Plošinová Hra

Dokumentace maturitní práce

Matěj Jirsa

školní rok: 2023/2024 třída: 8. V

Zadání maturitní práce:

Student naprogramuje roguelike hru v herním enginu Unity, ve které je hráči svěřena kontrola nad jeho postavou a jeho úkolem je v herním světě přežít co nejdéle. Součástí projektu bude vytvoření vlastní grafiky do hry.

Teoretická část: Procedurální generace herních světů

Prohlašuji, že jsem na práci pracoval samostatně pouze za pomoci použitých zdrojů a že v práci i v dokumentaci jasně vymezuji, které části kódů jsou mým originálním dílem, které jsou upravenou verzí a které jsou převzaty v plném rozsahu.

---

podpis žáka

OBSAH

1. TEORETICKÁ ČÁST

Procedurální generace herních světů

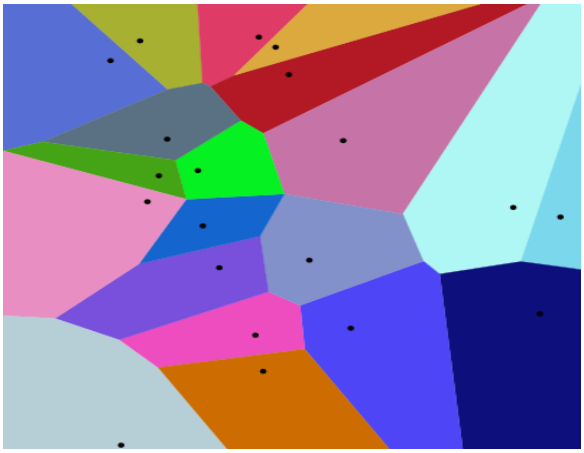
Procedurální generace herních světů je inovativní technika vytváření herních prostředí a obsahu prostřednictvím algoritmů a náhodných procesů. Tento přístup umožňuje vytvářet nekonečné množství různorodých prostředí bez nutnosti manuálního vytváření každého detailu ručně. V této práci se budeme zabývat především procedurální generací herních světů ve 2D rovině.

1.1. Princip procedurální generace herních světů

Procedurální generace herních světů využívá algoritmů k vytváření rozmanitých herních prostředí. Tento inovativní přístup umožňuje dynamické a neustále se měnící herní světy, což výrazně zvyšuje unikátnost a znovu-hratelnost her. Algoritmy používané v procedurální generaci mohou být založeny na různých principech, jako jsou Voroného diagramy, Fractal Noise nebo celulární automaty.

1.1.1. Voroného diagramy

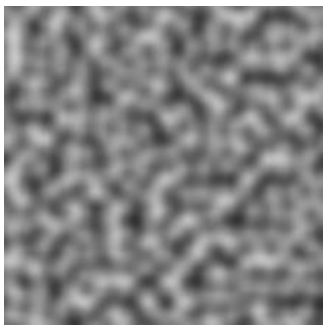
Voroného diagramy rozdělují prostor na buňky okolo zadaných bodů. Každý bod má buňku tvořenou oblastmi, které jsou k němu nejblíže. Vzniká tak mozaika s organickými tvary, která se dá využít v různých oblastech, například k vytváření realistických biomů.



Obrázek 1. Ukázka Voroného diagramu

1.1.2. Fractal Noise

Fractal Noise je matematický šum využívající fraktálů pro generování pseudonáhodných vzorců s multifraktálními vlastnostmi. To umožňuje vytvářet realistické detaily v různých měřítkách, což je vhodné pro terénní generaci, textury a atmosférické jevy v herním developmentu.



Obrázek 2. Perlinův Šum, jeden z často využívaných algoritmů

1.1.3. Celulární automaty

Celulární automaty (CA) jsou matematické modely, které simulují dynamické systémy na mřížce buněk. Každá buňka má konečný počet stavů a sousedství definované okolními buňkami. Pravidla CA definují, jak se stav buňky mění v čase v závislosti na stavech sousedních buněk. Jedním z ikonických příkladů CA je Conwayova hra života, která má simulovat život pouze za pomocí živých a mrtvých buněk, které žijí a umírají pouze v závislosti stavu okolních buněk.

Obrázek 3. Conwayova hra života

1.2. Výhody a nevýhody procedurální generace herních světů

Procedurální generace herních světů má mnoho výhod. Nabízí nekonečnou variabilitu a rozmanitost, což zaručuje, že hráči budou mít stále nové a unikátní zážitky. Tato technika také umožňuje snížit náklady na vývoj her, protože je možné vytvářet obsah pomocí opakovaného využití algoritmů a automatizace procesů. Díky procedurální generaci mohou herní světy rychle reagovat na hráčovy akce a vytvářet dynamické prostředí, které se přizpůsobuje hráčově hernímu stylu.

Nicméně, procedurální generace má také své nevýhody. Obtížné dosažení požadované kvality a ručního uměleckého dojmu může být výzvou. Některé procedurálně generované světy mohou působit stereotypně nebo nedostatečně autenticky, což může snížit celkový zážitek z hry. Zároveň vytváření složitých algoritmů pro procedurální generaci vyžaduje znalosti z oblasti matematiky, informatiky a designu, což může být náročné a časově náročné.

1.3. Použití procedurální generace v herním průmyslu

Procedurální generace herních světů je klíčovým prvkem v herním průmyslu, zejména při vytváření rozsáhlých otevřených světů, dungeonů a levelů. Tato technika nachází široké uplatnění v žánrech jako jsou hry s otevřeným světem, RPG a sandboxy. Je oblíbená díky schopnosti vytvářet dynamické a jedinečné prostředí pro každého hráče. Zejména v žánru ~~RogueLike~~Roguelike se procedurální generace prolíná s ostatními herními mechanikami, čímž vytváří unikátní zážitky a zajímavé výzvy pro hráče.

2. CÍLE PRÁCE

Cílem práce bylo naprogramovat vlastní hru inspirovanou tituly, jako je Noita či Rogue Legacy. Součástí hry je bojový systém, plynulý pohybový systém a vlastnoručně navrhnutá a vytvořená grafika. Hra by měla položit základy, ze kterých by bylo možno vytvořit plně funkční ~~RogueLike~~Roguelike hru, obsahující všechny charakteristiky typické pro tento žánr.

2.1. ~~RogueLike~~Roguelike

Roguelike je žánr RPG (herní žánr na hrdiny) charakteristický svou náročností, znovu-hratelností a důrazem na strategii. Většina roguelike her proto obsahuje aspoň část následujících prvků, které jsou pro tento žánr typické.

Pod tento žánr spadá název ~~RogueLite~~Roguelite, který označuje hry obsahující esenci těchto prvků, ale nezachovávající typickou povahu ~~RogueLike~~Roguelike her.

2.1.1. Permadeath (trvalá smrt):

Trvalá smrt se objevuje napříč celým žánrem, a je to jeden z nejdůležitějších. Pokud hráč v úrovni zemře, veškerý jeho postup je ztracen a začíná znovu. K tomuto znaku patří také obvykle vysoká obtížnost. Úplný průchod hry se napoprvé podaří málokomu, a každé vítezství je tak o to sladší.

2.1.2. Procedurálně generované úrovně:

Hlavním prvkem roguelike her je jejich znovu-hratelnost. Ta je nejčastěji docílena procedurální generací herních světů. Díky tomu je každý průchod hrou jiný a stává se pro hráče jedinečným zážitkem. Zároveň tím i jednodušší hra zabaví mnohonásobně déle.

2.1.3. Tahové hraní:

V některých hrách se akce hráče a nepřátel odehrávají v tazích, což umožňuje pečlivé zvažování každého pohybu a útoku a přidává novou vrstvu strategie do hry. Na hru tak mají větší dopad jednotlivé schopnosti a okolí hráče.

2.1.4. Gridové prostředí:

Svět v roguelike hrách je většinou znázorňován na mřížce čtverců, či jiných polí. Tato mřížka určuje jasnou strukturu a pomáhá při procedurální generaci. V tahových hrách se jí dostává ještě větší váhy, kdy většina veškerých interakcí se odehrává v jejím rozmezí.

3. ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

Program jsem vyvíjel současně jak z hlediska designu, tak programování. Většinou jsem se snažil nejdříve vytvořit grafiku pro plánované prvky, ale občas jsem vytvářel sprites až zpětně, to se mi stávalo zejména ke konci, kdy jsem implementoval nové věci na základě předloh jiných už hotových.

3.1. Program

3.1.1. Hráč

Vzhledem k tomu že se celá hra odvíjí od hráčské postavy, většina samotného kódu se nachází ve skriptech pod hráčem. To zahrnuje pohyb hráče a kamery, bojový systém a další funkce. Zaměřím se na ty, které mají největší dopad na hru a které jsou něčím zajímavé. Konkrétně na kameru se

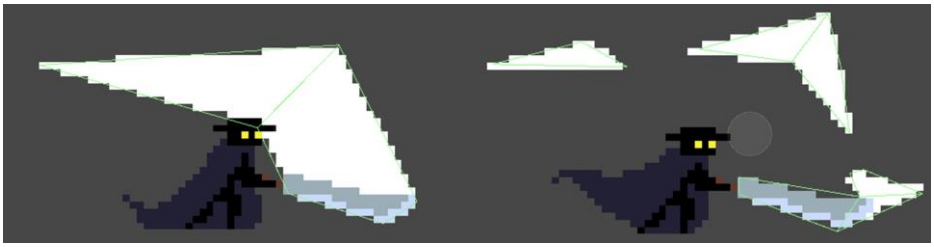
například zaměřovat nebudu. V případě zájmu by měli být všechny programy stručně popsané v komentářích.

3.1.1.1 Pohyb

Tento skript řídí pohyb hráče ve hře a také je jedním z prvních, který jsem napsal. Hráč se může pohybovat doleva a doprava pomocí klávesnice. Pokud stiskne klávesu Shift, hráč bude běžet rychleji (sprintovat). Mechanika skoku je také implementována, s hladkým a responsivním chováním. To je docíleno několika nadstavbami: Existuje "hang time", což je časový úsek, během kterého může hráč skočit i poté, co opustil pevnou podložku. Skok také vyžaduje menší přesnost od hráče. Po stisknutí klávesy skoku ve vzduchu je krátké období - "jump buffer", kdy se skok provede hned při dopadu na zem. Dále skript kontroluje, zda se hráč dotýká země a ovládá animátor v oblastech pohybu.

3.1.1.2 Bojový sytém

Tento skript řídí soubojové mechanismy hráče v herním prostředí. Umožňuje hráči útočit na nepřátele s určitou frekvencí, způsobuje jim poškození a může je odhodit zpět (knockback). Když hráč stiskne klávesu pro útok (s), spustí se v animátoru útočná animace. Až v samotné animaci probíhá funkce HitReg, která označí zasažené nepřátele a následně jim přidělí poškození. Tato funkce se volá celkově čtyřikrát, a to v každém snímku samotného útoku pomocí event markeru v animačním okně. Ke každému tomuto snímku je přiřazen speciální polygon collider, který přesně odpovídá tvaru útoku. Díky tomu útoky působí precisně a hráč přesně ví, kdy nepřítele zasáhne.



Obrázek 4. Ukázka dvou snímků z animace, které volají HitReg s jejich příslušnými collidery.

3.1.2. Nepřátelé

Nepřátele v této hře představují červení duchové, kteří lítají za hráčem a ubírají mu životy. Ačkoli jsem implementoval pouze jeden druh nepřátel, samotný prefab je rozdělený na grafiku a chování, takže by nemělo být těžké vytvořit další.

3.1.2.1 Umělá Inteligence nepřátel

Na umělou inteligenci nepřátel jsem využil algoritmu A-star, který na základě terénu na mapě vygeneruje mřížku, podle které se nepřátelé navigují. V knihovně A-star pathfinding project, kterou v projektu používám se nachází už předem vytvořené skripty, které většinu navigace vyřeší. Zároveň jsem napsal také skript, který by měl za pomoci A-staru plnit stejnou funkci, ale ve finální verzi jsem pouze upravil ty již hotové, které fungovaly výrazně lépe.

3.1.2.2 Ostatní mechaniky

Na prefabu se pak nachází pouze jeden další skript, který ovládá vše ostatní od animací, po život (na grafickém child objectu se nachází ještě jeden skript, který pouze plní funkci mazání objektu. Volá se z animátoru, proto bylo lehčí vytvořit speciální skript nacházející se zde.) Když nepřítel zemře, má šanci, že se z něj objeví nějaký z power upů, nebo předmět který hráče vyléčí.

Nepřátelé se do hry dostávají přes spawnery, které jsou rozmístěné po mapě. Ty pouze v určitém intervalu vytvářejí nepřátel. Pokud se hráč nachází v blízkosti, spawnery se vypnou. Čím déle je hráč naživu, tím se zvyšuje obtížnost hry – frekvencí a životy nepřátel.

3.1.3. Herní menu

Při zapnutí hry se zobrazí hlavní menu, kde hráč může hru zapnout, vypnout, nebo přejít do nastavení (zatím pouze vizuální nastínění). Pokud hráč umře, objeví se menu, které nabízí hráči hrát znovu, nebo jít zpět do hlavního. Zároveň si zde může hráč prohlédnout své statistiky, jak dlouho přežil, či jak byl na konci silný.

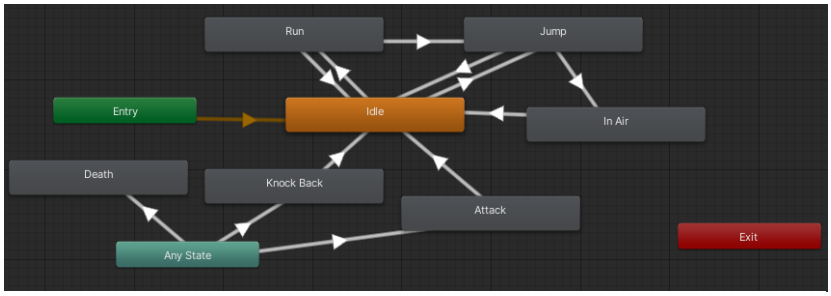
3.2. Grafika

Implementovat vlastní grafiku byl jeden z mých předních cílů. Grafický design je něco, čemu se občasně věnuji, a mám s ním již zkušenosti, i pomůcky které mi usnadnily práci (Grafický tablet One by Wacom S, notebook s dotykovým displejem…).

3.2.1. Animace

Jedním s časově nejnáročnějších procesů bylo vytváření animací. Animace jsou uloženy jako PNG soubory, kdy jeden soubor obsahuje celou animaci rozloženou na jednotlivé snímky – Spritesheet. Tento soubor je třeba v unity rozdělit zpět na jednotlivé snímky, které jsou pak užité v jednotlivých animacích. Jednotlivé animace ovládá poté animátor. V animátoru jsou mezi jednotlivými animacemi potřeba vytvořit přechody, které diktují, které animace má animátor přehrát. (Např. pokud stojí postava hráče na místě, program, který ovládá pohyb hráče, animátoru tuto informaci

sdělí a ten na základě toho přehrává idle animaci do té doby, než je mu vyslán jiný impuls, např. se dá hráč do pohybu).

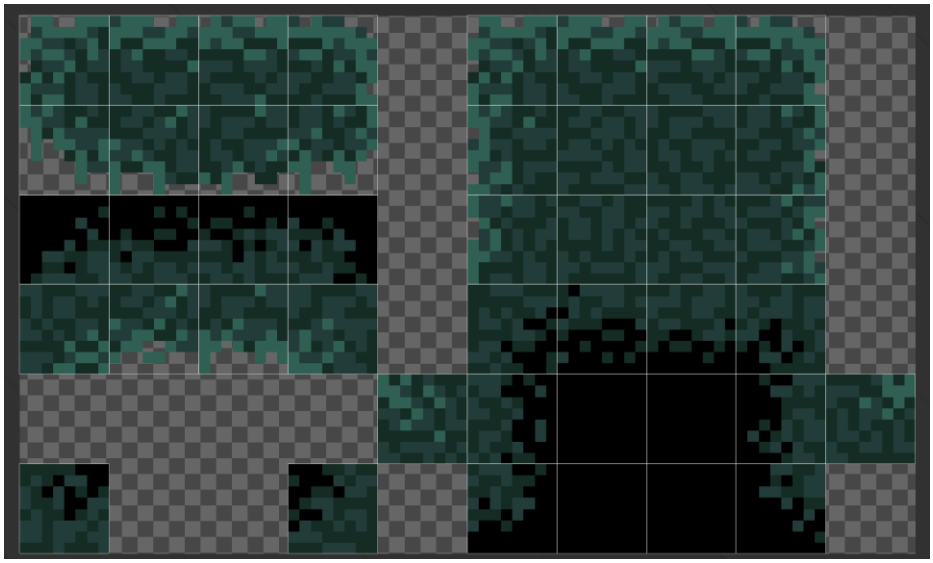


Obrázek 5 Logika V Animátoru Hráče

3.2.2. Tilemapy

Jiný typ spritesheetů jsou Tilemapy, což jsou opět PNG soubory, ale místo jednotlivých snímků jsou rozděleny na tiles – malé čtvercové designy, ze kterých se pak tvoří prostředí.

V unity je třeba vytvořit game object, který vytvoří pro tilemapy mřížku, do které se umisťují. Do toho se pak přidávají jednotlivé vrstvy. Na vrstvu pro zem jsou přidány herní komponenty, které automaticky rozpoznají přidané tiles do mřížky a přidají jim veškeré vlastnosti, které jsou třeba aby se chovaly jako funkční zem. Tento proces usnadní vytváření levelů do té míry, kdy vytvoření k vytvoření mapy je třeba pouze ji nakreslit do příslušné vrstvy.



Obrázek 6 Tilemapa země

4. PROGRAMOVACÍ PROSTŘEDKY

4.1. Programy využité ve hře

Celý program je v programovacím jazyku C#

4.1.1. Unity a Visual Studio

Celý projekt je dělán v herním enginu Unity (Verze 2021.3.15f1) a všechny individuální skripty ve Visual Studiu 2019.

4.1.2. A*star Pathfinding Project

K vytvoření umělé inteligence nepřátel ve hře jsem využil neplacené verze knihovny A*star Pathfinding Project, volně přístupné na jejich oficiálních stránkách.

4.2. Programy využité pro grafiku

4.2.1. Piskel

Protože jsem vybíral pouze z programů, které jsou přístupné zdarma, výběr jsem měl omezený. Původní vzhled hráče jsem vytvořil v programu Piskel, který sice obsahoval všechny funkce nezbytné k vytvoření grafiky, ovšem pro dlouhodobé využití byl velmi nepraktický. Hlavním problémem pro mě byla špatná práce ~~s~~se specifickými soubory .piskel a nepřehledná UI.

Obrázek 7 Ukázka programu Piskel spolu s první verzí Hráčské postavy

4.2.2. LibreSprite

Jednou z nejpoužívanějších aplikací pro vytváření PixelArtu je AsePrite, který je ale placený. Naštěstí je na internetu dostupná jeho původní verze, LibreSprite, zcela zdarma. I s touto neplacenou verzí se pracovalo mnohem lépe a zbytek projektu jsem dělal výhradně v ní. Rozhodl jsem se i aktualizovat hráčský model, se kterým jsem nebyl spokojen. Ve finální verzi hry je veškerá grafika dělána v tomto programu, kromě užitého fontu Caveat, volně dostupného v Unity Asset Store.

5. ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

Mým původním cílem bylo naprogramovat plnohodnotnou roguelike hru, srovnatelnou s hrami dělanými indie vývojáři. Neboli s vlastní grafikou, plnou hratelností a procedurální generací map. Vzhledem k tomu, že jde o můj první pořádný pokus o plnou hru v unity engine a vše jsem se učil v průběhu vytváření hry, byla má původní vize příliš ambiciózní. Omezil jsem se proto čistě na snahu vytvořit funkční prototyp, ze kterého je možné hru dále vyvíjet.

~~Za největší úspěch považuji grafiku, která dodává hře ucelený vzhled, a načrtává trajektorii pro další vývoj~~Za největší úspěch považuji grafiku, která nejen dodává hře ucelený vzhled, ale také poskytuje jasný směr a inspiraci pro další vývoj. Za další povedenou část považuji řešení problému útoků postavy.

Formatted: Pattern: Clear

Bohužel na implementaci procedurální generace je třeba výrazně pokročilejších znalostí matematiky a programování, které mi zatím schází. Kromě tohoto nedostatku jsem s projektem spokojený, i když dovedení této hry do prodejního stavu by vyžadovalo ještě další práci.

6. INSTALACE

Projekt lze stáhnout ze stránky GitHub na adrese: https://github.com/jirsamat/Capeman.git Na této stránce lze stáhnout zip soubor obsahující projekt. Po stažení souboru lze v aplikaci Unity Hub otevřít stažený soubor, Unity Hub poté projekt otevře. Projekt obsahuje dvě scény, hlavní scéna má název Menu. Ve scéně InGame hra probíhá.

Součástí staženého souboru je složka build, ve které je sestavená hra, která jde spustit jako aplikace.~~*JAK SE TO*~~

Hra je dělána pro systém Windows.

~~6.~~7. OVLÁDÁNÍ

Ovládání je velmi jednoduché a nijak se zásadně neliší od zajetých zvyků v herním světě.

V menu se hráč pohybuje myší a na jednotlivá tlačítka klikne levým kliknutím. Tlačítka pozná podle toho, že se černě zvýrazní, pokud na ně přejed myší.

V samotné hře se hráč pohybuje do stran klávesami A (doleva) a D (doprava). Skáče mezerníkem a útočí klávesou S.

~~7.~~8. SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] **Brackeys** [YouTube channel]. [Citováno 1. 4. 2024].

[2] **Unity Technologies**. Unity Documentation. [Online]. [Citováno 1. 4. 2024]. Available from: https://docs.unity3d.com/Manual/index.html

[3] **OpenAI**. OpenAI Chat. [Online]. [Citováno 1. 4. 2024]. Available from: https://openai.com/blog/chatgpt ~~(consider mentioning the specific page if it applies)~~

[4] **Wikipedia, The Free Encyclopedia**. Procedural generation. [Online]. Wikimedia Foundation, Inc., 2024. [Citováno 1. 4. 2024]. Available from: https://en.wikipedia.org/wiki/Procedural_generation

[5] **Wikipedia, The Free Encyclopedia**. Roguelike. [Online]. Wikimedia Foundation, Inc., 2024. [Citováno 1. 4. 2024]. Available from: https://en.wikipedia.org/wiki/Roguelike