DELTA – Střední škola informatiky a ekonomie, s.r.o.
Ke Kamenci 151, Pardubice

Generování a randomizace obsahu v rámci 2D hry

Castle of Ghosts

Pluhař Matyáš
4.A
Informační technologie 18-20-M/01
2024/2025

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	Matyáš Pluhař
Školní rok:	2024/25
Třída:	4. A
Obor:	Informační technologie 18-20-M/01

Téma práce:	Generování a randomizace obsahu v rámci 2D hry
Vedoucí práce:	Jan Mottl

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem práce je nastudovat princip náhodného generování a randomizace obsahu v herním prostředí. Student prokáže tuto znalost realizací 2D adventurní hry inspirované historickým tématem. Hra bude při každém průchodu nabízet unikátní zážitek díky generování světa a detektivní zápletky pomocí algoritmu, který student navrhne a zrealizuje.

Harmonogram příprav

- **Září:** Rešerše a plánování řešení, seznámení s vývojářskými nástroji
- **Říjen:** Začátek realizace a tvorba základních systému (dialogy, inventář atd.)
- **Listopad**: Práce na tvorbě algoritmů pro randomizaci obsahu
- **Prosinec**: Práce na dokončení hrubé verze hry.
- **Leden:** Dokončení práce na beta verzi hry a odladění chyb.
- **Únor:** Beta testování hry a začátek práce na dokumentaci.
- **Březen**: Ladění chyb nalezených při beta testu hry a dokončení dokumentace.

Prohlášení

Prohlašuji, že jsem maturitní projekt vypracoval samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 31. 3. 2025

Poděkování

Děkuji svému vedoucímu projektu Mgr. Janu Mottlovi za odborné vedení a cenné rady během celé realizace projektu.

Anotace

Tato dokumentace popisuje vývoj 2D adventurní hry s prvky hororu a náhodného generování. Hra *Castle of Ghosts* využívá hybridní přístup k tvorbě herního světa, kde jsou předem vytvořené místnosti propojeny náhodně, čímž vzniká pokaždé unikátní prostředí. Dokumentace zahrnuje popis herního konceptu, použitých technologií, herních mechanik a algoritmů pro náhodnou generaci obsahu, včetně rozmístění místností, NPC a hádanek. Součástí práce je také analýza přístupu k náhodnému generování a jeho vliv na znovuhratelnost hry.

Klíčová slova

Unity, C#, procedurální generování, adventura, herní design, náhodné generování

Annotation

This documentation describes the development of a 2D adventure game with horror elements and random generation. The game *Castle of Ghosts* utilizes a hybrid approach to level design, where pre-designed rooms are randomly interconnected, creating a unique environment each time the game is played. The documentation includes an overview of the game concept, used technologies, game mechanics, and algorithms for procedural content generation, such as room layout, NPC placement, and puzzles. Additionally, it analyzes random generation approaches and their impact on game replayability.

Keywords

Unity, C#, procedural generation, adventure, game design, random generation

Obsah

1 Technologie ... 2
1.1 Použité technologie ... 2
1.2 Alternativní technologie ... 2
2 Manuál ... 3
2.1 Ovládání a uživatelské rozhraní ... 3
2.2 Herní mechaniky a cíle ... 3
3 Game Design Dokument ... 4
3.1 Přehled hry ... 4
3.1.1 Koncept hry ... 4
3.1.2 Základní herní mechanismy ... 4
3.1.3 Žánr hry ... 4
3.1.4 Cílová skupina ... 5
3.1.5 Atmosféra hry ... 5
3.2 Herní mechaniky ... 5
3.2.1 Postup hrou ... 5
3.2.2 Struktura úkolů ... 5
3.3 Herní lokace a prostředí ... 6
3.4 Postavy ... 6
3.5 Ovládání a interakce ... 7
3.6 Grafický styl ... 7
4 Architektura ... 8
4.1 Use case diagram ... 8
4.2 Class diagram ... 8
5 Teoretická část ... 10
5.1 Přístupy ke generování obsahu ve hrách ... 10
5.2 Použité algoritmy ... 11
5.2.1 Algoritmus náhodného výběru místností ... 11
5.2.2 Algoritmus propojení místností ... 12
5.2.3 Algoritmus rozmístění NPC a hádanek ... 13
5.2.4 Algoritmus generování úkolů a klíčových předmětů ... 15
5.3 Výhody a nevýhody použitého přístupu ... 16
5.4 Srovnání s existujícími hrami ... 17
6 Beta test ... 18
6.1 Výsledky testování ... 18

Úvod

Můj maturitní projekt si klade za cíl vysvětlit a zmapovat možnosti randomizace a náhodného generování objektů v kontextu 2D hry.

Hlavním cílem praktické části projektu je vytvoření hry s vysokou znovuhratelností, ve které se jednotlivé místnosti, hádanky a nehráčské postavy (NPC) mění při každém spuštění. Tento přístup zajišťuje, že žádné dva herní průchody nebudou totožné. Hra je inspirována klasickými adventurami a rogue-like tituly, které využívají náhodně generovaný obsah.

Praktickým výstupem projektu je hra *Castle of Ghosts*, 2D adventura odehrávající se v prostředí temného hradu. Hráč se ocitá ve světě plném tajemství, záhadných postav a hádanek, které musí vyřešit, aby nalezl cestu ven.

Teoretická část práce se zaměřuje na zkoumání možností náhodného generování herního obsahu a jeho vlivu na herní zážitek. Hra využívá hybridní přístup, ve kterém jsou předem definované místnosti propojeny náhodně, čímž vzniká vždy jedinečné uspořádání prostoru. Náhodně jsou generovány také hádanky a NPC, což výrazně zvyšuje variabilitu hry.

1 Technologie

1.1 Použité technologie

Pro vývoj hry *Castle of Ghosts* jsem jako hlavní nástroj zvolil Unity (C#), herní engine, který umožňuje efektivní práci s 2D prostředím, animacemi a fyzikou. Unity jsem vybral především díky jeho rozsáhlé dokumentaci a široké komunitě vývojářů, což značně usnadňuje řešení případných problémů během vývoje. Programovací jazyk C# slouží jako základní stavební kámen projektu a nabízí silnou podporu pro objektově orientované programování, což je klíčové pro strukturovaný návrh herních systémů a mechanik. Pro verzování kódu a zálohování projektu jsem využil GitHub, což mi umožnilo systematicky sledovat změny a v případě potřeby se vracet k předchozím verzím. Pro samotný vývoj jsem používal Visual Studio jako hlavní vývojové prostředí (IDE) díky jeho pokročilým funkcím pro ladění a analýzu kódu v C#. Vizuální stránku hry jsem vytvářel pomocí nástroje Aseprite, který je specializovaný na tvorbu pixel artu a nabízí pokročilé možnosti pro animaci a editaci spritů.

1.2 Alternativní technologie

Při výběru technologií jsem zvažoval i alternativní řešení. Godot Engine představuje populární open-source alternativu k Unity, která nabízí vestavěný jazyk GDScript. Tento engine má však méně pokročilé nástroje pro 2D fyziku a uživatelské rozhraní než Unity, což by při vývoji adventurní hry mohlo představovat omezení. Další zvažovanou možností byl Unreal Engine, který by umožnil implementaci detailnější grafiky a pokročilejších vizuálních efektů. Nevýhodou Unreal Engine je však jeho primární zaměření na 3D hry, což by v případě 2D projektu mohlo vést k nadbytečné složitosti.

2 Manuál

2.1 Ovládání a uživatelské rozhraní

Hráč ovládá postavu pomocí klávesnice, přičemž základní pohyb je realizován klávesami WASD. Pro interakci s NPC a objekty ve hře slouží klávesa E, zatímco inventář lze otevřít a zavřít pomocí klávesy Q. Toto intuitivní schéma ovládání vychází z běžných standardů v žánru adventurních her a umožňuje hráčům rychlé přizpůsobení se herním mechanikám.

Grafické rozhraní hry obsahuje několik základních prvků. Inventář zobrazuje předměty, které hráč během průzkumu nalezl, a umožňuje jejich použití. Dialogové okno se objevuje při rozhovoru s NPC a zobrazuje konverzaci. Hlavní menu poskytuje možnosti pro spuštění nové hry nebo opuštění aplikace.

2.2 Herní mechaniky a cíle

Základní herní mechaniky zahrnují několik klíčových prvků. Prozkoumávání hradu představuje základní aktivitu, při které se hráč pohybuje mezi místnostmi a objevuje nové prostory. Během průzkumu může sbírat předměty, komunikovat s NPC a řešit různé hádanky. Řešení hádanek tvoří důležitou součást hratelnosti, přičemž každé spuštění hry generuje novou hádanku s částmi rozmístěnými v různých místnostech. Hráč musí tyto části najít a složit do kompletní odpovědi, aby mohl postupovat dále. Sbírané předměty, jako jsou klíče nebo nápovědy, se ukládají do inventáře, odkud je hráč může použít podle potřeby. Interakce s NPC obohacují herní zážitek, neboť každá nehráčská postava má svůj vlastní příběh a může poskytnout cenné informace nebo zadat úkoly, které pomáhají v postupu hrou.

Pro úspěšné projití hrou je důležité systematicky prozkoumávat prostředí, neboť každé spuštění generuje nové uspořádání místností. Hráč musí pečlivě sbírat nápovědy, protože některé dveře jsou zamčené hádankami, na které je nutné najít správnou odpověď. Komunikace s NPC poskytuje další vrstvu hratelnosti, neboť postavy mohou zadávat úkoly, které pomáhají při postupu. Konečným cílem hry je uniknout z hradu a přijít na to, co se na opuštěném hradě stalo.

3 Game Design Dokument

3.1 Přehled hry

3.1.1 Koncept hry

Hra *Castle of Ghosts* je 2D adventurní hra zasazená do temného, opuštěného hradu, ve kterém hráč prozkoumává náhodně generované prostory. Hlavním lákadlem hry je unikátní herní zážitek při každém spuštění díky hybridnímu generování obsahu. Místnosti jsou předem navrženy, ale jejich rozmístění a vzájemná propojení jsou vytvářena náhodně. Kromě dynamického layoutu místností hra nabízí i náhodně generované hádanky, klíče a interaktivní NPC v podobě duchů, jejichž pozice, trasy a úkoly se mění při každém novém průchodu.

Příběh hráče staví do role záhadného návštěvníka, jehož cílem je uniknout z tajemného hradu a přijít na to, co se na opuštěném hradě stalo. K tomu je zapotřebí nejen pečlivého průzkumu, ale také řešení logických hádanek a interakce s duchy, kteří poskytují klíčové informace nutné k úspěšnému dokončení hry.

3.1.2 Základní herní mechanismy

Hra kombinuje prvky klasických adventur s rogue-like mechanismy. Hlavními herními prvky jsou průzkum prostředí, sběr předmětů, řešení logických hádanek, dynamická interakce s NPC a správa inventáře. Průzkum hradu představuje jádro hry, přičemž místnosti, objekty a duchové jsou při každém spuštění umístěni jinak, což zajišťuje čerstvý zážitek.

Inventář umožňuje hráči sbírat klíče, předměty a části hádanek, které lze kombinovat k vyřešení úkolů. NPC (duchové) hráči poskytují dialogy, zadávají úkoly a reagují na jeho akce a rozhodnutí.

3.1.3 Žánr hry

Castle of Ghosts patří do žánru 2D adventur s rogue-like prvky, jelikož se v ní kombinuje klasická adventurní logika (řešení hádanek, interakce, příběhové prvky) s dynamickou, opakovatelnou strukturou světa typickou pro rogue-like hry. Adventurní žánr klade důraz na příběh a logiku, přičemž rogue-like elementy dodávají hře variabilitu, která podporuje opakovanou hratelnost a poskytuje hráči stále nové výzvy a zážitky.

3.1.4 Cílová skupina

Cílovou skupinou hry jsou hráči, kteří vyhledávají adventury s důrazem na hádanky, ale také hráči, kteří oceňují vysokou znovuhratelnost typickou pro rogue-like hry. *Castle of Ghosts* je určena všem hráčům, kteří mají rádi tajemnou atmosféru, příběhové prvky a jsou otevřeni opakovanému průzkumu herního prostředí s dynamickým obsahem.

3.1.5 Atmosféra hry

Atmosféra *Castle of Ghosts* je tajemná. Grafická stylizace hry se opírá o pixel-art estetiku. Duchové, i když nejsou explicitně nepřátelští, vytvářejí pocit nejistoty a napětí.

3.2 Herní mechaniky

3.2.1 Postup hrou

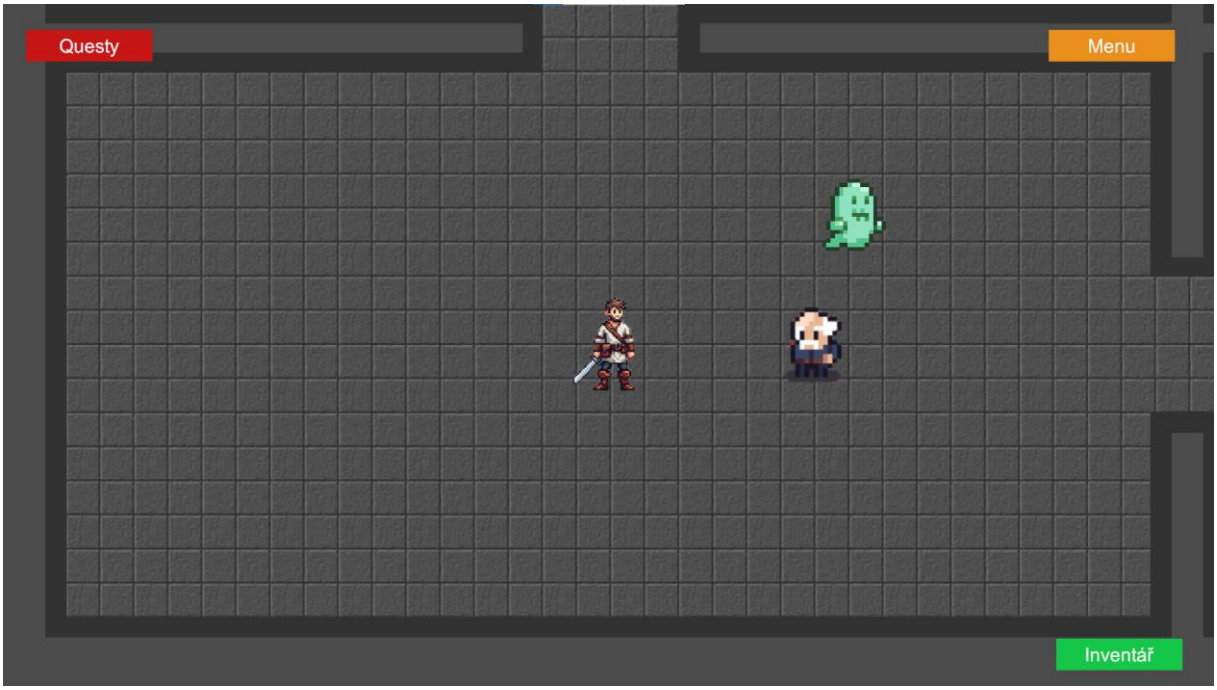
Hra se skládá z jednoho hlavního aktu – útěku z hradu. Po spuštění hráč vždy začíná ve vybrané počáteční místnosti. Jeho cílem je odemknout zamčené dveře, najít klíče, řešit hádanky a interagovat s duchy, kteří mu poskytují informace nebo zadávají úkoly. Progres je založený na postupném odemykání dalších místností a sbírání částí hádanek, přičemž každá dokončená hádanka hráči otevře další část hradu nebo poskytne důležitý předmět.

3.2.2 Struktura úkolů

Úkoly zadávají NPC – duchové. Každý duch má svůj vlastní příběh a motivace, což se odráží v dialogovém systému. Hráč dostává úkoly typu „najdi předmět“. Úkoly nejsou lineární, lze je plnit paralelně a jejich pořadí se mění dle náhodného generování obsahu při spuštění hry.

3.3 Herní lokace a prostředí

Hra se odehrává v interiéru hradu, který sestává z řady předpřipravených místností. Generování hradu se provádí náhodně, takže při každém spuštění hry vzniká nový unikátní layout hradu.



Obrázek 3.1: Ukázka ze hry

3.4 Postavy

Hlavní postavou je bezejmenný návštěvník, jehož příběh se odkrývá postupně. NPC jsou výhradně duchové bývalých obyvatel hradu, kteří zůstali uvězněni v prostoru. Každý duch má individuální příběhové pozadí a dialogy, které hráči pomáhají pochopit minulost hradu.



Obrázek 3.2: Postava ducha

3.5 Ovládání a interakce

Pohyb hráče je řešen pomocí kláves WASD, interakce s objekty a NPC pomocí klávesy E a inventář se ovládá klávesou Q.

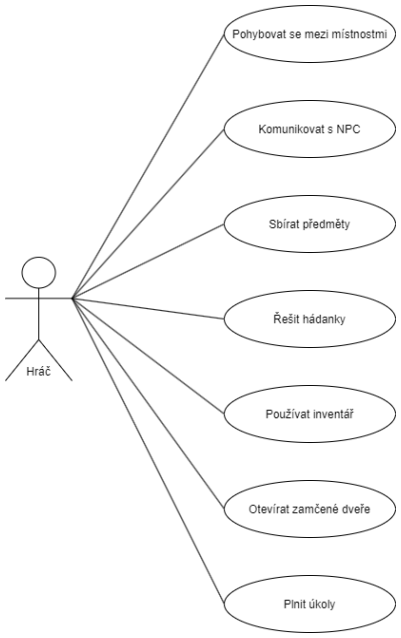
3.6 Grafický styl

Hra využívá pixel-art grafiku, která dodává hře minimalismus.

4 Architektura

4.1 Use case diagram

Use Case Diagram znázorňuje hlavní interakce hráče se hrou. Zobrazuje činnosti, které hráč může ve hře vykonávat, a jejich vzájemné propojení. Tento diagram pomáhá vizualizovat základní mechaniky hry a poskytuje přehled o herních funkcích.



Obrázek 4.1: Use case diagram

4.2 Class diagram

Class diagram (uveden v příloze **Příloha** I) znázorňuje strukturu a vztahy mezi hlavními třídami hry *Castle of Ghosts*. Tento diagram poskytuje přehled o tom, jak spolu jednotlivé komponenty komunikují a jaké role hrají v herním systému.

Hlavní třídou ve hře je **Player**, která reprezentuje hráčovu postavu a umožňuje pohyb, interakci s herními objekty a spravování inventáře. Hráčovo vybavení je spravováno třídou **Inventory**, která obsahuje předměty typu **Item**. Mezi speciálními předměty je **Key**, který slouží k odemykání dveří implementovaných třídou **Door**. Každé dveře propojují dvě místnosti reprezentované třídou **Room**, přičemž umístění místností je řízeno třídou **RoomGenerator**.

Hra obsahuje také systém nehráčských postav (NPC), který je řízen třídou **NPC**. Každé NPC může mít vlastní trasu pohybu, což zajišťuje **NPCMovement**. Interakce mezi hráčem a NPC probíhá prostřednictvím **DialogueManager**, který spravuje dialogy a umožňuje

hráči vybírat odpovědi definované třídou **PlayerChoice**.

Herní svět je dynamicky generován třídou **RoomGenerator**, která vytváří jednotlivé místnosti a přiřazuje jim náhodný obsah. Každá místnost může obsahovat NPC, předměty nebo hádanky. Hádanky jsou součástí úkolového systému řízeného **QuestManager**, který sleduje aktivní úkoly a jejich postup. Úkoly mohou vyžadovat nasbírání předmětů z **Inventory** nebo splnění specifických podmínek v rámci herního světa.

Důležitou roli v herním zážitku hraje také systém nápověd a interaktivních prvků. Třída **Hint** spravuje herní nápovědy a umožňuje hráči získat užitečné informace o dalším postupu. Pro zajištění pokročilejších interakcí je implementována třída **PasswordPanel**, která umožňuje zadávání hesel potřebných k otevření určitých dveří nebo k přístupu k dalším herním prvkům.

Vztahy mezi jednotlivými třídami jsou jasně definované. **Player** komunikuje s **Inventory** při sbírání předmětů, zatímco **DialogueManager** propojuje hráče s **NPC**. **QuestManager** kontroluje interakce mezi hráčem, předměty a NPC a řídí postup ve hře. **RoomGenerator** zajišťuje vytvoření a propojení místností, přičemž **Door** fungují jako spojovací uzly mezi místnostmi. NPC se pohybují mezi místnostmi podle předem definovaných cest spravovaných **NPCMovement**.

5 Teoretická část

Herní design se často potýká s problémem, jak vytvořit dostatečně dynamický a znovuhratelný obsah [1], který hráče nezačne nudit po několika průchodech. Tato výzva vedla k rozvoji algoritmického generování herního obsahu, které umožňuje vytvářet variabilní prostředí, hádanky a úkoly při každém spuštění hry. Randomizace a procedurální generování se staly významnými nástroji herních vývojářů [2], protože dokážou značně prodloužit životnost hry a zvýšit hodnotu opakovaných průchodů.

V mém projektu implementuji kombinovaný přístup, kde mám ručně vytvořené místnosti, které jsou propojeny náhodně [3] tak, aby hráč pokaždé procházel jinak sestaveným hradem. Tento přístup jsem zvolil proto, že umožňuje zachovat kontrolu nad designem jednotlivých místností (jejich vzhledem, rozmístěním objektů atd.), ale zároveň poskytuje dostatečnou variabilitu díky měnícímu se propojení těchto místností. Kromě samotného uspořádání hradu jsou náhodně generovány také hádanky, NPC a jejich pozice, což dále zvyšuje variabilitu herního zážitku.

5.1 Přístupy ke generování obsahu ve hrách

V herním designu existuje několik základních metod pro tvorbu herního světa, které se liší mírou kontroly designéra a variabilitou výsledného obsahu.

Ručně vytvořený obsah představuje tradiční přístup [4], kdy designéři manuálně vytvářejí všechny místnosti, hádanky i NPC. Výhodou tohoto přístupu je absolutní kontrola nad kvalitou a konzistencí obsahu, což umožňuje vytvářet pečlivě vyladěné herní zážitky. Nevýhodou je omezená variabilita a znovuhratelnost, neboť hráč po několika průchodech zná všechny překvapení, která hra nabízí. Tento přístup je typický pro lineární příběhové hry, kde je důraz kladen na pevně danou narativní strukturu.

Procedurální generování představuje opačný přístup [4], kdy je herní obsah vytvářen pomocí algoritmů. Tato metoda umožňuje dynamickou a teoreticky nekonečně rozmanitou hru, protože algoritmus může generovat BSP stále nové kombinace prvků. Nevýhodou je potenciální ztráta kontroly nad designem a možnost vzniku nelogických nebo nekonzistentních herních situací. Procedurální generování se využívá například v rouge-like hrách, kde je vysoká znovuhratelnost klíčovým prvkem herního zážitku.

Hybridní přístup, který jsem zvolil pro svůj projekt, spojuje ručně vytvořené prvky s náhodným generováním [3]. Tato metoda zachovává kontrolu nad klíčovými aspekty designu, ale zároveň poskytuje variabilitu, která zvyšuje znovuhratelnost. V mé hře jsou místnosti ručně navrženy, což zajišťuje jejich vizuální a funkční konzistenci, ale jejich

propojení je generováno náhodně, což vytváří pokaždé jinou mapu hradu.

5.2 Použité algoritmy

V projektu jsem implementoval čtyři hlavní algoritmické procesy, které společně vytvářejí unikátní herní zážitek při každém spuštění.

5.2.1 Algoritmus náhodného výběru místností

Algoritmus náhodného výběru místností funguje na principu fixního gridu (mřížky) [5], do jehož buněk se náhodně vybírají prefaby (předpřipravené objekty) místností s různými počty dveří. Algoritmus nejprve vytvoří prázdný grid (síť), představující strukturu hradu. Poté postupně vybírá místnosti z předpřipravené databáze prefabů (šablon) a umisťuje je do buněk gridu (rozložení). Pro zachování konzistence musí algoritmus zajistit, že mapa nezůstane nepropojená a hráč má možnost projít všemi místnostmi.

Postup generování zahrnuje tři klíčové kroky. Nejprve probíhá výběr místností z databáze prefabů (předdefinovaných prvků), přičemž se zohledňuje jejich typ (např. chodba, sál, ložnice) a počet možných východů. Následuje umístění těchto místností do gridu (mřížky) takovým způsobem, aby nedocházelo k jejich překryvu a byla zachována logická struktura hradu. Nakonec probíhá kontrola propojení místností, která zajišťuje, že každá místnost je připojena minimálně k jedné další místnosti, čímž vzniká souvislá a průchozí mapa.

Tento přístup jsem zvolil z několika důvodů. Především zajišťuje unikátní rozložení místností při každém spuštění hry, což významně zvyšuje její znovuhratelnost. Ručně vytvořené místnosti poskytují dostatečnou kontrolu nad designem, zatímco náhodné propojení zajišťuje variabilitu. Tento přístup efektivně kombinuje přednosti ručního designu a dynamického generování.

Při implementaci jsem zvažoval i alternativní přístupy. Perlin Noise generování je populární pro vytváření otevřených světů [6], například v Minecraftu, kde pomáhá vytvářet přirozeně vypadající krajiny. V interiérovém prostředí hradu, kde potřebujeme konzistentní rozložení místností, však není ideální, protože může vést k příliš organickým a méně strukturovaným výsledkům. BSP Tree (Binary Space Partitioning) je další možností [7], která byla využita například v původním Doomu. V tomto případě se algoritmus používal k rozdělení mapy na menší sektory a optimalizaci vykreslování, což zlepšovalo výkon a určovalo viditelnost jednotlivých částí úrovně. BSP se však dá aplikovat i na procedurální generování dungeonů, kde rozděluje prostor hierarchicky a následně do něj vkládá místnosti. Tento přístup je efektivní pro rogue-like struktury, ale

pro můj projekt by byl příliš rigidní a neposkytoval by dostatečnou kontrolu nad vzhledem jednotlivých místností.

5.2.2 Algoritmus propojení místností

Algoritmus propojení místností navazuje na umístění místností [5] a zajišťuje jejich logické propojení dveřmi. Po vytvoření základní struktury hradu musí algoritmus vytvořit průchody mezi sousedními místnostmi tak, aby hráč mohl projít celou mapu.

Dveře se mohou generovat pouze mezi místnostmi, které jsou přímo vedle sebe buď vodorovně, nebo svisle. Diagonální propojení místností není povoleno, protože by narušovalo jasnou strukturu hradu a způsobovalo by nesrovnalosti v navigaci hráče. Pokud má místnost více než jednoho souseda, algoritmus náhodně určí, kolik dveří se vytvoří, ale zároveň zajišťuje, že celá mapa zůstane průchozí [8]. Tento systém také počítá se zamčenými dveřmi, které mohou vyžadovat klíč nebo vyřešení hádanky k jejich otevření, čímž hra získává další vrstvu hratelnosti. Počet těchto zamčených dveří je však omezen, aby se nestalo, že hráč narazí na příliš mnoho bariér najednou a nemohl pokračovat v průchodu hrou.
Dveře se umisťují do středu odpovídající stěny mezi dvěma místnostmi, což zajišťuje vizuální konzistenci a zároveň umožňuje snadné propojení místností. Pokud místnost nemá žádného souseda, je odstraněna z gridu (mřížky), aby se předešlo vzniku

```
if (doorComponent != null)
{
    doorComponent.SetRoomGenerator(this);
    doorComponent.SetConnectedRoomPosition(neighborPosition);

    currentRoom.AddDoor(doorComponent);
    neighborRoom.AddDoor(doorComponent);

    // Správné odstranění stěn mezi propojenými místnostmi
    if (direction == new Vector2Int(1, 0)) // Right
    {
        currentRoom.RemoveWallSegment("right");
        neighborRoom.RemoveWallSegment("left");
    }
    else if (direction == new Vector2Int(-1, 0)) // Left
    {
        currentRoom.RemoveWallSegment("left");
        neighborRoom.RemoveWallSegment("right");
    }
    else if (direction == new Vector2Int(0, 1)) // Up
    {
        currentRoom.RemoveWallSegment("top");
        neighborRoom.RemoveWallSegment("bottom");
    }
    else if (direction == new Vector2Int(0, -1)) // Down
    {
        currentRoom.RemoveWallSegment("bottom");
        neighborRoom.RemoveWallSegment("top");
    }

    connectedRooms.Add(neighborPosition);
```

Obrázek 5.1: Ukázka mazání přebytečných stěn, po vytvoření dveří

nevyužitých prázdných prostor. Po vygenerování všech dveří probíhá kontrola dosažitelnosti místností, aby se zajistilo, že žádná z nich nezůstane odříznutá a hráč může navštívit celý hrad. Tímto způsobem se udržuje plynulost hry a hráč není frustrován neschopností postoupit dále.

Aby se zabránilo situaci, kdy by hráč mohl uvíznout v místnosti bez řešení, algoritmus provádí kontrolu dosažitelnosti všech místností. Po vygenerování celého hradu probíhá simulace pohybu, která ověřuje, že z každé místnosti existuje alespoň jedna možná cesta ven. Pokud algoritmus narazí na uzavřenou oblast, provede se opravné propojení dveří nebo vygenerování nové mapy. Tento proces zajišťuje, že hráč se nikdy neocitne v místnosti bez možnosti dalšího postupu.

Tento přístup jsem zvolil především proto, že zajišťuje, aby všechny místnosti byly propojené a hráč nemohl narazit na slepou uličku. Náhodné generování dveří v rámci fixního gridu (mřížky) poskytuje dostatečnou variabilitu rozložení hradu, ale zároveň zabraňuje vzniku nepropojených místností, které by mohly frustrovat hráče.

Zvažoval jsem i alternativní metody. Dijkstrův algoritmus, používaný v pathfindingu pro výpočet nejkratších cest (například ve strategických hrách jako Age of Empires), by zajistil efektivní propojení místností, ale je výpočetně náročnější a pro potřeby 2D adventury zbytečně komplexní [9].

5.2.3 Algoritmus rozmístění NPC a hádanek

Algoritmus rozmístění NPC a hádanek zajišťuje, že nehráčské postavy a výzvy jsou rozmístěny smysluplně a přispívají k hernímu zážitku [1]. NPC i hádanky jsou generovány náhodně, ale jejich pozice musí být logické a nesmí vést k situaci, kdy by hráč nemohl hádanku vyřešit nebo by se setkal s nepřekonatelnou překážkou.

Proces generování NPC začíná výběrem vhodného počtu postav, které se budou ve hře pohybovat. Každému NPC je následně přiřazena unikátní trasa, která se vytváří na základě předem vygenerované mapy místností [2]. Každá trasa NPC je definována jako seznam místností, které daná postava během hry navštíví, a pohyb mezi nimi probíhá vždy přes existující dveře. Tento systém zajišťuje, že se NPC nepohybují náhodně, ale mají předem stanovený směr, což přispívá k jejich konzistentnímu chování.

Aby se předešlo situaci, kdy by více NPC sdílelo stejnou trasu a docházelo by k jejich překrývání, algoritmus kontroluje již existující cesty a vybírá pro každou postavu jinou trasu. Pokud by přesto došlo k přiřazení dvou NPC na stejnou trasu, je jedné z postav přiděleno jiné startovní místo nebo je její cesta upravena tak, aby se v určitém bodě

rozcházela od ostatních. Kromě toho se NPC nikdy nepohybují náhodně v jedné místnosti, ale vždy sledují svou trasu, čímž se eliminuje riziko, že by se dlouhodobě shromažďovaly na jednom místě a blokovaly hráčovu cestu. Po rozmístění NPC probíhá generování hádanek, které jsou vybírány z připravené databáze. Každá hádanka je rozdělena na několik částí (například indicie vedoucí k heslu k zamčeným dveřím) a tyto části jsou strategicky rozmístěny do různých místností. Algoritmus dbá na to, aby žádná část hádanky nebyla umístěna v nepřístupné oblasti a aby se hráč nemohl dostat do situace, kdy mu chybí klíčová informace nutná k postupu. Hádanky jsou umísťovány do místností, kde je pravděpodobné, že se hráč zastaví a bude je hledat – například do knihoven, studoven nebo místností s NPC.

Celý tento systém byl navržen tak, aby hra působila dynamicky a podněcovala hráče k průzkumu. NPC nejsou statické, ale jejich trasy jsou předem definované a rozloženy rovnoměrně po mapě, což eliminuje situace, kdy by se více postav pohybovalo po stejné trase nebo zůstávalo na jednom místě. Tento přístup pomáhá udržet plynulost hry a zároveň dává hráči pocit, že postavy mají své vlastní cíle a nejsou pouze statickými objekty.

```
void GenerateNPCPaths()
{
    npcPaths = new List<List<Vector2Int>>();

    // Vytvoříme cesty pro NPC, které spojují všechny dostupné místnosti
    for (int i = 0; i < numberOfNPCs; i++) // Například vytvoříme cesty odpovídající počtu NPC
    {
        List<Vector2Int> path = new List<Vector2Int>();
        Vector2Int currentRoom = new Vector2Int(Random.Range(0, gridWidth), Random.Range(0, gridHeight)); // Začneme v náhodné místnosti

        // Pokud je startovní místnost zamčená, najdi jinou startovní místnost
        while (roomGrid[currentRoom.x, currentRoom.y] == null || roomGrid[currentRoom.x, currentRoom.y].isLocked)
        {
            currentRoom = new Vector2Int(Random.Range(0, gridWidth), Random.Range(0, gridHeight));
        }
```

Obrázek 5.2: Ukázka generování tras NPC

Alternativně jsem zvažoval použití NavMesh (Navigation Mesh), který se používá ve hrách s otevřenými světy, jako je Skyrim [3]. Tento systém umožňuje NPC volně se pohybovat prostředím podle dynamického pathfindingu, avšak pro potřeby 2D adventury se ukázal jako zbytečně komplexní a náročný na implementaci. Další možností bylo využití Markovových řetězců, což je pravděpodobnostní model [3] používaný pro simulaci pohybu postav ve hrách jako The Sims. Tento přístup by mohl zajistit realističtější a méně předvídatelný pohyb NPC, ale jeho implementace by byla výrazně složitější a obtížnější na ladění.

Zvolený systém naopak nabízí dostatečnou kontrolu nad pohybem NPC, minimalizuje neočekávané chování a zároveň zajišťuje variabilitu při každém spuštění hry.

5.2.4 Algoritmus generování úkolů a klíčových předmětů

Algoritmus generování úkolů a klíčových předmětů vytváří herní výzvy [4], které hráč musí řešit pro postup hrou. Úkoly jsou navrženy tak, aby hráč musel aktivně hledat nápovědy k hádankám nebo klíče k zamčeným dveřím, čímž je podporován průzkum herního světa a interakce s prostředím.

Proces generování začíná výběrem místnosti, která bude uzamčena a jejíž dveře budou vyžadovat klíč nebo správnou odpověď na hádanku. Aby se zajistilo, že tato místnost nebude hned na začátku hry a neomezila hráče v pohybu, algoritmus provádí několik kontrol. Nejprve se vyloučí místnosti, které se nacházejí v blízkosti startovní lokace hráče. Tento výběr probíhá podle předem definované vzdálenosti od výchozí místnosti (obvykle minimálně dvě místnosti daleko). Dále se ověřuje, že zamčená místnost není klíčovou spojnicí mezi ostatními místnostmi, což by mohlo vést k vytvoření neřešitelné situace, kde by hráč zůstal uvězněn v omezené části hradu. Pokud algoritmus narazí na situaci, kdy žádná místnost neodpovídá požadovaným kritériím, provede se náhodný výběr z místností, které jsou dostatečně vzdálené, ale stále umožňují přístup k jiným částem hradu.

Jakmile je zamčená místnost vybrána, algoritmus generuje klíč nebo nápovědy k hádance do jiné místnosti. Tato místnost musí být volně přístupná, což je zajištěno kontrolou její dosažitelnosti – tedy ověřením, že mezi startovní místností a místností s klíčem neexistuje žádná překážka v podobě zamčených dveří. Pokud by výběr vedl k nemožnosti získání klíče, je provedena oprava výběru, dokud není nalezena správná kombinace.

Další vrstvou variabilní hratelnosti je generování hádanek. Při každém spuštění hry se náhodně vybere jedna hádanka z databáze, přičemž její jednotlivé části jsou rozmístěny po různých místnostech. Tento systém zajišťuje, že hráč pokaždé hledá nápovědy na jiných místech, což zvyšuje znovuhratelnost. Hádanky jsou vybírány tak, aby odpovídaly atmosféře hry a hráč je mohl logicky odvodit z dostupných nápověd. Tento přístup jsem zvolil, protože umožňuje dynamickou změnu rozložení úkolů a hádanek, čímž zabraňuje tomu, aby hra působila předvídatelně. Náhodné rozmístění klíčových předmětů motivuje hráče k důkladnému prozkoumávání prostředí a zároveň zajišťuje, že hra je pokaždé jiná. Zároveň algoritmus obsahuje dostatek pojistek, aby se předešlo neřešitelným situacím, kdy by hráč zůstal uvězněn bez možnosti postupu.

Jako alternativu jsem zvažoval gramatické generování úkolů, které se používá například v sérii The Elder Scrolls [5]. Tento systém generuje úkoly na základě předdefinovaných pravidel a gramatických struktur, což umožňuje vytvářet velké množství variabilních a stále logických výzev. Pro menší adventurní hru by však tento přístup byl zbytečně

komplexní, protože by vyžadoval propracovaný systém syntaktických pravidel. Další možností byl ruční návrh úkolů, který se běžně používá v příběhově orientovaných hrách jako The Witcher 3 [5]. Tento přístup umožňuje detailnější vyprávění a větší důraz na atmosféru, avšak omezuje variabilitu a je výrazně náročnější na tvorbu obsahu.

Ve výsledku jsem se rozhodl pro kombinaci náhodného výběru s pevně danými pravidly, která zajišťují funkčnost generovaného obsahu a eliminují možné chyby spojené s nesprávným umístěním klíčových předmětů.

5.3 Výhody a nevýhody použitého přístupu

Hybridní přístup ke generování herního obsahu, který jsem zvolil pro svůj projekt, přináší několik významných výhod [6]. Především poskytuje značnou kontrolu nad kvalitou designu, protože ručně vytvořené místnosti umožňují zajistit vizuální i herní konzistenci. To je klíčové pro vytvoření koherentního a esteticky příjemného herního zážitku. Další významnou výhodou je variabilita při každém spuštění hry. Náhodné propojení místností a rozmístění objektů zajišťuje, že každé hraní bude unikátní, což výrazně zvyšuje znovuhratelnost. V neposlední řadě tento přístup umožňuje lepší balancování hratelnosti, protože náhodné generování udržuje určitou výzvu pro hráče, aniž by hra byla příliš obtížná nebo příliš jednoduchá.

Na druhou stranu hybridní přístup má i své nevýhody. Jednou z hlavních je potřeba ruční přípravy místností, která zabere více času než plně procedurální generování. Pro každou místnost je nutné vytvořit samostatný prefab, navrhnout jeho vzhled a definovat možná propojení, což představuje značnou práci ve fázi vývoje. Další nevýhodou jsou potenciální problémy s propojením místností. Algoritmus musí zajistit, že hráč neuvízne v nepropojené části mapy, což vyžaduje pečlivé testování a ladění. Třetí nevýhodou je vyšší náročnost na ladění celého systému. Náhodné generování propojení znamená, že se občas mohou objevit neočekávané chyby, jako například NPC stojící na stejném místě nebo hádanky, které nelze vyřešit kvůli nevhodnému rozmístění částí.

5.4 Srovnání s existujícími hrami

Pro lepší pochopení mého přístupu je užitečné srovnat jej s existujícími hrami, které využívají podobné techniky.

Hra Spelunky (2008) používá procedurální generování dungeonů [7], kde je každé kolo unikátní, ale pravidla generování zajišťují, že hra je stále hratelná. Na rozdíl od mého přístupu je generování v Spelunky plně automatizované, zatímco v mé hře jsou místnosti ručně vytvořené a pouze jejich propojení je náhodné. To poskytuje vyšší míru kontroly nad designem jednotlivých místností, ale omezuje možnosti variability.

Darkest Dungeon (2016) využívá hybridní přístup [8] podobný tomu, který jsem zvolil pro svou hru. Některé prvky jsou procedurálně generované, ale základní struktura dungeonů je pevně daná. Tento přístup umožňuje zachovat atmosféru a náročnost hry, ale zároveň poskytuje dostatečnou variabilitu pro opakované průchody.

The Binding of Isaac (2011) používá BSP Tree dungeon generation [9], což je alternativa k mému přístupu. Tento způsob generování je efektivní pro rogue-like hry, ale nehodil by se pro adventuru, kde je potřeba více kontroly nad rozmístěním místností a jejich propojením. The Binding of Isaac generuje kompletně nové levely při každém průchodu, což maximalizuje znovuhratelnost, ale může vést k méně konzistentnímu hernímu zážitku.

6 Beta test

Průběh beta testingu hry *Castle of Ghosts* probíhal v několika fázích a poskytl cenné informace pro další vývoj a optimalizaci projektu. Testování bylo zaměřeno na identifikaci chyb, ověření funkčnosti herních mechanik, uživatelské přívětivosti a celkového herního zážitku.

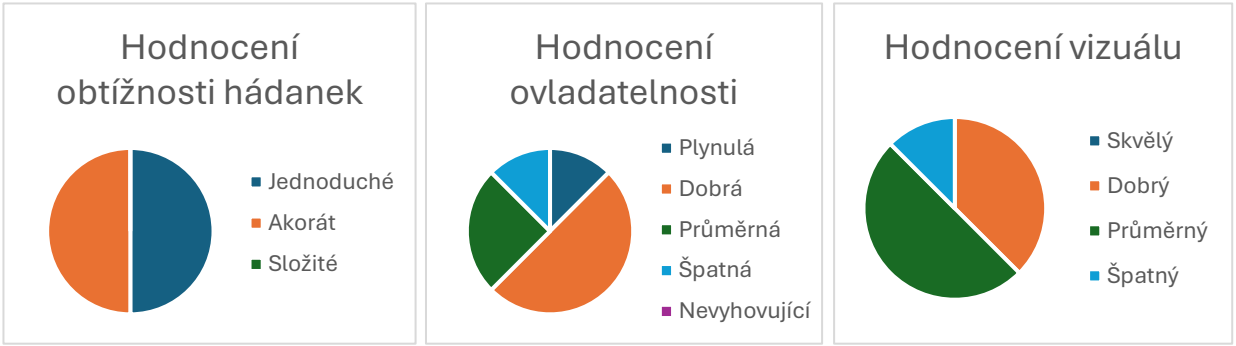
6.1 Výsledky testování

Beta testingu se zúčastnilo 8 testerů. Každý tester měl za úkol projít hru, vyřešit dostupné úkoly a zaznamenat své dojmy do připraveného dotazníku.

Všichni testeři odehráli hru několikrát do konce, aby zjistili funkčnost náhodného generování. Vizuální stránka hry byla z většiny hodnocena jako průměrná. To poukazuje na potřebu zlepšení grafických prvků, textur nebo animací. Ovládání bylo hodnoceno jako dobré. To naznačuje, že ovládání je funkční. Hádanky byly hodnoceny jako jednoduché a akorát složité. To ukazuje na vyváženou obtížnost hádanek pro cílovou skupinu hráčů.

Mezi největší technický problém patřil bug v inventáři. Kdy se do inventáře přidávala pouze jedna nápověda, která nahrazovala předchozí. Dále testeři zaznamenali drobné vizuální nedostatky, které narušovaly estetiku hry. Všechny bugy ať už větší či menší byly opraveny. Dále testeři doporučili přidat animace a zvukové efekty pro atmosféru nebo rozšířit nabídku úkolů.

Beta test poskytl cenné informace o stabilitě hry i uživatelském zážitku. Díky zpětné vazbě od testerů byla hra výrazně vylepšena před jejím uvedením do finální verze.



Obrázek 6.1: Grafy výsledků hodnocení

Závěr

Maturitní projekt *Castle of Ghosts* splnil svůj hlavní cíl – vytvoření funkční 2D hry, která využívá náhodné generování obsahu. Hra obsahuje základní mechaniky, jako je pohyb hráče, interakce s objekty, řešení hádanek a komunikace s NPC. Během vývoje byly implementovány klíčové prvky, jako je inventář a dynamické generování místností.

Projekt mi poskytl cenné zkušenosti nejen v oblasti programování v jazyce C# a práci s herním enginem Unity, ale také v plánování projektu. Přestože hra splňuje požadavky zadání, vidím prostor pro její další rozšíření – například přidání nových herních mechanik, NPC nebo map.

Celkově hodnotím projekt jako úspěšný a věřím, že získané zkušenosti budou přínosné pro mé budoucí profesní směřování.

Seznam literatury

[1] ADAMS, T. Generating Endless Worlds: The Past, Present and Future of Procedural Content Generation. IEEE Transactions on Computational Intelligence and AI in Games, 2017, vol. 9, no. 3, s. 215–229.

[2] DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1959, vol. 1, s. 269–271.

[3] DORMANS, J. Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games. In: Proceedings of the 2010 Workshop on Procedural Content Generation in Games. New York: ACM Press, 2010.

[4] GAME-ACE. Procedural Generation in Games [online]. 2024 [cit. 2025-03-18]. Dostupné z: https://game-ace.com/blog/procedural-generation-in-games

[5] GAMERANT. 6 Earliest Games That Used Procedural Generation [online]. 2023 [cit. 2025-03-18]. Dostupné z: https://gamerant.com/earliest-games-procedural-generation

[6] GAMEDEV STACK EXCHANGE. Creating and Connecting Rooms for a Roguelike [online]. 2013 [cit. 2025-03-18]. Dostupné z: https://gamedev.stackexchange.com/questions/50570/creating-and-connecting-rooms-for-a-roguelike

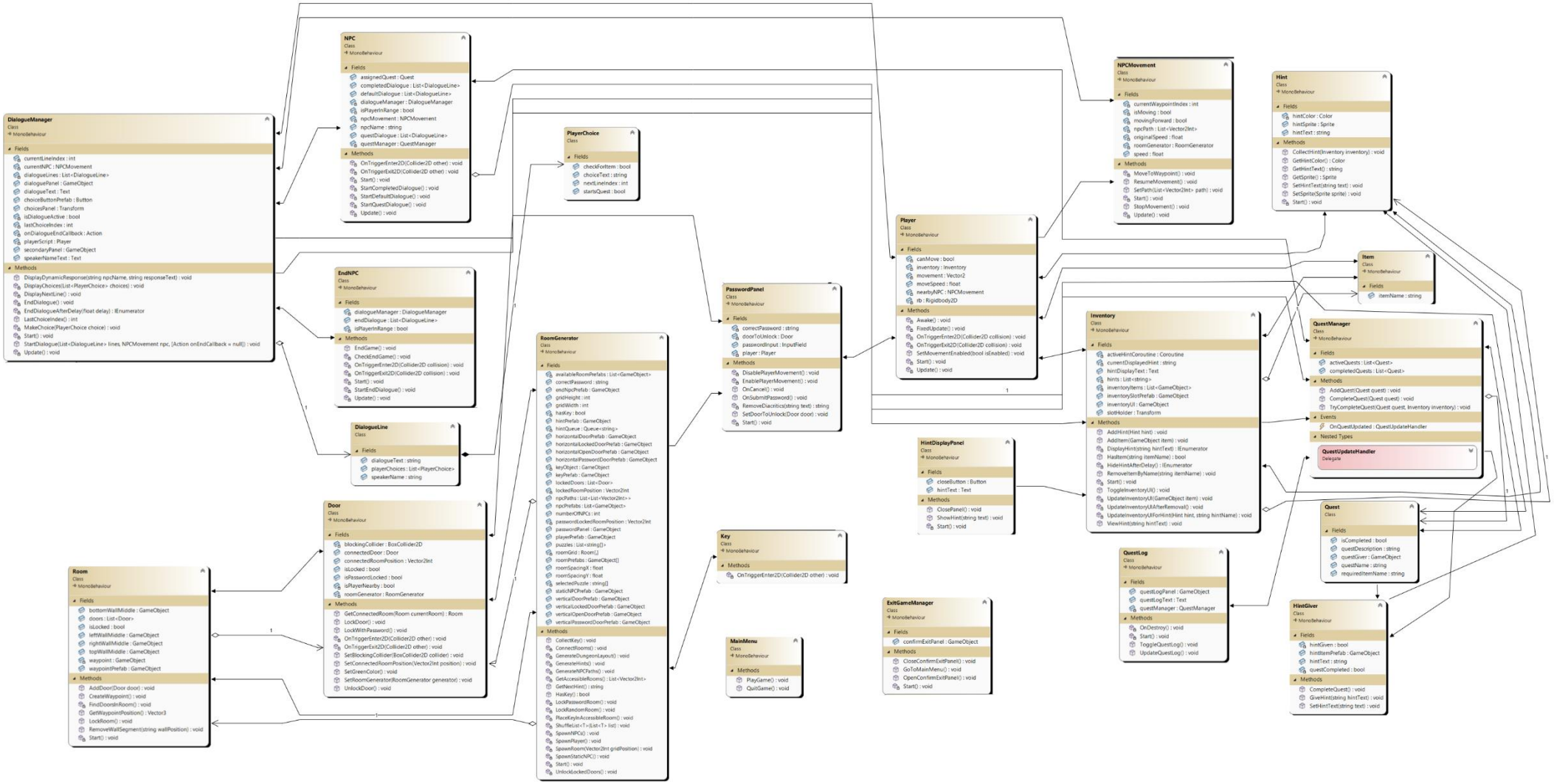
[7] LENOVO. Understanding Procedural Generation in Games [online]. 2024 [cit. 2025-03-18]. Dostupné z: https://www.lenovo.com/us/en/glossary/procedural-generation

[8] PERLIN, K. An Image Synthesizer. Computer Graphics, 1985, vol. 19, no. 3, s. 287-296.

[9] ROGUEBASIN. Basic BSP Dungeon generation [online]. 2020 [cit. 2025-03-18]. Dostupné z: https://www.roguebasin.com/index.php/Basic_BSP_Dungeon_generation

Seznam obrázků

Obrázek 3.1: Ukázka ze hry 6
Obrázek 3.2: Postava ducha 6
Obrázek 4.1: Use case diagram 8
Obrázek 5.1: Ukázka mazání přebytečných stěn, po vytvoření dveří 12
Obrázek 5.2: Ukázka generování tras NPC 14
Obrázek 6.1: Grafy výsledků hodnocení 18



Příloha I: Class diagram projektu