

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ

Katedra softwarového inženýrství

Studijní program: Aplikace informatiky v přírodních vědách

Specializace: —



Vývoj strategické hry na náhodně generované mapě

VÝZKUMNÝ ÚKOL

Vypracoval: Bc. Štěpán Bezděk

Vedoucí práce: RNDr. Zuzana Petříčková, Ph.D.

Rok: 2025

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Akademický rok 2024/2025

ZADÁNÍ VÝZKUMNÉHO ÚKOLU

Student: Bc. Štěpán Bezděk
Studijní program: Aplikace informatiky v přírodních vědách
Název práce: Vývoj strategické hry na náhodně generované mapě

Pokyny pro vypracování:

1. Nastudujte problematiku procedurálního generování obsahu, především se zaměřením na využití při tvorbě herních světů v počítačových hrách.
2. Navrhněte strategickou hru, která bude využívat náhodně generované herní pole.
3. Implementujte generování herního pole pomocí procedurálních technik.
4. Implementujte základní herní mechaniky navržené hry (např. jednotky, ekonomika, boj).
5. Proveďte simulaci navržených herních principů a ověřte hratelnost navržené hry.

Doporučená literatura:

- [1] Salen, K., Zimmerman, E. Rules of Play: Game Design Fundamentals. MIT Press.
- [2] Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press.
- [3] Scheibenpflug, A. Evolutionary Procedural 2D Map Generation using Novelty Search.
- [4] Shaker, N., Togelius, J., Nelson, M. J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer.
- [5] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., Worley, S. Texturing & Modeling: A Procedural Approach. Morgan Kaufmann.

Jméno a pracoviště vedoucího práce:

RNDr. Zuzana Petříčková, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta práce:

Ing. Vladimír Jarý, Ph.D.


Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Datum zadání výzkumného úkolu: 21. 10. 2024

Termín odevzdání výzkumného úkolu: 25. 8. 2025

V Praze dne 21. 10. 2024


.....
vedoucí práce


.....
garant programu


.....
vedoucí katedry

PROHLÁŠENÍ

Já, níže podepsaný(á)

Příjmení, jméno studenta:Bezděk Štěpán.....

Osobní číslo:509692.....

Název programu:Aplikace informatiky v přírodních vědách.....

prohlašuji, že jsem bakalářskou/diplomovou práci s názvem

Vývoj strategické hry na náhodně generované mapě

.....

vypracoval(a) samostatně a uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací a Rámcovými pravidly používání umělé inteligence na ČVUT pro studijní a pedagogické účely v Bc a NM studiu.

☒ Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce použil/a nástroje umělé inteligence. Vygenerovaný obsah jsem ověřil/a. Stvrzuji, že jsem si vědom/a, že za obsah závěrečné práce plně zodpovídám.

☐ Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce nepoužil/a žádný nástroj umělé inteligence. Jsem si vědom/a důsledků, kdy bude zjevné nepřiznané použití těchto nástrojů při tvorbě jakékoli části mé závěrečné práce.

V dne

.....

Podpis

Prohlášení

Prohlašuji, že jsem svůj výzkumný úkol vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne

.....

Bc. Štěpán Bezděk

Poděkování

Rád bych poděkoval své vedoucí, RNDr. Zuzaně Petříčkové, Ph.D., za její odborné vedení a trpělivost při konzultacích. Zároveň bych chtěl vyjádřit poděkování konzultantovi, Ing. Vladimíru Jarému, Ph.D., za jeho cenné připomínky a odborné rady.

Bc. Štěpán Bezděk

Název práce:

Vývoj strategické hry na náhodně generované mapě

Autor: Bc. Štěpán Bezděk

Studijní program: Aplikace informatiky v přírodních vědách

Specializace: –

Druh práce: Výzkumný úkol

Vedoucí práce: RNDr. Zuzana Petříčková, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze

Konzultant: Ing. Vladimír Jarý, Ph.D.

Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Abstrakt: Cílem tohoto výzkumného úkolu bylo navrhnout tahovou strategickou hru, která efektivně využívá procedurálně generované herní mapy. Na základě detailního studia dostupných technik procedurálního generování obsahu (PCG) byla pro tvorbu herních světů zvolena a implementována metoda založená na výškových mapách a gradientním šumu. Herní návrh specifikuje herní svět, základní mechaniky, jako jsou jednotky, ekonomika a boj, a definuje pravidla hry. Klíčovou součástí práce byla implementace herní logiky a základní umělé inteligence, která umožnila provedení simulací. Pomocí metody Monte Carlo byla ověřena vyváženost herních prvků a optimalizovány parametry generování map s cílem zajistit jejich průchodnost a strategickou zajímavost.

Klíčová slova: procedurální generování obsahu (PCG), náhodné generování map, návrh hry, herní mechaniky, Monte Carlo simulace

Obsah

| | |
|--|-----------|
| Úvod | 9 |
| 1 Teorie PCG | 10 |
| 1.1 Procedurální generování obsahu (PCG) | 10 |
| 1.1.1 Výhody procedurálního generování | 10 |
| 1.1.2 Nevýhody a výzvy procedurálního generování | 11 |
| 1.1.3 Procedurální generování ve hrách | 11 |
| 1.2 Metody procedurálního generování map | 12 |
| 1.2.1 Space Partitioning | 12 |
| 1.2.2 Agent-Based generování | 13 |
| 1.2.3 Grammar Algorithms | 14 |
| 1.2.4 Výškové mapy | 15 |
| 1.2.5 Fraktálové algoritmy | 16 |
| 1.2.6 Wave Function Collapse (WFC) | 17 |
| 1.2.7 GAN (Generative Adversarial Networks) | 18 |
| 1.3 Porovnání metod | 19 |
| 2 Implementace mapy | 21 |
| 2.1 Generování herního pole | 21 |
| 2.2 Implementované metody | 21 |
| 2.2.1 Náhodná výšková mapa | 21 |
| 2.2.2 Interpolovaná výšková mapa | 22 |
| 2.2.3 Gradientní šum | 24 |
| 2.2.4 Přiřazení typů terénu | 25 |
| 2.2.5 Vizualizace mapy | 25 |
| 3 Návrh | 27 |
| 3.1 Návrh strategické hry | 27 |
| 3.2 Prostor | 28 |
| 3.2.1 Návrh herního prostoru | 29 |
| 3.3 Objekty, atributy a stavy | 29 |
| 3.3.1 Návrh herních objektů | 31 |
| 3.4 Akce hráče | 34 |
| 3.4.1 Návrh herních akcí | 35 |
| 3.5 Pravidla hry | 36 |
| 3.5.1 Návrh pravidel | 37 |
| 3.6 Dovednost a náhoda | 38 |

| | | |
|----------|--|-----------|
| 3.6.1 | Dovednost a náhodnost v navržené hře | 39 |
| 4 | Implementace základních mechanik | 40 |
| 4.1 | Implementace prototypu herních mechanik | 40 |
| 4.1.1 | Architektura a účel prototypu | 40 |
| 4.1.2 | Třída <code>SpravceHry</code> | 41 |
| 4.1.3 | Třída <code>Hrac</code> | 42 |
| 4.1.4 | Třída <code>Jednotka</code> | 43 |
| 4.1.5 | Třída <code>Budova</code> | 44 |
| 5 | Simulace | 45 |
| 5.1 | Úvod do simulací | 45 |
| 5.1.1 | Cíle simulací | 45 |
| 5.2 | Teorie simulace | 46 |
| 5.2.1 | Monte Carlo metoda v kontextu hry | 46 |
| 5.2.2 | Reprodukovatelnost a správa náhodnosti | 46 |
| 5.3 | Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích | 46 |
| 5.3.1 | Popis testovaných scénářů | 46 |
| 5.3.2 | Metodika | 47 |
| 5.3.3 | Duely | 47 |
| 5.4 | Fáze 2: Testování náhodného herního pole | 60 |
| 5.4.1 | Metodika | 60 |
| 5.4.2 | Testování měřítka šumu | 61 |
| 5.4.3 | Testování různých prahových hodnot | 65 |
| | Závěr | 67 |
| | Literatura | 67 |
| | Přílohy | 71 |
| A | Přiložené soubory | 71 |

Úvod

Tradiční (ruční) metody tvorby herního obsahu jsou z časového i finančního hlediska velmi náročné. To nemusí představovat problém u lineárních her nebo her s omezeným a neměnným světem. U her s rozsáhlou či nekonečnou herní mapou však tyto nároky rostou jak při samotném návrhu a modelování, tak i při ukládání a distribuci obsahu. Z těchto důvodů se stále častěji uplatňuje technika procedurálního generování obsahu (PCG), která umožňuje distribuovat pouze algoritmus pro generování prostředí. Tím dochází nejen ke snížení nároků na úložiště, ale i ke zvýšení znovuhratelnosti, protože pro každý nový průchod hrou lze vygenerovat unikátní svět.

Tento výzkumný úkol si klade za cíl prozkoumat dostupné metody procedurálního generování obsahu (PCG), vybrat z nich vhodnou techniku pro generování náhodných herních map a následně ji aplikovat při návrhu a implementaci strategické tahové hry. Volba žánru je motivována především tím, že strategické hry silně těží ze znovuhratelnosti, kterou PCG poskytuje. Každá nová, unikátně vygenerovaná mapa představuje pro hráče novou taktickou výzvu, což prodlužuje herní životnost. Výsledná hra zároveň poslouží jako základ pro diplomovou práci, která se bude dále zabývat využitím metod strojového učení pro herní umělou inteligenci.

První kapitola se věnuje teoretickému úvodu do problematiky procedurálního generování obsahu. Jsou zde rozebrány výhody, nevýhody a různé způsoby využití této techniky, především v kontextu generování náhodných map. Následuje srovnání několika vybraných metod a výběr nejvhodnější z nich pro implementaci.

Ve druhé kapitole je podrobně popsána implementace zvolené PCG techniky, včetně popisu použitých algoritmů a prezentace výsledných generovaných map.

Ve třetí kapitole jsou představeny základy teorie návrhu videoher. Na jejich základě je připraven návrh strategické tahové hry a specifikována její pravidla.

Ve čtvrté kapitole jsou implementovány základní herní mechaniky strategické hry. Zároveň je připravena jednoduchá herní umělá inteligence, která v závěrečné kapitole umožní provedení simulací a optimalizaci navržených atributů herních objektů.

V poslední páté kapitole je implementovaný prototyp využit k simulování her. Pomocí metody Monte Carlo jsou optimalizovány atributy herních objektů tak, aby byla hra vyvážená. Současně jsou touto metodou optimalizovány parametry pro generování náhodných map s cílem zajistit jejich častou průchodnost a strategickou zajímavost.

Kapitola 1

Teorie PCG

1.1 Procedurální generování obsahu (PCG)

Procedurální generování je metoda algoritmického vytváření dat. Procedurálně vytvořená data se od ručně vytvořených liší tím, že jsou generována kombinací malého množství ručně vytvořených vstupních dat, na jejichž základě počítač podle daného algoritmu vytváří komplexnější výstup. Tento přístup je vhodný zejména pro aplikace, kde je vyžadováno velké množství různorodého obsahu nebo kde je ruční tvorba neefektivní [1].

Hlavní výhodou procedurálního generování je úspora času, lidských zdrojů a tím i finančních nákladů. Navíc umožňuje vytvářet obsah, který může být přizpůsoben konkrétním požadavkům nebo preferencím uživatele. Generovaný obsah je dynamický, což otevírá možnosti pro personalizaci zážitků, například při tvorbě jedinečných map ve hrách nebo simulace náhodných prostředí [2].

1.1.1 Výhody procedurálního generování

Procedurální generování nabízí řadu výhod, které jej činí atraktivním pro různé aplikace, zejména ve vývoji her a designu [1]:

- **Redukce nároků na úložiště:** Namísto ukládání velkého objemu dat je možné ukládat pouze algoritmus a jeho vstupní parametry.
- **Variabilita:** Algoritmus je schopen generovat nekonečné množství unikátních výsledků.
- **Dynamika:** Obsah se může přizpůsobovat v reálném čase na základě uživatelských interakcí nebo jiných faktorů.
- **Úspora lidských zdrojů:** Omezuje potřebu ruční práce při tvorbě rozsáhlých prostředí.

1.1.2 Nevýhody a výzvy procedurálního generování

Ačkoliv má procedurální generování mnoho výhod, existují také určité nevýhody a výzvy [1]:

- **Kontrola kvality:** Výstup algoritmu nemusí vždy splňovat očekávání nebo být konzistentní s požadovanými normami.
- **Složitost implementace:** Návrh a ladění algoritmů mohou být časově náročné.
- **Předvídatelnost:** Přílišná náhodnost může vést k obsahu, který nedává smysl nebo není použitelný.
- **Závislost na vstupních datech:** Kvalita výstupu je silně ovlivněna kvalitou a rozmanitostí vstupních dat.

1.1.3 Procedurální generování ve hrách

Procedurální generování obsahu hraje klíčovou roli v moderním herním vývoji, zejména díky schopnosti efektivně vytvářet rozsáhlé a rozmanité herní světy. Tento přístup je oblíbený především u her s otevřeným světem, jako jsou *Minecraft* [3], *No Man's Sky* [4] nebo žánru *roguelike* her, kde je kladen důraz na variabilitu a dynamické prostředí [1].

Procedurální generování umožňuje vytvářet herní světy, které jsou pro hráče vždy jedinečné, což zvyšuje atraktivitu a prodlužuje životnost hry. Například v *Minecraftu* se každý nový svět skládá z unikátní kombinace biomů, terénních útvarů a zdrojů, což motivuje hráče k dalšímu průzkumu [1]. Podobně ve hře *No Man's Sky* je generován celý vesmír s miliardami planet, z nichž každá má unikátní prostředí, faunu a flóru [1].

Hlavní výhodou využití PCG v herním designu je schopnost generovat obsah přizpůsobený hráčově zkušenosti. Například v akčních hrách mohou algoritmy procedurálního generování upravovat obtížnost úrovně v reálném čase na základě výkonu hráče. Tento adaptivní přístup může zajistit, že hra zůstane náročná, avšak nefrustrující, což zvyšuje angažovanost hráče [2].

Dalším využitím PCG je tvorba náhodných misí nebo úkolů. Ty mohou být generovány tak, aby obsahovaly různé cíle, nepřátele nebo interaktivní objekty. Tento přístup umožňuje vytvářet dynamický herní zážitek, kdy se žádná mise ani průběh hry neopakují. Tento princip se často využívá například ve hrách typu *Diablo* [5] nebo *Borderlands* [6], kde jsou zbraně a další vybavení generovány procedurálně.

Přes četné výhody s sebou procedurální generování přináší i určité výzvy. Například ve hře *No Man's Sky* čelili vývojáři kritice za přílišnou repetitivnost obsahu, přestože byl generován procedurálně. Dalším problémem je zajištění smysluplnosti a konzistence herního světa, což vyžaduje pečlivě navržené algoritmy a pravidla [1].

Procedurální generování textur a zvuků

Procedurální generování nachází své uplatnění také při tvorbě textur a zvuků, což významně přispívá k redukci nároků na úložný prostor. Textury povrchů, jako jsou dřevo, kámen nebo kov, mohou být generovány pomocí algoritmů, jako je **Perlin Noise** nebo **Simplex Noise** [7]. Také je tímto přístupem možné generovat vegetaci tak, aby každý strom a keř vypadal jedinečně, například pomocí **Grammar Algorithms** [8]. Tento přístup nejen šetří místo na disku, ale také zvyšuje variabilitu, protože každá textura může být jedinečná.

Podobně lze procedurálně generovat zvuky, jako je ambientní hudba nebo efekty prostředí (například šum deště, větru či zvuky zvířat). Použitím technik syntézy zvuku místo ukládání statických zvukových souborů je možné dynamicky přizpůsobit zvukové efekty aktuálním podmínkám ve hře, čímž se nejen šetří úložiště, ale také zlepšuje ponoření hráče do herního prostředí [1].

1.2 Metody procedurálního generování map

V rámci této práce je procedurální generování využito specificky pro vytvoření nové herní plochy pro každou hru. V této části jsou rozebrány některé možné techniky pro generování herních map. Hlavními požadavky na algoritmy jsou logické rozložení mapy, hratelnost a rychlost generování. Algoritmy vycházely mimo jiné z publikace [1].

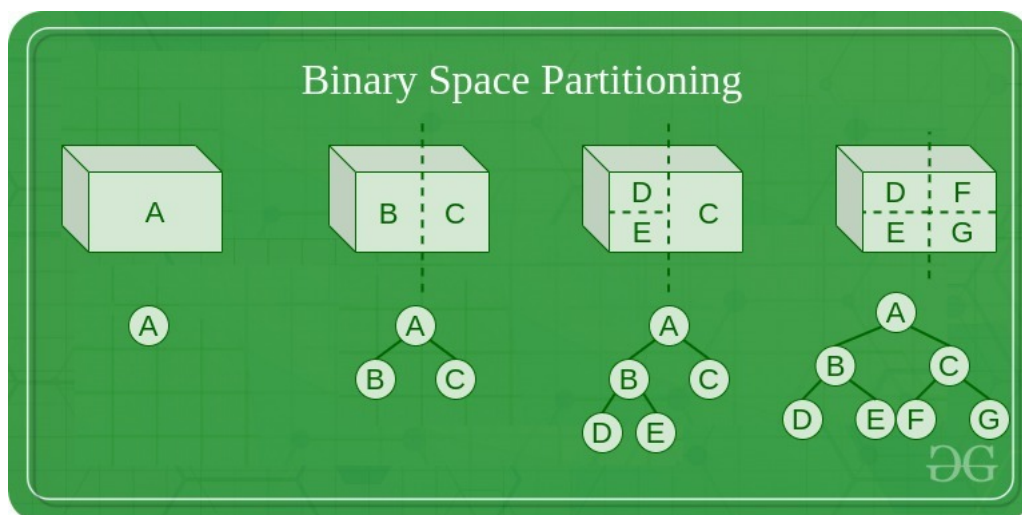
1.2.1 Space Partitioning

Dělení prostoru (Space Partitioning) [9] je metoda, která rekurzivně rozděluje herní mapu na menší, disjunktní oblasti, přičemž každý bod mapy náleží právě jedné z těchto oblastí. Výsledkem je hierarchická stromová struktura, která usnadňuje rychlé vyhledávání informací a manipulaci s daty.

Základním principem této metody je postupné dělení prostoru, dokud nejsou splněny předem definované podmínky, například dosažení minimální velikosti buňky nebo místnosti. Metoda nachází uplatnění nejen při generování herních map, ale i v grafických aplikacích, jako je ray-tracing, kde se využívá pro efektivní organizaci a vykreslování objektů ve scéně [10].

Nejběžnější metodou pro rozdělování prostoru je *binární dělení prostoru (BSP)* [11]. Jde o proces, který opakovaně rozděluje prostor na dvě části, dokud nejsou splněny určité požadavky. Ukázka *BSP* je zobrazena na obrázku 1.1. Další často využívanou variantou je *quadtree*, kde je prostor rozdělen na čtyři symetrické části. Hlavním rozdílem mezi *quadtree* a *BSP* je, že v *BSP* nemusí být na rozdíl od *quadtree* všechny buňky symetrické a buňky v *BSP* navíc nemusí mít stejnou orientaci [11].

Metoda je běžně využívána při vykreslování grafických scén, kde se dělicí roviny vybírají na základě polygonů, aby optimalizovaly výpočty. Pro renderování scény



Obrázek 1.1: Ukázka *BSP*. Převzato z [11].

je například důležité zajistit, že každý uzel stromu obsahuje polygony, které lze vykreslit efektivně [11].

1.2.2 Agent-Based generování

Nejjednodušší metoda generování založená na agentech (*agent-based method*) [12] přistupuje ke generování úrovní tak, že vygeneruje agenta, který „vykopává” chodby a vytváří oblasti v souvislé sekvenci. Na rozdíl od přístupu dělení prostoru se jedná spíše o mikro přístup ke generování, který vede k organicky vypadajícím výsledkům, které však mohou být chaotické a může docházet k překrývání místností.

Těmto problémům lze do jisté míry předejít optimálním nastavením pravidel chování agenta. Tato optimalizace je však časově náročná a vliv změny jednotlivých parametrů je bez testování obtížné odhadnout.

Pro jednoduché generování úrovní existují dva hlavní přístupy: „slepého” agenta („*blind-agent*”) a agenta s „přehledem” („*look-ahead*”) [10].

Slepý agent

Metoda se slepým agentem je stochastická. Proces generování začíná umístěním agenta na náhodný bod v mapě, odkud je náhodně vybrán směr (nahoru, dolů, vlevo nebo vpravo). Agent začne „kopat” tímto směrem, a každý vykopaný dlaždicový bod dungeonu je nahrazen „chodníkovou” dlaždicí. Po prvním „vykopání” je 5 % pravděpodobnost, že agent změní směr (vybere nový náhodný směr) a dalších 5 % pravděpodobnost, že agent umístí místnost náhodné velikosti. Každým dalším bodem ve směru, který se shoduje s předchozím, se pravděpodobnost na změnu směru zvyšuje o 5 %. Každým dalším dlaždicovým bodem bez přidání místnosti se pravděpodobnost na přidání místnosti zvyšuje o 5 %. Když agent změní směr,

pravděpodobnost na změnu směru se sníží na 0 %. Když agent přidá místnost, pravděpodobnost na přidání místnosti zůstává na 0 % [10].

Agent s přehledem

Metoda agenta s přehledem řeší nedostatky slepého agenta, jako je například generování překrývajících se místností a slepých chodeb. Dosahuje toho tím, že agent získává informace o aktuálním stavu generované mapy. Agent tak může simulovat důsledky přidání nové místnosti a ověřovat, zda dojde k propojení s již existujícími oblastmi. Zároveň agent díky přehledu může zvolit směr tak, aby se dostal do oblasti, kde nehrozí překrytí, a minimalizuje tak riziko vzniku shluků místností [10].

Vícefázové generování terénu

V kontrastu s předchozími přístupy existuje další metoda, která využívá agenty k vytváření složitějších terénů, a to prostřednictvím vícefázového procesu. Tento přístup zahrnuje tři hlavní fáze generování terénu:

1. Fáze pobřeží: V této fázi velké množství agentů pracuje na vytvoření obrysu pevninské masy, která může být obklopena vodou.
2. Fáze terénu: V této fázi agenti definují vlastnosti mapy, jako je tvarování hor, nížin a pláží.
3. Fáze eroze: V této fázi agenti vytvářejí řeky, které erodují terén a spojují horské oblasti s oceánem.

Každý agent v tomto systému je schopen vidět aktuální výšku jakéhokoli bodu na mapě a může tyto body modifikovat podle potřeby. Tímto způsobem agenti aktivně formují terén a přítomnost jiných agentů může způsobit změny v okolním prostředí. Pro kontrolu životnosti agenta je každý agent vybaven určitým počtem tokenů, které spotřebovává při vykonávání akcí. Tento mechanismus dává designérovi možnost ovlivnit, jak bude terén generován.

Designér může ovlivnit makrovlastnosti mapy specifikací počtu agentů a celkového počtu dostupných tokenů v každé fázi. To umožňuje generovat různé typy terénu, například pobřežní oblasti, hory nebo řeky.

Tento vícefázový přístup, který využívá různé typy agentů (pobřežní agenti, agenti hor, agenti řek, atd.), poskytuje designérovi značnou kontrolu nad výsledným terénem. V důsledku toho je možné vytvářet zajímavější a strukturovanější mapy ve srovnání s jednoduššími metodami založenými na jednom agentovi [12].

1.2.3 Grammar Algorithms

Gramatiky [8] představují formální systém pro popis struktury a dekompozici jejích podčástí. Například věta může obsahovat podmět a ten může obsahovat přídavná

jména. *Gramatiky* lze ale také aplikovat mimo mluvené jazyky. Příkladem může být binární strom, jehož struktura je znázorněna v ukázce 1.1 [13] [14].

```
1  BRANCH ->  
2      node BRANCH BRANCH |  
3      leaf
```

Ukázka 1.1: Příklad gramatiky: Strom

Algoritmy založené na gramatikách však ve své základní podobě nemohou generovat nový obsah. Pro účely generování lze jejich funkčnost rozšířit přidáním pravděpodobností. Algoritmus pak s určitou pravděpodobností generuje jednotlivé části struktury, například věty nebo jiné objekty. Na příkladu stromu 1.2 mohou být váhy nastaveny tak, že pravděpodobnost vygenerování listu je dvakrát vyšší než pravděpodobnost rozdělení uzlu [13].

```
1  BRANCH ->  
2      node BRANCH BRANCH [weight=1] |  
3      leaf [weight=2]
```

Ukázka 1.2: Příklad gramatiky: Strom s váhami

Gramatiky jsou silným nástrojem pro generování herního obsahu, protože poskytují strukturovaný způsob, jak popsat a vytvářet různé herní prvky, od prostorových uspořádání po specifické herní objekty. Tento přístup je analogický generativním gramatikám v přirozených jazycích. Tyto gramatiky využívají konečnou sadu rekurzivních pravidel k definování větších struktur z menších částí, což se jeví jako efektivní způsob generování komplexních herních prostorů.

Například lze použít grafovou gramatiku pro generování topologie úrovní, kde uzly reprezentují místnosti a hrany mezi nimi určují jejich propojení. Tento přístup je velmi flexibilní, jelikož umožňuje přizpůsobit generování úrovní specifickým parametřům, jako je obtížnost, velikost nebo hratelnost. Takový vysoký stupeň kontroly je v kontextu vývoje herních zážitků výhodný. Na druhou stranu však může být složité vytvořit univerzální gramatiku, která by pokryla veškeré herní scénáře [10].

1.2.4 Výškové mapy

Terén může být reprezentován jako dvourozměrná matice reálných čísel, kde šířka a výška matice odpovídají souřadnicím x a y . Hodnoty v jednotlivých buňkách poté představují výšku v daném bodě. Tato matice je označována jako *výšková mapa* (*heightmap*) [15] [16].

Náhodný terén

Nejjednodušší metodou pro vytvoření výškové mapy je použití generátoru náhodných čísel a následné vyplnění matice náhodnými hodnotami. Tato metoda vytvoří výškovou mapu, kterou je teoreticky možné vykreslit, ale výsledná mapa nevypadá jako mapa terénu, spíše jako náhodné výčnělky. Reálný terén se vyznačuje plynulou

změnou výšek. Výška v jednom bodě logicky souvisí s výškami v jeho okolí, náhodný generátor však generuje výšky nezávisle na okolních hodnotách, což vede k nepřirozenému výsledku [16] [1].

Interpolace terénu

Jednoduchým řešením tohoto problému je aplikace interpolace. Nejprve jsou náhodné hodnoty výšek generovány na hrubší mřížce, následně se výšky mezi těmito body dopočítají pomocí interpolace. Ačkoli tímto postupem nelze generovat některé přirozené útvary, jako jsou útesy, výsledný terén je hladší a realističtější [16].

Gradient-based náhodný terén (Šum)

Místo generování výškových hodnot a následné interpolace svahů můžeme generovat přímo svahy, tedy gradienty změny výšky terénu, a z těch následně odvozovat výšky [16].

*„Gradient noise je způsob generování terénu, kdy náhodná čísla interpretujeme jako náhodné gradienty, což znamená strmost a směr svahů. Tento přístup byl poprvé použit Kenem Perlinem pro film Tron z roku 1982, a proto se někdy nazývá **Perlin noise**.“* – přeloženo z [1].

Na hrubší mřížce jsou vygenerovány náhodné gradienty, reprezentované vektory dx a dy , které odpovídají sklonu v osách x a y . Gradienty mohou mít kladnou nebo zápornou hodnotu, což umožňuje modelovat stoupající i klesající svahy.

Výšky se získají inicializací každého bodu mřížky hodnotou 0. Pro určení výšky na místech mezi mřížkovými body se uvažují čtyři sousední body mřížky. Nejprve se provede výpočet pro hypotetickou situaci, kdy je zohledněn pouze gradient v levém horním rohu. Výška v aktuálním bodě by pak byla dána hodnotou gradientu vynásobenou vzdáleností, kterou jsme urazili podél svahu: strmostí na ose x (dx) násobenou vzdáleností od mřížky ve směru x , plus strmostí na ose y (dy) násobenou vzdáleností ve směru y . Tento výpočet je proveden pro každý ze čtyř sousedních bodů mřížky. Výsledkem jsou čtyři dílčí výšky, které odpovídají situaci, kdy by výška terénu byla závislá pouze na jednom z těchto gradientů. Tyto hodnoty jsou poté interpolovány [7].

Gradientní šum umožňuje vytvářet realistické terény s plynulými změnami výšek.

1.2.5 Fraktálové algoritmy

Fraktální algoritmy umožňují generovat realistický terén díky své schopnosti vytvářet detailní rozsáhlé struktury. Často používanou metodou pro generování fraktálního terénu je *Diamond-square algoritmus* (diamantovo-čtvercový algoritmus) [17]. Algoritmus je výpočetně nenáročný a snadno implementovatelný [18].

Algoritmus funguje následovně:

1. Inicializace: Nastaví se hodnoty čtyř rohů výškové mapy na náhodné hodnoty.
2. Diamantový krok: Je nalezen střed čtverce definovaného těmito čtyřmi rohy a jeho hodnota je určena průměrem hodnot rohů a přidáním náhodné hodnoty.
3. Čtvercový krok: Jsou nalezeny střední body stran čtverce a jejich hodnoty jsou určeny průměrem tří hodnot: dvou sousedních rohů a středu čtverce. Opět je přidána náhodná hodnota.
4. Rekurze: Po dokončení diamantového a čtvercového kroku je čtverec rozdělen na čtyři menší. Drsnost se sníží a celý proces se rekurzivně opakuje pro každý z menších čtverců. Tento proces pokračuje, dokud není dosaženo maximálního počtu iterací.

Velikost náhodných hodnot použitých v těchto krocích se označuje jako **drsnost** (*roughness*). Vyšší hodnoty vedou k drsnějšímu terénu, zatímco nižší hodnoty vytvářejí hladší terén.

Diamond-square algoritmus se využívá v mnoha aplikacích, jako jsou například hry (např. *Minecraft*) nebo simulace pro generování krajiny v leteckých simulátorech.

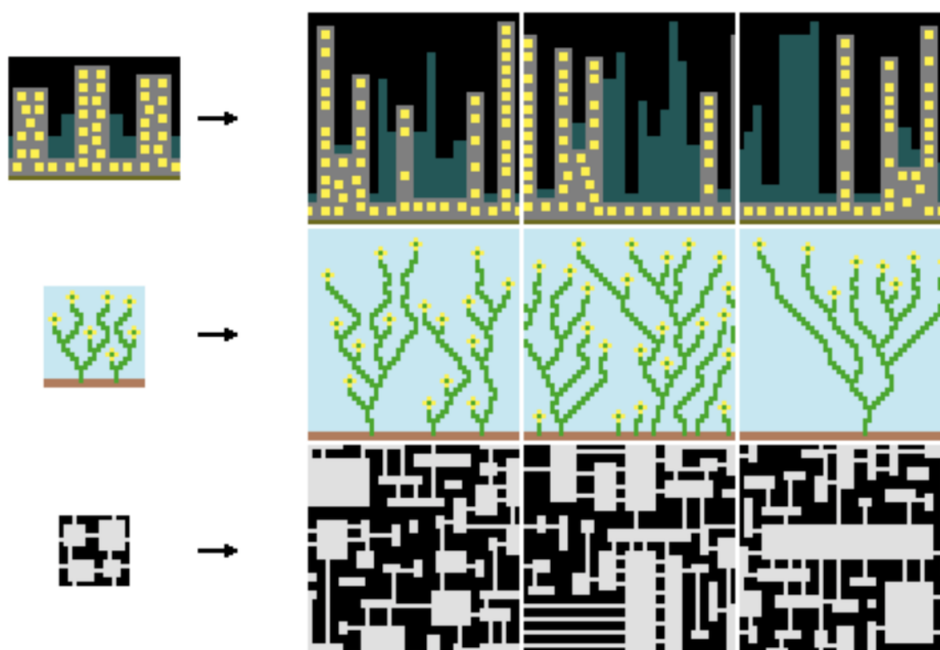
1.2.6 Wave Function Collapse (WFC)

Algoritmus *Wave Function Collapse* [19] je inspirován principy kvantové mechaniky. Využívá principu superpozice, kde každá buňka mapy obsahuje sadu možných hodnot, která je postupně „kolabována“ na jedinou, v souladu s definovanými pravidly.

WFC operuje s mřížkou buněk, přičemž každá z nich je na počátku v superpozici všech možných stavů. Algoritmus selektivně odstraňuje nekonzistentní možnosti na základě stavu sousedních buněk a předdefinovaných pravidel, což vede k postupnému „zúžení“ prostoru možných konfigurací, dokud není každá buňka v jednoznačném stavu.

Algoritmus funguje následovně:

1. Každá buňka mřížky začíná s plnou superpozicí všech možných hodnot.
2. Je vybrána buňka s nejnižší entropií (tj. s nejmenším počtem zbývajících možností).
3. Možnosti vybrané buňky se zúží na jednu jedinou hodnotu v souladu s pravidly. Tím dojde ke kolapsu jejího stavu.
4. Zúžení možností jedné buňky ovlivní okolní buňky, čímž se jejich možnosti také zúží.
5. Proces je rekurzivně opakován, dokud není celá mřížka vyřešena (tj. dokud každá buňka neobsahuje pouze jednu možnost).



Obrázek 1.2: Příklad vzorů a výsledků *WFC*. Převzato z [20]

WFC se využívá pro generování úrovní, textur a dalších prostorových uspořádání, protože umožňuje vytvářet složité a realistické struktury na základě jednoduchých vzorů. Příklad vzorů a jejich výsledků je vidět na obrázku 1.2 [20].

1.2.7 GAN (Generative Adversarial Networks)

Generativní adversariální sítě (GAN) [21] představují metodu generování obsahu, která je složena ze dvou komponent – **generátoru** a **diskriminátoru**, které „bojují“ v procesu učení. Generátor vytváří realistické výstupy napodobující reálná data. Diskriminátor se pak snaží rozlišit, zda jsou data originální nebo vygenerovaná.

Princip metody *GAN* spočívá v interakci mezi generátorem a diskriminátorem, která vede k postupnému zlepšování kvality generovaných dat. Tento proces je řízen pomocí zpětné propagace, která umožňuje optimalizovat váhy v obou sítích. Díky tomuto vzájemnému soupeření se data generovaná systémem *GAN* jeví jako realistická, a to i s menším množstvím trénovacích dat.

Mezi různé varianty *GAN* patří například *Deep Convolutional GAN (DCGAN)* [22], který využívá konvoluční vrstvy k lepšímu zachycení prostorových vzorců v datech. Tento přístup lze využít při generování výškových map, nicméně má některá omezení, jako jsou problémy s velikostí a škálovatelností výstupů. Dále existuje *Spatial GAN (SGAN)* [23] jedná se o vylepšenou variantu *DCGAN*, která eliminuje plně propojené vrstvy a místo nich používá konvoluční vrstvy s *krokem*, což umožňuje lépe zachovávat prostorové vztahy a zlepšuje efektivitu generování [24].

1.3 Porovnání metod

Kapitola 2 představila několik přístupů k procedurálnímu generování herních map. V následující sekci budou tyto metody porovnány a bude zvolen vhodný přístup pro implementaci v rámci tohoto výzkumného úkolu.

Hlavním cílem je generovat herní mapy pro strategické hry, které jsou tvořeny dlaždicemi nebo políčky. Výsledný terén by měl působit přirozeně. Z tohoto důvodu nejsou vhodné metody, které vytvářejí nesouvislé nebo lineárně rozdělené výsledky, jelikož se zaměřujeme na generování krajiny, nikoli městských oblastí.

Zároveň, jelikož pro každou hru bude generována nová unikátní mapa, by bylo vhodné, aby generování probíhalo v rozumném čase, tedy metody s nízkou výpočetní náročností budou preferované.

V tabulce 1.1 jsou popsány výhody a nevýhody jednotlivých metod generování.

Na základě porovnání metod bylo rozhodnuto využít přístup založený na *výškových mapách*. Hlavním důvodem je jeho schopnost generovat realisticky působící krajinné útvary při relativně nízké výpočetní náročnosti. Využití interpolace nebo gradientních metod umožní vytvářet plynulé přechody mezi různými typy terénu, což povede k realisticky vypadajícímu světu.

Mezi další zvažované metody patřily *Wave Function Collapse (WFC)* a *Generative Adversarial Networks (GAN)*. Tyto přístupy mají potenciál vést k ještě realističtějším a rozmanitějším výsledkům. *WFC* by zajistilo více vzorů v prostředí, zatímco *GAN* by umožnilo generovat zcela nové mapy na základě trénovacích dat. Nicméně, obě tyto metody jsou výpočetně náročné a vyžadují rozsáhlé trénovací sady dat, což je pro účely tohoto výzkumného projektu činí nevhodnými. Proto jsou zmíněné metody zvažovány jako možné rozšíření v rámci budoucí diplomové práce.

| Metoda | Výhody | Nevýhody |
|---------------------------------|---|--|
| Space Partitioning | Logická a hierarchická struktura Snadné dosažení konzistentních výsledků Vhodné pro vnitřní prostory | Výsledky mohou působit pravouhle Nevhodné pro organické mapy |
| Agent-Based generování | Přirozené, nelineární výsledky Velká variabilita výsledků Snadné přizpůsobení chování agentů | Náročné ladění parametrů Vysoká míra náhodnosti Nepředvídatelné výsledky |
| Grammar Algorithms | Vysoká míra kontroly Vhodné pro generování rozvržení úrovní | Náročné na definování pravidel Rychle narůstá složitost pravidel Nevhodné pro přirozené terény |
| Výškové mapy | Dobře reprezentují reálnou topografii Plynulé přechody mezi výškovými úrovněmi Snadná implementace Snadná kombinace s dalšími technikami | Náhodné generování vede k nereálným výsledkům Nevhodné pro generování budov a interiérů |
| Fraktálové algoritmy | Přirozeně vypadající krajina Efektivní pro nekonečné terény Relativně nenáročné na výkon | Omezená kontrola Výsledky mohou být homogenní |
| Wave Function Collapse | Konzistentní výsledky Vhodné pro mapy s pravidelnými vzory Snadné využití existujících vzorů | Výpočetně náročné Složitý vstupní dataset Může skončit ve stavu bez platného řešení |
| Generative Adversarial Networks | Realistické a variabilní výsledky Adaptivní na různé typy prostředí | Velké množství trénovacích dat Výpočetně náročné Obtížně laditelné |

Tabulka 1.1: Porovnání metod generování prostředí

Kapitola 2

Implementace mapy

2.1 Generování herního pole

Pro generování herního pole byla zvolena metoda výškových map. Výšková mapa je dvourozměrná matice reálných čísel, kde každá hodnota reprezentuje výšku v daném bodě. Tato mapa je následně transformována do konkrétních terénních typů, jako jsou voda, roviny, lesy a hory. V této kapitole jsou popsány implementace generování výškových map třemi způsoby:

- Náhodná výšková mapa – nejjednodušší metoda, která však vytváří nerealistický terén.
- Interpolace hrubé mřížky – metoda, která vyhlazuje terén pomocí interpolace mezi body náhodné mřížky.
- Gradientní šum – metoda, která generuje realisticky působící terén pomocí Perlinova šumu.

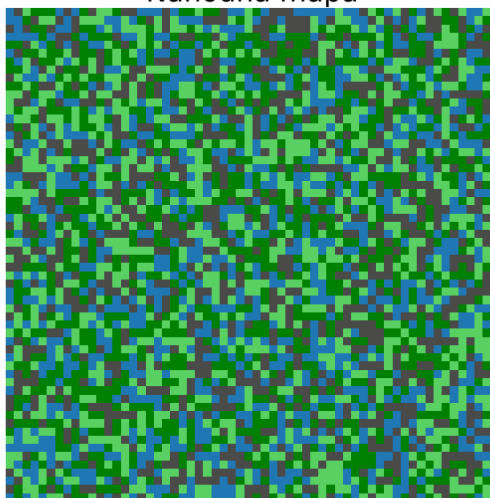
2.2 Implementované metody

2.2.1 Náhodná výšková mapa

Implementace této metody slouží primárně jako základní benchmark pro srovnání s ostatními přístupy, vzhledem k její jednoduchosti. Výsledný terén však nepůsobí realisticky a není vhodný pro herní účely, jelikož tato metoda negeneruje přirozené terénní struktury.

Celá implementace spočívá v generování náhodných hodnot, které jsou následně uloženy do dvourozměrného pole, které je vráceno jako výstup (ukázka kódu: 2.1). Výsledná mapa je vidět na obrázku 2.1.

Náhodná mapa



Obrázek 2.1: Ukázka mapy vygenerované metodou náhodné výškové mapy.

```
1 def nahodne_pole(rows, cols, min_value=0, max_value=1):  
2     return np.random.uniform(low=min_value, high=max_value, size=(  
    rows, cols))
```

Ukázka 2.1: Kód generující pole pro náhodnou mapu

2.2.2 Interpolovaná výšková mapa

Lepšího výsledku lze dosáhnout aplikací interpolace na náhodnou mřížku. Tato metoda spočívá v generování menší, řídké matice s náhodnými hodnotami. Její hodnoty jsou poté pravidelně rozmístěny do větší matice, přičemž zbývající prázdné hodnoty jsou dopočítány interpolací z původní menší matice. Pro implementaci interpolace byla použita knihovna *scipy* [25], jak je znázorněno v ukázce kódu 2.2. Výsledná funkce vrací matici s hodnotami v rozsahu od 0 do 1.



Obrázek 2.2: Ukázka mapy vygenerované metodou interpolace výškové mapy.

```

1 def interpolovane_pole(big_rows, big_cols, small_rows, small_cols,
2   min_value=0, max_value=1):
3     # Vytvoření menší mřížky
4     small_grid = nahodne_pole(small_rows, small_cols, min_value,
5   max_value)
6     # Indexy pro malou a velkou mřížku
7     small_x = np.linspace(0, big_cols - 1, small_cols)
8     small_y = np.linspace(0, big_rows - 1, small_rows)
9     big_x = np.arange(big_cols)
10    big_y = np.arange(big_rows)
11    # Vytvoření seznamu souřadnic
12    small_points = np.array([(x, y) for y in small_y for x in
13    small_x])
14    small_values = small_grid.flatten()
15    big_points = np.array([(x, y) for y in big_y for x in big_x])
16    # Doplnění hodnot seznamu interpolací
17    big_list = griddata(small_points, small_values, big_points,
18    method='cubic')
19    # Převedení na mřížku
20    big_grid = big_list.reshape(big_rows, big_cols)
21    return big_grid

```

Ukázka 2.2: Kód generující iterpolovanou výškovou mapu

Interpolace vytvoří plynulejší terén bez ostrých přechodů mezi výškami, ale výsledky mají stále tendenci být hranaté a ne zcela přirozeně vypadající, jak je vidět na obrázku 2.2.

2.2.3 Gradientní šum

Gradientní šum je komplexnější metoda, která generuje přirozenější struktury terénu než předchozí dvě metody. Za inspiraci je vzata metoda **Perlinův šum**, která generuje plynulé změny hodnot v prostoru, čímž vytváří realistické krajiny s kopci, údolími a horami.

Perlinův šum funguje na principu interpolace gradientních vektorů. Výsledkem je spojitá mapa hodnot, kde se výška plynule mění bez ostrých přechodů. Tímto způsobem vznikají přirozeně působící útvary. Tato metoda byla prvně implementována pomocí knihovny *noise* [26] jako vzorový výsledek a následně byla také napsána vlastní implementace.

Vlastní implementace

Napsaná funkce generuje gradientní šum v několika krocích:

1. Vytvoření mřížky gradientových vektorů – Každému bodu hrubé mřížky (dané parametrem *scale*) je přiřazen náhodný gradientový vektor, který určuje lokální směr změny šumu.
2. Lokální příspěvky (dot produkty) – Pro zvolený bod jemné mřížky se najdou čtyři nejbližší rohy příslušné buňky hrubé mřížky. Pro každý roh se spočítá vektor od rohu k bodu a jeho skalární součin s gradientem v tomto rohu. Tím získáme čtyři hodnoty, které popisují vliv rohů na daný bod.
3. Vyhlazené váhy (fade) – Z polohy bodu uvnitř buňky se určí vyhlazené váhy, které potlačují ostré přechody a zajišťují plynulost výsledného šumu.
4. Interpolace – Nejprve se hodnoty z rohů interpolují ve vodorovném směru a následně ve svislém směru. Tím vznikne výsledná hodnota šumu v daném bodě.
5. Normalizace – Nakonec se všechny hodnoty převedou do rozsahu $[0, 1]$, aby se s nimi dalo snadno pracovat.

Použitím různých měřítek (*scale*) je možné ovlivnit rozmanitost a velikost útvarů vygenerované krajiny.

Výsledná matice hodnot, která slouží jako výšková mapa pro tvorbu herního terénu, se vyznačuje plynulými přechody mezi jednotlivými výškami a vytváří tak realisticky působící krajinu, jak je vidět na obrázku 2.3.



Obrázek 2.3: Ukázka mapy vygenerované metodou gradientní výškové mapy.

2.2.4 Přiřazení typů terénu

Po vygenerování výškové mapy je nutné převést hodnoty na jednotlivé typy terénu, k tomu je použita funkce `cislo_na_policko` (Ukázka kódu: 2.3).

```

1 def cislo_na_policko(grid):
2     mapa = np.empty_like(grid, dtype='str')
3     for i in range(grid.shape[0]): # Počet řádků
4         for j in range(grid.shape[1]): # Počet sloupců
5             if grid[i][j] < 0.25:
6                 mapa[i][j] = "V" # Voda
7             elif grid[i][j] < 0.5:
8                 mapa[i][j] = "P" # Pláně
9             elif grid[i][j] < 0.75:
10                mapa[i][j] = "L" # Les
11            else:
12                mapa[i][j] = "H" # Hory
13    return mapa

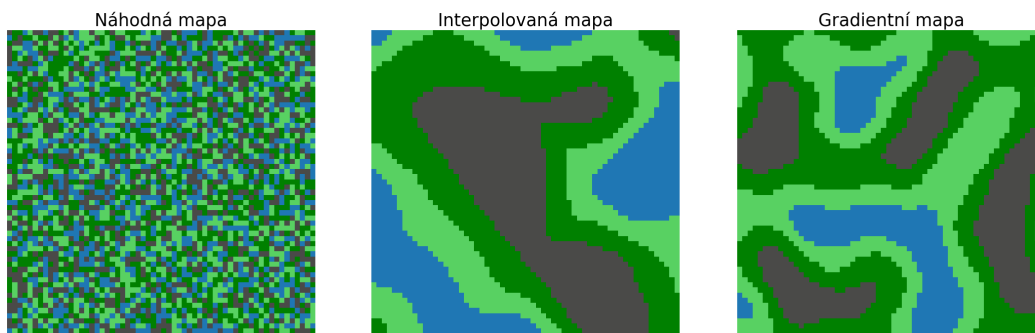
```

Ukázka 2.3: Převádějící výškovou mapu na konkrétní terény

Aktuální prahové hodnoty byly stanoveny experimentálně. Tyto hodnoty mohou být dále upraveny v závislosti na výsledcích testování v simulaci, s cílem zajistit vyvážený terén a optimální herní prostředí.

2.2.5 Vizualizace mapy

Matice terénů vycházející z funkce `cislo_na_policko` je vizualizována pomocí knihovny `matplotlib` [27] a funkce `zobraz_mapu` (Ukázka: 2.4).



Obrázek 2.4: Zobrazení vykreslených map generovaných všemi metodami.

```

1 def zobraz_mapu(mapa):
2     """
3     Barevně vykreslí terénní mapu.
4     """
5     # Definice barev pro jednotlivé typy terénu
6     barvy = {
7         "V": "#1f77b4", # Modrá - Voda
8         "P": "#58d162", # Světle zelená - Pláně
9         "L": "#0c3b10", # Zelená - Les
10        "H": "#4a4a48", # Šedá - Hory
11    }
12
13    # Převod mapy na numerickou matici s indexy
14    text_to_index = {"V": 0, "P": 1, "L": 2, "H": 3}
15    index_map = np.vectorize(text_to_index.get)(mapa)
16
17    # Vytvoření barevné mapy
18    cmap = ListedColormap(barvy.values())
19
20    # Vykreslení mřížky
21    plt.figure(figsize=(8, 8))
22    plt.imshow(index_map, cmap=cmap, interpolation='nearest')
23
24    plt.show()

```

Ukázka 2.4: Kód generující pole pro náhodnou mapu

Výstupem je čtyřbarevný graf složený ze čtvercových dlaždic. Na obrázku 2.4 jsou vidět zobrazené grafy generované různými metodami.

Kapitola 3

Návrh

3.1 Návrh strategické hry

Tato kapitola se zabývá postupem návrhu jednoduché strategické hry hrané na náhodně vygenerované mapě. Návrh strategické hry vyžaduje systematický přístup k definici jejích klíčových prvků. Pro zajištění hratelnosti, vyváženosti a zajímavosti je nutné systematicky analyzovat a promyslet návrh všech částí hry [28].

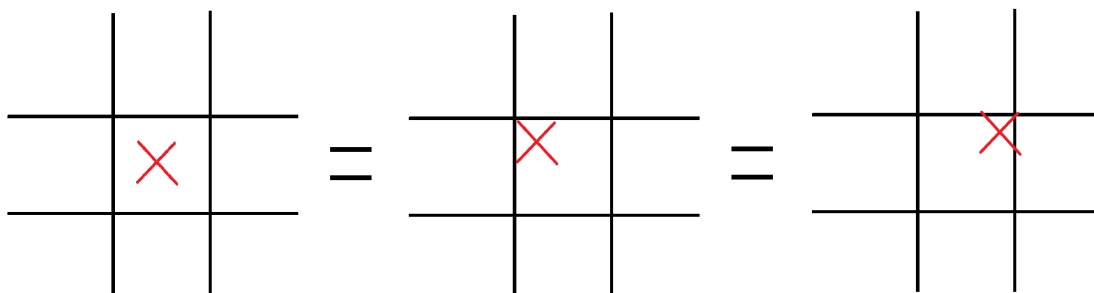
Návrh je založen na koncepci 2D tahové strategie se správou surovin. Hráči ovládají a vylepšují jednotky, spravují zdroje a budují infrastrukturu s cílem dosáhnout vítězství nad soupeřem. Herní svět je reprezentován mapou složenou ze čtvercových políček, přičemž typ terénu ovlivňuje pohyb jednotek, možnosti výstavby a dostupnost zdrojů.

Pro návrh hry byla využita metodologie z knihy *The Art of Game Design: A Book of Lenses* od Jesseho Schella [29], která strukturuje herní design do několika klíčových kategorií:

- Prostor – Jak je herní svět uspořádán a jak se v něm hráči pohybují.
- Objekty, atributy a stavy – Jaké herní entity existují, jaké mají vlastnosti a jaké stavy mohou při hře nastat.
- Akce hráče – Jaké interakce může hráč provádět a jak ovlivňují hru.
- Pravidla hry – Jaká jsou omezení a podmínky vítězství.
- Dovednost a náhoda – Jaký je poměr mezi strategickým rozhodováním a prvky náhody.

Tento rámec poskytuje ucelený pohled na návrh hry a pomáhá zajistit, aby všechny prvky dohromady tvořily soudržný a dobře vyvážený celek.

Ve zbytku kapitoly jsou podrobněji analyzovány tyto herní prvky a konkretizován celkový herní návrh.



Obrázek 3.1: Ekvivalence různých značení ve hře piškvorky.

3.2 Prostor

Každá hra se odehrává v herním prostoru, který vymezuje její lokace a určuje jejich vzájemné propojení. Z pohledu herních mechanik je prostor považován za matematickou konstrukci, tedy je potřeba odfiltrovat veškeré vizuální prvky a zaměřit se pouze na abstraktní uspořádání prostoru.

Kniha *The Art of Game Design: A Book of Lenses* [29] rozděluje herní prostory podle několika parametrů:

- Diskrétnost vs. kontinuita – Prostor může být buď diskrétní, nebo kontinuální. Diskrétní prostor je tvořen pevně stanovenými, oddělenými místy. Kontinuální prostor umožňuje pohyb v plynulém, nekonečném rozsahu.
- Počet dimenzí – Každý herní prostor má určitý počet dimenzí, které definují jeho rozsah a strukturu. Prostor může být jednorozměrný, dvourozměrný nebo dokonce třírozměrný.
- Ohraničenost a propojení oblastí – Prostor může být uzavřený, s pevně definovanými hranicemi, nebo otevřený, umožňující pohyb hráče nebo herních prvků mimo hranice.

Příkladem hry hrané na diskrétním dvoudimenzionálním poli jsou „piškvorky” (uvedené s herní plochou 3×3), kde je herní plocha rozdělena na devět oddělených políček. Herní deska je zobrazena jako souvislý prostor, ale z hlediska herních mechanik bereme v potaz pouze těchto devět specifických míst, která můžeme znázornit jako uzly v síti. Tedy dokud je jednoznačně rozeznatelné, do kterého políčka zapadá, jsou si mechanicky ekvivalentní, jak je naznačeno na obrázku 3.1 [30].

Příkladem jednodimenzionální hry jsou Monopoly. Přestože je herní deska vizuálně uspořádána do tvaru čtverce, po odstranění grafických prvků je patrné, že hra umožňuje pohyb po jediné řadě políček propojených v cyklické smyčce. V tomto případě se každé políčko na desce chová jako bod nulové dimenze, ačkoliv vizuálně vypadají některé čtverce odlišně, jejich funkce se neliší [29].

Herní prostor může zahrnovat i vnořené „podprostory“, což se často objevuje v počítačových hrách. Například může existovat venkovní prostor (kontinuální, dvou-rozměrný), ale hráč může narazit na ikony, které představují města nebo jeskyně. Tyto ikony přecházejí do zcela oddělených prostorů, které s venkovním prostorem nejsou přímo propojené, což je příkladem prostorového uspořádání, které je více založeno na mentálních modelech hráčů než na geografické realitě [29] [30].

3.2.1 Návrh herního prostoru

Jak již bylo zmíněno, navrhovaná hra bude 2D tahová strategie, tedy herní prostor bude dvoudimenzionální a diskrétní, složený ze čtvercových políček propojených po horizontální a vertikální ploše. To bude mít zásadní vliv na pohyb jednotek po mapě a tedy i veškerá strategická rozhodnutí hráčů.

Diskrétnost herního prostoru usnadňuje měření vzdáleností, po které se jednotky mohou po desce pohybovat, a také omezuje rozložení budov na jednu budovu na políčko. To samé platí pro jednotky, což umožňuje strategické tahy, jako blokování pohybu jednotek nepřítele.

Navrhovaná hra neobsahuje žádné podprostory ani separátní oblasti. Veškeré interakce probíhají na jediné herní ploše, přičemž se herní mechaniky soustředí na strategické využití této plochy.

Z hlediska návrhu a pozdější implementace tedy herní prostor uvažujeme jako dvou-rozměrné diskrétní pole, kde každé „políčko“ představuje bod s nulovou dimenzí. Tento pohled by měl zjednodušit návrh pravidel a interakcí mezi hráči a prostředím, což umožňuje efektivnější rozvoj herních taktik.

3.3 Objekty, atributy a stavy

V herním prostoru se pak nacházejí objekty, se kterými hráči v průběhu celé hry manipulují nebo s nimi interagují prostřednictvím jiných objektů. Může se jednat například o herní figurky, postavy, karty, nebo samotné herní prostředí. Objekty lze chápat jako „podstatná jména“ herních mechanik.

Každý objekt má atributy, které popisují jeho vlastnosti. Atributy lze chápat jako „přídavná jména“. Například auta v závodní hře mohou mít atributy jako *maximální rychlost* a *aktuální rychlost*.

Každý herní objekt má **stav**, který určuje jeho vlastnosti a chování ve hře. Stav je určen sadou jeho atributů, které představují jednotlivé charakteristiky objektu. Například figurka v šachu má atribut „mód pohybu“, který může nabývat stavů jako „volně se pohybující“, „v šachu“ a „matovaná“. V Monopoly má každý pozemek atribut „počet domů“, jehož stav se může měnit mezi 0 až 4 domy nebo hotelem.

Atributy mohou být statické nebo dynamické [29]:

- Statické atributy – Nemění se v průběhu hry. Například barva figurky v dámě nebo maximální rychlost auta.
- Dynamické atributy – Mohou se měnit v závislosti na akcích hráčů či mechanikách hry. Například aktuální rychlost auta se mění podle řízení hráče, životy jednotky klesají při útoku.

Každý herní objekt tak může mít kombinaci statických a dynamických atributů. Například v šachu má figurka statický atribut „barva”, ale dynamický atribut „pozice na šachovnici”, který se mění během hry.

Je důležité si uvědomit, že objekty ve hře často interagují mezi sebou. Například, jednotka může útočit na jinou jednotku, budova může produkovat suroviny, nebo terénní prvek může ovlivňovat pohyb jednotek. Tyto interakce by měly být navrženy tak, aby dávaly smysl a byly pro hráče srozumitelné.

Stav objektu není nutně viditelný celý – některé atributy mohou být skryté nebo viditelné jen pro určité hráče.

Důležitou součástí herního návrhu je rozhodnutí, které informace jsou hráčům přístupné. Pro účely návrhu jsou informace rozděleny na *veřejné*, *částečně skryté* nebo *zcela skryté* [29] [28].

- Veřejné informace – Všechny atributy a jejich stavy jsou viditelné pro všechny hráče. Například v šachu oba hráči vidí všechna pole a figurky na hrací desce, takže jedinou neznámou je strategie soupeře.
- Částečně skryté informace – Někteří hráči znají určité informace, které jsou ostatním skryté. Například v pokeru vidí hráči své vlastní karty, nikoliv karty soupeřů.
- Zcela skryté informace – Existují atributy, které zná pouze samotná hra. Například v počítačových hrách mohou být některé části světa před hráčem skryté, dokud je neodhalí.

Rozhodnutí o tom, kdo má přístup k jakým informacím, zásadně ovlivňuje herní strategii a atmosféru. Hry jako poker jsou postavené na utajení a odhadu soupeřových karet, zatímco v šachu mají hráči k dispozici informace o stavu celé herní plochy, a jedinou neznámou je strategie protivníka.

3.3.1 Návrh herních objektů

V této části jsou na základě probraných principů definovány jednotlivé objekty a jejich atributy. Ve hře se nachází několik typů objektů, se kterými hráči mohou manipulovat nebo s nimi interagovat. Základními objekty hry jsou **políčka**, která tvoří herní pole a určují podmínky pro pohyb jednotek a výstavbu budov. **Budovy** produkují suroviny, poskytují služby a plní strategické funkce. **Jednotky** jsou pohyblivé objekty ovládané hráčem, které slouží k různým činnostem, jako je boj či těžba surovin. Každý z těchto objektů má své specifické atributy a stavy, které určují jejich vlastnosti a možnosti ve hře.

Pokud jde o informace, bylo rozhodnuto, že hra nebude obsahovat skryté informace. Hráči tak od počátku vidí celý herní prostor i pohyby protivníka.

Políčka Políčka představují základní stavební jednotku herní mapy, na níž se odehrává hra. Každé políčko má několik atributů, popisujících jeho vlastnosti a interakce s ostatními objekty. Tyto atributy jsou:

| | | |
|-----------|---------------------|---|
| Statické | Pozice na mapě | Souřadnice určující umístění políčka v herním prostoru. |
| | Název | Název typu terénu. |
| | Zpomalení | Určité typy terénů mohou snižovat pohybovou rychlost jednotek. |
| | Bonusová obrana | Některé terény mohou zvyšovat obranu jednotek na daném políčku. |
| | Získatelné suroviny | Typ surovin které je možné na políčku získat. |
| Dynamické | Obsazenost jednotka | Zabraňuje vstupu více než jedné jednotky na dané políčko. |
| | Obsazenost budova | Na políčku může stát pouze jedna budova. |

Terén není jen grafický prvek, ale významně ovlivňuje hru. Správná volba umístění jednotek může znamenat rozdíl mezi vítězstvím a porážkou – například jednotka stojící na horách má lepší obranu, zatímco husté lesy mohou zpomalit postup nepřátel. Navíc různé druhy terénu určují, jaké budovy lze postavit a jaké suroviny lze těžit.

Budovy Budovy jsou struktury, které hráči staví na mapě za účelem generování zdrojů nebo poskytování jiných výhod. Každá budova má následující atributy:

| | | |
|-----------|------------------|--|
| Statické | Pozice na mapě | Pozice budovy na herní mapě. |
| | Název | Označení budovy. |
| | Vlastník | Určuje hráče, kterému budova patří. Změna vlastníků během hry nebude možná, pouze zničení nepřátelských budov. |
| | Typ terénu | Omezení, na kterých typech políček lze budovu postavit. |
| | Produkce za kolo | Množství surovin, které budova generuje za kolo. |
| | Životy max | Maximální počet životů budovy. |
| | Obrana | O kolik se zredukuje poškození způsobené útokem. |
| | Cena | Množství surovin potřebných pro stavbu budovy. |
| | Bonusová obrana | Budova může zvyšovat obranu jednotek nacházejících se na stejném políčku. |
| | Speciální funkce | Budova může umožňovat provádění speciálních akcí jako generování nebo vylepšování jednotek. |
| Dynamické | Životy | Aktuální počet životů budovy, pokud klesne na nulu, budova je zničena. |

Kromě generování surovin poskytují budovy hráčům strategické výhody, jako je vylepšování jednotek, zvyšování jejich obrany nebo blokování postupu nepřítele.

Jednotky Jednotky jsou pohyblivé objekty na mapě, které hráči ovládají a skrze které primárně interagují s herními mechanismy. Každá jednotka má své atributy, které určují její vlastnosti a schopnosti:

| | | |
|-----------|-----------------------------|---|
| Statické | Název | Název typu jednotky. |
| | Pozice na mapě | Souřadnice, na nichž se jednotka nachází v herním prostoru. |
| | Vlastník | Hráč, kterému jednotka patří. |
| | Cena | Množství surovin potřebné pro vytvoření jednotky. |
| | Cena za kolo | Náklady na udržování jednotky. Množství surovin, které jednotka spotřebuje každé kolo. |
| | Životy max | Maximální počet životů jednotky. |
| | Základní obrana | Hodnota, která redukuje poškození způsobené nepřátelským útokem. |
| | Útok | Základní útočná síla jednotky. Skutečné poškození se pohybuje v definovaném rozsahu (minimální a maximální poškození) a je ovlivněno šancí na kritický zásah, šancí na uhnutí cílové jednotky a je redukováno obranou cíle. |
| | Dosah | Vzdálenost, na kterou může jednotka útočit. |
| | Základní rychlost | Maximální počet políček, která může jednotka urazit za kolo. |
| | Násobitel kritického zásahu | Hodnota, kterou se násobí způsobené poškození v případě kritického zásahu. |
| | Šance na uhnutí | Pravděpodobnost, s jakou se jednotka může vyhnout přichozímu útoku a utrpět nulové poškození. |
| Dynamické | Životy | Aktuální počet životů jednotky, pokud klesne na nulu, jednotka zmizí. |
| | Obrana | Funkční obrana jednotky, po modifikaci prostředím (typem terénu nebo budovou). |
| | Rychlost | Skutečný počet políček, přes které se jednotka může v daném tahu pohybovat, po modifikaci terénem. |
| | Zaměstnaná | Indikuje, zda jednotka vykonala akci v tomto tahu. |
| | V pohybu | Indikuje, zda se jednotka v tahu pohybovala. |
| | Zkušenosti | Jednotka může být vylepšena v konkrétní budově po dosažení určitého počtu zkušeností, které získává prováděním akcí odpovídajících jejímu typu (válečníci získávají zkušenosti bojem, pracovníci těžbou surovin nebo opravami budov). |

Jednotky představují hlavní prostředek, kterým hráč ovlivňuje dění na mapě. Každá jednotka má specifickou roli – některé jsou určené pro boj, jiné pro těžbu surovin nebo výstavbu budov. Postupem času mohou jednotky získávat zkušenosti a vylepšovat své schopnosti, což přidává další vrstvu strategického rozhodování. Jednotky navíc každé kolo spotřebovávají určité množství surovin, proto hráč musí zvážit, zda

si může dovolit udržovat velkou armádu.

3.4 Akce hráče

Akce jsou jedním ze základních stavebních kamenů herních mechanik. Lze je chápat jako „slovesa” hry, jelikož definují, jak může hráč interagovat s herním světem [29]. Akce jsou rozděleny do dvou hlavních kategorií:

Operační akce jsou základní činnosti, které může hráč přímo vykonat, například pohyb jednotky nebo útok [29].

Výsledné akce vycházejí z kombinace operačních akcí. Tyto emergentní akce často nejsou přímo definovány pravidly, ale vznikají přirozeně během hry a přispívají k její hloubce [29].

Operační akce jsou základní mechanismy, které hráč využívá pro interakci s herním světem. V některých hrách je množství těchto akcí omezené, což vede k menší variabilitě herního stylu. V jiných hrách mají hráči k dispozici širokou škálu možností, což umožňuje kreativní přístupy. Například ve hře dáma má hráč k dispozici tři základní operační akce:

- Posun kamene vpřed.
- Přeskok soupeřova kamene.
- Pohyb zpět v případě dosažení úrovně krále.

Výsledné akce vznikají kombinací operačních akcí a přispívají ke strategické hloubce hry. Zatímco operační akce jsou pevně dané pravidly, výsledné akce se objevují jako důsledek interakcí mezi hráčem, herním prostředím a protivníky. Ve hře dáma mohou být výslednými akcemi například:

- Ochrana kamene – umístění jiného kamene za něj, aby zabránil zajetí.
- Vynucení tahu soupeře – postavení figurky tak, že soupeř musí provést nevýhodný tah.
- Obětování kamene – nabídnutí figurky soupeři s cílem získat lepší pozici na hrací ploše.

Výsledné akce přidávají hře hloubku a umožňují emergentní chování hráče. Čím větší je poměr výsledných akcí vůči operačním akcím, tím více hra podporuje kreativitu hráče [30].

3.4.1 Návrh herních akcí

Tato část popisuje operační akce, které může hráč provádět, a dále zmiňuje vzniklé výsledné akce. V rámci tahového systému může hráč během svého tahu každou jednotkou provést pohyb a jednu další akci. Některé akce mohou proběhnout pouze pokud jsou splněny určité podmínky, například pozice jednotky v určité budově. Vzhledem k tahovému systému a interakci mezi jednotkami a terénem mohou vznikat nové strategie, které nelze vždy předvídat.

Operační akce jsou základní činnosti, které může hráč vykonávat, primárně interakcí se svými jednotkami.

| | |
|--------------------|--|
| Pohyb | Jednotka se pohne o počet polí odpovídající jejímu typu a terénu. |
| Útok | Jednotka zaútočí na nepřátelskou jednotku v dosahu, způsobí poškození ($poškození = útok - obrana$) a pokud nepřítel přežije, provede protiútok. |
| Těžba surovin | Pracovní jednotka získá určité množství surovin v závislosti na typu terénu, na kterém těžba probíhá. |
| Práce | Pracovní jednotka v budově může získávat větší množství surovin, v budově která normálně generuje suroviny pasivně. |
| Stavba budovy | Budovatelská jednotka postaví novou budovu na vhodném políčku. |
| Oprava budovy | Pracovní nebo budovatelská jednotka obnoví část života poškozené budovy. |
| Vylepšení jednotky | Pokud jednotka splní požadavky (např. bojové zkušenosti, existenci potřebné budovy, ...), může být vylepšena na silnější variantu. |

Výsledné akce vznikají kombinací operačních akcí a strategického uvažování hráče.

| | |
|-----------------------|--|
| Obrana klíčových bodů | Hráč rozmístí jednotky tak, aby blokovaly přístup k důležitým budovám nebo oblastem. |
| Napadání zásobování | Cílený útok na pracovníky nebo budovy snižující zdroje nepřítele. |
| Taktický ústup | Ústup jednotek do bezpečnější oblasti, například k opravám nebo pod ochranu budov. |
| Obklíčení | Koordinovaný útok více jednotek k eliminaci klíčových nepřátel. |
| Zdržovací taktika | Hráč strategicky obětuje jednotky nebo využívá terén, aby zpomalil postup nepřítele. |
| Ofenzivní opevnění | Hráč strategicky buduje opevnění v blízkosti nepřátelské základny s cílem získat taktickou výhodu. |

3.5 Pravidla hry

Pravidla určují, jakým způsobem je hra hraná, jaké akce může hráč provádět a jaké jsou herní cíle. Bez jasně definovaných pravidel není možné vytvořit funkční a spravedlivé herní prostředí.

Špatně strukturovaná pravidla mohou vést k nevyváženosti, nechtěným exploitačním mechanikám nebo dokonce ke ztrátě zábavnosti. Správně nastavená pravidla naopak motivují hráče k objevování efektivních strategií vedoucích k vítězství.

Pravidla jsou soubor omezení a možností, které definují interakce hráčů s herním světem. V každé hře existují různé typy pravidel, která ovlivňují hratelnost [29]:

- Akce hráčů – Co mohou hráči během hry dělat?
- Stav hry – Jak se hra mění v průběhu času?
- Cíle hry – Kdy hra končí a kdo vyhrává?

Tato pravidla společně vytvářejí herní mechaniky, které určují dynamiku hry a poskytují hráčům výzvy k řešení.

David Parlett, britský herní teoretik, rozdělil pravidla do několika kategorií [29]:

| | |
|---------------------|--|
| Operační pravidla | Popisují základní akce hráčů, například „hráč se může pohnout o n políček”. |
| Základní pravidla | Definují matematický model hry, určují, jak se hra vyvíjí na základě akcí hráčů. |
| Chování hráčů | Implicitní pravidla „fair play” a sportovního chování. |
| Psaná pravidla | Formálně dokumentovaná pravidla hry. |
| Zákony | Další pravidla, která mohou být zavedena například pro turnajovou hru. |
| Oficiální pravidla | Spojují psaná pravidla a zákony do jednoho uceleného souboru. |
| Doporučená pravidla | Tipy a strategie pro efektivní hraní. |
| Domácí pravidla | Úpravy pravidel hráči pro přizpůsobení hry jejich preferencím. |

Herní pravidla mohou být ovlivněna herním režimem, který může do hry přidávat komplexitu přidáním nebo odebráním pravidel. Případně úpravou podmínek vítězství. [30]

Nejdůležitějšími pravidly jsou cílové podmínky. Každá hra musí mít jasně definovaný cíl, který hráče motivuje a určuje, kdy hra končí. Dobrý cíl by měl podle [29] splňovat:

- Konkrétnost – Je jasné, co musí hráči udělat.
- Dosažitelnost – Hráči mají reálnou možnost dosáhnout cíle.
- Odměňující charakter – Splnění cíle přináší uspokojení.

3.5.1 Návrh pravidel

Tato část shrnuje konkrétní návrh herních pravidel. Pravidla definují podmínky vítězství, možné akce hráčů a interakce mezi herními objekty. Primárním cílem hry je poražení protivníka, což je dosaženo zničením všech jeho jednotek a budov. Hra končí, jakmile na herní ploše zůstane pouze jeden hráč. Hráč si může sám stanovit sekundární cíle, jako jsou:

- Efektivní správa ekonomiky – těžba surovin a stavba podpůrných budov.
- Budování armády – rekrutování a vylepšování jednotek.
- Dobývání strategických bodů – kontrola území s cennými zdroji nebo taktickými výhodami.

Objekty Hra obsahuje tři typy herních objektů:

| | | |
|----------|------------|--|
| Jednotky | Bojové | primárně slouží k útoku a obraně (např. válečník). |
| | Stavební | mohou stavět nové budovy, opravovat poškozené budovy a těžit suroviny. |
| Budovy | Ekonomické | pasivně produkují zdroje (např. důl). |
| | Obranné | poskytují ochranu a strategickou kontrolu území. |
| | Vývojové | umožňují vylepšovat jednotky na vyšší úroveň. |
| Terén | Terén | Ovlivňuje pohyb a bojové vlastnosti jednotek. (Např. hory zpomalují pohyb, lesy poskytují krytí, voda je nepřekročitelná.) |

Akce V každém tahu může hráč provést s každou svou jednotkou jeden pohyb a jednu akci s tím, že některé akce vyžadují specifické podmínky, například vylepšení vyžadují určité budovy nebo sběr surovin má různý efekt podle terénu. Akce zahrnují:

| | |
|--------------------------------|---|
| Pohyb jednotek | Jednotky se mohou pohybovat pouze vertikálně a horizontálně, diagonální pohyb není možný. |
| | Terén ovlivňuje pohyb jednotek (např. hory snižují jejich pohybovou vzdálenost). |
| Útok | Hráč vybere cílovou jednotku v dosahu útoku. |
| | Útok je vyhodnocen podle atributů jednotek (útok vs. obrana). |
| | Pokud obránce přežije, provede protiútok. |
| Výstavba a interakce s objekty | Stavební jednotky mohou budovat budovy na určitých místech. |
| | Pracovní jednotky mohou těžit suroviny, případně opravovat budovy |

Vynucování pravidel Pravidla jsou automaticky vynucována herním systémem. Hráč může provádět pouze akce, které jsou v dané situaci povolené (např. jednotky nemohou cestovat přes políčka s vodou). Hra automaticky vyhodnocuje útoky s ohledem na náhodnou variabilitu poškození (mezi minimálním a maximálním poškozením útočníka), šanci na kritický zásah (která násobí způsobené poškození) a šanci na uhnutí cílové jednotky (která může vést k nulovému poškození). Poškození je redukováno obranou cíle. Pokud životy jednotky klesnou na nulu, je automaticky odstraněna. Hra automaticky končí, jakmile některý hráč přijde o všechny jednotky a budovy, čímž ztratí možnost provádět jakékoliv další akce.

3.6 Dovednost a náhoda

V herním designu je klíčové pochopit a vyvážit interakci mezi dovedností hráče a prvky náhody. Jesse Schell ve své knize *The Art of Game Design: A Book of Lenses* zdůrazňuje, že jak dovednost, tak náhoda jsou nezbytné pro vytvoření poutavého a znovu hratelného herního zážitku.

Dovednost v kontextu her odkazuje na schopnost hráče dosahovat lepších výsledků prostřednictvím učení, praxe a strategického myšlení. Hry založené primárně na dovednosti odměňují hráče za jejich schopnost rozpoznávat vzorce, plánovat dopředu, efektivně provádět akce a přizpůsobovat se měnícím se podmínkám. Příkladem dovedností může být taktické umístění jednotek, efektivní správa zdrojů nebo rychlé rozhodování pod tlakem [29].

Přílišná závislost na dovednosti však může vést k tomu, že bude hra nepřístupná pro nové hráče, nebo se stane předvídatelnou a repetitivní.

Náhoda zavádí do hry nepředvídatelnost a variabilitu. Může se projevat v mnoha formách, jako jsou hody kostkou, náhodné události, nepředvídatelné chování protivníků nebo procedurální generování herního světa. Náhoda může hru oživit, vytvořit dramatické momenty a zajistit, že žádné dvě hry nebudou identické, což zvyšuje znovuhratelnost [29].

Na druhou stranu, přílišná náhoda může vést k pocitu, že rozhodnutí hráče nemají smysl, a hra se tak stane čistě závislou na štěstí. To může být frustrující a podkopávat strategickou hloubku.

Vyvážení dovednosti a náhody je umění. Cílem je navrhnout hru tak, aby dovednost hráče umožňovala zmírnit dopady nepříznivé náhody nebo naopak využít příznivou náhodu. Náhoda by měla vytvářet zajímavé volby a scénáře rizika/odměny, spíše než negovat strategická rozhodnutí hráče [30] [29].

3.6.1 Dovednost a náhodnost v navržené hře

V kontextu navrhované strategické hry jsou dovednost hráče a prvky náhody pečlivě integrovány s cílem dosáhnout dynamického a znovu hratelného zážitku.

Vliv dovednosti hráče: Dovednost hráče se ve hře projevuje především v:

- **Strategickém plánování:** Schopnost hráče efektivně spravovat zdroje, budovat armádu s vhodnou složením a plánovat dlouhodobé ofenzivní či defenzivní operace.
- **Taktickém provedení:** Optimální umístění jednotek na mapě, výběr nejvhodnějších cílů pro útok s ohledem na jejich typ a stav, a efektivní využití terénních výhod pro boj i pohyb.

Vliv náhody: Náhoda je do hry integrována na několika úrovních, aby zajistila variabilitu a nepředvídatelnost průběhu hry, což je klíčové pro smysluplné simulace Monte Carlo:

- **Procedurálně generovaná mapa:** Jak je popsáno v Kapitole 2, herní mapa je generována procedurálně. Každá hra tak začíná na jedinečném herním poli s odlišným rozložením terénu a zdrojů, což nutí hráče i AI k adaptaci strategií.
- **Variabilní poškození:** Při každém útoku je způsobené poškození náhodně vybráno z definovaného rozsahu. To znamená, že i identické jednotky útočící na stejný cíl mohou mít mírně odlišné výsledky, což zvyšuje nepředvídatelnost jednotlivých střetů.
- **Kritické zásahy:** S pevnou globální šancí může útok způsobit kritický zásah, který násobí způsobené poškození. Tyto momenty vysokého dopadu přidávají do bojů napětí a dynamiku.
- **Šance na uhnutí:** Cílová jednotka má šanci se útoku zcela vyhnout a utrpět nulové poškození. Tento prvek náhody ovlivňuje přežití jednotek a nutí hráče zvažovat pravděpodobnosti při plánování útoků.

Kapitola 4

Implementace základních mechanik

4.1 Implementace prototypu herních mechanik

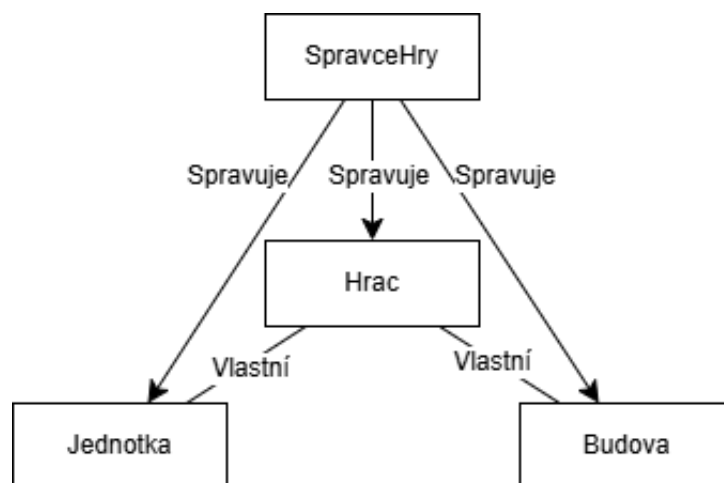
Implementovaný prototyp, včetně generátoru map a základních herních mechanik, je napsaný v jazyce **Python 3.9**. Pro správu a analýzu dat, a také pro vizualizaci výstupů, byly použity knihovny jako *pandas*, *matplotlib* a *seaborn*. Pro práci se souborovým systémem a generování náhodných hodnot byly využity knihovny *os*, *glob* a *random*, zatímco pro manipulaci s daty ve formátu CSV a s časovými razítky sloužily knihovny *csv* a *datetime*.

Kompletní kód prototypu je v příloženém souboru ve složce `generator_map/`.

4.1.1 Architektura a účel prototypu

V tomto prototypu jsou implementovány základní herní mechaniky nezbytné pro provádění simulací a shromažďování dat, na jejichž základě budou v dalších fázích výzkumu rozšiřovány typy jednotek a budov a upravovány jejich statistiky. Architektura prototypu je založena na čtyřech klíčových třídách (diagram tříd: 4.1), které zajišťují funkčnost herního cyklu a interakci mezi herními objekty:

- **SpravceHry**: Centrální řídicí prvek, který spravuje průběh hry, střídání hráčů a implementuje základní umělou inteligenci. Zajišťuje koordinaci mezi ostatními třídami.
- **Hrac**: Reprezentuje jednotlivého hráče ve hře. Spravuje jeho ekonomiku (suroviny), vlastněné jednotky a budovy.
- **Jednotka**: Představuje herní jednotky s atributy jako útok, obrana, životy a pohyb.



Obrázek 4.1: Zjednodušený diagram tříd.

- **Budova:** Reprezentuje hráčské budovy, které v prototypu primárně slouží k produkci surovin. Jejich implementace je v prototypu zjednodušená, spíše abstraktní.

Účelem tohoto prototypu je získat funkční základ pro simulace herních scénářů s různými konfiguracemi jednotek a budov. Získaná data poslouží k informovanému návrhu nových typů jednotek a budov a k úpravě jejich statistik pro dosažení vyvážené hratelnosti v plné verzi hry.

4.1.2 Třída SpravceHry

Třída **SpravceHry** je hlavním řídicím prvkem celé hry. Zajišťuje koordinaci všech částí systému, spravuje průběh jednotlivých tahů a obsahuje také jednoduchou implementaci umělé inteligence ovládající hráče pro účely testování. Jedná se o centrální bod, který propojuje třídy **Hrac**, **Jednotka** a **Budova** a zabezpečuje jejich souhru v rámci herního cyklu.

Hlavní úlohy třídy zahrnují:

- aktualizaci stavu hry v jednotlivých tazích,
- správu střídání hráčů a ukončení hry při splnění podmínky vítězství,
- rozhodování AI o pohybu a útocích jednotek.

Aktualizace hry. Před začátkem hry je pomocí `inicializace_hry` vytvořeno počáteční rozestavení hráčů včetně jejich základů a startovní ekonomiky (`startovni_domek`). Pomocí metody `aktualni_hrac` se zjistí, který hráč je aktuálně na tahu. Střídání hráčů je zajištěno metodou `dalsi_hrac`, která po odehrání všech hráčů zvyšuje číslo kola. Při splnění podmínky vítězství (např. zničení základny) je hra ukončena metodou `konec`, která zaznamená výsledek a ukončí herní cyklus.

Provedení tahu. Metoda `proved_tah` spravuje celý tah aktuálního hráče, včetně aktualizace surovin, vyhodnocení údržby jednotek a provedení akcí AI. Metoda `kontrola_bojeschopnosti` ověřuje, zda hráč po ztrátách ještě má bojeschopné jednotky.

Rozhodování AI. Jednoduchá AI, implementovaná metodami `ai_tah`, `ai_pohyb_a_utoke` a `stavba_a_verbovani_ai`, vyhodnocuje akce jednotek hráče v daném tahu. Rozhodování AI je ovlivněno globálními vahami, které definují preference pro různé typy akcí a cílů. Pokud jsou v dosahu útoku jednotky AI více nepřátelských cílů, AI vybere cíl pravděpodobnostně na základě těchto vah (např. s vyšší prioritou pro střelecké jednotky nebo budovy). Specifickou výjimkou jsou jednotky s větším dosahem (střelci), které před útokem kontrolují, jestli nestojí nepřítel vedle nich, a pokud ano, pokusí se od nepřítele 'utéct', aby jim nehrozil protiútok. Pokud jednotka nenajde žádný cíl k útoku ani cestu k nepřátelské základně, může se s určitou pravděpodobností pohnout náhodným směrem. Po pohybu jednotek, na základě aktuálního zisku surovin a s ohledem na globální váhy, AI rozhoduje o stavbě nových budov nebo verbování nových jednotek.

Verbování a Stavba Nové jednotky jsou vytvářeny metodou `verbovani`, která ověřuje podmínky a najde volné místo v okolí základny. Stavba budov probíhá obdobně metodou `stavba_budovy`, pouze je zde na pevně daná pozice, jelikož umístění budov neberu v prototypu v potaz, pracuji s nimi jako s abstraktními.

Souboje Metoda `vyhodnot_souboj` řeší útok jednotky na protivníka, včetně protiútoků a odstranění padlých jednotek. Pokud padne základna, hra je ukončena metodou `konec`.

4.1.3 Třída Hrac

Třída `Hrac` reprezentuje jednotlivého hráče ve hře. Uchovává jeho jméno, seznam jednotek a budov, dostupné suroviny. Hlavní odpovědností třídy je správa ekonomiky hráče, zejména manipulace se surovinami, zpracování údržby jednotek a vyhodnocení následků nedostatku zdrojů.

Hlavní úlohy třídy zahrnují:

- správu zásob surovin a jejich změn,
- evidenci vlastněných jednotek a budov,
- zajištění výroby surovin budovami,
- řešení údržby jednotek a následků nedostatku surovin.

Práce se surovinami. Třída umožňuje přidávat nové suroviny pomocí metody `pridej_suroviny` a odebírat suroviny pomocí `odecti_suroviny`. Při odečítání je kontrolováno, zda má hráč dostatek požadovaných zdrojů.

Správa ekonomiky a údržby. Každé kolo může hráč získat nové suroviny produkováné budovami prostřednictvím metody `zisk_z_budov`. Údržba jednotek je řešena metodou `zpracuj_udrzbu`, která odečítá příslušné množství surovin. Pokud hráč nemá dostatek zdrojů, všechny jeho jednotky utrpí ztrátu životů.

Nedostatek jídla. Konkrétním případem správy ekonomiky je hladovění jednotek, implementované metodou `zpracuj_nedostatek_jidla`. Pokud hráč nemá dostatek jídla, jeho jednotky ztrácejí životy.

4.1.4 Třída Jednotka

Třída `Jednotka` reprezentuje jednu funkční jednotku ve hře. Její hlavní úlohou je definovat chování jednotky v rámci herního světa, včetně pohybu, boje a interakce s herním prostředím.

Hlavní úlohy třídy zahrnují:

- uchovávání atributů jednotky (typ, pozice, statistiky, cena, vlastník),
- výpočet možných pohybů na základě rychlosti a terénu,
- realizaci pohybu na zvolenou cílovou pozici,
- vyhledávání nepřátelských jednotek v dosahu útoku,
- provádění útoku na nepřátelské jednotky a přijímání protiútoků,
- správu životů a mechanismus zániku jednotky.

Pohyb. Metoda `vypocet_moznych_pohybu` na základě aktuální pozice, rychlosti jednotky a typu terénu na herní mapě určuje všechny dosažitelné polohy. Při výpočtu zohledňuje překážky, jako je voda nebo přítomnost jiných jednotek. Metoda `proved_pohyb` následně aktualizuje pozici jednotky, pokud je cílová pozice v seznamu možných pohybů.

Boj. Metoda `najdi_cile_v_dosahu` prozkoumá okolí jednotky a vrátí seznam nepřátelských jednotek, které se nacházejí v jejím dosahu útoku. Samotný útok je realizován metodou `proved_atak`, která počítá poškození s náhodnou variabilitou (vybírá hodnotu mezi `min_damage` a `max_damage` útočící jednotky). Dále zohledňuje šanci na kritický zásah (s globální konstantou `CRITICAL_HIT_CHANCE` a násobitelem `crit` útočníka) a šanci na uhnutí cílové jednotky (`uhyb` obránce). Poškození je redukováno obranou bránící se jednotky, s případnými modifikátory terénu. Metoda `proved_protiatak` umožňuje bránící se jednotce odpovědět na útok, pokud je útočník v jejím dosahu. Protiútok je vyhodnocen stejnou logikou jako standardní útok, což zajišťuje konzistentní aplikaci náhodných bojových faktorů.

4.1.5 Třída Budova

Třída **Budova** obecně reprezentuje strukturu vlastněnou hráčem, která plní specifickou funkci v rámci hry. Může se jednat o budovy produkující suroviny, obranné stavby nebo jiné speciální budovy. Třída uchovává informace o typu budovy, její pozici na mapě, vlastníkovi, životech, obraně, produkci surovin a ceně za postavení.

Hlavní úlohy třídy zahrnují:

- uchovávání atributů budovy (typ, pozice, vlastník, životy, obrana, produkce, cena),
- generování surovin.

Produkce surovin. Pokud je budova určena k produkci surovin, slovník **produkce** definuje typy surovin a jejich množství, které budova vyprodukuje za jedno herní kolo. Metoda **generuj_suroviny** vrací tento slovník, čímž umožňuje hráči získávat zdroje.

Umístění. V tomto prototypu je pozice budovy pevně daná při jejím vytvoření a nelze ji měnit. Metoda pro stavbu budovy ve třídě **SpravceHry** zajišťuje její umístění na předdefinovanou pozici. V prototypu budovy neinteragují s pohybem jednotek ani mezi sebou, takže jejich přesné umístění nemá funkční dopad.

Speciální funkce a blokování cesty. Budovy mají v prototypu implementovanou pouze schopnost generovat suroviny. Jejich speciální schopnosti a schopnost blokovat cestu jsem se rozhodl implementovat až v rámci celé hry.

Kapitola 5

Simulace

5.1 Úvod do simulací

Tato kapitola podrobně popisuje průběh simulací, jejichž cílem je získání dat pro vyvážení parametrů herních jednotek, budov a celého herního pole. Pro zajištění robustnosti dat simulace využívají specifické scénáře a metodu Monte Carlo [31]. Vybraná data jsou prezentována přímo v textu ve formě tabulek; kompletní datové soubory jsou k dispozici ve složce `vysledky_simulace` ve formátu CSV.

Simulace jsou rozděleny do dvou hlavních fází:

- **Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích.** Tato fáze se zaměřuje na analýzu bojů jednotlivých jednotek a menších skupin v kontrolovaném prostředí.
- **Fáze 2: Testování vlivu náhodného herního pole.** V této fázi je podrobně analyzován vliv parametrů generování mapy na charakteristiky a dynamiku herního pole.

5.1.1 Cíle simulací

Hlavními cíli těchto simulací jsou:

- Systematické testování atributů jednotek, produkce budov a nákladů herních prvků.
- Nalezení optimálního a vyváženého nastavení herních parametrů, zajišťujícího, že žádný herní prvek nebude dominantní ani irelevantní.
- Analýza charakteristik generovaného herního pole v závislosti na jeho parametrech a posouzení jejich vlivu na strategické možnosti hry.

5.2 Teorie simulace

Pro provádění simulací je využit přístup založený na Monte Carlo metodě. Tato metoda spočívá v opakovaném náhodném vzorkování a simulaci s cílem získat statisticky robustní odhady chování komplexních systémů.

5.2.1 Monte Carlo metoda v kontextu hry

V kontextu této hry je metodu Monte Carlo možné využít díky prvkům náhodnosti. Konkrétně se jedná o variabilní poškození, kritické zásahy a šanci na uhnutí v bojovém systému, a také o náhodný pohyb a vážený výběr cíle v rozhodování jednoduché AI vytvořené pro simulace. Tyto prvky zajišťují, že i při identických počátečních podmínkách se jednotlivé herní průběhy mohou výrazně lišit.

5.2.2 Reprodukovatelnost a správa náhodnosti

Pro zajištění reprodukovatelnosti výsledků simulací je klíčová správa generování náhodných čísel. To znamená, že pro samotný sběr dat pro Monte Carlo analýzu je nezbytné, aby každý simulační běh byl nezávislý a zároveň reprodukovatelný. Reprodukovatelnosti je dosaženo použitím ID dané simulace jako *seedu* pro generátor náhodných čísel

(`random.seed(simulation_id)`). Tento přístup zajišťuje, že získaná data skutečně reprezentují rozložení možných výsledků v náhodném prostředí a zároveň umožňuje přesnou reprodukci jakéhokoli konkrétního simulačního běhu pro detailní analýzu.

5.3 Fáze 1: Testování vyváženosti jednotek v izolovaných scénářích

Tato fáze se zaměřuje na testování jednotlivých typů jednotek v kontrolovaných bojových situacích. Cílem je získání detailních dat o výkonu každé jednotky a dopadu náhodnosti na výsledky soubojů.

5.3.1 Popis testovaných scénářů

V rámci Fáze 1 byli pro testování vyváženosti jednotek použity předpřipravené scénáře, které poskytují kontrolované prostředí pro analýzu výkonu jednotek.

- **Scénář: Rovina**

- **Popis mapy:** Velká obdélníková mapa (např. 15x15 polí typu 'P' - pláň).
- **Obecné umístění jednotek:** Jednotky budou začínat v opačných rozích mapy (např. (0, 0) a (14, 14)).

- Scénář je určen ke srovnání atributů jednotek v otevřeném prostoru, kde prostředí nemá žádný zásadní vliv.

- **Scénář: Pohoří**

- **Popis mapy:** Obdélníková mapa (např. 15x15) s horami ('H') rozdělujícími mapu napůl, oblepenými lesy ('L').
- **Obecné umístění jednotek:** Jednotky budou začínat v opačných rozích mapy (např. (0, 0) a (14, 14)).
- Horský terén ('H') modifikuje obranu jednotek stojících na něm (+2 obrana) a zároveň zásadně zpomaluje pohyb. Lesy ('L') podobně snižují rychlost pohybu jednotek. Scénář je určen ke srovnání schopností jednotek v komplexnějším terénu, který ovlivňuje pohyb a bojové schopnosti jednotek.

5.3.2 Metodika

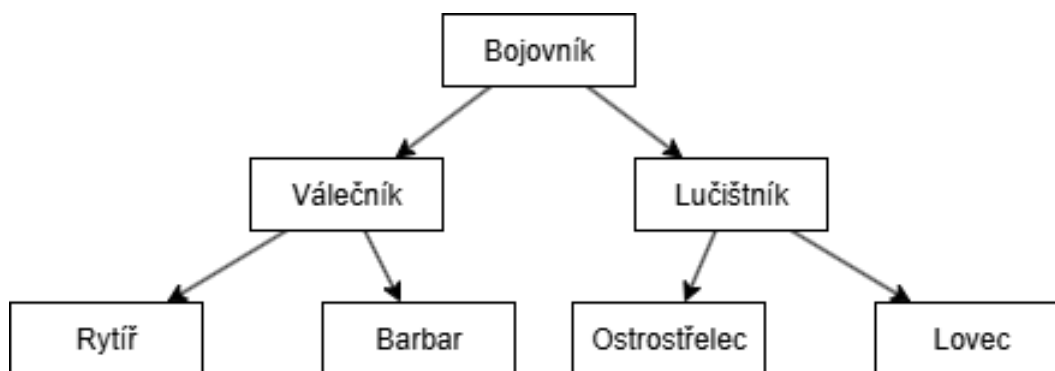
Postup simulací byl následující:

1. **Definice sady atributů:** Nastaví se konkrétní sada atributů pro porovnávání jednotky.
2. **Spuštění simulací:** Pro definovanou sadu atributů se provede 1000 simulací v rámci každého scénáře („Rovina” a „Pohoří”).
3. **Sběr a uložení dat:** Data z každého simulačního běhu jsou uložena ve strukturovaném formátu (CSV). Separátně se ukládají souhrnné informace o simulaci a detailní informace o stavu jednotek v každém kole.
4. **Opakování:** Kroky 1-3 se opakují pro několik dalších sad atributů, aby se shromáždila dostatečná data pro komplexní porovnání.
5. **Analýza dat:** Komplexní analýza a porovnání výsledků se provede teprve po shromáždění dat z více sad atributů.

5.3.3 Duely

Primárním cílem je posoudit, zda jsou jednotky na stejné úrovni vzájemně vyvážené, což je klíčové pro férové a strategicky zajímavé herní prostředí. Každý duel byl testován v různých terénních podmínkách (scénářích), aby se zohlednil vliv prostředí na bojové výsledky.

Zároveň byly jednotky testovány proti jednotkám z nižší úrovně, aby byla zachována vyváženost mezi jednotlivými úrovněmi a jednotky na vyšší úrovni bylo stále nutné využívat strategicky i proti slabším jednotkám. Teoretický vývojový strom jednotek je vidět na obrázku 5.1.



Obrázek 5.1: Teoretická ukázka postupu jakým se jednotky budou v plné hře moci vyvíjet.

Válečník vs Lučištník

Tento duel porovnával Válečníka, první vývojový stupeň pro boj na blízko s vyšší obranou, a Lučištníka, což je vývojový stupeň vedoucí k boji na dálku s nižší obranou, ale se schopností útočit z bezpečné vzdálenosti. Obě jednotky představují základní archetypy bojových jednotek a očekává se, že jsou dostupné v rané fázi hry.

Jelikož se jedná o druhý vývojový stupeň, a tedy jsou na této úrovni jen tyto dvě jednotky, očekává se, že bude souboj vyrovnaný. Válečník by měl mít mírnou výhodu na rovině, kde mu nic brání v rychlém útoku na lučištníka. Zatímco lučištník by měl vyhrávat v komplexním terénu, jelikož zde má více času útočit z bezpečné vzdálenosti.

Již ze simulace první sady atributů (tabulka: 5.1; ID: 1) bylo vidět, že souboj trvá neúměrně dlouho vzhledem k velikosti mapy (15x15 polí). Souboj na hladkém terénu trval průměrně téměř deset kol, z toho je vidět, že v první řadě musí dojít ke zvýšení rychlosti. Spolu s tím se logicky dá předpokládat, že je zapotřebí zvyšovat dosah Lučištníka.

Ze simulací s upravenými atributy je vidět, že efekt na rychlost hry postupně klesal se zvyšujícím se atributem rychlosti, zároveň se v simulacích ukázalo, že zvýšení dosahu téměř nebylo zapotřebí. Výsledky jsou názorněji vidět z grafů 5.2, který zobrazuje poměr vítězství na rovné mapě, a 5.3, který zobrazuje poměr vítězství v komplexním terénu. Pro další simulace se tedy jako základ zvýší atributy rychlosti a o jeden bod se také zvýší dosah Lučištníka (tyto jsou atributy v tabulce 5.1 s ID: 6).

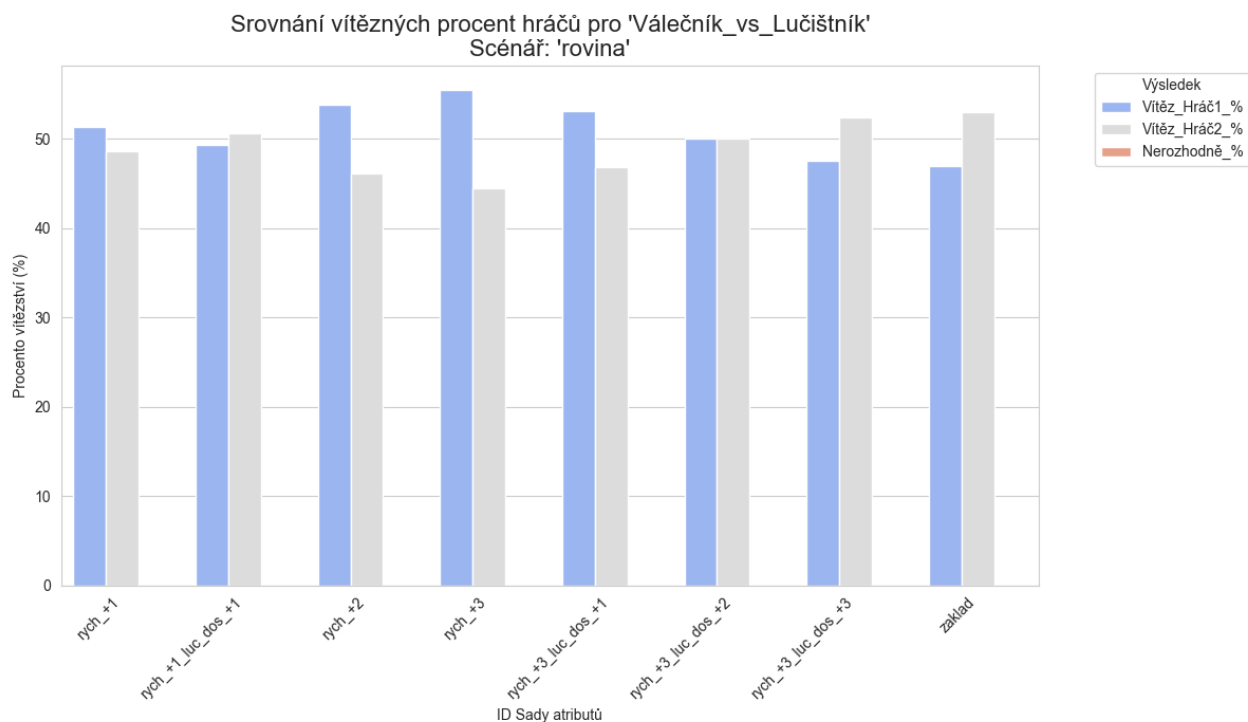
Kompletní data získaná z této simulace jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Duely/2_uroven/rychlost`.

| Sada atributů | ID | Typ | Útok Min | Útok Max | Obrana | Rychlost | Dosah | Krit. mult. | Úhyb | Max. životy |
|--------------------|----|-----------|----------|----------|--------|----------|-------|-------------|------|-------------|
| zaklad | 1 | valecnik | 6 | 8 | 4 | 3 | 1 | 1.2 | 0.05 | 15 |
| zaklad | 1 | lucistnik | 6 | 8 | 3 | 3 | 4 | 1.2 | 0.05 | 10 |
| rych_+1_luc.dos_+1 | 2 | valecnik | 6 | 8 | 4 | 4 | 1 | 1.2 | 0.05 | 15 |
| rych_+1_luc.dos_+1 | 2 | lucistnik | 6 | 8 | 3 | 4 | 5 | 1.2 | 0.05 | 10 |
| rych_+1 | 3 | valecnik | 6 | 8 | 4 | 4 | 1 | 1.2 | 0.05 | 15 |
| rych_+1 | 3 | lucistnik | 6 | 8 | 3 | 4 | 4 | 1.2 | 0.05 | 10 |
| rych_+2 | 4 | valecnik | 6 | 8 | 4 | 5 | 1 | 1.2 | 0.05 | 15 |
| rych_+2 | 4 | lucistnik | 6 | 8 | 3 | 5 | 4 | 1.2 | 0.05 | 10 |
| rych_+3 | 5 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| rych_+3 | 5 | lucistnik | 6 | 8 | 3 | 6 | 4 | 1.2 | 0.05 | 10 |
| rych_+3_luc.dos_+1 | 6 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| rych_+3_luc.dos_+1 | 6 | lucistnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| rych_+3_luc.dos_+2 | 7 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| rych_+3_luc.dos_+2 | 7 | lucistnik | 6 | 8 | 3 | 6 | 6 | 1.2 | 0.05 | 10 |
| rych_+3_luc.dos_+3 | 8 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| rych_+3_luc.dos_+3 | 8 | lucistnik | 6 | 8 | 3 | 6 | 7 | 1.2 | 0.05 | 10 |

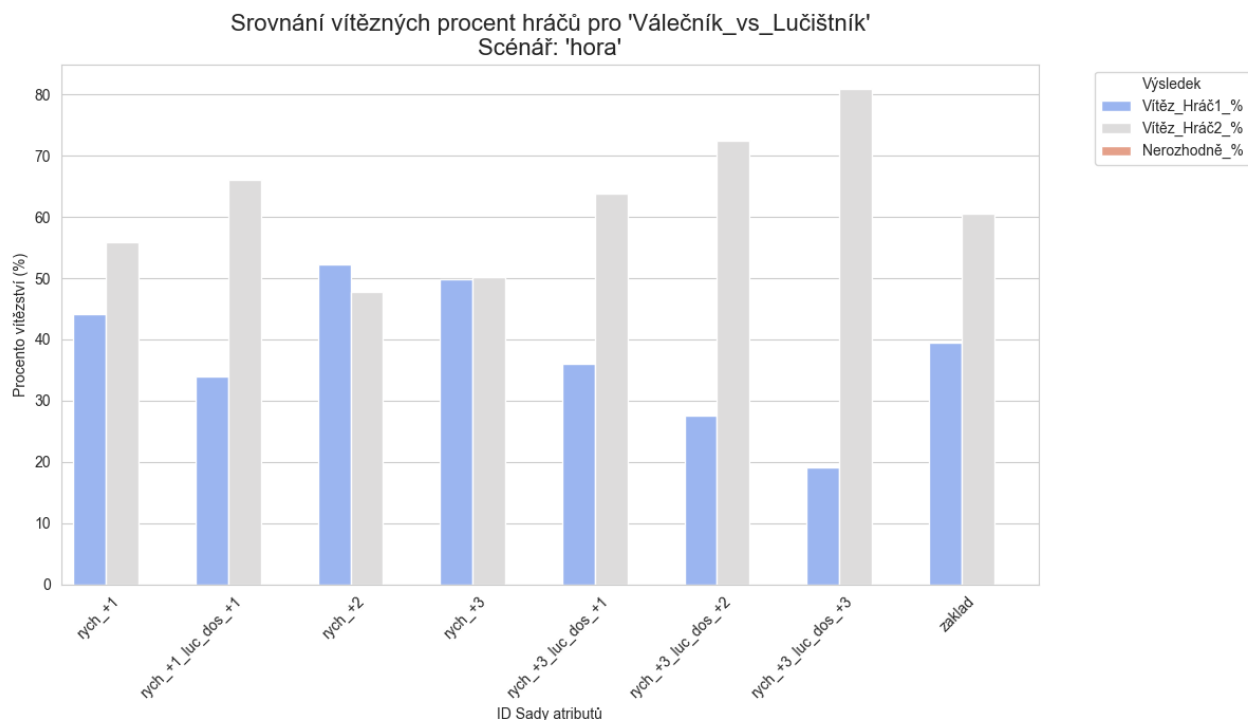
Tabulka 5.1: Tabulka sad atributů upravující rychlost souboje Válečníka a Lucištníka

| Scénář: Rovina | | | Scénář: Hora | |
|----------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| ID | Poměr vítězství Hráče 1 [%] | Počet kol (průměr, min–max) | Poměr vítězství Hráče 1 [%] | Počet kol (průměr, min–max) |
| 1 | 47.0 | 9.76 (6–22) | 39.5 | 15.22 (8–37) |
| 2 | 51.4 | 7.66 (5–14) | 44.2 | 13.48 (6–27) |
| 3 | 49.4 | 7.47 (5–14) | 34.0 | 13.45 (6–30) |
| 4 | 53.8 | 6.54 (5–12) | 52.2 | 10.00 (5–25) |
| 5 | 55.5 | 5.67 (4–12) | 49.9 | 9.95 (5–23) |
| 6 | 53.1 | 5.56 (4–10) | 36.1 | 10.14 (5–23) |
| 7 | 50.0 | 5.41 (4–10) | 27.5 | 9.94 (5–23) |
| 8 | 47.6 | 5.32 (4–9) | 19.1 | 9.71 (5–22) |

Tabulka 5.2: Tabulka výsledků po úpravách rychlosti pohybu Válečníka a Lučištníka
Hráč 1 ovládá jednotku Válečník a Hráč 2 má jednotku Lučištník.



Obrázek 5.2: Poměr vítězství duelu Válečníka a Lučištníka na rovině.



Obrázek 5.3: Poměr vítězství duelu Válečníka a Lučištníka na horské mapě.

Další simulace odrážející se od výsledků z tabulky 5.1, které jsou zopakovány v tabulce 5.3 pod ID 1, už se zaměřují na vyvažování ostatních atributů jednotek za účelem prozkoumat, jak úpravy atributů ovlivňují výsledky simulací, tak abych se dosáhlo vyvážení základních jednotek.

Tabulka 5.3 ukazuje testované varianty atributů. V tabulce 5.4 je zřejmé, že největší fluktuace výsledků způsobuje úprava *obrany* jednotek. Naopak rovnoměrné rozšíření intervalu poškození vedlo k téměř nulovým změnám v poměru výsledků a k minimální změně trvání hry. Na grafech 5.4 a 5.5 jsou pak znovu vidět poměry vítězství u jednotlivých úprav atributů.

Jako nejvyváženější úprava se nakonec ukázala varianta snížení *rychlosti* Lučištníka o jedna, což vedlo k drobnému zvýšení úspěšnosti Válečníka v obou terénech, tak, že se síly jednotek v tomto stavu dají považovat za vyvážené, s tím, že Lučištník více benefituje z komplexního terénu. Stále existující nevyváženost výsledků bude kompenzována tím, že jednotka Lučištník bude dražší na naverbování.

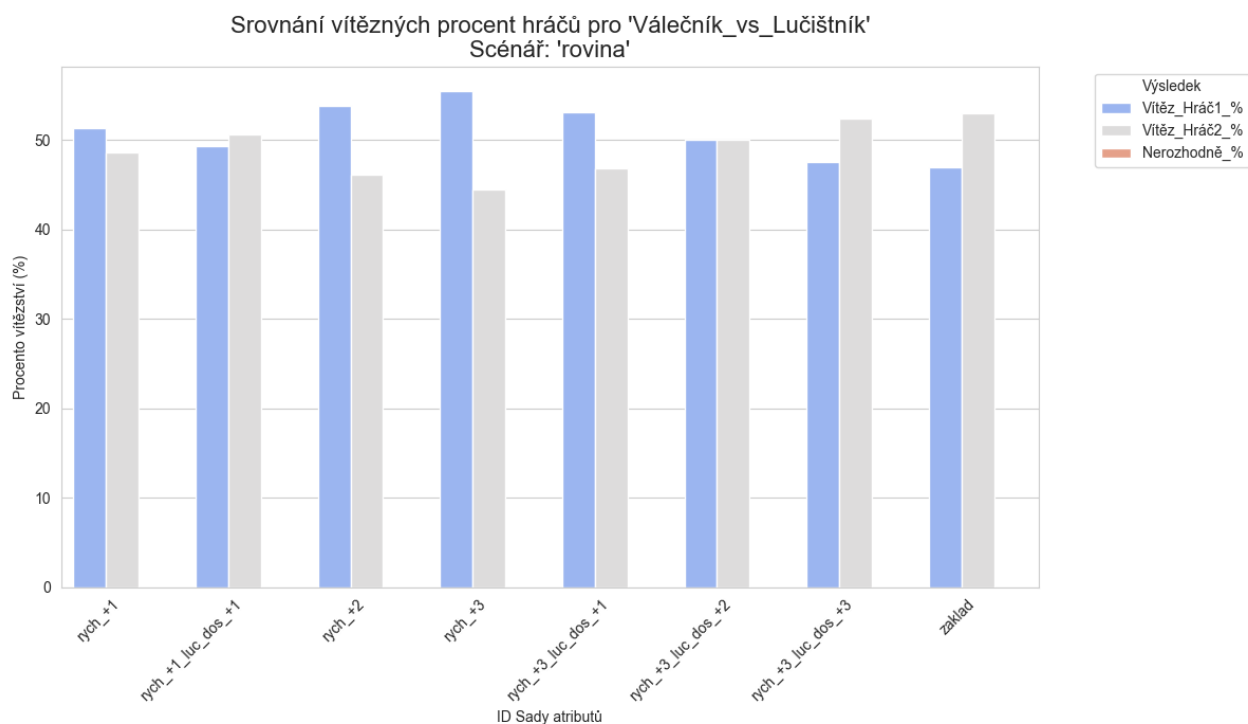
Kompletní data získaná z této simulace jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Duely/2_uroven/Val_Luc`.

| Sada atributů | ID | Typ | Útok Min | Útok Max | Obrana | Rychlost | Dosah | Krit. | Úhyb | Životy |
|---------------------------------|----|----------|----------|----------|--------|----------|-------|-------|------|--------|
| zaklad2 | 1 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| zaklad2 | 1 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_minAtk_+1 | 2 | valecnik | 7 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_minAtk_+1 | 2 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_maxAtk_+1 | 3 | valecnik | 6 | 9 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_maxAtk_+1 | 3 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_minAtk_-1_maxAtk_+1 | 4 | valecnik | 5 | 9 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_minAtk_-1_maxAtk_+1 | 4 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_minAtk_+1 | 5 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_minAtk_+1 | 5 | lucisnik | 7 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_minAtk_-1 | 6 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_minAtk_-1 | 6 | lucisnik | 5 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_maxAtk_+1 | 7 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_maxAtk_+1 | 7 | lucisnik | 6 | 9 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_minAtk_-1_maxAtk_+1 | 8 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_minAtk_-1_maxAtk_+1 | 8 | lucisnik | 5 | 9 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_def_-1 | 9 | valecnik | 6 | 8 | 3 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_def_-1 | 9 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_def_+1 | 10 | valecnik | 6 | 8 | 5 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_def_+1 | 10 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_def_-1 | 11 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_def_-1 | 11 | lucisnik | 6 | 8 | 2 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_def_+1 | 12 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_def_+1 | 12 | lucisnik | 6 | 8 | 4 | 6 | 5 | 1.2 | 0.05 | 10 |
| val_luc_minAtk_-1_maxAtk_+1 | 13 | valecnik | 5 | 9 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| val_luc_minAtk_-1_maxAtk_+1 | 13 | lucisnik | 5 | 9 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_minAtk_-1_val_luc_maxAtk_+1 | 14 | valecnik | 6 | 9 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_minAtk_-1_val_luc_maxAtk_+1 | 14 | lucisnik | 5 | 9 | 3 | 6 | 5 | 1.2 | 0.05 | 10 |
| luc_hp_-1 | 15 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_hp_-1 | 15 | lucisnik | 6 | 8 | 3 | 6 | 5 | 1.2 | 0.05 | 9 |
| luc_rych_-1 | 16 | valecnik | 6 | 8 | 4 | 6 | 1 | 1.2 | 0.05 | 15 |
| luc_rych_-1 | 16 | lucisnik | 6 | 8 | 3 | 5 | 5 | 1.2 | 0.05 | 10 |

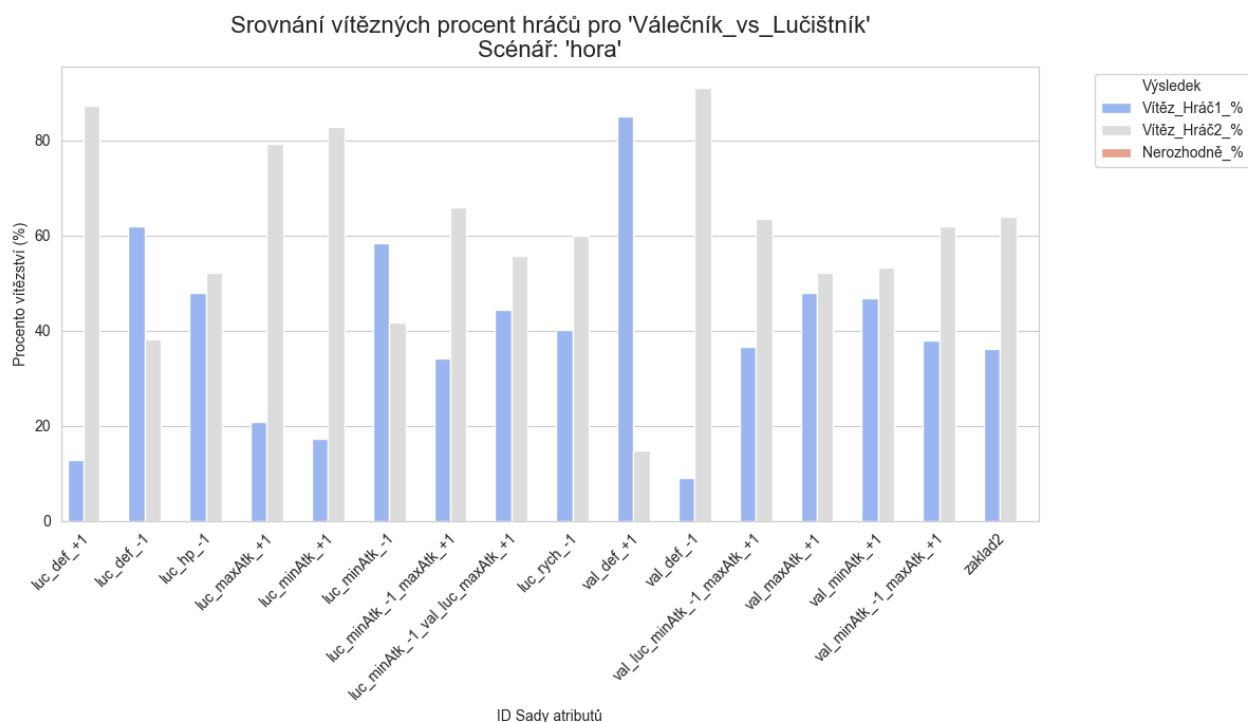
Tabulka 5.3: Tabulka sad atributů souboje Válečníka a Lucíštníka

| Scénář: Rovina | | | Scénář: Hora | |
|----------------|----------------------------------|--------------------------------|----------------------------------|--------------------------------|
| ID | Poměr vítězství Hráče 1 { % } | Počet kol (průměr, min–max) | Poměr vítězství Hráče 1 { % } | Počet kol (průměr, min–max) |
| 1 | 53.1 | 5.56 (4–10) | 36.1 | 10.14 (5–23) |
| 2 | 62.2 | 5.44 (4–10) | 46.7 | 10.09 (5–23) |
| 3 | 64.7 | 5.41 (4–10) | 47.9 | 10.06 (5–23) |
| 4 | 52.0 | 5.49 (4–10) | 38.0 | 10.11 (5–23) |
| 5 | 33.2 | 5.25 (4–10) | 17.2 | 9.36 (5–21) |
| 6 | 75.6 | 5.86 (4–12) | 58.4 | 10.52 (5–28) |
| 7 | 35.1 | 5.26 (3–11) | 20.7 | 9.36 (5–22) |
| 8 | 53.9 | 5.54 (4–10) | 34.1 | 9.96 (5–26) |
| 9 | 22.6 | 5.03 (3–9) | 9.0 | 8.89 (5–21) |
| 10 | 90.3 | 6.07 (4–10) | 85.1 | 11.16 (6–32) |
| 11 | 79.3 | 5.23 (4–9) | 61.8 | 9.93 (5–23) |
| 12 | 24.7 | 5.69 (4–10) | 12.8 | 10.24 (5–23) |
| 13 | 52.9 | 5.48 (3–10) | 36.6 | 9.92 (5–26) |
| 14 | 64.4 | 5.40 (3–10) | 44.3 | 9.83 (5–26) |
| 15 | 64.3 | 5.41 (4–10) | 47.9 | 10.07 (5–23) |
| 16 | 56.0 | 6.09 (4–11) | 40.2 | 9.09 (5–24) |

Tabulka 5.4: Tabulka výsledků simulací soubojů Válečníka a Lučištníka



Obrázek 5.4: Poměr vítězství duelu Válečníka a Lučištníka na rovině.



Obrázek 5.5: Poměr vítězství duelu Válečníka a Lučištníka na horské mapě.

Duely Bojovník proti Válečníkovi a Lučištníkovi

Vyvážené jednotky Válečník a Lučištník, jsou využity jako odrazový můstek pro zbytek simulací. Tato sekce se zaměřuje na vyvážení základní jednotky Bojovník proti jejím pokročilým variantám, Válečníkovi a Lučištníkovi. Jelikož jsou Válečník i Lučištník přímý vývoj Bojovníka, předpokládá se, že by měli být dominantní, na druhou stranu je potřeba, aby existovala šance na vítězství, proto se atributy Bojovníka upraví tak, aby se jeho úspěšnost pohybovala v rozmezí 10-20 %.

Základní atributy Bojovníka, jež byly použity jako výchozí bod, jsou specifikovány v tabulce 5.5.

| Útok Min | Útok Max | Obrana | Rychlost | Dosah | Krit. | Úhyb | Životy |
|----------|----------|--------|----------|-------|-------|------|--------|
| 5 | 8 | 3 | 5 | 1 | 1.2 | 0.05 | 12 |

Tabulka 5.5: Tabulka základní atributů Bojovníka před začátkem simulací.

Tabulka 5.6 ukazuje, že většina úprav atributů nevedla k zásadním změnám v úspěšnosti bojovníka. Z dat vyplývá, že největší vliv mělo zvýšení *životů* a zvýšení *obranu*. Zvýšení životů na patnáct (ID: *boj.hp_+3*) je těsně pod požadovanou úrovní úspěšnosti, zvýšení životů o jeden nebo dva by pravděpodobně vedlo k ideální úspěšnosti. Na druhou stranu by pak Bojovník měl nejen více životů než Lučištník, ale také více než Válečník. Což z designového hlediska nedává smysl.

Z toho důvodu je nejlepší variantou sada se zvýšenou obranou Bojovníka (*boj.def_+1*). Tedy do hry budou použity atributy v tabulce 5.7.

| | Válečník: | | | | Lučičtník: | | | | |
|---------------|---------------------------|-----------------------|---------------------------|-----------------------|---------------------------|-----------------------|---------------------------|-----------------------|--|
| | Rovina | | | Hora | Rovina | | | Hora | |
| ID | Vítězství Bojovník [%] | Průměrný počet kol | Vítězství Bojovník [%] | Průměrný počet kol | Vítězství Bojovník [%] | Průměrný počet kol | Vítězství Bojovník [%] | Průměrný počet kol | |
| zaklad | 1.8 | 5.43 | 4.4 | 8.4 | 5.0 | 5.34 | 3.2 | 8.46 | |
| boj_uhyb_0.1 | 2.8 | 5.53 | 6.5 | 8.53 | 7.7 | 5.44 | 4.1 | 8.6 | |
| boj_rych_+1 | 1.1 | 4.99 | 7.0 | 9.23 | 5.0 | 5.02 | 3.7 | 7.68 | |
| boj_minAtk_+1 | 3.6 | 5.41 | 10.1 | 8.37 | 8.1 | 5.32 | 6.1 | 8.45 | |
| boj_hp_+3 | 5.7 | 5.85 | 10.9 | 8.92 | 13.1 | 5.69 | 7.5 | 9.02 | |
| boj_def_+1 | 9.3 | 5.97 | 18.3 | 9.63 | 16.6 | 5.83 | 13.1 | 9.47 | |
| boj_atk_-1+1 | 3.6 | 5.41 | 7.7 | 8.39 | 8.8 | 5.32 | 6.0 | 8.44 | |

Tabulka 5.6: Tabulka průměrných výsledků soubojů Bojovníka.

| Útok Min | Útok Max | Obrana | Rychlost | Dosah | Krit. | Úhyb | Životy |
|----------|----------|--------|----------|-------|-------|------|--------|
| 5 | 8 | 4 | 5 | 1 | 1.2 | 0.05 | 12 |

Tabulka 5.7: Tabulka atributů Bojovníka po dokončení simulací.

| | Útok Min | Útok Max | Obrana | Rychlost | Dosah | Krit. | Úhyb | Životy |
|--------------|----------|----------|--------|----------|-------|-------|------|--------|
| barbar | 8 | 12 | 3 | 7 | 1 | 2.5 | 0.1 | 15 |
| rytir | 7 | 10 | 6 | 4 | 1 | 1.5 | 0.01 | 20 |
| lovec | 6 | 8 | 2 | 8 | 4 | 2.0 | 0.2 | 12 |
| ostrostrelec | 8 | 10 | 2 | 6 | 8 | 1.5 | 0.05 | 10 |

Tabulka 5.8: Tabulka základních atributů jednotek třetí úrovně.

Tímto nastavením Bojovník dosáhl dostatečné odolnosti, která mu zajišťuje minimální šanci na vítězství proti svým pokročilejším formám, a to bez narušení celkové vyváženosti herních jednotek.

Kompletní data získaná z této simulace jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Duely/1_uroven`.

Simulace jednotek třetí vývojové úrovně

Po vyvážení jednotek první a druhé vývojové úrovně se tato část zaměřuje na simulace s cílem vyvážit jednotky třetí úrovně, a to jak vzájemně mezi sebou, tak i ve vztahu k jednotkám druhé úrovně.

Třetí úroveň obsahuje čtyři jednotky: Rytíře, Barbaru, Ostrostřelce a Lovce. Vzhledem k vyššímu počtu jednotek a jejich odlišnému zaměření, se na rozdíl od druhé úrovně nepředpokládá dosažení plné vzájemné vyváženosti. Místo toho se očekává, že odlišné rozložení atributů povede k asymetrickému rozložení sil (tj. existence silnějších či slabších jednotek vůči konkrétním protivníkům), což bude vyžadovat specifické strategické přístupy pro každou jednotku.

Zároveň je nezbytné testovat tyto změny proti jednotkám z druhé vývojové úrovně, aby i slabší jednotky měly teoretickou šanci na vítězství. V tomto kontextu je cílová úspěšnost v rozmezí 10–30 %.

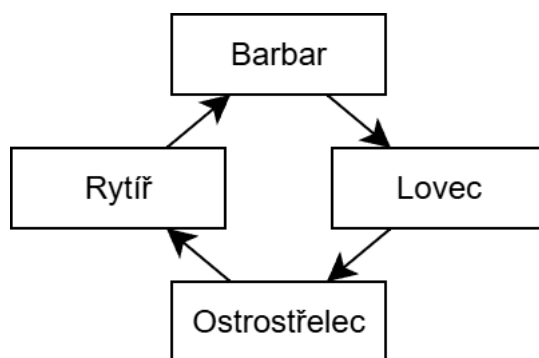
Základní atributy jednotek jsou uvedeny v tabulce 5.8. Předpokládané silné a slabé stránky jednotlivých jednotek jsou vizualizovány na obrázku 5.6.

Tabulka 5.9 zobrazuje poměr vítězství duelu mezi jednotkami na třetí úrovni, procenta určují poměr vítězství první jednotky ve sloupci.

Základní atributy (tabulka 5.9; ID: 1.) se ukázaly příliš vysoké ve srovnání s jednotkami z druhé úrovně. Proto byl prvním krokem návrh na mírné oslabení všech jednotek.

| Vítězství Hráč 1 [%] | Barbar Lovec | | Barbar Ostrostřelec | | Ostrostřelec Lovec | | Rytíř Barbar | | Rytíř Lovec | | Rytíř Ostrostřelec | |
|-------------------------|--------------|------|---------------------|------|--------------------|------|--------------|------|-------------|------|--------------------|------|
| | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora |
| 1. | 62.6 | 57.0 | 44.5 | 14.1 | 20.6 | 22.6 | 85.3 | 84.6 | 99.9 | 99.4 | 58.5 | 34.3 |
| 2. | 51.8 | 31.2 | 31.7 | 14.6 | 28.5 | 33.5 | 70.2 | 70.8 | 91.6 | 78.8 | 25.6 | 10.3 |
| 3. | 52.5 | 33.0 | 38.2 | 17.3 | 28.5 | 34.4 | 57.5 | 52.2 | 91.6 | 78.4 | 25.6 | 10.3 |
| 4. | 64.3 | 38.8 | 43.4 | 21.8 | 28.5 | 34.4 | 43.2 | 41.8 | 87.8 | 69.9 | 16.9 | 6.4 |
| 5. | 53.1 | 26.8 | 27.7 | 12.1 | 23.7 | 30.7 | 65.4 | 64.5 | 92.9 | 76.7 | 21.1 | 8.0 |
| 6. | 60.3 | 34.5 | 27.7 | 12.1 | 22.6 | 22.3 | 65.4 | 64.5 | 95.8 | 84.9 | 21.1 | 8.0 |

Tabulka 5.9: Tabulka výsledků duelů jednotek třetí úrovně.

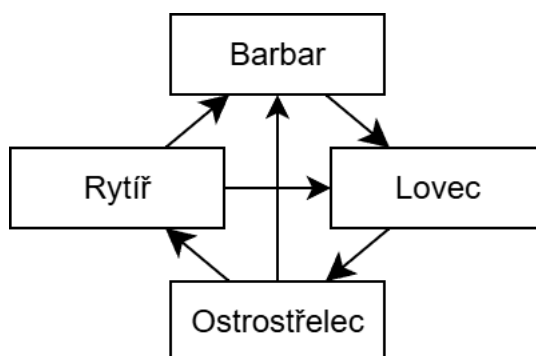


Obrázek 5.6: Diagram ideálního rozložení sil jednotek třetí úrovně mezi sebou.

Po oslabení jednotek se ukázalo, že nebude možné upravit atributy tak, aby byly síly jednotek specificky účinné pouze proti jedné konkrétní jednotce stejné úrovně. Přesto se po dalších úpravách atributů podařilo dosáhnout výsledků, které jsou přijatelné (ID: 6.).

Oproti původní představě došlo k rozdělení jednotek třetí úrovně na silnější a slabší. Konkrétně Barbar a Lovec se ukázali jako slabší, respektive vysoce specializované jednotky, jejichž silné stránky nejsou plně využitelné primitivní herní umělou inteligencí v simulacích. Naopak jednotky Rytíř a Ostrostřelec se prokázaly jako efektivnější, k čemuž přispívá jejich snadnější optimální využití. Výsledný diagram silových vztahů je znázorněn na obrázku 5.7.

Rozdíl v síle bude kompenzován úpravou ekonomických parametrů; výkonnější jednotky budou mít vyšší pořizovací náklady i náklady na údržbu. Předpokládá se, že toto opatření motivuje hráče k delšímu shromažďování surovin pro verbování silnějších jednotek a zároveň je povede k efektivnějšímu využívání méně výkonných jednotek.



Obrázek 5.7: Diagram výsledného rozložení sil jednotek třetí úrovně mezi sebou.

Tabulka 5.10 ukazuje, že úpravy atributů vedly k minimálním změnám v délce soubojů. Souboje silnějších jednotek jsou zároveň obecně kratší (výjimkou jsou souboje Rytíře, které trvají srovnatelně dlouho jako souboje slabších jednotek).

Kompletní data získaná z této simulace jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Duely/3_uroven`.

| Průměrný počet kol | Barbar Lovec | Barbar | | Ostrostřelec | | Ostrostřelec Lovec | | Rytíř Barbar | | Rytíř Lovec | | Rytíř Ostrostřelec | |
|--------------------|--------------|-------------|---------------|--------------|---------------|--------------------|---------------|--------------|---------------|-------------|---------------|--------------------|---------------|
| | | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina |
| ID | | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina | Hora | Rovina |
| 1. | 4.31 | 7.04 | 3.78 | 5.53 | 3.29 | 3.95 | 5.41 | 8.07 | 5.96 | 10.28 | 7.62 | 11.15 | 11.15 |
| 2. | 4.39 | 7.72 | 4.82 | 6.37 | 3.35 | 4.01 | 5.7 | 8.89 | 5.94 | 10.77 | 7.19 | 10.43 | 10.43 |
| 3. | 4.4 | 7.7 | 4.8 | 6.36 | 3.35 | 4.02 | 5.64 | 8.77 | 5.94 | 10.78 | 7.19 | 10.43 | 10.43 |
| 4. | 4.2 | 7.66 | 4.78 | 6.35 | 3.35 | 4.02 | 5.54 | 8.66 | 5.87 | 10.57 | 6.84 | 9.88 | 9.88 |
| 5. | 3.91 | 7.04 | 4.32 | 5.75 | 3.32 | 3.99 | 5.38 | 8.49 | 5.74 | 10.52 | 6.96 | 10.09 | 10.09 |
| 6. | 3.85 | 6.98 | 4.32 | 5.75 | 3.3 | 3.89 | 5.38 | 8.49 | 5.54 | 10.16 | 6.96 | 10.09 | 10.09 |

Tabulka 5.10: Tabulka průměrné délky duelů jednotek třetí úrovně.

5.4 Fáze 2: Testování náhodného herního pole

Tato fáze se zaměřuje na testování funkčnosti procedurálně generovaných náhodných map. Je nutné ověřit, zda jsou náhodně vygenerované mapy pro hru vhodné a použitelné. Současně je zde provedena vizuální kontrola vybraných map, zda vypadají přirozeně.

Cílem této fáze je validace funkčnosti generátoru map, se zaměřením na tři hlavní oblasti:

- Rozložení terénů: Dochází k analýze procentuálního zastoupení jednotlivých typů terénu (voda, pláně, lesy, hory) na generovaných mapách. To pomůže posoudit, jak se rozložení terénů a s nimi spojených zdrojů mění a zda je dostatečně variabilní.
- Průchodnost a konektivita: Ověření průchodnosti map. Jsou zkoumány cesty mezi náhodně vybranými body na mapě za účelem detekce případného generování neprůchodných konfigurací.
- Vizuální inspekce: Kromě sběru kvantitativních dat je také provedena vizuální kontrola map s cílem posoudit jejich estetiku, přirozenost a celkovou strategickou zajímavost. Tato inspekce je klíčová pro odhalení problémů, které nemusí být zjevné z číselných dat.

5.4.1 Metodika

Postup byl následující:

1. Definice ovladatelných parametrů:

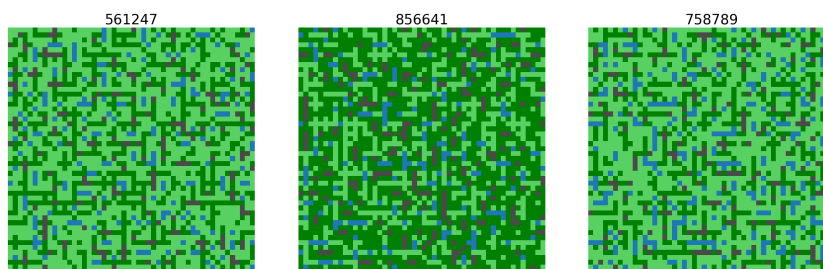
- rozměr herního pole,
- měřítko šumu (`scale`), které ovládá velikost náhodného gradientního pole, podle kterého se pak mapa generuje
- a prahové hodnoty, které definují rozdělení terénních typů.

2. Generování tří map pro vizuální inspekci:

Byly vygenerovány tři náhodné mapy s danými parametry. Tyto mapy byly následně posouzeny z hlediska vizuální přirozenosti, herní zajímavosti a potenciální balanční spravedlnosti.

3. Generování map pro kvantitativní kontrolu:

Pokud vygenerované mapy prošly vizuální kontrolou (nebyl s nimi zásadní problém), vygenerovalo se tisíc map, ke každé se uložil poměr množství políček jednotlivých typů. Dále byl pro každou mapu náhodně vygenerováno sto dvojic startovních a cílových bodů, mezi nimiž se zaznamenala úspěšnost nalezení cesty.



Obrázek 5.8: Mapy vygenerované pro vizuální hodnocení. (`scale` = 2)

5.4.2 Testování měřítka šumu

Nejprve byly testovány krajní hodnoty parametru `scale`, pak se testovaly případy mezi extrémy a na těch nejlepších budou testovány změny prahových hodnot.

Jako počáteční nastavení byly použity prahové hodnoty:

- Voda: $[0 - 0.25)$
- Pláně: $[0.25 - 0.5)$
- Lesy: $[0.5 - 0.75)$
- Hory: $[0.75 - 1]$

Malé `scale`

Malé měřítko šumu (`scale`) znamená, že základní náhodná mřížka, z níž je odvozena výsledná mapa, je srovnatelná s rozměrem samotné mapy. To vede k generování malých a fragmentovaných struktur.

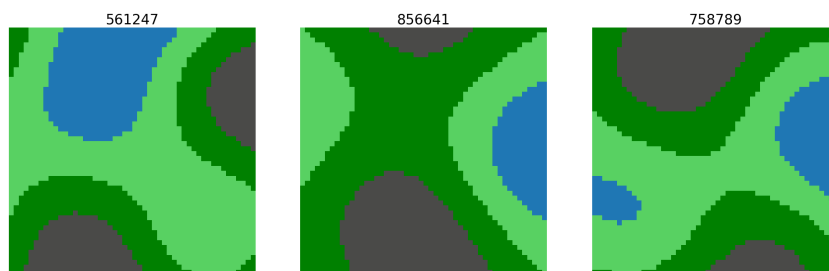
Jako mezní hodnota pro malé měřítko šumu byl zvolen parametr `scale` = 2 pro mapu o rozměrech 50×50 polí.

Na obrázku 5.8 je vidět, že struktury jsou tak malé, že mapy vypadají jako náhodné, tudíž není důvod s těmito parametry pokračovat.

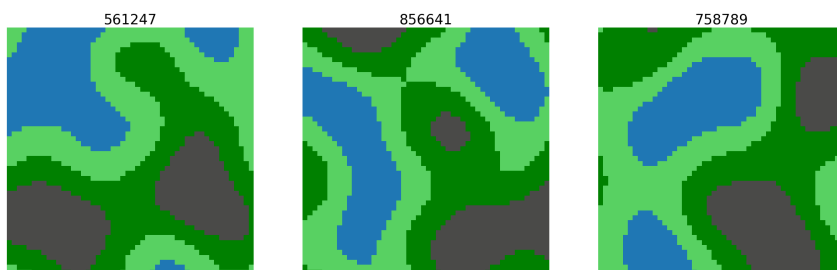
Velké `scale`

Naopak, velké měřítko šumu (`scale`) vede k výraznému zmenšení základní náhodné mřížky. Následná interpolace mezi takto omezeným počtem referenčních bodů vytváří rozsáhlé a souvislé struktury. Jako velké `scale` bylo použito `scale` = 50 na mapě 50×50 .

Na obrázku 5.9 je vidět, že výsledný terén vypadá relativně přirozeně, ale z herního hlediska není strategicky zajímavý. I přes to, že se nejedná o optimální výsledek, byl pro toto nastavení proveden kvantitativní sběr dat.



Obrázek 5.9: Mapy vygenerované pro vizuální hodnocení. (`scale` = 50)



Obrázek 5.10: Mapy vygenerované pro vizuální hodnocení. (`scale` = 25)

Střední `scale`

Velké měřítko šumu se ukázalo být výrazně vhodnější variantou. Proto byl jako další testovací parametr zvolen největší celočíselný dělitel rozměru herní mapy, což pro mapu 50×50 znamená `scale` = 25.

Na obrázku 5.10 je vidět, že výsledný terén opět vypadá poměrně přirozeně, ale navíc už má určitou úroveň komplexity, která jej činí vhodným pro využití jako herní mapa.

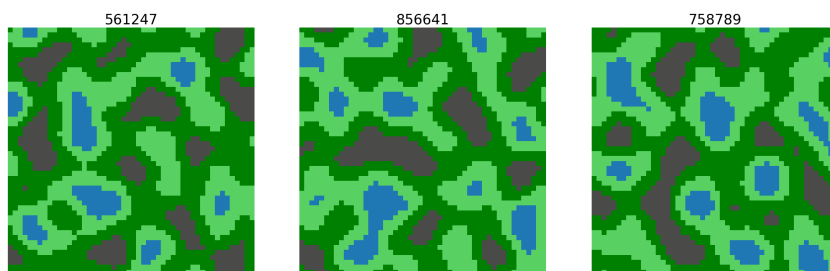
Další varianta `scale`

Zmenšení `scale` vedlo ke zvýšení komplexity, otázka je, zda další zmenšení nepovede k chaotické mapě, podobně jako tomu bylo u malého `scale`. Další celočíselné dělení mapy 50×50 je `scale` = 10.

Výsledné mapy, jak ilustruje obrázek 5.11, jsou podle očekávání mnohem komplexnější, což je lepší pro hru ze strategického hlediska. Nicméně, mapa začíná působit poněkud fragmentovaně, z toho důvodu menší měřítko šumu již nebylo testováno.

Zvětšená mapa

Vzhledem k tomu, že měřítko šumu `scale` = 25 poskytuje nedostatečnou herní komplexitu a `scale` = 10 vede k chaotickým, respektive neorganickým strukturám,



Obrázek 5.11: Mapy vygenerované pro vizuální hodnocení. (`scale` = 10)



Obrázek 5.12: Mapy vygenerované pro vizuální hodnocení. (`scale` = 20)

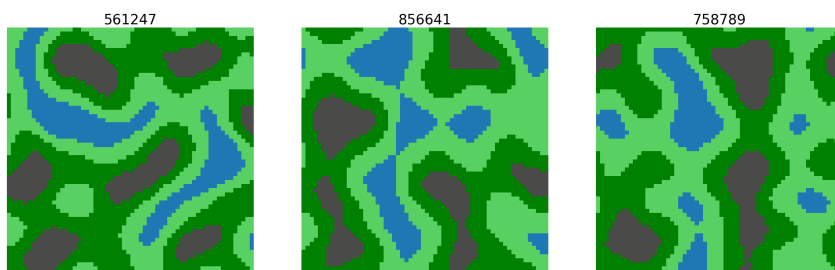
bylo přistoupeno k úpravě rozměrů mapy. Cílem je umožnit použití hodnoty měřítka šumu mezi tím. Pro tento účel bylo zvoleno `scale` = 20 na mapě 60×60 .

Výsledné mapy, na obrázku 5.12, jsou dle očekávání velmi podobné mapám s měřítkem 25. Nicméně, díky vyšší komplexitě struktur působí přirozeněji.

Menší `scale` na větší mapě

V rámci závěrečné simulace bylo přistoupeno k další redukci měřítka šumu. Konkrétně bylo použito `scale` = 15 na mapě 60×60 .

Mapy na obrázku 5.13 působí velmi organicky a jsou strategicky dostatečně komplexní, aniž by působily fragmentovaně. Tedy z vizuálního hlediska je tato varianta



Obrázek 5.13: Mapy vygenerované pro vizuální hodnocení. (`scale` = 15)

| | Voda[%] | Pláně[%] | Les[%] | Hory[%] | Existující cesty [%] |
|-----------------------------|---------|----------|--------|---------|----------------------|
| scale=50 | 21.033 | 28.818 | 28.610 | 21.539 | 95.783 |
| scale=25 | 16.662 | 33.663 | 33.228 | 16.447 | 96.335 |
| scale=10 | 11.567 | 38.707 | 38.344 | 11.382 | 99.734 |
| 60 x 60 scale=20 | 13.471 | 36.481 | 36.673 | 13.375 | 98.658 |
| 60 x 60 scale=15 | 12.861 | 37.138 | 37.222 | 12.780 | 99.289 |

Tabulka 5.11: Tabulka průměrných výsledků získaných generováním náhodných map s různou velikostí **scale**.

považována za optimální.

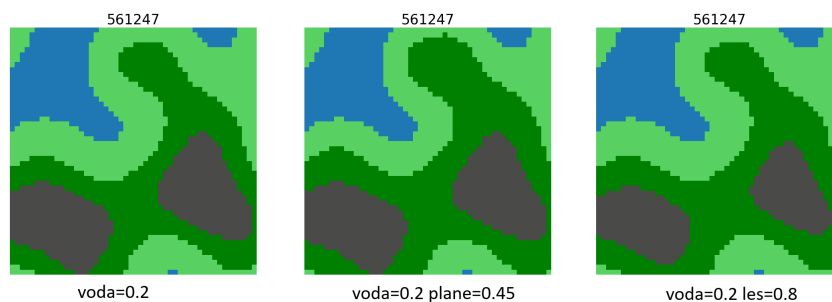
Výsledky simulací změn **scale**

Testy ukázaly, že měřítko šumu (**scale**) výrazně ovlivňuje podobu generovaných map. Příliš malé měřítko (**scale** = 2) vedlo k chaotickým fragmentovaným mapám, příliš velké (**scale** = 50) sice působilo přirozeně, ale postrádalo strategickou hloubku.

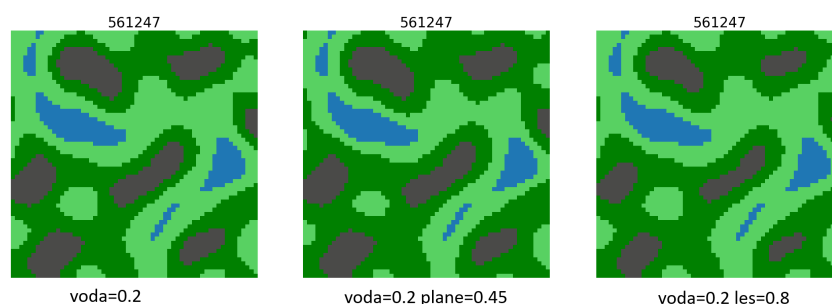
Optimální výsledky na mapě 50×50 poskytlo **scale** = 25. Při větší mapě 60×60 se jako nejvhodnější ukázalo **scale** = 15, které dosahuje nejlepší rovnováhy mezi vizuální přirozeností, komplexitou a průchodností.

Průměrná průchodnost každého simulovaného nastavení je prezentována v tabulce 5.11. Z dat je patrné, že s klesající hodnotou **scale** průchodnost narůstá, což bylo neočekávané zjištění. Pro další testování různých prahových hodnot byly použity mapy s **scale**=25 a **scale**=15.

Kompletní data získaná z těchto simulací jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Mapy/map_log_scale`.



Obrázek 5.14: Ukázka map po úpravě hraničních hodnot. (`scale` = 25)



Obrázek 5.15: Ukázka map po úpravě hraničních hodnot. (`scale` = 15)

5.4.3 Testování různých prahových hodnot

Po identifikaci optimálních měřítek šumu (`scale`), konkrétně `scale` = 25 pro mapu 50×50 a `scale` = 15 pro mapu 60×60 , se další fáze zaměřila na testování prahových hodnot. Ty přímo ovlivňují poměrné zastoupení jednotlivých terénních typů na generované mapě. Cílem je doladit distribuci vody, plání, lesů a hor tak, aby výsledné mapy byly strategicky vyvážené a herně zajímavé.

Úprava prahových hodnot nepovede k zásadním vizuálním změnám, pouze upravuje poměry jednotlivých typů terénu, proto byly změny spíše minimální. Cílem bylo zvětšit prostor, na kterém se mohou hráči pohybovat, ale zachovat komplexnost terénu, aby bylo strategické manévrování stále relevantní.

Jako výchozí bod pro úpravy prahových hodnot byly použity následující:

- Voda: $[0 - 0.25)$
- Pláně: $[0.2 - 0.5)$
- Lesy: $[0.5 - 0.75)$
- Hory: $[0.75 - 1]$

Tabulka 5.12 potvrzuje očekávání, že snížení prahové hodnoty pro vodu zvyšuje úspěšnost průchodů mapy. Ačkoliv je zvýšení průchodnosti žádoucí, současně vede k poklesu komplexnosti terénu, neboť eliminuje zcela neprůchozí bariéry. Z tohoto důvodu nelze prahovou hodnotu pro vodu zásadně redukovat.

| | Voda[%] | Pláně[%] | Les[%] | Hory[%] | Existující cesty [%] |
|---|---------|----------|--------|---------|----------------------|
| 50x50 voda=0.2 | 11.846 | 38.479 | 33.228 | 16.447 | 98.524 |
| 60x60 voda=0.2 | 8.057 | 41.988 | 37.771 | 12.184 | 99.809 |
| 50x50 voda=0.2 plan=0.45 | 11.846 | 31.080 | 40.627 | 16.447 | 98.524 |
| 60x60 voda=0.2 plan=0.45 | 8.057 | 33.313 | 46.446 | 12.184 | 99.809 |
| 50x50 voda=0.2 les=0.8 | 11.846 | 38.479 | 37.979 | 11.695 | 98.524 |
| 60x60 voda=0.2 les=0.8 | 8.057 | 41.988 | 41.942 | 8.013 | 99.809 |

Tabulka 5.12: Tabulka průměrných výsledků získaných generováním náhodných map s různými prahovými hodnotami.

Na obrázcích 5.14 a 5.15 je vidět změna vybraných map při změně prahových hodnot. Po zhodnocení vizuálních výsledků a dat z kvantitativních simulací byla jako optimální varianta zvolena následující konfigurace:

- Voda: $[0 - 0.2)$
- Pláně: $[0.2 - 0.5)$
- Lesy: $[0.5 - 0.75)$
- Hory: $[0.8 - 1]$

Tato konfigurace maximalizuje průchozí herní prostor, usnadňuje pohyb jednotek, a tím zvyšuje efektivitu jednotek pro boj na blízko. Současně však zachovává dostatečnou terénní rozmanitost, což umožňuje hráčům strategické využití prostředí. Dále, snížení množství horských oblastí na mapě oddaluje získávání kritických zdrojů pro verbování finálních jednotek, což teoreticky podporuje využití jednotek nižších úrovní.

Kompletní data získaná z těchto simulací jsou dostupná v příloženém souboru ve složce `vysledky_simulace/Mapy/map_log_prahy`.

Závěr

Hlavním cílem výzkumného úkolu bylo prozkoumat a prakticky aplikovat principy procedurálního generování obsahu (PCG) pro tvorbu herních světů a následně navrhnout strategickou tahovou hru využívající náhodně vygenerované mapy.

V první kapitole jsem rozebral problematiku procedurálního generování obsahu (PCG), analyzoval jeho využití, výhody a nevýhody. Na základě detailního srovnání různých metod PCG vhodných pro generování náhodných herních map (např. dělení prostoru, fraktálové algoritmy či komplexnější metody jako wave function collapse) jsem jako ideální pro mé potřeby a omezení vybral metodu výškových map.

Ve druhé kapitole jsem se zaměřil na implementaci zvolené metody výškových map. Tu jsem implementoval ve třech variantách, a to konkrétně jako náhodnou výškovou mapu, interpolovanou výškovou mapu a gradientní výškovou mapu. Zároveň jsem v této kapitole zmínil, jak výsledné výškové mapy převádím na políčka s různými typy terénu, a pro účely prototypu jsem představil, jak tyto mapy vypadají.

Ve třetí kapitole jsem prostudoval teorii návrhu videoher a na základě zpracovaných znalostí jsem připravil návrh mé strategické hry. Specifikoval jsem herní prostor, objekty a jejich možné stavy a atributy, dále akce hráče, pravidla a cíle hry, a zmínil jsem důležitost dovednosti vs. náhody ve hře.

Ve čtvrté kapitole jsem implementoval prototyp obsahující základní herní mechaniky navržené strategické hry, včetně pohybu jednotek, správy zdrojů a bojového systému. Klíčovým prvkem byla také příprava jednoduché herní umělé inteligence, která se stala nezbytnou pro automatizované testování a systematickou optimalizaci v závěrečné fázi projektu.

V poslední, páté kapitole jsem využil implementovaný prototyp k simulacím herních scénářů. Konkrétně jsem se zaměřil na duely jednotek, aby jednotky s podobnou hodnotou měly srovnatelnou bojovou sílu. Pomocí metody Monte Carlo jsem optimalizoval atributy herních objektů, čímž jsem dosáhl vyvážení hratelnosti. Dále jsem stejnou metodou optimalizoval parametry pro generování náhodných map, což vedlo ke zvýšení průchodnosti map a zvýšení jejich strategické zajímavosti a variability.

Celkově tento výzkumný úkol poskytl komplexní vhled do procedurálního generování obsahu pro herní účely. Byly navrženy a implementovány systémy pro generování herního prostředí a základní herní mechaniky, jejichž vyvážení bylo empiricky ověřeno a optimalizováno pomocí simulačních metod. Výsledky práce poslouží jako vhodný základ pro další práci v rámci diplomové práce, kde se zaměřím na dokončení hry a využití strojového učení pro herní umělou inteligenci.

Literatura

- [1] SHAKER, Noor; TOGELIUS, Julian a NELSON, Mark J. *Procedural Content Generation in Games*. Switzerland: Springer International Publishing, 2016. ISBN 978-3-319-42714-0.
- [2] SMITH, Gillian. *An Analog History of Procedural Content Generation*. Paper. 360 Huntington Ave, 100 ME Boston, MA 02115, USA: Northeastern University, 2015.
- [3] MOJANG AB. TM MICROSOFT CORPORATION. *Minecraft*. Online. 2025. Dostupné z: <https://www.minecraft.net/en-us>. [cit. 2025-08-12].
- [4] HELLO GAMES. *No man's sky*. Online. [2016]. Dostupné z: <https://www.nomanssky.com/>. [cit. 2025-08-12].
- [5] BLIZZARD ENTERTAINMENT, INC. *Diablo*. Online. C2025. Dostupné z: <https://diablo4.blizzard.com/en-us/>. [cit. 2025-08-12].
- [6] GEARBOX SOFTWARE. *Borderlands*. Online. C2025. Dostupné z: <https://borderlands.2k.com/>. [cit. 2025-08-12].
- [7] *Perlin Noise*. Online. Scratchapixel.com. 2022. Dostupné z: <https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/perlin-noise-part-2/perlin-noise.html>. [cit. 2025-01-16].
- [8] *Introduction To Grammar in Theory of Computation*. Online. Geeksforgeeks.org. 2025. Dostupné z: <https://www.geeksforgeeks.org/theory-of-computation/introduction-to-grammar-in-theory-of-computation/>. [cit. 2025-08-13].
- [9] NYSTROM, Robert. *Spatial Partition*. Online. Gameprogrammingpatterns.com. C2009-2021. Dostupné z: <https://gameprogrammingpatterns.com/spatial-partition.html>. [cit. 2025-08-13].
- [10] ANTONIOS LIAPIS. *Constructive Generation Methods for Dungeons and Levels*. Online. Antoniosliapis.com. 2017. Dostupné z: https://antoniosliapis.com/articles/pcgbook_dungeons.php. [cit. 2025-01-13].
- [11] GEEKS FOR GEEKS. *Binary Space Partitioning*. Online. Wwww.geeksforgeeks.org. 2020, 30 Sep, 2020. Dostupné z: <https://www.geeksforgeeks.org/binary-space-partitioning/>. [cit. 2025-01-13].

- [12] DORAN, Jonathon a PARBERRY, Ian. Controlled Procedural Terrain Generation Using Software Agents. *ReasearchGate*. 2010, vol. 2, no. 2, s. 111 - 119.
- [13] *Procedural Location Generation with Weighted Attribute Grammars*. Bakalářská práce. University of Twente P.O. Box 217, 7500AE Enschede The Netherlands: University of Twente, 2021.
- [14] *Generative Grammars*. Online. Fandom.com. 2016. Dostupné z: https://procedural-content-generation.fandom.com/wiki/Generative_Grammars. [cit. 2025-01-15].
- [15] IVSON, Paulo; TOLEDO, Rodrigo a GATTASS, Marcelo. Solid height-map sets: modeling and visualization. Online. In: *ReasearchGate*. New York, USA: Stony Brook, 2008, s. 1-8. Dostupné z: <https://doi.org/10.1145/1364901.1364953>. [cit. 2025-08-13].
- [16] Travall. *Procedural 2D Island Generation - Noise Functions*. Online. <https://medium.com>. 2018. Dostupné z: <https://medium.com/@travall/procedural-2d-island-generation-noise-functions-13976bddeaf9>. [cit. 2025-01-16].
- [17] O'BRIEN, Nick. *Diamond-Square Algorithm Explanation and C++ Implementation*. Online. Medium.com. 2018. Dostupné z: <https://medium.com/@nickobrien/diamond-square-algorithm-explanation-and-c-implementation-5efa891e486f>. [cit. 2025-08-13].
- [18] BROWN, Adam. *The Maths of Fractal Landscapes and Procedural Landscape Generation*. Online. Fractal-landscapes.co.uk. 2002 - 2020. Dostupné z: <https://www.fractal-landscapes.co.uk/maths.html>. [cit. 2025-01-16].
- [19] *Wave Function Collapse Explained*. Online. Boristhebrave.com. 2020. Dostupné z: <https://www.boristhebrave.com/2020/04/13/wave-function-collapse-explained/>. [cit. 2025-08-13].
- [20] HEATON, Robert. *The Wavefunction Collapse Algorithm explained very clearly*. Online. Robertheaton.com. 17 Dec 2018. Dostupné z: <https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/>. [cit. 2025-01-16].
- [21] GOODFELLOW, Ian J.; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David et al. *Generative Adversarial Nets*. International Conference. Montreal: Universite de Montreal, 2014.
- [22] RADFORD, Alec a METZ, Luke. *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*. Paper. Boston, MA, 2016.
- [23] JETCHEV, Nikolay. *Texture Synthesis with Spatial Generative Adversarial Networks*. Paper. Barcelona, Spain: Zalando Research, 2017.

- [24] SPICK, Ryan J.; COWLING, Peter a WALKER, James Alfred. *Procedural Generation using Spatial GANs for Region-Specific Learning of Elevation Data*. Paper. University of York, UK: University of York, 2019.
- [25] *SciPy*. Online. C2025. Dostupné z: <https://scipy.org/>. [cit. 2025-08-14].
- [26] *Noise*. Online. 2015. Dostupné z: <https://pypi.org/project/noise/>. [cit. 2025-08-14].
- [27] *Matplotlib: Visualization with Python*. Online. C2012 — 2025. Dostupné z: <https://matplotlib.org/>. [cit. 2025-08-14].
- [28] ROGERS, Scott. *Level UP! The guide to great video game design*. John Wiley & Sons, 2010. ISBN 978-0-470-68867-0.
- [29] SCHELL, Jesse. *The art of game design: a book of lenses*. Boca Raton : CRC Press, 2008. ISBN 978-0-12-369496-6.
- [30] SALEN, Katie a ZIMMERMAN, Eric. *Rules of Play - Game Design Fundamentals*. 2. Massachusetts Institute of Technology, 2004. ISBN 0-262-24045-9.
- [31] BAJAJ, Aayush. *Monte Carlo Simulation: -A Hands-On Guide*. Online. Neptune.ai. 2023. Dostupné z: <https://neptune.ai/blog/monte-carlo-simulation>. [cit. 2025-08-12].

Příloha A

Přiložené soubory

`generator_map` Adresář obsahuje Python projekt s prototypem strategické hry a skripty pro simulace.

`vysledky_symulaci` Adresář obsahuje do jednotlivých složek roztríděné výsledky různých simulací. Konkrétní rozdělení výsledků je zmíněno v hlavním textu práce.