DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, Pardubice

AI generovaný prostor vytvořený v rámci 3D hry

Příjmení, jméno: *Prudilová Vendula*

Třída: *4.A*

Studijní obor: *Informační technologie 18-20-M/01*

Školní rok: *2025/2026*

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení: Vendula Prudilová

Pro školní rok: 2025/2026

Třída: 4.A

Obor: Informační technologie 18-20-M/01

Téma práce: AI generovaný prostor vytvořený v rámci 3D hry.

Vedoucí práce: Mgr. Jan Mottl

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

Projekt je zaměřen na využití umělé inteligence pro generování herního prostoru ve 3D hře. Půjde o rogue-like titul. AI na základě definovaných pravidel vytvoří textový návrh úrovně, který bude následně převeden do plně funkční 3D mapy. Do budoucna může být implementována i funkce, která hráči umožní komunikovat s AI a vytvořit si vlastní mapu.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

Září – rešerše a vytvoření game design dokumentu

Říjen – implementace základních herních mechanik (pohyb, akce)

Listopad – generování 3D mapy z textového souboru

Prosinec – dokončení základní verze hry bez integrace AI

Leden – implementace AI modelu pro tvorbu map na základě zadání

Únor – beta testování hry a ladění chyb

Březen – sepsání závěrečné dokumentace

Prohlášení

Prohlašuji, že jsem svou práci vypracovala samostatně a použila jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

V Pardubicích dne _____ _____

Vendula Prudilová

Poděkování

Ráda bych poděkovala svému vedoucímu projektu Mgr. Janu Mottlovi za cenné rady, trpělivost a podporu při realizaci tohoto projektu.

Zvláštní poděkování také patří mé rodině za podporu během tohoto školního roku i celého studia.

Anotace

Tato práce se zabývá návrhem a vývojem počítačové hry *Peak* v herním enginu Unity 2022.3 LTS. Těžiště projektu spočívá v generování herních map pomocí velkého jazykového modelu GPT-4o-mini prostřednictvím OpenAI API. Model na základě strukturovaného textového promptu generuje ASCII mapu dungeonu o rozměrech 12×7 buněk, z níž Unity ProBuilder dynamicky sestaví 3D prostředí. Systém zahrnuje sanitizaci výstupu, opravu konektivity flood fill algoritmem a doplnění chybějících herních prvků. Výsledkem je funkční first-person dungeon shooter s roguelike prvky a vysokou znovuhratelností.

Klíčová slova: Procedurální generování obsahu, umělá inteligence, generování map, GPT-4o-mini, Unity, herní vývoj, dungeon crawler, roguelike

Abstract

This thesis presents the design and development of a computer game titled *Peak* built in the Unity 2022.3 LTS game engine. The central focus lies in AI-driven map generation using the GPT-4o-mini large language model via the OpenAI API. Given a structured text prompt, the model generates a 12×7 ASCII dungeon map which Unity ProBuilder then assembles into a 3D environment at runtime. The system includes output sanitisation, flood-fill connectivity repair, and automatic placement of missing game elements. The result is a functional first-person dungeon shooter with roguelike elements and high replayability.

Keywords: Procedural content generation, artificial intelligence, map generation, GPT-4o-mini, Unity, game development, dungeon crawler, roguelike

Obsah

Úvod

Tato maturitní práce se zabývá využitím umělé inteligence pro generování herního prostoru ve 3D hře v enginu Unity. Jde o first-person dungeon shooter, jehož hlavním technickým přínosem je systém generování herních map pomocí jazykového modelu GPT-4o-mini přes OpenAI API.

Myšlenka automatického generování map mě zaujala proto, že řeší jeden z nejčastějších problémů menších herních projektů: opakovanost. Pokud hra vždy vypadá stejně, hráče brzy přestane bavit. Procedurální generování tento problém obchází tím, že každý run přináší jiné rozmístění místností, nepřátel i předmětů.

Teoretická část práce popisuje, jak se k automatizované tvorbě herního obsahu přistupuje obecně, od klasických algoritmů jako Perlinův šum nebo BSP stromy až po moderní metody využívající strojové učení. Jsou zde také rozebrány konkrétní hry, které tyto technologie používají. Praktická část pak dokumentuje samotný návrh a vývoj hry: herní koncept, použité nástroje, průběh implementace a dosažené výsledky.

Cílem bylo vytvořit funkční hratelný prototyp, ve kterém systém AI generování map funguje spolehlivě, tedy dokáže opravit chybný výstup modelu, zajistit průchodnost dungeonu a doplnit chybějící herní prvky.

1 Procedurální generování obsahu

Tato kapitola se věnuje procedurálnímu generování obsahu jako disciplíně herního vývoje. Jsou zde vysvětleny základní pojmy a motivace pro jeho použití a popsány klasické algoritmy, které se v praxi nejčastěji využívají.

1.1 Definice a motivace

Procedurální generování obsahu (PCG, z anglického *Procedural Content Generation*) označuje automatizované procesy, které pomocí algoritmů a pseudonáhodnosti vytvářejí herní obsah – mapy, terény, příběhy nebo textury, bez nutnosti vše navrhovat ručně. Shaker, Togelius a Nelson definují PCG jako *„algoritmy pro tvorbu herního obsahu s omezeným nebo nepřímým vstupem uživatele"* [1].

Přístupy k PCG se dělí na dvě základní skupiny. Konstruktivní metody generují obsah přímo v jednom průchodu podle pevně daných pravidel; jsou rychlé, ale výstup záleží čistě na nastavených parametrech. Prohledávací metody naproti tomu opakovaně vytvářejí varianty a hodnotí je pomocí fitness funkce, čímž dosahují lepší kvality, ovšem za cenu vyšší výpočetní náročnosti [4].

Zájem o PCG vzrostl hlavně díky tomu, že umožňuje generovat velké množství různorodého obsahu z relativně malé sady pravidel. Pro malé vývojářské týmy nebo jednotlivce je to zejména způsob, jak udělat hru znovuhratelnou bez nutnosti ručně navrhovat desítky unikátních map.

1.2 Klasické algoritmy

Tato podkapitola popisuje klasické deterministické algoritmy používané pro procedurální generování herního obsahu, jako jsou Perlinův šum, buněčné automaty, BSP stromy a další.

1.2.1 Perlinův šum

Perlinův šum, popsaný Kenem Perlinem v roce 1985 [2], je gradientní šumová funkce, která produkuje přirozené, organicky vyhlížející terény a textury. Výsledek je hladký a koherentní, takže se hodí například pro generování horské krajiny. Vrstvením více oktáv s různými frekvencemi a amplitudami vzniká tzv. fraktální Brownův pohyb, který je ve hrách standardním nástrojem pro terén.

1.2.2 Buněčné automaty

Buněčné automaty se v PCG používají hlavně pro generování jeskyní [7]. Základní princip je jednoduchý: mřížka se inicializuje náhodně a pak se opakovaně aplikují pravidla, například „přežij, pokud má buňka alespoň čtyři sousedy“. Po několika iteracích vznikají přirozené jeskynní systémy bez ostrých hran a pravých úhlů.

1.2.3 BSP stromy

Dělení binárního prostoru (*Binary Space Partitioning*) rekurzivně rozděluje prostor na menší části. V herním vývoji se tato technika používá pro generování dungeonů: prostor se opakovaně dělí, v každé části se umístí místnost a podél hranic stromu se vedou chodby. Výsledkem jsou přehledné dungeony se zaručenou průchodností.

1.2.4 Algoritmus diamant-čtverec a L-systémy

Algoritmus diamant-čtverec je rekurzivní metoda pro generování výškové mapy terénu, střídají se v něm dva kroky (diamant a čtverec), přičemž v každém kroku se přidává čím dál menší náhodná odchylka. L-systémy jsou formální gramatiky původně navržené pro modelování růstu rostlin; v hrách se uplatňují při procedurálním generování vegetace nebo větevnatých struktur.

1.2.5 Srovnání klasických algoritmů

Každý z popsaných algoritmů se hodí pro jiný typ herního obsahu, žádný z nich není univerzální. V praxi se proto často kombinují.

Tabulka 1 Srovnání klasických algoritmů

Algoritmus	Typické využití	Výhody	Nevýhody
Perlinův šum	Terény, textury	Plynulý, organický výsledek	Nevhodný pro strukturované prostory
Buněčné automaty	Jeskyně	Přirozené, nepravidelné tvary	Nelze zaručit průchodnost
BSP stromy	Dungeony	Zaručená průchodnost	Pravidelná, méně organická struktura
Diamant-čtverec	Výškové mapy	Jednoduché škálování detailu	Pouze pro výškové mapy
L-systémy	Vegetace, větve	Přirozený růst struktur	Složitá konfigurace pravidel

2 AI generování map

Zatímco klasické algoritmy pracují s explicitně definovanými pravidly, přístupy využívající umělou inteligenci dokáží tato pravidla odvozovat z dat nebo hodnotit kvalitu generovaného obsahu z hlediska hratelnosti. Tato kapitola popisuje nejvýznamnější metody: algoritmus Wave Function Collapse, generativní adversariální sítě, PCGML, zpětnovazební učení a evoluční algoritmy.

2.1 Wave Function Collapse

Wave Function Collapse (WFC) je algoritmus pro generování vzorů s omezeními, který publikoval Maxim Gumin v roce 2016 [3]. Princip vychází z kvantové mechaniky: každá buňka výstupní mřížky je zpočátku v superpozici všech přípustných stavů a postupně „kolabuje" do jednoho konkrétního stavu na základě toho, co bylo zvoleno v okolních buňkách.

Algoritmus existuje ve dvou variantách. Overlapping Model analyzuje vstupní vzorový obrázek a z něj extrahuje pravděpodobnostní tabulku sousedností. Simple Tiled Model pracuje s ručně definovanou sadou dlaždic a pravidly jejich sousedství, tato varianta je pro herní vývoj praktičtější.

Průběh algoritmu vypadá takto: nejprve se vybere buňka s nejnižší entropií (nejméně přípustnými stavy), zvolí se jeden z těchto stavů a z kolapslé buňky se šíří omezení do sousedů. Postup se opakuje, dokud nejsou obsazeny všechny buňky. Pokud se algoritmus dostane do slepé uličky (*contradiction*), vrátí se zpět nebo začne znovu.

WFC je atraktivní proto, že garantuje lokální koherenci výstupu a nevyžaduje trénovací dataset. Stačí vzorový obrázek nebo ručně definovaná pravidla. Používají ho například hry *Caves of Qud* nebo *Bad North* [3]. Nevýhodou je, že kvalita výstupu závisí zcela na kvalitě vzorové sady.

2.2 Generativní adversariální sítě (GAN)

Generativní adversariální sítě navrhli Goodfellow et al. v roce 2014 [8]. Architektura GAN se skládá ze dvou sítí: generátor produkuje syntetická data ze šumového vektoru, zatímco diskriminátor se snaží rozlišit reálná data od generovaných. Trénink probíhá jako minimax, generátor se snaží diskriminátor přelstít, diskriminátor se mu snaží vyrovnat.

V kontextu herních map byl zásadní práce Volze et al. (2018) [6], kteří trénovali DCGAN na levelech *Super Mario Bros.* a pak evolučně prohledávali latentní prostor generátoru s cílem nalézt hratelné úrovně. Ukázalo se, že sousední body v latentním prostoru odpovídají sémanticky podobným levelům, takže lze plynule přecházet mezi různými typy dungeonů.

Hlavní nevýhodou GAN je nestabilní trénink a nutnost velkého tréninkového datasetu. Garance hratelnosti výstupu navíc není přímočará, vyžaduje buď post-processing, nebo integraci herního simulátoru přímo do trénovací smyčky.

2.3 PCGML

Summerville et al. (2018) [5] zavedli pojem PCGML (*Procedural Content Generation via Machine Learning*) jako zastřešující označení pro přístupy, kde jsou modely strojového učení trénovány na existujícím herním obsahu a pak generují obsah nový.

Zvláštní pozornost si v rámci PCGML zaslouží rekurentní neuronové sítě (LSTM). Sítě trénované na sekvencích dlaždic ze hry *Super Mario Bros.* nebo *The Legend of Zelda* dokážou generovat nové úrovně, které zachovávají stylistické i strukturální vlastnosti originálu. Variační autoencodéry (VAE) nabízejí alternativu s plynulejším a strukturovanějším latentním prostorem, vhodným pro interaktivní editaci map.

2.4 Zpětnovazební učení pro level design

Zpětnovazební učení (*Reinforcement Learning*, RL) přistupuje k problému z jiného úhlu: agent se učí navrhovat herní prostředí tak, aby maximalizoval odměnu. Odměnová funkce může zachycovat hratelnost, vyváženost obtížnosti nebo délku průměrné herní session.

Přístup RLPCG trénuje agenta jako interního level designéra – agent navrhne mapu, simulátor ji vyhodnotí (třeba spuštěním A* prohledávání) a výsledek slouží jako zpětná vazba. Výhodou je přímá optimalizace vůči herním metrikám bez nutnosti existujícího datasetu, nevýhodou jsou vysoké výpočetní nároky a pomalá konvergence.

2.5 Evoluční algoritmy

Search-based PCG (SBPCG) využívá evoluční algoritmy k prohledávání prostoru možných map s cílem najít ty, které splňují definovaná kritéria [4]. Fitness funkce může hodnotit průchodnost, estetiku nebo herní rovnováhu. Hybridní přístupy kombinující GAN s evolučními algoritmy, jak ukázali Volz et al. [6], jsou zvláště efektivní: GAN latentní prostor slouží jako kompaktní prohledávací prostor a evoluční algoritmus v něm naviguje směrem k požadovaným vlastnostem.

2.6 Srovnání přístupů

Žádný z popsaných přístupů nebyl pro tento projekt přímo použit. WFC by vyžadovalo ruční definici rozsáhlé sady dlaždic a pravidel, GAN ani PCGML nebyly reálné kvůli absenci tréninkového datasetu. Zpětnovazební učení a evoluční algoritmy pak představují vysokou výpočetní náročnost nevhodnou pro real-time generování.

Místo toho byla zvolena metoda promptování velkého jazykového modelu GPT-4o-mini, který na základě strukturovaného textového promptu generuje ASCII mapu dungeonu. Tento přístup nevyžaduje trénink ani dataset a je snadno integrovatelný přes API. Jeho nevýhodou je závislost na kvalitě promptu a nutnost post-processingu pro opravu chybného výstupu.

Tabulka 2 Srovnání přístupů

Přístup	Klíčová vlastnost	Výhody	Nevýhody
Wave Function Collapse	Constraint propagation ze vzorů	Bez datasetu, garantovaná koherence	Contradiction, závislost na vzorové sadě

Přístup	Klíčová vlastnost	Výhody	Nevýhody
GAN	Adversariální trénink na datech	Varieta, interpolovatelný latentní prostor	Nestabilní trénink, nutný dataset
LSTM / PCGML	Sekvenční generování z dat	Zachovává styl	Slabá kontrola hratelnosti
Zpětnovazební učení	Optimalizace herních metrik	Bez datasetu	Pomalé, náročné na simulaci

3 Existující hry

Než jsem začala navrhovat vlastní systém generování map, prohlédla jsem si, jak tento problém řeší komerčně úspěšné hry. Každá ze zmíněných her přistupuje k PCG trochu jinak a každá z nich mi dala jiný pohled na to, co funguje a co ne.

3.1 Minecraft

Minecraft [I11] je nejznámějším příkladem procedurálně generovaného světa. Terén vzniká vrstvením Perlinova šumu v různých frekvencích, výsledkem je výšková mapa, na níž se pak generují biomy, jeskynní systémy a různé struktury. Každý svět je jednoznačně určen číselným seedem, takže jej lze sdílet nebo reprodukovat. Od verze 1.18 jeskyně využívají třírozměrný Simplexní šum, který umožňuje tvorbu prostornějších podzemních komplexů. Minecraft dokazuje, že i relativně jednoduchá kombinace šumových funkcí dokáže při správné parametrizaci vytvořit svět, který hráče zabaví na stovky hodin.

3.2 No Man's Sky

No Man's Sky [I12] generuje celou galaxii s obrovským množstvím planet, z nichž každá má vlastní terén, faunu, floru a atmosféru určené deterministickým seedem. Pro terén se používá voxelová reprezentace s Marching Cubes triangulací. Tato hra je dobrým příkladem

toho, co se stane, když se PCG použije ve velké míře bez dostatečného designu: astronomický počet planet přichází za cenu opakujících se vzorů, které mohou po čase snižovat dojem z originality.

3.3 Spelunky

Spelunky [I8] řeší věčný problém PCG her jinak, kombinuje ruční návrh s procedurálním skládáním. Každá úroveň se skládá ze čtvercové mřížky šablon o rozměrech 4×4 místnosti. Šablony jsou ručně navrženy tak, aby každá zaručovala průchodnost, a procedurální složka spočívá v jejich náhodném výběru a rozmístění. Výsledek je pokaždé jiný, ale vždy hratelný. Spelunky je pro mě zajímavý proto, že ukazuje, že i relativně jednoduchá pravidla mohou vést ke složitým a zábavným výsledkům.

3.4 Hades a Dead Cells

Hades [I13] a *Dead Cells* [I9] sdílejí podobný přístup: herní prostředí se skládá z ručně navržených místností, které jsou pak procedurálně vybírány a propojovány při každém běhu. Tím je zachována vysoká vizuální a herní kvalita (místnosti jsou ručně vytvořená díla) a zároveň je zajištěna dostatečná rozmanitost pro znovuhratelnost. V případě *Hades* jsou navíc procedurálně generovány dialogy a odměny, takže každý run působí trochu jinak.

3.5 Projekt Peak v tomto kontextu

Projekt Peak v tomto kontextu Na rozdíl od výše zmíněných her, které využívají šum, šablony nebo grafové algoritmy, jsem se v projektu Peak rozhodla pro přístup přes jazykový model GPT-4o-mini. Model dostane strukturovaný textový prompt a vrátí ASCII mapu dungeonu. Tento přístup mi přišel zajímavý proto, že nevyžaduje trénování vlastního modelu ani ruční tvorbu sady pravidel, stačí přesně popsat, co má mapa obsahovat. Nevýhodou je závislost na externím API a nutnost opravovat chybné výstupy, což vedlo k vývoji trojstupňového post-processingu popsaného v kapitole Metodika a řešení.

4 Game Design Document

4.1 Přehled hry

Tabulka 3 Přehled hry

Název hry	Peak
Žánr	3D first-person shooter / dungeon crawler s roguelike prvky
Platforma	PC (Windows)
Engine	Unity 2022.3.62f1 LTS
Programovací jazyk	C#
Věkové hodnocení	PEGI 12 (fantasy násilí)
Délka herní session	5–15 minut (jeden run)

4.1.1 Koncept

Peak je 3D first-person dungeon crawler, ve kterém je každá mapa žalářů generována umělou inteligencí prostřednictvím jazykového modelu GPT-4o-mini. Hráč se ocitá v procedurálně sestaveném dungeonu, jehož cílem je zničit všechny nepřátelské krystaly-spawners dříve, než podlehne nepřátelům. Každé nové spuštění přináší odlišné rozmístění místností, nepřátel i předmětů.

4.1.2 Logline

Hráč prozkoumává náhodně generovaný dungeon složený z propojených místností, likviduje nepřátelské krystaly a sbírá předměty, vše v dynamickém prostředí, jehož podobu pokaždé navrhuje umělá inteligence.

4.2 Herní mechaniky

Tato podkapitola popisuje herní mechaniky projektu Peak, základní herní smyčku, pohyb a ovládání, herní systémy, postavu hráče, nepřátele a podmínky výhry a prohry.

4.2.1 Základní herní smyčka (Core Loop)

1. Spuštění – AI vygeneruje novou ASCII mapu (až 15 pokusů), ProBuilder sestrojí 3D dungeon.
2. Průzkum – hráč se pohybuje v first-person perspektivě, prozkoumává místnosti.
3. Boj – nepřátelé se periodicky spawnují z krystalů, hráč střílí a krystaly ničí.
4. Sběr předmětů – hráč sbírá léčivé nebo energetické předměty stiskem klávesy E.
5. Výsledek – zničení všech spawnerů = výhra, smrt hráče = prohra. Hra nabídne restart s novou mapou.

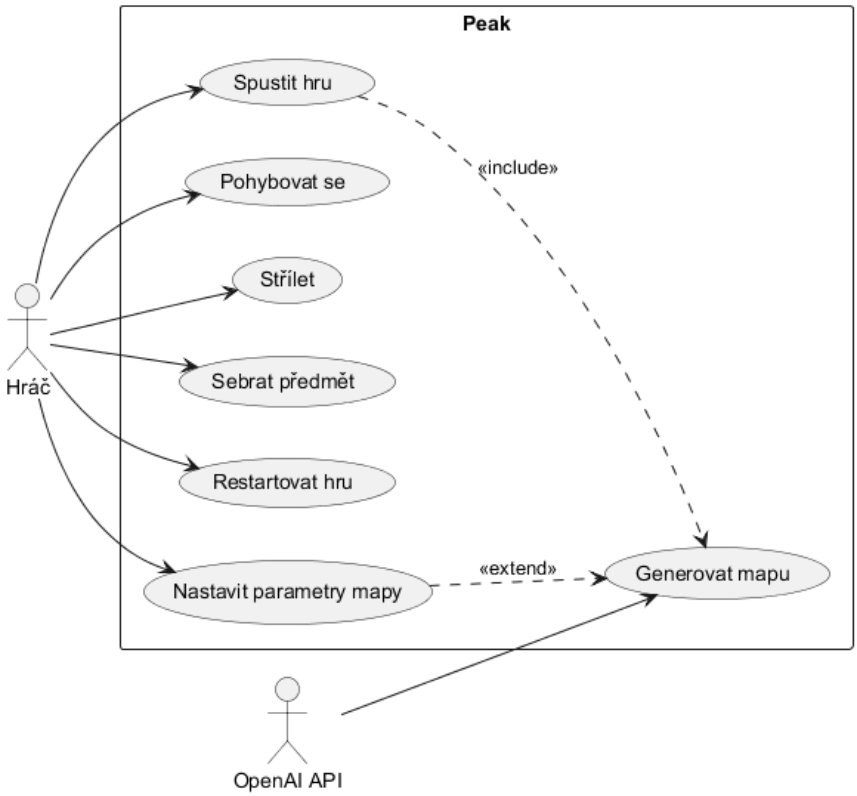
4.2.2 Pohyb a ovládání

Tabulka 4 Pohyp a ovládání

Akce	Ovládání
Pohyb	W / A / S / D
Pohled (kamera)	Pohyb myší
Sprint	Levý Shift (spotřebuje staminu)
Střelba	Levé tlačítko myši
Sebrat předmět	E (při kontaktu s předmětem)

- **Chůze:** rychlost 5 jednotek/s
- **Sprint:** rychlost 10 jednotek/s, spotřeba staminy 25/s, při vyčerpání staminy se rychlost sníží na 4 jednotky/s
- Kurzor je uzamčen, pohybem myši se otáčí celá postava i kamera

4.2.3 Diagram případů použití



Obrázek 1 Use case diagram

4.2.4 Herní systémy

Zdraví (Health)

- Maximální hodnota: 100 HP
- Poškození způsobují nepřátelé kontaktem nebo projektily
- Pasivní regenerace dostupná po prodlevě bez zásahu (rychlost 10 HP/s)
- HUD zobrazuje plnicí indikátor zdraví

Stamina

- Maximální hodnota: 100
- Regenerace v klidu: 20/s

- Sprint vyčerpá staminu, po úplném vyčerpání se sprint zablokuje dokud stamina plně nedobije
- HUD zobrazuje plnicí indikátor staminy

Střelba

- Každý klik levým tlačítkem myši vystřelí projektil odpovídající rychlostí vpřed

Spawneři nepřátel (Enemy Spawners / Krystaly)

- Na mapě je umístěn 1–N spawnerů (konfigurovatelný počet)
- Každý spawner má 5 HP a každých 25 sekund spawnuje nového nepřítele
- Zničení spawneru ho odstraní ze scény
- Zničení všech spawnerů spustí herní výhru

Předměty (Pickup Items)

Tabulka 5 Předměty (Pickup Items)

ID v mapě	Typ předmětu	Efekt při sebírání (klávesa E)
1	Health	+25 HP
2	Energy	+25 staminy
3	Both	+25 HP i +25 staminy

4.2.5 Postava hráče

- First-person kamera, Rigidbody fyzika
- Komponenty: Player, HealthController, StaminaController, Gun
- Animátor reaguje na velikost pohybového vektoru (parametr Speed)

4.2.6 Nepřátelé

- Typ nepřítele: humanoidní postava (Enemy_Woman)
- Nepřátelé jsou spawnováni krystalym, jejich počet roste s časem

- Způsobují poškození hráče při kontaktu

4.2.7 Výhra a prohra

Tabulka 6 Výhra a prohra

Podmínka	Výsledek
Zničení všech spawnerů	Výhra – zobrazí se Win Panel
HP hráče klesne na 0	Prohra – zobrazí se Lose Panel
Restart	Nová mapa vygenerovaná AI

4.3 Mapa a úrovně

4.3.1 Struktura mapy

Herní mapa je reprezentována jako ASCII mřížka o rozměrech **12 sloupců × 7 řádků**. Každá buňka odpovídá jedné místnosti o fyzické velikosti 15 × 15 × 11 jednotek Unity.

Tabulka 7 Struktura mapy

Znak	Význam
#	Průchozí podlaha (místnost)
.	Prázdné místo (void – nic se nestaví)
P	Spawn hráče (právě jeden)
*	Spawner nepřítele (krystal)
1	Předmět: léčení (+25 HP)
2	Předmět: energie (+25 staminy)
3	Předmět: oba efekty

Příklad mapy:

.....

.##.....*..

.#P######...

.##....##...

..####..##..

..*.#....*..

....#.......

4.3.2 Generování map pomocí AI

Systém generování map zasílá strukturovaný prompt modelu GPT-4o-mini přes OpenAI API. Model vrátí ASCII mapu, která je dále zpracována třemi kroky:

1. **Sanitizace** – ověření a oprava rozměrů (přesně 12×7), nahrazení neplatných znaků, zajištění existence právě jednoho P.
2. **Oprava konektivity** – flood fill z pozice hráče, buňky nedosažitelné z P jsou převedeny na void..
3. **Doplnění chybějících prvků** – pokud AI nevygenerovala dostatečný počet spawnerů nebo předmětů, systém je umístí náhodně na volné dlaždice #.

Generátor provede až 15 pokusů. Pokud žádný pokus nevede k validní mapě, použije se záložní uložená mapa ze složky Resources.

4.3.3 Zaručené invarianty každé mapy

- Existence právě jednoho spawnu hráče (P)
- Všechny herní buňky jsou dosažitelné z pozice hráče (flood fill oprava)
- Minimálně jeden spawner nepřítele
- Mapa má přesně 12 sloupců a 7 řádků

4.3.4 Fyzická konstrukce místností (ProBuilder)

Každá místnost je sestavena dynamicky pomocí Unity ProBuilder:

- Podlaha a strop jako kvádry (stíny vypnuty pro výkon)
- Pokud sousední buňka je místnost → dveřní otvor (šířka 8 j., výška 5 j.), jinak plná zeď

- Bodové světlo (PointLight) v každé místnosti, barva teplé svíčky
- Podlaha rozšířena o 1 jednotku do stran pro kontinuální NavMesh

4.4 Vizuální styl

4.4.1 Celkový vizuální styl

Low-poly/stylizovaný 3D, minimalistická estetika. Místnosti jsou bílé, stíny jsou vypnuty pro výkon a čistý vzhled. Osvětlení teplými bodovými světly vytváří atmosféru dungeonu.

4.4.2 Barevná paleta

- Stěny a strop: bílá
- Podlaha: tmavě šedá
- Světla místností: teplá žlutooranžová (RGB 1.0, 0.9, 0.75)
- UI: minimalistické, tmavé pozadí s barevnými indikátory

4.4.3 UI/HUD

- Indikátor zdraví: plnicí Image (červená)
- Indikátor staminy: plnicí Image (zelená)
- Win Panel/Lose Panel: překrytí přes celou obrazovku s tlačítkem Restart

4.4.4 Zvuk

Zvukový design není v aktuální verzi implementován.

4.5 Cílová skupina

4.5.1 Primární skupina

Casual a indie hráči ve věku 14–25 let se zájmem o first-person shootery a roguelike hry. Hráči, kteří oceňují krátké herní session s vysokou znovuhratelností díky procedurálnímu generování.

4.5.2 Referenční tituly

Hru *Peak* budou pravděpodobně oslovovat hráči her jako Spelunky, Dead Cells nebo Enter the Gungeon, zejména z důvodu sdíleného konceptu každého runu jako unikátní výzvy s procedurálně generovaným prostředím.

4.6 Technické požadavky

Tabulka 8 Technické požadavky

Požadavek	Minimální konfigurace	Doporučená konfigurace
Operační systém	Windows 10 (64-bit)	Windows 10/11 (64-bit)
Procesor	Intel Core i3 / AMD Ryzen 3	Intel Core i5 / AMD Ryzen 5
Operační paměť	4 GB RAM	8 GB RAM
Grafická karta	Integrovaná grafika (DirectX 11)	Dedikovaná GPU
Místo na disku	500 MB	1 GB
Internetové připojení	Vyžadováno (OpenAI API)	Stabilní připojení

5 Metodika a řešení

5.1 Použité nástroje

Projekt byl vyvíjen v herním enginu Unity 2022.3.62f1 LTS s C# jako programovacím jazykem. Pro dynamické sestavování 3D místností z ASCII mřížky byl použit Unity ProBuilder. Komunikace s OpenAI API zajišťovala Unity knihovna com.openai.unity. Kód byl psán ve Visual Studio Code a verzován přes Git a GitHub.

Tabulka 9 Použité nástroje

Nástroj/technologie	Verze	Účel
Unity	2022.3.62f1 LTS	Herní engine, fyzika, rendering
C#	9.0	Programovací jazyk
Unity ProBuilder	součást Unity	Dynamická tvorba 3D geometrie místností
TextMesh Pro	součást Unity	Renderování textu v UI
OpenAI API (GPT-4o-mini)	REST API	Generování ASCII map
com.openai.unity	–	Unity wrapper pro OpenAI API
Visual Studio Code	aktuální	Editor kódu
Git/GitHub	2.x	Správa verzí, záloha projektu

5.2 Průběh vývoje

5.2.1 Rešerše a návrh

Na začátku jsem prostudovala dostupnou literaturu k procedurálnímu generování obsahu a AI přístupům ke generování map. Na základě toho jsem se rozhodla pro generování pomocí jazykového modelu GPT-4o-mini, tedy bez nutnosti trénovat vlastní model nebo

implementovat složité grafové algoritmy. Model dostane strukturovaný prompt a vrátí ASCII mapu, která se pak zpracuje v Unity. Souběžně vznikl Game Design Document.

5.2.2 Prototypování

V prototypovací fázi jsem ověřila, zda je GPT-4o-mini schopný konzistentně generovat validní ASCII mapy v požadovaném formátu. Testovala jsem různé formulace systémového promptu. Ukázalo se, že model občas vrací mapy s nesprávnými rozměry nebo neplatnými znaky, to vedlo k návrhu sanitizačního a opravného systému. Paralelně jsem sestavila základní prototyp dynamické stavby místností přes ProBuilder.

5.2.3 Implementace

Implementace zahrnovala dva hlavní systémy a sadu herních komponent.

Systém generování map (AIMapGenerator)

Třída AIMapGenerator komunikuje s OpenAI API: odešle prompt, přijme odpověď a zpracuje ji třemi kroky:

1. Sanitize() – ořeže nebo doplní řádky a sloupce na přesně 12×7, nahradí neplatné znaky, ověří existenci právě jednoho P. Pokud P chybí, pokus se zahodí.
2. RepairConnectivity() – flood fill z pozice hráče; buňky nedosažitelné z P jsou převedeny na void.
3. FillMissing() – pokud AI nevygenerovala dostatečný počet spawnerů nebo předmětů, doplní je na náhodné volné dlaždice #.

Generátor provede až 15 pokusů. Pokud ani jeden nevede k validní mapě, načte se záložní mapa ze složky Resources.

Systém stavby místností (ProBuilderMapGenerator)

Třída ProBuilderMapGenerator přijme textovou mapu a pro každou buňku označenou jako místnost dynamicky sestaví 3D pokoj přes ProBuilder API. Každá místnost obsahuje

podlahu, strop, čtyři zdi (nebo dveřní otvory podle sousedů) a bodové světlo. Po sestavení jsou na příslušné pozice instancovány prefaby hráče, spawnerů a předmětů.

Herní systémy

Implementovány byly tyto komponenty: pohyb hráče (Player), zdraví (HealthController), stamina (StaminaController), střelba (Gun), spawner nepřátel (EnemySpawner), sbírání předmětů (PickupItem) a správa výhry/prohry (GameManager).

5.3 Testování

Pro ověření funkčnosti a hratelnosti prototypu bylo provedeno uživatelské testování formou online dotazníku. Testování proběhlo 18.–23. března 2026. Dotazník vyplnilo celkem 12 respondentů.

5.3.1 Profil respondentů

Věkové rozmezí respondentů bylo 14–30 let, přičemž většina (75 %) spadala do věkové skupiny 18–19 let, která odpovídá primární cílové skupině hry.

Tabulka 10 Věk respondentů

Věk	Počet
14	1
18	3
19	6
20	1
30	1

Co se týče herních návyků, 16,7 % respondentů hraje hry každý den, 41,7 % několikrát týdně a 41,7 % několikrát měsíčně. Nejčastější platformou byl PC (50 %), následovaný mobilem (25 %).

5.3.2 Herní zážitek

Celkový herní zážitek byl hodnocen průměrnou známkou 4,33/5. Žádný respondent nehodnotil hru pod 3/5.

Tabulka 11 Hodnocení herního zážitku

Hodnocení	Počet	%
3	1	8,2 %
4	5	41,7 %
5	6	50 %

Ovládání bylo považováno za pohodlné, 33,3 % respondentů označilo ovládání jako zcela pohodlné, 58,3 % jako většinou pohodlné, 8,3 % neutrálně. Nikdo nehodnotil ovládání negativně.

Obtížnost považovalo 41,7 % respondentů za přiměřenou a 58,3 % za příliš lehkou. Nikdo hru neoznačil za příliš těžkou, což naznačuje prostor pro zvýšení obtížnosti v dalších verzích.

5.3.3 AI generování map

Funkci Generovat mapu vyzkoušelo 11 z 12 respondentů (91,7 %). Zajímavost a různorodost AI generovaných map hodnotili respondenti průměrnou známkou 4/5. Shoda

vygenerované mapy se zadáním dosáhla průměru 4,6/5.

Tabulka 12 Hodnocení generace mapy

Hodnocení	Zajímavost map	Shoda se zadáním
2	1	0
3	2	0
4	4	4
5	4	7

5.3.4 Nalezené chyby

Ze čtyř respondentů, kteří popsali konkrétní chybu, byly identifikovány tyto problémy:

Tabulka 13 Nalezené chyby

Chyba	Počet výskytů
Problémy s průchodností dveřními otvory	3
Chyba kolize u spawner objektu (nešel zničit)	1
Vygenerovaná místnost bez propojení s ostatními	1

5.3.5 Návrhy na zlepšení

Respondenti by si nejčastěji přáli přizpůsobit tyto aspekty:

Tabulka 14 Návrhy na zlepšení

Možnost přizpůsobení	Počet	%
Velikost mapy	7	58,3 %
Více itemů	7	58,3 %
Typy nepřátel	6	50 %
Styl rozmístění	4	33,3 %
Více spawnů	3	25,0 %

Z otevřených odpovědí respondenti nejčastěji zmiňovali citlivost ovládání, kameru procházející zdí a celkový nedostatek herního obsahu. Jeden respondent ocenil projekt jako funkční „proof of concept" AI generování map a poznamenal, že systém by bylo možné snadno upravit pro různé herní styly.

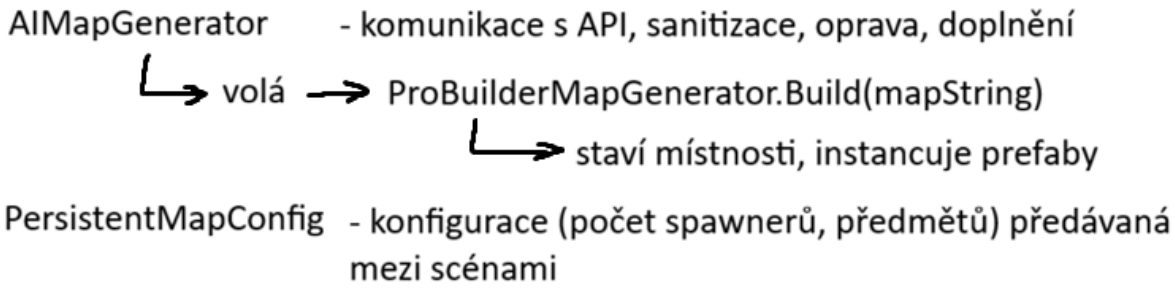
5.3.6 Doporučení hry

Ochotu doporučit hru přátelům hodnotilo 11 respondentů průměrnou známkou 4,09/5.

5.3.7 Závěr testování

Uživatelské testování potvrdilo, že prototyp je funkční a hratelný. Herní zážitek a AI generování map bylo hodnoceno nadprůměrně. Testování odhalilo tři opakující se problémy: průchodnost dveřními otvory, kolize u spawner objektů a izolované místnosti, tyto limity jsou blíže rozebrány v kapitole 7.3. Z výsledků dále vyplývá, že hráči by ocenili větší variabilitu obsahu a vyšší obtížnost.

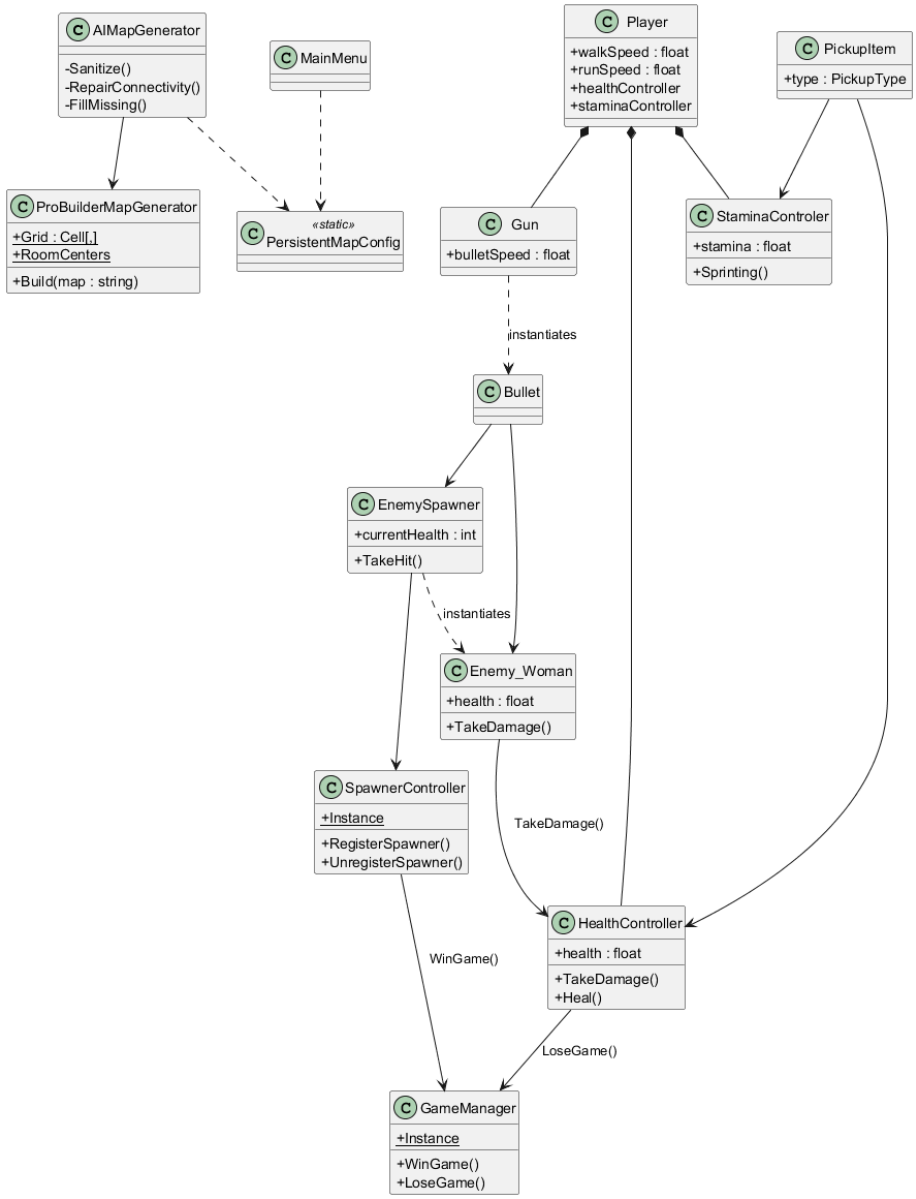
5.4 Architektura systému



Obrázek 2 Architektura systému

AIMapGenerator je MonoBehaviour připojený ke gameobjectu ve scéně. Metoda Start() spouští asynchronní metodu GenerateUntilValid() přes async/await, takže generování neblokuje hlavní vlákno Unity.

5.4.1 Diagram tříd



Obrázek 3 Class diagram

5.5 Problémy a jejich řešení

Největší výzvou bylo nekonzistentní chování modelu GPT-4o-mini, občas vrátil mapu s jiným počtem řádků nebo neplatnými znaky. Řešením byl vícestupňový sanitizační systém popsaný výše. Dalším problémem byla průchodnost NavMesh přes úzké dveřní otvory, vyřešila jsem ho rozšířením podlahy místností o jednu jednotku do všech stran, čímž se

překryvy zajistí spojitou navigační síť. Asynchronní volání API v Unity vyžadovalo použití async void metody kompatibilní s Unity hlavním vláknem.

6 Výsledky

6.1 Přehled dosažených výsledků

V rámci projektu *Peak* byl vytvořen funkční herní prototyp 3D first-person dungeon shooteru v enginu Unity 2022.3.62f1 LTS. Klíčovým výstupem je systém generování herních map pomocí velkého jazykového modelu GPT-4o-mini, který každé spuštění hry vytváří unikátní dungeon z propojených místností. Prototyp je plně hratelný: hráč může dungeon procházet, střílet na nepřátele, sbírat předměty a dosáhnout výhry zničením všech spawnerů.

6.2 Implementovaný systém generování map

6.2.1 Funkčnost generátoru

Systém generování map funguje na principu zasílání strukturovaného promptu modelu GPT-4o-mini přes OpenAI API. Model vrátí ASCII mapu (12×7 znaků), která je zpracována třemi kroky: sanitizace, oprava konektivity a doplnění chybějících prvků. Systém podporuje konfiguraci počtu spawnerů a předmětů. Při selhání API nebo nevalidních odpovědích (chybějící P) je k dispozici záložní mapa.

6.2.2 Kvalita generovaného obsahu

Vygenerované mapy se liší tvarem a rozmístěním prvků při každém spuštění. Díky opravě konektivity (flood fill) je zaručeno, že hráč vždy dosáhne na všechny spawnery a předměty. Záložní mechanismus zajišťuje, že hra nikdy nezůstane zaseknutá na obrazovce načítání.

6.2.3 Výkonnostní parametry

Generování mapy probíhá asynchronně a nezpůsobuje zamrznutí Unity. Celková doba generování závisí na latenci OpenAI API.

Tabulka 15 Výkonnosti parametry

Metrika	Hodnota	Poznámka
Počet pokusů API (typicky)	1–3	Závisí na kvalitě odpovědi modelu
Maximum pokusů před fallback	15	Nastaveno v kódu
Rozměry mapy	12 × 7 buněk	Pevně dáno
Fyzická velikost místnosti	15 × 15 × 11 j.	Jednotky Unity
Světla na místnost	1 bodové světlo	PointLight, range = 22,5 j.

6.3 Herní prototyp

6.3.1 Implementované herní mechaniky

Následující herní mechaniky jsou implementovány a funkční:

- **Pohyb hráče** – chůze (5 j/s), sprint (10 j/s) s využitím staminy, first-person kamera
- **Stamina systém** – drenáž při sprintu (25/s), regenerace v klidu (20/s), vizuální HUD
- **Zdraví** – HP 100, poškození od nepřátel, léčení předměty, vizuální HUD
- **Střelba** – vystřelení projektilu levým tlačítkem myši
- **Spawner nepřátel** – krystal s 5 HP, spawn nepřítele každých 25 sekund
- **Sbírání předmětů** – tři typy (Health / Energy / Both), klávesa E při kontaktu
- **Podmínky výhry/prohry** – Win/Lose panel, restart
- **AI generování mapy** – GPT-4o-mini generuje nový dungeon při každém spuštění
- **Procedurální stavba dungeonu** – ProBuilder sestavuje 3D místnosti z ASCII mřížky

6.3.2 Srovnání s cíli projektu

Tabulka 16 Srovnání s cíly projektu

Cíl projektu	Splněn

Cíl projektu	Splněn
Implementovat AI generování map pomocí LLM	Ano
Zajistit průchoznost každé vygenerované mapy (flood fill oprava)	Ano
Dosáhnout variety generovaného obsahu (unikátní dungeon / run)	Ano
Implementovat základní herní mechaniky (pohyb, boj, předměty)	Ano
Vytvořit funkční hratelný prototyp	Ano
Záložní mapa při selhání API	Ano

7 Diskuse

7.1 Zhodnocení zvolené metody

Pro generování herních map byl zvolen přístup přes velký jazykový model GPT-4o-mini. Tato volba se ukázala jako funkční, model je schopen generovat ASCII mapy v požadovaném formátu a výsledky jsou dostatečně variabilní pro potřeby prototypu.

Hlavní výhodou tohoto přístupu je jednoduchost implementace. Stačí zaslat textový prompt a zpracovat textovou odpověď, není potřeba trénovat vlastní model ani implementovat složité algoritmy. Prompt lze navíc upravit v přirozeném jazyce – například „zajisti, aby spawnery nebyly hned u vchodu", bez jakýchkoliv změn v kódu. To dává přístupu přes LLM výraznou flexibilitu.

Nevýhody jsou také zřejmé. Hra potřebuje aktivní připojení k internetu, protože bez přístupu k OpenAI API nelze novou mapu vygenerovat. Volání API trvá typicky 1-5 sekund, což je při startu hry znatelná pauza. Při časté hře mohou narůstat i náklady za API volání. Model navíc občas vrátí mapu s chybnou strukturou, takže sanitizační systém a opakované pokusy jsou nezbytné.

7.2 Srovnání s alternativami

Při výběru metody jsem zvažovala několik alternativ.

Wave Function Collapse by pravděpodobně generoval koherentnější mapy bez potřeby opravného systému, protože lokální pravidla sousednosti zaručují validitu výstupu. Nevýhodou je nutnost ručně definovat kompletní sadu dlaždic a pravidel, pro prototyp, kde se požadavky na design mohou měnit, by to bylo časově náročné.

Buněčné automaty jsou vhodné pro organické jeskynní systémy (jak je používá Minecraft), ale pro dungeon složený z diskrétních místností jsou méně přirozené a vyžadují dodatečný post-processing pro umístění herních prvků.

GAN nebo PCGML by vyžadovaly rozsáhlý dataset tréninových map a trénink modelu, což je pro maturitní projekt neúměrně složité.

Přístup přes LLM byl proto vyhodnocen jako nejpřiměřenější pro daný rozsah projektu. Nabízí dobrou rovnováhu mezi jednoduchostí implementace a flexibilitou výstupu.

7.3 Limity projektu

Z technického hlediska je mapa omezena na 12×7 místností – větší mapy by vyžadovaly strukturovanější výstup modelu nebo jiný post-processing. Nepřátelé nemají implementovaný pathfinding (NavMesh agenty), takže se pohybují pouze základním způsobem. Chybí také zvukový design.

Z designového hlediska mají všechny místnosti stejnou velikost a vizuální styl, takže varieta prostředí je omezená. Hra nemá systém progrese mezi runy (meta-progression), obtížnost se neadaptuje.

7.4 Možnosti dalšího rozvoje

Logickým dalším krokem by byla implementace lokálního záložního generátoru, jednoduchý BSP nebo celulární automat, který by fungoval offline bez potřeby API. Pro nepřátele by bylo zajímavé přidat NavMesh pathfinding, aby hráče aktivně pronásledovali. Roguelike meta-progression (odemykání schopností nebo typů místností mezi runy) by výrazně prodloužil životnost hry. Z hlediska generování map by stálo za vyzkoušení podmíněné generování, předávání parametrů jako obtížnost nebo styl dungeonu přímo v promptu.

Závěr

Cílem projektu *Peak* bylo navrhnout a implementovat počítačovou hru v Unity, jejímž klíčovým prvkem je generování herních map pomocí umělé inteligence. Tento cíl se podařilo splnit.

V teoretické části byly analyzovány přístupy k automatizované tvorbě herního obsahu, od klasických algoritmů jako Perlinův šum, buněčné automaty a BSP stromy až po moderní metody využívající strojové učení (GAN, PCGML, Wave Function Collapse). Rozbor existujících her ukázal, jak různě lze k procedurálnímu generování přistoupit a jaké kompromisy přináší každá varianta.

V praktické části vznikl funkční hratelný prototyp 3D first-person dungeon shooteru. Systém AI generování map funguje spolehlivě: model GPT-4o-mini dostane strukturovaný prompt a vrátí ASCII mapu, která je trojstupňovým post-processingem (sanitizace, flood fill, doplnění prvků) upravena do použitelné podoby. Každý run tak přináší unikátní dungeon, a přitom je zaručeno, že mapa bude vždy průchozí a bude obsahovat všechny potřebné herní prvky.

Projekt ukázal, že velký jazykový model lze prakticky využít jako generátor strukturovaného herního obsahu, a to bez nutnosti trénovat specializovaný model nebo implementovat komplexní grafové algoritmy. Zároveň potvrdil, že post-processing je při použití LLM výstupu v herním enginu nezbytný. Přístup má jasná omezení (závislost na API, latence, náklady), ale pro prototyp nabídl dobrou rovnováhu mezi jednoduchostí implementace a flexibilitou výsledku.

Reference

[1] SHAKER, N., TOGELIUS, J., NELSON, M. J. *Procedural Content Generation in Games* [online]. Cham: Springer, 2016. ISBN 978-3-319-42716-4. [Cit. 2025-11-07]. Dostupné z URL: http://pcgbook.com

[2] PERLIN, K. An image synthesizer. *ACM SIGGRAPH Computer Graphics*. 1985, roč. 19, č. 3, s. 287–296. ISSN 0097-8930. DOI: 10.1145/325165.325247.

[3] GUMIN, M. *Wave Function Collapse* [online]. GitHub, 2016. [Cit. 2025-11-07]. Dostupné z URL: https://github.com/mxgmn/WaveFunctionCollapse

[4] TOGELIUS, J., YANNAKAKIS, G. N., STANLEY, K. O., BROWNE, C. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*. 2011, roč. 3, č. 3, s. 172–186. ISSN 1943-068X. DOI: 10.1109/TCIAIG.2011.2148116.

[5] SUMMERVILLE, A., SNODGRASS, S., GU, M. a kol. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games*. 2018, roč. 10, č. 3, s. 257–270. ISSN 2475-1502. DOI: 10.1109/TG.2018.2846639.

[6] VOLZ, V., SCHRUM, J., LIU, J., LUCAS, S. M., SMITH, A., RISI, S. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018)*. New York: ACM, 2018, s. 221–228. ISBN 978-1-4503-5618-3. DOI: 10.1145/3205455.3205517.

[7] JOHNSON, L., YANNAKAKIS, G. N., TOGELIUS, J. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. New York: ACM, 2010. DOI: 10.1145/1814256.1814266.

[8] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M. a kol. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. Red Hook: Curran Associates, 2014, s. 2672–2680.

[I1] Unity Technologies. *Unity Real-Time Development Platform* [online]. [Cit. 2025-11-07]. Dostupné z URL: https://unity.com

[I2] Microsoft. *C# Language Reference* [online]. [Cit. 2025-11-07]. Dostupné z URL: https://docs.microsoft.com/cs-cz/dotnet/csharp/

[I5] Unity Technologies. *ProBuilder* [online]. [Cit. 2026-03-17]. Dostupné z URL: https://unity.com/features/probuilder

[I6] OpenAI. *GPT-4o mini – Advancing cost-efficient intelligence* [online]. 2024. [Cit. 2026-03-17]. Dostupné z URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[I7] OpenAI. *OpenAI API Reference* [online]. [Cit. 2026-03-17]. Dostupné z URL: https://platform.openai.com/docs/

[I8] YU, Derek (Mossmouth). *Spelunky* [videohry]. USA: Mossmouth, 2012.

[I9] Motion Twin. *Dead Cells* [videohry]. Francie: Motion Twin, 2018.

[I10] Dodge Roll. *Enter the Gungeon* [videohry]. USA: Devolver Digital, 2016.

Seznam obrázků

Seznam tabulek