FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL



Bakalářská práce

Tvorba 2D platformer hry ve zvoleném herním enginu

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Jan Smutný**
Vedoucí práce: Ing. Lukáš Zedek, Ph.D.
Ústav nových technologií a aplikované informatiky

Konzultant práce: Ing. Vratislav Žabka, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2025

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL



Zadání bakalářské práce

Tvorba 2D platformer hry ve zvoleném herním enginu

Jméno a příjmení:	**Jan Smutný**
Osobní číslo:	M22000225
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2024/2025

Zásady pro vypracování:

1. Proveďte rešerši dostupných herních enginů využívaných ke tvorbě 2D her včetně jejich vzájemného srovnání.
2. Vytvořte příběh pro hru a navrhněte herní mechaniky vhodné pro žánr ”platformer”.
3. Implementujte zvolené herní mechaniky ve vybraném herním enginu.
4. Vyhodnoťte výpočetní náročnost hry a navrhněte způsob jejího snížení.
5. Vývoj hry zdokumentujte.
6. Otestujte funkčnost hry na vybrané skupině uživatelů. Zhodnoťte funkčnost a opravte chyby.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] HARDMAN, Casey, 2023. Game Programming with Unity and C#: A Complete Beginner's Guide. 2. vydání. Apress. ISBN 978-1484256558.

[2] HOLAN, Tomáš, 2020. Unity: první seznámení s tvorbou počítačových her. Praha: CZ.NIC, z.s.p.o., 172 s. CZ.NIC. ISBN 978-80-88168-57-7.

[3] SCHELL, Jesse, 2010. The Art of Game Design. 2010. 1. vydání: Taylor & Francis. ISBN 8535241981.

Vedoucí práce:	Ing. Lukáš Zedek, Ph.D. Ústav nových technologií a aplikované informatiky
Konzultant práce:	Ing. Vratislav Žabka, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	15. října 2024
Předpokládaný termín odevzdání:	9. května 2025

L.S.

doc. Ing. Josef Černohorský, Ph.D.
děkan

doc. Ing. Josef Chaloupka, Ph.D.
garant studijního programu

V Liberci dne 18. října 2024

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

4. května 2025 Jan Smutný

Tvorba 2D platformer hry ve zvoleném herním enginu

Abstrakt

Tato práce se zabývá tvorbou 2D platformer hry ve zvoleném herním enginu. Na úvod byla provedena rešerše současně dostupných herních enginů určených pro vývoj 2D her, včetně jejich srovnání z hlediska funkcionality, výkonu a uživatelské přívětivosti. Na základě této analýzy byl zvolen vhodný engine pro implementaci projektu. Dále byl navržen herní příběh a specifikovány herní mechaniky charakteristické pro žánr platformer. Tyto mechaniky byly následně implementovány a integrovány do funkčního herního prototypu. Součástí práce je také analýza výpočetní náročnosti vytvořené hry a návrh optimalizačních opatření pro její snížení. Celý vývoj byl systematicky dokumentován a výsledná hra byla testována na vybrané skupině uživatelů. Na základě zpětné vazby byla zhodnocena funkčnost hry a případně upraven její design a kód. Výsledkem je plně funkční 2D platformer hra doplněná o vývojovou dokumentaci a poznatky z procesu tvorby.

Klíčová slova: Unity, C#, 2D, Platformer, FPS

Creation of a 2D platform game in the chosen game engine

Abstract

This thesis deals with the creation of a 2D platformer game in the chosen game engine. To begin with, a survey of currently available game engines for 2D game development was conducted, including their comparison in terms of functionality, performance and user--friendliness. Based on this analysis, a suitable engine was selected for the project implementation. Furthermore, a game story was designed and game mechanics characteristic of the platformer genre were specified. These mechanics were then implemented and integrated into a working game prototype. The thesis also includes an analysis of the computational complexity of the developed game and a proposal of optimization measures to reduce it. The whole development was systematically documented and the resulting game was tested on a selected group of users. Based on the feedback, the functionality of the game was evaluated and its design and code were modified if necessary. The result is a fully functional 2D platformer game supplemented with development documentation and insights from the design process.

Keywords: Unity, C#, 2D, Platformer, FPS

Poděkování

Rád bych poděkoval panu Ing. Lukáši Zedkovi, Ph.D., za jeho trpělivé vedení, pravidelné konzultace a ochotu věnovat se mé práci s mimořádným nasazením. Oceňuji, že se aktivně zajímal o vývoj hry, pravidelně ji testoval, pročítal dokumentaci a vždy mi poskytl cennou zpětnou vazbu a konkrétní návrhy na zlepšení. Jeho přístup mi výrazně pomohl posunout projekt dál. Dále bych chtěl poděkovat Mgr. Dominice Strečanské za její trpělivost, odhodlání a úsilí při tvorbě grafiky, shaderů a animací pro hru, které hře vtiskly osobitý charakter a výrazně přispěly k její výsledné podobě. Na závěr děkuji všem, kteří se podíleli na testování hry a pomohli mi tak s jejím laděním a vylepšováním.

Obsah

Seznam zkratek . . . 11
Seznam pojmů . . . 12

1 Úvod **13**

2 Teoretická část **14**
2.1 Co je to herní engine? . . . 14
2.2 Klíčové funkce herních enginů . . . 14
2.3 Dělení herních enginů . . . 15
2.3.1 2D herní enginy . . . 15
2.3.2 3D herní enginy . . . 15
2.3.3 Hybridní herní enginy . . . 16
2.3.4 Licenční dostupnost . . . 17
2.3.5 Podporované cílové platformy . . . 17
2.3.6 Zaměření enginů podle účelu vývoje . . . 18
2.3.7 Podpora programování a implementace . . . 19
2.4 Představení herních enginů . . . 19
2.4.1 Unreal Engine . . . 19
2.4.2 Unity . . . 21
2.4.3 Godot . . . 23
2.4.4 GameMaker Studio 2 . . . 24
2.5 Porovnání herních enginů . . . 25
2.6 Zvolený engine . . . 27

3 Představení herního příběhu **28**
3.1 Expozice a kolize . . . 28
3.2 Krize a peripetie . . . 28
3.3 Katastrofa (rozuzlení) . . . 29

4 Implementace herních mechanik **30**
4.1 Zvukový systém . . . 30
4.1.1 Architektura systému . . . 30
4.1.2 Přehrávání audia . . . 30
4.1.3 Ovládání hlasitosti . . . 31
4.1.4 Kategorie a knihovna zvuků . . . 31
4.2 Lobby a navigace . . . 31

4.2.1 Lobby . . . 32
4.2.2 Navigace . . . 34
4.3 Hráčův průvodce . . . 34
4.3.1 Funkce průvodce v lobby . . . 36
4.3.2 Funkce průvodce v herních levelech . . . 36
4.4 Herní předměty, měna a obchod . . . 37
4.4.1 Herní předměty . . . 38
4.4.2 Herní měna . . . 39
4.4.3 Herní obchod . . . 40
4.4.4 Herní inventář . . . 41
4.5 Prostředí úrovní . . . 41
4.5.1 Checkpointy . . . 42
4.5.2 Průchozí plošiny . . . 43
4.5.3 Pohyblivé plošiny . . . 43
4.5.4 Praskající plošiny . . . 44
4.5.5 Truhly s předměty . . . 45
4.5.6 Automaty . . . 46
4.5.7 Vlaštovky . . . 47
4.5.8 Rekurzivní koule . . . 48
4.5.9 Déšť předmětů . . . 49
4.6 Mikrofon – Změna ročního období . . . 49
4.6.1 Ukládání volby mikrofonu do nastavení . . . 49
4.6.2 Detekce hlasitého zvuku . . . 49
4.6.3 Změna ročního období . . . 50
4.7 Systém ukládání herního postupu a nastavení . . . 51
4.7.1 Obsah souboru . . . 52
4.7.2 Příklad obsahu souborů . . . 53

5 Otestování funkčnosti hry **54**
5.1 Obsah formulářů . . . 54
5.2 Typy chyb . . . 55
5.2.1 Neintuitivní chování . . . 55
5.2.2 Lehká chyba . . . 55
5.2.3 Těžká chyba . . . 55
5.2.4 Kritická chyba . . . 56
5.3 Nalezené chyby . . . 56
5.3.1 Snížení výchozí hlasitosti . . . 56
5.3.2 Možnost sběru předmětů v podzimním období . . . 56
5.3.3 Zvuková odezva při zásahu plechovkou . . . 57
5.3.4 Špatné dialogy v budově B . . . 57
5.3.5 Délka zobrazení titulků . . . 57

6 Vyhodnocení výpočetní náročnosti hry **58**
6.1 Počáteční měření FPS . . . 58
6.2 Užité způsoby snížení výpočetní náročnosti . . . 59
6.2.1 Odstranění zbytečného logování . . . 59
6.2.2 Odstranění prázdných metod Update . . . 60
6.2.3 Optimalizace viditelnosti UI prvků pomocí komponenty Canvas 60
6.2.4 Převedení audio stop do mono režimu . . . 61
6.2.5 Optimalizace sledování cíle pomocí cache pozic . . . 61
6.2.6 Odstranění opakovaného načítání prefabů a optimalizace hledání . . . 61
6.3 Naměřené hodnoty po optimalizaci . . . 62

Závěr **63**

Použitá literatura **64**

A Příklad souboru „progress.json“ **71**

B Skript pro zpracování herního logu v jazyce Python **73**

C Měření FPS **74**
C.1 Hlavní menu . . . 74
C.2 Lobby . . . 75
C.3 Level G . . . 75

Seznam zkratek

AR	Rozšířená realita (Augmented Reality)
CPU	Procesor počítače (Central Processing Unit)
FPS	Snímky za sekundu (Frames Per Second)
GML	Jednoduchý skriptovací jazyk určený pro tvorbu her v GameMaker Studiu (GameMaker Language)
JSON	Jednoduchý textový formát pro výměnu dat mezi aplikacemi, často používaný na webu (JavaScript Object Notation)
MIT	Massachusettský technologický institut (Massachusetts Institute of Technology)
RPG	Hra na hrdiny (Role-Playing Game)
UI	Uživatelské rozhraní (User Interface)
VR	Virtuální realita (Virtual Reality)

Seznam pojmů

Asset Jakýkoliv zdroj použitý ve hře (model, textura, zvuk nebo skript).

Blueprints Systém vizuálního programování v enginu Unreal Engine.

C# Programovací jazyk použitý v enginech Unity a Godot.

C++ Programovací jazyk použitý v enginu Unreal Engine.

Checkpoint Uložený bod v levelu, ke kterému se hráč vrací po neúspěchu (například po pádu mimo herní oblast); aktivovaný checkpoint ukládá aktuální pozici hráče.

Collider Komponenta v Unity určující tvar objektu pro detekci kolizí a fyziku („hitbox“).

Cutscéna Neinteraktivní sekvence, která slouží k vyprávění příběhu.

Framework Soubor nástrojů, knihoven, pravidel a komponent pro vývoj konkrétních softwarových aplikací.

Freemium licenční model Licenční model, který základní funkce poskytuje zadarmo, avšak za pokročilé funkce je již potřeba zaplatit.

GDScript Programovací jazyk vytvořený speciálně pro Godot Engine.

Grafický rendering Proces zobrazení 2D nebo 3D scény.

Level Samostatná část hry s vlastním prostředím a cíli, která představuje určitý úsek herního postupu.

Lobby Oblast ve hře, která slouží jako přechodový prostor mezi hlavním menu a jednotlivými herními levely.

Log Textový soubor obsahující informativní výpisy ze hry.

Multiplayerové hry Hry pro více hráčů.

Multithreading Rozdělení výpočetní zátěže na více výpočetních vláken.

Save Soubor JSON (JavaScript Object Notation), který v sobě uchovává informace o nastavení (např. hlasitost zvuků a hudby, zvolený mikrofon, ...) a o herním postupu (stav inventáře, naposledy uložený checkpoint, ...).

Sprite 2D obrázek, který ve scéně reprezentuje objekt.

Tile-based grafika Prostředí levelu je poskládáno do čtvercové sítě.

1 Úvod

Počítačové hry se v posledních desetiletích staly nejen významnou formou zábavy, ale také důležitým médiem pro vyjadřování nápadů, emocí a příběhů. Jejich vývoj propojuje technické i kreativní disciplíny a dává prostor k realizaci vlastních vizí. Tato práce se zaměřuje na tvorbu 2D platformer hry, tedy žánru, který kombinuje jednoduchou hratelnost s možnostmi různorodého designového zpracování.

Zvolené téma je příležitostí spojit technické znalosti s kreativním návrhem herních prvků, příběhu a herních mechanik. Motivací pro jeho výběr byla osobní záliba v počítačových hrách a touha vytvořit vlastní herní projekt, který by přinesl originální koncept a zároveň byl proveditelný v rámci časových a technických možností bakalářské práce. Důraz byl kladen na udržitelnost vývoje – tak, aby bylo možné projekt realizovat v průběhu přibližně osmi měsíců bez potřeby rozsáhlého vývojového týmu.

V první části práce je provedena rešerše dostupných herních enginů vhodných pro tvorbu 2D her a jejich vzájemné porovnání. Následuje návrh příběhu a specifikace herních mechanik odpovídajících zvolenému žánru. Praktická část se věnuje samotné implementaci hry, technickému řešení vybraných mechanik, způsobům dokumentace vývoje a optimalizace výpočetní náročnosti. V závěru jsou představeny výsledky testování a vyhodnocení funkčnosti hry na základě zpětné vazby od uživatelů.

2 Teoretická část

2.1 Co je to herní engine?

Herní engine je softwarový framework, který primárně umožňuje vývoj videoher. Jeho hlavním účelem je usnadnit a urychlit proces tvorby videoher tím, že poskytuje hotová řešení pro technické problémy, jako je vykreslování grafiky, přehrávání zvuků nebo simulace fyzikálních jevů. Díky tomu se vývojáři mohou soustředit na kreativní aspekty hry, jako jsou herní obsah, příběh a mechaniky, aniž by museli tyto základní funkce vytvářet ručně.

Protože je vývoj her vysoce kreativním procesem, herní enginy jsou navrženy s důrazem na modularitu a univerzálnost. Mnoho herních enginů podporuje tvorbu her ve dvou i třírozměrném prostoru. Zároveň nejsou vývojáři omezeni žánrem, což znamená, že jeden engine může být použit pro různé typy her. Existují však i specializované enginy navržené pro konkrétní žánry.

2.2 Klíčové funkce herních enginů

Herní enginy poskytují širokou škálu funkcí, které vývojářům usnadňují a zefektivňují vývoj jejich produktu. I když se konkrétní nabízený rozsah možností může u jednotlivých enginů lišit, všechny obsahují několik základních prvků. Jedním z nejdůležitějších je vykreslování (grafický rendering), což je proces vizuálního zobrazení herního prostředí, postav a dalších objektů v reálném čase na obrazovce. Do renderingu spadá také například podpora osvětlení a stínů, které dodávají hře vizuální hloubku.

Vedle grafického vykreslování hraje klíčovou roli také fyzikální engine. Tento modul zajišťuje správnou simulaci fyzikálních jevů, jako jsou kolize mezi objekty, dynamika tekutin, gravitace nebo tření. Fyzikální engine tak pomáhá vytvářet hry, jejichž fyzikální zákony důvěryhodně napodobují ty skutečné a stará se o interakce mezi objekty v reálném čase [60].

Další zásadní komponentou je audio systém, který se stará o přehrávání zvuků a hudby. Moderní herní enginy nabízejí například podporu prostorového zvuku, díky kterému hráč slyší zvuky v závislosti na jeho poloze vůči zdroji [8, 82]. Pokud se například hráč nachází vpravo od zdroje zvuku, hudba je slyšet především v levém sluchátku, což přispívá k ponoření hráče do herního světa.

Herní enginy rovněž zajišťují správu uživatelských vstupů. Tato funkce zahrnuje zpracování interakcí, jako jsou stisky kláves, pohyby myší, dotyky na dotykovém dis-

pleji, ovládání gamepadem nebo i vstup z mikrofonu. Díky tomu je možné intuitivní a flexibilní ovládání, které lze přizpůsobit preferencím hráče.

Mnoho herních enginů dále nabízí pokročilé funkce, jako je podpora vytváření animací, multithreadingu nebo multiplayeru (hra pro více hráčů).

2.3 Dělení herních enginů

Herní enginy lze rozdělit podle mnoha hledisek, která hrají zásadní roli při výběru vhodného nástroje pro vývoj konkrétní hry. Jedním z nejvýznamnějších kritérií je funkcionalita enginu umožňující vizualizovat herní prostředí ve 2D nebo 3D. Většina moderních enginů podporuje vývoj her s výslednou prezentací jak ve dvourozměrném, tak trojrozměrném prostoru, což z nich činí univerzální nástroje vhodné pro široké spektrum projektů.

2.3.1 2D herní enginy

První kategorií jsou enginy, ve kterých lze vytvořit pouze 2D projekty. Jedním z 2D enginů je GameMaker Studio 2 od společnosti YoYo Games Ltd. [31]. Tento engine byl původně navržen jako uživatelsky přívětivé řešení, které umožní tvorbu videoher i začátečníkům bez zkušeností s programováním. Zároveň ale dovoluje profesionální a plnohodnotný vývoj multižánrových a multiplatformních her. Výslednou hru lze hrát na Windows, Mac, Linux, Android nebo třeba iOS. I když GameMaker Studio 2 obsahuje základní možnosti pro tvorbu 3D her, jeho podpora trojrozměrného obsahu je omezená. Realizace 3D projektů často vyžaduje využití externích knihoven nebo složitých úprav [14]. Z tohoto důvodu není tento engine považován za vhodný nástroj pro vývoj plnohodnotných 3D her.

Dalším z 2D herních enginů je RPG Maker [78] od společnosti Gotcha Gotcha Games Inc., určený ke tvorbě 2D RPG (Role-Playing Games – hra na hrdiny) her. Podobně jako GameMaker Studio 2 je navržen s důrazem na jednoduchost a snadné použití, což z něj činí oblíbený nástroj mezi začátečníky i pokročilými vývojáři. RPG Maker nabízí množství předpřipravených funkcí, jako je snadné editování herního světa, vytváření postav a podpora databází. RPG Maker byl sice dříve označován jen jako nástroj, díky kterému si mohli začínající vývojáři vyzkoušet tvorbu her, avšak tituly jako Omori [66] nebo Seraphic Blue [25] jasně dokazují, že se v tomto herním enginu dá vytvořit i plnohodnotná hra. Co se týče podpory vývoje ve 3D, je na tom RPG Maker podobně jako GameMaker Studio 2. Má pouze omezenou podporu trojrozměrných prvků prostřednictvím pluginů, jako je MV3D [63], což ho činí méně vhodným pro vývoj komplexních 3D her [88].

2.3.2 3D herní enginy

Protipólem enginů, ve kterých lze tvořit pouze 2D obsah, jsou enginy, které naopak 2D nepodporují a soustřeďují se výhradně na 3D hry. Tvorbu 2D her buď vůbec nepodporují, nebo ji podporují jen omezeně. Tyto případy jsou však vzácné a obvykle

reflektují specifické potřeby vývojářů nebo konkrétního žánru her.

Příkladem takového enginu je CryEngine [21] od společnosti Crytek GmbH, který byl použit při vývoji série Far Cry (první díl) a následně upraven společností Ubisoft Entertainment SA na modifikaci známou jako Dunia Engine [50] pro další díly. Na CryEnginu byla rovněž postavena česká hra Kingdom Come: Deliverance 2 [55] od firmy Warhorse Studios s.r.o., která využívala jeho schopnost vykreslovat realistická prostředí a simulovat fyziku v otevřeném světě.

Dalším příkladem enginu podporující pouze 3D je Frostbite Engine [30] od Electronic Arts Inc., jenž byl navržen primárně pro fotorealistické 3D hry a je využíván u titulů jako FIFA 23 [26] , Battlefield 2042 [13], NHL 24 [65] nebo Need for Speed Unbound [64].

Tyto 3D-specializované enginy jsou vzácností, protože většina vývojářů [23] hledá nástroje, které podporují flexibilitu a umožňují snadný vývoj jak 2D, tak 3D her. Enginy zaměřené výhradně na 3D proto často slouží specifickým typům projektů, které vyžadují maximální využití 3D grafiky a realistických fyzikálních simulací.

2.3.3 Hybridní herní enginy

Poslední kategorií jsou herní enginy, které podporují jak vizualizaci 2D prostoru, tak vizualizaci 3D. Tento přístup z nich činí mimořádně flexibilní nástroje vhodné pro široké spektrum projektů, ať už jde o jednoduché dvourozměrné plošinovky, nebo rozsáhlé trojrozměrné světy. Právě možnosti těchto enginů podporovat vývoj her obou druhů dávají vývojářům svobodu volby a prostor pro experimentování s různými herními žánry.

Největšími zástupci této kategorie jsou například herní engine Unreal Engine [94], vytvořený společností Epic Games Inc. I když Unreal Engine ve svém základu neposkytuje podporu 2D vývoje, jeho oficiální plugin Paper 2D [68] toto umožňuje jako doplňkové řešení pro vývoj 2D her.

Dalším enginem v této kategorii je komunitně vyvíjený herní engine Godot [44]. Podpora vývoje dvou-a trojrozměrných her je v Godotu integrovaná přímo, bez nutnosti využívat jakýkoli plugin [100]. Díky svému otevřenému licenčnímu modelu je Godot oblíbený jak mezi nezávislými vývojáři, tak mezi profesionálními studii.

V neposlední řadě je to herní engine Unity [92], jehož autorem je společnost Unity Technologies Inc. Unity je aktuálně jedním z nejpoužívanějších enginů na světě, a to jak ve 2D, tak ve 3D vývoji [23]. Tento engine vyniká svou multiplatformní podporou, což jej činí ideální volbou pro vývoj her pro mobilní zařízení, desktop i virtuální realitu.

Univerzální enginy jako Unreal Engine, Godot a Unity nabízejí vývojářům výjimečnou flexibilitu a možnost oslovit široké publikum díky podpoře různých platforem i herních žánrů. Díky své všestrannosti se staly základními nástroji jak pro nezávislé vývojáře, tak i vícečlenná herní studia [62].

2.3.4 Licenční dostupnost

Licenční podmínky představují jeden z klíčových parametrů při výběru herního enginu. Licence určuje, jaké podmínky musí vývojáři splnit při používání konkrétního enginu. Tyto podmínky mohou zahrnovat finanční závazky, jako je jednorázová platba za licenci, procentuální podíl z tržeb vytvořené hry, nebo mohou být enginy dostupné zcela zdarma. Dále licence definuje, zda má vývojář právo upravovat zdrojový kód enginu, což je důležité zejména pro technicky náročné projekty vyžadující vysokou míru přizpůsobení. Herní engine s volně přístupným zdrojovým kódem je například Godot [56].

Dalším specifickým případem jsou herní enginy, které nejsou veřejně dostupné běžným uživatelům a jsou vyvíjeny výhradně pro potřeby jednoho konkrétního herního studia. Tyto proprietární enginy jsou často navrženy na míru konkrétním požadavkům studia a her, které vytvářejí (například Frostbite Engine [30]).

Proprietární enginy nabízejí vývojářským studiím absolutní kontrolu nad vývojem i výslednou technologií, což jim umožňuje přizpůsobit engine specifickým potřebám jejich her. Na druhou stranu však jejich vývoj a údržba vyžaduje vysoké náklady a specializované týmy. Tento přístup je proto typický pro velká studia s dostatečnými finančními i lidskými zdroji.

Licenční modely mohou být také ovlivněny typem vývojářů, pro které je engine určen. Například nezávislí vývojáři často preferují open-source licence (volný přístup ke zdrojovému kódu [87]) nebo freemium modely [28], které umožňují začít zdarma a platit pouze za pokročilé funkce nebo při dosažení určitého komerčního úspěchu. Naopak velká studia si často pořizují placené licence, které jim zaručují podporu od výrobce enginu, přístup k nejnovějším aktualizacím a celkově více možností při vývoji.

Kromě toho licence stanovuje, jak může být výsledná hra distribuována. Některé licence povolují pouze nekomerční využití, což znamená, že výsledná hra nesmí být prodávána nebo jinak monetizována. Jiné licence poskytují vývojářům úplnou volnost v komerčním využití. Významným aspektem je také podpora multiplatformního vývoje – některé enginy umožňují publikaci na více platformách bez dodatečných nákladů, zatímco jiné vyžadují zakoupení samostatných licencí pro každou platformu.

V neposlední řadě je důležité zmínit otázku právního vlastnictví. Některé licenční podmínky mohou určovat, že tvůrce enginu má právo na určitou část intelektuálního vlastnictví nebo zisku vytvořeného hrou. Tento aspekt je klíčový zejména pro vývojáře, kteří plánují svůj projekt komerčně využít.

2.3.5 Podporované cílové platformy

Mezi nejdůležitější parametry herních enginů bezpochyby patří i množství a typy platforem, pro které je engine schopen výslednou hru uvolnit, protože tato vlastnost přímo ovlivňuje dostupnost hry pro hráče a možnosti jejího komerčního uplatnění. Různé enginy podporují různé operační systémy, včetně Windows, macOS a Linuxu, herních konzolí, mobilních platforem jako Android a iOS, webových technologií vy-

užívajících WebGL [99] nebo WebAssembly, a také specializovaných nástrojů, jako jsou VR (Virtual Reality – virtuální realita) a AR (Augmented Reality – rozšířená realita) zařízení.

Širší podpora platforem umožňuje vývojářům oslovit větší publikum, avšak přináší i výzvy v podobě optimalizace výkonu, uživatelského rozhraní a kompatibility s hardwarem každé z cílových platforem. Například vývoj hry pro mobilní zařízení klade důraz na dotykové ovládání, úsporu baterie a optimalizaci výkonu pro slabší hardware, zatímco hry pro herní konzole či desktopové systémy mohou využívat vyšší výpočetní výkon, podporu gamepadů a širší možnosti grafického zpracování. Některé enginy nabízejí nativní podporu multiplatformního exportu, což umožňuje s minimálními úpravami spustit hru na různých zařízeních, zatímco jiné vyžadují individuální úpravy kódu pro každou platformu zvlášť.

Při výběru enginu vývojáři zohledňují nejen aktuálně podporované operační systémy a zařízení, ale i předpokládanou délku budoucí podpory.

2.3.6 Zaměření enginů podle účelu vývoje

Většina moderních herních enginů není primárně zaměřena na jeden konkrétní herní žánr. Univerzální enginy jako Unity, Godot nebo Unreal Engine poskytují širokou škálu nástrojů, které umožňují vývoj nejrůznějších typů her – od vizuálních románů přes RPG až po komplexní 3D akční tituly. Jejich flexibilita je jednou z největších výhod, protože dávají vývojářům volnost v tom, jaké mechaniky a prvky do své hry zakomponují.

Výjimkou však nejsou ani specializované enginy, které jsou určeny výhradně pro tvorbu her určitého žánru. Takové enginy se zaměřují na specifické potřeby daného typu hry a často přicházejí s vestavěnými funkcemi, které usnadňují a urychlují vývoj. Ren'Py [101] je například optimalizován pro tvorbu vizuálních románů a nabízí předpřipravené nástroje pro práci s dialogy, větvením příběhu a grafikou ve stylu interaktivních příběhů. RPG Maker je zase navržen pro tvorbu klasických 2D RPG her se systémem skládání map z dlaždic (gridový systém), zabudovanými soubojovými mechanikami a správou postav, což umožňuje vývojářům soustředit se více na obsah než na technickou implementaci.

Přestože se specializované enginy mohou zdát limitující, přinášejí několik klíčových výhod:

- Zjednodušený vývoj – Díky vestavěným funkcím a optimalizovanému workflow lze hru vyvíjet rychleji, aniž by bylo nutné implementovat základní mechaniky od nuly.
- Nižší nároky na znalost programování – Mnohé specializované enginy umožňují tvorbu her s minimálním nebo žádným programováním, což je ideální pro menší týmy nebo jednotlivce bez hlubších technických znalostí.
- Optimalizace pro daný žánr – Enginy jsou uzpůsobeny konkrétním požadavkům daného žánru, což může vést k lepší výkonnosti a stabilitě výsledné hry.

Na druhou stranu má tento přístup i své nevýhody:

- Omezená flexibilita – Vývojáři jsou vázáni na předdefinované mechaniky a možnosti enginu, což může být problém při dalším rozšiřování hry.
- Podpora a ekosystém enginu – Kvůli užšímu využití specializovaných enginů mají tyto enginy logicky menší uživatelskou základnu v porovnání s nejpoužívanějšími univerzálními enginy.

2.3.7 Podpora programování a implementace

Podpora programování v herních enginech hraje klíčovou roli při tvorbě herní logiky a interakcí. Většina moderních enginů umožňuje programování v běžných programovacích jazycích, přičemž konkrétní výběr se liší podle daného enginu. Unity primárně využívá C#, Unreal Engine podporuje C++ a Python, Godot nabízí svůj vlastní jazyk GDScript, ale zároveň umožňuje skriptování v C# a C++. CryEngine se zaměřuje na C++, ale podporuje i Lua. Další enginy jako Source Engine či id Tech využívají převážně C++, přičemž některé z nich poskytují i možnost rozšíření pomocí Pythonu nebo jiných jazyků.

Kromě tradičního programování některé herní enginy umožňují i vizuální programování, což vývojářům poskytuje flexibilitu při tvorbě herní logiky. Unreal Engine například kromě podpory C++ nabízí systém Blueprints [53], který umožňuje vytvářet interakce a herní mechaniky pomocí uzlového grafického rozhraní. Podobně Godot kombinuje GDScript s vizuálním programováním, čímž umožňuje snadnější návrh interakcí bez nutnosti psát kód. Tento hybridní přístup se využívá i v dalších enginech, přičemž dává vývojářům možnost kombinovat vizuální nástroje s výkonností a kontrolou klasického programování.

Vizuální programování umožňuje tvorbu herních mechanik a interakcí prostřednictvím grafického rozhraní, ve kterém se funkce propojují pomocí jednotlivých bloků reprezentující části programu. Tento přístup snižuje potřebu psaní kódu a usnadňuje práci s logikou hry.

2.4 Představení herních enginů

2.4.1 Unreal Engine

Herní engine Unreal od společnosti Epic Games Inc. je známý především svou schopností vykreslovat vysoce realistické 3D scény, což z něj činí populární nástroj nejen pro herní průmysl, ale i pro oblasti jako kinematografie, architektura nebo automobilový design. Unreal Engine však nativně nepodporuje vývoj 2D her, a proto Epic Games vytvořili plugin Paper2D, který tuto funkcionalitu doplňuje. Paper2D umožňuje práci se sprity (2D obrázky reprezentující objekty ve scéně) a sprite sheety, které sdružují více spritů v jednom souboru a usnadňují tvorbu animací. Dále podporuje práci s tile-based grafikou, kterou proslavily hry jako Super Mario Bros. [85] nebo Celeste [17, 68].

V Unreal Engine je licencování založené na bezplatném přístupu s podílem na zisku. Vývojáři mohou engine stáhnout a používat zdarma, včetně přístupu k jeho

zdrojovému kódu. Pokud ale hra nebo aplikace vytvořená v Unreal Engine vydělá více než 1 000 000 USD, Epic Games si nárokuje 5 % z hrubého příjmu nad touto hranicí. Existují i výjimky – například pokud je hra distribuována přes Epic Games Store, podíl na zisku se snižuje na 3,5 %. Pro nekomerční projekty, vzdělávací účely a menší vývojáře (s ročním příjmem pod 1 milion USD) žádné poplatky nejsou. Unreal Engine tak umožňuje vývoj bez počátečních investic, ale při komerčním úspěchu vyžaduje podíl na výdělku [95].

Unreal Engine podporuje širokou škálu cílových platforem, pro které lze hru sestavit. Kromě běžných desktopových systémů (Windows, Linux, macOS) umožňuje vývoj také pro mobilní zařízení (Android, iOS) a herní konzole (Xbox, PlayStation, Nintendo Switch). Kromě těchto standardních platforem nabízí podporu i pro specifická zařízení, jako je Steam Deck a Oculus pro VR aplikace.

Unreal Engine poskytuje dvě hlavní metody pro tvorbu herní logiky: vizuální skriptování pomocí Blueprints a tradiční programování v C++. Blueprints je vizuální programovací jazyk založený na funkčních blocích, který nabízí odlišný pracovní postup ve srovnání s C++. Tento přístup umožňuje uživatelům vytvářet herní funkce bez nutnosti psaní kódu, což je výhodné pro rychlé prototypování a pro ty, kteří nemusí být obeznámeni s tradičním programováním. Na druhé straně, C++ poskytuje širší přístup k funkcionalitám Unreal Engine a umožňuje přímou manipulaci s pamětí, což je vhodné pro vývoj systémů vyžadujících vysoký výkon.

Výkonové rozdíly mezi těmito dvěma přístupy jsou významné. C++ se kompiluje do strojového kódu, který se provádí přímo na CPU (Central Processing Unit – procesor počítače), zatímco Blueprints se kompilují do bytecode, který běží na virtuálním stroji. Tato skutečnost znamená, že Blueprints mají dodatečnou režii při provádění skriptů, což může vést k nižšímu výkonu ve srovnání s C++ . Nicméně, v mnoha případech jsou tyto rozdíly zanedbatelné a volba mezi Blueprints a C++ by měla záviset na konkrétních požadavcích projektu.

V praxi se často kombinuje použití Blueprints a C++. Například, základní systémy hry mohou být implementovány v C++ pro zajištění vysokého výkonu, zatímco specifické herní chování nebo úpravy mohou být prováděny pomocí Blueprints, což umožňuje rychlejší iteraci a snadnější úpravy. Tento hybridní přístup umožňuje využít výhod obou metod a přizpůsobit vývojový proces potřebám týmu a projektu [18].

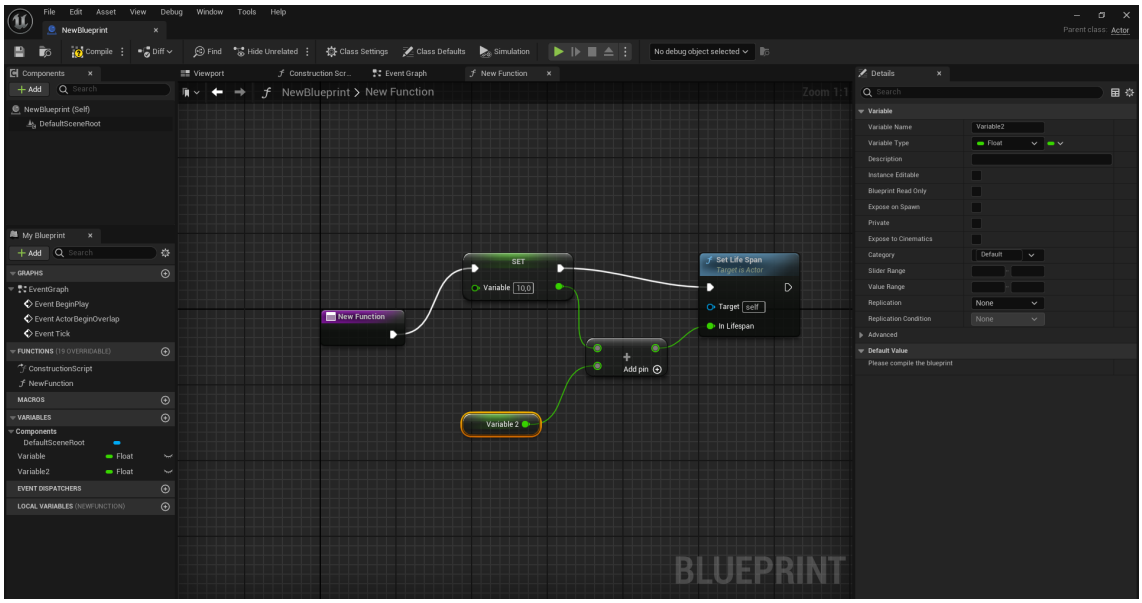
Unreal Engine nabízí také podporu pro skriptování pomocí jazyka Python, který je primárně využíván pro automatizaci procesů a správu editoru. Python umožňuje například dávkové importování assetů, vytváření nástrojů pro vývojáře nebo modifikaci uživatelského rozhraní editoru. Na rozdíl od C++ a Blueprints, které slouží k vývoji samotné hry, se Python nepoužívá k běhu herní logiky v reálném čase, ale spíše ke zjednodušení pracovních postupů v editoru. Tento přístup je užitečný zejména ve větších týmech, kde je potřeba efektivně pracovat s velkým množstvím assetů nebo provádět opakující se úlohy [79].

Unreal Engine disponuje fyzikálním enginem Chaos Physics [49], který se stará o simulaci fyziky v projektech, včetně kolizí, destrukce objektů a simulace pevných těles. Nativní podpora 2D fyziky však chybí, takže uživatelé musí využít již zmíněný plugin Paper2D nebo implementovat vlastní řešení.

Unreal Engine nabízí komunitní obchod s assety jménem Fab [24], který zahr-

nuje širokou škálu 2D i 3D modelů, zvuků, materiálů, animací, návodů a dokonce i hotových herních systémů. Fab je přímo integrovaný do Unreal Engine a dalších nástrojů Epic Games, což umožňuje snadný import assetů do projektů. Většina obsahu je placená, ale k dispozici je také rozsáhlá knihovna bezplatných assetů, včetně měsíčně aktualizovaných balíčků od Epic Games. Tyto zdarma dostupné assety mohou být užitečné zejména v raných fázích vývoje pro prototypování herních mechanik a funkcí.

Unreal Engine byl použit k vývoji mnoha herních titulů napříč různými žánry. Mezi nimi jsou například Fortnite, battle royale hra od Epic Games, Valorant [97], taktická střílečka od Riot Games Inc. a Sea of Thieves [80], multiplayerová pirátská adventura od Rare Limited. Unreal Engine se také využívá u her jako PUBG: Battlegrounds [75], Final Fantasy VII Remake [27] nebo Gears of War 5 [42]. Díky své flexibilitě nachází uplatnění v rozsáhlých komerčních projektech i menších nezávislých hrách. Příklad vývojového prostředí Unreal Enginu je vidět na Obrázku 2.1.



Obrázek 2.1: Vývojové prostředí Unreal Enginu

2.4.2 Unity

Herní engine Unity, vyvinutý společností Unity Technologies Inc., na rozdíl od Unreal Enginu nabízí nativní podporu pro vývoj 2D her, aniž by bylo nutné používat externí pluginy. Zároveň obsahuje nástroje pro tvorbu 3D her.

Unity nabízí několik licenčních modelů, které se liší podle potřeb vývojářů a jejich finančních možností. Každá licence definuje, jak lze engine používat, jaká jsou omezení a co je součástí balíčku. Unity Personal je zdarma a určené pro jednotlivce a malé týmy s ročním příjmem nebo financováním do 100 000 USD. Obsahuje základní funkce enginu, ale zahrnuje Unity Splash Screen, který nelze odstranit. Unity

Splash Screen je úvodní obrazovka zobrazující logo Unity při spuštění hry, což může působit neprofesionálně u komerčních titulů. Není vhodné pro firmy s vyššími příjmy nebo pro vývojáře, kteří chtějí plně přizpůsobit branding své hry.

Unity Student je speciální varianta dostupná zdarma pro studenty a akademické instituce, poskytující přístup k pokročilým nástrojům včetně některých funkcí Unity Pro, ale s omezeními týkajícími se komerčního využití.

Unity Pro je placená licence určená pro profesionální vývojáře a firmy s příjmem nad 100 000 USD. Umožňuje odstranit Unity Splash Screen, nabízí pokročilé funkce, podporu a přístup k prioritnímu technickému servisu a zahrnuje možnosti pro rozšíření o Unity Industry nebo Enterprise řešení.

Unity Enterprise je nejvyšší úroveň licence s individuální cenovou nabídkou. Nabízí nejlepší podporu, škálovatelnost a přístup k pokročilým nástrojům a je určeno pro velká studia a společnosti, které potřebují individuální přístup. Existují také dodatečné podmínky a omezení. Unity nelze používat pro hazardní hry a některé jiné regulované oblasti bez zvláštního povolení. Projekty využívající Unity jako hlavní součást vestavného systému vyžadují zvláštní licenci [93].

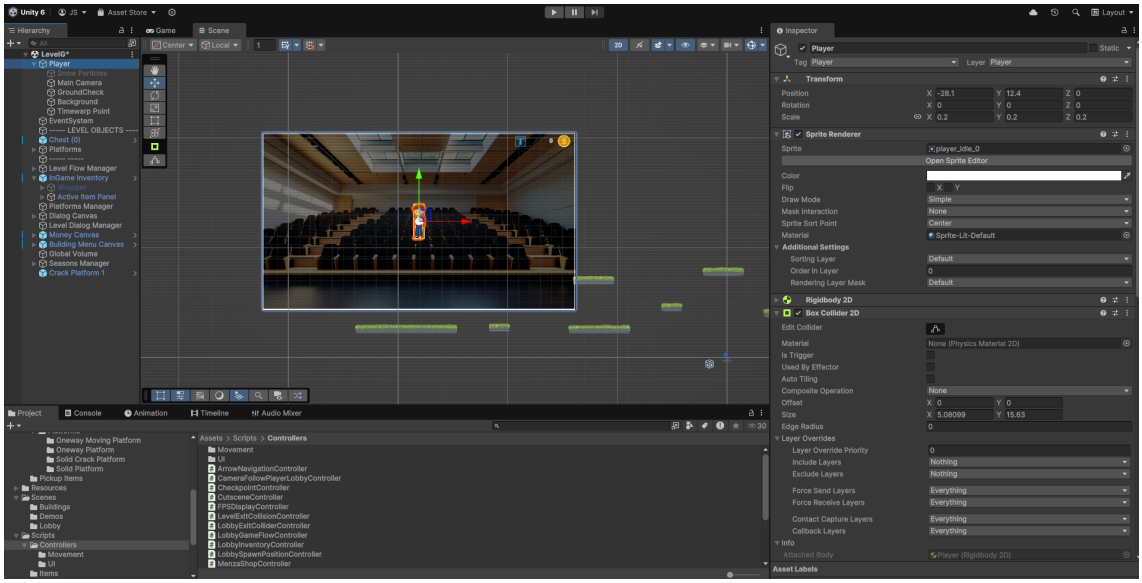
Tento engine podporuje širokou škálu platforem, což z něj činí flexibilní engine pro vývoj her i dalších interaktivních aplikací. Mezi podporované platformy patří Windows, macOS a Linux pro vývoj i spouštění aplikací, mobilní systémy Android a iOS, herní konzole jako PlayStation, Xbox a Nintendo Switch, webové aplikace prostřednictvím WebGL a také platformy pro virtuální a rozšířenou realitu, například Meta Quest [89] nebo HoloLens [59]. Unity rovněž umožňuje vývoj aplikací pro rozšířenou realitu (AR) na mobilních zařízeních pomocí ARKit [4] (iOS) a ARCore [2] (Android).

Programovací jazyk, se kterým se vývojáři v Unity setkají, je C#. Podle různých nezávislých testů [47, 15] je C# pomalejší, než C++, což může být jeden z důvodů, proč někteří herní vývojáři sáhnou raději po Unreal Enginu než po Unity.

V Unity je k dispozici přímá podpora fyzikálních jevů jak ve 3D, tak i ve 2D prostoru. O simulaci 3D fyziky se stará engine Nvidia PhysX, který umožňuje zpracování kolizí, sil, pohybu, gravitace nebo například ragdoll (fyzikálně řízený pád nebo pohyb těla postavy) animací. Pro 2D fyziku Unity využívá Box2D, což je kompaktní fyzikální engine optimalizovaný pro simulaci pohybu a kolizí v rovině. Oba systémy jsou v Unity integrovány nativně a umožňují nastavení parametrů jako je hmotnost, tření, pružnost, nebo detekce kolizí [69].

Stejně jako Unreal Engine, i Unity disponuje vlastní platformou pro distribuci komunitních i komerčních assetů – Unity Asset Store [6]. Ten je integrován přímo do editoru Unity, což umožňuje rychlé vyhledávání a import assetů bez nutnosti opustit prostředí vývojového nástroje. Asset Store podporuje publikaci obsahu ze strany uživatelů a vývojářských studií, přičemž nabízí jak bezplatné položky, tak i rozsáhlou škálu placených balíčků pro různé fáze vývoje.

Mezi známé tituly vytvořené v Unity patří například Rust, Among Us, Cuphead, VR hra Beat Saber od českého studia Beat Games s.r.o. nebo Phasmophobia (dle nejprodávanějších her v Unity v knihovně Steam k 29. 4. 2025) [41]. Příklad vývojového prostředí Unity je vidět na Obrázku 2.2.



Obrázek 2.2: Vývojové prostředí Unity

2.4.3 Godot

Godot je open-source herní engine vyvíjený pod vedením Juana Linietského a Ariela Manzu. Představuje vhodnou volbu pro vývoj 2D plošinových her, přičemž zároveň nabízí podporu pro vývoj 3D projektů. Engine obsahuje nástroje pro práci s 2D i 3D scénami, fyzikou, animacemi, skriptováním a grafickým uživatelským rozhraním, a je tak možné ho využít pro různé typy herních žánrů.

Godot Engine je svobodný a open-source herní engine, který je šířen pod permisivní licencí MIT (Massachusetts Institute of Technology – Massachusettský technologický institut). Tato licence umožňuje uživatelům engine volně používat, upravovat a dále distribuovat – a to jak v nezměněné, tak v modifikované podobě, včetně komerčního využití. Jedinou podmínkou je uvedení původního copyrightového oznámení a textu licence při jakékoli formě redistribuce. Licence se nevztahuje na obsah vytvořený pomocí Godotu, takže vývojáři mohou svá díla licencovat dle vlastního uvážení. Distribuovaná hra ale musí obsahovat odkaz na licenci Godot Enginu – například ve formě odkazu na stránky enginu [56].

Godot Engine podporuje export her na více platforem, včetně Windows, macOS a Linuxu. Dále je možné cílit na mobilní zařízení se systémy Android a iOS, nebo na webové prohlížeče prostřednictvím exportu do WebAssembly/WebGL. Exporty jsou dostupné jak ve 32bitové, tak 64bitové verzi, přičemž výchozí volbou je 64bitová architektura. Oficiální macOS buildy navíc podporují nativně jak procesory Apple Silicon, tak x86_64. Godot lze navíc sestavit a spustit i na některých ARM zařízeních s operačním systémem Linux, jako je například Raspberry Pi, a to i přesto, že tato platforma není oficiálně podporována [29].

Godot Engine podporuje více oficiálních skriptovacích jazyků, které lze využít při tvorbě herní logiky. Primárním jazykem je GDScript, který byl vytvořen speciálně pro potřeby Godotu. Má jednoduchou syntaxi, rychlé načítání a úzkou integraci

s editorem. Dále je podporováno C#, které vyžaduje použití .NET verze editoru a externího vývojového prostředí. C# je vhodné pro vývojáře, kteří již s tímto jazykem mají zkušenosti, a umožňuje využití .NET knihoven. Pomocí technologie GDExtension je možné rozšiřovat engine také v jazycích C a C++, což je vhodné zejména pro výpočetně náročné části projektu. Uživatelé mohou kombinovat více jazyků v rámci jednoho projektu podle potřeb. Skripty v Godotu se připojují k uzlům (nodes) ve scéně a rozšiřují jejich funkcionalitu, přičemž přebírají vlastnosti a metody daného uzlu.

Godot Engine využívá pro simulaci fyziky vlastní integrované enginy, a to jak pro 2D, tak i 3D prostor. Ve 2D využívá Godot od verze 4 přepsaný interní fyzikální engine napsaný v jazyce C++, který podporuje simulaci kolizí, tuhých těles, kloubů a senzorů. Pro 3D fyziku je ve výchozím nastavení použit engine GodotPhysics [71], který je součástí Godotu. Alternativně lze využít také JoltPhysics [96], moderní externí fyzikální engine, který lze aktivovat v nastavení projektu jako alternativní backend.

Stejně jako Unreal Engine a Unity, i Godot nabízí svým uživatelům přístup ke komunitně sdíleným assetům. Jednou z možností je Godot Marketplace [45], který provozují členové komunity a částí svých příjmů finančně podporuje vývoj samotného enginu. Druhou cestou je oficiální Godot Asset Library, která obsahuje množství assetů dostupných pod licencí MIT. Díky tomu je možné tyto zdroje bezplatně využívat, upravovat a distribuovat v rámci vlastních projektů [5]. Tento engine byl využit například při tvorbě her Swords and Sandals Immortals [86] nebo Road to Vostok [77].

2.4.4 GameMaker Studio 2

GameMaker Studio 2 od společnosti YoYo Games Ltd. patří mezi enginy zaměřené výhradně na vývoj 2D her. Možnosti tvorby 3D obsahu nejsou nativně podporovány a spoléhají na komunitní knihovny, jako je například bbmod [14]. Vzhledem k tomu, že tyto nástroje nejsou oficiální součástí enginu, nelze garantovat jejich dlouhodobou podporu nebo kompatibilitu s budoucími verzemi.

Licence Free je zdarma a je vhodná zejména pro začátečníky, studenty nebo vývojáře, kteří si chtějí GameMaker vyzkoušet bez jakýchkoliv nákladů. Tato licence je platná po celou dobu existence uživatelského účtu a umožňuje exportovat projekty na platformy GX.games [46], Windows, macOS, Ubuntu, HTML5, Android, iOS a tvOS. Komerční využití je povoleno pouze pro platformu GX.games. Nevyžaduje žádné poplatky za instalace, neobsahuje žádné skryté náklady ani podíl na zisku. Licence Professional je určena pro nezávislé vývojáře a menší studia, která chtějí distribuovat hry komerčně na více platformách. Tato licence je placená, cena závisí na aktuální nabídce společnosti Opera. Jde o trvalou (perpetuální) licenci, která nevyprší. Umožňuje komerční export na všechny hlavní platformy: GX.games, Windows, macOS, Ubuntu, HTML5, Android, iOS a tvOS. Neobsahuje žádný revenue share, náklady za instalace ani další poplatky. Licence Enterprise je určena profesionálním vývojářům a studiím, která cílí i na konzolový trh. Licence je placená a její platnost se odvíjí od zvoleného časového období při nákupu. Nabízí stejné

platformy jako Professional licence a navíc přidává export na PlayStation, Xbox a Nintendo Switch. Pro tyto konzolové platformy je však nutné mít schválený developerský účet u příslušných výrobců konzolí. Ani tato licence neobsahuje žádné poplatky za revenue share nebo instalace, a nejsou zde účtovány žádné dodatečné náklady [35, 36, 34].

V GameMakeru se k programování používá jazyk vytvořený přímo pro tento engine – GameMaker Language (GML) [32]. Jde o skriptovací jazyk s jednoduchou syntaxí, která se podobá jazykům jako JavaScript nebo C, a je navržen tak, aby byl snadno pochopitelný i pro začátečníky. Umožňuje psát logiku hry, ovládat objekty, reagovat na události nebo pracovat s grafikou, zvukem a animacemi. Vedle klasického psaní kódu je k dispozici také nástroj GML Visual [43], který nabízí možnost vizuálního programování pomocí funkčních bloků a schémat. Tento přístup je vhodný zejména pro uživatele, kteří nemají předchozí zkušenosti s programováním. GameMaker tak umožňuje začít s tvorbou her i úplným začátečníkům a postupně přejít k psaní vlastního kódu. O fyziku v enginu se stejně jako v Unity stará Box2D [70].

GameMaker disponuje vlastním komunitním tržištěm, které umožňuje uživatelům sdílet a stahovat assety. Ve srovnání s konkurenčními platformami je však nabídka assetů v GameMaker Marketplace méně rozsáhlá a obsah je zřídkakdy aktualizován. Na základě provedené rešerše nebyl dohledán žádný asset, který by obdržel aktualizaci v období od dubna 2023 do dubna 2025, což ukazuje nízkou frekvenci aktualizací dostupného obsahu [33].

V tomto enginu byly vytvořeny tituly jako například Undertale [91] nebo Hyper Light Drifter [48].

2.5 Porovnání herních enginů

Základní vlastnosti představených herních enginů, potažmo i jejich výhody a nevýhody, jsou shrnuty a rozvedeny v Tabulkách 2.1, 2.2.

Tabulka 2.1: Obecné vlastnosti a podpora platforem

Název herního enginu	3D	2D	Licenční dostupnost	Cílové platformy	Žánr
Unreal Engine	Ano	Plugin Paper 2D	Zdarma pro menší herní studia	Windows, Linux, macOS, Android, iOS, VR/AR zařízení	Multižánrový
Unity	Ano	Ano	Zdarma pro menší herní studia	Windows, Linux, macOS, Android, iOS, VR/AR zařízení	Multižánrový
Godot	Ano	Ano	Opensource	Windows, Linux, macOS, Android, iOS	Multižánrový
GameMaker Studio 2	Ne nativně	Ano	Zdarma pro nekomerční projekty	GX.games, Windows, Linux, macOS, HTML5, Android, iOS, tvOS	Multižánrový

Tabulka 2.2: Technické možnosti a assety

Název herního enginu	Podpora programování	Obchod s assety	Fyzikální engine
Unreal Engine	C++, Blueprints (Python)	Fab	Nativně Chaos Physics pro 3D
Unity	C#	Unity Asset Store	Nvidia PhysX (3D), Box2D (2D)
Godot	GDScript, C#	Godot Marketplace, Godot Asset Library	GodotPhysics, JoltPhysics
GameMaker Studio 2	GML	GameMaker Marketplace	Box2D

2.6 Zvolený engine

Pro tvorbu hry a implementaci herních mechanik bylo zvoleno Unity. Učiněno tak bylo především z důvodu široké funkcionality, podpory 2D vývoje, rozsáhlé dokumentace a aktivní komunity. Při výběru sehrála roli také dostupnost bohatého ekosystému nástrojů, assetů a podpůrných komunitních zdrojů, které usnadňují a urychlují vývojový proces. V neposlední řadě bylo přihlédnuto k tomu, že autor práce s tímto enginem již v minulosti pracoval, a díky těmto praktickým zkušenostem mohl být vývoj projektu zahájen bez nutnosti seznamování se se základy prostředí. Unity tak bylo vyhodnoceno jako nejvhodnější nástroj pro naplnění cílů této bakalářské práce. Hra je vytvořena ve verzi Unity 6000.0.25, která byla v počáteční fázi vývoje nejaktuálnější.

3 Představení herního příběhu

Herní příběh je jednoduchý, lineární a odehrává se během dvou dnů. Sleduje osud nejmenovaného studenta fakulty mechatroniky Technické univerzity v Liberci a jeho cestu k vysněnému bakalářskému titulu. Celý příběh je vystavěn podle tříaktové struktury.

3.1 Expozice a kolize

Hráč se vžívá do role vzorného a schopného studenta, který dosud se studiem neměl sebemenší potíže. Postupem času se však kvůli rostoucí náročnosti látky začíná ztrácet a jeho počáteční nadšení střídá únava, frustrace a narůstající nejistota. Neúspěšné pokusy o zvládnutí několika zkoušek ho doženou až k pocitu naprosté beznaděje.

Vrchol této fáze nastává ve chvíli, kdy student sedí ve svém pokoji na kolejích a dochází mu, že pokud nezvládne nadcházející zkoušku, jeho studium se zásadně zkomplikuje a velmi pravděpodobně skončí. Po neúspěchu u testu propadá bezmoci a vydává se zapomenout do nedalekého baru.

Právě tam se mu v opilosti zjeví tajemná postava – černý kocour ve futuristickém kosmonautském skafandru. Kocour projeví studentovi pochopení a nabídne mu pomoc. Pokud vypije zvláštní tekutinu, kterou mu kocour podá, získá šanci své studium dokončit. Student váhá, ale nakonec souhlasí a nápoj vypije.

3.2 Krize a peripetie

Dalšího rána se student probouzí na univerzitním náměstí, zmatený a dezorientovaný. Vedle něj stojí známý kocour, který mu podává čisté oblečení a vysvětluje, že účinky nápoje mu upravily hlasivky. Nyní dokáže měnit roční období hlasitým výkřikem.

V průběhu následující části hry se hráč (student) vydává na cestu kampusem, během níž musí překonat tři obtížné zkoušky. Každý level odpovídá jedné z nich a je založen na kombinaci logického uvažování, prostorové orientace a práce s mechanikou změny ročního období. Kocour slouží jako průvodce a rádce, ale veškerá rozhodnutí zůstávají na hráči.

3.3 Katastrofa (rozuzlení)

Po splnění všech zkoušek se student ocitá v reprezentativní místnosti, kde se konají státní závěrečné zkoušky. Neví, jak se tam dostal, ale komise před ním už sedí a vyčkává jeho prezentace. Student stojí zmateně v obleku, který si nepamatuje, že by si oblékal, bez batohu a s podivným pocitem v krku, jako by ztratil schopnost křičet.

Dřív než stihne cokoliv říct, celou scénu přeruší ostrý zvuk budíku. Student se probouzí ve svém pokoji. Celý příběh byl jen sen. Přesto v něm zůstává silný dojem z prožitého snového dobrodružství. Podívá se na ciferník – za hodinu ho čeká zkouška, kterou musí zvládnout, pokud nechce, aby se sen stal skutečností.

4 Implementace herních mechanik

4.1 Zvukový systém

Zvukový systém zajišťuje přehrávání hudby, zvukových efektů a spravuje hlasitost v rámci celé hry. Je navržen tak, aby byl jednoduše rozšiřitelný, přehledně kategorizovaný a respektoval uživatelská nastavení, která se načítají ze souboru „settings.json“ (popsáno v Kapitole 4.7).

4.1.1 Architektura systému

Centrálním prvkem systému je vytvořená třída `AudioManager`, která je implementována jako singleton, aby byla dostupná z jakéhokoli místa ve hře a zároveň nedocházelo k vytváření více instancí. Obsahuje následující prvky:

- `AudioSource SFXAudioSource` – slouží k přehrávání zvukových efektů [10].
- `AudioSource MusicAudioSource` – určený pro přehrávání hudby na pozadí.
- `AudioMixer` – umožňuje dynamické řízení hlasitosti (pomocí hodnot v decibelech) a je propojen s UI (User Interface – uživatelské rozhraní) ovládáním [9].
- `AudioLibrary` – strukturovaná kolekce všech zvukových klipů rozdělená podle kategorií (např. uživatelské rozhraní, hráč, hudba).

4.1.2 Přehrávání audia

Pro přehrávání zvuků je využívána metoda `PlayClipByName`, která na základě jména vyhledá klip v příslušné kategorii a přehraje ho. Podle potřeby lze zvolit:

- `PlayOneShot` – přehrání klipu bez přerušení jiného přehrávaného zvuku (využíváno pro zvukové efekty) [12].
- `Play` – nastavení klipu jako hlavního a přehrání s možností smyčky (využíváno pro hudbu) [11].

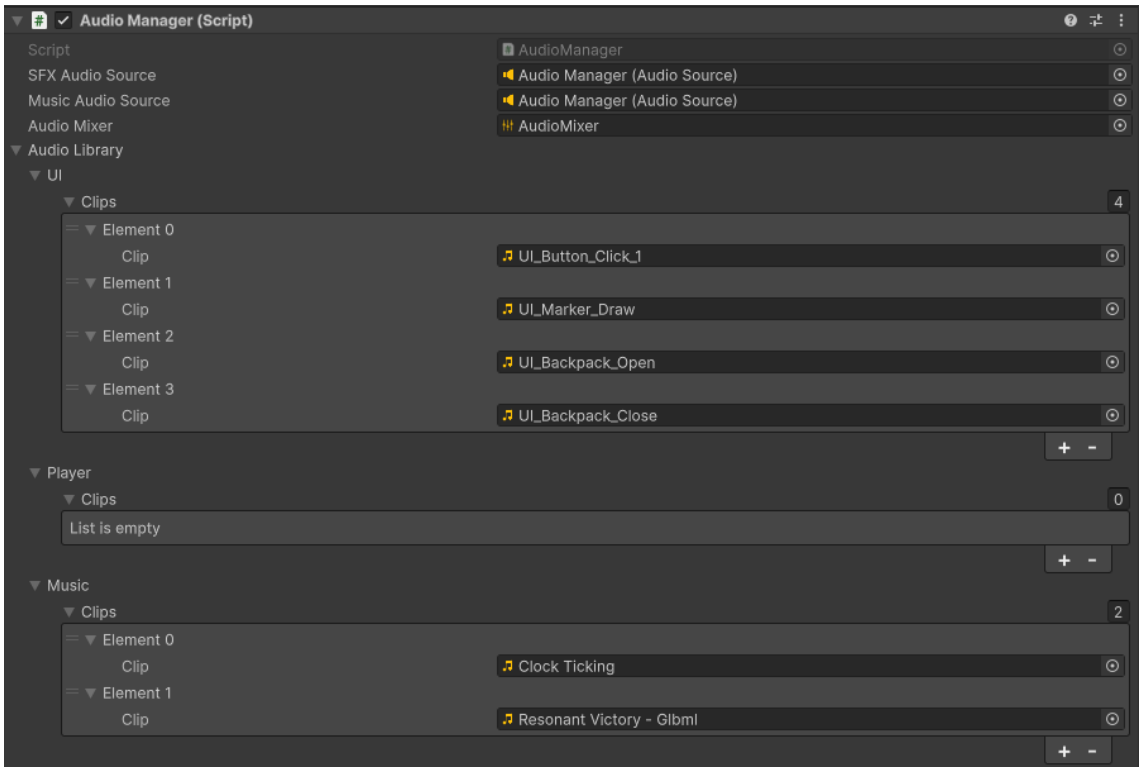
4.1.3 Ovládání hlasitosti

Hlasitost hudby i zvukových efektů je řízena pomocí vlastností MusicVolume a SFX-Volume, které převádějí hodnoty mezi rozsahem 0–1 (používaným v UI a souboru „settings.json“) a decibelovým rozsahem AudioMixeru (od -80–20 dB).

Uživatelské nastavení hlasitosti se načítá z uloženého souboru „settings.json“ (popsáno v Kapitole 4.7) a aplikuje ihned po spuštění hry.

4.1.4 Kategorie a knihovna zvuků

Zvuky jsou uloženy ve třídě `AudioLibrary`, která je rozdělena do několika kategorií (UI, Player, Music). Každá kategorie je reprezentována třídou `AudioCategory`, která obsahuje seznam pojmenovaných klipů NamedAudioClip. Příklad definování této třídy je na Obrázku 4.1.



Obrázek 4.1: Rozdělení zvukových souborů do kategorií v editoru Unity

4.2 Lobby a navigace

V rámci herního světa existuje samostatná vrstva přechodových scén, označovaná jako lobby, která slouží jako rozhraní mezi hlavním menu a samotnými herními levely. Hráč se v této části může volně pohybovat podobně jako v klasickém levelu, přičemž jednotlivé lokace (např. obchod, vstupy do úrovní) jsou rozmístěny v herním prostředí a přístupné skrze interaktivní objekty. Tímto přístupem je nahrazeno

tradiční grafické uživatelské rozhraní, které v některých hrách plní stejnou funkci.

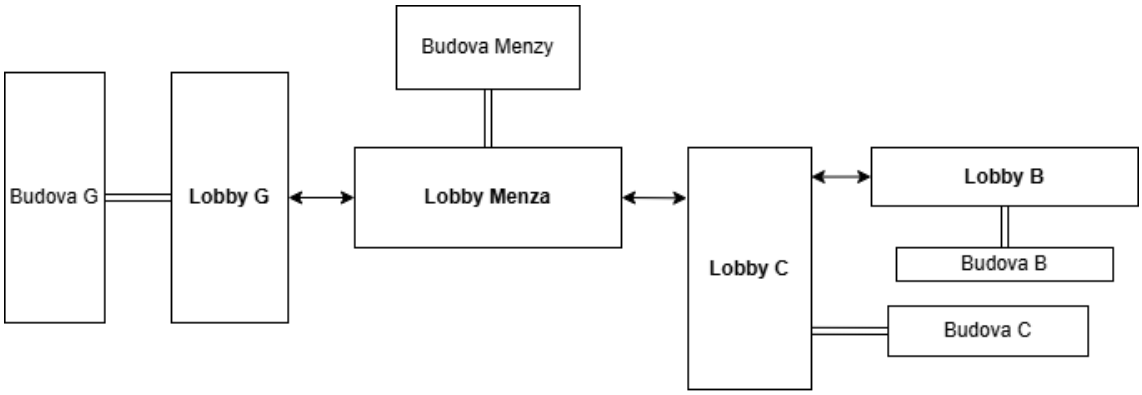
Mimo herního průvodce (viz Kapitola 4.3) pomáhá hráči s orientací v lobby také navigační systém ve formě šipky, která se zobrazuje nad postavou hráče. Tato šipka dynamicky ukazuje směr k nejbližšímu relevantnímu cíli (například vchod do dalšího levelu, obchod nebo collider do sousední scény), čímž zajišťuje plynulý a intuitivní pohyb po herním světě i pro hráče, kteří si lobby ještě neosvojili.

4.2.1 Lobby

Lobby se skládá ze čtyř samostatných scén, které dohromady vytvářejí jeden funkční celek. Jejich rozložení i vzhled čerpají inspiraci z reálného kampusu Technické univerzity v Liberci (znázorněno na Obrázku 4.2). Každá ze scén reprezentuje prostor před určitou budovou a slouží jako přechodový prostor mezi jednotlivými herními úrovněmi.

Jednotlivé lobby scény jsou následující:

- Lobby G
  - Obsahuje vstup do levelu G.
- Lobby Menza
  - Výchozí oblast při spuštění nové hry.
  - Slouží jako přechodová oblast mezi G a C a zároveň obsahuje přístup do samostatné scény Obchod, kde si hráč může nakupovat herní předměty.
- Lobby C
  - Propojuje menzu a budovu C.
  - Obsahuje vstup do levelu C.
- Lobby B
  - Poslední přístupová oblast, která vede k finálnímu levelu v budově B.



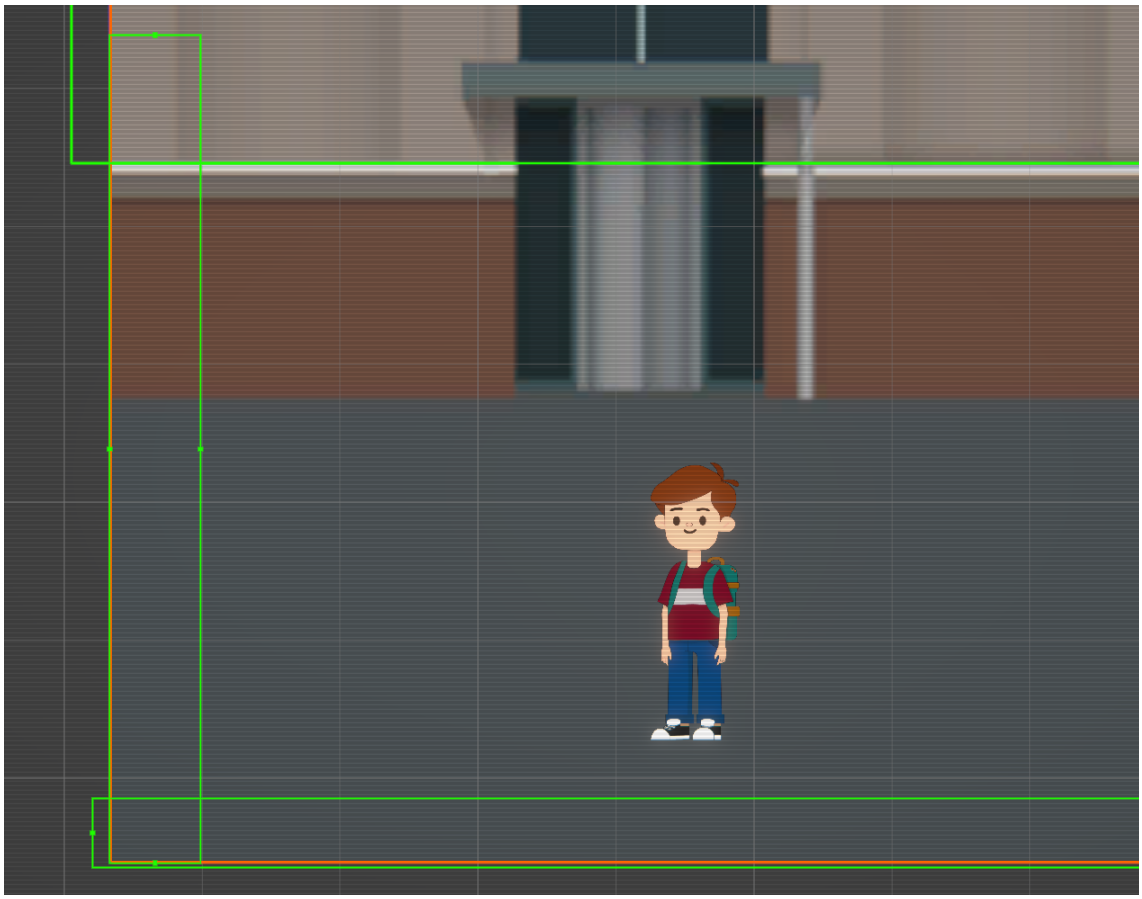
Obrázek 4.2: Schéma lobby

Pohyb hráče v rámci každé lobby scény je vymezen pomocí 2D Colliderů [19], které slouží jako neviditelné bariéry a zabraňují hráči opustit definovaný herní prostor (vyobrazeno v Obrázku 4.3). Kromě těchto okrajových colliderů obsahuje každá lobby scéna další specifické collidery, které zajišťují načítání navazujících scén. Každá lobby scéna má:

- Vstupní collider do budovy, která reprezentuje samostatný level (např. budova G, C, B) nebo speciální scénu (Obchod z Lobby Menza).
- Alespoň jeden přechodový collider, který umožňuje přesun do sousední lobby scény.

Počet přechodových colliderů odpovídá počtu přímo navazujících scén – například Lobby G sousedí pouze s Lobby Menza, a tedy obsahuje pouze jeden přechodový collider, zatímco Lobby Menza sousedí s Lobby G i Lobby C, a má tedy dva.

Přechod do jiné scény je realizován při kontaktu hráče s daným colliderem – po jeho vstupu se provede načtení cílové scény.



Obrázek 4.3: Collidery v lobby s menzou

4.2.2 Navigace

Pro snazší orientaci v lobby byl do hry implementován navigační systém ve formě šipky nad hlavou hráče. Úkolem šipky je vést hráče ke správnému cíli, aby mohl plynule pokračovat v herním postupu. Chování šipky se řídí podle toho, kde se hráč aktuálně nachází:

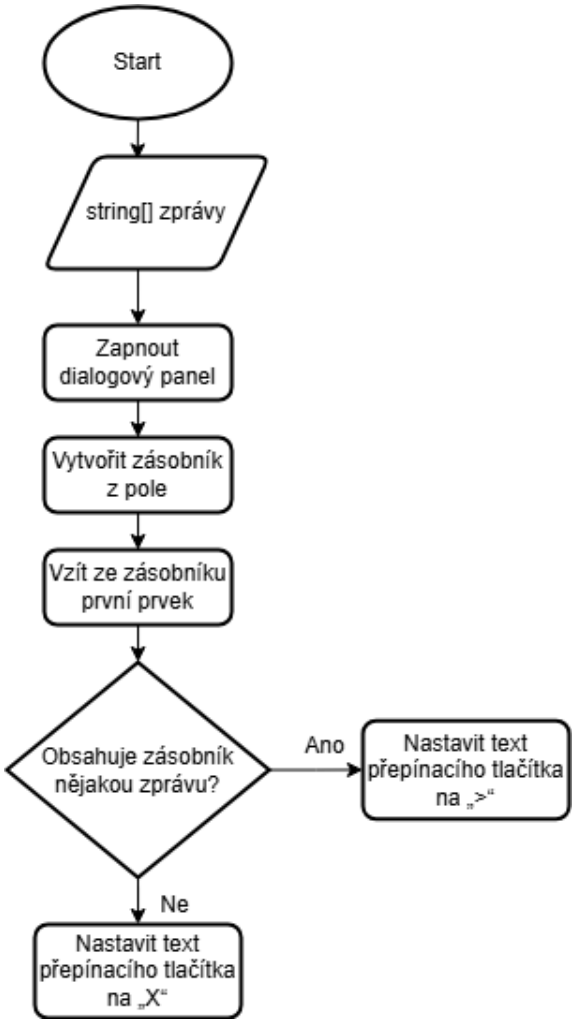
- Pokud hráč není ve scéně, kde se nachází cílové dveře (například do dalšího levelu), šipka ukazuje na přechodový collider, který po kontaktu načte sousední scénu vedoucí blíž k cíli.
- Pokud se hráč již nachází ve scéně, kde jsou cílové dveře, šipka ukazuje přímo na tyto dveře.

Tímto způsobem navigační systém postupně provádí hráče jednotlivými lobby scénami až k jeho aktuálnímu cíli, ať už jde o vstup do levelu nebo přístup do speciální lokace (např. obchod v menze).

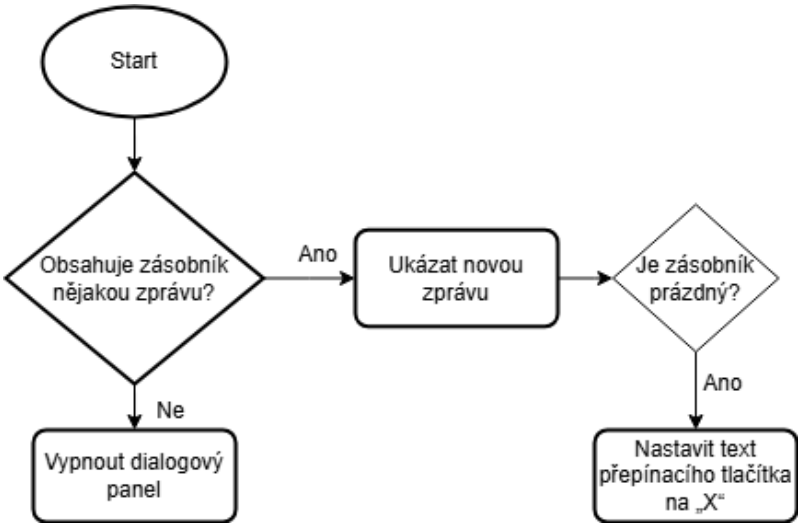
4.3 Hráčův průvodce

Herní průvodce představuje dynamický systém nápovědy, který hráče provází hrou a usměrňuje jeho postup. Pomáhá v lobby scénách a na klíčových místech herních levelů tím, že poskytuje instrukce, tipy a zpětnou vazbu formou krátkých zpráv.

Implementace systému zpráv je realizována pomocí zásobníku [83], do kterého je celá skupina zpráv uložena (viz Obrázek 4.4). Hráč postupně klikáním na tlačítko pro pokračování přehrává jednu zprávu za druhou (viz Obrázek 4.5), dokud v zásobníku nějaká zpráva zbývá. Po zobrazení všech zpráv se rozhraní automaticky skryje.



Obrázek 4.4: Vytvoření zásobníku



Obrázek 4.5: Ošetření kliknutí

4.3.1 Funkce průvodce v lobby

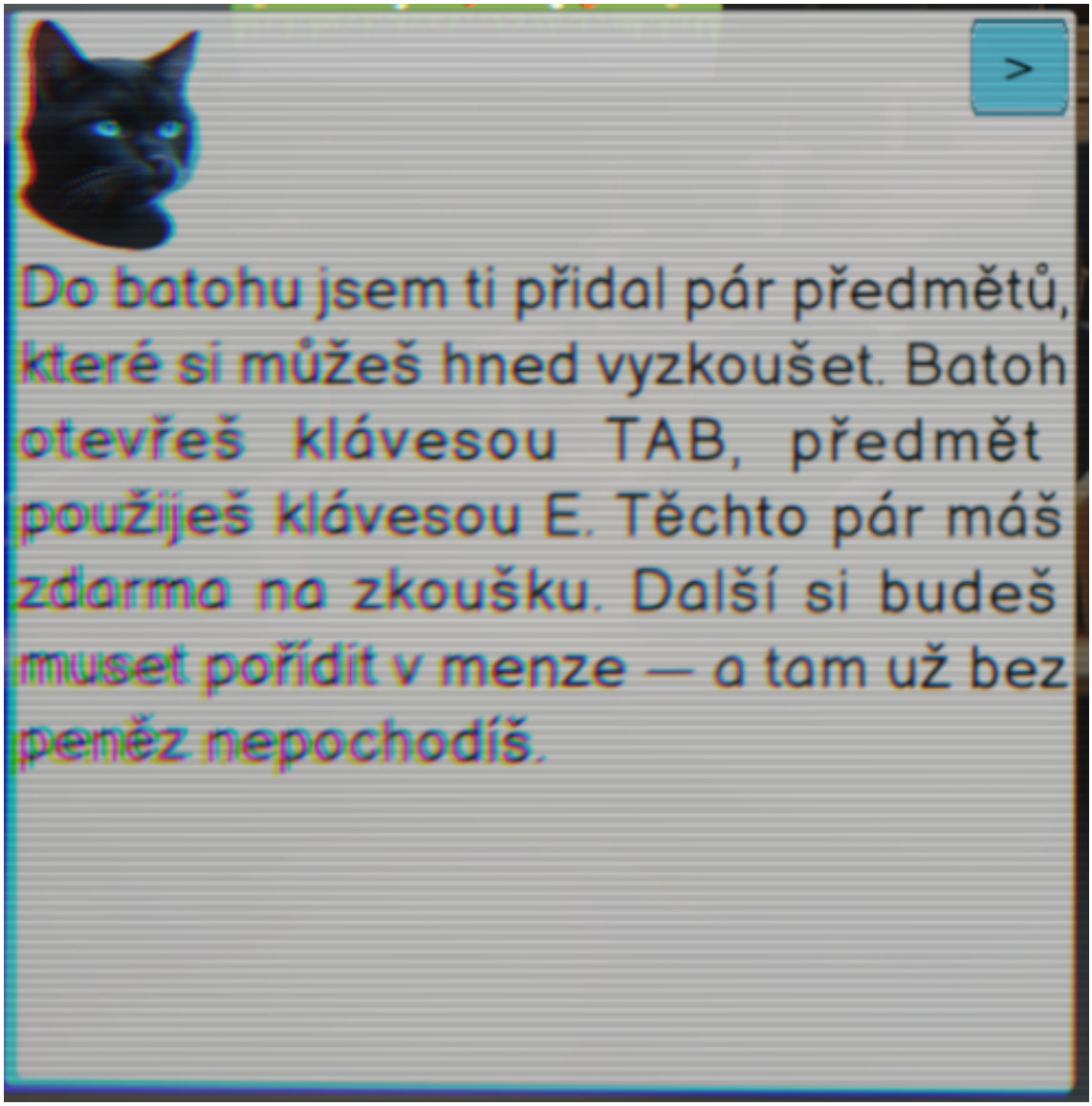
V lobby je úkolem průvodce nasměrovat hráče k dalšímu cíli a vysvětlit mu aktuální úkoly. Při vstupu do nové scény nebo při splnění určitého milníku se hráči zobrazí sekvence zpráv, které například:

- Shrnují, čeho právě dosáhl (např. dokončení levelu).
- Upozorňují na nově dostupné možnosti (např. otevřený obchod v menze).
- Udávají další cíl v lobby (např. přechod do jiné budovy).

4.3.2 Funkce průvodce v herních levelech

V jednotlivých herních levelech slouží průvodce k vysvětlení herních mechanik a motivaci hráče (viz Obrázek 4.6). Zprávy jsou kontextové a aktivují se při dosažení checkpointů nebo při vstupu do levelu. Typické informace, které hráč od průvodce v levelech dostane:

- Vysvětlení nových schopností (např. změna ročního období hlasem).
- Instrukce k ovládání hry (např. otevření batohu klávesou TAB, použití předmětu klávesou E, pohyb klávesami W, A, S, D nebo šipkami).
- Představení nových překážek (pohyblivé plošiny, vlaštovky, střílející automaty).
- Motivace a příběhové poznámky, které budují atmosféru hry.



Obrázek 4.6: Příklad herního dialogu

4.4 Herní předměty, měna a obchod

Ve hře jsou implementovány mechaniky, které přinášejí speciální herní předměty, jež hráč využívá k usnadnění dokončení jednotlivých levelů. Tyto předměty rozšiřují možnosti pohybu, interakce s prostředím a překonávání překážek.

Předměty lze získat dvěma způsoby: buď v průběhu jednotlivých levelů jako odměnu za postup, nebo nákupem v herním obchodě za herní měnu, kterou hráč během hraní získává.

4.4.1 Herní předměty

Hráč má během hry k dispozici sadu speciálních předmětů, které mu umožňují lépe se vyrovnávat s nástrahami prostředí a nepřáteli. Tyto předměty mají různorodé funkce – od podpory pohyblivosti přes útočné schopnosti až po nástroje pro lepší orientaci v prostoru. Každý herní předmět má přiřazenou sadu vlastností, které určují jeho charakter a využití ve hře:

- Název – hlavní jméno předmětu zobrazované v uživatelském rozhraní.
- Podtitul – krátké heslo vystihující komickým způsobem hlavní účel předmětu.
- Popis – podrobnější vysvětlení funkcionality a doporučeného použití.
- Cena za kus – hodnota v herní měně potřebná pro zakoupení jednoho kusu.
- Doba účinku – délka trvání efektu v sekundách, pokud je efekt časově omezen; u jednorázových předmětů je hodnota nastavena na -1.
- Ikona – grafická reprezentace předmětu zobrazovaná v obchodě a inventáři.

Přehled dostupných předmětů:

- Kreslířův pomocník
  - Umožňuje hráči házet propisky jako projektily.
  - Zasažení nepřítele způsobí poškození, zásah nepřátelského projektilu vede ke zničení obou střel.
- Dimenzionální perla
  - Po hození umožňuje okamžitý teleport na místo dopadu.
  - Vhodné pro rychlé úniky nebo překonávání obtížných překážek.
- Písek času
  - Po aktivaci vrátí hráče na pozici, kde se nacházel před pěti sekundami.
  - Okolní prostředí zůstává nezměněné.
- Bleskoboty
  - Dočasně výrazně zvyšují rychlost pohybu hráče.
  - Užitečné pro rychlé přesuny a úniky, avšak zvyšují náročnost ovládání.
- Orlí oči
  - Dočasně oddálí kameru, čímž poskytuje širší rozhled a lepší přehled o okolním prostředí.

Každý předmět je vizuálně reprezentován unikátní ikonou a při aktivaci se jeho efekt okamžitě projeví. Předměty s časově omezeným účinkem (Bleskoboty, Skoková pružina, Orlí oči) mají předem definovanou délku působení, po jejímž uplynutí efekt přestane účinkovat.

Každý z herních předmětů implementuje za tímto účelem rozhraní [52] IUsableItem, díky kterému celý systém funguje modulárně a přidávání nových předmětů nevyžaduje zásahy do stávající struktury.

```
public interface IUsableItem
{
    public void InitializeItem();
    public void UseItem();
    public void UnselectItem();
}
```

Metody rozhraní mají následující funkcionalitu:

- InitializeItem
  - Volána při výběru předmětu v inventáři.
  - Zavádí potřebné závislosti (např. referenci na objekt hráče), spouští další požadované procesy (například u předmětu „Písek času“ začíná sledování hráčovy pozice) a přihlašuje se k potřebným událostem.
- UseItem
  - Volána při stisku klávesy pro použití aktivního předmětu.
  - Zajišťuje vlastní použití předmětu – například hráč hodí perlu, aktivuje zvýšení rychlosti apod.
- UnselectItem
  - Volána ve chvíli, kdy si hráč vybere jiný předmět nebo když aktuální předmět vyčerpá (má od daného typu 0 kusů).
  - Metoda provádí odhlášení ze všech připojených událostí a ukončuje aktivní procesy (například zastaví sledování pozice hráče u předmětu „Písek času“).

4.4.2 Herní měna

Nejjednodušší způsob získávání herních předmětů je prostřednictvím obchodu v menze, kde si hráč může zakoupit všechny dostupné předměty za herní měnu. Herní měnu je možné získat několika způsoby:

- Za dokončení herního levelu, jako součást odměny za splnění úkolů a postup ve hře.

- Z truhel, které jsou rozmístěné v jednotlivých levelech a obsahují buď herní měnu, nebo přímo některé herní předměty.
- Sebráním volně dostupných objektů, které se v levelech v průběhu hry objevují na různých místech.

Aktuální stav herní měny je pro hráče neustále viditelný v pravém horním rohu obrazovky, a to jak při pohybu v lobby, tak během samotných herních levelů. Tím má hráč neustálý přehled o svých finančních možnostech a může lépe plánovat nákupy v obchodě nebo využívání herních předmětů.

4.4.3 Herní obchod

Herní obchod, přístupný z lobby v lokaci Lobby Menza, slouží jako místo, kde si hráč může za herní měnu nakupovat předměty, které mu usnadní postup hrou. Obchod má jednoduché a přehledné uživatelské rozhraní rozdělené do dvou částí (viz Obrázek 4.7):

- Nabídka obchodu – seznam všech dostupných předmětů, které si hráč může zakoupit.
- Hráčovy předměty – seznam předmětů, které již hráč vlastní, včetně jejich aktuálního počtu.

Po kliknutí na jakýkoliv předmět se zobrazí podrobné informace v detailním panelu:

- název,
- podtitul,
- popis,
- cena (u položek v obchodě),
- množství (u položek v inventáři).

Nákup probíhá následovně:

- Hráč vybere předmět ze seznamu nabídky.
- Pokud má dostatek herní měny, stiskne tlačítko „Koupit“.
- Po nákupu se cena předmětu odečte z aktuálního stavu herní měny a předmět se přidá do hráčova inventáře.
- Inventář a stav měny se okamžitě aktualizují a uloží.

Při pokusu o nákup bez dostatku měny je hráč upozorněn chybovým zvukem a nákup se neprovede.



Obrázek 4.7: Herní obchod

4.4.4 Herní inventář

Mimo herního obchodu lze zobrazit přehled vlastněných předmětů i v libovolné lobby scéně a v každém z herních levelů. V lobby je účel čistě informativní. V levelech inventář také zobrazuje informace o předmětech, ale po kliknutí na ikonu předmětu je předmět zvolen jako aktivní (metoda `InitializeItem`) a po stisknutí klávesy E se předmět použije (metoda `UseItem`).

4.5 Prostředí úrovní

Hlavní obsah hry tvoří tři samostatné levely, jejichž cílem je dostat se z výchozího bodu na druhý konec mapy.

Hráč má během průchodu levely k dispozici herní předměty, které mu mohou výrazně pomoci překonat překážky a usnadnit dosažení cíle.

Levely jsou navrženy tak, aby hráče postupně prověřily v různých herních mechanikách. Kromě podpůrných prvků se v levelech nachází také různé typy překážek a nástrah, jejichž účelem je průchod levely ztížit a donutit hráče efektivně využívat dostupné prostředky.

4.5.1 Checkpointy

Každý level obsahuje několik checkpointů, které slouží jako body, kam se hráč může bezpečně vrátit po neúspěchu (např. po pádu mimo herní oblast).

Checkpoint je vizuálně reprezentován jako stolní kancelářská lampa, ze které vychází světlo (viz Obrázek 4.8).

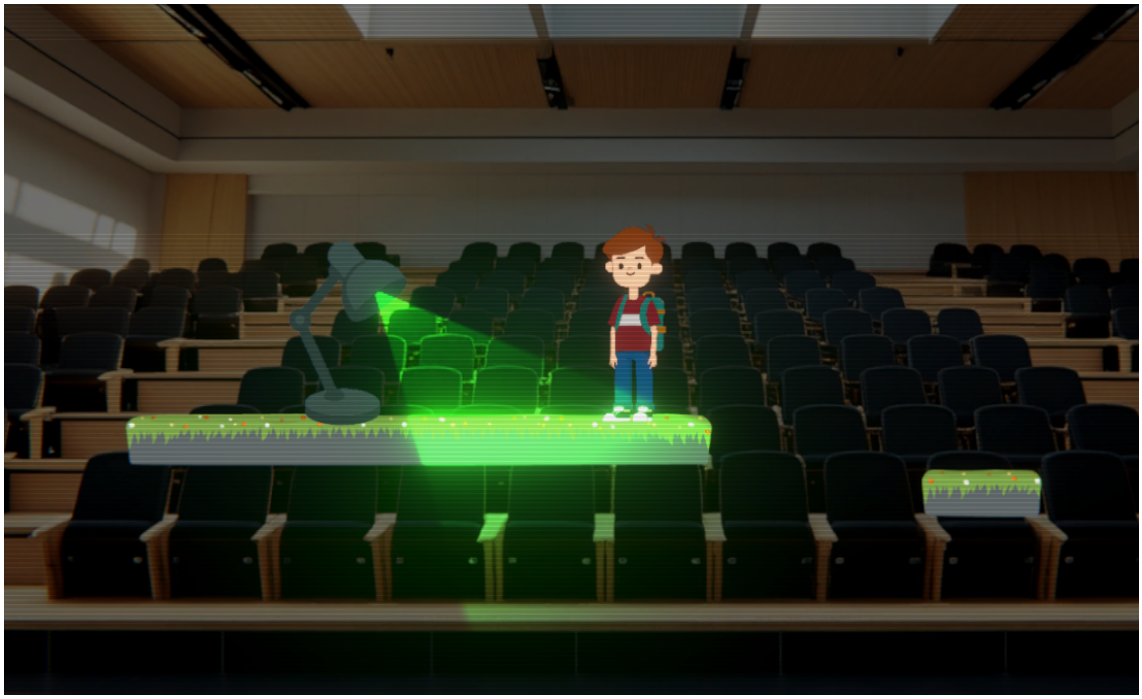
- Výchozí stav checkpointu: světlo svítí červenou barvou, což signalizuje, že checkpoint není aktivní.
- Po aktivaci checkpointu: světlo se změní na zelenou, aby hráč jasně viděl, že checkpoint byl úspěšně aktivován.

Funkce checkpointu:

- Při vstupu hráče do checkpointu, který ještě nebyl aktivován, se:
  - Deaktivuje případný dříve aktivovaný checkpoint (změní se jeho světlo zpět na červené).
  - Aktivuje nový checkpoint (světlo se změní na zelené).
- Současně se do herního postupu uloží aktuální pozice checkpointu.
- Tato pozice se využívá při:
  - Resetování pozice hráče po pádu mimo mapu.
  - Načítání levelu při opětovném spuštění hry.

Rozmístění checkpointů v levelech má zásadní vliv na výslednou obtížnost hry. Pokud je checkpoint umístěn až za delším a náročnějším úsekem, hráč je nucen zvládnout celou tuto část bez jediné chyby, což výrazně zvyšuje náročnost dané úrovně. Naopak častější rozmístění checkpointů umožňuje častější ukládání postupu a průchod hrou zjednodušuje.

Množství checkpointů a jejich strategické umístění jsou proto důležitým nástrojem pro vyvažování herní obtížnosti a ovlivnění celkového herního zážitku.



Obrázek 4.8: Příklad aktivního checkpointu

4.5.2 Průchozí plošiny

Hlavním stavebním prvkem herních levelů jsou plošiny, po kterých se hráč pohybuje pomocí skákání.

Plošiny se v levelu vyskytují v několika variantách. Nejčastějším typem jsou průchozí plošiny, které umožňují pohyb jak nahoru, tak dolů.

Pokud hráč vyskočí směrem vzhůru a narazí na takovou plošinu zespodu, automaticky jí proskočí a dostane se nad ni. Pokud se na plošině nachází a stiskne šipku dolů nebo klávesu S, propadne se skrz plošinu směrem dolů.

Tento typ pohybu umožňuje dynamičtější a flexibilnější průchod levelem a zároveň rozšiřuje možnosti, jak se hráč může dostat v levelu výš.

Průchozí plošiny jsou implementovány pomocí komponenty PlatformEffector2D [73], kterou Unity poskytuje.

Ve chvíli, kdy hráč stiskne šipku dolů, dojde k dočasnému otočení efektoru o 180 stupňů, což způsobí, že hráč plošinou propadne. Po krátkém časovém intervalu (0,4 sekundy) se efektor automaticky vrátí zpět do původní polohy, aby se plošina opět chovala jako pevná.

4.5.3 Pohyblivé plošiny

V některých levelech se nacházejí také pohyblivé plošiny, které se automaticky přesouvají mezi dvěma předem definovanými body.

Tyto plošiny slouží k překonávání větších vzdáleností, dostávání se na jinak nepřístupná místa nebo tvoří součást překážek, které musí hráč správně načasovat.

K pohybu se využívá funkce `Vector2.MoveTowards` [98], která zajišťuje hladký, rovnoměrný přesun bez trhání nebo prudkých změn rychlosti.

Pohyblivé plošiny výrazně rozšiřují možnosti designu levelů a kladou na hráče větší nároky na načasování skoků a plánování pohybu.

4.5.4 Praskající plošiny

Některé plošiny v levelech nejsou stabilní a postupně praskají při kontaktu s hráčem. Ve chvíli, kdy hráč na plošinu skočí nebo na ní přistane, dojde ke zvětšení praskliny (viz Obrázek 4.9). Tento proces má několik fází:

- První kontakt – plošina začne jemně praskat, objeví se první malé trhliny.
- Druhý kontakt – trhliny se zvětší, praskání je výraznější.
- Třetí kontakt – plošina je silně popraskaná, její stabilita je narušená.
- Poslední kontakt – plošina se úplně zhroutí a na určitou dobu zmizí z herního světa.

Po určitém čase se plošina automaticky obnoví a hráč ji může znovu použít. Během svého zhroucení je plošina neviditelná a fyzicky neexistuje, což hráči znemožňuje na ni vstoupit.

Tento typ plošin vytváří v levelech dynamické situace, kdy hráč musí správně načasovat své pohyby, aby se vyhnul pádu, a zároveň plánovat trasu s ohledem na dočasnou nedostupnost některých cest.



Obrázek 4.9: Praskající platforma

4.5.5 Truhly s předměty

V levelech se nacházejí truhly, které obsahují různé odměny v podobě herních předmětů nebo herní měny. Tyto truhly slouží jako motivační prvek pro průzkum levelu a zároveň pomáhají hráči vylepšovat jeho vybavení.

Princip fungování:

- Při kontaktu s hráčem se truhla automaticky otevře a spustí animaci otevření.
- Následně truhla začne ze svého středu postupně „vystřelovat“ obsah – herní předměty nebo mince (viz Obrázek 4.10).
- Předměty i mince jsou po otevření krátkou dobu nepřístupné (mají aktivní zpoždění sběru), aby nedošlo k okamžitému nasbírání všech položek během animace.
- Po uplynutí krátké prodlevy se předměty a mince stávají sebratelnými.

Každý vystřelený předmět:

- Po kolizi s hráčem se přidá do inventáře a je přehrán zvukový efekt potvrzující sebrání.
- Pokud jde o minci, zvýší se hráčovo množství herní měny a částka se ihned aktualizuje v UI v pravém horním rohu.

Každý vystřelený objekt, pokud není sebrán hráčem včas, automaticky zmizí po 25 sekundách, aby zbytečně nezaplňoval scénu.

Každá truhla má v názvu přiřazený unikátní index (např. „Chest (2)“), který ji jednoznačně identifikuje. Ve chvíli, kdy hráč truhlu poprvé otevře, se tento index uloží do seznamu „ChestsOpenedIndexes“ v aktuální konfiguraci levelu, která je součástí uloženého herního postupu. Při každém načtení levelu je zkontrolován seznam otevřených truhel. Pokud v seznamu nalezen odpovídající index, vyhledá v aktuální scéně truhlu s daným indexem a automaticky ji nastaví jako již otevřenou, aby zůstala v trvale otevřeném stavu. Tím je zajištěno, že hráč nikdy neotevře stejnou truhlu dvakrát a že obsah jednou otevřené truhly zůstane trvale nedostupný i po opětovném spuštění hry.



Obrázek 4.10: Truhla vystřelující předměty

4.5.6 Automaty

V levelech se nacházejí automaty, které v pravidelných intervalech vystřelují plechovky různých barev a efektů. Tyto plechovky fungují jako pohyblivé překážky – mohou hráče přímo poškodit nebo mu krátkodobě ztížit pohyb.

Automat v pravidelných (a mírně náhodných) intervalech vystřelí plechovku ve směru své orientace. Po každém výstřelu je náhodně vybrán jeden ze čtyř typů plechovek:

- Černá plechovka: při zásahu vrátí hráče zpět na poslední aktivní checkpoint.
- Modrá plechovka: na určitou dobu deaktivuje možnost používat předměty z inventáře.
- Zelená plechovka: na určitou dobu přiblíží kameru, čímž snižuje hráčův přehled o okolí.

- Červená plechovka: na omezenou dobu obrátí ovládání – stisknutí šipky vpravo nebo klávesy D způsobí pohyb doleva, šipka vlevo nebo klávesa A pohyb doprava, šipka nahoru nebo klávesa W způsobí pád dolů a šipka dolů nebo klávesa S vyvolá skok nahoru.

Každá plechovka po svém vystřelení zůstává ve hře pouze omezenou dobu (3–7 sekund), nebo zmizí po nárazu do hráče či překážky.

Po zásahu hráče se okamžitě aktivuje odpovídající efekt a plechovka se odstraní ze scény.

4.5.7 Vlaštovky

Papírové vlaštovky představují v levelech pohyblivou překážku, která funguje obdobně jako pohyblivá plošina – neustále se přesouvá tam a zpět mezi dvěma předem definovanými body (viz Obrázek 4.11).

Vlaštovka představuje rizikový prvek v průběhu hry. Pokud se jí hráč dotkne, je okamžitě vrácen na poslední aktivovaný checkpoint. Oproti tomu pokud hráč na vlaštovku zaútočí pomocí herních předmětů (například po ní hodí propisku), vlaštovka je zničena a na její místě se objeví mince, kterou může hráč sebrat. Tyto mince mají náhodně generovanou hodnotu v určitém rozmezí (po sebrání mince se přičte 5 až 10 jednotek herní měny).

Každá vlaštovka má v názvu svého rodičovského objektu uveden unikátní číselný identifikátor. V momentě, kdy hráč vlaštovku zničí, je tento identifikátor uložen do seznamu zničených vlaštovek v rámci aktuálního postupu hry. Při dalším načtení levelu se tento seznam načte a všechny vlaštovky, jejichž identifikátory v něm figurují, jsou automaticky odstraněny ještě před zahájením hry.



Obrázek 4.11: Vlaštovka

4.5.8 Rekurzivní koule

Papírové koule představují nejvyšší stupeň obtížnosti a objevují se výhradně v posledním levelu hry. Tyto objekty jsou generovány náhodně v oblasti nad levelem, odkud následně padají směrem dolů. Každá koule začíná s určitým počtem „životů“. Výchozí hodnota jsou 2 životy, v zimě jsou to 4 (viz Kapitola 4.6).

Pokud se hráč papírové koule dotkne, je okamžitě vrácen na poslední aktivovaný checkpoint. Pokud koule narazí na jiný objekt, dojde podle jejího aktuálního stavu k jedné ze dvou reakcí: pokud má koule zbývající počet životů větší než 0, rozdělí se po nárazu na dvě nové koule, které mají o jeden život méně. Každá z nových koulí pokračuje ve volném pohybu a může se dále rozdělit podle stejného pravidla, dokud se počet jejich životů nesníží na nulu. Koule bez zbývajících životů se při nárazu zničí.

Aby nedocházelo k zahlcení scény nekonečným množstvím objektů, má každá koule omezenou životnost – po určité době od svého vzniku se automaticky odstraní ze scény, i když se nezničí kvůli nedostatku životů.

Díky tomuto mechanismu se herní prostředí zaplňuje dalšími nebezpečnými objekty, což zvyšuje náročnost pohybu a nutí hráče k rychlým reakcím. Kromě přímého ohrožení hráče papírové koule také ovlivňují stav praskajících plošin – každý jejich náraz do plošiny způsobí, že se plošina více poškodí a přiblíží se jejímu rozbití.



Obrázek 4.12: Rozdvojené papírové koule

4.5.9 Déšť předmětů

Ve všech levelech se nachází speciální oblast rozprostřená nad herním prostředím, která slouží pro náhodné generování sebratelných objektů, podobně jako oblast pro generování papírových koulí.

Tato oblast využívá mechanismus náhodného výběru a rozmístění předmětů, mezi které patří jak všechny dostupné herní předměty, tak mince sloužící k získávání herní měny. Generování funguje na principu časovače: v pravidelných intervalech se v náhodném bodě této oblasti vytvoří nový objekt, přičemž výběr konkrétního předmětu je náhodný ze seznamu všech možných objektů.

Díky tomu je každá hra jiná a hráči se mohou v průběhu průchodu levelem setkávat s odlišnými příležitostmi ke sběru předmětů.

Pokud hráč nestihne předmět v daném časovém intervalu (25 sekund) sebrat, předmět ze scény zmizí, aby se předešlo zahlcení scény.

Tento systém podporuje plynulost hry a zároveň motivuje hráče k rychlejšímu pohybu a efektivnímu sběru, protože sebratelné objekty v levelech nejsou trvalé.

4.6 Mikrofon – Změna ročního období

Dalším herním aspektem, který zásadním způsobem mění dynamiku hry, je změna ročního období v levelu pomocí hlasitého zvuku zachyceného mikrofonem.

4.6.1 Ukládání volby mikrofonu do nastavení

Hráč si může v nastavení zvolit, jaký mikrofon chce k této funkci využívat (pokud uživatel nemá připojený mikrofon, funkce změny ročního období nebude dostupná). Seznam mikrofonů zprostředkovává Unity třída `Microphone` [57], díky které lze například získat seznam všech dostupných mikrofonů připojených k zařízení. Tato zařízení jsou poté v nastavení vloženy do rozbalovací nabídky a hráč si tak může zvolit, jaký mikrofon ke hře používá. Hráčova volba mikrofonu je uložena v souboru „settings.json“. Pokud si hráč žádný mikrofon nezvolí, je automaticky použit první dostupný (nebo žádný, pokud nemá hráč žádný připojený mikrofon).

4.6.2 Detekce hlasitého zvuku

Pro správu a zpracování zvukového vstupu ve hře slouží autorem vytvořená třída `MicrophoneManager`. Tato třída využívá také třídu `Microphone`. Tentokrát pro přístup k nahrávání zvuku z vybraného mikrofonu.

Po spuštění hry se `MicrophoneManager` pokusí najít a spustit nahrávání z mikrofonu uloženého v nastavení hry. Pokud není žádný mikrofon v nastavení vybrán, automaticky se použije první dostupné zařízení ze seznamu. Samotné nahrávání probíhá nepřetržitě ve smyčce s délkou jedné sekundy při vzorkovací frekvenci 44100 Hz [58].

V každém snímku dochází k vyhodnocení aktuální úrovně hlasitosti vstupního signálu. Tento proces probíhá následujícím způsobem:

- Z mikrofonového záznamu se načte malý úsek dat (o velikosti 128 vzorků).
- Z těchto dat se zjistí maximální absolutní hodnota amplitudy (tedy momentální intenzita zvuku).
- Pokud změřená hodnota překročí nastavený práh, je vyvolána událost OnLoudNoiseDetected.

Práh hlasitosti je nastavitelný a určuje, jak silný zvuk musí být, aby byl systémem rozpoznán jako hlasitý zvuk a spustil změnu ročního období.

Třída zároveň podporuje změnu aktivního mikrofonu během běhu hry pomocí metody `ChangeActiveMicrophone`, což umožňuje hráči kdykoliv přepnout mezi dostupnými vstupními zařízeními bez nutnosti restartování hry.

4.6.3 Změna ročního období

Ve chvíli, kdy hra detekuje hlasitý zvuk z mikrofonu, dojde ke změně ročního období. Přepínání probíhá sekvenčně v pevně daném pořadí – jaro, léto, podzim, zima – a není možné přeskočit rovnou na libovolné jiné období.

Změna ročního období výrazně ovlivňuje herní prostředí i chování prvků v levelech a stává se klíčovou herní mechanikou, která hráči otevírá nové možnosti nebo naopak komplikuje jeho postup. Každé roční období přitom přináší jedinečné podmínky a hodí se pro odlišné herní situace a strategie.

Jaro:

- Výhody:
  - Efekty předmětů trvají déle.
  - Efekty plechovek trvají kratší dobu.
- Nevýhody:
  - Pohybující se plošiny se hýbou rychleji.
  - V truhlách je při otevření menší odměna.
  - Automaty střílejí rychleji.

Léto:

- Výhody:
  - Hráč má k dispozici dvojitý skok.
  - Papírové koule padají méně často.
  - Ze stropu padají sebratelné předměty.
- Nevýhody:
  - Praskající plošiny se pomaleji obnovují.

  - Vlaštovky se pohybují rychleji.
  - Truhly nejdou otevřít a nejsou vidět.

Podzim: Na podzim začne foukat vítr, který ovlivňuje ostatní objekty ve scéně [3].

- Výhody:
  - Vítr odfoukne papírové koule.
  - Automaty nestřílejí.
- Nevýhody:
  - Vítr znesnadňuje pohyb.
  - Vítr odfoukne hozené předměty (propisky, perly).
  - Efekty plechovek trvají déle.

Zima: Na všechny plošiny je aplikován fyzikální materiál [72], díky kterému se hráč po platformách klouže, což pohyb znesnadňuje.

- Výhody:
  - Plošiny se pohybují pomaleji.
  - Plošiny nepraskají.
  - Automaty střílejí pomaleji.
  - V truhlách je při otevření větší odměna.
- Nevýhody:
  - Plošiny kloužou.
  - Efekty předmětů trvají kratší dobu.
  - Papírové koule se vícekrát rozdvojují.

Autor v práci řešil pouze změnu herních mechanik. Změna vizuální stránky hry (práce se shadery, vypínání a zapínání efektu listí, sněžení) je řešena v práci Dominiky Strečanské [84].

4.7 Systém ukládání herního postupu a nastavení

Ukládání herního postupu a uživatelského nastavení představuje jednu z klíčových mechanik, které zajišťují kontinuitu herního zážitku. Díky této funkci hráč nepřichází o svůj aktuální stav ani v případě ukončení nebo pádu hry. Data o postupu i nastavení jsou serializována do formátu JSON (zápis objektů v JavaScriptu) a ukládána lokálně do dvou souborů na uživatelské zařízení. K serializaci a deserializaci dat je využita třída JsonUtility z knihovny UnityEngine [54], která umožňuje jednoduchou konverzi objektů do formátu JSON

a zpět. Tento formát je čitelný nejen pro stroj, ale i pro člověka, což výrazně usnadňuje ladění a testování během vývoje. Konkrétní umístění souborů závisí na cílové platformě. Na testované platformě Windows 10/11 jsou tyto soubory umístěny ve složce „`C:\Users\[uživatelské_jméno]\AppData\LocalLow\FM TUL Smutny Strecanska\TULtimátní hra"`, která odpovídá hodnotě neměnné proměnné `Application.persistentDataPath` v Unity [1].

4.7.1 Obsah souboru

Soubor JSON s herním postupem obsahuje následující informace:

- Index scény, která se má načíst při zvolení možnosti „Načíst hru“ v hlavním menu.
- Stav hry v podobě číselného identifikátoru, který určuje, zda se hráč nachází v konkrétním levelu, nebo je na cestě do určité budovy.
- Aktuální počet herní měny.
- Konfigurace aktuálního levelu, která zahrnuje:
  - Index konfigurovaného levelu.
  - Souřadnice hráče, na kterých se má objevit po načtení levelu.
  - Seznam indexů již otevřených beden.
  - Seznam indexů zničených vlaštovek.
  - Stav hráčova inventáře před vstupem do levelu.
  - Množství herní měny, kterou hráč vlastnil před vstupem do levelu.

Soubor s nastavením slouží k uchování preferencí hráče mezi jednotlivými spuštěními hry. Obsahuje následující informace:

- Nastavenou hlasitost zvukových efektů a hudby, vyjádřenou hodnotou v rozsahu 0 až 1.
- Informaci, zda se má hra spustit v režimu celé obrazovky.
- Informaci, zda se má zobrazovat počet FPS (Frames Per Second – snímky za sekundu).
- Informaci, zda má být zapnutá vertikální synchronizace.
- Název vybraného mikrofonu, který je používán ve hře při interakci s mechanikami ročního období.

4.7.2 Příklad obsahu souborů

V Příloze A je znázorněn příklad obsahu souboru „progress.json“ ve chvíli, kdy se hráč nachází uprostřed rozehraného levelu.

Následující příklad ukazuje obsah souboru „settings.json“, který zachycuje preferovaná nastavení hráče:

```
{
    "SFXVolume": 0.8096692562103272,
    "MusicVolume": 0.43804001808166506,
    "IsFullScreen": false,
    "IsShowingFPS": true,
    "VSync": false,
    "Microphone": "Mikrofon (Samson Q2U Microphone)"
}
```

5 Otestování funkčnosti hry

Aby bylo možné ověřit stabilitu, bezchybnost a uživatelskou přívětivost výsledné hry, byla v rámci závěrečné fáze vývoje prováděna série uživatelských testů. Testování bylo založeno na přesně definovaných testovacích scénářích, které pokrývají klíčové části hry – od základní interakce v menu, přes průchod jednotlivými levely až po použití herních předmětů. Kompletní návod k testování je k dispozici ve veřejném repozitáři: https://github.com/SmutnyJan/PDO-semestral-project/tree/main.

Testování se zúčastnilo celkem 8 osob. Každý tester měl k dispozici příručku s detailními pokyny, která obsahovala technické požadavky, návod ke spuštění hry a popis práce se soubory logů a uložených herních stavů. Díky tomu bylo možno zaručit, že testování probíhá konzistentně, opakovatelně a za stejných podmínek napříč jednotlivými zařízeními i testery. Příručka rovněž obsahovala přehled používané terminologie a jednotný systém kategorizace chyb podle jejich závažnosti – od neintuitivního chování až po kritické pády hry.

Tester si nejprve stáhnul z GitHubu verzi hry *1.1*, na které testování probíhalo, a podle pokynů otevřel odpovídající adresář s logy a uloženou hrou. Následně měl za úkol hru dohrát až do závěrečných titulků. Toto prvotní hraní mohlo skončit buď úspěšným dokončením, nebo detekcí chyby. Pro záznam výsledků testování sloužily dva samostatné online formuláře:

- Protokol o chybě – vyplňuje se v případě, že tester narazí na jakýkoliv problém v chování hry.
- Protokol o úspěchu testovacího scénáře – vyplňuje se po bezchybném splnění scénáře dle zadání.

5.1 Obsah formulářů

Po dokončení každého testovacího scénáře bylo po testerech vyžadováno, aby vyplnili buď protokol o chybě, pokud se scénář nepodařilo dokončit, nebo protokol o úspěchu testovacího scénáře, pokud test proběhl bez problémů. Oba formuláře obsahovaly jméno a příjmení testera a název testovacího scénáře, kterého se protokol týká. Protokol o chybě navíc zahrnoval typ chyby, její podrobný popis a kroky potřebné k reprodukci problému.

5.2 Typy chyb

V rámci testování byly identifikované problémy kategorizovány podle jejich závažnosti. Tento systém kategorizace umožňoval rychle rozlišit, jaký dopad má daný problém na hratelnost a stabilitu hry, a následně určit jeho prioritu při opravě. Každá chyba byla při hlášení označena jednou z následujících úrovní:

5.2.1 Neintuitivní chování

Chování hry, které není technicky chybné, ale může být pro některé hráče matoucí či neočekávané. Tyto případy sice nenarušují průběh hry, ale snižují její uživatelskou přívětivost.

Příklady chyby:

- Možnost odeslat test bez zaškrtnuté odpovědi.
- Přeskakování cutscén i pomocí jiných kláves, než je uvedeno v návodu (např. E nebo NUM ENTER).
- Automatické odstranění předmětu z inventáře po jeho úplném spotřebování.

5.2.2 Lehká chyba

Chyba, která má drobný negativní vliv na hratelnost, ale hráč může bez problémů pokračovat dál.

Příklady chyby:

- Krátkodobé zaseknutí postavy v textuře.
- Tlačítko zareaguje až na druhý pokus.
- Postava se nemůže pohybovat, dokud nevyskočí.

5.2.3 Těžká chyba

Chyba, která zásadně narušuje hratelnost a brání v pokračování, přesto však nedojde k pádu hry.

Příklady chyby:

- Nefunkční průchod z levelu.
- Tlačítko nereaguje vůbec.
- Možnost používat neomezeně předměty.
- Nefunkční ukládání hry.
- Vypnutí nebo absence hudby.

5.2.4 Kritická chyba

Chyba, která způsobí pád celé aplikace. Má nejvyšší prioritu pro opravu.

Příklady chyby:

- Pád hry po použití herního předmětu.
- Pád hry po kliknutí na konkrétní tlačítko.

5.3 Nalezené chyby

Během testování identifikovala testovací skupina několik chyb a zároveň poskytla cenné podněty vedoucí ke zlepšení celkového herního zážitku.

5.3.1 Snížení výchozí hlasitosti

Prvním z návrhů na zlepšení bylo snížení výchozí hlasitosti hry. Testeři opakovaně zmiňovali, že výrazně neocení, když je po spuštění nové hry okamžitě přivítá extrémně hlasitý zvuk či hudba. Přestože testovaná hra nebyla v tomto ohledu takto extrémní, i tak byla podle jejich slov zbytečně hlasitá, a doporučili proto výchozí hlasitost mírně snížit, aby byl úvodní dojem příjemnější – zejména při hraní se sluchátky.

Při prvním spuštění, kdy ještě není vytvořen konfigurační soubor „settings.json“, hra přebírá výchozí hodnoty z komponenty AudioMixer [9]. V původní verzi byla hlasitost hudby i speciálních efektů nastavena na –20 dB. Na základě zpětné vazby byla tato hodnota upravena na –40 dB, čímž se problém podařilo efektivně vyřešit.

5.3.2 Možnost sběru předmětů v podzimním období

Jedním z podnětů testerů bylo zpřístupnění sběru předmětů během podzimního období, konkrétně v situaci, kdy jsou předměty vystřeleny z truhly a ihned ovlivněny silným větrem. V původní implementaci truhla po otevření spouštěla krátkou animaci „fontány“, při níž byly jednotlivé předměty postupně vyhazovány různými směry. Během této animace, která trvala přibližně 1,5 sekundy, nebylo možné předměty sebrat, aby nedošlo k jejímu narušení. Tento přístup je však nevhodný v případě podzimního období, kdy vlivem větru animace reálně neprobíhala – předměty byly okamžitě odfouknuty mimo dosah hráče ještě před tím, než bylo možné je sebrat.

Na základě zpětné vazby byl tedy doplněn výjimkový mechanismus, který v případě, že je aktivní období podzim, zruší zpoždění sběru a umožní okamžitou interakci s předměty. Díky tomu může hráč předměty sebrat i ve chvíli, kdy jsou ihned po vytvoření ovlivněny herním větrem – za předpokladu, že se postaví do jejich trajektorie.

5.3.3 Zvuková odezva při zásahu plechovkou

Během testování se ukázalo, že zásah červenou plechovkou může být pro hráče matoucí, protože chyběla jakákoli okamžitá zpětná vazba. Tato plechovka způsobuje prohození ovládání, ale bez zvukového nebo vizuálního efektu nebylo na první pohled zřejmé, že k zásahu skutečně došlo. U ostatních typů plechovek se takový problém neprojevil:

- Černá způsobí okamžité přemístění na checkpoint.
- Modrá zablokuje používání předmětů, což se projeví červeným křížem na ikoně vybraného předmětu.
- Zelená změní přiblížení kamery, což je viditelné ihned.

Na základě této zpětné vazby byl přidán zvukový efekt při zásahu jakoukoli plechovkou, čímž se zajistila konzistentní odezva napříč všemi typy a zlepšila se celková přehlednost situace ve hře.

5.3.4 Špatné dialogy v budově B

V závěrečné fázi testování se v posledním herním levelu objevila chyba, kdy všechny checkpointy zobrazovaly nesprávné dialogy. Příčina tohoto problému spočívala ve způsobu práce s prefaby v Unity [74]. Prefab představuje šablonu objektu, jejíž instance sdílejí stejné nastavení a logiku, dokud nejsou přepsány lokálně.

V projektu byl použit jeden obecný prefab pro checkpoint, jehož součástí byl i vstupní seznam zpráv zobrazovaných hráči formou dialogu. Při úpravě jednoho konkrétního checkpointu autor omylem klikl na volbu „Apply“, čímž došlo k přepsání všech instancí tohoto prefabu novými (nesprávnými) zprávami. Přestože většina scén byla následně opravena ručním vrácením původních hodnot, v případě levelu B ke kontrole nedošlo a chyba zůstala neodhalena až do fáze testování.

5.3.5 Délka zobrazení titulků

Jedním ze závažnějších nedostatků odhalených během testování byla příliš krátká doba zobrazení titulků ve všech třech cutscénách. Ačkoli se nejedná o chybu v herní mechanice, mohla mít zásadní dopad na pochopení příběhu – hráči často nestihli přečíst celý text dříve, než zmizel. Vzhledem k tomu, že titulky jsou ve hře zobrazovány pomocí komponenty Timeline [90], kterou Unity nabízí právě za účelem tvorby cutscén, bylo nutné ručně upravit délku každého jednotlivého klipu, ve kterém se daný text zobrazoval. Na základě zpětné vazby testerů byla délka zobrazení všech titulků prodloužena tak, aby byl obsah srozumitelný i při běžném tempu čtení.

6 Vyhodnocení výpočetní náročnosti hry

Pro účely této práce bylo potřeba ověřit, jak výpočetně náročná je výsledná hra na běžném koncovém zařízení. Jako hlavní metrika byla zvolena hodnota snímků za sekundu, protože ji lze snadno a efektivně měřit i bez použití externích nástrojů. Měření probíhalo přímo ve hře, kde je implementován jednoduchý ukazatel FPS a funkce pro automatické zaznamenávání hodnot do logu.

Vybraný přístup umožňuje zaznamenávat výkon hry v různých částech – od hlavního menu přes nastavení až po samotný herní level. Získaná data poskytují přehled o tom, jak plynule hra běží za běžných podmínek a jaké nároky klade na koncový hardware. Měření probíhalo dle předem definovaného scénáře, aby byla zajištěna konzistence a srovnatelnost výsledků.

Společně s hodnotami snímkové frekvence v jednotlivých scénách hra automaticky zaznamenávala i další relevantní údaje, jako jsou hardwarová specifikace koncového zařízení (typ procesoru, grafická karta, velikost operační paměti), testovaná verze hry a datum provedení testu. Celý proces měření probíhá podle přesně definovaného scénáře *Měření FPS* (viz https://github.com/SmutnyJan/PDO-semestral-project/blob/main/Sc%C3%A9n%C3%A1%C5%99e/M%C4%9B%C5%99en%C3%AD%20FPS.md [81]), který zajišťuje jednotný postup všech testerů a minimalizuje riziko zkreslení výsledků.

6.1 Počáteční měření FPS

V první fázi testování jsou sbírána data z neoptimalizované verze hry. Testeři po dokončení měření odesílají logovací soubory, které jsou následně analyzovány. Pro zpracování dat je využit vlastní skript v jazyce Python (viz Příloha B), jenž z logů automaticky vypočítává průměrné hodnoty snímkové frekvence pro jednotlivé části hry: hlavní menu a nastavení, lobby menza a lobby G, a level G. Tento postup umožňuje rychlé vyhodnocení výkonu a identifikaci míst s potenciálně vyšší výpočetní náročností.

Další sledovanou metrikou byla celková velikost hotové hry, konkrétně velikost archivu obsahujícího sestavený spustitelný soubor spolu s ostatními potřebnými daty. Tento archiv je zároveň nahráván na GitHub jako vydaná verze hry a slouží k distribuci pro testery. Velikost hry má přímý dopad na dostupnost a pohodlí uživatelů – čím menší soubor, tím rychlejší stažení a menší požadavky na úložný prostor.

V rámci optimalizačních kroků bylo proto sledováno, jak se změny v kódu a ve

správě assetů projevují na velikosti výsledného archivu. Zmenšení velikosti sice přímo neovlivňuje výpočetní náročnost hry, ale představuje důležitý aspekt z pohledu uživatelské přívětivosti a udržitelnosti projektu.

6.2 Užité způsoby snížení výpočetní náročnosti

Ačkoli výsledná hra dosahovala i v neoptimalizované podobě dostatečného počtu FPS (za standard je považováno 60 FPS [51]) na všech testovaných zařízeních (snímková frekvence stabilně přesahovala 60 FPS), byla v souladu se zadáním práce provedena analýza a realizace základních optimalizačních kroků. Tyto úpravy přispívají nejen k efektivnějšímu běhu hry, ale také ke škálovatelnosti projektu – zajišťují připravenost na případné rozšíření herního světa, přidání nových mechanik či nasazení na slabších koncových zařízeních. V následující části jsou představeny použité postupy vedoucí ke snížení výpočetní náročnosti.

Většina použitých technik optimalizace vychází z publikace Optimize your game performance for consoles and PCs in Unity [67], která poskytuje praktické návody a osvědčené postupy pro snížení výpočetní zátěže v hrách vyvíjených v enginu Unity. Jednotlivé kroky uvedené v knize byly poté přizpůsobeny specifikům této konkrétní hry.

6.2.1 Odstranění zbytečného logování

Jako první optimalizační krok byla provedena analýza výskytu volání metody `Debug.Log` [22], kterou Unity poskytuje pro výpis ladicích informací – buď do konzole při spuštění v editoru, nebo do souboru „Player.log“ při spuštění sestavené hry. Přestože je tato metoda užitečná během vývoje, její časté použití může negativně ovlivnit výkon, zejména pokud je volána opakovaně v rámci často spouštěných funkcí (např. v `Update` [61]). Všechny nadbytečné výpisy byly proto odstraněny.

Výpisy, které zůstaly kvůli své užitečnosti pro ladění, byly ohraničeny blokem podmíněné kompilace [20], díky čemuž se ve finální verzi hry (sestavené jako .exe) vůbec nevyskytují a nemají tedy žádný dopad na výkon během hraní.

Příkladem takto podmíněně ošetřeného výpisu je kontrola správného pojmenování herních objektů typu truhla. Každá instance truhly v levelu musí mít název ve formátu „Chest (index)“, kde index je číselný identifikátor, jenž se při otevření truhly zapisuje do souboru „progress.json“. Tento záznam slouží k evidenci již otevřených truhel a zajišťuje, že se po opětovném načtení hry truhly při kontaktu s hráčem znovu neotevřou. Pokud název neodpovídá očekávanému formátu, mohl by být proces ukládání narušen.

Díky využití podmíněné kompilace se při spuštění hry v Unity editoru vypíše do konzole chybová hláška upozorňující na nesprávně pojmenovaný objekt. Ve finální sestavené hře se však tento výpis do souboru „Player.log“ již neprojeví, čímž se eliminuje zbytečná výpočetní režie i zahlcení logovacích dat.

```
#if UNITY_EDITOR
        if (!Regex.Match(this.name, @"^Chest \((\d+)\)$").Success)
        {
            Debug.LogWarning("Chest name " + this.name +
            " incorrect format!");
        }
#endif
```

6.2.2 Odstranění prázdných metod Update

Dalším krokem ke snížení výpočetní náročnosti bylo odstranění prázdných metod `Update` [61], které se v projektu vyskytovaly u některých komponent. V Unity je metoda Update volána každý snímek, a to i v případě, že její tělo neobsahuje žádný kód. Pokud je takových metod více, dochází ke zbytečnému volání na straně enginu a tím i k neefektivnímu využití výpočetních prostředků.

V rámci refaktoringu byly všechny tyto prázdné metody ze skriptů odstraněny. Tento krok sice nemá zásadní dopad na výkon v malých projektech, ale v kontextu škálovatelnosti a dobré praxe při optimalizaci kódu je jeho význam nezanedbatelný – zejména u her, které mohou obsahovat desítky nebo stovky aktivních komponent.

6.2.3 Optimalizace viditelnosti UI prvků pomocí komponenty Canvas

Další z provedených optimalizačních úprav se týkala efektivnějšího skrývání uživatelského rozhraní. Ve hře se nachází množství prvků, které nejsou na obrazovce zobrazené neustále – typicky například menu v levelu, inventář, dialogové okno nebo herní obchod. Tyto prvky jsou ve výchozím stavu „skryté“ a zobrazeny jsou až v určitém momentu podle kontextu hry.

Původní řešení spočívalo v tom, že byl celý daný prvek (například menu) při skrytí kompletně deaktivován. V Unity se každý prvek na scéně reprezentuje tzv. GameObjectem [37], což je základní stavební jednotka všech objektů ve hře – může jít o postavu, menu, zvuk nebo světlo. Každý GameObject může obsahovat různé komponenty, které definují jeho chování – u prvků uživatelského rozhraní je to například komponenta Canvas [16], která umožňuje jejich vykreslení na obrazovku.

K deaktivaci GameObjectu slouží metoda SetActive(false) [40], která způsobí, že daný objekt i se všemi svými komponentami přestane být ve scéně aktivní – hra ho přestane vykreslovat a přestane s ním i počítat. To je sice efektivní z hlediska výpočtu, ale problém nastane při jeho opětovném zobrazení: Unity musí celý prvek znovu připravit, včetně výpočtu jeho geometrie a dalších vlastností, což může způsobovat drobné zpomalení, především u složitějších UI struktur.

Optimalizace tedy spočívala v tom, že místo deaktivace celého GameObjectu se nyní skrývání řeší pouze vypnutím komponenty Canvas. Prvek tak zůstane aktivní ve scéně, ale nebude se vykreslovat. Výhodou je, že se jeho struktura nemusí při každém zobrazení znovu vytvářet, což vede k nižší výpočetní náročnosti a plynulejšímu chodu hry.

6.2.4 Převedení audio stop do mono režimu

Jedním z kroků ke snížení paměťové náročnosti hry bylo vynucení mono režimu pro některé zvukové efekty a hudbu. V Unity lze pomocí volby *Force To Mono* [7] v nastavení jednotlivých audio souborů převést stereo nahrávky na mono, což vede k podstatnému snížení velikosti výsledného audioklipu v paměti – typicky na polovinu.

Tato optimalizace byla aplikována u zvukových efektů a hudebních stop, kde prostorový vjem nehraje zásadní roli – tedy například u hudby na pozadí, zvuků uživatelského rozhraní nebo efektů, které jsou přehrávány přímo v reakci na událost spojenou s hráčem, jako je například zvuk zásahu plechovkou. Ten se ozývá pouze ve chvíli, kdy dojde ke kontaktu s hráčem, a je tedy vždy slyšen „z hráčovy pozice“, čímž stereo ztrácí význam. Naopak u zvuků, které mají působit prostorově (například ambientní šumy nebo směrové zvuky v herním prostředí), bylo stereo zachováno.

Je důležité zdůraznit, že tato změna nemá přímý dopad na snímkovou frekvenci (FPS), ale snižuje využití operační paměti během běhu hry, což je důležité zejména u méně výkonných zařízení. Úspora paměti může navíc nepřímo přispět ke stabilitě a plynulosti chodu hry, pokud je v systému dostupné úložiště kritickým faktorem.

6.2.5 Optimalizace sledování cíle pomocí cache pozic

V původní verzi skriptu se výpočet směru a rotace navigační šipky v lobby prováděl v každém snímku, kdy byla navigace aktivní. To znamenalo, že i když se objekt ani jeho cíl nepohnuly, stejně docházelo ke zbytečným výpočtům, což je neefektivní, obzvlášť při větším počtu objektů ve scéně nebo na slabším hardwaru.

Pro zlepšení výkonu byla do skriptu přidána jednoduchá optimalizace. Skript si nyní pamatuje poslední známou pozici cíle a vlastní pozici objektu. Před výpočtem nové rotace nejprve ověří, jestli se některá z těchto pozic změnila. Pokud zůstaly stejné jako v předchozím snímku, výpočet se přeskočí a rotace zůstane beze změny. Tím se výrazně snižuje množství výpočtů potřebných za sekundu, aniž by to ovlivnilo funkčnost.

6.2.6 Odstranění opakovaného načítání prefabů a optimalizace hledání

Byla provedena optimalizace výkonu u předmětů, které při použití vytvářejí nové objekty ve scéně. Původní implementace volala metodu `Resources.Load` [76] při každém použití předmětu, čímž docházelo k opakovanému načítání prefabů z disku. Tento přístup je nevhodný z hlediska výkonu, zejména pokud se předmět používá často. Nově se prefab načítá jednorázově při inicializaci a následně se uchovává v paměti, což výrazně snižuje zátěž a zrychluje reakci na akci hráče.

Zároveň byla nahrazena metoda `GameObject.FindGameObjectsWithTag` [38], která vytváří pole všech objektů s daným tagem, rychlejší alternativou `GameObject.FindWithTag` [39], která vrací první nalezený objekt bez další alokace.

Tím se zlepšila efektivita hledání objektů ve scéně a snížilo se riziko výkonnostních problémů v rozsáhlejších prostředích.

6.3 Naměřené hodnoty po optimalizaci

Optimalizace měla pozitivní dopad jak na výkon hry, tak na její výslednou velikost. Na všech testovaných zařízeních byl ve všech sledovaných částech hry zaznamenán nárůst průměrné snímkové frekvence, což potvrzuje efektivitu provedených změn. Kromě toho došlo také ke zmenšení distribuovaného archivu – velikost archivu s vydanou verzí hry klesla z původních 84,4 MB na 65 MB, což představuje snížení o přibližně 23 %. Výsledkem je nejen plynulejší běh hry, ale i rychlejší stahování a jednodušší správa verzí. Naměřené hodnoty snímků za sekundu lze jsou uvedeny v Přílohách (C.1, C.2, C.3).

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat hratelnou 2D plošinovou hru ve zvoleném herním enginu Unity a následně ověřit její funkčnost, výkonnost a uživatelskou přívětivost. V průběhu vývoje vznikl herní prototyp obsahující několik herních úrovní, interaktivní lobby, dialogový systém, sběr a používání předmětů, herní obchod, checkpointy a centrální mechaniku přepínání ročních období. Všechny verze hry jsou k dispozici ke stažení v repozitáři (v den odevzdání je nejaktuálnější verze číslo 1.2): https://github.com/SmutnyJan/BP-Unity-Smutny/releases. Zdrojový kód je k dispozici zde: https://github.com/SmutnyJan/BP-Unity-Smutny.

Součástí řešení bylo i vytvoření systému pro ukládání hry, práce s externími soubory JSON, návrh uživatelského rozhraní a tvorba herního prostředí. Výsledná hra byla podrobena sérii uživatelských testů, jejichž cílem bylo identifikovat chyby a získat zpětnou vazbu. Na základě připomínek testerů došlo k úpravám, které přispěly k lepšímu uživatelskému zážitku – např. prodloužení doby zobrazení titulků, přidání zvukové odezvy při zásahu projektily nebo oprava dialogových oken v některých levelech.

Důležitou součástí práce byla také analýza výpočetní náročnosti. Byla provedena měření snímkové frekvence na několika hardwarových konfiguracích a sledována velikost výsledného archivu hry. Přestože i neoptimalizovaná verze dosahovala stabilních hodnot FPS, následná optimalizace vedla ke zlepšení výkonu a zmenšení velikosti balíčku. Mezi hlavní techniky patřilo odstranění nadbytečných výpisů do konzole, zjednodušení práce s UI, použití mono zvukových stop tam, kde není potřeba stereo, a odstranění nevyužívaných metod volaných v každém snímku.

Unity se během vývoje ukázalo jako silný nástroj pro vývoj 2D her. Umožnilo nejen implementaci herní logiky a interaktivních prvků, ale také efektivní správu prostředí, zvuku, testování i optimalizace. Výsledná hra tak splňuje cíle práce a zároveň slouží jako praktický příklad využití enginu Unity pro tvorbu plošinovek.

Použitá literatura

[1] *Application.persistentDataPath* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Application-persistentDataPath.html.

[2] *AR* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://developers.google.com/ar.

[3] *Area Effector 2D reference* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/Manual/2d-physics/effectors/area-effector-2d-reference.html.

[4] *ARKit 6* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://developer.apple.com/augmented-reality/arkit/.

[5] *Asset Library* [online]. c2025. [cit. 2025-04-24]. Dostupné z: https://godotengine.org/asset-library/asset.

[6] *AssetStore* [online]. c2023. [cit. 2025-04-24]. Dostupné z: https://assetstore.unity.com/.

[7] *Audio Clip* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/Manual/class-AudioClip.html.

[8] *Audio Listener* [online]. c2025. [cit. 2025-04-08]. Dostupné z: https://docs.unity3d.com/Manual/class-AudioListener.html.

[9] *Audio Mixer* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/Manual/AudioMixer.html.

[10] *AudioSource* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/AudioSource.html.

[11] *AudioSource.Play* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/ScriptReference/AudioSource.Play.html.

[12] *AudioSource.PlayOneShot* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/AudioSource.PlayOneShot.html.

[13] *Battlefield 2042* [online]. c2025. [cit. 2025-04-20]. Dostupné z: https://www.ea.com/games/battlefield/battlefield-2042.

[14] *BBMOD | Make 3D games in GameMaker!* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://blueburn.cz/bbmod/.

[15] *C++ VS C# benchmarks* [online]. 2025. [cit. 2025-04-24]. Dostupné z: https://programming-language-benchmarks.vercel.app/cpp-vs-csharp.

[16] *Canvas* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/ScriptReference/Canvas.html.

[17] *Celeste* [online]. 2025. [cit. 2025-04-22]. Dostupné z: https://www.celestegame.com/.

[18] *Coding in UE: Blueprint vs. C++* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/coding-in-unreal-engine-blueprint-vs.-cplusplus.

[19] *Collider 2D* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/Manual/2d-physics/collider/collider-2d-landing.html.

[20] *Conditional compilation in Unity* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/Manual/platform-dependent-compilation.html.

[21] *CryEngine* [online]. c2025. [cit. 2025-04-08]. Dostupné z: https://www.cryengine.com/.

[22] *Debug.Log* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.1/Documentation/ScriptReference/Debug.Log.html.

[23] *Did you know that 60% of game developers use game engines?* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.slashdata.co/post/did-you-know-that-60-of-game-developers-use-game-engines.

[24] *FAB* [online]. c2024. [cit. 2025-04-23]. Dostupné z: https://www.fab.com/.

[25] *Fandom* [online]. 2023. [cit. 2025-04-21]. Dostupné z: https://www.fandom.com/.

[26] *FIFA 23* [online]. c2025. [cit. 2025-04-20]. Dostupné z: https://www.ea.com/cs-cz/games/fifa/fifa-23.

[27] *FINAL FANTASY VII REMAKE INTERGRADE* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://store.steampowered.com/app/1462040/FINAL_FANTASY_VII_REMAKE_INTERGRADE/.

[28] *Freemium: Definition, Examples, and Pros & Cons for Business* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.investopedia.com/terms/f/freemium.asp.

[29] *Frequently asked questions* [online]. c2014. [cit. 2025-04-24]. Dostupné z: https://docs.godotengine.org/en/stable/about/faq.html.

[30] *Frostbite* [online]. c2025. [cit. 2025-04-20]. Dostupné z: https://www.ea.com/frostbite/engine.

[31] *GameMaker* [online]. c2024. [cit. 2025-04-20]. Dostupné z: https://gamemaker.io/en.

[32] *GameMaker Language* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://manual.gamemaker.io/monthly/en/index.htm?%5C#t=GameMaker_Language.htm.

[33] *GameMaker Marketplace* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://marketplace.gamemaker.io/.

[34] *GameMaker Runtime Licence - Enterprise* [online]. c2024. [cit. 2025-04-24]. Dostupné z: https://gamemaker.io/en/legal/gamemaker-runtime-licence-enterprise.

[35] *GameMaker Runtime Licence - Free* [online]. c2024. [cit. 2025-04-24]. Dostupné z: https://gamemaker.io/en/legal/gamemaker-runtime-licence-free.

[36] *GameMaker Runtime Licence - Professional* [online]. c2024. [cit. 2025-04-24]. Dostupné z: https://gamemaker.io/en/legal/gamemaker-runtime-licence-professional.

[37] *GameObject* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/ScriptReference/GameObject.html.

[38] *GameObject.FindGameObjectsWithTag* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/GameObject.FindGameObjectsWithTag.html.

[39] *GameObject.FindWithTag* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.1/Documentation/ScriptReference/GameObject.FindWithTag.html.

[40] *GameObject.SetActive* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.1/Documentation/ScriptReference/GameObject.SetActive.html.

[41] *Games Made With Unity* [online]. c2025. [cit. 2025-04-24]. Dostupné z: https://store.steampowered.com/curator/39750107-Games-Made-With-Unity/.

[42] *Gears 5* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://www.gearsofwar.com/games/gears-5/.

[43] *GML Visual Overview* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://manual.gamemaker.io/monthly/en/index.htm?%5C#t=Drag_And_Drop%5C%2FDrag_And_Drop_Overview%5C%2FDnD_Overview.htm.

[44] *Godot* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://godotengine.org/.

[45] *Godot Marketplace* [online]. c2020. [cit. 2025-04-24]. Dostupné z: https://godotmarketplace.com/.

[46] *GX.games* [online]. c2025. [cit. 2025-04-24]. Dostupné z: https://gx.games/.

[47] *Head-to-head benchmark: C++ vs .NET* [online]. c2025. [cit. 2025-04-24]. Dostupné z: https://www.codeproject.com/articles/212856/head-to-head-benchmark-csharp-vs-net.

[48] *Hyper Light Drifter* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://store.steampowered.com/app/257850/Hyper_Light_Drifter/.

[49] *Chaos Physics Overview* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/chaos-physics-overview?application_version=4.27.

[50] *In-Depth: Far Cry 2's Guay Talks Dunia Engine, State Of PC* [online]. c2025. [cit. 2025-04-08]. Dostupné z: https://www.gamedeveloper.com/game-platforms/in-depth-i-far-cry-2-i-s-guay-talks-dunia-engine-state-of-pc.

[51] *Information About Frame Rate* [online]. c2025. [cit. 2025-05-05]. Dostupné z: https://www.logicalincrements.com/articles/framerate.

[52] *Interface (C# Reference)* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface.

[53] *Introduction to Blueprints* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.unrealengine.com/en-US/blog/introduction-to-blueprints.

[54] *JsonUtility* [online]. c2025. [cit. 2025-04-26]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/JsonUtility.html.

[55] *Kingdom Come: Deliverance 2* [online]. c2025. [cit. 2025-04-08]. Dostupné z: https://kingdomcomerpg.com/.

[56] *License* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://godotengine.org/license/.

[57] *Microphone* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/ScriptReference/Microphone.html.

[58] *Microphone.Start* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/ScriptReference/Microphone.Start.html.

[59] *Microsoft HoloLens* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://learn.microsoft.com/cs-cz/hololens/.

[60] MILLINGTON, Ian. *Game physics engine development: how to build a robust commercial-grade physics engine for your game*. 2nd ed. Amsterdam: Morgan Kaufmann Publishers, [2010]. ISBN 978-0-12-381976-5.

[61] *MonoBehaviour.Update()* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.1/Documentation/ScriptReference/MonoBehaviour.Update.html.

[62] *Most used Game Engines* [online]. c2025. [cit. 2025-02-18]. Dostupné z: https://itch.io/game-development/engines/most-projects.

[63] *MV3D - 3D Rendering Plugin for RPG Maker MV* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://cutievirus.itch.io/mv3d.

[64] *Need for Speed Unbound* [online]. c2025. [cit. 2025-04-20]. Dostupné z: https://www.ea.com/games/need-for-speed/need-for-speed-unbound.

[65] *NHL 24* [online]. c2025. [cit. 2025-04-20]. Dostupné z: https://www.ea.com/games/nhl/nhl-24.

[66] *OMORI* [online]. 2025. [cit. 2025-04-21]. Dostupné z: https://www.omori-game.com/en.

[67] *Optimize your game performance for consoles and PCs in Unity (Unity 6 edition)* [online]. c2025. [cit. 2025-05-05]. Dostupné z: https://unity.com/resources/console-pc-game-performance-optimization-unity-6.

[68] *Paper 2D* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/paper-2d-overview-in-unreal-engine.

[69] *Physics* [online]. c2025. [cit. 2025-04-24]. Dostupné z: https://docs.unity3d.com/Manual/PhysicsSection.html.

[70] *Physics* [online]. c2024. [cit. 2025-04-26]. Dostupné z: https://manual.gamemaker.io/lts/en/index.htm%5C#t=GameMaker_Language%5C%2FGML_Reference%5C%2FPhysics%5C%2FPhysics.htm.

[71] *Physics introduction* [online]. c2014. [cit. 2025-04-24]. Dostupné z: https://docs.godotengine.org/en/stable/tutorials/physics/physics_introduction.html.

[72] *Physics Material 2D reference* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/Manual/2d-physics/physics-material-2d-reference.html.

[73] *Platform Effector 2D reference* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/Manual/2d-physics/effectors/platform-effector-2d-reference.html.

[74] *Prefabs* [online]. c2025. [cit. 2025-05-03]. Dostupné z: https://docs.unity3d.com/Manual/Prefabs.html.

[75] *PUBG: BATTLEGROUNDS* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://www.pubg.com/en/main.

[76] *Resources.Load* [online]. c2025. [cit. 2025-05-06]. Dostupné z: https://docs.unity3d.com/6000.1/Documentation/ScriptReference/Resources.Load.html.

[77] *Road to Vostok* [online]. c2024. [cit. 2025-04-24]. Dostupné z: https://www.roadtovostok.com/.

[78] *RPG Maker* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://rpgmakerofficial.com/en/.

[79] *Scripting the Unreal Editor Using Python* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/scripting-the-unreal-editor-using-python.

[80] *Sea of Thieves* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://www.seaofthieves.com/.

[81] SMUTNÝ, Jan. *Měření FPS* [online]. 2025. [cit. 2025-05-05]. Dostupné z: https://github.com/SmutnyJan/PDO-semestral-project/blob/main/Sc%5C%C3%5C%A9n%5C%C3%5C%A1%5C%C5%5C%99e/M%5C%C4%5C%9B%5C%C5%5C%99en%5C%C3%5C%AD%5C%20FPS.md.

[82] *Spatialization Overview* [online]. c2025. [cit. 2025-04-08]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/spatialization-overview-in-unreal-engine.

[83] *Stack Class* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/api/system.collections.stack?view=net-9.0.

[84] STREČANSKÁ, Dominika. *Využití shaderů a procedurálního generování grafiky ve 2D pro herní engine Unity* [online]. Liberec, 2025 [cit. 2025-04-30]. Dostupné z: https://dspace.tul.cz/. Bakalářská práce. Technická univerzita v Liberci.

[85] *Super Mario Game* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://supermario-game.com/.

[86] *Swords and Sandals Immortals* [online]. [cit. 2025-04-24]. Dostupné z: https://store.steampowered.com/app/1707650/Swords_and_Sandals_Immortals/.

[87] *The Open Source Definition* [online]. 2025. [cit. 2025-04-22]. Dostupné z: https://opensource.org/osd.

[88] *The Showdown Between Unity vs RPG Maker* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://aircada.com/blog/unity-vs-rpg-maker.

[89] *This is Meta Quest* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://www.meta.com/quest/.

[90] *Timeline* [online]. c2025. [cit. 2025-05-03]. Dostupné z: https://docs.unity3d.com/Manual/com.unity.timeline.html.

[91] *Undertale* [online]. 2015. [cit. 2025-04-26]. Dostupné z: https://undertale.com/.

[92] *Unity Engine* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://unity.com/products/unity-engine.

[93] *Unity Terms of Service* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://unity.com/legal/terms-of-service.

[94] *Unreal Engine* [online]. c2025. [cit. 2025-04-21]. Dostupné z: https://www.unrealengine.com/en-US.

[95] *Unreal Engine End User License Agreement* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.unrealengine.com/en-US/eula/unreal.

[96] *Using Jolt Physics* [online]. c2014. [cit. 2025-04-24]. Dostupné z: https://docs.godotengine.org/en/4.4/tutorials/physics/using_jolt_physics.html.

[97] *VALORANT* [online]. c2025. [cit. 2025-04-23]. Dostupné z: https://playvalorant.com/en-us/.

[98] *Vector2.MoveTowards* [online]. c2025. [cit. 2025-04-29]. Dostupné z: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Vector2.MoveTowards.html.

[99] *WebGL Overview* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.khronos.org/webgl/.

[100] *What Are the Best Game Engines?* [online]. c2025. [cit. 2025-04-22]. Dostupné z: https://www.perforce.com/blog/vcs/most-popular-game-engines.

[101] *What is Ren'Py?* [online]. 2025. [cit. 2025-04-22]. Dostupné z: https://www.renpy.org/.

A Příklad souboru „progress.json“


```
{
    "SpawnScene": 12,
    "GameState": 4,
    "Money": 296,
    "Items": [
        {
            "ItemType": 0,
            "Amount": 13
        },
        {
            "ItemType": 1,
            "Amount": 1
        },
        {
            "ItemType": 2,
            "Amount": 3
        },
        {
            "ItemType": 3,
            "Amount": 3
        },
        {
            "ItemType": 4,
            "Amount": 2
        },
        {
            "ItemType": 5,
            "Amount": 3
        }
    ],
    "LevelConfig": {
        "Scene": 12,
        "SpawnPoint": {
            "x": 200.83999633789063,
            "y": 15.585000038146973,
            "z": 0.0
        },
        "ChestsOpenedIndexes": [
            0,
            1,
            2
        ],
        "PlanesDestroyedIndexes": [
```

```
            0,
            3
        ],
        "ItemsRevert": [
            {
                "ItemType": 0,
                "Amount": 10
            },
            {
                "ItemType": 1,
                "Amount": 1
            },
            {
                "ItemType": 2,
                "Amount": 1
            },
            {
                "ItemType": 3,
                "Amount": 2
            },
            {
                "ItemType": 4,
                "Amount": 2
            },
            {
                "ItemType": 5,
                "Amount": 2
            }
        ],
        "MoneyRevert": 290
    }
}
```

B Skript pro zpracování herního logu v jazyce Python

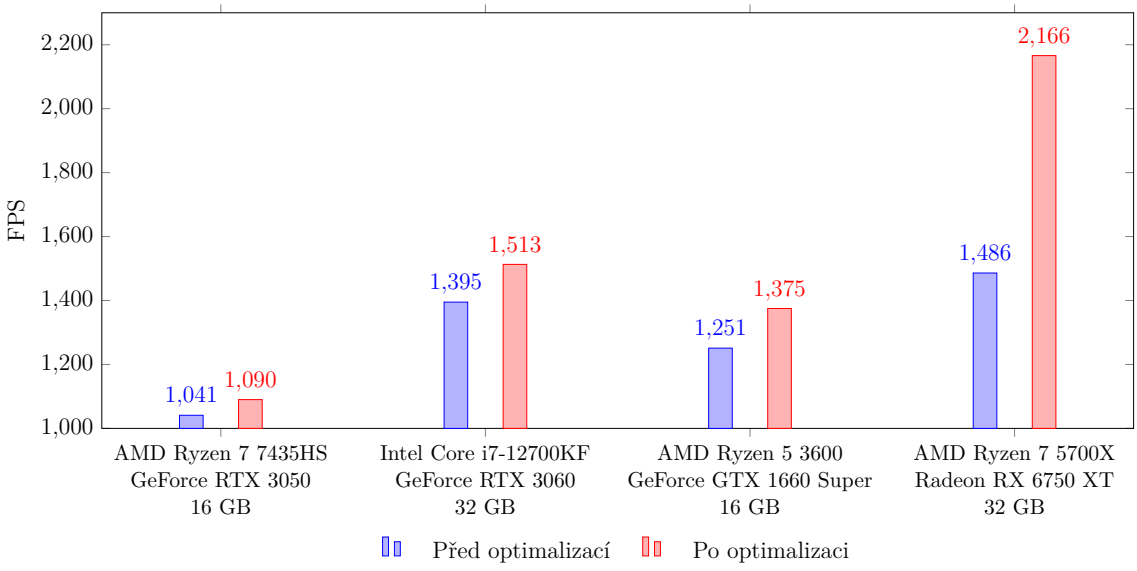
```
log_file_path = "Player.log"
mainmenu_sum = 0
mainmenu_count = 0
lobby_sum = 0
lobby_count = 0
levelg_sum = 0
levelg_count = 0

with open(log_file_path, 'r', encoding='utf-8') as file:
    for line in file:
        line = line.strip()
        if (line.startswith("MainMenu FPS:")
            or line.startswith("Settings FPS:")):
            fps = float(line.split(":")[1].strip().replace(",", "."))
            mainmenu_sum += fps
            mainmenu_count += 1
        elif (line.startswith("LobbyMenza FPS:")
              or line.startswith("LobbyG FPS:")):
            fps = float(line.split(":")[1].strip().replace(",", "."))
            lobby_sum += fps
            lobby_count += 1
        elif line.startswith("LevelG FPS:"):
            fps = float(line.split(":")[1].strip().replace(",", "."))
            levelg_sum += fps
            levelg_count += 1
def calc_avg(total, count):
    return total / count if count > 0 else 0

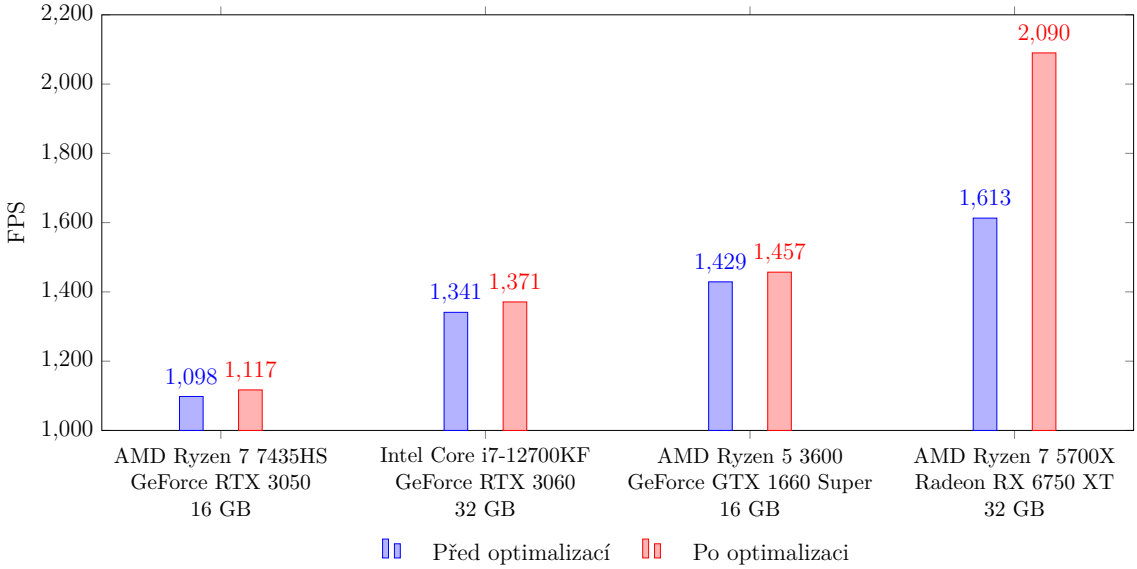
print(
    f"Průměrné FPS pro MainMenu (včetně Settings): "
    f"{calc_avg(mainmenu_sum, mainmenu_count):.2f}")
print(f"Průměrné FPS pro Lobby (LobbyMenza/LobbyG):"
      f"{calc_avg(lobby_sum, lobby_count):.2f}")
print(f"Průměrné FPS pro LevelG: {calc_avg(levelg_sum, levelg_count):.2f}")
```

C Měření FPS

C.1 Hlavní menu



C.2 Lobby



C.3 Level G

