VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HERNÍ DEMO V UNITY

GAME DEMO IN UNITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB TIMKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Timko Jakub**
Program: Informační technologie
Název: **Herní demo v Unity**
Game Demo in Unity
Kategorie: Počítačová grafika

Zadání:

1. Nastudujte herní engine Unity a techniky tvorby her.
2. Navrhněte herní demo.
3. Implementujte navrženou hru a vytvořte několik levelů.
4. Zhodnoťte a vytvořte demonstrační video.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 a kostra aplikace.

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

Vedoucí práce: **Milet Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem této práce je vytvoření herního dema v herním enginu Unity. Demo se skládá ze dvou částí. První částí je hra samotná, kde hráč ovládá kuličku, se kterou překonává překážky. Druhou částí je editor, který umožňuje vytváření dalších úrovní hry. V práci lze najít stručné rozebrání existujících her s podobnou tématikou a porovnání herních enginů. Rovněž se zde nachází popis návrhu a implementace obou částí dema.

Abstract

The aim of this thesis is to develop a game demo using the Unity game engine. The demo consist of two parts. The first part is the game itself, where player controls a ball, with which he overcomes obstacles. The second part is a game editor, which serves to create additional levels of the game. The thesis contains a brief analysis of existing games with similar themes as well as a comparison of game engines. It also contains a description of the design and implementation of both parts of the demo.

Klíčová slova

hra, počítačová hra, herní demo, Unity, Unity3D, C#, command pattern, PlayerPrefs, editor, runitme editor, vytváření levelů, překážková dráha

Keywords

game, computer game, game demo, Unity, Unity3D, C#, command pattern, PlayerPrefs, editor, runtime editor, level creation, obstacle course

Citace

TIMKO, Jakub. *Herní demo v Unity.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet

Herní demo v Unity

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Mileta. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Timko
12. května 2021

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce panu Ing. Tomáši Miletovi za rady, které mi po celou dobu tvorby práce dával, za čas, který se mnou strávil při řešení nejrůznějších problémů, které mě při této práci potkaly, za ochotu, vstřícnost a skvělý přístup.

Obsah

Kapitola 1

Úvod

Když se řekne slovo „hra“, každý si může představit zcela něco jiného. Někdo si představí hru divadelní, někdo jiný zase hru deskovou, či snad hru na schovávanou. Spoustě lidí, převážně však mladším generacím, se ale vybaví videohra. Tedy hra, kterou si můžou zahrát na svém počítači, chytrém telefonu nebo herní konzoli. A tohle je přesně ten druh her, který v poslední době zaznamenává raketový rozmach a lidé se s ním čím dál více setkávají.

V dnešní době se denně vydávají desítky videoher. Některé jsou od obrovských herních studií, které zaměstnávají tisíce lidí, jiné od malých studií, které mají desítky až stovky zaměstnanců. A potom jsou tu hry vyvinuté jedním člověkem, který tak musí být v jednu chvíli návrhář, vývojář i tester. I tyto hry vznikají kvůli zisku, ale mnohdy převážně ze zapálení daného člověka pro věc a touhy poskytnout sobě a dalším lidem zábavu a odreagování. Toto je možné i díky tomu, že dnes existuje mnoho veřejně dostupných nástrojů a tvorba videoher je tak stále snazší.

Už delší dobu mě lákalo vytvořit si vlastní videohru. Pořád jsem se k tomu, ale nemohl dostat. Až mi jednou kamarád ukázal hru, kterou si vytvořil pro chytrý telefon v herním enginu Unity. Začal jsem se o tento nástroj více zajímat, shlédnul několik tutoriálových videí a vytvořil první herní prototypy, abych si alespoň trošku osvojil základy práce v tomto enginu. Nakonec jsem se pustil do tvorby první větší hry, kde jsem chtěl všechny své nově získané znalosti skloubit dohromady. Její vývoj mě bavil natolik, že když jsem uviděl možnost dělat na toto téma bakalářskou práci, nemusel jsem dlouho váhat. Chci ukázat, že vytvořit hru pro zábavu, ať už pro svoji, svých kamarádů nebo širšího okolí, se dá zvládnout a je k tomu potřeba jen chuť, trpělivost a spousta volného času.

Cílem této bakalářské práce je vytvořit herní demo v Unity. To se skládá ze dvou hlavních částí. První částí je hra samotná, kde hráč ovládá kuličku, se kterou překonává překážky. Druhou částí je herní editor sloužící pro vytváření vlastních úrovní hry. Díky němu u hry může hráč strávit více času, aniž by ho omrzela a rovněž ukázat svoji kreativitu.

Dále se v této práci nachází:

Stručné rozebrání existujících her s podobnou tématikou, které byly inspirací pro tuto práci v kapitole 2.

Seznámení se s pojmem „herní engine“ a popis vybraných herních enginů v kapitole 3.

V kapitole 4 se nachází popis návrhu a přístupu k tvorbě obou částí herního dema.

Shrnutí implementace těchto dvou částí se poté nachází v kapitole 5.

A v poslední kapitole 6 je zhodnocení průběhu a výsledku celé práce.

Kapitola 2

Hry založené na překonávání překážek

Existuje opravdu mnoho herních titulů, které jsou založeny na tom, že hráč musí nějakým způsobem překonávat nejrůznější překážky. Spousta z nich rovněž obsahuje editor, umožnující vytvářet vlastní úrovně hry. V této kapitole budou uvedeny tři z nich, které byly inspirací pro herní demo, jež je cílem této práce – ať už z hlediska herní části nebo z hlediska editoru.

2.1 Golf It!

Golf It! je multiplayerová[1] hra, která se snaží co nejrealističtěji napodobit fyziku skutečného golfu. Hráč se zde, jako v klasickém golfu nebo minigolfu snaží svůj míček dostat do jamky, na co nejmenší počet úderů. Míček odpaluje pomocí myši. Systém odpalování zde funguje tak, že čím rychleji hráč posune myší dopředu, tím větší silou míček odpálí (díky čemuž míček doletí dál). Na každé jamce musí hráči překonat velké množství nejrůznějších překážek a nástrah.

Ve hře je k dispozici několik již vytvořených map. Každá mapa má odlišné prostředí a skládá se z několika jamek, kterými hráči postupně procházejí. Počty úderů jednotlivých hráčů na dané jamce se průběžně sčítají a na konci vyhrává ten, který jich má celkově nejméně.

Nedílnou součástí je editor, ve kterém hráč může vytvářet vlastní mapy, které si potom může kdokoliv, kdo hru vlastní, zahrát. Mapu může vytvářet vícero hráčů zároveň. Obrázek 2.1 ukazuje, jak vypadá grafické uživatelské rozhraní (GUI) editoru.

Hra byla vyvinuta a vydána v roce 2017 nezávislou společností Perfuse Entertainment[2]. Jedná se o jejich první a zatím jedinou hru.

[1] Hra pro více hráčů.

[2] https://store.steampowered.com/developer/PerfuseEntertainment

Obrázek 2.1: Pohled na editor ve hře Golf It!. Jeho hlavní část tvoří ScrollView[3], kde si hráč vybere objekt, který chce do scény umístit. Na levé straně je menu, kde si vybere, jaký typ objektu chce umístit a v pravém menu si vybírá vzhled objektů podle jednotlivých tématických prostředí. Objekty ve scéně lze následně přemisťovat, rotovat, či měnit jejich velikost.

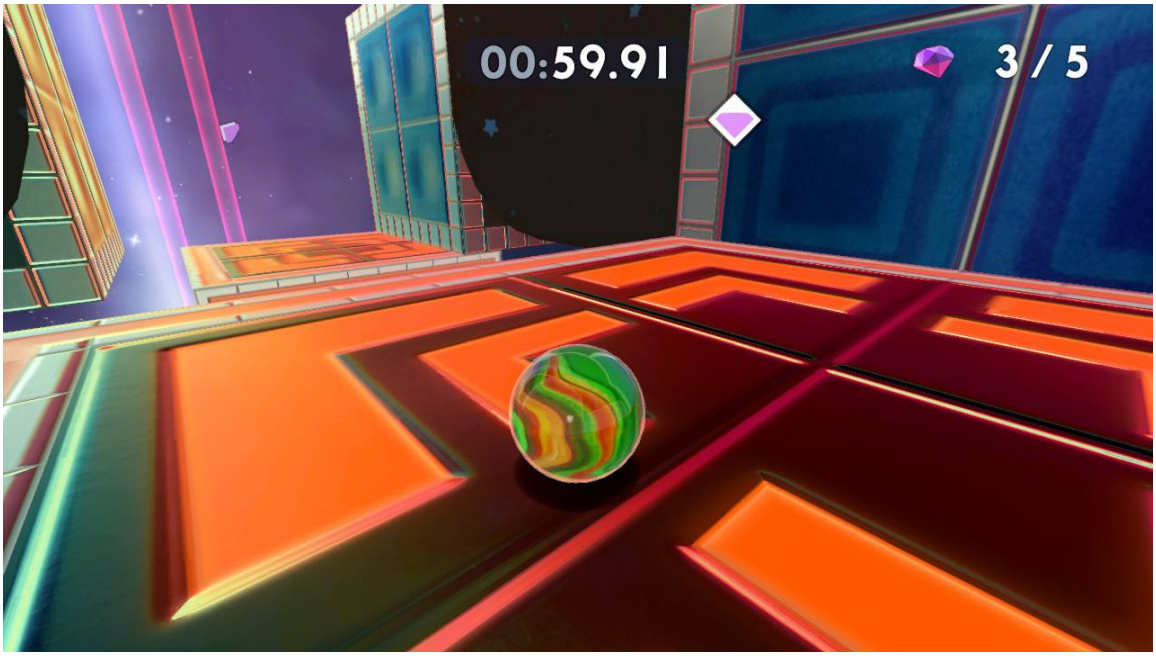
2.2 Marble It Up!

Marble It Up! je singleplayerová[4] hra, ve které hráč ovládá kuličku, s níž se musí prokutálet do cíle v co nejkratším čase. V každém levelu (úrovní hry) mu v cestě stojí variace nejrůznějších překážek, které musí překonat. Na konci každého levelu hráč obdrží medaili, podle toho, jaký měl čas. Každý level má stanovený časový limit, který hráč musí splnit k tomu aby získal určitou medaili – ty jsou od bronzové, přes stříbrnou a zlatou až po diamantovou. Ke zdolání levelu v nejlepším čase a získání diamantové medaile je častokrát potřeba úroveň opakovat pořád dokola. Hra je dělaná tak, že nezjevnější cesta není vždy ta nejkratší a nejrychlejší a hráč tak musí hledat nejrůznější zkratky, které mu pomohou vylepšit jeho čas o dalších pár vteřin. Pohled na hru je možné vidět na obrázku 2.2.

Hra obsahuje takzvaný Ghost Mode, kde hráč závodí proti svému nejlepšímu času v daném levelu. Na trati je tak další kulička, která je vlastně záznam hráčova nejlepšího pokusu. Je tedy hned zřejmé kde je hráč v nynějším pokusu rychlejší nebo naopak pomalejší. V každém levelu se různě vyskytují vylepšení (power-ups), které umožňují po krátkou dobu rychlejší pohyb, super skok nebo například létání.

Na vývoji hry se podílelo hned několik nezávislých herních studií – Blueteak, The Engine Company, Shapes and Lines a Arcturus Interactive. Vydána byla v roce 2018 vydavatelstvím Bad Habit Productions.

[3] Okno, ve kterém jde rolovat.
[4] Hra pro jednoho hráče.
[5] https://www.darkstation.com/reviews/marble-it-up-review



Obrázek 2.2: Obrázek jednoho z levelů ve hře Marble It Up! V horní části obrázku lze vidět stopky zaznamenávající čas a také počítadlo sebraných drahokamů. Obrázek převzatý z internetu[5].

2.3 Fall Guys: Ultimate Knockout

Fall Guys je multiplayerová battle royal[6] hra až pro 60 hráčů, nápadně se podobající televizním soutěžím jako Takešiho hrad[7] nebo Wipeout[8]. Každá hra se skládá z několika kol, či miniher, kterými hráči postupně procházejí. Na konci každého kola ze hry vypadne určitý počet hráčů, až nakonec zůstane jen jeden, který se stává vítězem. Jednotlivá kola jsou sestavena z nejrůznějších překážek, kterými musí hráči projít, co nejrychleji, jinak jsou vyřazeni. Hráči hrají sami za sebe, ale při některých minihrách jsou rozděleni do týmů a k tomu, aby porazily další týmy musí výrazným způsobem spolupracovat. Po neúspěchu pak ze hry vypadává celý tým. Hra obsahuje velký počet překážkových drah, či miniher, které jsou pro každou hru vybrány náhodně, tím hra působí rozmanitěji. Obrázek 2.3 ukazuje jednu z herních překážek s názvem Door Dash.

Důležitým aspektem hry je fyzika herních postav, které na první pohled připomínají fazolky. Jedná se o takzvanou RagDoll fyziku neboli fyziku hadrového panáčka. Kdy by se měla postava například při pádech chovat bezvládně a tím by měl její pohyb působit co nejvíce realisticky.

Hra byla vyvinuta nezávislým herním studiem Mediatonic[9] a vydána vydavatelem Devolver Digital v roce 2020.

[6]Herní žánr, ve kterém je v jedné hře velké množství hráčů naráz a vyhrává ten, který zůstane jako poslední. Hráči se buď vyřazují mezi sebou nebo je, jako v tomto případě, vyřazuje hra.

[7]https://www.csfd.cz/film/134166-takesiho-hrad/prehled/

[8]https://www.csfd.cz/film/265116-drtiva-porazka/prehled/

[9]https://www.mediatonicgames.com/

Obrázek 2.3: Obrázek ze hry Fall Guys: Ultimate Knockout. Snímek je z kola s názvem Door Dash. Zde hráči překonávají několik po sobě jdoucích stěn, které jsou plné dveří, z nichž jsou ovšem jen některé pravé a dá se jimi projít, ty ostatní hráče zastaví a shodí jej na zem. Obrázek převzatý z internetu[10].

[10] https://www.playstation.com/cs-cz/games/fall-guys-ultimate-knockout/

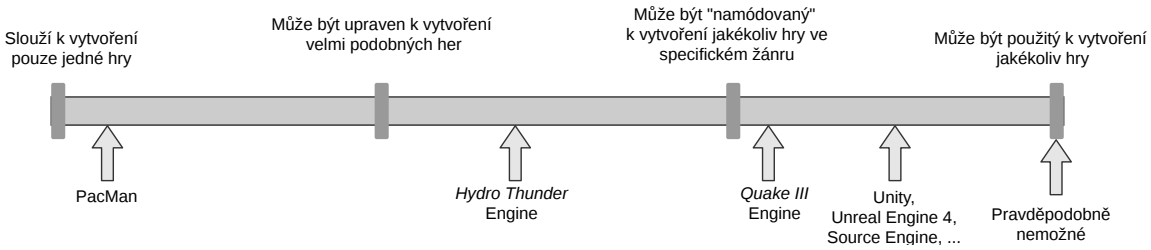
Kapitola 3

Herní engine

Herní engine je software sloužící k vývoji videoher. Pojem „herní engine“ se začal poprvé objevovat v polovině devadesátých let ve spojitosti s hrami žánru FPS[1], jako je například, v té době velice slavný, Doom od id Software. Tato hra byla navržena tak, že měla rozumným způsobem oddělené jádro (vykreslování 3D grafiky, systém pro detekce kolizí nebo audio systém) a herní svět, umělecké prvky (art assests) nebo pravidla hry, které tvořily herní požitek hráče. Hodnota tohoto rozdělení se projevila v momentě, kdy id Software začal toto jádro licencovat a vznikaly první hry, u kterých stačilo pouze vytvořit vlastní prostředí, pravidla hry a umělecké prvky a samotné jádro pak zůstalo v podstatě nezměněno. Toto dalo za vznik „mód komunitě“ – skupině jedinců nebo malých nezávislých studiích, které vytvářely nové hry tím, že modifikovaly ty již existující, za použití nástrojů od původních vývojářů.

V dnešní době si můžou vývojáři herní engine licencovat a využít jeho klíčové softwarové komponenty za účelem vytváření her. Tato skutečnost je pro většinu z nich ekonomicky daleko výhodnější než kdyby si měli celé jádro sami vyvinout. Herní enginy se dnes snaží nezaměřovat jen na jeden typ her a jsou více univerzální. Existují ale stále i enginy, které jsou udělané na míru konkrétní hře nebo hernímu žánru a rozdíl mezi tím, co je engine a co už je hra je velmi nejasný. [6]

Na obrázku 3.1 je graficky znázorněna míra znovupoužitelnosti vybraných herních enginů.



Obrázek 3.1: Škála znovupoužitelnosti vybraných herních enginů. Obrázek převzatý z [6]

.

[1] FPS – first-person-shooter (střílečka z první osoby)

3.1 Unreal Engine

Unreal Engine vyniká svými grafickými schopnostmi, prací se světly, shadery a mnohým dalším. Díky těmto skutečnostem stojí za spoustou z těch nejpopulárnějších her, které jsou dnes k dispozici. Tento engin byl vyvinut velice specificky tak, aby zvládal ty nejnáročnější úkoly lépe než ostatní enginy a proto dnes patří k těm nejmodernějších herním enginům.

Mezi jeho vlastnosti patří fotorealistické vykreslování, dynamická fyzika objektů, efekty jako realistické animace a mnoho dalšího. Slouží pro tvorbu 3D i 2D her. Jeho programovacím jazykem je C++. Cena je 5% z vydělané částky nad 1 000 000 $, jinak je tento engine zdarma. [11]

Unreal engine stojí za některými z nejúspěšnějších her všech dob, mezi které můžeme řadit například: série Borderlands, Gears of War, Hellblade: Senua's Sacrifice (viz obrázek 3.2), PUBG, Fortnite nebo sérii Mass Effect.

Obrázek 3.2: Snímek ze hry Hellblade: Senua's Sacrifice. Jedná se o fantasy akční adventuru, která byla vyvinuta a publikována nezávislým studiem Ninja Theory v roce 2017. Byl zde kladen velký důraz na grafické zpracování a mnozí ji považují za umělecké dílo. V roce 2018 Hellblade obdržel několik ocenění, mimo jiné za umělecké, zvukové a herecké zpracování. Obrázek převzatý z internetu[2].

[2] https://games.tiscali.cz/recenze/hellblade-senuas-sacrifice-recenze-301401

3.2 Amazon Lumberyard

Amazon Lumberyard slouží pro tvorbu 3D her a jeho hlavní předností je přímá a snadná integrace na Twitch a AWS[3]. Zaměřuje se na kvalitní vizuální prvky a výkon. Je zcela zdarma, uživatel platí pouze za využití služeb AWS. Programovacím jazykem zde je Lua nebo se dá tvořit ve vizuálním skriptovacím prostředí Script Canvas. [1]

Mezi hry vytvořené v Lumberyard enginu patří The Grand Tour, Star Citizen nebo New World (viz obrázek 3.3).

Obrázek 3.3: Snímek ze hry New World. Jedná se o MMORPG (massively multiplayer online role-playing game – online hra na hrdiny o více hráčích) hru od společnosti Amazon Games, která by měla být vydána v létě 2021. Obrázek převzatý z internetu[4].

3.3 CryEngine

Podobně jako Unreal Engine, popsaný v sekci 3.1, je CryEngine vhodný pro tvorbu těch graficky nejnáročnějších 3D her. Mezi jeho hlavní přednosti patří tvorba detailně propracovaných a realisticky vypadajících postav. Byl vyvinut společností Crytek.

Crytek si nárokuje 5% z vydělané částky, pokud roční tržba přesáhne 5 000 $, jinak je engine zdarma. Ke skriptování se v něm používá jazyk Lua. [2]

V CryEngine vzniklo mnoho her té nejvyšší úrovně, jako jsou: Kingdom Come: Deliverance (viz obrázek 3.4), série Crysis, Hunt, Prey, série Sniper Ghost Warrior nebo Evolve. V modifikované podobě, pod názvem Dunia Engine jej používá Ubisoft a vznikly v něm například hry ze série Far Cry (až na první díl Far Cry, ten byl vyvinut v samotném CryEnginu).

[3] AWS – Amazon Web Services (cloudové výpočetní služby)

[4] https://www.newworld.com/en-us/game/media

[5] https://www.kingdomcomerpg.com/cs/media

Obrázek 3.4: Snímek ze hry Kingdome Come: Deliverance. Jedná se o historickou RPG hru od českého herního studia Warhorse Studios, která vyšla v roce 2018. Hra se odehrává v roce 1403 a je založena na skutečném příběhu krále Karla IV. a jeho synů. Ve hře se nachází podrobná rekonstrukce země Českého království z 15. století. Hra se vyjímá unikátním bojovým systémem, detailně zpracovaným prostředím a mnohým dalším. Obrázek převzatý z internetu[5].

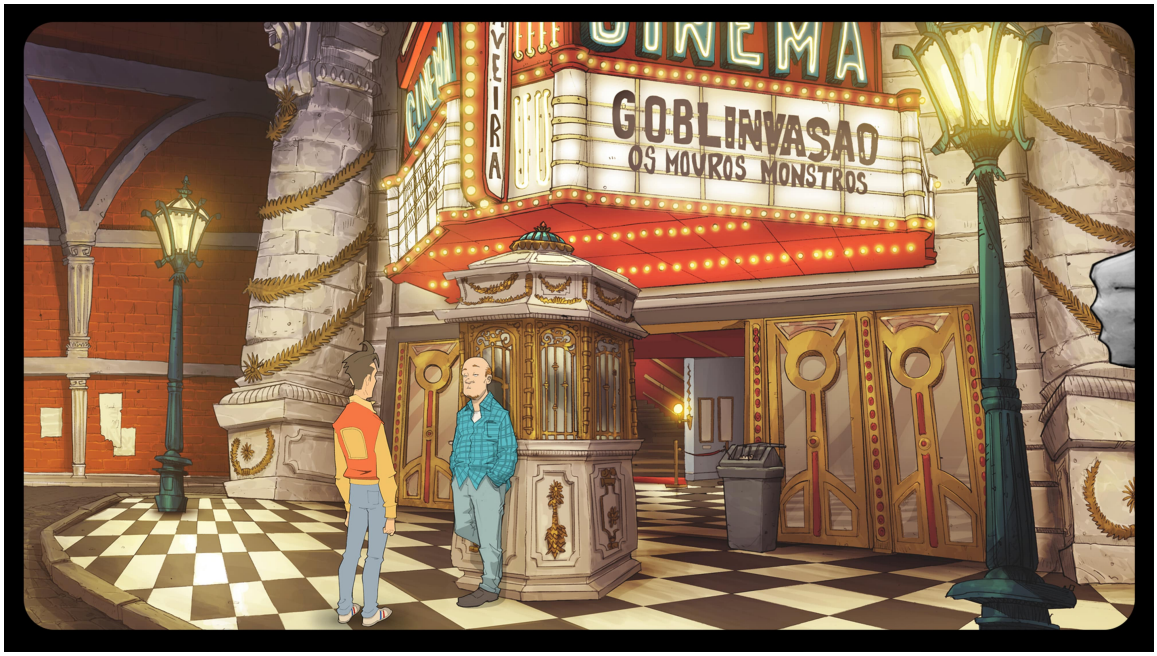
3.4 Godot

Godot je zcela zdarma a open-source[6], se svobodnou licencí MIT. Bez žádných poplatků nebo závazků. Jeho nejsilnější stránkou je systém scén a uzlů, který usnadňuje organizaci her a tím urychluje jejich vývoj a zlepšuje jejich škálovatelnost. [5]

Kolem Godotu je velká komunita, která neustále vyvíjí nové funkcionality a opravuje případné chyby. Engine je vhodný pro tvorbu 2D i 3D her a je zcele zdarma, bez žádných poplatků z výdělku na hře. Disponuje vícero možnostmi, jak v něm skriptovat – GDScript, C#, C++ nebo vizuální skriptování.

Mezi hry vyvinuté v tomto enginu patří Gun-Toting Cats, Kingdoms of the Dump nebo například The Interactive Adventures of Dog Mendonça & Pizzaboy (viz obrázek 3.5).

[6] otevřený software – počítačový software s volně přístupným zdrojovým kódem



Obrázek 3.5: Snímek ze hry The Interactive Adventures of Dog Mendonça & Pizzaboy, vytvořené v herním enginu Godot. Jedná se o dobrodružnou adventuru, která byla vyvinuta společností OKAM Studio a vydal ji Ravenscourt v roce 2016. Obrázek převzat z internetu[7].

3.5 Unity

Real-time 3D vývojová platforma Unity nechává umělce, designéry a vývojáře společně pracovat, aby vytvořili úžasné a pohlcující interaktivní zážitky. [8]

Existuje mnoho herních enginů, ale Unity se prokázalo jako jeden z nejlepších. V Unity lze vytvořit 2D nebo 3D hry mnoha herních žánrů, jako například FPS, RPG, adventury, závodní hry a mnohé další. Kromě toho, že Unity umožňuje vytvářet hry té nejvyšší kvality se snadno použitelným rozhraním, nabízí také možnost exportovat hru na široké množství platforem včetně mobilní platformy (Android, iOS, Windows), platformy pro virtuální realitu (Oculus Rift, Google Cartboard nebo PlayStation VR), desktopové platformy (Windows, Mac nebo Linux) nebo herní konzole (PS4, Xbox One a Nintendo Switch).

Unity obsahuje všechny potřebné nástroje k tvorbě her a také zjednodušuje použití užitečných technik pro vylepšení kvality her. Obsahuje například Visual Studio – vývojové prostředí (IDE), které urychluje a usnadňuje tvorbu skriptů, vestavěnou umělou inteligenci (navmesh navigation, ...), kterou lze použít i bez předešlých zkušeností s ní, světla, vestavěné objekty nebo konečný stavový automat použitelný k definování chování postav a animací. Vlastnosti jednotlivým herním objektům se přidávají pomocí skriptů, které je možné psát v jazyce C#. [3]

Unity je zcela zdarma, při výdělku nižším než 100 000 $ ročně. Pokud výdělek tuto částku přesáhne, je nutné platit licenci ve výši 40 $ měsíčně nebo 399 $ ročně za každého člena týmu (Unity Plus). Pokud roční výdělek přesáhne částku 200 000 $ je nutné si platit licenci ve výši 150 $ měsíčně nebo 1 800 $ ročně za každého člena týmu (Unity Pro).

[7] https://games.tiscali.cz/the-interactive-adventures-of-dog-mendonca-pizzaboy-17787

Mezi hry vyvinuté v tomto enginu patří: Escape from Tarkov (viz obrázek 3.6), Hollow Knight, Cuphead, In the Valley of Gods, Orisis: New Dawn, GTFO, Call of Duty: Mobile, Fall Guys: Ultimate Knockout, Valheim a mnohé další.

Obrázek 3.6: Snímek ze hry Escape from Tarkov. Jedná se o realistickou RPG střílečku z pohledu první osoby, která byla vyvinuta studiem Battlestate Games a vydána v roce 2017. Obrázek převzatý z internetu[8].

[8] https://www.gamebro.cz/divoke-prestrelky-escape-from-tarkov-videu/

Kapitola 4

Návrh

Zadáním této práce bylo vytvořit herní demo v herním enginu Unity. Rozhodl jsem se vytvořit 3D hru, ve které hráč bude ovládat kuličku, se kterou bude překonávat nejrůznější překážky. Cílem každého levelu hry je posbírat všechny „drahokamy“ v co nejkratším čase. Toto je tedy složení a smysl hry samotné. Druhá část tohoto dema je herní editor, ve kterém hráč může vytvářet vlastní levely pomocí sady předdefinovaných objektů (prefabs), které přidává do scény podle své představy. Každý prefab má předdefinované vlastnosti, jako například směr pohybu, rychlost, délka dráhy jeho pohybu atd. Tyto vlastnosti může hráč u každého objektu upravovat tak, aby vše sedělo do jeho vlastního levelu.

Ve hře nebyl žádným způsobem kladen důraz na její grafické zpracování, veškeré objekty v ní, až na malé výjimky, jsou tvořený základními geometrickými tvary (koule, krychle, atd.) bez přidaných textur. Za důležité jsem považoval rozmanitost herních mechanik a co nejlepší zpracování editoru, aby se v něm levely daly tvořit co nejjednodušším, nejrychlejším a intuitivním způsobem.

Inspirací při tvorbě tohoto dema byly hry (viz kapitola 2):

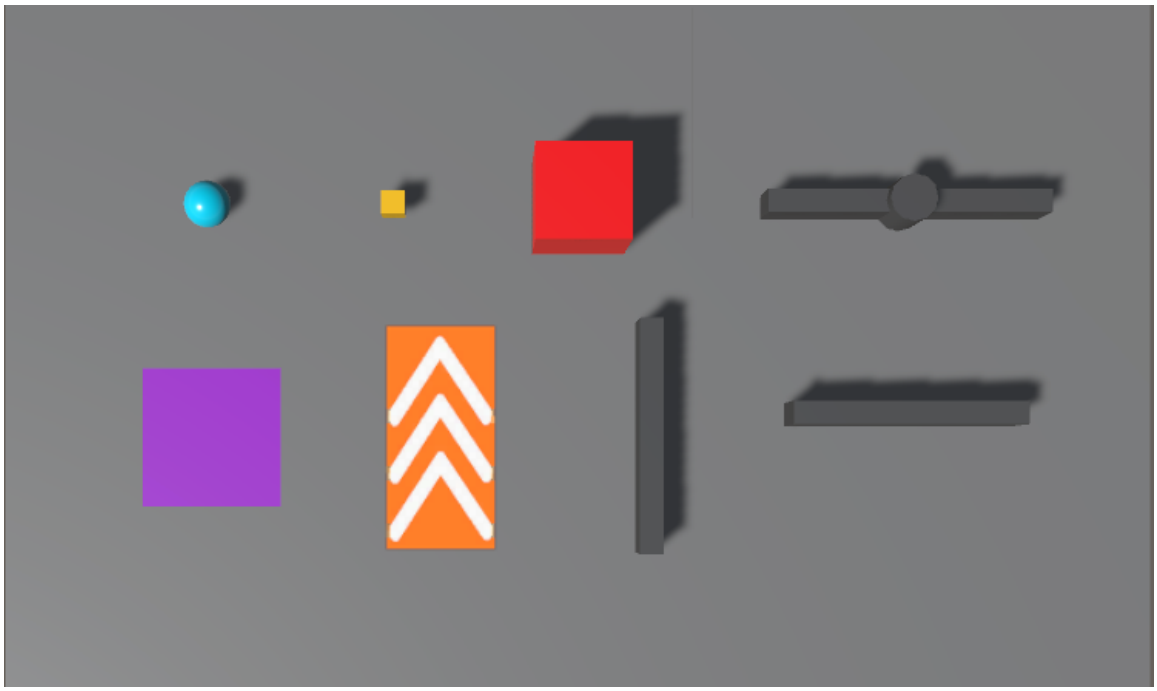
- Golf It! – inspirace při tvorbě editoru,
- Marble It Up! – inspirace při tvorbě herní části, systému sbírání drahokamů a při tvorbě herních překážek,
- Fall Guys – inspirace při tvorbě herních překážek

4.1 Herní část

Návrh této části byl poměrně přímočarý. Hra bude obsahovat několik ukázkových levelů a dále pak levely vytvořené hráčem pomocí editoru. V každém levelu (scéně) se nachází různě rozmístěné překážky tak, aby přes každou z nich hráč musel projít. Překážek (objektů) se ve hře nachází hned několik typů (viz obrázek 4.1):

- Stěny, podlahy a mosty – vymezují prostor, ve kterém se hráč může pohybovat.
- Pohyblivé plošiny – slouží k přemisťování mezi úseky prázdnoty. Jinými slovy se tato platforma pohybuje v prostoru sem a tam (může se pohybovat do všech směrů) a hráč se po ní může „svézt“, aby překonal úsek mezi dvěma oddělenými kusy podlah.

- Drahokamy – jsou rozmístěné po levelu a hráč je, k tomu aby vyhrál, musí posbírat všechny. Drahokamy se otáčejí a levitují ve vzduchu, aby na první pohled přitáhly hráčovu pozornost.
- Nepřátelé – pokud se jich hráč dotkne, musí daný level opakovat od začátku nehledě na to, jak daleko už byl. Nepřítel se pohybuje sem a tam po určité dráze.
- Speciální nepřítel – pronásleduje hráče po vyhrazeném úseku levelu. Podobá se to hře na honěnou, hráč se nesmí nechat chytit, v opačném případě musí level opakovat.
- Otáčedla (spinnery) – snadno mohou hráče shodit do prázdného prostoru po levelem. Spinnery se otáčejí dokola kolem vlastní osy.
- Trampolíny – vymrští hráče směrem nahoru.
- Zrychlovadla – vymrští hráče směrem vpřed.

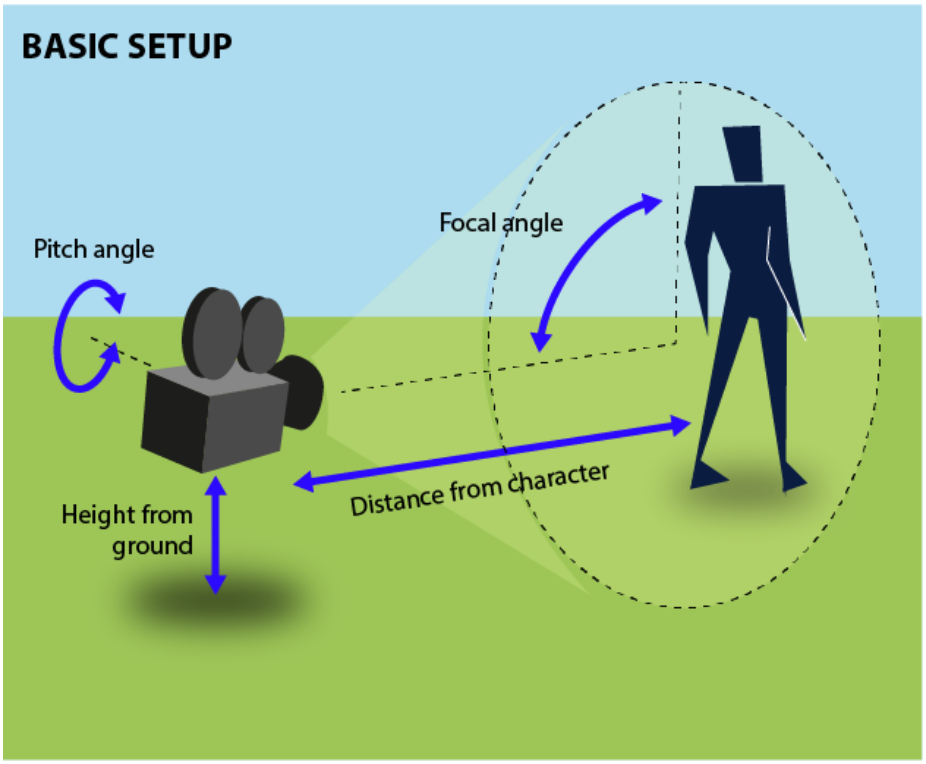


Obrázek 4.1: Ukázka návrhu některých z herních objektů. V první řadě zleva – hráč, drahokam, nepřítel a otáčedlo. Ve druhé řadě zleva – trampolína, zrychlovadlo, svislá a vodorovná zeď.

Hráč ovládá kuličku pomocí klávesnice a myši. Klávesnicí ovládá její samotný pohyb a pomocí myši kolem ní otáčí kameru. Kamera se stále „dívá“ na kuličku a pohybuje se spolu s sní, přičemž si stále udržuje určitý odstup. Hráč má tedy kuličku stále před sebou. Tento druh kamery se nazývá „pohled z třetí osoby“ (viz obrázek 4.2). O svém čase a počtu sebraných drahokamů, včetně jejich celkového počtu, který se ve scéně nachází, má hráč stále přehled.

Pokud hráč spadne z levelu do prázdného prostoru, nemusí opakovat celý level od znova, ale hra ho vrátí zpět na místo před pádem, tím ovšem ztratí drahocenné vteřiny. Na konci levelu – poté, co posbírá všechny drahokamy – si hráč může uložit čas, za který ho překonal,

do tabulky, která existuje zvlášť pro každý level hry. V této tabulce jsou zaznamenány všechny uložené časy hráčů na daném levelu. Hráči tak mají možnost mezi sebou lokálně soutěžit, aby zaujali první místo v tabulce, podobně jako na klasických arkádových hrách[1].



Obrázek 4.2: Ukázka kamery z pohledu třetí osoby. Na obrázku jsou znázorněny údaje důležité při nastavování kamery. Je potřeba určit její natočení, vzdálenost od hráče, výšku od země, úhel ohniska, atd. Obrázek převzatý z internetu[2].

4.2 Editor

Editor je velmi důležitou součástí tohoto herního dema. Dává hráčům možnost vytvářet vlastní úrovně a tím připravit nové výzvy pro sebe nebo ostatní hráče. Mezi klíčové operace, které je možné provádět v editoru patří:

- Vkládat do scény nové objekty.
- Upravovat vlastnosti již vložených objektů (měnit rychlost jejich pohybu, délku dráhy pohybu, atd.) Každý objekt má již předdefinované vlastnosti, aby fungoval i bez hráčova zásahu. Nicméně lze tyto vlastnosti u jednotlivých objektů upravit na míru každému levelu.
- Měnit rotaci již vložených objektů podle všech tří os (x, y, z).
- Přemisťovat již vložené objekty.

[1] https://en.wikipedia.org/wiki/Arcade_game
[2] https://gamasutra.com/blogs/YoannPignole/20150928/249412/Third_person_camera_design.php

- Odstraňovat již vložené objekty.
- Uložit vytvořený level pod nějakým názvem.
- Načíst již dříve vytvořený level podle jeho názvu.

Editor má dva režimy – Edit a Play. Režim Edit je výchozí režim a editor se v něm nachází bezprostředně po jeho spuštění. V tomto režimu je kamera se kterou hráč může libovolně pohybovat do všech směrů, natáčet ji, rotovat, přibližovat, atd. Jedná se o takzvaný „volný pohled“, díky kterému si hráč může kameru nastavit vždy do ideálního úhlu pro provádění konkrétní operace. Pomocí myši si poté hráč vybírá objekty z nabídky a umisťuje je do scény. Po kliknutí na některý z objektů v nabídce se kolem myši objeví obrys daného objektu, aby měl hráč jasnou představu, kam objektu umisťuje. Tento obrys se pohybuje souběžně s kurzorem myši a může mít buď zelenou (na tohle místo lze objekt umístit) nebo červenou barvu (na tohle místo objekt umístit nelze). Po kliknutí levým tlačítkem myši na vyhovující místo ve scéně se do ní daný objekt vloží. Tomuto objektu lze upravit jeho vlastnosti, změnit jeho natočení, přemístit ho na jiné vhodné místo nebo objekt ze scény odstranit.

Pokud je hráč se stavem levelu již spokojený, může si jej uložit pod nějakým názvem. Případně pokud chce pokračovat v práci na nějakém již dříve vytvořeném levelu, může si ho v editoru jednoduše znovu otevřít tím, že zadá jeho název – do scény se poté načtou všechny objekty, které v tomto levelu byly uloženy, včetně všech jejich vlastností a všech dalších modifikací, které na nich hráč provedl.

Jedním kliknutím se hráč může přepnout do režimu Play a všechny objekty rázem ožijí a rozhýbou se podle jejich vlastností. V tomto režimu ovládá kuličku stejně jako v hře samotné (viz sekce 4.1). Tento mód je zjednodušenou verzí samotné herní části dema, slouží k tomu, aby si hráč mohl vyzkoušet jak daný level vypadá, jestli jej lze projít a jestli se v něm všechny objekty chovají tak, jak si představoval. Ke všem hráči vytvořeným levelům se dá přistoupit přímo z hlavní nabídky. Odtud si je poté možno level zahrát se vším všudy.

Editor obsahuje další funkcionality, které jsou na první pohled skryté a aktivují se pomocí klávesových zkratek. Funguje v něm například funkce „zpět“, kterou hráč provede stiskem kláves CTRL + Z. Tato funkce vrátí zpět poslední provedenou operaci, takže pokud například hráč právě umístil do scény nový objekt a stiskne tyto dvě klávesy, objekt bude ze scény odebrán. Opačně to funguje při vymazání objektu, objekt tedy bude umístěn zpět do scény na jeho původní místo. Obdobně se editor zachová i při rotaci nebo přemístění objektu a následné aktivaci této funkce. Dále se zde dá uvést funkce „snappingu“ (CTRL + X), která se využívá při umisťování objektů – objekty na sebe navazují. Poslední takovou funkcí je „stepping“ (CTRL + C) (krokování), který se využívá při změně rotace objektů – objekty mění svoji rotaci po 30°.

Kapitola 5

Implementace

Celé demo bylo vytvořeno v herním enginu Unity – verze 2019.4.10f1 Personal. Jednotlivé skripty byly psány v jazyce C# a jako IDE[1] jsem použil Visual Studio 2019, které je pro vývoj v Unity ideální.

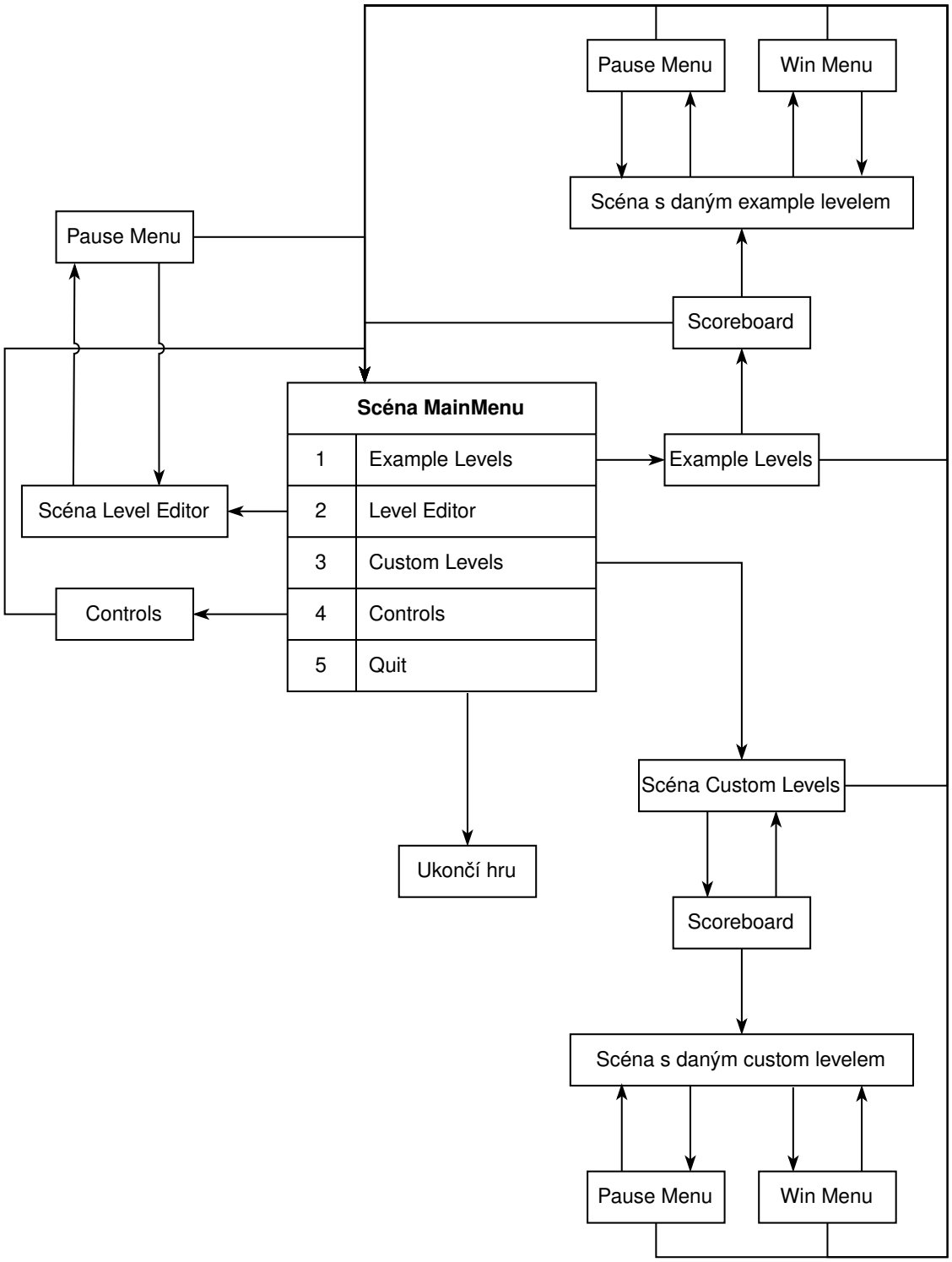
Strukturu dema tvoří tyto scény:

- Scéna hlavního menu
- Scény ve kterých se nachází ukázkové levely
- Scéna editoru
- Scéna pro výběr uživatelem vytvořeného (custom) levelu
- Scéna ve které je možné zahrát si vybraný custom level
- Scéna ve které je tabulka s časy pro daný level

Grafické znázornění propojení jednotlivých menu a scén lze vidět na diagramu 5.1.

Všechny zvuky ve hře jsou řešeny ve třídě `SoundManagerScript` náležející k objektu SoundManager, který se nachází v každé scéně. Kromě této třídy SoundManager obsahuje komponentu Audio Source, která je potřebná k přehrávání zvuků. Všechny zvukové soubory se nacházejí ve složce Resources, která se nachází mezi ostatními assety. Ke všem souborům, které jsou v této složce se dá přistoupit z jakékoliv scény příkazem `Resources.Load`. Ve třídě `SoundManagerScript` se tedy všechny tyto zvuky uloží do proměnných typu AudioClip a na příslušných místech napříč všemi scénami se poté volá funkce `PlaySound("název zvukového klipu")`, která si jako parametr bere název zvukového klipu, který se následně přehraje (zvuky se přehrávají například při sebrání drahokamu, při odrazu od trampolíny, atd.). Dále tato třída obsahuje funkce `OnButtonHover()` a `OnButtonClick()`. Tyto funkce se volají při najetí myší na tlačítko respektive při kliknutí myší na tlačítko a přehrají se v nich dané zvuky. Všechny zvukové efekty byly pořízeny v Unity Asset Store.

[1] IDE - Integrated Development Environment (vývojové prostředí)



Obrázek 5.1: Diagram znázorňující propojení jednotlivých menu a scén ve hře. Šipky znázorňují, kam se dá z jednotlivých menu a scén dostat.

5.1 Menu

Ve hře se vyskytují 3 hlavní druhy menu – Hlavní menu (Main menu), menu při pozastavení hry (Pause menu) a menu které se zobrazí při dokončení levelu (Win menu). Všechna tato menu tvoří tři základní UI prvky: plátno (canvas), tlačítka a textová pole. Pro všechna textová pole byl použit element TextMeshPro (TMP) Text.

TMP používá pokročilé techniky pro vykreslování textu nahrazuje klasické Unity textové elementy. Poskytuje větší kontrolu nad formátováním a rozložením textu. [9]

Scéna Main menu

Main menu je úvodní scéna, takže se zobrazí jako první při spuštění hry (viz obrázek 5.2). Tato scéna kromě kamery a světla obsahuje také Canvas, který má pozadí a jsou v něm tři samy o sobě prázdné herní objekty – MainMenu, ExampleLevelsMenu a ControlsMenu. Tyto objekty fungují jako takzvané „rodičovské objekty“ – seskupují jednotlivé UI elementy stejnojmenných podmenu.

Obrázek 5.2: Screenshot ze scény MainMenu. Je zde možné vidět podmenu MainMenu, které je pro tuto scénu výchozí. Toto podmenu se skládá z pěti tlačítek, které obsahují TMP - Text elementy.

- **Example levels**

  Po kliknutí na toto tlačítko se deaktivuje objekt MainMenu a aktivuje se objekt ExampleLevelsMenu (tím se ve stejné scéně zobrazí jiné menu) (viz obrázek 5.3), které obsahuje seznam ukázkových levelů.

- **Level editor**

  Po kliknutí na toto tlačítko se otevře scéna LevelEditor, o té podrobněji v sekci 5.4. Přepínání mezi scénami řeší příkaz `SceneManager.LoadScene("NazevSceny")`.

- **Custom Levels**

  Po kliknutí na toto tlačítko se otevře scéna SelectCustomLevel, ve které se zobrazí seznam všech existujících hráčem vytvořených levelů (viz obrázek 5.4). Středem tohoto menu je ScrollView jehož obsah se v tomto případě načítá dynamicky – tuto záležitost řeší třída `GetLevelsFromFolder`, která je součástí prázdného objektu GetLevels nacházejícího se v této scéně. Zde se nejdříve zjistí, jestli existuje složka, která custom levely obsahuje – tedy jestli vůbec nějaké custom levely existují. Pokud ano, uloží se názvy všech souborů, které tato složka obsahuje, do pole. Poté se pro každý název v tomto poli zavolá funkce `GenerateItem()`, ve které se do ScrollView vloží nové tlačítko obsahující textový element právě s tímto názvem. Zároveň se ke každému tomuto tlačítku přidá `OnCLick()` event, který zajistí, že pokud se na tlačítko klikne, zavolá se funkce `OpenScoreBoard()`, která si převezme dané tlačítko a získá z něj název levelu.

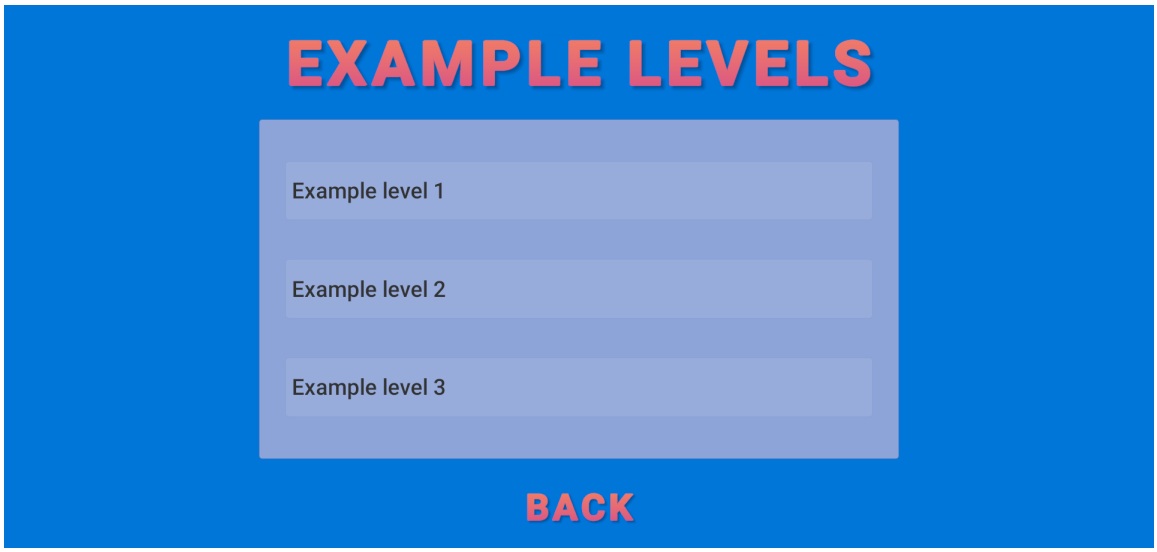
  Dále se otevře scéna Scoreboard (viz podsekce 5.1), která zobrazí uložené časy pro daný level nebo prázdnou tabulku. Dále obsahuje tlačítko Back a Play. Kliknutím na tlačítko Play se zavolá funkce `PrepareScene()`, která si převezme název levelu a otevře scénu PlaySelectedCustomLevels, ve které se díky zavolání funkce `DontDestroyOnLoad("objekt")`, která je vestavěnou funkcí Unity, stále nachází objekt GetLevels, z předchozí scény, nesoucí název levelu, který se má načíst. V této scéně se také nachází prázdný objekt SceneLoader, který nese třídu `LoadScene`. V této třídě se vezme název levelu, který si drží objekt GetLevels a zavolá se funkce `LoadLevel()`, které se tento název předá jako parametr. Následně se najde daný soubor podle názvu ve složce obsahující custom levely. Tento soubor se předá jako parametr funkci `CreateFromFile()`, která do scény načte všechny objekty, které jsou zde uloženy a inicializují se. Více o této funkci v podsekci 5.4.

- **Controls**

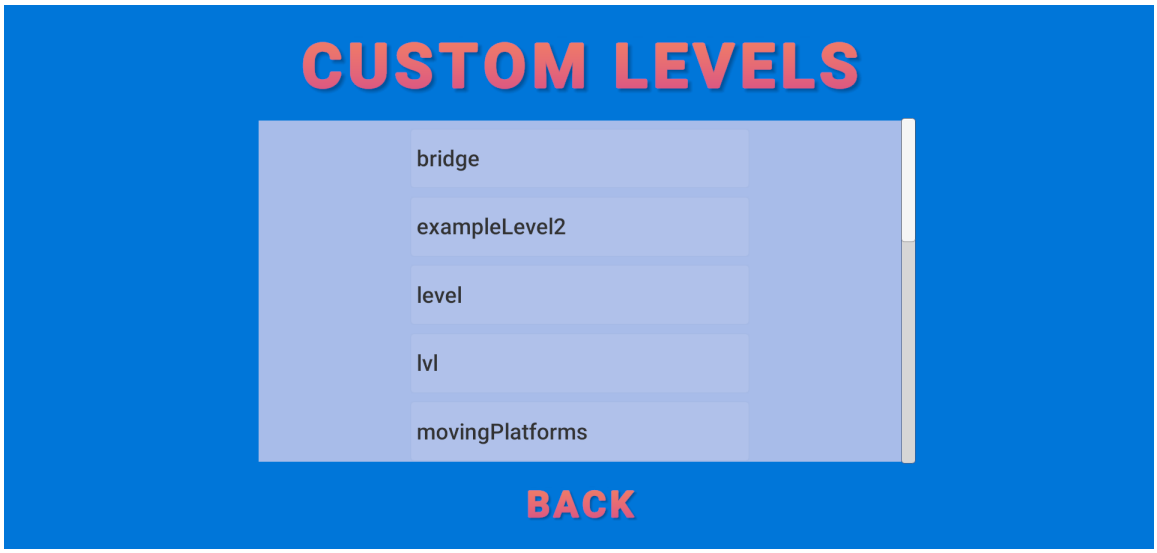
  Po kliknutí na toto tlačítko se deaktivuje objekt MainMenu a aktivuje se objekt ControlsMenu (viz obrázek 5.5). V tomto menu je popsáno ovládání hry.

- **Quit**

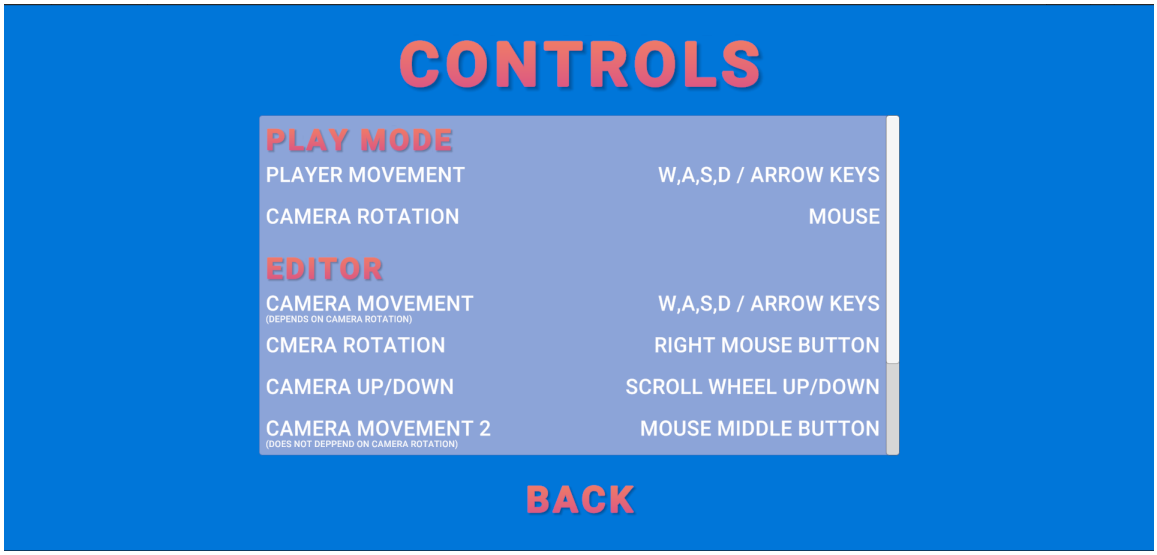
  Při kluknutí na toto menu se zavolá funkce `QuitGame()`, která ukončí hru pomocí příkazu `Application.Quit()`.



Obrázek 5.3: Screenshot z ExampleLevelsMenu, kde se zobrazují předvytvořené levely, které si hráč může zahrát. Menu je tvořené nadpisem (TMP - Text element), UI elementem ScrollView obsahujícím tlačítka, které načtou scénu Scoreboard (viz podsekce 5.1), ve které je tabulka s časy pro daný level. Dále se zde nachází tlačítko Back, které deaktivuje objekt ExampleLevelsMenu a znovu aktivuje objekt MainMenu.



Obrázek 5.4: Screenshot z CustomLevelsMenu, kde se zobrazuje seznam hráčem vytvořených levelů, které je možné si zahrát. Menu je tvořené nadpisem (TMP - Text element), UI elementem ScrollView obsahujícím tlačítka, které načtou scénu Scoreboard (viz podsekce 5.1), která obsahuje tabulku s časy pro daný level. Dále se zde nachází tlačítko Back, které načte scénu MainMenu.



Obrázek 5.5: Screenshot z Controls menu, kde je popsáno ovládání hry a ovládání editoru. Menu je tvořené nadpisem (TMP - Text element), UI elementem ScrollView obsahujícím popisky co se jak ovládá a tlačítkem Back, které deaktivuje objekt ControlsMenu a znovu aktivuje objekt MainMenu.

Scoreboard

Scoreboard je scéna, která se zobrazí vždy, když hráč klikne na konkrétní level, který si chce zahrát. Ať už jde o ukázkový nebo custom level. Jedná se o tabulku časů zaznamenaných jednotlivými hráči na daném levelu. Tato tabulka je tvořená UI elementem ScrollView, jehož obsah se podobně jako u seznamu levelů ve scéně SelectCustomLevel načítá dynamicky (viz obrázek 5.6).

Načítání a ukládání dat je zde řešeno pomocí třídy PlayerPrefs. Jedná se o třídu, která ukládá data mezi jednotlivými relacemi hry. Umí ukládat hodnoty typu string, float nebo integer. Ukládání dat se liší na jednotlivých operačních systémech (na windows se například ukládají do registrů) [10]. Serializovaná data se do této třídy ukládají ve formátů json. Každý záznam obsahuje tyto 3 hodnoty: minuty, sekundy a jméno hráče. Načítání a přidávání se děje ve třídě `ScoreBoard`. Zde se nejprve zjišťuje, jestli již existuje záznam s časy pro daný level. Pokud ne, vytvoří se prázdný. Pokud již záznam s jednotlivými časy existuje, tak se nejprve tyto časy seřadí od nejkratšího po nejdelší (řadí se prvně podle minut a potom podle sekund) a poté se ve funkci `CreateHighscoreEntryTransform()` začnou postupně doplňovat, jako obsah ScrollView elementu.

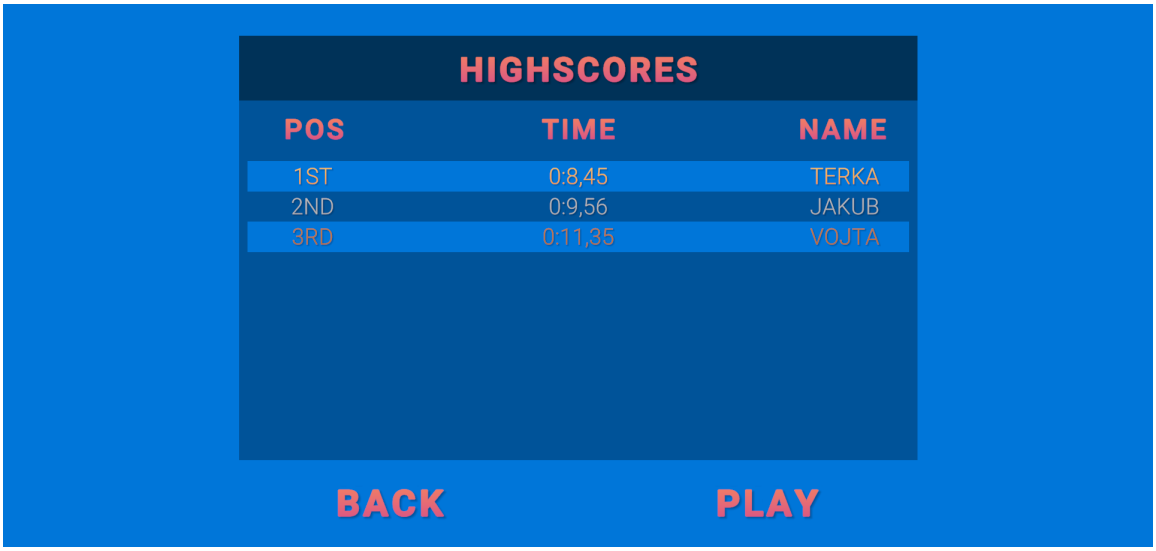
Ukládání nových časů řeší funkce `AddTimeEntry()`, která se volá poté, co hráč dokončí level a rozhodne se uložit si svůj čas. Vyplní svoje jméno, které se spolu s časem vezme a ve formátu json uloží (viz podsekce 5.1).

Načítání uložených dat pomocí PlayerPrefs:

```
1        string jsonString = PlayerPrefs.GetString("nazev levelu");
2        Timescores timescores = JsonUtility.FromJson<Timescores>(jsonString);
```

Zápis dat pomocí PlayerPrefs:

```
1        string json = JsonUtility.ToJson(timescores);
2        PlayerPrefs.SetString("nazev levelu", json);
3        PlayerPrefs.Save();
```



Obrázek 5.6: Screenshot ze Scoreboard, kde se zobrazují hráči uložené časy, za které překonali daný level. Tabulka je tvořená UI elementem ScrollView, do kterého jsou dynamicky po řádcích vkládány jednotlivé časy, seřazené od nejkratšího. Dále se zde nachází tlačítko Back, které deaktivuje objekt Scoreboard a znovu aktivuje objekt MainMenu a tlačítko Play, které načte scénu s daným levelem.

Pause menu

Pause menu je dostupné ve scénách s ukázkovými levely, ve scéně editoru a ve scéně, kam se načítají custom levely. Menu se zobrazí po stisknutí klávesy „Esc“ (viz obrázek 5.7a). V tomto momentě se kromě zobrazení menu také pozastaví hra. K tomuto menu náleží třída `PauseMenu`, ve které se neustále kontroluje jestli nebyla stisknuta klávesa „Esc“. Pokud byla stisknuta, když menu není aktivní, tak se menu aktivuje. Pokud při stisku aktivní bylo, tak se naopak deaktivuje. Dále tato třída obsahuje funkce, které se volají po kliknutí na jednotlivá tlačítka menu. Pozastavení času při aktivaci menu řeší příkaz:

```
1    Time.timeScale = 0f;
```

Win menu

Win menu je dostupné ve scénách kde si hráč může zahrát ukázkové a nebo custom levely. Menu se automaticky zobrazí v momentě, kdy hráč dokončí daný level, tedy po sebrání posledního drahokamu (viz obrázek 5.7b). K tomuto menu náleží třída `WinMenu` obsahující funkce, které jsou volány po kliknutí na jednotlivá tlačítka. Důležitou funkcí této třídy je `SaveLevelTime()`, která obstarává ukládání hráčova času do tabulky příslušného levelu. Tato funkce zpracovává jméno hráče, které vyplnil do textového pole a čas, který ukazují

stopky v momentě sebrání posledního drahokamu. Tento čas je rozdělen na řetězec obsahující minuty a řetězec obsahující vteřiny. S těmito hodnotami (minuty, vteřiny a jméno hráče) a názvem levelu se volá funkce `AddTimeEntry()`, kterou obsahuje třída `Scoreboard`. Zde se jednotlivé hodnoty uloží pomocí `PlayerPrefs` do souboru (viz podsekce 5.1).

(a) PauseMenu obsahuje čtyři tlačítka. „Resume“ zavře toto menu a umožní pokračovat ve hře, „Retry“ znovu načte scénu a tím se restartuje level, „Menu“ otevře scénu hlavního menu a „Quit“ ukončí hru.

(b) WinMenu obsahuje textový element a čtyři tlačítka. „Save level time!“ otevře okno, kde hráč může uložit čas, za který tento level překonal, do tabulky Scoreboard popsané výše. „Menu“ otevře scénu hlavního menu, „Play again“ restartuje level a „Quit“ ukončí hru.

Obrázek 5.7: Screenshoty z PauseMenu a WinMenu.

5.2 Herní objekty

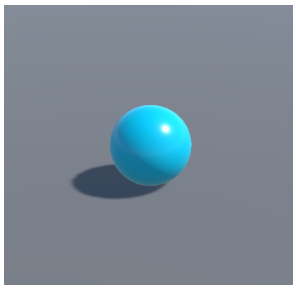
Všechny objekty (překážky), se kterými se hráč může setkat ať už při samotném hraní nebo při vytváření levelu v editoru jsou tvořeny takzvanými prefaby.

Prefaby jsou flexibilní přístup k definování herních objektů. Prefab je tedy plnohodnotný herní objekt (s již připojenými a nastavenými komponentami), který neexistuje v žádné konkrétní scéně, ale je uložený jako tzv. asset[2] ve složce s hrou a může být nakopírován do kterékoliv herní scény. Takovéto vkládání kopií – terminologické označení pro tuto kopii je instance – prefabu do scény může být provedeno buď manuálně v samotném Unity editoru scén, ale důležitější je, že může být do scény vložen ze skriptu pomocí příkazu `Instantiate`. Ve scéně se poté nachází instance, které jsou kopiemi daného prafabu. Pokud se změní vlastnosti prefabu, změní se vlastnosti i jeho instancí. [7]

Všechny objekty obsahují komponentu Collider, která zajišťuje zaznamenávání kolizí mezi objekty. Pohybující se objekty navíc ještě obsahují komponentu Rigidbody, díky které lze jejich pohyb ovládat a také jim to přidá fyziku (bude na ně působit gravitace, atd.).

[2] Asset je jakýkoliv soubor, který se nachází ve složce projektu, Může se jednat o 2D obrázky, 3D modely, skripty obsahující kód, scény atd.

Player



Objekt samotného hráče je tvořen prostým 3D objektem – koulí. Má k sobě přiřazeny 3 skripty – `Player Controller.cs`, `Time Body.cs` a `Timer.cs`.

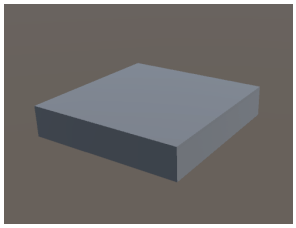
Třída `Player Controller` řeší hráčův pohyb, sbírání a počítání drahokamů a kolize s nepřátelskými objekty, po kterých restartuje level. Dále spouští příslušné akce, pokud hráč spadne z levelu.

Ve třídě `Time Body` se po 10 relativních časových jednotek zaznamenává hráčova pozice ve scéně ve `Vector3` souřadnicích (souřadnice na ose x, y a z). Pokud hráč spadne z levelu a dosáhne určité záporné výšky (y souřadnice), tak se zavolá akce, která vrací hráče v čase na jeho jednotlivé pozice, dokud se na dobu alespoň 0,3 vteřin neocitne zpátky na podlaze levelu. Při navracení hráče je čtyřikrát zrychlen čas.

Třída `Timer` definuje stopky, které měří čas, za jak dlouho hráč překonal daný level. Stopky se spouští až poté, co se hráč v levelu poprvé pohne.

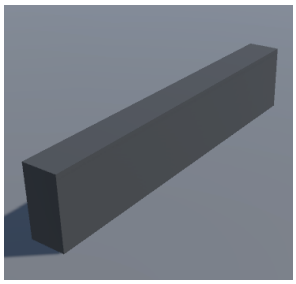
Objekt hráče může být ve scéně jen jeden.

Ground



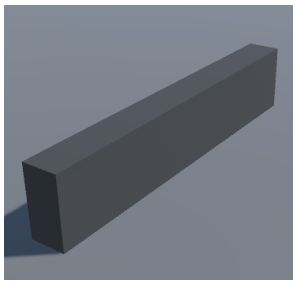
Ground (země) je statický objekt, po kterém se hráč pohybuje. Je vytvořena z prostého 3D objektu – krychle.

Wall



Wall (stěna) je statický objekt, který hráči brání v průchodu do určité oblasti. Je vytvořena z prostého 3D objektu – krychle.

Dynamic Wall

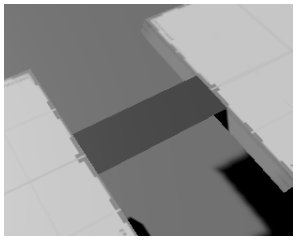


Dynamic Wall (dynamická stěna) je speciální stěna, která nemá dopředu dané rozměry. Podstata tohoto objektu se využije hlavně při tvoření levelu v editoru.

Vytváření Dynamic Wall má na starosti třída `CreateWalls`, která pracuje se dvěma souřadnicemi ze scény. První souřadnici získá, když hráč poprvé stiskne levé tlačítko a tím umístí začátek stěny. Umístěný počáteční objekt se poté roztahuje ve směru táhnutí myši. Po vypuštění stisknutého tlačítka se

určí druhá a tedy koncová souřadnice stěny, zde se ukončí její umisťování a stěna se mezi těmito dvěma body vytvoří. Tuto stěnu lze vytvářet pouze na objektu Ground. Více o stavění této stěny v sekci 5.4.

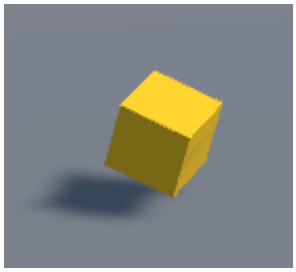
Bridge



Bridge (most) je objekt, který spojuje dva oddělené kusy podlah. Tento objekt je podobně jako Dynamic Wall tvořen dynamicky.

Vytváření Bridge objektu je velice podobné vytváření objektu Dynamic Wall. Hráč nejprve stiskne tlačítko myši, tím položí základ mostu na jednom kusu podlahy a poté myší táhne až ke druhému kusu podlahy, kde tlačítko vypustí v libovolné pozici a tím se, mezi těmito dvěma pozicemi, vytvoří most. Více o jeho stavění v sekci 5.4.

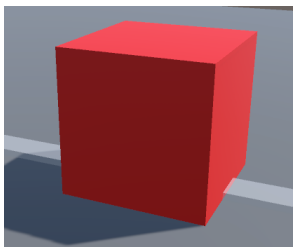
Pick Up



Pick Up's nebo také gem's (drahokamy) jsou objekty, které hráč sbírá. Pick Up je tvořen prostým 3D objektem – krychlí. Tento objekt neustále mění svoji rotaci a k tomu ještě levituje. Tím ihned přitáhne pozornost hráče. Má k sobě přiřazenou třídu `Rotator`, ve které se toto rotování řeší – `transform.Rotate("jak se má otáčet podle jednotlivých os")`.

Hráč musí posbírat všechny drahokamy v levelu, aby zvítězil. Sbírání probíhá tím způsobem, že pokud dojde ke kolizi mezi hráčem a drahokamem, aktivuje se takzvaný trigger[3], daná instance drahokamu ve scéně se deaktivuje a hráči se zvýší počet sebraných drahokamů. Poté co počítadlo dosáhne stejné hodnoty, jako je celkový počet drahokamů ve scéně, hra se pozastaví a aktivuje se Win Menu.

Enemy



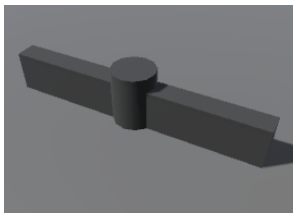
Enemy (nepřítel) je pro hráče nepřátelským objektem. Je tvořen prostým 3D objektem – krychlí. Nepřítel se pohybuje po určené dráze tam a zpět. Jeho pohyb řeší funkce `PingPong()`, která je součástí struktury `Mathf` z Unity enginu. Tato funkce vrací hodnotu, která se bude inkrementovat a dekrementovat mezi hodnotou 0 a hodnotou, která se této funkci předá jako parametr (v tomto případě vzdálenost dráhy, po které se bude nepřítel pohybovat). Tato funkce je volána ve třídě `Enemy Mover`, která k tomuto objektu náleží. Pokud dojde ke kolizi mezi hráčem a nepřítelem, aktivuje se trigger a dojde k restartování levelu. Nepřítel má čtyři vlastnosti, které hráč může modifikovat v editoru levelů:

- Speed (rychlost) – určuje rychlost, jakou se nepřítel pohybuje.

[3] https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html

- Distance (vzdálenost) – určuje délku dráhy, po které se bude pohybovat.
- Angle (úhel) – určuje směr, jakým se bude pohybovat. Tato hodnota se mění rotováním nepřítele podle osy Y.
- Phase (fáze) – určuje v jaké fázi dráhy nepřítel začne. Například se tím dá určit, jestli se bude nepřítel po dráze pohybovat zprava doleva nebo zleva doprava.

Enemy Spinner



Enemy Spinner (nepřátelské otáčedlo) je rotující objekt, který může hráče snadno shodit z levelu. Je vytvořen v nástroji ProBuilder, který je součástí Unity. Jedná se o nástroj určený pro 3D modelování a návrh úrovní. Je optimalizovaný pro vytváření jednoduché 3D geometrie, ale rovněž schopný detailního editování.[4]

K Enemy Spinneru náleží třída `EnemySpinnerMover`, ve které je řešeno jeho rotování (podobně jako u objektu Pick Up). Objekt má rovněž dvě vlastnosti, které může hráč modifikovat v editoru levelů:

- Speed (rychlost) – určuje rychlost, jakou se bude spinner otáčet.
- Direction (směr) – určuje jakým směrem se bude spinner otáčet.

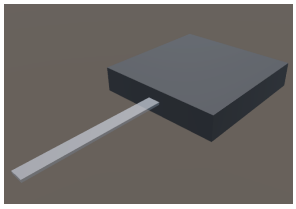
Jump Pad

Jump Pad (trampolína) je objekt, který hráče, když na něj najede, vymrští směrem vzhůru. K tomuto objektu náleží třída `JumpPad`, ve které je řešeno odrážení hráče a to tím způsobem, že pokud dojde ke kolizi mezi hráčem a trampolínou, tak na hráče začne působit určitá síla, která ho tlačí směrem nahoru (síla skoku). Sílu skoku může hráč měnit v editoru levelů (atribut Jump Pad Force).

Speed Boost

Speed Boost (zrychlovadlo) je objekt, který hráče, když na něj najede, vymrští směrem vpřed. K tomuto objektu náleží třída `SpeedBoost`, kde je řešeno zrychlení hráče a to tím způsobem, že pokud hráč na zrychlovadlo najede, začne na něj působit síla, která ho vymrští směrem vpřed. Tuto sílu může hráč měnit v editoru levelů (atribut Speed Boost Force).

Moving Platform



Moving Platform (pohybující-se platforma) je objekt, který se pohybuje sem a tam po určité dráze. Hráč se po něm může svézt. Platforma tedy může v podstatě nahradit most, dá se na ní překonat prázdné místo mezi dvěma oddělenými úseky

[4] https://unity3d.com/unity/features/worldbuilding/probuilder

objektů podlahy. Rovněž může fungovat jako překážka, která může hráče srazit z levelu (například pokud je na mostě).

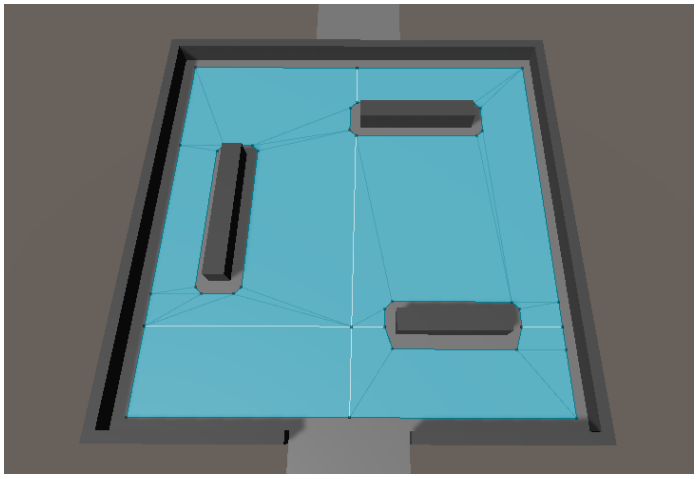
Náleží k ní třída `PlatformMover`, ve které je řešen její pohyb. Hráč může v editoru levelů měnit čtyři vlastnosti z této třídy:

- Speed (rychlost) – určuje jakou rychlosti se bud platforma pohybovat.
- Distance (vzdálenost) – určuje délku dráhy, po které se bude platforma pohybovat.
- Úhel A (Rotation – směr pohybu podel os X a Z) – určuje jakým směrem se bude platforma v rovině pohybovat.
- Úhel B (Direction – směr pohybu podle osy Y) – určuje jakým směrem se bude platforma pohybovat (nahoru nebo dolů).

Tag Game Enemy

Tag Game Enemy (pronásledující nepřítel) je speciální nepřítel, který se z důvodu složité implementace, neobjevuje v herním editoru a hráč tedy nemá možnost ho vložit do svého levelu. Tento nepřítel se vyskytuje pouze ve scéně Example Level 1 a v určité oblasti pronásleduje hráče, jako by se jednalo o hru na honěnou. Jedné se tedy o implementaci umělé inteligence (AI).

AI je zde implementována pomocí NavMesh, který je součástí Unity. Objekt nepřítele má k sobě přiřazenou komponentu zvanou Nav Mesh Agent, která mu umožní pohybovat se v oblasti vygenerovaného navmeshe. Dále se vymezí oblast, ve které se tento agent může pohybovat – vyberou se objekty po kterých se buď může pohybovat nebo jsou jeho překážkami a musí je obejít (zdi, podlahy, atd.) a označí se jako „Navigation Static“. V této oblasti se navmesh vygeneruje (viz obrázek 5.8). Ve třídě `Tag Game Enemy` potom už jen stačí určit, že cíl, který má agent pronásledovat je právě objekt hráče. Agent si pak neustále hledá nejkratší cestu ke hráči v závislosti na jeho pohybu a na překážkách, které mu stojí v cestě.

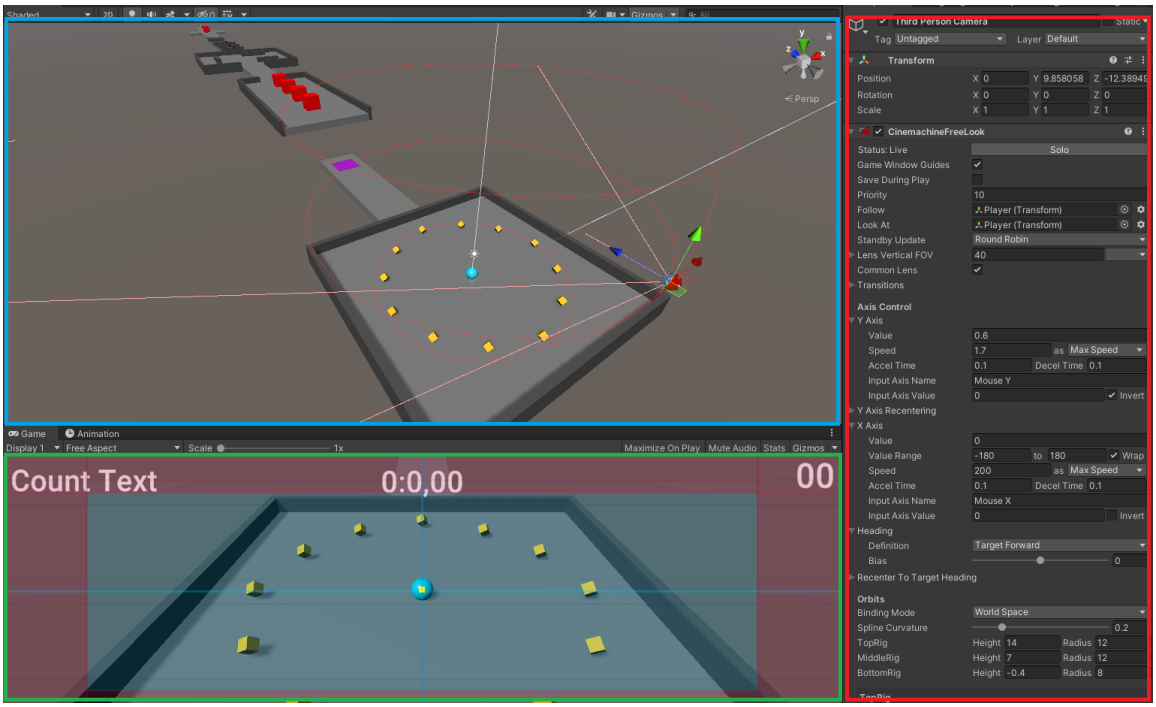


Obrázek 5.8: Screenshot vygenerovaného navmeshe. Světle modrá oblast je ta, po které se nepřítel (agent) může pohybovat. Překážky ve formě zdí jsou ohraničeny a agent se jim vyhýbá.

5.3 Herní část

Herní část je ta část hry, ve které hráč přímo ovládá kuličku a překonává jednotlivé levely. Tato část zahrnuje scény s ukázkovými levely a potom scénu, do které se načítají vybrané custom levely.

Pohyb kuličky se ovládá pomocí kláves W,A,S,D nebo pomocí šipek. Její otáčení se ovládá myší. Ve scénách se nachází kamera, která zajišťuje pohled z třetí osoby. Byl využit, v Unity volně dostupný, nástroj Cinemashine[5] (viz obrázek 5.9), který řeší všechny problémy spojené s fungováním kamery. Zajišťuje neustálý pohled na hráče, jeho následování, udržování si určitého odstupu podle vzdálenosti kamery od země, otáčení kamery spolu s hráčem atd. Cinemashine rovněž poskytuje tzv. „FreeLook“ (volný pohled) – kamera může volně obíhat kolem hráče, není pevně fixovaná na jednom místě.



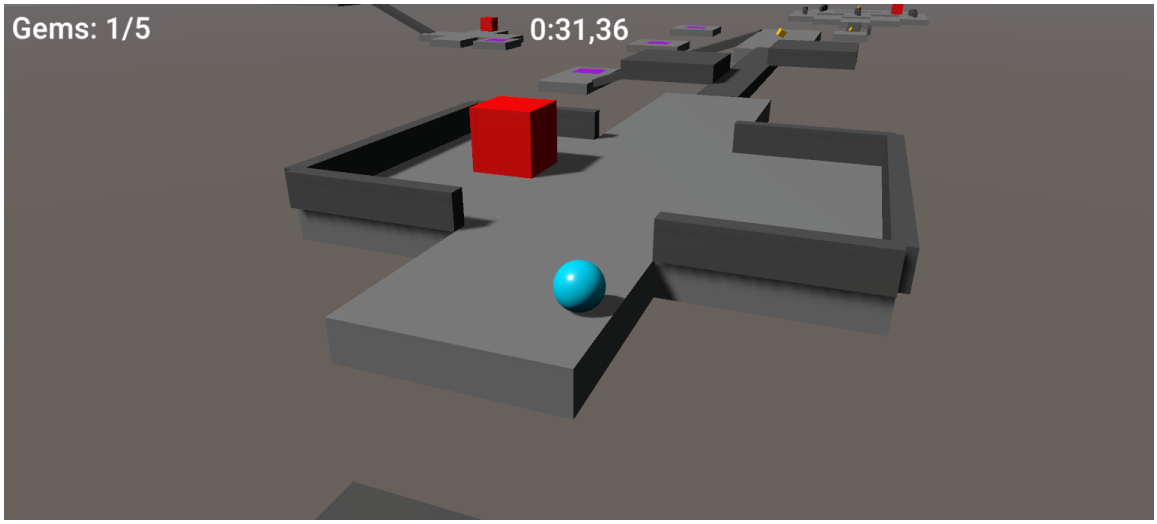
Obrázek 5.9: Znázornění nástroje Cinemashine, použitého pro implementaci kamery z pohledu třetí osoby. Na obrázku je možné vidět screenshot se samotného unity editoru. V modrém obdélníku je vidět pohled na scénu, kde je vyznačená kamera. Lze si všimnout tří oběžnice okolo hráče – znázorňují jak blízko něho kamera v bude při změnách její výšky a natočení vzhledem k zemi. V zeleném obdélníku je vidět pohled ze samotné hry, tedy jak kamera bude snímat hráče. A v červeném obdélníku napravo je vidět Unity inspektor, ve kterém se nastavují jednotlivé vlastnosti vybraného objektu. Je zde například vidět, že kamera má následovat a neustále zabírat objekt hráče (atributy Follow a Look At – přiřadí se k nim objekt hráče).

[5] https://unity.com/unity/features/editor/art-and-design/cinemachine

Example Levels

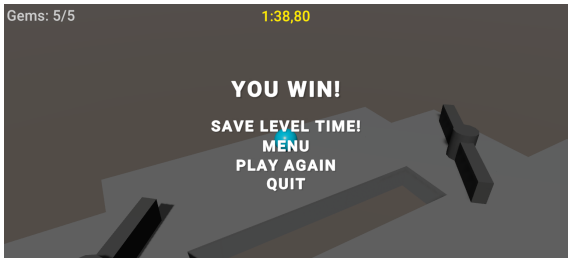
Example levely jsou ve hře celkem tři. Druhý a třetí jsou vytvořeny v herním editoru, který je součástí tohoto dema. První ukázkový level vznikl dříve, než byl editor implementován, ale nachází se v něm stejně herní objekty, které lze v editoru vytvořit.

V těchto třech scénách se kromě kamery popsané výše nachází také světlo (directional light) a canvas obsahující UI elementy, mezi které patří počítadlo sebraných drahokamů, stopky, PauseMenu, WinMenu a dialogové okno pro uložení času levelu. Na obrázku 5.10a je zachycena ukázka z levelu Example Level 3, na obrázku 5.10b je vidět PauseMenu (aktivované klávesou „Esc“) a na obrázku 5.10c WinMenu (aktivované sebráním posledního drahokamu).



(a) Screenshot z ukázkového levelu Example Level 3. Na obrázku lze vidět hráče v pohybu. Před ním se nachází objekt nepřítele. V dáli lze pozorovat objekty jako drahokam, trampolíny, pohyblivé platformy nebo spinnery. Co se týče UI prvků, tak jejich podkladem je canvas a samotné prvky jsou TMP-text elementy. V levém horním rohu se nachází počítadlo drahokamů (počet sebraných drahokamů/celkový počet drahokamů v levelu). Nahoře uprostřed se potom nacházejí stopky měřící čas překonání levelu.

(b) Screenshot PauseMenu z ukázkového levelu Example Level 3.



(c) Screenshot WinMenu z ukázkového levelu Example Level 3.

Obrázek 5.10: Screenshoty z ukázkového levelu Example Level 3.

Custom Levels

Custom levely se dynamicky načítají do scény Play Selected Custom Level z json souboru. Scéna tedy neobsahuje žádné herní objekty. Ty se do ní načítají až během hraní, podle zvoleného levelu. Více o načítání objektů do scény v podsekci 5.4.

5.4 Editor

Editor je stěžejní částí tohoto herního dema a jeho implementace zabrala nejvíce času. Při jeho tvorbě byl kladen důraz na jeho přívětivost z uživatelského hlediska. Editor má být jednoduchý, přehledný a efektivní při tvorbě levelů. Nejdůležitější prvky v editoru jsou prázdné objekty EditorManager a Mouse, ke kterým náleží třídy `EditorManagerScript` a `MouseScript`. Tyto třídy řeší v podstatě veškeré operace prováděné v editoru. Třída `EditorManagerScript` má na starosti zpracovávání dat (ukládání, načítání, atd.) a práci s UI elementy. Třída `MouseScript` zpracovává vstup myši a řeší jednotlivé operace prováděné nad herními objekty (vytváření, modifikování, mazání, přemisťování, atd.). Pozice myši ve scéně se určuje pomoci paprsku (Ray), který prochází kurzorem myši. Podrobněji budou tyto třídy rozebrány dále v této sekci.

Další důležitou třídou v editoru je `EditorObject`. Tato třída definuje a uchovává typy objektů, které mohou být vytvářeny a jejich vlastnosti (pozice, rotace, velikost, typ, atd.). Data jsou serializovatelná. Třída `LevelEditor` následně tato data ukládá do seznamu `editorObjects` – nachází se v něm informace o všech objektech umístěných v editoru, pro možnost jejich ukládání do souboru.

V editoru se nachází tzv. volná kamera – je možné s ní libovolně pohybovat a otáčet ji. Ovládání kamery řeší třída `CameraMover`, která odchytává vstup z klávesnice a myši a podle něj řídí kameru.

Command Pattern

Operace Create, Destroy, Move a Rotate, které lze v editoru provádět, jsou zapouzdřené pomocí Command Pattern.

Command Pattern (viz obrázek 5.11) zapouzdřuje všechny informace potřebné k provedení akce, jako objekt. Tyto informace zahrnují název metody, objekt kterému metoda náleží a hodnoty parametrů metody. Command Pattern je neodmyslitelně spojen s těmito čtyřmi výrazy – příkaz, příjemce, volající a klient. Objekt příkaz ví o příjemci a vyvolá jeho metodu. Hodnoty parametrů metody příjemce jsou uloženy v příkazu. Objekt příjemce je agregací rovněž uložen v příkazu. Příjemce poté provede požadovanou operaci, když je v příkazu volána metoda `execute()`. Objekt volajícího ví, jak provést příkaz, ale nic o daném příkazu neví, zná jen jeho rozhraní. Objekty volajícího, příkazu a příjemce jsou drženy objektem klienta, který rozhodne, které objekty příjemce jsou přiřazeny objektu příkazu a které příkazy jsou přiřazeny volajícímu. Klient rozhoduje, kdy se mají které příkazy provést. Aby příkaz provedl, předá objekt příkazu objektu příjemce. [4]

V tomto demu je Command Pattern implementován následovně. Je dáno rozhraní `ICommand`, ve kterém jsou definovány funkce `Execute`, která provede danou operaci a `Undo`, která vrátí zpět poslední provedenou operaci. Dále je implementována třída `CommandManager`, ve které je definovaná seznam (commandBuffer), kam se ukládají objekty jednotlivých příkazů, včetně parametrů, se kterými se volají. Ukládání probíhá po zavolání funkce `Execute()`. Je zde rovněž řešeno odebírání objektů příkazů z toho commandBufferu, pokud

je volána funkce `Undo()`. Poté jsou definovány čtyři třídy (konkrétní příkazy) – `CreateObjectCommand`, `DestroyObjectCommand`, `MoveObjectCommand`, `RotateObjectCommand` – které dědí z rozhraní `ICommand` a obstarávají zpracování operací Create, Destroy, Move a Rotate.

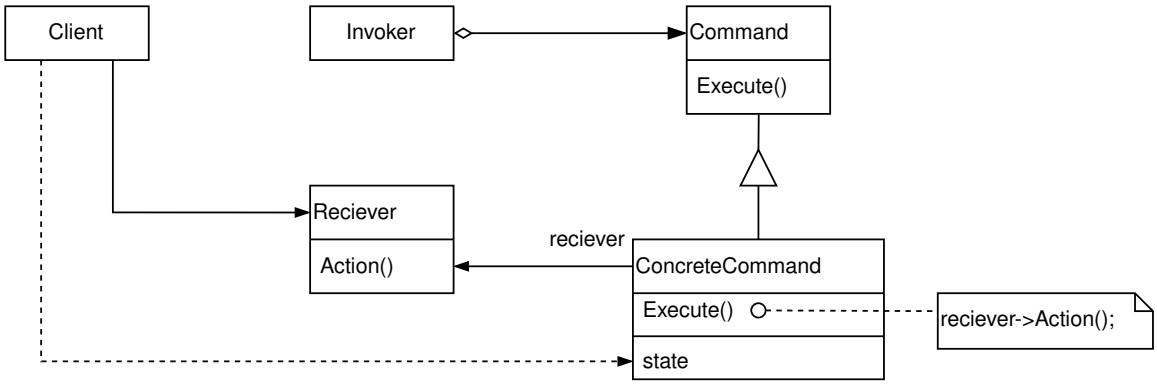
Volání příkazů probíhá ze třídy `MouseScript`. Pokud například, chce hráč ze scény odstranit nějaký objekt, je třeba mít zvolenou operaci Destroy a po kliknutí na daný objekt, se provedou následující příkazy:

```
1  ICommand destroyObject = new DestroyObjectCommand(hit.collider.gameObject, ms);
2  destroyObject.Execute();
3  CommandManager.Instance.AddCommand(destroyObject);
```

První příkaz vytvoří nový objekt příkazu. Ve druhém se volá funkce `Execute()` ze třídy `DestroyObjectCommand`, kde je definována. Zde se uloží instance objektu, který má být zničen a deaktivuje se pro případ, že by měl být později do scény znovu vrácen. Třetí příkaz uloží objekt příkazu do commandBufferu.

Pokud je následně v editoru stisknuta klávesová zkratka „CTRL+Z“ (která vrací zpět naposledy provedené operace), tak v commandBufferu najde poslední přidaný objekt příkazu, zavolá se pro něj funkce `Undo`, v tomto případě se znovu aktivuje objekt, který byl v předchozím kroku deaktivován a daný objekt příkazu je z commandBufferu odstraněn.

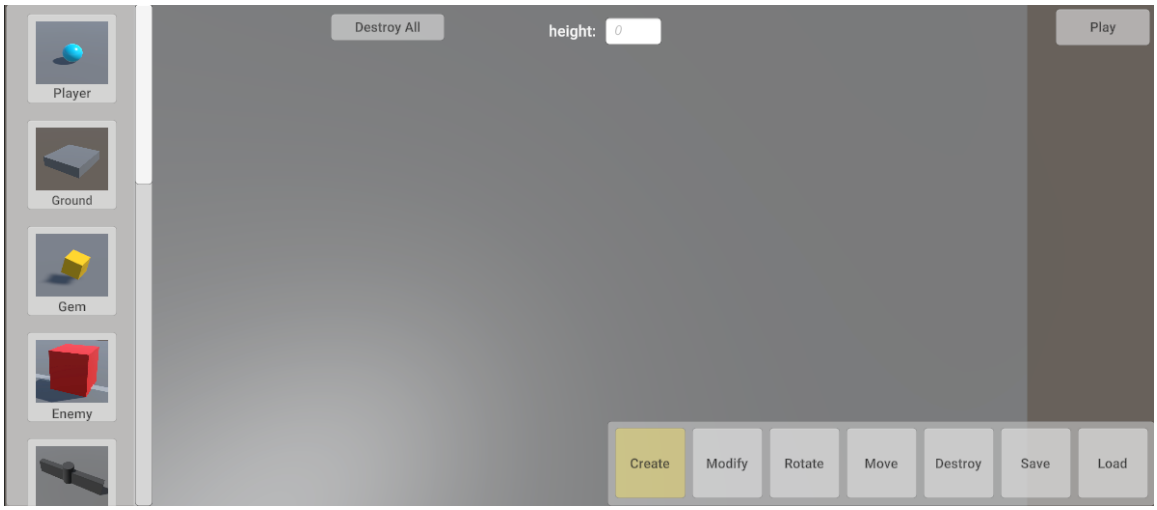
Obdobně tyto akce probíhají i při operacích Create, Move a Rotate.



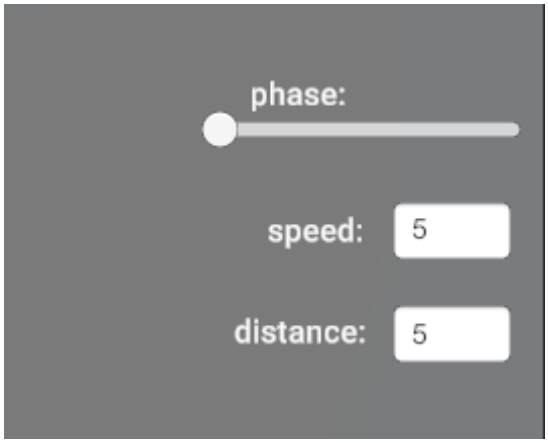
Obrázek 5.11: Struktura Command Pattern. Command (příkaz) – deklaruje rozhraní pro provedení operace. ConcreteCommand (konkrétní příkaz) – definuje vazbu mezi příjemcem a akcí, implementuje `Execute` zavoláním korespondující operace na příjemci. Client (klient) – vytvoří objekt ConcreteCommand a určí jeho příjemce. Invoker (volající) – požádá příkaz, aby provedl požadavek. Reciever (příjemce) – ví, jak provést operaci související s požadavkem. Obrázek převzatý z [4].

Uživatelské rozhraní

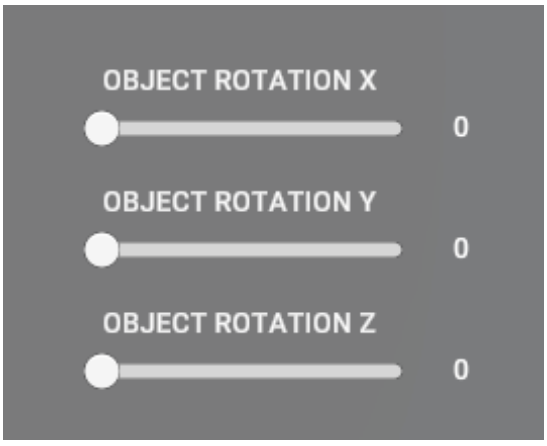
UI editoru je tvořeno canvasem, tlačítky, posuvníky, vstupními textovými poli a scrolovacím oknem (viz obrázek 5.12).



Obrázek 5.12: Screenshot z editoru levelů. Na levé straně se nachází UI element ScrollView, který obsahuje tlačítka zastupující jednotlivé objekty – jedná se o nabídku herních objektů, které je možné umístit do scény. Každé tlačítko obsahuje textový element a obrázek. Dále ve spodní části na pravé straně je nabídka operací, které je možná v editoru provádět – vytvořit objekt, modifikovat objekt 5.13a, změnit rotaci 5.13b objektu, zničit objekt, uložit level 5.13c, načíst level 5.13d. Tlačítko vybrané operace je žlutě podbarveno. V horní části se nachází tlačítko „Destroy All“, které zničí všechny objekty ve scéně najednou, po kliknutí na něj ještě následuje dialogové okno, kde hráč potvrdí, že chce operaci skutečně provést. Napravo od něj se nachází element InputField, do kterého se zadává výška ve které budou umisťovány objekty. V pravém horním rohu se poté nachází tlačítko, které editor přepíná mezi režimy Edit a Play.



(a) Screenshot UI prvku, který se aktivuje při operaci Modify – upravují se zde vlastnosti vybraného objektu. Na obrázku je zobrazen případ modifikace Enemy objektu. Skládá se z elementu Slider a dvou InputFields. Pomocí posuvníku se určuje fáze, ve které nepřítel začne svůj pohyb. Do prvního vstupního pole se zadává rychlost, jeho pohybu a do druhého délka dráhy, po které se bude pohybovat.



(b) Screenshot UI prvku, který se aktivuje při operaci Rotate. Mění se zde rotace vybraného objektu podle jednotlivých os. Prvek se skládá ze tří Slider elementů. Napravo od posuvníků se nachází textová pole, kde je rotace uvedena číslem ve stupních.

(c) Screenshot dialogového okna, které se aktivuje po stisknutí tlačítka Save. Skládá se ze dvou tlačítek a jednoho vstupního pole. Do pole se zadá název, pod kterým má být level uložen a stisknutím tlačítka uložit se uloží. Tlačítkem v pravém horním rohu se okno zavře.

(d) Screenshot dialogového okna, které se aktivuje po stisknutí tlačítka Load. Skládá se ze dvou tlačítek a jednoho vstupního pole. Do pole se zadá název levelu, který má být načten a stisknutím tlačítka načíst se načte. Tlačítkem v pravém horním rohu se okno zavře.

Obrázek 5.13: Screenshoty UI prvků z editoru levelů.

Vytváření objektů

Pří spuštění editoru se v něm nenacházejí žádné herní objekty (viz obrázek 5.12), hráč je musí všechny vytvořit. Vytvoření objektů probíhá tak, že je potřeba mít vybranou operaci Create a poté si zvolit objekt, který má být umístěn do scény, z nabídky objektů. Po kliknutí na tlačítko konkrétního objektu, se kolem kurzoru myši objeví mesh daného objektu, který má buď zelenou barvu, pokud lze objekt na dané místo vytvořit nebo červenou barvu, pokud jej vytvořit nelze (viz obrázek 5.14). Tuhle funkcionalitu řeší třída `EditorManagerScript`, ze které se vždy zavolá příslušná funkce k danému objektu (například `ChooseEnemy()`), ve které se z prfabu tohoto objektu získá jeho mesh. Rovněž se zde objektu Mouse přídá komponenta BoxCollider ve velikosti daného objektu, aby bylo možné testovat, jestli nenastala při umisťování kolize s jiným objektem. Obarvení meshe se již řeší ve třídě `MouseScript`, kde se právě kontroluje, jestli při umisťování objekt nekoliduje s jiným. Pokud je s vybraným objektem kliknuto na místo, kam jej lze umístit, tak se ve třídě `MouseScript` vytvoří nový objekt příkazu se všemi potřebnými parametry a zavolá se funkce `Execute()` ze třídy `CreateObjectCommand`.

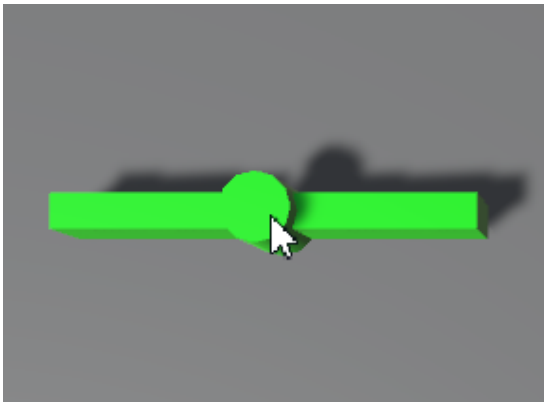
V této třídě se nejdříve zjistí, jaký objekt má být vytvořen (podle toho, který byl zvolen), poté se vytvoří instance tohoto objektu z jeho prefabu. Následně se k němu přiřadí třída `EditorObject` a uloží se do ní všechna potřebná data o tomto objektu (pozice, rotace, typ, atd.). O pohybujících se objektech (enemy, spinner, gem, atd.) se uchovávají také informace o jejich pohybu (rychlost, směr, atd.). Tímto způsobem (jedním kliknutím) jsou vytvářeny všechny objekty až na dvě výjimky, těmi jsou DynamicWall a Bridge. Ty jsou vytvářeny ve třídách `CreateWalls` a `CreateBridge`.

Ve třídě `CreateObjectCommand` se rovněž nachází funkce `Undo()`, která je zavolána v případě, že byla stisknuta klávesová zkratka „CTRL+Z“. Tato funkce zničí právě vytvořený objekt (vrátí zpět jeho vytvoření).

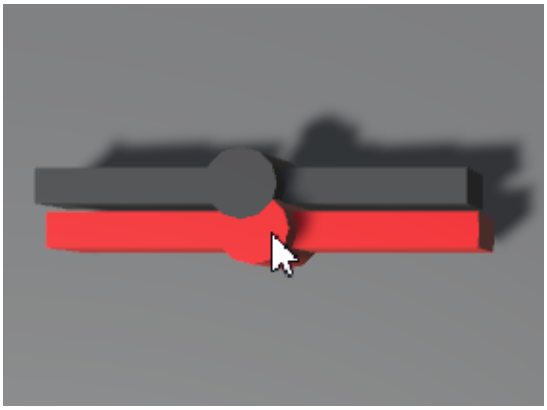
Objekty jsou vytvářeny na pozici kurzoru myši. Při vytváření lze zapnout tzv. snapping stisknutím klávesové zkratky „CTRL+X“. Tím se pozice objektů zaokrouhluje vždy na celá čísla a je tak snazší umisťovat objekty rovnoměrně nebo je na sebe navazovat (např. při umisťování objektů Ground).

- DynamicWall – vytváří se následovně: nejdříve je potřeba stisknout levé tlačítko myši na místě, kde má zeď začínat, tím se položí její základ (vytvoří se instance z wallPrefab, což je jeden sloupek). Dále se táhnutím myši do určitého směru určuje její velikost, tím se mění rozměry položeného sloupku. Vypuštěním tlačítka se zeď vytvoří. Zeď je nutné vytvářet na objektu podlahy (viz obrázek 5.15).
- Bridge – vytvoří se velice podobně, jako DynamicWall. Nejdříve se stisknutím levého tlačítka myši položí základ mostu, který se poté roztahuje táhnutím myši do určitého směru. Po vypuštění tlačítka se most vytvoří. Most je nutné vytvářet mezi dvěma objekty podlahy (viz obrázek 5.16).

Objekty Enemy a MovingPlatform obsahují tzv. DystanceIndicator, který naznačuje směr a délku dráhy, po které se pohybují (viz obrázek 5.17). Tento indikátor je viditelný pouze v editoru v režimu edit. Při hrání nebo zkoušení úrovní tento indikátor viditelný není.



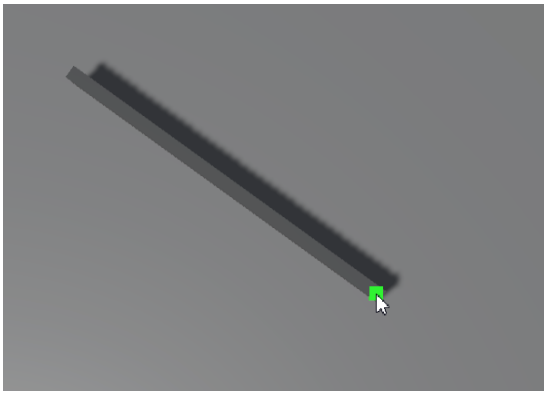
(a) Screenshot ukazující vytváření objektů a stav, kdy na dané místo lze vytvořit objekt.



(b) Screenshot ukazující vytváření objektů a stav, kdy na dané místo nelze vytvořit objekt, protože se překrývá s jiným objektem.

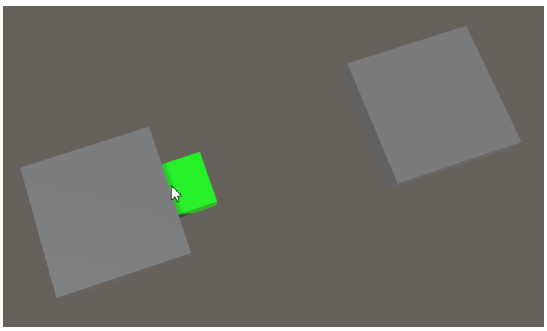
Obrázek 5.14: Screenshoty ukazující umisťování herního objektu do levelu.

(a) Položení základu zdi.

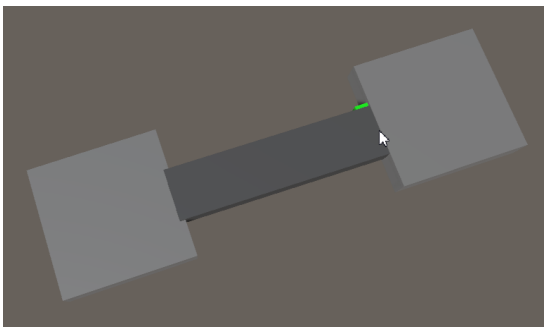


(b) Natažení zdi pomocí táhnutí myši.

Obrázek 5.15: Screenshoty ukazující vytváření objektu DynamicWall.

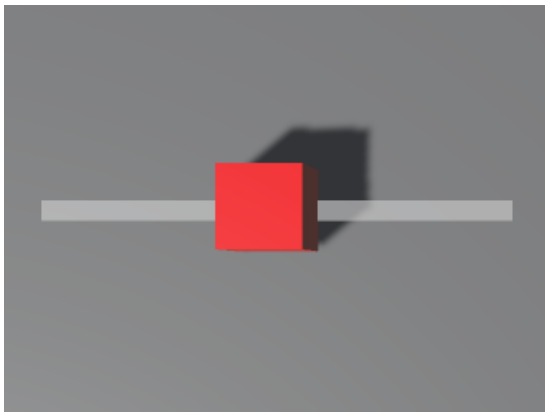


(a) Položení základu mostu.



(b) Natažení mostu pomocí táhnutí myši.

Obrázek 5.16: Screenshoty ukazující vytváření objektu Bridge.



(a) DistanceIndicator znázorňující směr a vzdálenost pohybu objektu Enemy.

(b) DistanceIndicator znázorňující směr a vzdálenost pohybu objektu MovingPlatform.

Obrázek 5.17: Screenshoty ukazující DistanceIndicator u objektů Enemy a MovingPlatform.

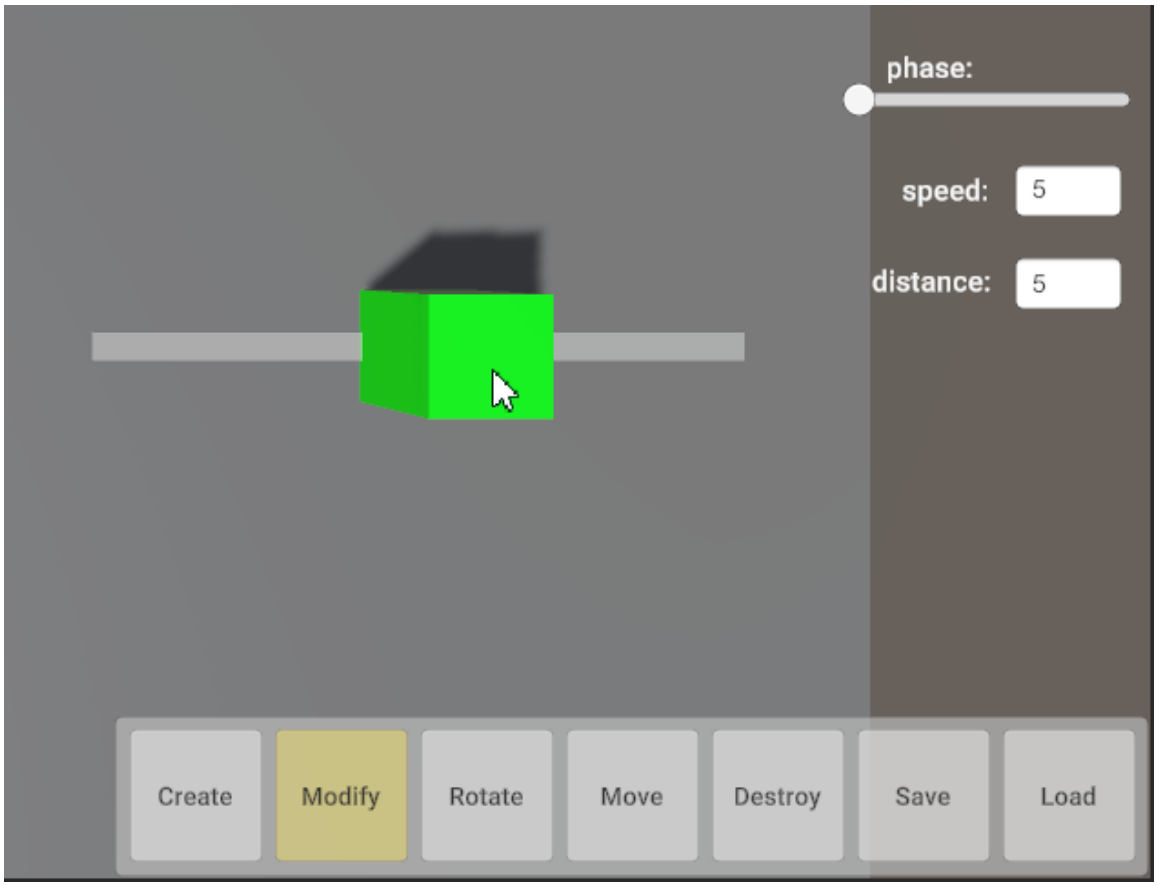
Modifikace objektů

Vytvořené objekty je možné v editoru rovněž modifikovat – lze měnit některé jejich vlastnosti (záleží na objektu). Modifikace se provádí pomocí operace Modify. Pokud je ve scéně umístěný nějaký objekt (např. Enemy), je vybraná operace Modify a na tento objekt se klikne, tak změní barvu na zelenou (aby bylo poznat, který objekt je vybraný) a aktivuje se nabídka pro modifikaci (viz obrázek 5.18).

Ve třídě `MouseScript` se nejdříve zjistí, jaký objekt byl pro modifikaci vybrán (ne všechny objekty lze modifikovat), poté se aktivuje a inicializuje nabídka modifikace pro daný objekt (zobrazí se v ní aktuální hodnoty vlastností, které jsou vytaženy ze samotného objektu). Pokud dojde ke změně nějaké vlastnosti, zavolá se (v případě objektu Enemy) funkce `ModifyEnemy()`, ve které se všechny nové hodnoty vlastností uloží do třídy `EditorObject` náležící k danému objektu.

Jednotlivé vlastnosti jsou napojeny na příslušnou třídu, která daný objekt řídí, v tomto případě se jedná o `EnemyMover`.

Změna vzdálenosti u objektu Enemy se okamžitě projevuje na jeho DistanceIndicátoru, který mění svoji velikost, aby vždy přesně znázorňoval odkud kam se bude nepřítel pohybovat.



Obrázek 5.18: Screenshot ukazující modifikaci objektu Enemy. Je vidět, že je zvolená operace Modify a je vybrán daný objekt nepřítele. Vpravo nahoře je nabídka, co se dá na tomto objektu modifikovat. V tomto případě se jedná o fázi, rychlost a vzdálenost pohybu.

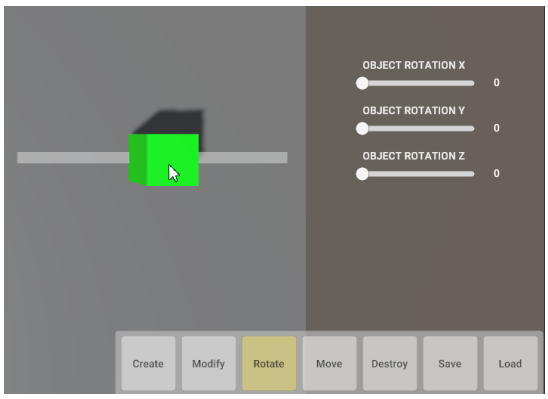
Rotování objektů

Změna rotace objektů se provádí pomocí operace Rotate. Je nutné mít vybranou tuto operaci, dále musí být ve scéně umístěn nějaký objekt. Pokud je na tento objekt kliknuto, objekt změní barvu na zelenou, podobně jako při operaci Modify a otevře se nabídka pro změnu jeho rotace (viz obrázek 5.19).

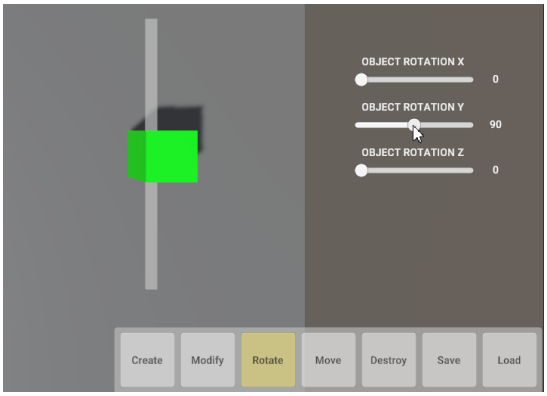
Ve třídě `MouseScript` se aktivuje a inicializuje nabídka změny rotace pro daný objekt (zobrazí se v ní aktuální hodnoty rotace, které jsou vytaženy ze samotného objektu). Dále se vytvoří nový objekt příkazu, zavolá se pro něj funkce `Execute()` a uloží do commandBufferu. Ve třídě `RotateObjectCommand` se uloží do pomocných proměnných aktuální hodnoty rotace objektu, aby se k nim bylo možno v případě volání funkce `Undo()` vrátit.

Pokud dojde ke změně hodnot rotace, zavolá se funkce `RotationValueChange()` ze třídy `EditorManagerScript`. Zde se získají aktuální hodnoty rotace z jednotlivých sliderů, zrotuje se podle nich daný objekt a tyto hodnoty se uloží do třídy `EditorObject`.

Při změnách hodnot pomocí slideru, ať už při rotaci nebo při modifikaci, lze zapnout tzv. stepping stisknutím klávesové zkratky „CTRL+C“. Tím se bude hodnota na slideru měnit vždy po 30°.



(a) Screenshot ukazující změny rotace objektu Enemy. Je vidět, že je zvolená operace Rotate a je vybrán daný objekt nepřítele. Vpravo nahoře je nabídka tvořená ze sliderů, každý pro jednu osu.



(b) Screenshot ukazující provedené změny rotace podle osy Y o 90°. Touto změnou rotace se určuje směr pohybu nepřítele.

Obrázek 5.19: Screenshoty ukazující změnu rotace u objektu Enemy.

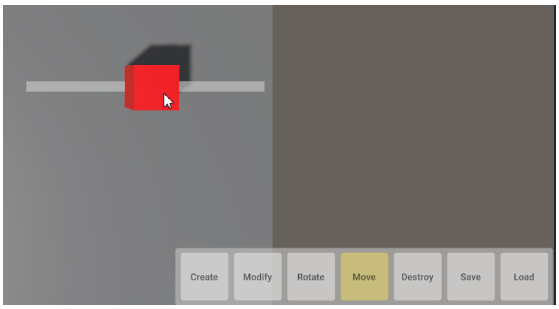
Přemisťování objektů

Přemístit již vytvořený objekt lze pomocí operace Move. Po kliknutí na nějaký objekt ve scéně, tento objekt zmizí, kolem kurzoru myši se objeví jeho mesh a objekt je možné umístit na nové místo (viz obrázek 5.20). Tato operace v podstatě funguje jako operace Destroy a Create dohromady.

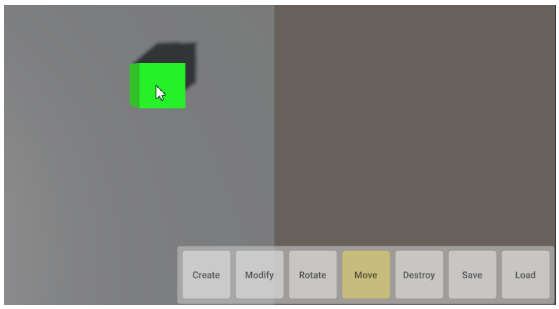
Nejdříve se ve třídě `MouseScript` vytvoří nový objekt příkazu, kterému se přemisťovaný objekt předá jako parametr. Dále se pro tento příkaz zavolá funkce `Execute()` a příkaz se uloží do commandBufferu. Ve třídě `MoveObjectCommand` se nejprve vytvoří kopie přemisťovaného objektu a deaktivuje se. Poté se podle daného objektu zavolá příslušná funkce ze třídy `EditorManagerScript` (například `ChooseEnemy()`), která kolem myši vytvoří mesh

tohoto objektu. Původní objekt je zničen a umisťuje se nová instance stejného objektu. Editor se chová jako při operaci Create.

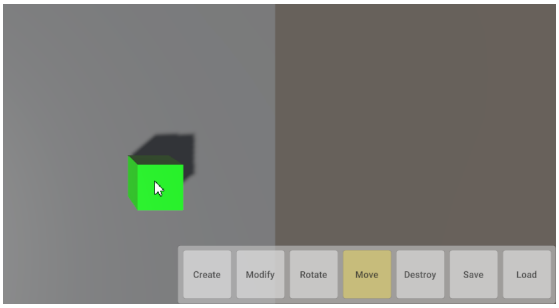
Při zavolání funkce `Undo()`, stisknutím klávesové zkratky „CTRL+Z“, se nový objekt zničí a aktivuje se kopie původního objektu – tím se objekt vrátí na své původní místo.



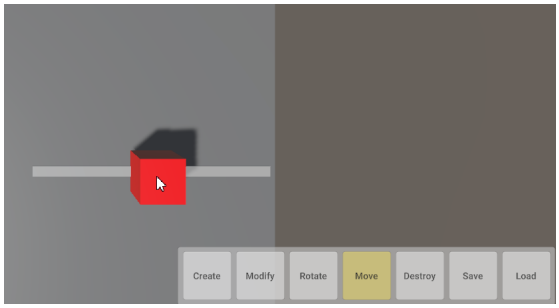
(a) Původní pozice objektu Enemy.



(b) Kliknutí na objekt Enemy.



(c) Přemístění objektu Enemy.



(d) Nová pozice objektu Enemy.

Obrázek 5.20: Screenshoty ukazující přemístění objektu Enemy.

Ničení objektů

Ničení objektů se provádí pomocí operace Destroy. Objekt na který se klikne, při této operaci, se zničí.

Ve třídě `MouseScript` se nejdříve vytvoří nový objekt příkazu, kterému se ničený objekt předá jako parametr. Dále se pro tento příkaz zavolá funkce `Execute()` a příkaz se uloží do commandBufferu. Ve třídě `DestroyObjectCommand` se ničený objekt zkopíruje do nové instance. Tato kopie se deaktivuje a objekt se zničí.

Při zavolání funkce `Undo()`, stisknutím klávesové zkratky „CTRL+Z“, se aktivuje kopie objektu – tím se objekt znovu objeví ve scéně.

Editor rovněž umožňuje odstranit všechny objekty najednou pomocí operace Destroy All. Po kliknutí na toto tlačítko se otevře dialogové okno, kde je nutné potvrdit, že si hráč skutečně přeje odstranit všechny objekty ze scény. Po potvrzení této operace se zavolá funkce `DestroyAll()` ze třídy `EditorManagerScript`. V této funkci se nejprve všechny hráčem umístěné objekty ve scéně uloží do pole. Tyto objekty jsou posléze jeden po druhém ze scény odstraněny.

Ukládání a načítání dat

- Ukládání dat

  Level vytvořený v editoru lze uložit pomocí operace Save. Je-li kliknuto na tlačítko Save, otevře se okno, které bude požadovat název, pod jakým má být level uložen.

  Následně se zavolá funkce `SaveLevel()` ze třídy `EditorManagerScript`. Zde se nejdříve uloží všechny hráčem vytvořené objekty do pole. Data, která obsahuje každý tento objekt (pozice, velikost, rotace, atd.) se uloží do třídy `LevelEditor` jako jeden objekt (level). Tento objekt level se dále převede do formátu json:

  ```
  1        string jsonLevel = JsonUtility.ToJson(level);
  ```

  Poté se nastaví cesta ke složce, kam budou levely ukládány – všechny levely budou uloženy do složky LevelData, která bude umístěna ve složce aplikace. K hráčem zadanému názvu levelu se připojí koncovka „.json“. Pokud nebyl zadán žádný název, level se uloží pod výchozím názvem „`new_level.json`“. Zkontroluje se, jestli složka LevelData existuje, pokud ne, vytvoří se. Dále se nastaví cesta k samotnému souboru s levelem – zkombinuje se cesta ke složce LevelData a název levelu. Pokud již existuje soubor se stejným názvem, tak se smaže a vytvoří znovu s novými daty. Do daného souboru se poté uloží data o objektech:

  ```
  1        File.WriteAllText("cesta k souboru", jsonLevel);
  ```

  Na obrazovku se nakonec vypíše zpráva, že byl level úspěšně uložen (ve scéně se nachází textový element, který ale nenese žádný text, tato zpráva se stane textem tohoto elementu a po třech vteřinách se smaže).

- Načítání dat

  V level editoru lze rovněž načíst dříve vytvořený level. Po kliknutí na tlačítko Load v editor se otevře okno, které bude požadovat název levelu, který má být načten.

  Následně se zavolá funkce `LoadLevel` ze třídy `EditorManagerScript`. Zde se vezme cesta ke složce, ve které jsou levely uloženy a zkombinuje se s názvem levelu, který byl zadán hráčem. Pokud žádný název zadán nebyl, použije se výchozí hodnota „`new_level.json`“. Takto vznikne cesta k souboru, ze kterého se mají načítat data. Pokud tato cesta existuje (existuje level s daným názvem), tak se nejprve odstraní všechny objekty, které se aktuálně vyskytují ve scéně a následně se obsah tohoto souboru, který je ve formátu json uloží jako řetězec do proměnné:

  ```
  1        string jsonLevel = File.ReadAllText("cesta k souboru");
  ```

  ... a vytvoří se třída `LevelEditor` level, do které se uloží data o jednotlivých objektech obsažené v tomto řetězci:

  ```
  1        level = JsonUtility.FromJson<LevelEditor>(jsonLevel);
  ```

  Dále se zavolá funkce `CreateFromFile()`, které se třída level předá jako parametr. Následně se prochází jednotlivé objekty, které tato třída obsahuje a vkládají se do scény, podobně jako při jejich vytváření operací Create. Vytvoří se instance daného objektu a předají se jí všechny uložená data (pozice, rotace, rychlost, atd.). Tato data se poté opět ukládají do třídy `EditorObjects`. Po umístění všech objektů do scény se vypíše zpráva, stejným způsobem jako u ukládání, že načtení proběhlo úspěšně.

Pokud level s daným názvem neexistuje, je o tom hráč rovněž informován zprávou, která se mu zobrazí.

Tímto způsobem nejsou levely načítány jen do level editoru, ale i když se načítá level do scény PlaySelectedCustom level, aby si ho hráč mohl zahrát.

Přepínání mezi režimy Edit a Play

Výchozím režimem editoru je režim Edit. V tomto režimu mají všechny objekty deaktivované jejich vlastnosti (žádný z nich se nehýbe, všechny třídy, které objekty ovládají jsou deaktivované). V pravém horním rohu scény nachází tlačítko Play, které přepne editor do režimu Play (viz obrázek 5.12).

Kliknutím na tohle tlačítko se zavolá funkce `ChoosePlay()`, která je definovaná ve třídě `EditorManagerScript`. Zde se nejdříve level uloží pod výchozím názvem „new_level.json“ (viz podsekce 5.4). Následně se deaktivují všechny UI elementy týkající se editoru. Dále se deaktivuje volná kamera editoru a aktivuje se kamera pohledu z třetí osoby (stejná jako při samotném hraní example nebo custom levelů). Zavolá se funkce `InitializeObjects()`. Zde se postupně projdou všechny objekty nacházející se ve scéně a provedou se u nich potřebné akce, aby mohly fungovat. U objektu Player se například aktivuje třída `PlayerController`, aby se dal ovládat a nastaví se jako objekt, na který se má dívat a který má následovat kamera. U objektu nepřítele se aktivuje třída `EnemyMover`, aby se začal pohybovat a deaktivuje se u něj DistanceIndicator. Takto se postupně inicializují všechny objekty.

Tento režim má funkci vyzkoušet si daný level, jestli v něm funguje vše, jak má, jestli jej lze překonat, atd. Hra se v něm chová stejným způsobem, jako při hraní example nebo custom levelů – hráč ovládá objekt Player, může sbírat drahokamy, může ho chytit objekt nepřítele, atd. Nenachází se zde ale stopky, ani počítadlo drahokamů, není zde tedy možno level „vyhrát“.

V tomto režimu se v pravém horním rohu nachází tlačítko Edit (na stejném místě, jako se nachází tlačítko Play v režimu Edit). Po kliknutí na toto tlačítko se zavolá funkce `ChooseEdit()` ze třídy `EditorManagerScript`. Zde je volána funkce `LoadLevel()` s výchozím názvem levelu – „new_level.json“. Všechny objekty jsou ze scény odstraněny a vytvoří se na jejich původních místech (viz podsekce 5.4). Opět se aktivují všechny UI elementy náležící editoru, včetně volné kamery a je možné pokračovat ve vytváření levelu.

Kapitola 6

Závěr

Cílem této práce bylo nastudovat si herní engine Unity a techniku pro tvorbu her, navrhnout a implementovat herní demo, včetně několika ukázkových levelů. Mým záměrem bylo vytvořit hru, ve které hráč bude překonávat překážky a k dispozici bude mít editor, kde si bude moci vytvořit vlastní levely.

Vývoj dema začal vytvářením jednotlivých herních objektů (překážek). Funkčnost těchto objektů jsem si zkoušel na testovací scéně, ze které nakonec vznikl první ukázkový level. Po vytvoření několika funkčních překážek jsem přešel na tvorbu editoru levelů. Tvorba tohoto editoru mi zabrala nejvíce času (zhruba půl roku). Když byl editor již v použitelném stavu, tedy když se v něm daly v celku pohodlným způsobem vytvářet plnohodnotné levely, vytvořil jsem v něm další dva ukázkové levely. Demo se nacházelo ve stavu, kdy bylo vytvořeno několik scén, které mezi sebou nebyly nijak propojeny. A tak přišlo na řadu vytvořit uživatelské rozhraní, které by jednotlivé scény propojilo. Vytvořil jsem tedy Několik menu, mezi kterými se dá procházet a demo již působilo jako celistvá hra, kterou si mohl kdokoliv vyzkoušet.

Práce mi dala mnoho nových zkušeností týkajících se tvorby her v enginu Unity a také programování v jazyce C#. Naučil jsem se spoustu technik potřebných pro tvorbu her a vyzkoušel jsem si práci na opravdu rozsáhlém projektu. I přes několik nepříznivých okamžiků mě vývoj dema velice bavil a přivedl mě k myšlence, že bych se v tomto oboru chtěl živit.

Záměr této práce byl splněn – je vytvořeno plně funkční herní demo s několika ukázkovými levely a editorem, který umožňuje tvorbu neomezeného počtu dalších levelů. Demo ovšem nabízí mnoho možností, jak se na něm dá dále pokračovat. Do budoucna bych chtěl přidat daleko více rozmanitějších překážek a vytvořit další ukázkové levely. Demo by rovněž potřebovalo hezký vizuální kabát, tedy aby všechny objekty nebyly tvořeny jen základními geometrickými tvary s určitými barvami. Určitě bych tedy chtěl demo vylepšit po grafické stránce. Dále bych chtěl přidat rozumný způsob, jakým mezi sebou hráči mohou sdílet jimi vytvořené custom levely, tak aby si je mohl kdokoliv zahrát. Věřím, že když se budu vývoji věnovat i nadále, případně kdyby se ke mně přidalo více lidí, tak z tohoto dema vznikne velmi povedená plnohodnotná hra.

Literatura

[1] Amazon. *Fully customizable game engine – Amazon Lumberyard – Amazon Web Services* [online]. 2021 [cit. 2021-04-09]. Dostupné z: https://aws.amazon.com/lumberyard/.

[2] CRYENGINE. *CRYENGINE | The complete solution for next generation game development by Crytek* [online]. 2021 [cit. 2021-04-09]. Dostupné z: https://www.cryengine.com/.

[3] Felicia, P. *Unity From Zero to Proficiency (Foundations): A step-by-step guide to creating your first game, 1.* 4. vyd. Vydáno nezávisle, 2019. ISBN 9781518699894.

[4] Gamma, E., Helm, R., Johnson, R. a Vlissides, J. *Design Patterns : Elements of Reusable Object-Oriented Software.* 1. vyd. Pearson Education, 1994. ISBN 0201633612.

[5] Godot Engine. *Godot Engine – Free and open source 2D and 3D game engine* [online]. 2021 [cit. 2021-04-09]. Dostupné z: https://godotengine.org/.

[6] Gregory, J. *Game Engine Architecture, Third Edition.* 3. vyd. Taylor & Francis Ltd, 2018. ISBN 9781138035454.

[7] Hocking, J. *Unity in Action, Second Edition.* 2. vyd. Manning Publications, 2018. ISBN 1617294969.

[8] Unity Technologies. *2D 3D Game Creator & Editor | Augmented / Virtual Reality Software | Game Engine | Unity* [online]. 2021 [cit. 2021-04-09]. Dostupné z: https://unity.com/products/unity-platform.

[9] Unity Technologies. *Unity - Manual: TextMeshPro* [online]. 2021 [cit. 2021-04-11]. Dostupné z: https://docs.unity3d.com/Manual/com.unity.textmeshpro.html.

[10] Unity Technologies. *Unity - Scripting API: PlayerPrefs* [online]. 2021 [cit. 2021-04-11]. Dostupné z: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html.

[11] Unreal Engine. *The most powerful real-time 3D creation platform – Unreal Engine* [online]. 2021 [cit. 2021-04-09]. Dostupné z: https://www.unrealengine.com.