhne příjem sto tisíc dolarů v posledních dvanácti měsících je kompletně zdarma a pokud přesáhne musí se nakoupit licencované verze. Aktuální stabilní verze je Unity 2021.3.1f1.[13]

Unity je 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 přes eventy/události, dokonce i podpora 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 jeho 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. [10]

4.2 Unreal engine

Unreal engine je opět zdarma dokud osoba či společnost z prodejů nedostane jeden milion dolarů poté si společnost Epic Games účtuje 5% zisků. 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 zrychlené

a zjednodušené animovaní animací v kontextu, 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 vyvynutých 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. [11]

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 jak Unreal engine 4. A to 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]

Cry engine je známý tituly: Ryse: Son of Rome, Far Cry, Crysis a známým českým titulem od Warhorse studia Kingdom Come. Deliverance [9]

4.4 Godot

Godot je 3D kompletně zdarma, open-source engine, který podporuje velké množství jazyků pro skripty. Mezi hlavní funkce považují uzly usnadňující design hry, flexibilní systém scén a možnost vytvářet vlastní nástroje.[4]

Významné hry pro Godot jsou: Sonic Colors: Ultimate, Commander Keen in Keen Dreams, Deponia[8]

4.5 Monogame

Monogame je podobně jako Godot zdarma a open-source, ale narozdíl od Godotu se jedná o framework, sadu nástrojů do c#.[5]

Pomocí Monogame vzniklo několik novějších známých 2D titulů jako: Celeste, Jump King, Stardew Valley [6]

Kapitola 5

Princip hry a návrh

Produktem této práce je vytvoření 2D roguelite hry. V této hře se nepočítá, že člověk uspěje na poprvé a, že bude muset mnohokrát začít od znovu s možným vylepšením statistik charakteru. Takto bude mít hráč pocit, nějakého pokročení dál 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 v základu jsou tyto hry poměrně jednoduché. Takže pro detailnější návrh.

Pro hráče by měl být pohyb omezenější, neměl by být schopný například přeskočit celou místnost nebo skákat již ve vzduchu, pokud nechceme zavést více skoků. Dále by měl hráč mít možnost vyhnout se nepříteli nějakým stylem úskoku během kterého ho nebude možné zasáhnout jakým jakýmkoliv útokem. Pro útok z blízka je asi jedna z nejlepších možností útok do čtyř směrů, doleva, doprava, nahoru a dolů, je možné i do osmi směrů, do diagonál a pro útok z dálky do 8 směrů pro lepší využitelnost. Hráč bude mít nějaké životy a pro tento styl hry bude pohodlnější využít číselného představení v posuvníku místo jednotlivých srdíček, jelikož když se využívají srdíčka tak obvykle není rozdíl mezi zásahy a když je je to složitější a méně přehledné než u číselné reprezentace, jeden zásah jedno srdce (například v The Binding of Isaac jsou životy reprezentovány srdíčky a jediný rozdíl mezi zásahy jsou půl srdce nebo od elitních nepřátel celé). Když hráčovi životy klesnou na nulu zemře a vrátí se do lobby (více o lobby v 6.1.2). Jako druhý využitelný zdroj bude Mana, zdroj obvykle označující schopnost využít nadpřirozené síly(Magie). Mana bude využívaná některými pomůckami reprezentující magii. Celkově bude moci hráč mít 3 pomůcky a ty nemagické nebudou omezeny manou ale množstvím použití.

Nepřátelé by měli fungovat 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 bude mít více typů unikátních útoků.

Kapitola 6

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 tři scény, a to hlavní menu, lobby a Herní scéna pro mě v projektu pojmenována Level.

Pro práci v Unity se bude vytvářet hodně "prefabů"objektů předem vytvořených a uložených do dat hry, aby se dali jen vytvořit kopie. 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 z stáhnuté z Unity Asset storu.

6.1 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, 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

Objekty se dají přidávat buď přetažením prefabu nebo kliknutím tlačítka přidat v záležce hirearchie objektů. Každý objekt ve scéně lze pojmenovat a je výhodné pojmenovávat objekty podle jejich funkčnosti, zmiňuji to tu zde protože 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, aniž by se prolínali. Takto je to udělané pro usnadnění práce. Hlavní menu je úvodní obrazovka, zařizuje veškerá nastavení a výběr jednoho ze tří možných uložených souborů. Lobby slouží jako přestupní bod mezi menu a hrou, zde hráč využívá roguelite mechanik. Level je scéna kde se odehrává hra jako taková.

6.1.1 Hlavní menu

Jak je již zmíněno hlavní menu je úvodní scéna, kterou hráč vidí jako první věc po zapnutí hry a 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 ještě nějaké 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 panel nastavený pro tento projekt s kotvou na pravý roh kde si přeji mít menu jako takové. Panel je už objekt obsahující 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
- 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.

Obrázek 6.1: Výstřižek vzhledu hlavního menu

6.1.2 Lobby

Lobby je scéna, kde přichází na řadu implementace roguelite, funguje jako mezi patro mezi jednotlivými hrami. Jako v každé scéně je zde kamera aby bylo cokoliv vidět, ale tato kamera má na sobě skript, který vyhledá objekt hráče ve funkci Start, vyhledání je řešeno přes funkci GameObject.Find("jmeno") (tato funkce prochází všechny objekty ve scéně proto ji není dobré používat více nebo ve funkcích jako Update) nebo pokud má objekt přiřazené označení/tag tak GameObject.FindGameObjectWithTag("tag") (tato funkce je lepší, jelikož prochází jen objekty s přiřazeným označením) 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.

Po kameře je zde Canvas tentokrát obsahující všechny prvky uživatelského rozhraní pro hru i pro lobby 6.4. Tyto prvky uživatelského rozhraní 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 vylepšení koupit, podle těchto funkcí se zamknou tlačítka pro nákup.

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 sr zapnutým spouštěčem ve tvaru co nejlépe sedí tvaru portálu, zde je to ovál a skript. Tento skript ovládá kolize přes OnTriggerEnter2D kde 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 a

výstupu do kolizní zóny plus snímání stisknutí námi 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 6.7, LevelManager je prázdný herní objekt kam se generují v lobby hráčovi projektily, a EventSystem.

Obrázek 6.2: Výstřižek vzhledu Lobby při pohledu ze hry

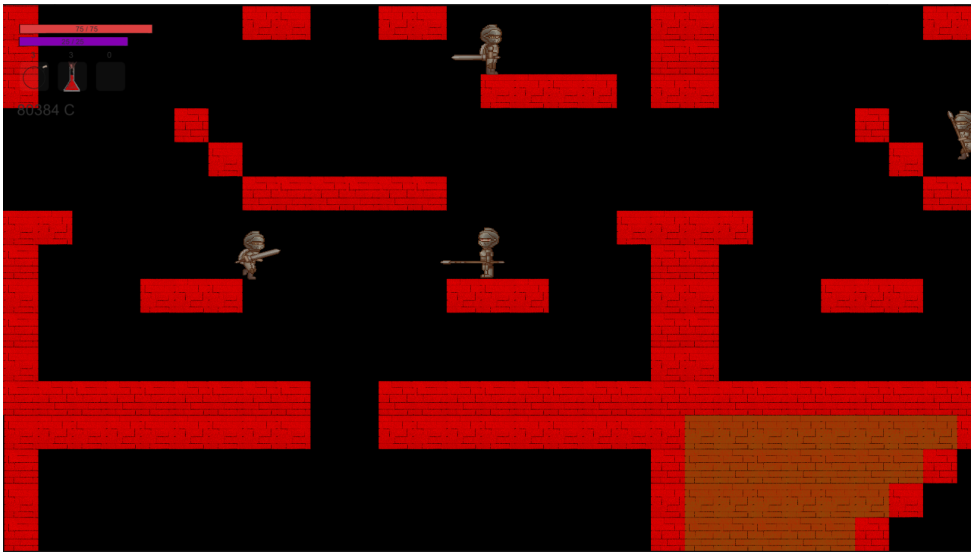
6.1.3 Herní scéna

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.

Kamera zde funguje stejně jako v Lobby, vlastně se dá říci, že je identická.

V této scéně je opět LevelManager, ale zde obsahuje skript, generující úroveň a zároveň se jedná o rodičovský herní objekt 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 6.9.

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 specificky pojmenované objekty.



Obrázek 6.3: Ukázka z herní scény

6.2 Ukládání dat

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

Pro ukládání do JSON je zde vytvořený 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, určuje se, co má přepínač udělat podle řetězcového klíče z aktuálního GameData objektu.

6.2.1 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 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 ty se potom zapíší na disk pomocí FileManageru.
- LoadSaveFile() - Pomocí FileManageru se načte soubor do listu dat a ten se načte do lokálních proměnných.

Funkce SaveFile() se volá automaticky v případu smrti hráče v úrovni, to znamená že se nepodporuje ukládání během hry jen automaticky na ukončení běhu. Neukládá se to ani na začátku běhu, aby si hráč mohl rozmyslet vybrané vylepšení.

6.3 Animace

Animování je dlouhý a složitý proces, bohužel nejsem zrovna umělec, takže v tomto projektu je využita sada z Asset Storu, oficiální web pro prodej, nákup a stahování assetů. I když je postavička zakoupená, neznamená, že se člověk vyhne animování. Asset jménem Customizable Pixel Character od uživatele Cainos obsahuje animovaný lidsky vypadající pixelový charakter. Ale neobsahuje alespoň 3 potřebné animace.

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 na kliknutí 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.

6.3.1 Prohazováním obrazů

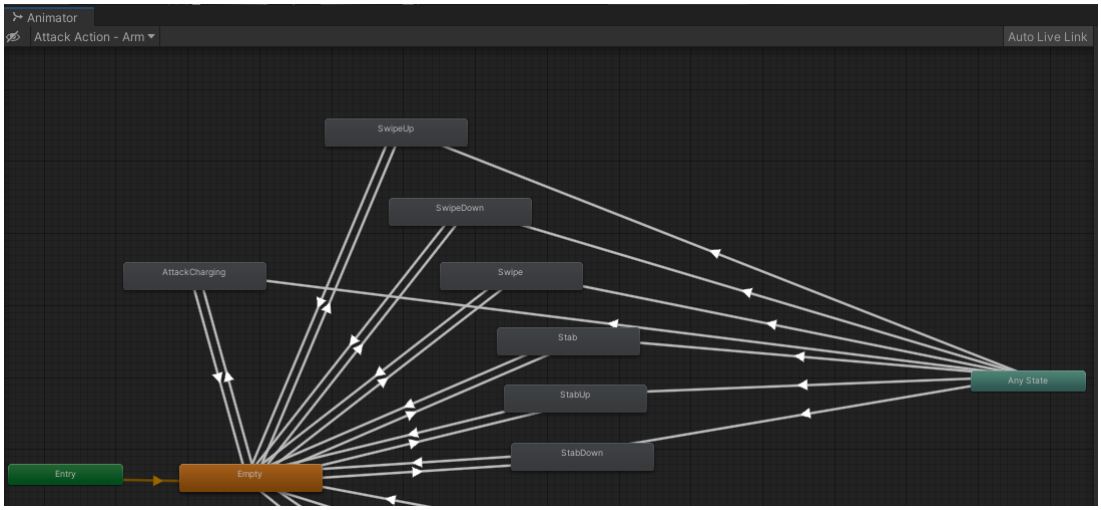
Pro tento styl animace je potřeba více spritů nebo jeden obsahující více, jelikož Unity obsahuje sprite editor a jedna z funkcí sprite editoru je rozpoznání jednotlivých spritů v souboru obsahujícím více spritů. Po rozdělení na jednotlivé sprity se přetáhnou do okna animace a nastaví se, jak rychle se 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 pro například efekty.

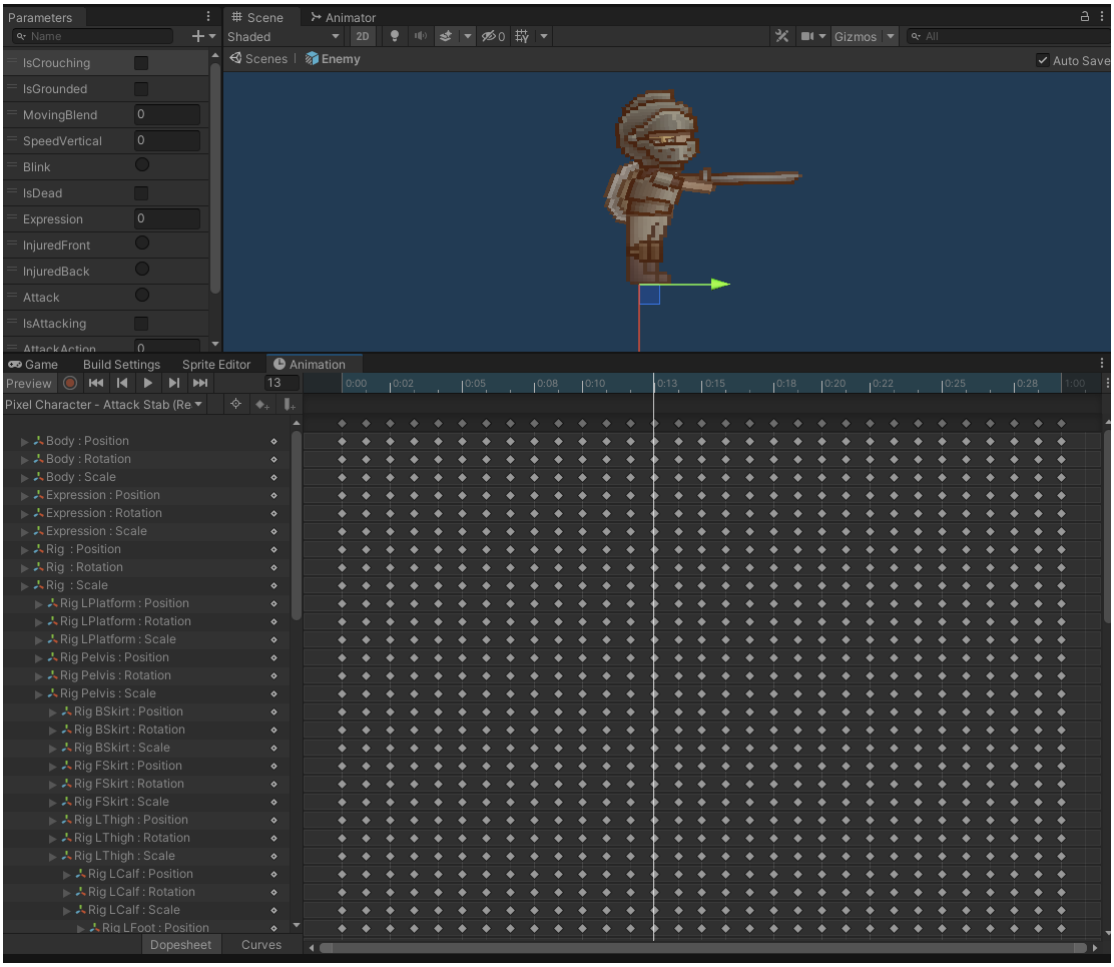
6.3.2 Nahráváním pohybu kostí

Tento styl vyžaduje sprite ideálně vytvořený na několika vrstvách, ve 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 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 lepší pro charaktery v tomto projektu.



Obrázek 6.4: Výstřižek okna animátoru s ukázkou, jak můžou stavy vypadat



Obrázek 6.5: 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)

6.4 Prvky uživatelského rozhraní

Uživatelské rozhraní je velice důležitá komponenta 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, usnadňuje to následující rozmisťování a udržování pohromadě. Většina segmentů se vyplatí udržovat u okraje, kde u 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 do scény a operace s rozhraním je zde vytvořený a pžipojený k objektu Canvas, skript UIManager.

6.4.1 Životy a Mana

Životy i Mana jsou udělané z posuvníků. Posuvníky jsou nastavené, aby mohly obsahovat jen celá čísla a v základu nastavená 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 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.

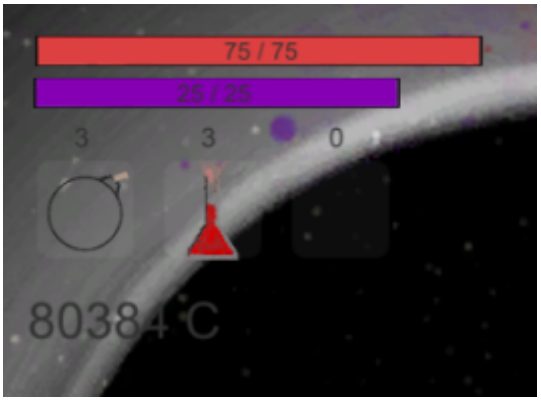
Oba dva posuvníky jsou ovládané ze skriptu hráče, pro ovládání je potřeba ve skriptu referenci na objekt, tu je zapotřebí najít nebo mít vloženou. Pro uvedení příkladu řekněme že máme přes projekt vloženou referenci nebo nalezenou přes GameObject.Find("jméno");, tato reference se bude jmenovat HpBar. 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;

6.4.2 Použitelné pomůcky a peníze

Vyobrazení pomůcek je udělané ze sady dvou obrázků a jednoho textu. První obrázek je nastavený 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 komponentu 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, se jménem například Usable. Příkaz pro vložení obrázku je Usable.sprite = Resources.Load<Sprite>("Cesta ke spritu ve složce Resources neuvádí se zde přípona souboru"); a pro nastavení viditelnosti Usable.color = Color.White;, 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 nastaví se podle uloženého souboru při vytvoření scény. Změna proběhne příkazem reference.text = "XXXX C"; kde XXXX představuje aktuální množství peněz a C je jen označení měny.



Obrázek 6.6: Výstřižek vzhledu uživatelských informací

6.4.3 Menu během hry

Jedná se o malé menu, které je snad absolutní nezbytností, jelikož umožňuje hráči vrátit se do hlavního menu nebo ukončit hru, často se u podobného menu pozastaví hra, 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 složené 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();.

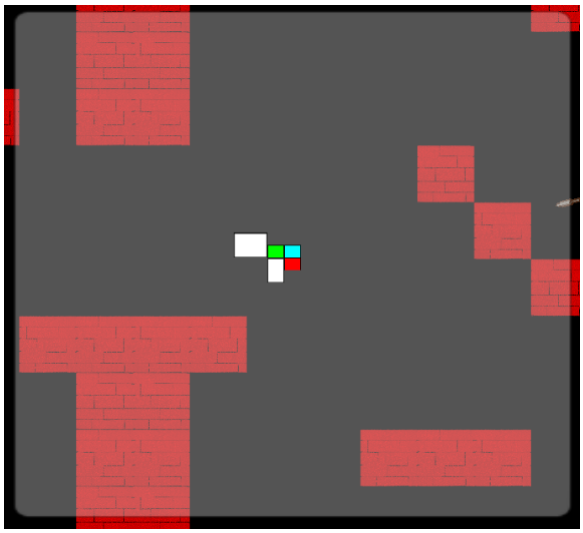
6.4.4 Mini mapa

Mini mapa je další častou a důležitou součástí. Mini mapa zobrazuje rozložení úrovně, ať už vzdáleným pohledem na danou úroveň nebo vygenerováním 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 nepřekrývala po zapnutí s dalšími prvky.

První možnost, vzdálené zobrazení úrovně by fungovalo 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 minimapa 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 extrémě 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 6.9. U boss části úrovně se negeneruje nová mapa v tomto případě, jelikož je zbytečná. Výhoda tohoto stylu je lehčí zobrazení, navštívené, aktuální a doposud nenavštívené místnosti. Pro rozpoznání místnosti je přes přístup k proměnné color v komponentu Image, příkaz je reference.color = Color.Cyan; obsažený při vstup 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, a červenou (Color.Red) pro začáteční místnost. Začáteční místnost barvu nikdy nemění. Pohyb mini mapy by zde byl složitější, probíhal by skrz posouvání o velikost objektu, aby byla aktuální místnost vždy ve středu.



Obrázek 6.7: Výstřižek ukazující vybraný styl mini mapy

6.4.5 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 vložením 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 dost peněz. Obchod je rozdělený na dvě části, obchod s pomůckami a obchod s vylepšeními. 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 vyobrazujíc 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 pro vylepšení 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 volající funkci BuyUtility(int utility), kde utility označuje nastavené očíslování ve výčtovém typu pomůcek, využije se zde vygenerování nové pomůcky blíže popsané v 6.7. touto funkcí vygenerovaná pomůcka se přiřadí do volného místa, pokud není volné místo jsou tlačítka vypnuta nebo alespoň jejich interaktivita. 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.

Objekt kováře obsahuje čtyři tlačítka volající funkci BuyUpgrade(int Upgrade), Upgrade je číslo pro přepínač vybírající které vlastnosti se vylepší. Při vylepšení vlastnosti se aktualizuje úložný soubor o velikost vylepšení, zvýší se v úložném souboru i množství zakoupených vylepšení a aktualizují se vlastnosti hráče přes nalezenou referenci na hráče a přístup na komponentu hráčova skriptu, dále se strhnou peníze a zkontroluje, zda hráč může dále nakupovat. Další omezení na nakupování vylepšení je maximální hodnota specifického vylepšení. Například základní životy nemohou být zvýšeny přes sedmdesát pět a každé vylepšení je zvýší o pět, když hráč dosáhne daných sedmdesát pět životů tlačítko se opět zamkne i přes dostatečné množství peněz.

6.4.6 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. Bonus si hráč vždy může vybrat jen jeden v dané řadě. Tlačítka jsou vypnuty, pokud není v uloženém souboru uvedeno, že porazil hráč 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 bonuse 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.

6.5 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, a 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 dosah, 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, typ zbraně 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.

6.5.1 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 6.8. Pro tuto kolizi je třeba nastavit všechny body:

- První bod má nastavené hodnoty pozičního vektoru na x rovno sedmi desetinám krát dosah krát strana útoku (strana útoku nabývá hodnoty plus nebo mínus jedna) a y rovno čtyřem desetinám plus čtyřem desetinám krát dosah
- Druhý bod má nastavené hodnoty pozičního vektoru na x rovno nule a y rovno čtyřiceti pěti setinám
- Třetí bod má nastavené hodnoty pozičního vektoru na x rovno nule a y rovno mínus polovině
- Čtvrtý bod má nastavené hodnoty pozičního vektoru na x rovno padesáti pěti setinám krát dosah krát strana útoku (strana útoku nabývá hodnoty plus nebo mínus jedna) a y rovno mínus polovině
- Pátý bod má nastavené hodnoty pozičního vektoru na x rovno sedmdesáti pěti setinám krát dosah krát strana útoku (strana útoku nabývá hodnoty plus nebo mínus jedna) a y rovno čtvrtině

Obrázek 6.8: Výstřižek ukazující možný vzhled polygonové kolize

6.5.2 Projektily

Projektil je uložený jako prefab, prefab se vytvoří na aktuální pozici hráče plus sedm desetin na ose y, kterému se hned po vytvoření musí 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 zbraň a proměnné pro směr útoku jménem AttackX pro x hodnotu vektoru a AttackY pro y hodnotu vektoru. Při vytvoření a po přiřazení proměnných se ve funkci Start najde hráčův skript přes 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, zda zbraň je luk nebo hůlka. pro luk se přidá sílá přes příkaz RigidBody2D.AddRelativeForce (Vektor, typ síly (zde impuls)), Vektor zde bude new Vector2(AttackX, AttackY).normalized (.normalized změní vektor na vektor s magnitudou jedna) a aby se hnul chtěnou rychlostí se modifikuje přes rychlost projektilu a dosah uložený ve zbrani, 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 nastavý vzhled projektilu na šíp přístupem na komponentu Sprite Renderer a již párkrát zmíněnou funkcí Resource.Load se načte sprite šípu. Pro hůlku je třeba jen nastavit barvu spritu podle typu prvního zranění na zbrani. Výběr barvy je napevno daný pro typ zranění. Výběr barvy je jednoduchý přepínač podle typu zranění. Jako poslední se nastaví nová pozice v komponentě Transform, vlastně se jedná o stejnou pozici na 2D ploše ale pozice z bude -5 aby byl projektil zobrazený 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". U kontroly, 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 zavolá se funkce ze skriptu hráče skript.TakeDmg(List<Dmg> zraněníVeZbrani), poté se zkontroluje zda zbraň nemá nastavené aby penetrovala více objektů nebo výbušnost, když je výbušná a není nastavená pro průraz více objektů se zavolá Destroy(this.gameObject);, což zničí projektil.

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 výbušnost, když je výbušná a není nastavená pro průraz více objektů se zavolá Destroy(this.gameObject);, což zničí projektil.

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 6.6.2.

6.6 Pomůcky

Pomůcky jsou objekty usnadňující průchod hrou, mohou sloužit pro doplňování využitelných zdrojů nebo jako další varianty zranění. Všechny pomůcky jsou nějakým způsobem omezeny ať již počtem použití nebo manou. Pomůcky se získávají za prvé nákupem v obchodu 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 to typ 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.

6.6.1 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)).

6.6.2 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.

6.6.3 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 a 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.

6.7 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 Animator a WeaponSlot, objektu pro kolize útočení, objektu pro kouzlení a objektu snímajícího kolize ze spodku pro skok na nepřátelích.

Hlavní objekt Player, obsahuje dvě kolizní komponenty, kapslovou bez spouštěče pro pohyb po terénu a krajovou se spouštěčem pro snímaní přistání ze skoku, dále obsahuje dva skripty, skript ovlivňující vzhled a animace a 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 skript snímající dané kolize s odpočtem mezi snímáním a nakonec SpellThrowerDummy pro vytvoření kolize plamenometné magie popsané v 6.6.3 s přiloženým skriptem snímající zranění způsobené magií.

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

Funkce Start si vytvoří skript pro ovládání modelu, který jsem si pojmenoval PlayerViewModel pvm, dále si najde skript úložného objektu a kolize na hlavním objektu. Dále se vybere pomocí funkce pro náhodný výběr čísla z rozsahu nula až šest 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, 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 vloží se správný obrázek zbraně.

Update je zde pro snímání vstupů uživatele a kontrolu podmínek. Vstup uživatele se snímá přes funkce, Input.GetAxis("Horizontal/Vertical"); pro pohyb a Input.GetKeyDown(Kód tlačítka) pro útok, výběr pomůcek, aktivaci úskoku a posun mezi úrovněmi. Dále se zde kontroluje, zda hráč umřel nebo vyhrál.

FixedUpdate je funkce, kde se probíhá většina funkčnosti. Většina věcí funkcí má jistý čas trvání nebo dobu mezi aktivacemi, odpočet funguje pomocí proměnná -= Time.deltaTime;.

Po odpočtech je kontrola, zda 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.

6.7.1 Model

Základ modelu je využívá jak nepřítel tak i hráč, 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.

6.7.2 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), 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 nastavý 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.

6.7.3 Útok a pomůcky

Útok probíhá buď útokem zbraní 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, rozdíl je 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 zautočí dolů aktivuje se objekt na hráči AttackJump, který pokud trefí nepřítele odrazí hráče nahoru. Útok zbraní na dálku je lehčí, vytvoří se projektil a nastavý se mu zbraň a směr útoku, na rozdíl od útočení zblízka je je dálkový útok do osmi směrů. Každá zbraň má po trefení 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.

6.7.4 Zranění

Hráč když obdrží zranění tak prvně 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 modifikována zbrojí. Po aktualizaci životů v modelu se aktualizují životy na obrazovce a nastaví se čas nesmrtelnosti.

6.8 Nepřátelé

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 nemohl se 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 vytváří si každý nepřítel vlastní paprsek(Raycast2D), pokud tento paprsek nezasáhne vrstvu bloků nepřítel se zastaví.

Nepřítelé jsou funkčně velice podobní, směr pohybu a útoku se určuje po aktivaci(vstup hráče do místnosti) podle aktuální pozicí 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 provedeny 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 aktivovaný, a jeho počáteční souřadnice potřebné pro restartování nezabitých nepřátel po opuštění místnosti.

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

6.8.1 Boss

Boss je 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.

6.9 Generování mapy

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 a zároveň nemohl vyskočit z mapy. 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 počáteční bod. 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 se tam daná místnost vejde. Následující generování probíhá, dokud nedosáhne hodnota místností úrovně nuly. Po konci generování Proběhne kontrola míst kolem každé místnosti a každé dveře co 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, velikostí místnosti, svým objektem a zda je 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.

6.9.1 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, načte si aktuálně vybranou past (Pro každou místnost se vybírá náhodně jiná), nastaví se časovače a velikost spoušťové kolize, a nakonec pokud nemá nastavený aktivační čas se aktivuje.

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

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

Kapitola 7

Závěr

Cílem práce bylo naučit se s enginem Unity, nastudovat si alternativy, zjistit si něco o historii her, a nakonec vytvořit 2D hru. Cílem bylo vytvořit hru kde hráč musí porážet nepřátele jenž, jsou z počátku mnohem složitější než později, když si nakoupí vylepšení.

Práce mi přinesla spousty nových znalostí k Unity ale dokonce i jazyku c#. Mezi znalosti Unity patří naučení se nových předtím mě neznámých funkčností samotného enginu usnadňujících práci, mimo jiné jsem si vyzkoušel, jak složité je dělat sprity a animace do her a to ještě bez přidání třetí dimenze. 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.

Cíl bych řekl, že je splněn, je vytvořena buď kratší hra nebo demo vetší hry. Další práce by obsahovala rozšíření, která mě jako vždy napadají až když je pozdě na samotnou jejich implementaci a vytvoření dalšího obsahu, více typů nepřátel, bossů, místností a typů terénu (bloků).

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] Gailloreto, C. *Historie roguelike žánru* [online]. Roguebasin, prosinec 2020 [cit. 2022-01-30]. Dostupné z: https://screenrant.com/roguelike-definition-games-rogue-hades-roguelite-dungeon-crawler/.

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

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

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

[7] Neznámý. *Berlínská interpretace* [online]. Roguebasin, květen 2013 [cit. 2022-01-28]. Dostupné z: http://www.roguebasin.com/index.php/Berlin_Interpretation.

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

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

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

[11] 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.

[12] Neznámý. *Platform Game:history* [online]. Wikipedia, duben 2022 [cit. 2022-25-04]. Dostupné z: https://en.wikipedia.org/wiki/Platform_game.

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

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